Appendix

1. Code

1.1 C code

1.1.1 C code to implement the adaptation algorithm and generate a Markov Chain (Example 1~3)
1.1.2 C code to implement the adaptation algorithm and generate a Markov Chain (Example 4)

1.2 R code

1.2.1 R code to plot the final $g(X)$ and $\sigma(X)$ (Example 1~3)

1.2.2 R code to plot the final $g(X)$ and $\sigma(X)$ (Example 4)

1.2.3 R code to generate Markov Chain without adaption (Example 1)
1.2.4 R code to generate Markov Chain without adaption (Example 2)
1.2.5 R code to generate Markov Chain without adaption (Example 3)
1.2.6 R code to generate Markov Chain without adaption (Example 4)
1.2.7 R code to compute different efficiency measures (Example 1~3)
1.2.8 R code to compute different efficiency measures (Example 4)

1.2.9 R code to implement OLS in approximation of the final $\sigma(X)$

2. Output

2.1 Example 1: normal distribution in R^1

2.1.1 kernel function $K(x) = e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

2.1.1.1 case 1 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.1.2 case 2 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.1.3 case 3 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.1.4 case 4 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.1.5 case 5 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 0.5, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.1.6 case 6 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 2, \alpha_2 = 1, C = 2, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.1.7 case 7 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 2, \alpha_2 = 1, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.1.8 case 8 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.1.9 case 9 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 2, \alpha_2 = 1, C = 0.5, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.1.10 case 10 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 2, \alpha_2 = 1, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.1.11 case 11 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 10$ with fixed bandwidth $b_n = 1$

2.1.1.12 case 12 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.1.1.13 case 13 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 1$ with fixed bandwidth $b_n = 1$

2.1.1.14 case 14 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth $b_n = 1$

2.1.1.15 case 15 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.1.16 case 16 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 10$

2.1.1.17 case 17 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 0.1$

2.1.1.18 case 18 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.1.2 kernel function $K(x) = \frac{1}{1 + \alpha_1 \mid x \mid^{\alpha_2}}$

2.1.2.1 case 19 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.2.2 case 20 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.2.3 case 21 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.2.4 case 22 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.2.5 case 23 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.2.6 case 24 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.01, \gamma = 2$ with fixed bandwidth

$b_n = 1$

2.1.2.7 case 25 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 1$ with fixed bandwidth $b_n = 1$

2.1.2.8 case 26 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 10$ with fixed bandwidth $b_n = 1$

2.1.2.9 case 27 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 10$

2.1.2.10 case 28 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 0.1$

2.1.2.11 case 29 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth $b_n = 10$

2.1.2.12 case 30 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.1.2.13 case 31 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 10$

2.1.3 kernel function $K(x) = \begin{cases} 0, & |x| \geq 2 * width \\ -1 * hegith, & width < |x| < 2 * width \\ 1 * height, & |x| \leq width \end{cases}$

2.1.3.1 case 32 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.2 case 33 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.3 case 34 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.4 case 35 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.5 case 36 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.6 case 37 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.001, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.7 case 38 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.001, \gamma = 10$ with fixed bandwidth $b_n = 1$

2.1.3.8 case 39 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.001, \gamma = 1$ with fixed bandwidth $b_n = 1$

2.1.3.9 case 40 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.001, \gamma = 0.1$ with fixed bandwidth $b_n = 1$

2.1.3.10 case 41 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 1, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.11 case 42 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 10, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.12 case 43 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.1, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.13 case 44 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 2, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.14 case 45 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.15 case 46 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.1, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.3.16 case 47 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 10$

2.1.3.17 case 48 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 0.1$

2.1.3.18 case 49 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 2$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.1.3.19 case 50 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.4 kernel function $K(x)=\begin{cases}0,|x|\geq 2*width \quad or \quad |x|\leq width\\ 1, width<|x|<2*width\end{cases}$

2.1.4.1 case 51 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=0.5, C=1, \gamma=2$ with fixed bandwidth $b_n=1$

2.1.4.2 case 52 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=0.5, C=10, \gamma=2$ with fixed bandwidth $b_n=1$

2.1.4.3 case 53 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=0.5, C=0.1, \gamma=2$ with fixed bandwidth $b_n=1$

2.1.4.4 case 54 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=0.5, C=0.01, \gamma=2$ with fixed bandwidth $b_n=1$

2.1.4.5 case 55 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=0.5, C=0.001, \gamma=2$ with fixed bandwidth $b_n=1$

2.1.4.6 case 56 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=1, C=0.01, \gamma=2$ with fixed bandwidth $b_n=1$

2.1.4.7 case 57 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=5, C=0.01, \gamma=2$ with fixed bandwidth $b_n=1$

2.1.4.8 case 58 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=0.1, C=0.01, \gamma=2$ with fixed bandwidth $b_n=1$

2.1.4.9 case 59 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=8, C=0.01, \gamma=2$ with fixed bandwidth $b_n=1$

2.1.4.10 case 60 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=5, C=0.01, \gamma=5$ with fixed bandwidth $b_n=1$

2.1.4.11 case 61 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=5, C=0.01, \gamma=1$ with fixed bandwidth $b_n=1$

2.1.4.12 case 62 $\eta_n=\frac{1}{(n+5)^{0.5}},\ \text{width}=5, C=0.01, \gamma=0.1$ with fixed bandwidth

$b_n = 1$

2.1.4.13 case 63 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth

$b_n = 10$

2.1.4.14 case 64 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth

$b_n = 0.1$

2.1.4.15 case 65 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with decreasing bandwidth

$b_n = \frac{1}{n^{0.2}}$

2.1.4.16 case 66 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.1.5 constant $\sigma(x) = C$

2.1.6 varfact comparison

2.1.7 variance comparison

2.1.8 comparison of average squared jump distance

2.1.9 OLS in approximation of the final $\sigma(x)$

2.2 Example 2: mixture of two normal distributions in R^1

2.2.1 kernel function $K(x) = e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

2.2.1.1 case 67 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.2.1.2 case 68 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.2.1.3 case 69 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.2.1.4 case 70 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

$b_n = 1$

2.2.1.5 case 71 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.5, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

$b_n = 1$

2.2.1.6 case 72 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.01, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

$b_n = 1$

2.2.1.7 case 73 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 10, \gamma = 2$ with fixed bandwidth

$b_n = 1$

2.2.1.8 case 74 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 1$

2.2.1.9 case 75 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 10$ with fixed bandwidth

$b_n = 1$

2.2.1.10 case 76 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 1$ with fixed bandwidth

$b_n = 1$

2.2.1.11 case 77 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth

$b_n = 1$

2.2.1.12 case 78 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 10$

2.2.1.13 case 79 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 0.1$

2.2.1.14 case 80 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad \alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.2.1.15 case 81 $\eta_n = \frac{1}{(n+5)^{0.8}}, \quad \alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 10$

2.2.2 kernel function $K(x) = \frac{1}{1 + \alpha_1 |x|^{\alpha_2}}$

2.2.2.1 case 90 $\eta_n = \frac{1}{(n+5)^{0.5}}, \ \alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.2.2.2 case 91 $\eta_n = \frac{1}{(n+5)^{0.5}}, \ \alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.2.2.3 case 92 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad \alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.2.2.4 case 93 $\eta_n = \frac{1}{(n+5)^{0.5}}, \ \alpha_1 = 12, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.2.2.5 case 94 $\eta_n = \frac{1}{(n+5)^{0.5}}, \ \alpha_1 = 5, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.2.2.6 case 95 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad \alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 10$ with fixed bandwidth $b_n = 1$

2.2.2.7 case 96 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad \alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 0.1$ with fixed bandwidth $b_n = 1$

2.2.2.8 case 97 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad \alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.1$ with fixed bandwidth

$b_n = 1$

2.2.2.9 case 98 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth

$b_n = 1$

2.2.2.10 case 99 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth

$b_n = 10$

2.2.2.11 case 100 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth

$b_n = 0.1$

2.2.2.12 case 101 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.2.2.13 case 102 $\eta_n = \frac{1}{(n+5)^{0.8}},\ \alpha_1 = 10, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth

$b_n = 1$

2.2.3 kernel function $K(x) = \begin{cases} 0, & |x| \geq 2 * width \\ -1 * hegith, & width < |x| < 2 * width \\ 1 * height, & |x| \leq width \end{cases}$

2.2.3.1 case 103 $\eta_n = \frac{1}{(n+5)^{0.5}},\ C = 1, \gamma = 2, \text{height} = 0.5, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

2.2.3.2 case 104 $\eta_n = \frac{1}{(n+5)^{0.5}},\ C = 10, \gamma = 2, \text{height} = 0.5, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

2.2.3.3 case 105 $\eta_n = \frac{1}{(n+5)^{0.5}},\ C = 0.1, \gamma = 2, \text{height} = 0.5, \text{width} = 0.5,$ with fixed

bandwidth $b_n = 1$

2.2.3.4 case 106 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 10, \text{height} = 0.5, \text{width} = 0.5,$ with fixed

bandwidth $b_n = 1$

2.2.3.5 case 107 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 0.5, \text{width} = 0.5,$ with fixed

bandwidth $b_n = 1$

2.2.3.6 case 108 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 2, \text{height} = 1, \text{width} = 0.5,$ with fixed

bandwidth $b_n = 1$

2.2.3.7 case 109 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 2, \text{height} = 5, \text{width} = 0.5,$ with fixed

bandwidth $b_n = 1$

2.2.3.8 case 110 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 2, \text{height} = 0.1, \text{width} = 0.5,$ with fixed

bandwidth $b_n = 1$

2.2.3.9 case 111 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 1, \gamma = 0.1, \text{height} = 1, \text{width} = 1,$ with fixed

bandwidth $b_n = 1$

2.2.3.10 case 112 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 5,$ with fixed

bandwidth $b_n = 1$

2.2.3.11 case 113 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with fixed

bandwidth $b_n = 1$

2.2.3.12 case 114 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with fixed

bandwidth $b_n = 10$

2.2.3.13 case 115 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with fixed bandwidth $b_n = 0.1$

2.2.3.14 case 116 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.2.3.15 case 117 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.2.4 kernel function $K(x) = \begin{cases} 0, & |x| \geq 2 * width \quad or \quad |x| \leq width \\ 1, & width < |x| < 2 * width \end{cases}$

2.2.4.1 case 118 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 1, \gamma = 2, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

2.2.4.2 case 119 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 2, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

2.2.4.3 case 120 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 10, \gamma = 2, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

2.2.4.4 case 121 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 10, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

2.2.4.5 case 122 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

2.2.4.6 case 123 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad C = 0.1, \gamma = 0.1, \text{width} = 1,$ with fixed bandwidth $b_n = 1$

2.2.4.7 case 124 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad C = 0.1, \gamma = 0.1, \text{width} = 2,$ with fixed bandwidth $b_n = 1$

2.2.4.8 case 125 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad C = 0.1, \gamma = 0.1, \text{width} = 5,$ with fixed bandwidth $b_n = 1$

2.2.4.9 case 126 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad C = 0.1, \gamma = 0.1, \text{width} = 0.2,$ with fixed bandwidth $b_n = 1$

2.2.4.10 case 127 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad C = 0.1, \gamma = 0.1, \text{width} = 0.2,$ with fixed bandwidth $b_n = 10$

2.2.4.11 case 128 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad C = 0.1, \gamma = 0.1, \text{width} = 0.2,$ with fixed bandwidth $b_n = 0.1$

2.2.4.12 case 127 $\eta_n = \frac{1}{(n+5)^{0.5}}, \quad C = 0.1, \gamma = 0.1, \text{width} = 0.2,$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.2.4.13 case 128 $\eta_n = \frac{1}{(n+5)^{0.8}}, \quad C = 0.1, \gamma = 0.1, \text{width} = 0.2,$ with fixed bandwidth $b_n = 1$

2.2.5 constant $\sigma(x) = C$

2.2.6 varfact comparison

2.2.7 variance comparison

2.2.8 comparison of average squared jump distance

2.3 Example 3: mixture of three normal distributions in R^1

2.3.1 kernel function $K(x) = e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

2.3.1.1 case 129 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.1.2 case 130 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.1.3 case 131 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.1.4 case 132 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.1.5 case 131 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.1.6 case 132 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.1.7 case 133 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.1.8 case 134 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth

$b_n = 1$

2.3.1.9 case 135 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 0.1, \gamma = 0.5$ with fixed bandwidth

$b_n = 1$

2.3.1.10 case 136 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 10$

2.3.1.11 case 137 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 0.1$

2.3.1.12 case 138 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 0.5$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.3.1.13 case 139 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth

$b_n = 1$

2.3.2 kernel function $K(x) = \frac{1}{1 + \alpha_1 |x|^{\alpha_2}}$

2.3.2.1 case 140 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.2.2 case 141 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

$b_n = 1$

2.3.2.3 case 142 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

$b_n = 1$

2.3.2.4 case 143 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.2.5 case 144 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 1, \alpha_2 = 1, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.2.6 case 145 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.2.7 case 146 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.2.8 case 147 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.2.9 case 148 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.2.10 case 149 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.2.11 case 150 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.1$ with fixed bandwidth $b_n = 1$

2.3.2.12 case 151 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 1$ with fixed bandwidth $b_n = 1$

2.3.2.13 case 152 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 10$

2.3.2.14 case 153 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 0.1$

2.3.2.15 case 154 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ withdecreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.3.2.16 case 155 $\eta_n = \frac{1}{(n+5)^{0.8}},\ \alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ fixed bandwidth $b_n = 1$

2.3.3 kernel function $K(x) = \begin{cases} 0, & |x| \geq 2 * width \\ -1 * hegith, & width < |x| < 2 * width \\ 1 * height, & |x| \leq width \end{cases}$

2.3.3.1 case 156 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{height} = 0.5, \text{width} = 0.5, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.3.2 case 157 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{height} = 0.5, \text{width} = 0.5, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.3.3 case 158 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{height} = 0.5, \text{width} = 0.5, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.3.4 case 159 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{height} = 2, \text{width} = 0.5, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.3.5 case 160 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.5, \text{width} = 2, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.3.6 case 161 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.5, \text{width} = 0.5, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.3.7 case 162 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.5, \text{width} = 0.5, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.3.8 case 163 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.5, \text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.3.9 case 164 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 2, \text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.3.10 case 165 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 5, \text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.3.11 case 166 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 10, \text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.3.12 case 167 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.3.13 case 168 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.3.14 case 169 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 8, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.3.15 case 170 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 15, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.3.16 case 171 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 0.1, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.3.17 case 172 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.1$ with fixed bandwidth $b_n = 1$

2.3.3.18 case 173 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 10$

2.3.3.19 case 174 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 0.1$

2.3.3.20 case 175 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.3.3.21 case 176 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.4 kernel function $K(x) = \begin{cases} 0, & |x| \geq 2 * width \quad or \quad |x| \leq width \\ 1, & width < |x| < 2 * width \end{cases}$

2.3.4.1 case 177 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 0.5, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.4.2 case 178 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 0.5, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.4.3 case 179 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 0.5, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.4.4 case 180 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 2, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.4.5 case 181 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 0.1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.3.4.6 case 182 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 0.5, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.4.7 case 183 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 0.5, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.4.8 case 184 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.4.9 case 185 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 2, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.4.10 case 186 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 8, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.4.11 case 187 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 15, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.4.12 case 189 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.4.13 case 190 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 10$

2.3.4.14 case 191 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 0.1$

2.3.4.15 case 192 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 1, \gamma = 0.5$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

2.3.4.16 case 193 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\text{width} = 0.5, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

2.3.5 constant $\sigma(x) = C$

2.3.6 varfact comparison

2.3.7 variance comparison

2.3.8 comparison of average squared jump distance

2.4 Example 4: normal distribution in R^2

2.4.1 kernel function $K(x) = e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

2.4.1.1 case 194 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.4.1.2 case 195 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.5, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.4.1.3 case 196 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 0.5, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.4.1.4 case 197 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.4.2 kernel function $K(x) = \frac{1}{1 + \alpha_1 |x|^{\alpha_2}}$

2.4.2.1 case 198 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.4.2.2 case 199 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 1$ with fixed bandwidth $b_n = 1$

2.4.2.3 case 200 $\eta_n = \frac{1}{(n+5)^{0.8}},\ \alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 10$

2.4.3 kernel function $K(x) = \begin{cases} 0, & |x| \geq 2 * width \\ -1 * hegith, & width < |x| < 2 * width \\ 1 * height, & |x| \leq width \end{cases}$

2.4.3.1 case 201 $\eta_n = \frac{1}{(n+5)^{0.5}},\ height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 10$ with fixed bandwidth $b_n = 1$

2.4.4 kernel function $K(x) = \begin{cases} 0, & |x| \geq 2 * width \quad or \quad |x| \leq width \\ 1, & width < |x| < 2 * width \end{cases}$

2.4.4.1 case 202 $\eta_n = \frac{1}{(n+5)^{0.5}},\ \text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

2.4.5 constant $\sigma(x) = C$

2.4.6 varfact comparison

2.4.7 variance comparison
2.4.8 comparison of average squared jump distance

Appendix

1. Code

1.1 C code

1.1.1 C code to implement the adaptation algorithm and generate a Markov Chain (Example 1~3)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <sys/time.h>

#define RUNLENGTH 10000
#define PRINTSP 100
#define PI 3.1415926536
#define N 10000
#define    M 100000
#define B 10000
#define numsig 16000
#define left -30
#define right 30
#define gap 0.01

#define m0 0x100000000LL
#define c0 0xB16
#define a0 0x5DEECE66DLL

double alpha1,alpha2,bn, beta, C, ggamma;
double height, width;
double x[RUNLENGTH],xlist[M];
int accept[RUNLENGTH],addsign[RUNLENGTH];
double appg[numsig],appsig[numsig];

int main()
{
      int i, n, numacc;
      double X, Y, U;
      FILE *fpg, *fpx, *fpsig,*fpout,*fpxlist;
      clock_t start, end;
      double time1,time2,time3,time4;

      double normal();
```

```
double logpi(double);
double g(int,double);
double K(double);
double sigma(int,double),appsigma(double);
double logaccprob(int,double,double),applogaccprob(double, double);
void seedrand();
double mean(double xx[],int nn);
double a(double);
int divides(int,int);
double eta(int);
double sq(double);
int indicator(int);
double mean(double xx[],int nn);
double a(double);

seedrand();

/* Initial values. */
alpha1 = 10.0;
alpha2 = 1.0;
bn = 1.0;/*bandwidth*/
beta = 0.0;/*the initial value for beta*/
C = 1.0;/*one parameter in sigma*/
ggamma = 2.0;/*one parameter in sigma*/
height = 0.5;/*one parameter in the kernel function*/
width = 0.5;/*one parameter in the kernel function*/
X = 0.0;
numacc = 0;

printf("\nHere we go!\n\n");

/* Do the run. */
start = clock();
for (n=0; n<RUNLENGTH; n++) {
   Y = X + sigma(n,X) * normal();
   U = drand48();
   if ( log(U) < logaccprob(n,X,Y) ) {
      X = Y;
      accept[n] = 1;
      numacc++;
      beta = beta + 1.0/pow((n+5),0.5)*(1.0-0.234);
   }
   else {
      accept[n] = 0;
```

```
            beta = beta - 1.0/pow((n+5),0.5)*0.234;
          }
          if ( log(0.234) < logaccprob(n,X,Y) )
            addsign[n] = 1 ;
          else
            addsign[n] = -1 ;
          x[n] = X;
          /*bn = 1.0/pow(n,0.2);*/
          if (divides(PRINTSP,n))
          printf("n=%d,   X=%f,   Y=%f, K=%f,G=%f,sigma=%f, beta=%f,   accept=%d\n",n,
X, Y, K(X),g(n,X),sigma(n,X),beta, accept[n]);
        }
        end = clock();
        time1 = (double) (end - start)/CLOCKS_PER_SEC;
        printf("Accepted %d out of %d (%.3f%%).\n", numacc, RUNLENGTH,
100.0*numacc/RUNLENGTH);
        printf("final beta = %f\n\n", beta);

        /*calculate g and sigma for some X*/
        start = clock();
        for(i=0;i<numsig;i++){
        X = left + i * gap;
        appg[i] = g(RUNLENGTH-1,X);
        appsig[i] = sigma(RUNLENGTH-1,X);
        }

        /*Plot the resulting g values to a file ... */
        printf("Writing gvals ...\n");
        if ((fpg = fopen("gvals","w")) == NULL) {
        fprintf(stderr, "Unable to open output file 'gvals'.\n");
        exit(1);
        }
        fprintf(fpg, "gvals = c(");
        for (i=0; i<numsig; i++)
          fprintf(fpg, "%f,\n", appg[i]);
        fprintf(fpg, "%f)\n\n", appg[numsig-1]);
        fclose(fpg);

        /*Plot the resulting sigma values to a file ... */
        printf("Writing sigvals ...\n");
        if ((fpsig = fopen("sigvals","w")) == NULL) {
        fprintf(stderr, "Unable to open output file 'sigvals'.\n");
        exit(1);
        }
```

```
    fprintf(fpsig, "sigvals = c(");
    for (i=0; i<numsig; i++) {
       fprintf(fpsig, "%f,\n", appsig[i]);
    }
    fprintf(fpsig, "%f)\n\n", appsig[numsig-1]);
    end = clock();
    time2 = (double) (end - start)/CLOCKS_PER_SEC;
    fclose(fpsig);

    /*Plot the resulting X values to a file ... */
    printf("Writing xvals ...\n");
    if ((fpx = fopen("xvals","w")) == NULL) {
    fprintf(stderr, "Unable to open output file 'xvals'.\n");
    exit(1);
    }
    fprintf(fpx, "xvals = c(");
    for (i=0; i<RUNLENGTH-1; i++)
       fprintf(fpx, "%f,\n", x[i]);
    fprintf(fpx, "%f)\n\n", x[RUNLENGTH-1]);
    fprintf(fpx, "hist(xvals)");
    fclose(fpx);

    /*generate a Markov chain with scale proposed by the algorithm*/
    start = clock();
    printf("generating Markov Chain ...\n");
    xlist[0] = drand48()*15;
    numacc = 0;
    for(i=1;i<M;i++){
       X = xlist[(i-1)];
       Y = X + sigma(RUNLENGTH-1,X) * normal();
       U = drand48();
       if ( log(U) < logaccprob(RUNLENGTH-1,X,Y) ) {
          X = Y;
          numacc++;
       }
       xlist[i] = X;
    }
    printf("run the M-H algorithm for %d times, and acceptance rate is (%.3f%%).\n",
M,100.0*numacc/M);

   /*save the resulting xlist values to a file*/
    printf("Writing xlist vals ...\n");
    if ((fpxlist1 = fopen("xlist1","w")) == NULL) {
    fprintf(stderr, "Unable to open output file 'xlist1'.\n");
```

```
exit(1);
}
fprintf(fpxlist1,"xlist=c(");
for (i=0; i<(M-1); i++) {
   fprintf(fpxlist1,  "%f,\n", xlist[i]);
}
fprintf(fpxlist1,  "%f)\n\n", xlist[M-1]);
fclose(fpxlist1);
end = clock();
time3 = (double) (end - start)/CLOCKS_PER_SEC;

/*test alpha(X) and save them to a file ...  */
start = clock();
printf("Writing final vals ...\n");
if ((fpout = fopen("out","w")) == NULL) {
fprintf(stderr, "Unable to open output file 'out'.\n");
exit(1);
}
fprintf(fpout,"Accepted %d out of %d (%.3f%%).\n", numacc, M ,100.0*numacc/M);
fprintf(fpout,"final beta = %f\n", beta);
/*for normal distribution in one dimension*/
X = 0;
fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
X = 2;
fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
X = 5;
fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
X = 10;
fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
X = 20;
fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
X = 50;
fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
X = 100;
fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
X = 150;
fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
/*for the mixture of two normal distributions in one dimension*/
/*X = 5;
fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
X = 7;
fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
X = 10;
fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
```

```
        X = 20;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 30;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 50;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 70;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));*/
        /*for the mixture of three normal distributions in one dimension*/
        /*X = 0;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 2;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 5;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 8;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 10;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 13;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 15;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 20;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 30;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));
        X = 50;
        fprintf(fpout,"alpha(%f) is %f\n",X,a(X));*/
        end = clock();
        Time4 = (double) (end - start)/CLOCKS_PER_SEC;
        fprintf(fpout,"it takes %f seconds to run the adaptation algorithm\n", time1);
        fprintf(fpout,"it takes %f seconds to compute g and sigma\n", time2);
        fprintf(fpout,"it takes %f seconds to generate the Markov Chain\n", time3);
        fprintf(fpout,"it takes %f seconds to test the local acceptance\n", time4);
        fclose(fpout);

        printf("Done.\n");
        return(0);
}

/* normal:    return a standard normal random number.  */
double normal()
{
```

```
        double R, theta, drand48();
        R = - log(drand48());
        theta = 2 * PI * drand48();
        return( sqrt(2*R) * cos(theta));
}

/* seedrand: seed random number generator. */
void seedrand()
{
        int seed;
        struct timeval tmptv;
        gettimeofday (&tmptv, (struct timezone *)NULL);
        seed = (int) tmptv.tv_usec;
        srand48(seed);
        (void)drand48();    /* Spin it once. */
}

/*divides: whether aa divides bb. If yes, return 1; otherwise return 0.*/
int divides(int aa, int bb)
{
        return( (bb/aa)*aa == bb );
}

/*logpi: compute logarithm of target distribution*/
/*for normal distribution in one dimension*/
double logpi(double xx)
{
        double sq();
        return( -sq(xx)/2.0 );
}
/*for the mixture of two normal distributions in one dimension*/
/*double logpi(double xx)
{
        double sq();
        if(xx<0.0) return (-sq(xx)/2.0);
        if(xx>10.0) return (-sq(xx-10.0)/2.0);
        return( log( exp(-sq(xx)/2.0) + exp(-sq(xx-10.0)/2.0) ) );
}*/
/*for the mixture of three normal distributions in one dimension*/
/*double logpi(double xx)
{
        double sq();
        if(xx<-12) return (log(0.3)-sq(xx+10.0)/2.0);
        if(xx>12) return (log(0.3)-sq(xx-10.0)/2.0);
```

```
    return(log(0.3*exp(-sq(xx+10.0)/2.0)+0.4*exp(-sq(xx)/2.0)+0.3*exp(-sq(xx-10.0)/2.0)));
}*/

/*g: function g*/
double g(int nn, double xx)
{
      int ii;
      double   eta(),K(), val,abdif;
      val = 0.0;
      for (ii=0; ii<nn; ii++) {
         abdif = fabs(xx-x[ii]);
         val = (1.0 - eta(ii)) *   val + eta(ii) * addsign[ii] / bn * K(abdif/bn);
      }
      return(val);
}

/*K: kernel function*/
double K(double xx)
{    double abdif = fabs(xx);
      return
      ( exp(-pow(abdif,alpha1)/alpha2) );
      /*you can try other kernel function like: */
      /*( 1.0 / (1.0 + alpha1*pow( abdif, alpha2 ))); */
      /*height*( 2*indicator(abdif<width) -indicator(abdif<2*width) ); */
      /* indicator((abdif>=width)&&(abdif<2*width))); */
}

/*sigma: function sigma*/
double sigma(int nn, double xx)
{
      double g();
      /*for normal distribution in 1 dimension*/
      return( exp(beta) * pow(1.0 + C*fabs(xx), ggamma) * exp( g(nn,xx) ) );
      /*for the mixture of two normal distributions in 1 dimension*/
      /*return( exp(beta) * pow(1.0 + C*fabs(xx-5.0), ggamma) * exp( g(nn,xx) ) );*/
      /*for the mixture of three normal distributions in 1 dimension*/
      /*return(exp(beta)*pow(1.0+C*fabs(xx-5.0)*fabs(xx+5.0),ggamma)*exp( g(nn,xx) ) );*/
}

/*logaccprob: logarithm of acceptance probability from xx to yy*/
double logaccprob(int nn, double xx, double yy)
{
   double sigma(), sq();
```

```
return(logpi(yy)-logpi(xx)+log(sigma(nn,xx))-log(sigma(nn,yy))+sq(xx-yy)*(0.5/sq(sigma(n
n,xx))-0.5/sq(sigma(nn,yy))));
}

/*eta: function eta*/
double eta(int nn)
{
      return(1.0/pow((nn+5),0.5) );
      /*return(1.0/pow(nn+5,0.8) );*/
}

/*sq: compute the square of xx*/
double sq(double xx)
{
   return(xx*xx);
}

/*indicator: return 1/0 if the xx is non-zero/zero*/
int indicator(int xx)
{
   if (xx==0)
      return(0);
   else
      return(1);
}

/*a: compute the local acceptance rate of X*/
double a(double X)
{
      double appsigma(),applogaccprob(),mean();
      int ii;
      double Y, U, tmpval = 0.0;
      double alphaproposal[N];
      for(ii=0;ii<N;ii++)
      {
            Y = X + appsigma(X) * normal();
            U = drand48();
            if(log(U) < applogaccprob(X,Y))
                  tmpval = tmpval +1;
      }
      return (tmpval/N);
}

/*appsigma: approximation of sigma*/
```

```
double appsigma(double xx)
{     double sigma();
      int ii;
      if((xx<left)||(xx>=right)) return (sigma(RUNLENGTH-1,xx));
      else{
            for(ii=0;ii<numsig;ii++) {
                  if((xx>=(left+ii*gap))&&(xx<(left+(ii+1)*gap)))
                        return (appsig[ii]);
                }
      }
}

/*applogaccprob: approximation of logaccprob*/
double applogaccprob(double xx,double yy)
{     double appsigma(), sq();

      return(  logpi(yy)-logpi(xx)+log(appsigma(xx))-log(appsigma(yy))+sq(xx-yy)*(0.5/sq(ap
psigma(xx))-0.5/sq(appsigma(yy))) );
}
```

If we use approximation II, change "*sigma(RUNLENGTH-1,X)*" *to "appsigma(X)" and "logaccprob(RUNLENGTH-1,X,Y)"* to "*applogaccprob(X,Y)*" in the part to generate Markov Chain and the function a(double);

1.1.2 C code to implement the adaptation algorithm and generate a Markov Chain (Example 4)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <sys/time.h>

#define RUNLENGTH 10000
#define PRINTSP 100
#define PI 3.1415926536
#define N 10000
#define M 100000
#define B 10000
#define numsigma 100
#define left -5
#define right 5
#define bottom -5
#define top 5
#define gap 0.1
```

```
#define m0 0x100000000LL
#define c0 0xB16
#define a0 0x5DEECE66DLL

double alpha1, alpha2, bn, beta, C, ggamma;
double height, width;
double x[2][RUNLENGTH],xlist[2][M];
int accept[RUNLENGTH];
int addsign[RUNLENGTH];
double appg[numsigma][numsigma],appsig[numsigma][numsigma];

int main()
{
    int i, j, n, numacc;
    double X[2], Y[2], U;
    FILE *fpg, *fpx, *fpsig,*fpout,*fpxlist;
    int graphsp, graphleft, graphright,graphtop,graphbottom;
    clock_t start, end;
    double time0,time1,time2,time3,time4;

    double normal();
    double logpi(double,double);
    double g(int,double,double), K(double);
    double sigma(int,double,double), logaccprob(int,double,double,double,double);
    void seedrand();
    double a(double[2]);
    double appsigma(double,double), applogaccprob(double,double,double,double);

    seedrand();
    start = clock();

    /* Initial values. */
    alpha1 = 1.0;
    alpha2 = 1.0;
    bn = 1.0;
    beta = 0.0;
    C = 0.01;/*one parameter in the second term of sigma*/
    ggamma = 10.0;/*one parameter in the second term of sigma*/
    height = 0.5;
    width = 0.5;
    X[0] = 0.0;
    X[1] = 0.0;
    numacc = 0;
    graphsp = 15;
```

```
    graphleft = graphbottom = -30;
    graphright = graphtop = 30;

    printf("\nHere we go!\n\n");

    /* Do the run. */
    for (n=0; n<RUNLENGTH; n++) {
       Y[0] = X[0] + sigma(n,X[0],X[1]) * normal();
       Y[1] = X[1] + sigma(n,X[0],X[1]) * normal();
       U = drand48();
       if ( log(U) < logaccprob(n,X[0],X[1],Y[0],Y[1]) ) {
          X[0] = Y[0];
          X[1] = Y[1];
          accept[n] = 1;
          numacc++;
          beta = beta + 1.0/pow((n+5),0.5)*(1.0-0.234);
       }
       else {
          accept[n] = 0;
          beta = beta - 1.0/pow((n+5),0.5)*0.234;
       }
       if ( log(0.234) < logaccprob(n,X[0],X[1],Y[0],Y[1]) )
          addsign[n] = 1    /* *(1-0.234)*/;
       else
          addsign[n] = -1 /* * 0.234*/;
       x[0][n] = X[0];
       x[1][n] = X[1];
       if (divides(PRINTSP,n)) {
          printf("n=%d,    X=(%f,%f),    Y=(%f,%f),    beta=%f,    accept=%d\n", n, X[0],X[1],
Y[0],Y[1], beta, accept[n]);
       }
    /*bn = 1/pow((n+1),0.2); *//*fixed or decreasing bandwidth*/
    }
    printf("Accepted    %d    out    of    %d    (%.3f%%).\n",    numacc,
RUNLENGTH,100.0*numacc/RUNLENGTH);
    printf("final beta = %f\n\n", beta);
    end = clock();
    time0 = (double) (end - start)/CLOCKS_PER_SEC;

    start = clock();
    printf("computing g and sigma values.... ");
    for(i=0;i<numsigma;i++){
        for(j=0;j<numsigma;j++){
            X[0] = left + i*gap;
```

```
            X[1] =    bottom + j*gap;
            appg[i][j] = g((RUNLENGTH-1),X[0],X[1]);
            appsig[i][j] = sigma((RUNLENGTH-1),X[0],X[1]);
      }
}
end = clock();
time1 = (double) (end - start)/CLOCKS_PER_SEC;

/*Plot the resulting g values to a file ...    */
printf("Writing gvals ...\n");
if ((fpg = fopen("gvals","w")) == NULL) {
fprintf(stderr, "Unable to open output file 'gvals'.\n");
exit(1);
}
fprintf(fpg,  "gvals = c(");
for (i=graphleft*graphsp; i<(graphright*graphsp-1); i++){
      for(j=graphbottom*graphsp;j<(graphtop*graphsp-1);j++){
            fprintf(fpg,  "%f,\n", g(n,1.0*i/graphsp,1.0*j/graphsp));
            }
      fprintf(fpg,"%f,\n",g(n,1.0*i/graphsp,1.0*(j+1)/graphsp));
}
for(j=graphbottom*graphsp;j<(graphtop*graphsp-1);j++){
            fprintf(fpg,  "%f,\n", g(n,1.0*(i+1)/graphsp,1.0*j/graphsp));
            }
fprintf(fpg,"%f)\n",g(n,1.0*i/graphsp,1.0*(j+1)/graphsp));
fclose(fpg);

/*Plot the resulting sigma values to a file ...  */
printf("Writing sigvals ...\n");
if ((fpsig = fopen("sigvals","w")) == NULL) {
fprintf(stderr, "Unable to open output file 'sigvals'.\n");
exit(1);
}
fprintf(fpsig,  "sigvals = c(");
for (i=graphleft*graphsp; i<(graphright*graphsp-1); i++){
      for(j=graphbottom*graphsp;j<(graphtop*graphsp-1);j++){
            fprintf(fpsig,  "%f,\n", sigma(n,1.0*i/graphsp,1.0*j/graphsp));
            }
      fprintf(fpsig,"%f,\n",sigma(n,1.0*i/graphsp,1.0*(j+1)/graphsp));
}
for(j=graphbottom*graphsp;j<(graphtop*graphsp-1);j++){
            fprintf(fpsig,  "%f,\n", sigma(n,1.0*(i+1)/graphsp,1.0*j/graphsp));
            }
fprintf(fpsig,"%f)\n",sigma(n,1.0*i/graphsp,1.0*(j+1)/graphsp));
```

```
    fclose(fpsig);
    end = clock();
    time2 = (double) (end - start)/CLOCKS_PER_SEC;

    /*Plot the resulting X values to a file ... */
    printf("Writing xvals ...\n");
    if ((fpx = fopen("xvals","w")) == NULL) {
    fprintf(stderr, "Unable to open output file 'xvals'.\n");
    exit(1);
    }
    fprintf(fpx, "xvals1 = c(");
    for (i=0; i<n-1; i++)
       fprintf(fpx, "%f,\n", x[0][i]);
    fprintf(fpx, "%f)\n\n", x[0][n-1]);
    fprintf(fpx, "xvals2 = c(");
    for (i=0; i<n-1; i++)
       fprintf(fpx, "%f,\n", x[1][i]);
    fprintf(fpx, "%f)\n\n", x[1][n-1]);
    fprintf(fpx, "plot(xvals1,xvals2)");
    fclose(fpx);

   /*generate a Markov chain with scale proposed by the algorithm */
    start = clock();
    printf("generating Markov Chain ...\n");
    xlist[0][0] = drand48()*20;
    xlist[1][0] = drand48()*20;/*starting point*/
    numacc = 0;
    for(i=1;i<M;i++){
       Y[0] = xlist[0][(i-1)] + appsigma(xlist[0][i-1],xlist[1][i-1]) * normal();
       Y[1] = xlist[1][(i-1)] + appsigma(xlist[0][i-1],xlist[1][i-1]) * normal();
       U = drand48();
       if ( log(U) < applogaccprob(xlist[0][(i-1)],xlist[1][(i-1)],Y[0],Y[1]) ) {
          xlist[0][i] = Y[0];
          xlist[1][i] = Y[1];
          numacc++;
       }
       else {
          xlist[0][i] = xlist[0][i-1];
          xlist[1][i] = xlist[1][i-1];
       }
    }
    printf("run the M-H algorithm for %d times, and acceptance rate is (%.3f%%).\n",
M,100.0*numacc/M);
```

```
/*save the resulting xlist values to a file */
 printf("Writing xlist vals ...\n");
 if ((fpxlist = fopen("xlist","w")) == NULL) {
 fprintf(stderr, "Unable to open output file 'xlist'.\n");
 exit(1);
 }
 fprintf(fpxlist,"xlist1=c(");
 for (i=0; i<(M-1); i++) {
    fprintf(fpxlist, "%f,\n", xlist[0][i]);
 }
 fprintf(fpxlist, "%f)\n\n", xlist[0][M-1]);
 fprintf(fpxlist,"xlist2=c(");
 for (i=0; i<(M-1); i++) {
    fprintf(fpxlist, "%f,\n", xlist[1][i]);
 }
 fprintf(fpxlist, "%f)\n\n", xlist[1][M-1]);
 fprintf(fpxlist,"\n\n");
 fclose(fpxlist);
 end = clock();
 time3 = (double) (end - start)/CLOCKS_PER_SEC;

 /*test alpha(x) for some specific vector x*/
 start = clock();
 printf("testing alpha ...\n");
 if ((fpout = fopen("out", "w")) == NULL) {
 fprintf(stderr, "Unable to open output file 'xout'.\n");
 exit(1);
 }
 X[0] = 0; X[1] = 0;
 fprintf(fpout,"alpha(%f,%f) is about %f\n",X[0],X[1],a(X));
 fprintf(fpout,"alpha(%f,%f) is about %f\n",X[0],X[1],a(X));
 X[0] = 5; X[1] = 5;
 fprintf(fpout,"alpha(%f,%f) is about %f\n",X[0],X[1],a(X));
 X[0] = 10; X[1] = 10;
 fprintf(fpout,"alpha(%f,%f) is about %f\n",X[0],X[1],a(X));
 X[0] = 20; X[1] = 50;
 fprintf(fpout,"alpha(%f,%f) is about %f\n",X[0],X[1],a(X));
 end = clock();
 time4 = (double) (end - start)/CLOCKS_PER_SEC;
 fprintf(fpout,"it takes %f seconds to run the adaptation algorithm\n", time0);
 fprintf(fpout,"it takes %f seconds to compute g and sigma\n", time1);
 fprintf(fpout,"it takes %f seconds to plot g and sigma\n", time2);
 fprintf(fpout,"it takes %f seconds to generate the Markov Chain\n", time3);
 fprintf(fpout,"it takes %f seconds to test the local acceptance\n", time4);
```

```
        fclose(fpout);
        fclose(fpout);

        printf("Done.\n");
        return(0);
}

/* NORMAL:    return a standard normal random number. */
double normal()
{
        double R, theta, drand48();
        R = - log(drand48());
        theta = 2 * PI * drand48();
        return( sqrt(2*R)  * cos(theta));
}

/* SEEDRAND: SEED RANDOM NUMBER GENERATOR. */
void seedrand()
{
        int i, seed;
        struct timeval tmptv;
        gettimeofday (&tmptv, (struct timezone *)NULL);
        seed = (int) tmptv.tv_usec;
        srand48(seed);
        (void)drand48();    /* Spin it once. */
}

int divides(int aa, int bb)
{
        return( (bb/aa)*aa == bb );
}

double logpi(double xx,double yy)
{
        double sq(),val;
        return (-sq(xx)/2-sq(yy)/2);
}

double g(int nn, double xx1, double xx2)
{
        int ii;
        double    eta(),K(), val,abdif;
        val = 0.0;
        for (ii=0; ii<nn; ii++) {
```

```
        abdif =   pow(sq(xx1-x[0][ii])+sq(xx2-x[1][ii]),0.5);
        val = (1.0 - eta(ii)) *   val + eta(ii) * addsign[ii] / bn * K(abdif/bn);
      }
      return(val);
}

double K(double xx)
{    double abdif = fabs(xx);
     /*the kernel function is the exponential function*/
         return
      /*(     exp(-pow(abdif,alpha1)/alpha2) );*/
     /*you can try other kernel function like: */
     /*( 1.0 / (1.0 + alpha1*pow( abdif, alpha2 ))); */
     height*( 2*indicator(abdif<width) -indicator(abdif<2*width) );
     /* indicator((abdif>=width)&&(abdif<2*width))); */
}

double sigma(int nn, double xx1, double xx2)
{
     double g();
     return( exp(beta) * pow(1.0 + C*pow(sq(xx1)+sq(xx2),0.5), ggamma) *
exp( g(nn,xx1,xx2) ));
}

double logaccprob(int nn, double xx1, double xx2, double yy1, double yy2)
{
   double sigma(), sq();
   return( logpi(yy1,yy2) - logpi(xx1,xx2) + log(sigma(nn,xx1,xx2)) - log(sigma(nn,yy1,yy2))
      +          (sq(xx1-yy1)+sq(xx2-yy2))          *          (0.5/sq(sigma(nn,xx1,xx2))-
0.5/sq(sigma(nn,yy1,yy2))));
}

double eta(int nn)
{
     return (1.0/pow(nn+5,0.5));
     /*return (1.0/pow(nn+5,0.8)); */
}

double sq(double xx)
{
   return(xx*xx);
}

int indicator(int bb)
```

```
{
   if (bb==0)
      return(0);
   else

      return(1);
}

double a(double X[2])
{
      int ii;
      double Y[2], U, tmpval=0.0;
      double alphaproposal[N];
      for(ii=0;ii<N;ii++)
      {
            Y[0] = X[0] + appsigma(X[0],X[1]) * normal();
            Y[1] = X[1] + appsigma(X[0],X[1]) * normal();
            U = drand48();
            if(log(U)<applogaccprob(X[0],X[1],Y[0],Y[1]))
                  tmpval = tmpval + 1;
      }
      tmpval = tmpval/N;
      return tmpval;
}

double appsigma(double xx1,double xx2)
{
      int ii,jj;
      double val;
      if(    (abs(xx1)> right)  ||     (abs(xx2)>right) || (abs(xx1)==right) ||     (abs(xx2)==right)
      )
            val = sigma((RUNLENGTH-1),xx1,xx2);
      for(ii=0;ii<numsigma;ii++){
            for(jj=0;jj<numsigma;jj++){
                        if(((xx1>left+ii*gap)||(xx1==left+ii*gap))&(xx1<left+(ii+1)*gap)&

((xx2>bottom+jj*gap)||(xx2==bottom+jj*gap))&(xx2<bottom+(jj+1)*gap))
                              val = appsig[ii][jj];
                        }
            }
      return val;
}

double applogaccprob(double xx1,double xx2,double yy1,double yy2)
```

```
{
    double appsigma(), sq();
    return( logpi(yy1,yy2) - logpi(xx1,xx2) + log(appsigma(xx1,xx2)) -
log(appsigma(yy1,yy2))
    + (sq(xx1-yy1)+sq(xx2-yy2)) * (0.5/sq(appsigma(xx1,xx2))-
0.5/sq(appsigma(yy1,yy2))));
}
```

1.2 R code

1.2.1 R code to plot the final $g(X)$ and $\sigma(X)$

```
source("gvals")
source("sigvals")

L = length(gvals)
x = seq(-50,50,length = L)
M = 10^5
B = 10^4
varfact = function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
asjd = function(x){
val = 0
for(i in (B+1):M )
  val = val +(xlist[i]-xlist[i-1])^2
return (val/(M-B))
}
par(mfrow=c(1,2))
plot(x,gvals)
plot(x,sigvals)
pdf(file = ifelse(T, "plot.pdf","plot%03d.pdf"),width=10,height=5,pointsize = 1/200)
par(mfrow=c(1,2))
plot(x,gvals)
plot(x,sigvals)
dev.off()
```

1.2 R code

1.2.1 R code to plot the final $g(X)$ and $\sigma(X)$ (Example 1~3)

```
source("gvals")
source("sigvals")

L = length(gvals)
x = seq(-50,50,length = L)
```

```
M = 10^5
B = 10^4
varfact = function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
asjd = function(x){
val = 0
for(i in (B+1):M )
  val = val +(xlist[i]-xlist[i-1])^2
return (val/(M-B))
}
par(mfrow=c(1,2))
plot(x,gvals)
plot(x,sigvals)
pdf(file = ifelse(T, "plot.pdf", "plot%03d.pdf"),width=10,height=5,pointsize = 1/200)
par(mfrow=c(1,2))
plot(x,gvals)
plot(x,sigvals)
dev.off()
```

1.2.2 R code to plot the final $g(X)$ and $\sigma(X)$ (Example 4)

```
source("gvals")
source("sigvals")
L = length(gvals)
L;sqrt(L);
x = seq(-30,30,length = sqrt(L))
M = 10^5
B = 10^4
varfact = function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }

plot(x,gvals[(449*900+1):(450*900)],main="g(x, *)",ylab="g")
lines(x,gvals[((15*28)*900+1):((15*28+1)*900)],col="blue")
lines(x,gvals[((15*25)*900+1):((15*25+1)*900)],col="green")
lines(x,gvals[(300*900+1):(301*900)],col="yellow")
lines(x,gvals[1:900],col="red")

plot(x,sigvals[(449*900+1):(450*900)],ylim=c(0,50),main="sig(x, *)",ylab="sig")
lines(x,sigvals[(300*900+1):(301*900)],col="yellow")
lines(x,sigvals[((15*25)*900+1):((15*25+1)*900)],col="green")
lines(x,sigvals[((15*28)*900+1):((15*28+1)*900)],col="blue")
lines(x,sigvals[1:900],col="red")

pdf(file = ifelse(T, "plot.pdf", "plot%03d.pdf"),width=10,height=5,pointsize = 1/200)
par(mfrow=c(1,2))
plot(x,gvals[(449*900+1):(450*900)])
```

```
lines(x,gvals[(300*900+1):(301*900)],col="yellow")
lines(x,gvals[((15*25)*900+1):((15*25+1)*900)],col="green")
lines(x,gvals[((15*28)*900+1):((15*28+1)*900)],col="blue")
lines(x,gvals[1:900],col="red")
plot(x,sigvals[(449*900+1):(450*900)],ylim=c(0,50),main="sig(x,*)",ylab="sig")
lines(x,sigvals[(300*900+1):(301*900)],col="yellow")
lines(x,sigvals[((15*25)*900+1):((15*25+1)*900)],col="green")
lines(x,sigvals[((15*28)*900+1):((15*28+1)*900)],col="blue")
lines(x,sigvals[1:900],col="red")
dev.off()
```

1.2.3 R code to generate Markov Chain without adaption (Example 1)

```
logpi = function(x){
      return ( - x^2/2);
}

M = 100000;
B = 10000;
xlist = rep(0.0,M);
num = 0;
X = runif(1);
sigma = 5.21;

for(i in 1:M){
      Y = X + sigma  * rnorm(1);
      U = runif(1);
      logalpha = logpi(Y) - logpi(X);
            if (log(U) < logalpha) {
      X = Y;
         num = num + 1;
      }
      xlist[i] = X;
}
cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
cat("acceptance rate =", num/M, "\n");

varfact <- function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
varf = varfact(xlist[(B+1):M]);
se = sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
val = 0
for(i in (B+1):M )
  val = val +(xlist[i]-xlist[i-1])^2
asjd = val/(M-B)
```

```
varf;
se;
asjd;

a = function(x){
        tmpval = 0;
    for(i in 1:M){
          y = x + sigma * rnorm(1);
          u = runif(1);
          if (log(u) < (logpi(y)-logpi(x)))
                tmpval = tmpval +1;
          }
    return (tmpval/M);
}
```

1.2.4 R code to generate Markov Chain without adaption (Example 2)

```
logpi = function(x){
y = 0.0;
if(x<=0) y = - x^2/2;
if(x>=10) y = -(x-10)^2/2;
if(x>0&&x<10) y = log(exp(- x^2/2)+exp(- (x-10)^2/2));
return (y);
}
L = 10^4
x = seq(-30,30,length=L)
y = rep(0,L)
for(i in 1:L)
  y[i] = exp(logpi(x[i]))
plot(x,y)
M = 100000;
B = 10000;
xlist = rep(0.0,M);
num = 0;
X = runif(1);
sigma = 7.7;

for(i in 1:M){
    Y = X + sigma * rnorm(1);
    U = runif(1);
    logalpha = logpi(Y) - logpi(X);
        if (log(U) < logalpha) {
    X = Y;
      num = num + 1;
```

```
      }
      xlist[i] = X;
}
varfact <- function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
varf = varfact(xlist[(B+1):M]);
se = sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );

val = 0
for(i in (B+1):M )
  val = val +(xlist[i]-xlist[i-1])^2
asjd = val/(M-B)
cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
cat("acceptance rate =", num/M, "\n");

varf;
se;
asjd;
```

1.2.5 R code to generate Markov Chain without adaption (Example 3)

```
logpi = function(x){
y = 0.0;
if(x<=-12) y = log(0.3)-(x+10.0)^2/2.0;
if(x>=12)    y = log(0.3)-(x-10.0)^2/2.0;
if(x>-12&&x<12) y = log(0.3*exp(- (x+10.0)^2/2)+0.4*exp(-x^2/2)+0.3*exp(-(x-10.0)^2/2));
return (y);
}

M = 100000;
B = 10000;
xlist = rep(0.0,M);
num = 0;
X = runif(1);
sigma = 10.8;

for(i in 1:M){
      Y = X + sigma * rnorm(1);
      U = runif(1);
      logalpha = logpi(Y) - logpi(X);
            if (log(U) < logalpha) {
      X = Y;
         num = num + 1;
      }
      xlist[i] = X;
}
```

```
varfact <- function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
varf = varfact(xlist[(B+1):M]);
se = sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );

val = 0
for(i in (B+1):M )
  val = val +(xlist[i]-xlist[i-1])^2
asjd = val/(M-B)
cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
cat("acceptance rate =", num/M, "\n");
cat("varfact is about",varf,"\n");
cat("standard error is about",se,"\n");
cat("average squared jump distance is about",asjd,"\n");
par(mfrow=c(1,2))
plot(xlist[(B+1):M],type="l");
plot(xlist[(B+1):M],type="p");
```

1.2.6 R code to generate Markov Chain without adaption (Example 4)

```
logpi = function(x,y){
return (-x^2/2-y^2/2);
}

M = 100000;
B = 10000;
xlist1 = xlist2 = rep(0.0,M);
num = 0;
X1 = runif(1);
X2 = runif(1);
sigma = 2.39;

for(i in 1:M){
      Y1 = X1 + sigma * rnorm(1);
      Y2 = X2 + sigma * rnorm(1);
      U = runif(1);
      if (log(U) < (logpi(Y1,Y2) - logpi(X1,X2))) {
            X1 = Y1;
            X2 = Y2;
            num = num + 1;
        }
        xlist1[i] = X1;
      xlist2[i] = X2;
}

varfact = function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
```

```
asjd = function(xlist){
      val = 0
      for(i in (B+1):M )
      val = val +(xlist[i]-xlist[i-1])^2
      return (val/(M-B));
}
cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
cat("acceptance rate =", num/M, "\n");
cat("varfact is about",varfact(xlist1[(B+1):M]), "and",varfact(xlist2[(B+1):M]), "\n");
cat("standard error is about",(sd(xlist1[(B+1):M]) / sqrt(M-B) *
sqrt( varfact(xlist1[(B+1):M]))), "and",(sd(xlist2[(B+1):M]) / sqrt(M-B) *
sqrt( varfact(xlist2[(B+1):M]))), "\n");
cat("average squared jump distance is about",asjd(xlist1), "and",asjd(xlist2), "\n");
par(mfrow=c(2,2));
plot(xlist1[(B+1):M],type="l");
plot(xlist1[(B+1):M],type="p");
plot(xlist2[(B+1):M],type="l");
plot(xlist2[(B+1):M],type="p");
par(mfrow=c(1,2));
plot(xlist1[(B+1):M],xlist2[(B+1):M],type="p");
plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l")
```

1.2.7 R code to compute different efficiency measures (Example 1~3)

```
par(mfrow=c(3,2))
source("xlist1");plot(xlist[(B+1):M],type="l");source("xlist2");plot(xlist[(B+1):M],type="l");
source("xlist3");plot(xlist[(B+1):M],type="l");source("xlist4");plot(xlist[(B+1):M],type="l");
source("xlist5");plot(xlist[(B+1):M],type="l");

pdf(file = ifelse(T, "trace plot.pdf","trace plot%03d.pdf"),width=10,height=5,pointsize =
1/200)
par(mfrow=c(3,2))
source("xlist1");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
source("xlist2");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
source("xlist3");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
dev.off()
```

1.2.8 R code to compute different efficiency measures (Example 4)

```
pdf(file = ifelse(T, "plot.pdf","plot%03d.pdf"),width=10,height=5,pointsize = 1/200)
par(mfrow=c(1,2))
plot(x,gvals[(449*900+1):(450*900)])
lines(x,gvals[(300*900+1):(301*900)],col="yellow")
lines(x,gvals[((15*25)*900+1):((15*25+1)*900)],col="green")
lines(x,gvals[((15*28)*900+1):((15*28+1)*900)],col="blue")
lines(x,gvals[1:900],col="red")
plot(x,sigvals[(449*900+1):(450*900)],ylim=c(0,50),main="sig(x,*)",ylab="sig")
lines(x,sigvals[(300*900+1):(301*900)],col="yellow")
lines(x,sigvals[((15*25)*900+1):((15*25+1)*900)],col="green")
lines(x,sigvals[((15*28)*900+1):((15*28+1)*900)],col="blue")
lines(x,sigvals[1:900],col="red")
dev.off()
```

1.2.9 R code to implement OLS in approximation of the final $\sigma(X)$

```
####comparison######

L = 10001
left = -50
right = 50
gap = 0.01

I1 = I2 = I3 = rep(0,L)
source("sigvals-11")
for(i in 1:L) I1[i] = sigvals[i]
source("sigvals-14")
for(i in 1:L) I2[i] = sigvals[i]
source("sigvals-16")
for(i in 1:L) I3[i] = sigvals[i]

x = seq(-50,50,length = L)

par(mfrow=c(1,3))
plot(x,I1)
plot(x,I2)
plot(x,I3)


####OLS for CASE 14######
left = -3
right = 3
L = 601
gap = 0.01
```

```
l = rep(0,L)
for(i in    1:L )
      l[i]=l2[i+4700]

c = 0.0;
exponnet = 1.0;
n = 200;
k = 120;
sse = matrix(rep(0.0,n*k),nrow=k);
rowindex = 1;
colindex = 1;


for(row in 1:k){
      exponent = 12*row/k;
      for(col in 1:n){
            c = 20*col/n;
                  for(i in 1:L){
                        X = left + (i-1)* gap;
                        sse[row,col] = sse[row,col] +    (l[i]-(c*abs(X)^exponent+l[(L-1)/2+1]))^2 ;
                  }
      if( sse[rowindex,colindex]>sse[row,col] ) {
            rowindex = row;
            colindex = col;}}
}

c =20*colindex/n;
exponent = 12*rowindex/k;
c1=c;
e1=exponent;

app = rep(0.0,L);
for(i in 1:L) {
X = left + (i-1)* gap;
app[i] = c*abs(X)^exponent+l[(L-1)/2+1];
}
x = seq(left,right,length=L)
plot(x,app);
lines(x,l,col="red")

c1*abs(-3)^e1+l[(L-1)/2+1]

left = -50
```

```
right = -3
L = 4701
gap = 0.01

l = rep(0,L)
for(i in    1:L )
      l[i]=l2[i]

c = 0.0;
exponnet = 0.0;
n = 200;
k = 100;
sse = matrix(rep(0.0,n*k),nrow=k);
rowindex = 1;
colindex = 1;


for(row in 1:k){
      exponent = row/k;
      for(col in 1:n){
            c = 20*col/n;
                  for(i in 1:L){
                        X = left + (i-1)* gap;
                        sse[row,col]                =                sse[row,col]                +
(l[i]-(c*abs(X)^exponent+6.973254-3^exponent*c))^2 ;
                  }
      if( sse[rowindex,colindex]>sse[row,col] ) {
            rowindex = row;
            colindex = col;}}
}

c = 20*colindex/n;
exponent = rowindex/k;
c;c2=c
exponent;e2=exponent

app = rep(0.0,L);
for(i in 1:L) {
X = left + (i-1)* gap;
app[i] = c*abs(X)^exponent+6.973254-3^exponent*c;
}

x = seq(left,right,length=L)
plot(x,app);
```

```
lines(x,l,col="red")

sigma = function(x){
if(abs(x)<3) y = c1*(abs(x))^e1 + 5.572407;
if(abs(x)==3||abs(x)>3) y = c2*(abs(x))^e2 + 6.973254-3^e2*c2;
return (y)
}

x = seq(-50,50,length=10001)
y = rep(0,10001)
for( i in 1:10001)
y[i] = sigma(x[i])
plot(x,l2)
lines(x,y,col="red")


#### check ####
logpi = function(x){
return (- x^2/2);
}

f = function(x,y){
return
( logpi(y)-logpi(x)+(x-y)^2/2*(1/sigma(x)^2-1/sigma(y)^2))+log(sigma(x))-log(sigma(y)    );
}

M = 1000
x = c(30,50,80,150,200,300);
z = rep(1,M)

par(mfrow=c(3,2))
for(i in 1:6){
y = seq(x[i]-3*sigma(x[i]),x[i]+3*sigma(x[i]),length=M)
      for(j in 1:M){
            if(f(x[i],y[j])<0) z[j]= exp(f(x[i],y[j]));
      }
plot(y,z,main=c("when x=",x[i]))
}
```

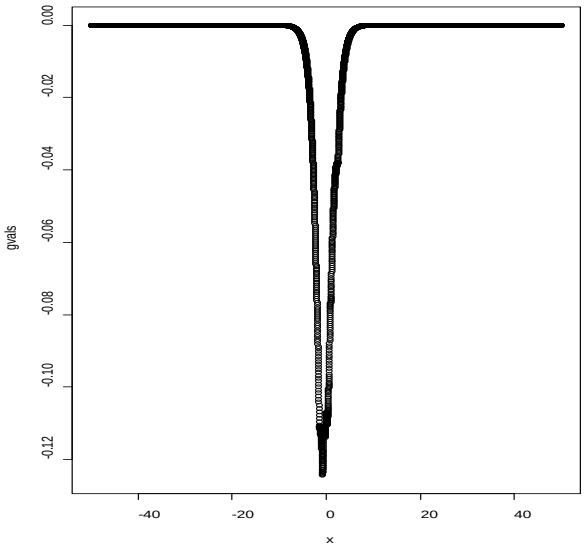
2. Output

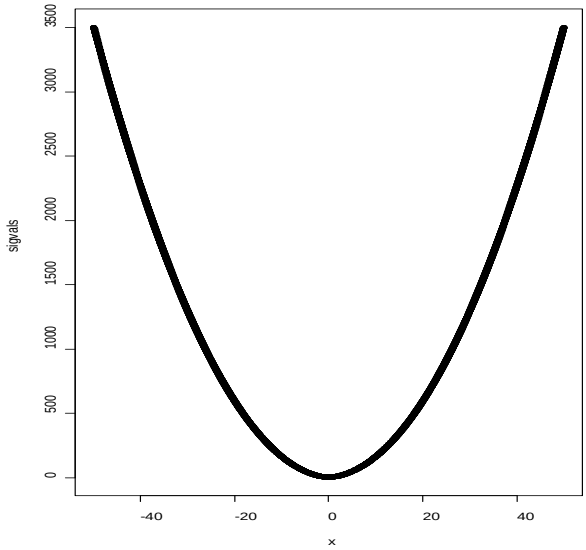
2.1 Example 1: normal distribution in R^1

2.1.1 kernel function $K(x) = e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

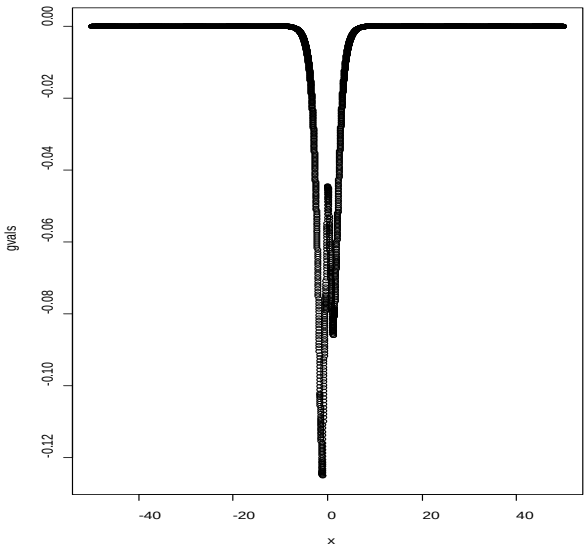
2.1.1.1 case 1 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

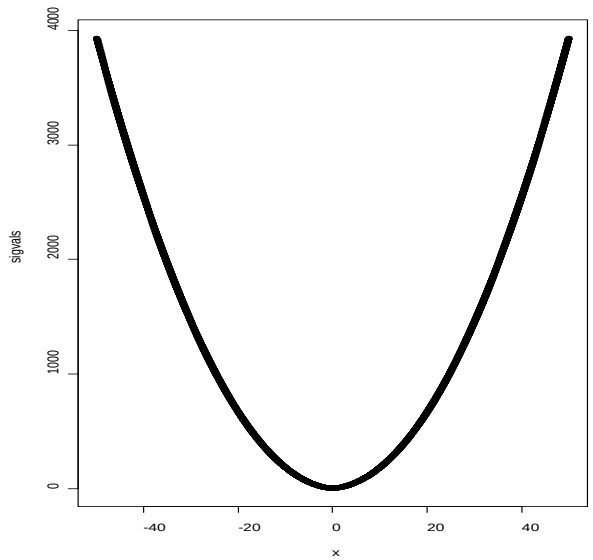
Without approximation:





With approximation:





Accepted 22920 out of 100000 (22.920%).
final beta = 0.413548

alpha(0.000000) is 0.294400
alpha(2.000000) is 0.146900
alpha(5.000000) is 0.079300
alpha(10.000000) is 0.047600
alpha(20.000000) is 0.022500
alpha(50.000000) is 0.009100
alpha(100.000000) is 0.005800
alpha(150.000000) is 0.002600
it takes 233.63636 seconds to run the adaptation algorithm

it takes 95.5646 seconds to compute g and sigma

it takes 21.525 seconds to generate the first Markov Chain

it takes 25.2524 seconds to generate the second Markov Chain

it takes 23.8585 seconds to generate the third Markov Chain

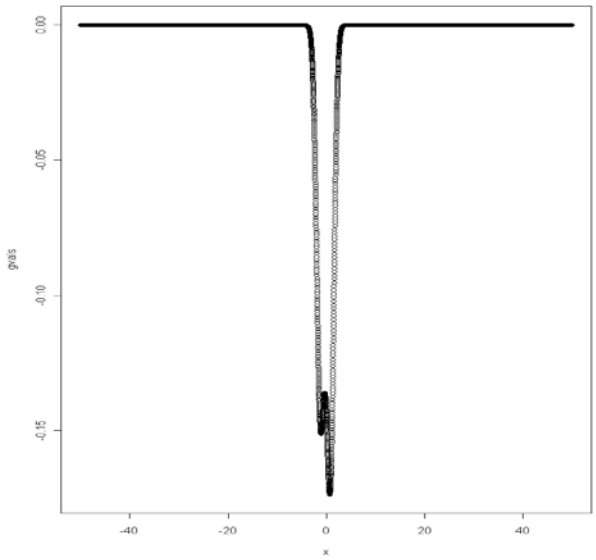
it takes 21.156 seconds to generate the fourth Markov Chain

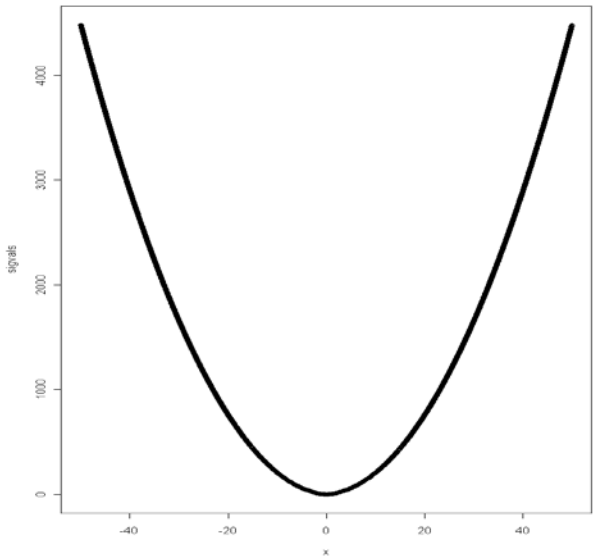
it takes 26.6363 seconds to generate the fifth Markov Chain

it takes 1564.99 seconds to test the local acceptance

```
>    source("xlist1");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 11.35909
[1] 0.01122983
[1] 0.33542
>    source("xlist2");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 12.05121
[1] 0.01172226
[1] 0.3463196
>    source("xlist3");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 11.45263
[1] 0.01136355
[1] 0.3383667
>    source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 10.70036
[1] 0.01089844
[1] 0.3378794
>    source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 10.27269
[1] 0.01078979
[1] 0.3482317
```

2.1.1.2 case 2 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$



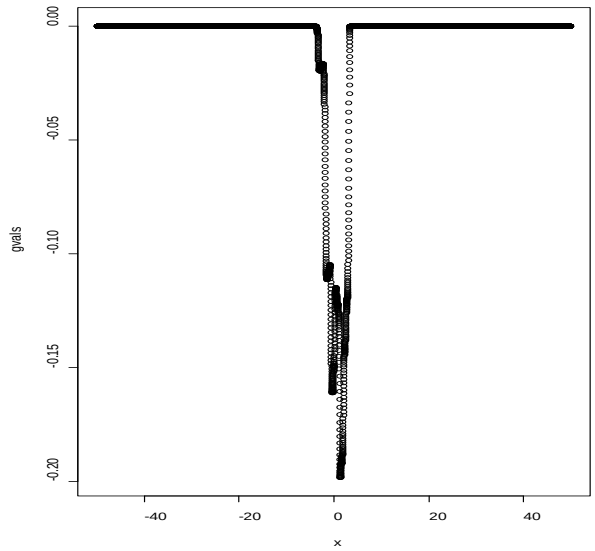


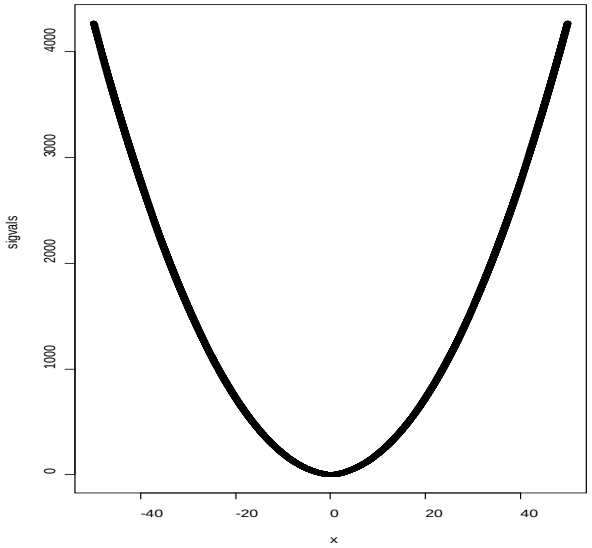
Accepted 21393 out of 100000 (21.393%).

final beta = 0.543719
alpha(0.000000) is 0.275100
alpha(2.000000) is 0.123300
alpha(5.000000) is 0.065000
alpha(10.000000) is 0.036300
alpha(20.000000) is 0.020400
alpha(50.000000) is 0.008500
alpha(100.000000) is 0.003700
alpha(150.000000) is 0.004400
it takes 231.632000 seconds to run the adaptation algorithm
it takes 153.155000 seconds to compute g and sigma
it takes 22.924000 seconds to generate the first Markov Chain
it takes 23.594000 seconds to generate the second Markov Chain
it takes 23.285000 seconds to generate the third Markov Chain
it takes 21.130000 seconds to generate the fourth Markov Chain
it takes 21.698000 seconds to generate the fifth Markov Chain
it takes 1627.749000 seconds to test the local acceptance

> source("xlist2");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 12.98504
[1] 0.01201336
[1] 0.3106892
> source("xlist3");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 13.36214
[1] 0.01222844
[1] 0.3051376
> source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 12.5798
[1] 0.01166746
[1] 0.3061098
> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 11.6263
[1] 0.01115963
[1] 0.309262

2.1.1.3 case 3 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$





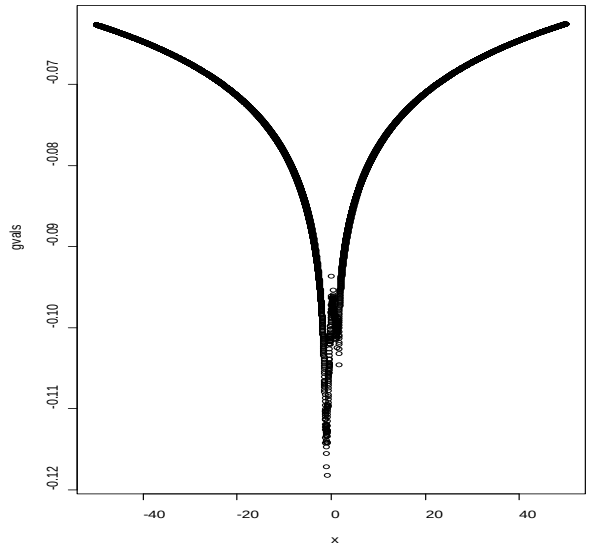
Accepted 22184 out of 100000 (22.184%).
final beta = 0.496175
alpha(0.000000) is 0.287000
alpha(2.000000) is 0.146700
alpha(5.000000) is 0.067300
alpha(10.000000) is 0.042100

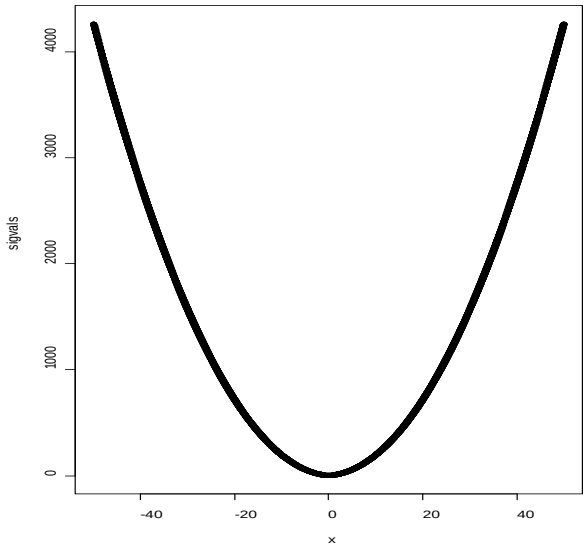
alpha(20.000000) is 0.022000
alpha(50.000000) is 0.008600
alpha(100.000000) is 0.005500
alpha(150.000000) is 0.002500
it takes 482.924000 seconds to run the adaptation algorithm
it takes 231.983000 seconds to compute g and sigma
it takes 36.292000 seconds to generate the first Markov Chain
it takes 38.465000 seconds to generate the second Markov Chain
it takes 36.953000 seconds to generate the third Markov Chain
it takes 35.281000 seconds to generate the fourth Markov Chain
it takes 36.833000 seconds to generate the fifth Markov Chain
it takes 2345.843000 seconds to test the local acceptance

\

> source("xlist3");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 12.00902
[1] 0.0117056
[1] 0.3388592
> source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 11.17787
[1] 0.01105268
[1] 0.3208399
> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 12.35328
[1] 0.01156459
[1] 0.3275879

2.1.1.4 case 4 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$



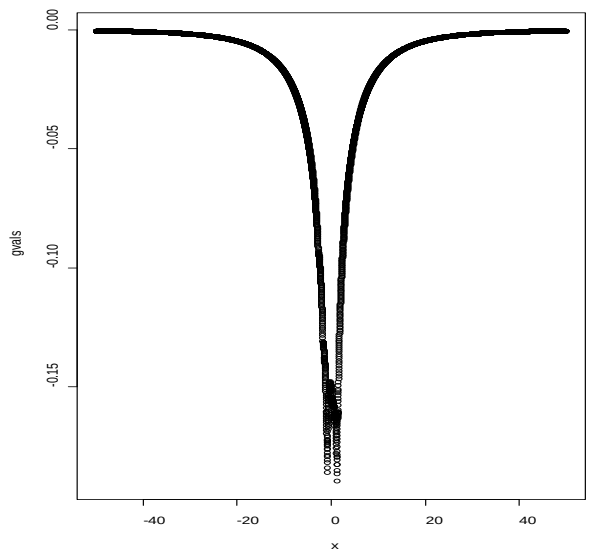


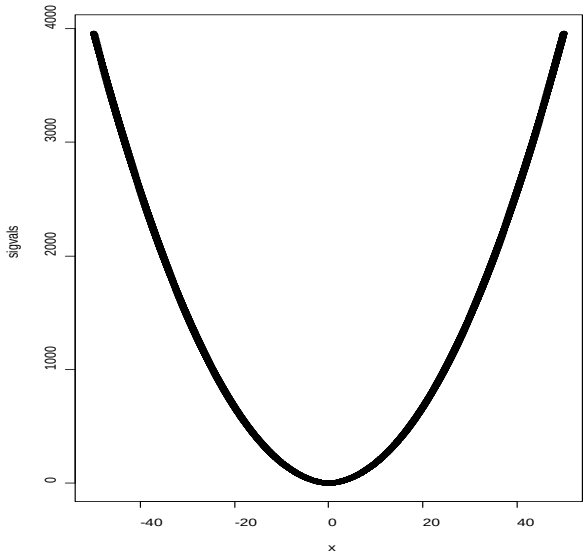
Accepted 20175 out of 100000 (20.175%).
final beta = 0.557074
alpha(0.000000) is 0.263300
alpha(2.000000) is 0.135800
alpha(5.000000) is 0.070300
alpha(10.000000) is 0.041900
alpha(20.000000) is 0.021200
alpha(50.000000) is 0.008200
alpha(100.000000) is 0.005600
alpha(150.000000) is 0.004000

it takes 436.537000 seconds to run the adaptation algorithm
it takes 191.065000 seconds to compute g and sigma
it takes 36.543000 seconds to generate the first Markov Chain
it takes 36.102000 seconds to generate the second Markov Chain
it takes 39.417000 seconds to generate the third Markov Chain
it takes 37.934000 seconds to generate the fourth Markov Chain
it takes 37.694000 seconds to generate the fifth Markov Chain
it takes 1960.229000 seconds to test the local acceptance

[1] 0.3049649
> source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 12.47907
[1] 0.01187855
[1] 0.3073156
> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 13.20204
[1] 0.01218168
[1] 0.303518

2.1.1.5 case 5 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.5, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$



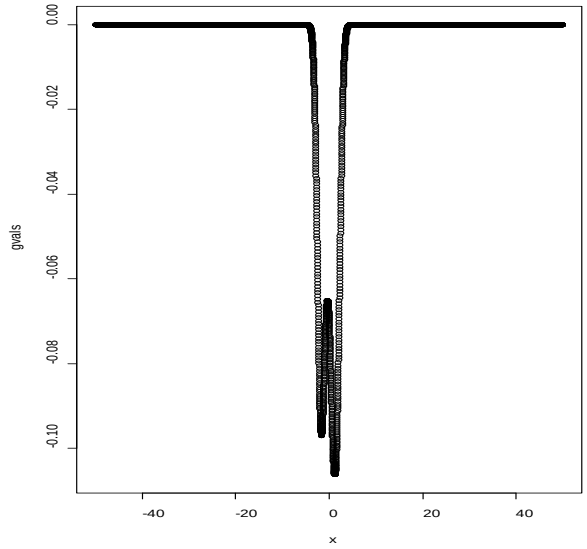


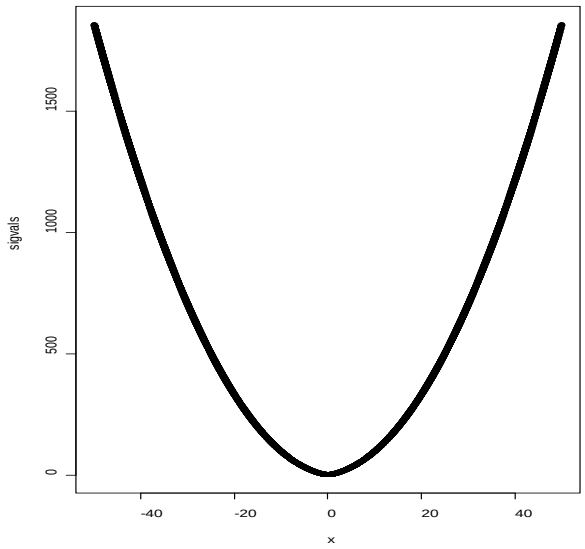
Accepted 24190 out of 100000 (24.190%).
final beta = 0.420755
alpha(0.000000) is 0.309200
alpha(2.000000) is 0.150900
alpha(5.000000) is 0.076500
alpha(10.000000) is 0.045300
alpha(20.000000) is 0.022600
alpha(50.000000) is 0.010200
alpha(100.000000) is 0.003300
alpha(150.000000) is 0.003400
it takes 458.670000 seconds to run the adaptation algorithm
it takes 208.940000 seconds to compute g and sigma
it takes 106.120000 seconds to generate the first Markov Chain
it takes 108.450000 seconds to generate the second Markov Chain
it takes 107.710000 seconds to generate the third Markov Chain

it takes 106.870000 seconds to generate the fourth Markov Chain
it takes 108.390000 seconds to generate the fifth Markov Chain
it takes 2139.130000 seconds to test the local acceptance

[1] 0.3406725
> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 10.34777
[1] 0.01072979
[1] 0.3623652

2.1.1.6 case 6 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 2, \gamma = 2$ with fixed bandwidth $b_n = 1$

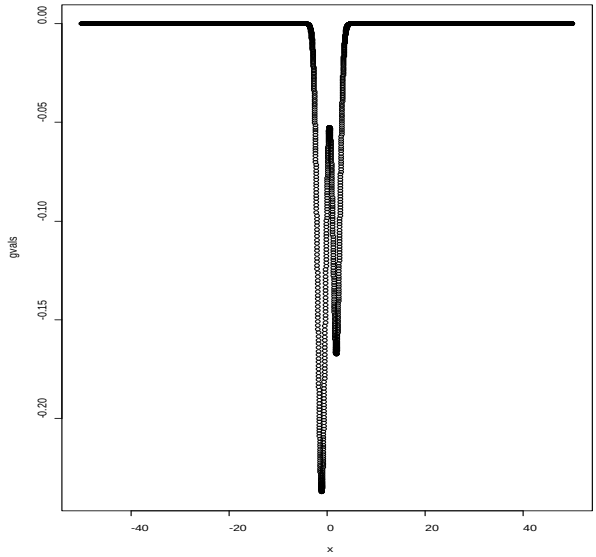


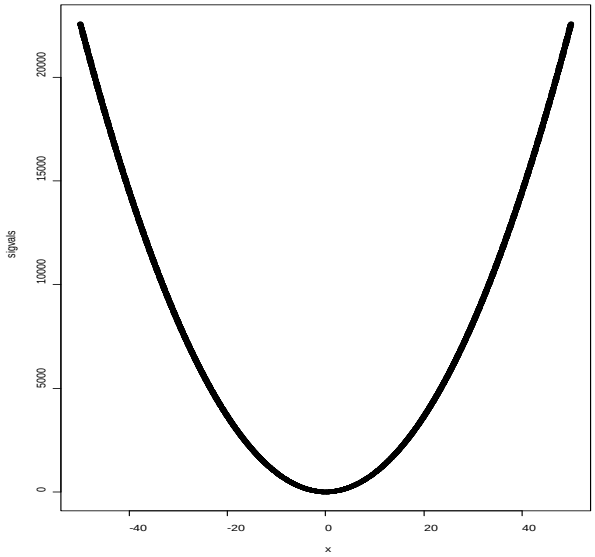


Accepted 21452 out of 100000 (21.452%).
final beta = -0.217726
alpha(0.000000) is 0.345100
alpha(2.000000) is 0.092500
alpha(5.000000) is 0.043700
alpha(10.000000) is 0.019400
alpha(20.000000) is 0.011800
alpha(50.000000) is 0.004900
alpha(100.000000) is 0.001900
alpha(150.000000) is 0.002400
it takes 453.780000 seconds to run the adaptation algorithm
it takes 325.460000 seconds to compute g and sigma
it takes 142.590000 seconds to generate the first Markov Chain
it takes 143.880000 seconds to generate the second Markov Chain
it takes 136.990000 seconds to generate the third Markov Chain
it takes 139.650000 seconds to generate the fourth Markov Chain
it takes 135.720000 seconds to generate the fifth Markov Chain
it takes -1420.447296 seconds to test the local acceptance

[1] 14.36675
[1] 0.01244616
[1] 0.2595380

2.1.1.7 case 7 $\alpha_1 = 2, \alpha_2 = 1, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$

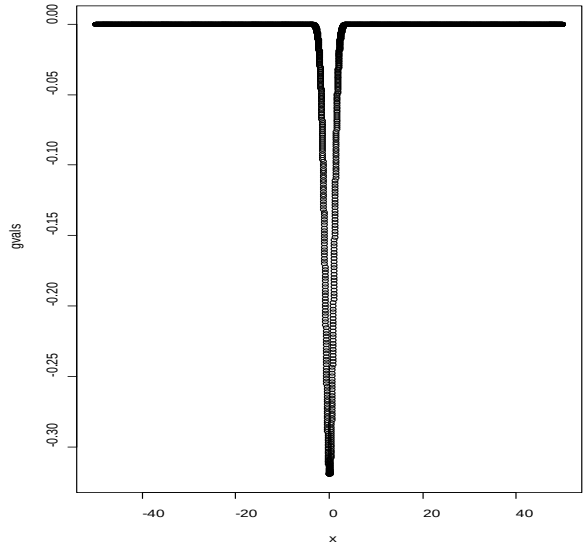


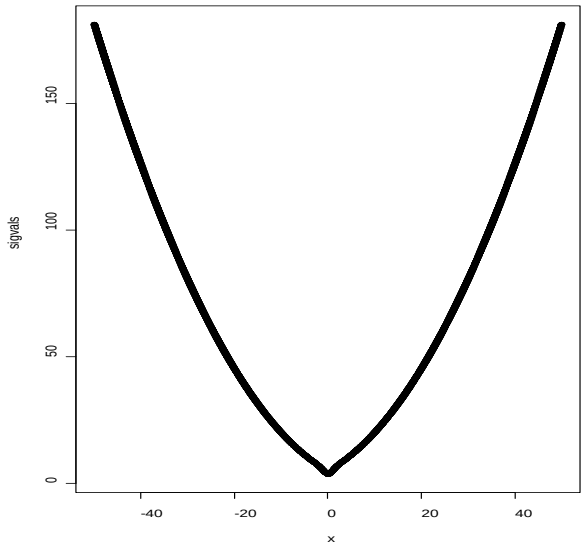


Accepted 23083 out of 100000 (23.083%).
final beta = -2.408190
alpha(0.000000) is 0.638400
alpha(2.000000) is 0.050600
alpha(5.000000) is 0.015100
alpha(10.000000) is 0.008200
alpha(20.000000) is 0.003700
alpha(50.000000) is 0.001800
alpha(100.000000) is 0.000600
alpha(150.000000) is 0.000300
it takes 464.760000 seconds to run the adaptation algorithm
it takes 320.620000 seconds to compute g and sigma
it takes 245.030000 seconds to generate the first Markov Chain
it takes 240.520000 seconds to generate the second Markov Chain
it takes 235.190000 seconds to generate the third Markov Chain
it takes 230.100000 seconds to generate the fourth Markov Chain
it takes 228.730000 seconds to generate the fifth Markov Chain
it takes -1298.167296 seconds to test the local acceptance

[1] 28.38106
[1] 0.01755109
[1] 0.1614916

2.1.1.8 case 8 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

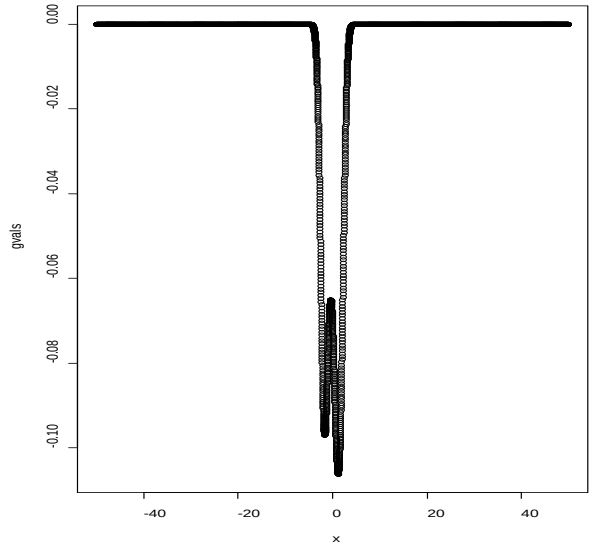


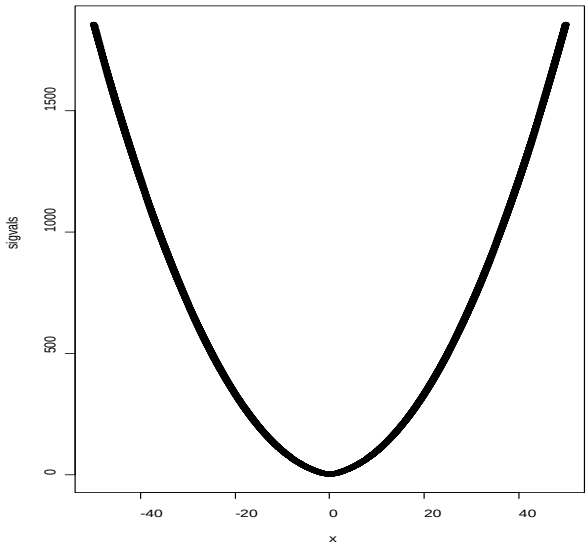


Accepted 23789 out of 100000 (23.789%).
final beta = 1.685565
alpha(0.000000) is 0.209326
alpha(2.000000) is 0.310810
alpha(5.000000) is 0.450666
alpha(10.000000) is 0.503885
alpha(20.000000) is 0.509007
alpha(50.000000) is 0.500492
alpha(100.000000) is 0.497852
alpha(150.000000) is 0.502459
alpha(200.000000) is 0.498544
alpha(300.000000) is 0.501603
it takes 219.483000 seconds to run the adaptation algorithm
it takes 144.400000 seconds to compute g and sigma
it takes 19.396000 seconds to generate the first Markov Chain
it takes 19.387000 seconds to generate the second Markov Chain
it takes 19.118000 seconds to generate the third Markov Chain
it takes 19.331000 seconds to generate the fourth Markov Chain
it takes 19.341000 seconds to generate the fifth Markov Chain
it takes 2855.818000 seconds to test the local acceptance

[1] 6.911154
[1] 0.008697184
[1] 0.5301068

2.1.1.9 case 9 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.5, \gamma = 2$ with fixed bandwidth $b_n = 1$



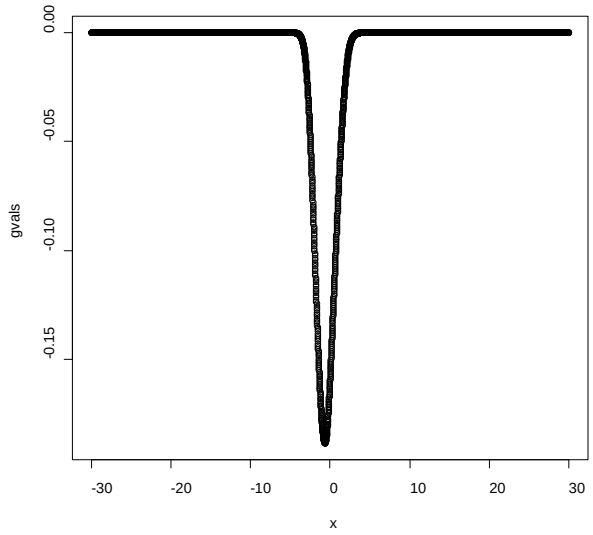


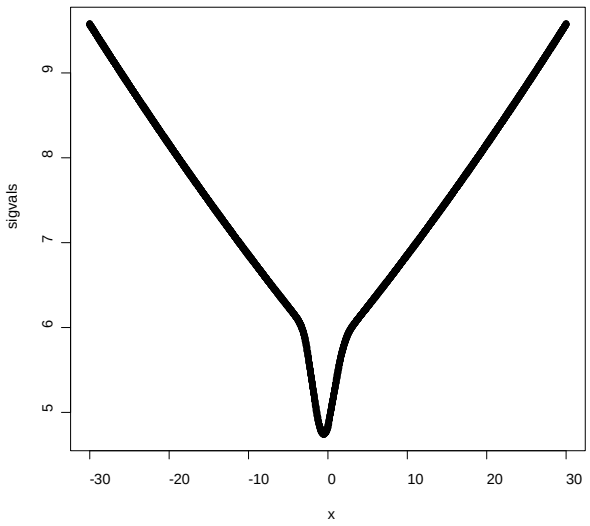
Accepted 21014 out of 100000 (21.014%).
final beta = 1.010674
alpha(0.000000) is 0.232100
alpha(2.000000) is 0.183200
alpha(5.000000) is 0.124800
alpha(10.000000) is 0.083000
alpha(20.000000) is 0.049300
alpha(50.000000) is 0.021500
alpha(100.000000) is 0.010600
alpha(150.000000) is 0.006200
it takes 441.064000 seconds to run the adaptation algorithm
it takes 244.381000 seconds to compute g and sigma
it takes 28.692000 seconds to generate the first Markov Chain
it takes 28.120000 seconds to generate the second Markov Chain
it takes 28.852000 seconds to generate the third Markov Chain
it takes 28.320000 seconds to generate the fourth Markov Chain
it takes 29.292000 seconds to generate the fifth Markov Chain
it takes 2695.236000 seconds to test the local acceptance

[1] 10.2363
[1] 0.01076473
[1] 0.3784802

2.1.1.10 case 10 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.01, \gamma = 2$ with fixed bandwidth

$b_n = 1$

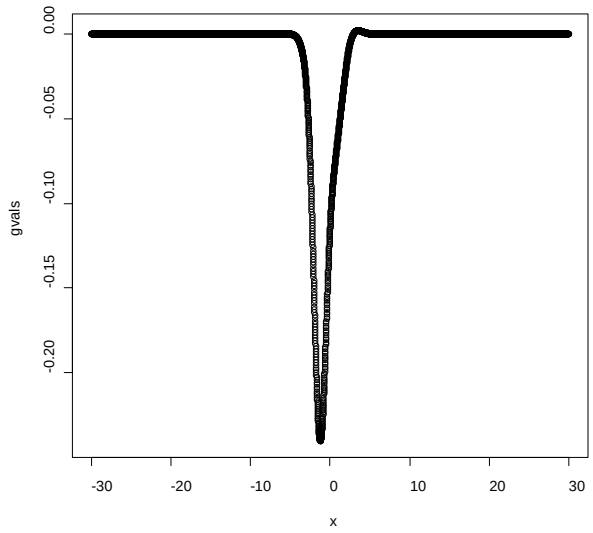


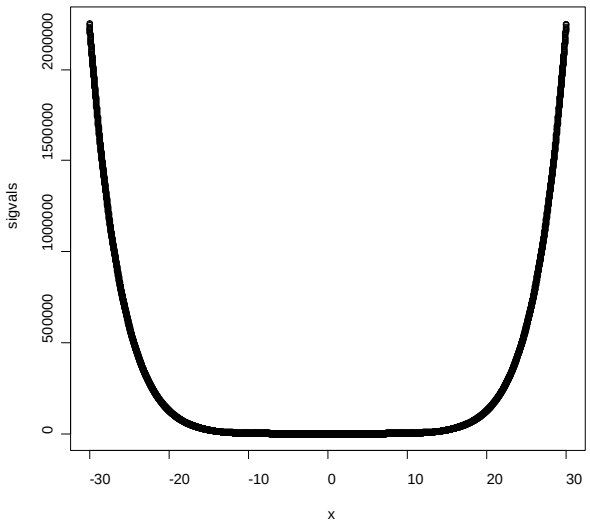


Accepted 23515 out of 100000 (23.515%).
final beta = 1.734452
alpha(0.000000) is 0.197174
alpha(2.000000) is 0.303052
alpha(5.000000) is 0.461148
alpha(10.000000) is 0.507069
alpha(20.000000) is 0.504815
alpha(50.000000) is 0.501644
alpha(100.000000) is 0.500501
alpha(150.000000) is 0.496824
it takes 216.113000 seconds to run the adaptation algorithm
it takes 66.348000 seconds to compute g and sigma
it takes 11.710000 seconds to generate the first Markov Chain
it takes 11.877000 seconds to generate the second Markov Chain
it takes 11.680000 seconds to generate the third Markov Chain
it takes 11.616000 seconds to generate the fourth Markov Chain
it takes 11.568000 seconds to generate the fifth Markov Chain
it takes 1890.326000 seconds to test the local acceptance

```
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008994134
> asjd(xlist)
[1] 0.5588261
>
> source("xlist4")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 5.971771
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008119293
> asjd(xlist)
[1] 0.5441406
>
> source("xlist5")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 6.151567
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008323233
> asjd(xlist)
[1] 0.5366647
```

2.1.1.11 case 11 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 10$ with fixed bandwidth $b_n = 1$



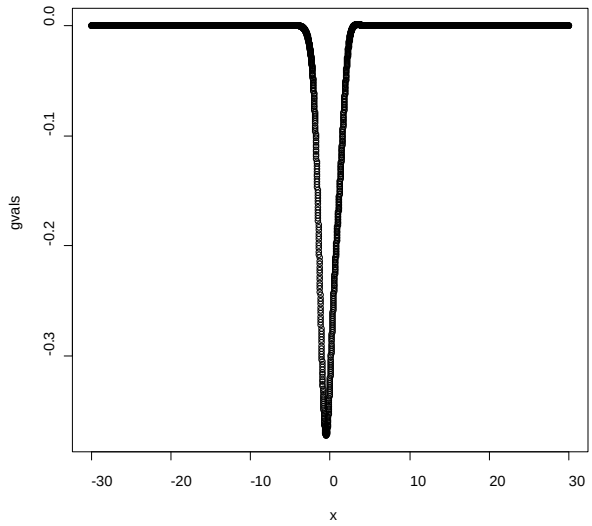


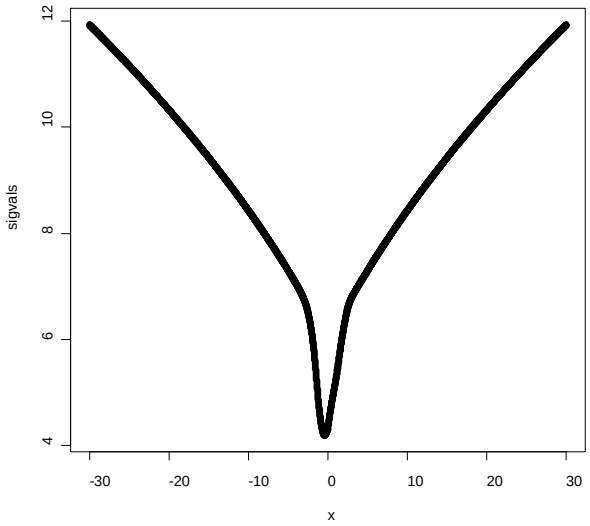
Accepted 23997 out of 100000 (23.997%).
final beta = 0.765112

alpha(0.000000) is 0.283786
alpha(2.000000) is 0.143202
alpha(5.000000) is 0.030523
alpha(10.000000) is 0.002805
alpha(20.000000) is 0.000000
alpha(50.000000) is 0.000000
alpha(100.000000) is 0.000000
alpha(150.000000) is 0.000000
it takes 214.196000 seconds to run the adaptation algorithm
it takes 67.179000 seconds to compute g and sigma
it takes 23.495000 seconds to generate the first Markov Chain
it takes 20.797000 seconds to generate the second Markov Chain
it takes 21.091000 seconds to generate the third Markov Chain
it takes 27.039000 seconds to generate the fourth Markov Chain
it takes 19.527000 seconds to generate the fifth Markov Chain
it takes 2947.240000 seconds to test the local acceptance

```
> asjd(xlist)
[1] 0.3507836
>
> source("xlist2")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 11.71392
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.01135253
> asjd(xlist)
[1] 0.355038
>
> source("xlist3")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 10.21051
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.01073888
> asjd(xlist)
[1] 0.3733516
>
> source("xlist4")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 10.77967
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.01113413
> asjd(xlist)
[1] 0.3672165
>
> source("xlist5")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 10.77254
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.01087192
> asjd(xlist)
[1] 0.3593188
```

2.1.1.12 case 12 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

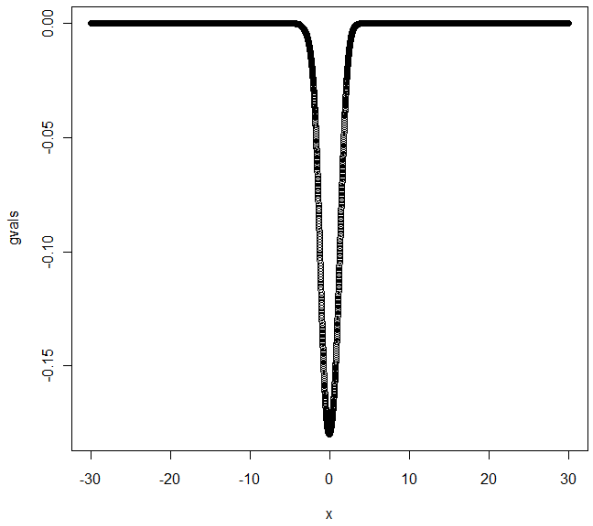


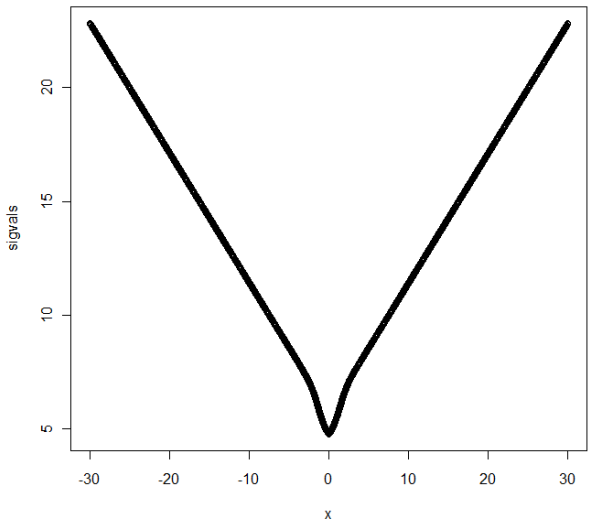


Accepted 23701 out of 100000 (23.701%).
final beta = 1.785324
alpha(0.000000) is 0.206467
alpha(2.000000) is 0.285679
alpha(5.000000) is 0.432219
alpha(10.000000) is 0.494165
alpha(20.000000) is 0.491928
alpha(50.000000) is 0.505376
alpha(100.000000) is 0.495385
alpha(150.000000) is 0.497050
it takes 396.720000 seconds to run the adaptation algorithm
it takes 202.050000 seconds to compute g and sigma
it takes 118.802704 seconds to generate the first Markov Chain
it takes 120.982704 seconds to generate the second Markov Chain
it takes 120.712704 seconds to generate the third Markov Chain
it takes 118.832704 seconds to generate the fourth Markov Chain
it takes 121.202704 seconds to generate the fifth Markov Chain
it takes -1983.704592 seconds to test the local acceptance

```
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008346926
> asjd(xlist)
[1] 0.5154546
>
> source("xlist4")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 6.739889
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.00864459
> asjd(xlist)
[1] 0.5293424
>
> source("xlist5")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 6.701922
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.00869634
> asjd(xlist)
[1] 0.5233073
```

2.1.1.13 case 13 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 1$ with fixed bandwidth $b_n = 1$



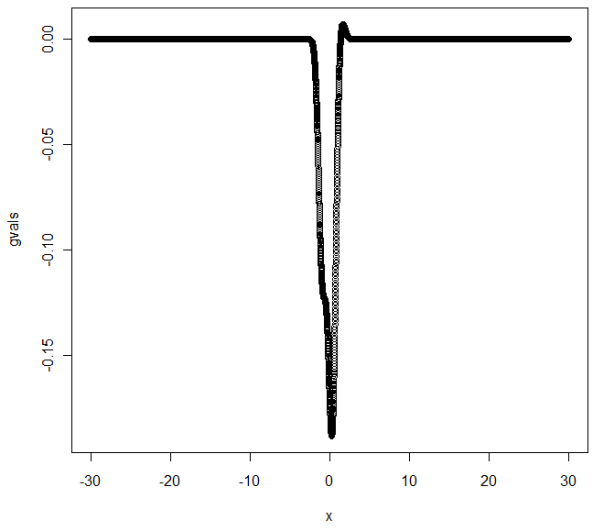


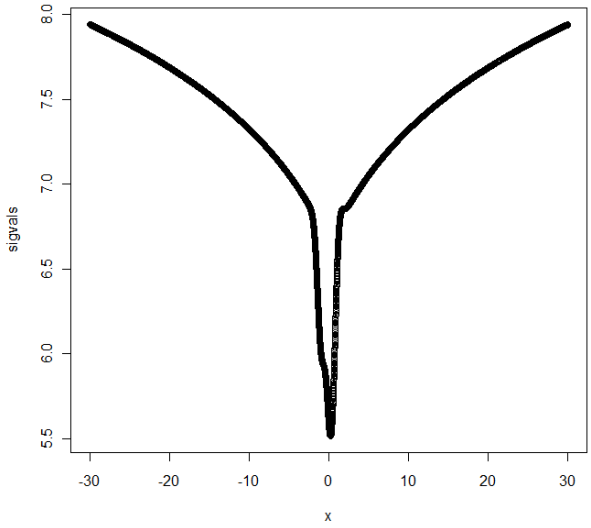
Accepted 21399 out of 100000 (21.399%).
final beta = 1.741715

alpha(0.000000) is 0.187300
alpha(2.000000) is 0.261245
alpha(5.000000) is 0.379365
alpha(10.000000) is 0.467408
alpha(20.000000) is 0.486090
alpha(50.000000) is 0.496151
alpha(100.000000) is 0.498099
alpha(150.000000) is 0.494002
it takes 393.160000 seconds to run the adaptation algorithm
it takes 203.950000 seconds to compute g and sigma
it takes 93.012704 seconds to generate the first Markov Chain
it takes 94.202704 seconds to generate the second Markov Chain
it takes 90.432704 seconds to generate the third Markov Chain
it takes 88.402704 seconds to generate the fourth Markov Chain
it takes 88.382704 seconds to generate the fifth Markov Chain
it takes 1974.234592 seconds to test the local acceptance

[1] 0.008915139
> asjd(xlist)
[1] 0.4866257
>
> source("xlist2")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.440486
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));
[1] 0.009014073
> asjd(xlist)
[1] 0.4721054
>
> source("xlist3")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.480085
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));
[1] 0.009195997
> asjd(xlist)
[1] 0.4982576
>
> source("xlist4")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.304401
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));
[1] 0.009095373
> asjd(xlist)
[1] 0.4961344
>
> source("xlist5")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 8.057888
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));
[1] 0.009576378
> asjd(xlist)
[1] 0.4929519

2.1.1.14 case 14 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth $b_n = 1$



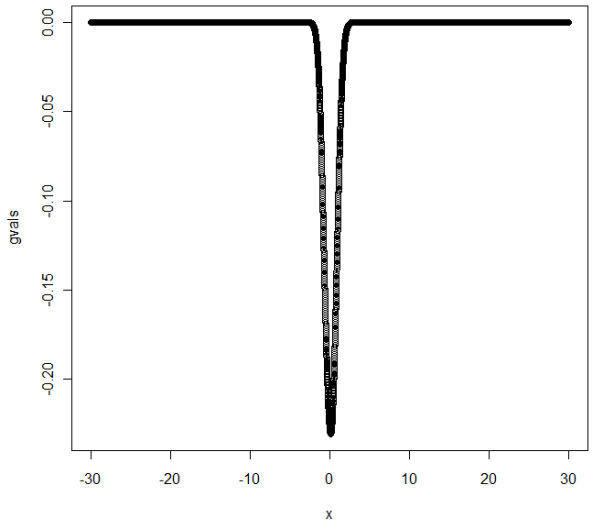


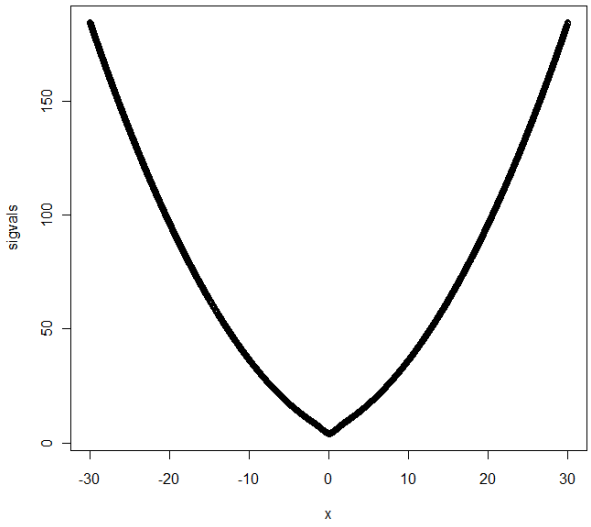
Accepted 20686 out of 100000 (20.686%).
final beta = 1.892872
alpha(0.000000) is 0.168239
alpha(2.000000) is 0.273334
alpha(5.000000) is 0.440324
alpha(10.000000) is 0.502495
alpha(20.000000) is 0.499843
alpha(50.000000) is 0.496880
alpha(100.000000) is 0.498992
alpha(150.000000) is 0.500321
it takes 148.470000 seconds to run the adaptation algorithm
it takes 121.910000 seconds to compute g and sigma
it takes 16.530000 seconds to generate the first Markov Chain
it takes 16.430000 seconds to generate the second Markov Chain
it takes 16.540000 seconds to generate the third Markov Chain
it takes 16.620000 seconds to generate the fourth Markov Chain
it takes 16.720000 seconds to generate the fifth Markov Chain
it takes 1005.607296 seconds to test the local acceptance

```
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008678129
> asjd(xlist)
[1] 0.4830742
>
> source("xlist4")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.697511
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.009185244
> asjd(xlist)
[1] 0.4759112

>
> source("xlist5")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.061558
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.00885177
> asjd(xlist)
[1] 0.4897989
```

2.1.1.15 case 15 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$



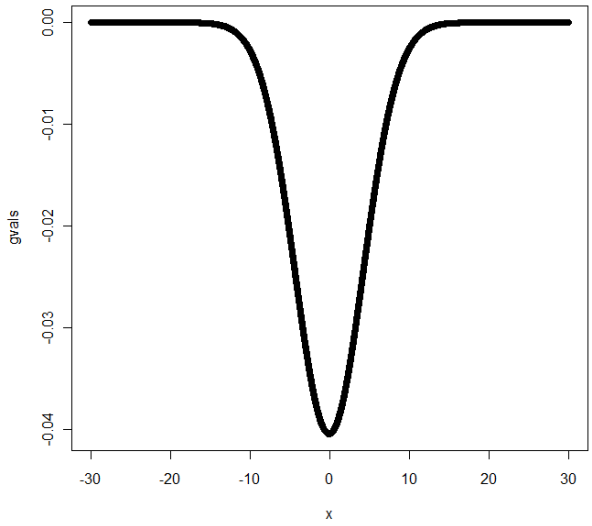


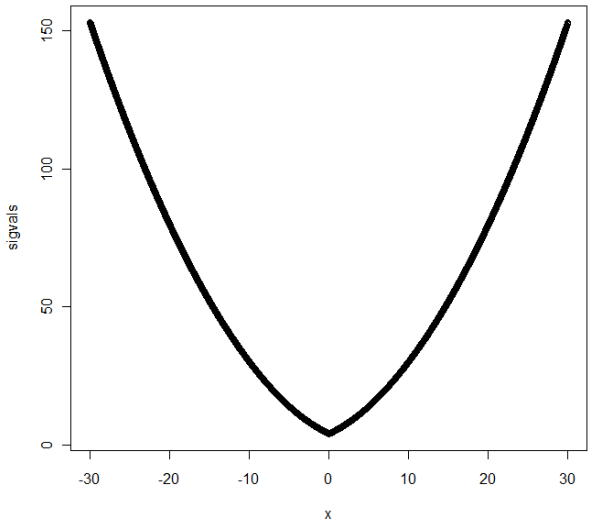
Accepted 22330 out of 100000 (22.330%).

final beta = 1.633162
alpha(0.000000) is 0.202263
alpha(2.000000) is 0.267583
alpha(5.000000) is 0.311616
alpha(10.000000) is 0.334902
alpha(20.000000) is 0.306255
alpha(50.000000) is 0.206723
alpha(100.000000) is 0.125700
alpha(150.000000) is 0.088200
it takes 807.930000 seconds to run the adaptation algorithm
it takes 465.320000 seconds to compute g and sigma
it takes 168.840000 seconds to generate the first Markov Chain
it takes 168.350000 seconds to generate the second Markov Chain
it takes 168.410000 seconds to generate the third Markov Chain
it takes 168.490000 seconds to generate the fourth Markov Chain
it takes 168.340000 seconds to generate the fifth Markov Chain
it takes 1450.812704 seconds to test the local acceptance

```
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008917038
> asjd(xlist)
[1] 0.4945916
>
> source("xlist2")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.612522
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.009193089
> asjd(xlist)
[1] 0.4812469
>
> source("xlist3")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.921151
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.009351136
> asjd(xlist)
[1] 0.4784871
>
> source("xlist4")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.508852
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.009159896
> asjd(xlist)
[1] 0.4711793
>
> source("xlist5")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.411572
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.009003102
> asjd(xlist)
[1] 0.4769998
```

2.1.1.16 case 16 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 10$

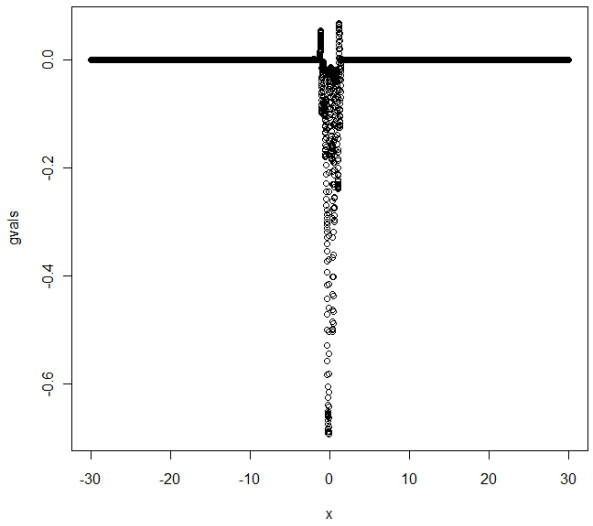


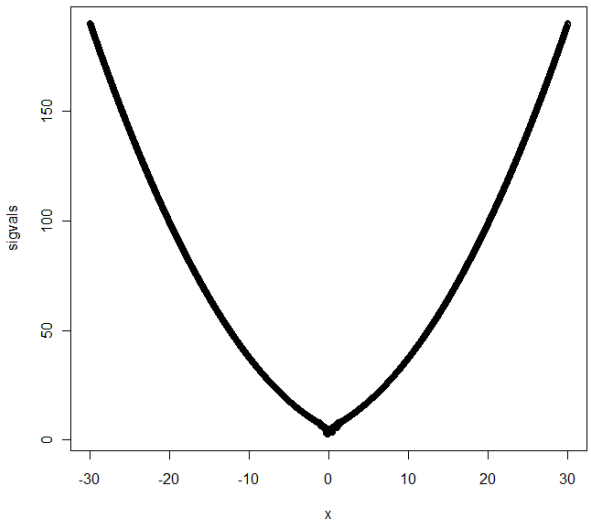


Accepted 23891 out of 100000 (23.891%).
final beta = 1.447071
alpha(0.000000) is 0.212424
alpha(2.000000) is 0.299157
alpha(5.000000) is 0.375196
alpha(10.000000) is 0.394662
alpha(20.000000) is 0.350472
alpha(50.000000) is 0.243851
alpha(100.000000) is 0.150100
alpha(150.000000) is 0.109600
it takes 436.900000 seconds to run the adaptation algorithm
it takes 200.010000 seconds to compute g and sigma
it takes 168.410000 seconds to generate the first Markov Chain
it takes 168.390000 seconds to generate the second Markov Chain
it takes 168.310000 seconds to generate the third Markov Chain
it takes 168.250000 seconds to generate the fourth Markov Chain
it takes 168.680000 seconds to generate the fifth Markov Chain
it takes -1118.967296 seconds to test the local acceptance

```
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008132099
> asjd(xlist)
[1] 0.5293508
>
> source("xlist4")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 6.549738
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.00844165
> asjd(xlist)
[1] 0.5197645
>
> source("xlist5")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 6.704092
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008546772
> asjd(xlist)
[1] 0.5158363
```

2.1.1.17 case 17 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 0.1$



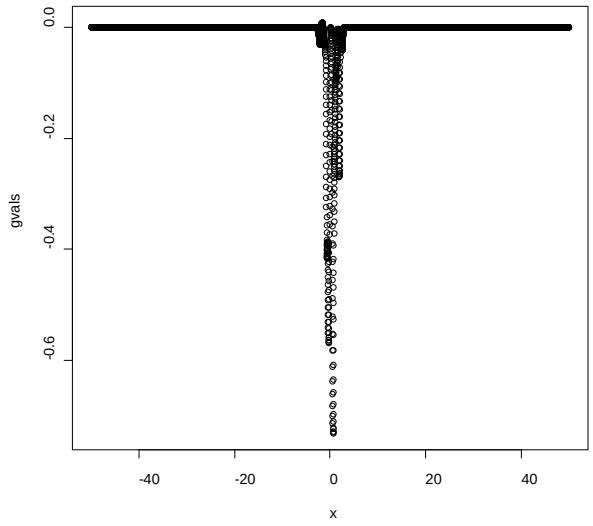


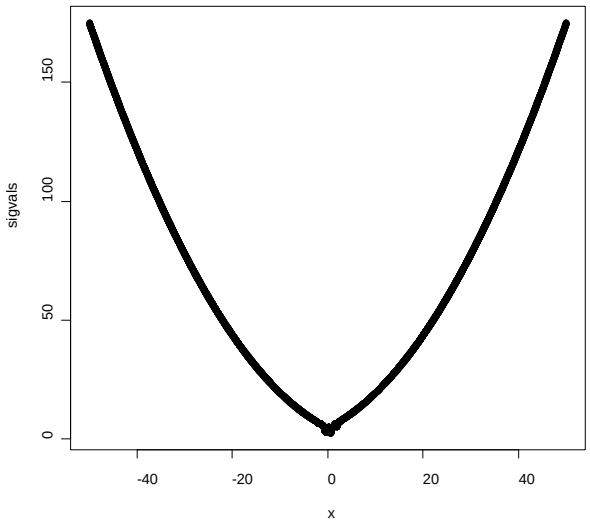
Accepted 21377 out of 100000 (21.377%).
final beta = 1.664225

alpha(0.000000) is 0.180821
alpha(2.000000) is 0.232952
alpha(5.000000) is 0.308967
alpha(10.000000) is 0.331137
alpha(20.000000) is 0.304751
alpha(50.000000) is 0.198707
alpha(100.000000) is 0.112710
alpha(150.000000) is 0.084200
it takes 557.440000 seconds to run the adaptation algorithm
it takes 277.870000 seconds to compute g and sigma
it takes 168.700000 seconds to generate the first Markov Chain
it takes 168.520000 seconds to generate the second Markov Chain
it takes 168.430000 seconds to generate the third Markov Chain
it takes 168.420000 seconds to generate the fourth Markov Chain
it takes 168.310000 seconds to generate the fifth Markov Chain
it takes -559.027296 seconds to test the local acceptance

```
> asjd(xlist)
[1] 0.4587334
>
> source("xlist2")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.725935
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.009216995
> asjd(xlist)
[1] 0.4531863
>
> source("xlist3")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 8.350977
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.009555587
> asjd(xlist)
[1] 0.4569404
>
> source("xlist4")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.279248
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008898774
> asjd(xlist)
[1] 0.4481364
>
> source("xlist5")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.577468
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.009148531
> asjd(xlist)
[1] 0.4577722
```

2.1.1.18 case 18 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$





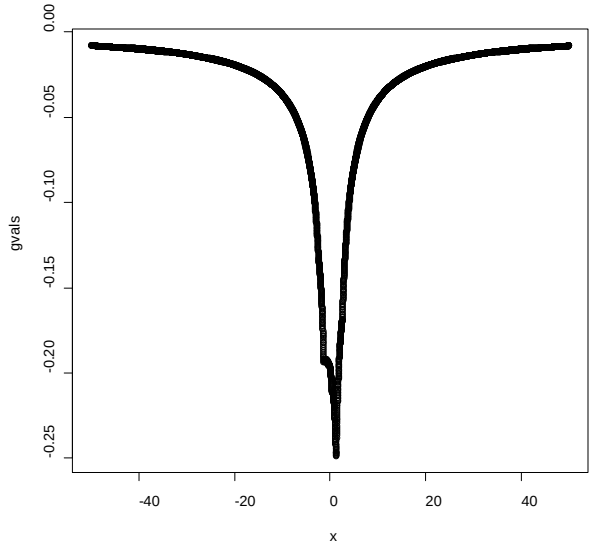
Accepted 23550 out of 100000 (23.550%).
final beta = 1.581161
alpha(0.000000) is 0.205731
alpha(2.000000) is 0.300010
alpha(5.000000) is 0.330864
alpha(10.000000) is 0.353506
alpha(20.000000) is 0.317601
alpha(50.000000) is 0.211646
alpha(100.000000) is 0.136600
alpha(150.000000) is 0.089400
it takes 248.510000 seconds to run the adaptation algorithm
it takes 155.567000 seconds to compute g and sigma
it takes 18.628000 seconds to generate the first Markov Chain
it takes 19.038000 seconds to generate the second Markov Chain
it takes 19.105000 seconds to generate the third Markov Chain
it takes 18.861000 seconds to generate the fourth Markov Chain
it takes 18.436000 seconds to generate the fifth Markov Chain
it takes 1793.310000 seconds to test the local acceptance

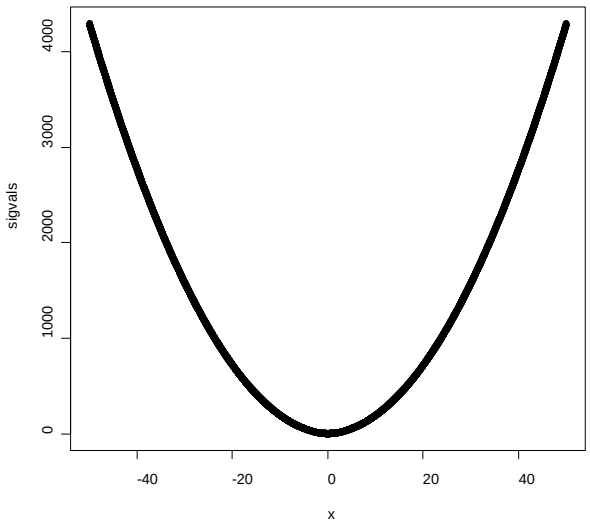
```
[1] 7.525139
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.009061946
> asjd(xlist)
[1] 0.4880344
>
> source("xlist4")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.189149
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008921259
> asjd(xlist)
[1] 0.4885126
>
> source("xlist5")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 7.331579
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008833732
> asjd(xlist)
[1] 0.4807859
```

>

2.1.2 kernel function $K(x)=\frac{1}{1+\alpha_1 |x|^{\alpha_2}}$

2.1.2.1 case 19 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=1,\alpha_2=1,C=1,\gamma=2$ with fixed bandwidth $b_n=1$

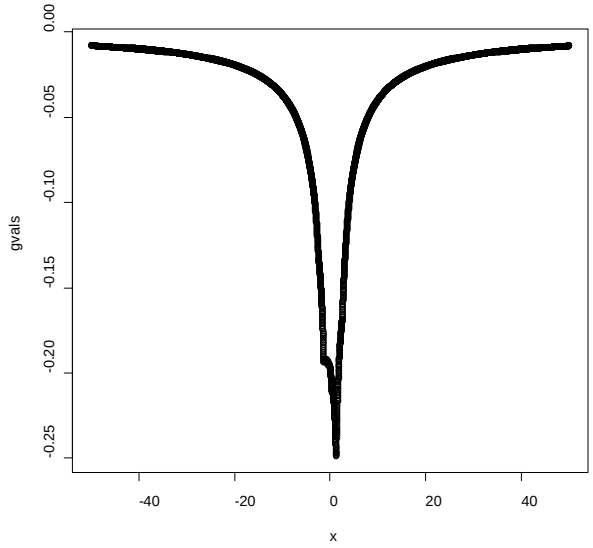


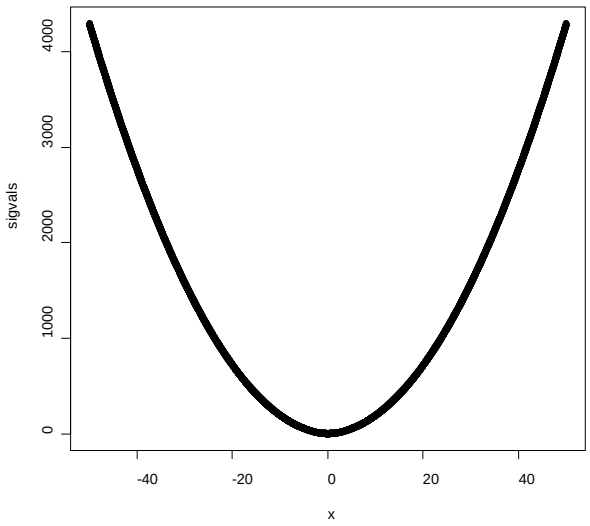


Accepted 23270 out of 100000 (23.270%).
final beta = 0.509653
alpha(0.000000) is 0.308718
alpha(2.000000) is 0.154510
alpha(5.000000) is 0.072492
alpha(10.000000) is 0.044654
alpha(20.000000) is 0.022523
alpha(50.000000) is 0.010300
alpha(100.000000) is 0.005800
alpha(150.000000) is 0.003400
it takes 182.254000 seconds to run the adaptation algorithm
it takes 79.529000 seconds to compute g and sigma
it takes 18.465000 seconds to generate the first Markov Chain
it takes 19.243000 seconds to generate the second Markov Chain
it takes 18.764000 seconds to generate the third Markov Chain
it takes 18.643000 seconds to generate the fourth Markov Chain
it takes 18.732000 seconds to generate the fifth Markov Chain
it takes 1417.510000 seconds to test the local acceptance

[1] 11.41702
[1] 0.01129308
[1] 0.3478964

2.1.2.2 case 20 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$



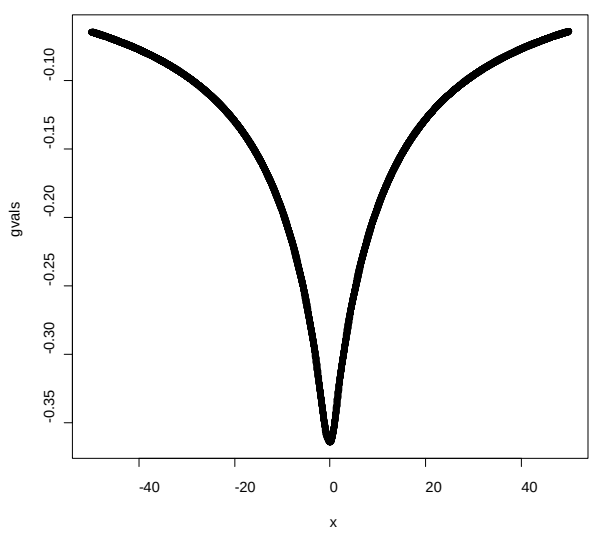


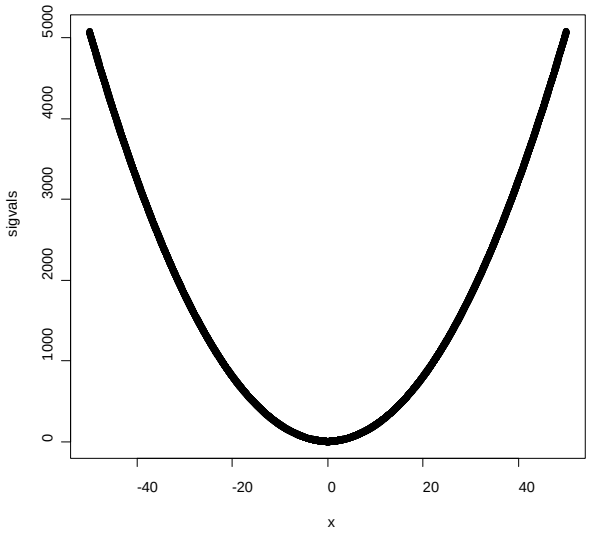
Accepted 23090 out of 100000 (23.090%).
final beta = 0.371945
alpha(0.000000) is 0.294688
alpha(2.000000) is 0.156616
alpha(5.000000) is 0.083929
alpha(10.000000) is 0.049232
alpha(20.000000) is 0.027440
alpha(50.000000) is 0.010606
alpha(100.000000) is 0.005000
alpha(150.000000) is 0.004300
it takes 182.737000 seconds to run the adaptation algorithm
it takes 79.860000 seconds to compute g and sigma
it takes 19.045000 seconds to generate the first Markov Chain
it takes 18.773000 seconds to generate the second Markov Chain
it takes 19.008000 seconds to generate the third Markov Chain
it takes 18.930000 seconds to generate the fourth Markov Chain
it takes 19.085000 seconds to generate the fifth Markov Chain
it takes 1412.770000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 11.41702
[1] 0.01129308
[1] 0.3478964

2.1.2.3 case 21 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

$b_n = 1$

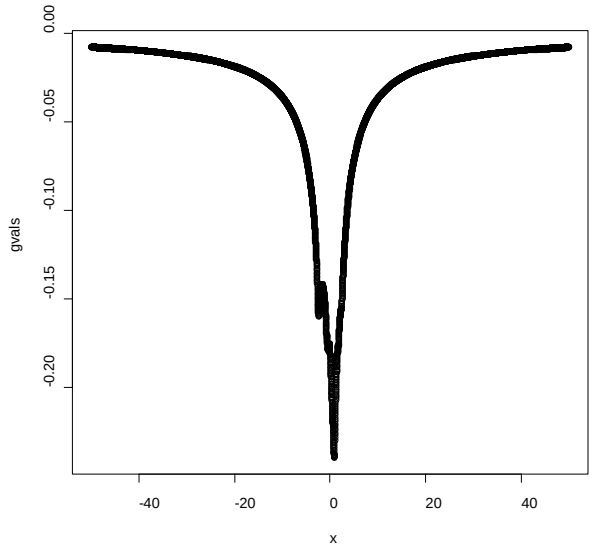


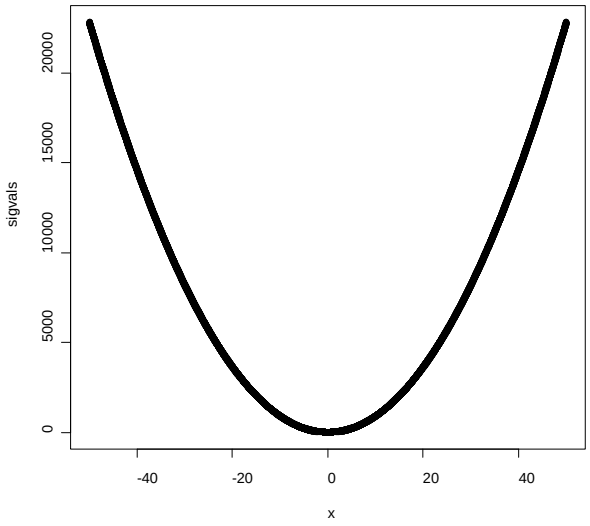


Accepted 21634 out of 100000 (21.634%).
final beta = 0.733597
alpha(0.000000) is 0.283550
alpha(2.000000) is 0.132628
alpha(5.000000) is 0.071787
alpha(10.000000) is 0.038704
alpha(20.000000) is 0.018586
alpha(50.000000) is 0.008800
alpha(100.000000) is 0.003800
alpha(150.000000) is 0.002600
it takes 183.559000 seconds to run the adaptation algorithm
it takes 79.844000 seconds to compute g and sigma
it takes 20.529000 seconds to generate the first Markov Chain
it takes 21.000000 seconds to generate the second Markov Chain
it takes 19.672000 seconds to generate the third Markov Chain
it takes 19.536000 seconds to generate the fourth Markov Chain
it takes 19.809000 seconds to generate the fifth Markov Chain
it takes 1480.876000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 11.66403
[1] 0.01155291
[1] 0.3238721

2.1.2.4 case 22 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$



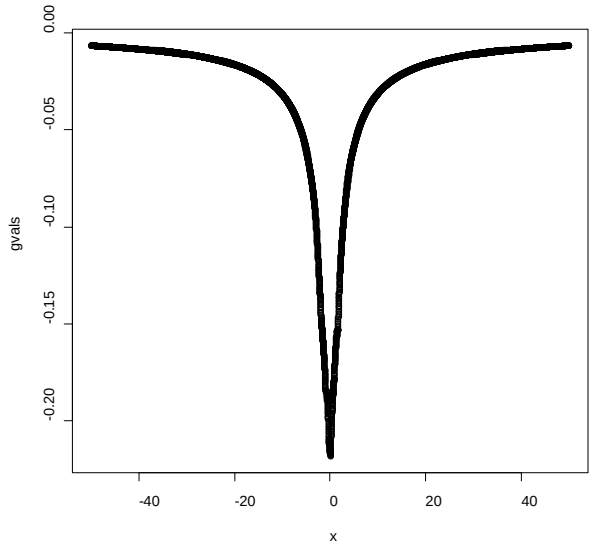


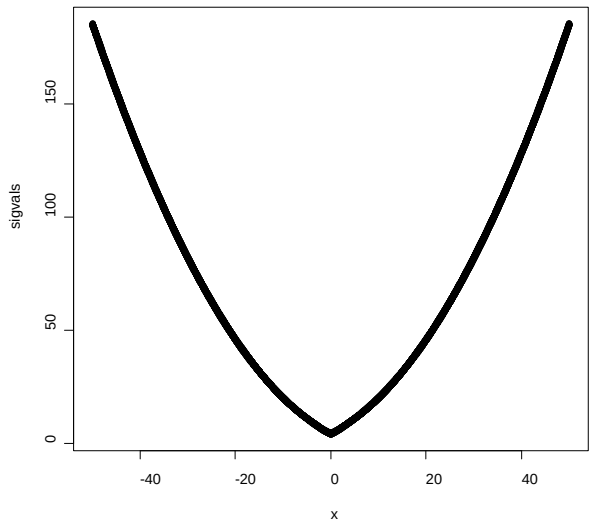
Accepted 23550 out of 100000 (23.550%).
final beta = -2.389302
alpha(0.000000) is 0.692987
alpha(2.000000) is 0.049448
alpha(5.000000) is 0.018691
alpha(10.000000) is 0.009739
alpha(20.000000) is 0.004700
alpha(50.000000) is 0.001200
alpha(100.000000) is 0.000700
alpha(150.000000) is 0.000400
it takes 189.708000 seconds to run the adaptation algorithm
it takes 82.037000 seconds to compute g and sigma
it takes 33.573000 seconds to generate the first Markov Chain
it takes 34.799000 seconds to generate the second Markov Chain
it takes 33.047000 seconds to generate the third Markov Chain
it takes 37.563000 seconds to generate the fourth Markov Chain
it takes 32.954000 seconds to generate the fifth Markov Chain
it takes 1589.008000 seconds to test the local acceptance

[1] 25.62197
[1] 0.01656713
[1] 0.1643911

2.1.2.5 case 23 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 1$



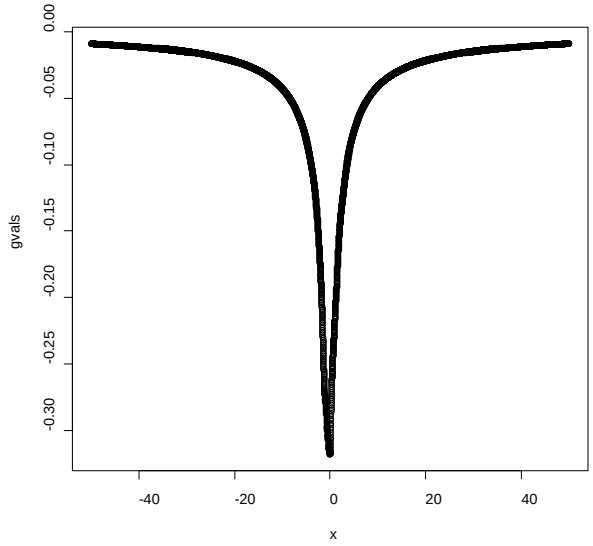


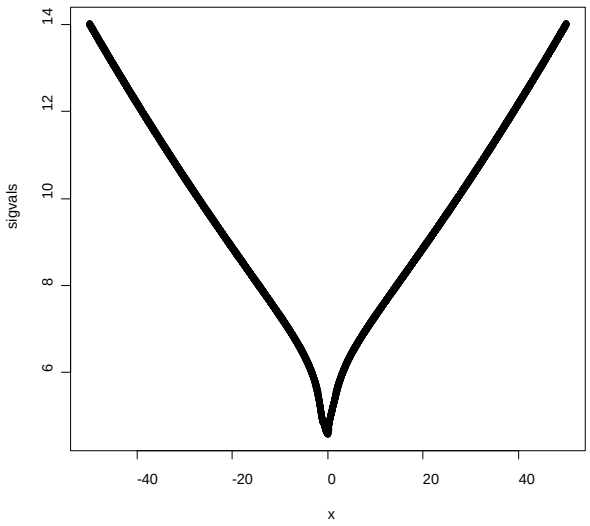
Accepted 22407 out of 100000 (22.407%).
final beta = 1.644476
alpha(0.000000) is 0.194444
alpha(2.000000) is 0.275256
alpha(5.000000) is 0.325757
alpha(10.000000) is 0.347718
alpha(20.000000) is 0.304018
alpha(50.000000) is 0.199149
alpha(100.000000) is 0.123500
alpha(150.000000) is 0.093000
it takes 185.075000 seconds to run the adaptation algorithm
it takes 81.005000 seconds to compute g and sigma
it takes 18.974000 seconds to generate the first Markov Chain
it takes 19.055000 seconds to generate the second Markov Chain
it takes 19.158000 seconds to generate the third Markov Chain
it takes 19.246000 seconds to generate the fourth Markov Chain
it takes 19.145000 seconds to generate the fifth Markov Chain
it takes 1080.022000 seconds to test the local acceptance

[1] 6.847748
[1] 0.008521907
[1] 0.4765741

2.1.2.6 case 24 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.01, \gamma = 2$ with fixed bandwidth

$b_n = 1$

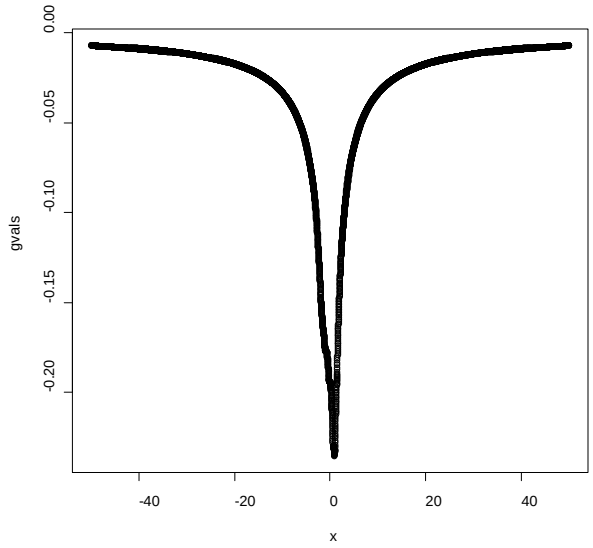


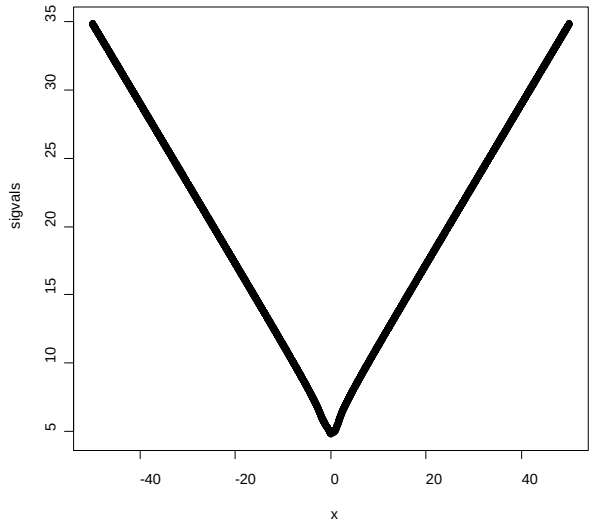


Accepted 23657 out of 100000 (23.657%).
final beta = 1.838878
alpha(0.000000) is 0.200781
alpha(2.000000) is 0.309067
alpha(5.000000) is 0.451643
alpha(10.000000) is 0.498835
alpha(20.000000) is 0.495384
alpha(50.000000) is 0.492733
alpha(100.000000) is 0.499651
alpha(150.000000) is 0.505605
it takes 185.098000 seconds to run the adaptation algorithm
it takes 81.129000 seconds to compute g and sigma
it takes 19.323000 seconds to generate the first Markov Chain
it takes 19.333000 seconds to generate the second Markov Chain
it takes 19.919000 seconds to generate the third Markov Chain
it takes 19.601000 seconds to generate the fourth Markov Chain
it takes 19.271000 seconds to generate the fifth Markov Chain
it takes 822.868000 seconds to test the local acceptance

[1] 6.463722
[1] 0.008421049
[1] 0.5429365

2.1.2.7 case 25 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 1$ with fixed bandwidth $b_n = 1$



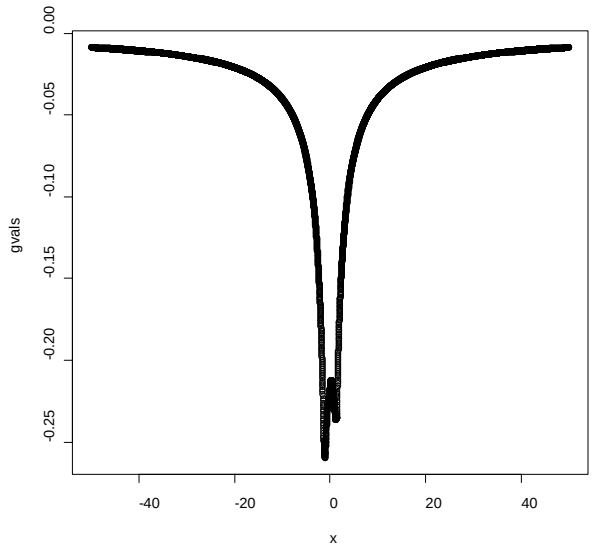


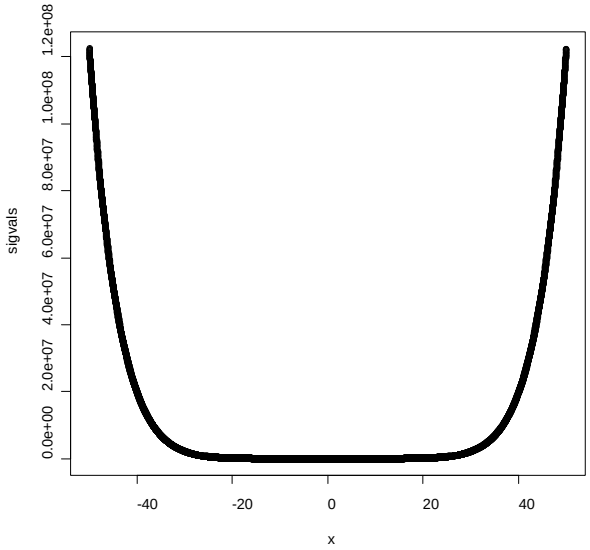
Accepted 22607 out of 100000 (22.607%).
final beta = 1.766949
alpha(0.000000) is 0.193104
alpha(2.000000) is 0.291302
alpha(5.000000) is 0.412738
alpha(10.000000) is 0.463710
alpha(20.000000) is 0.499346
alpha(50.000000) is 0.496252
alpha(100.000000) is 0.493020
alpha(150.000000) is 0.499200
it takes 188.056000 seconds to run the adaptation algorithm
it takes 81.384000 seconds to compute g and sigma
it takes 19.928000 seconds to generate the first Markov Chain
it takes 20.102000 seconds to generate the second Markov Chain
it takes 19.930000 seconds to generate the third Markov Chain
it takes 19.622000 seconds to generate the fourth Markov Chain
it takes 19.886000 seconds to generate the fifth Markov Chain
it takes 810.732000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 7.156546
[1] 0.009006958
[1] 0.5071021

2.1.2.8 case 26 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 10$ with fixed bandwidth

$b_n = 1$



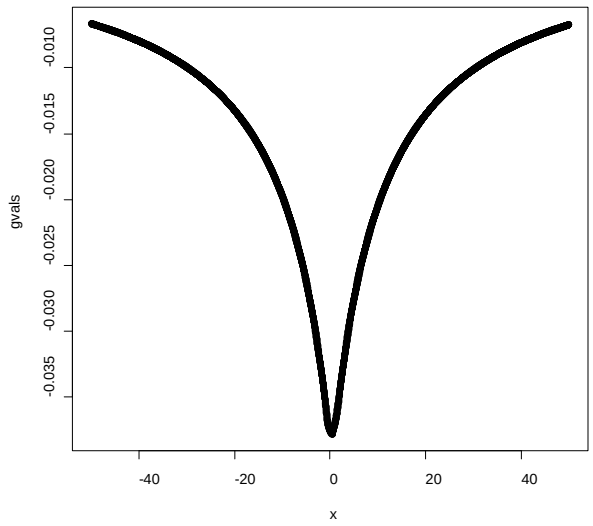


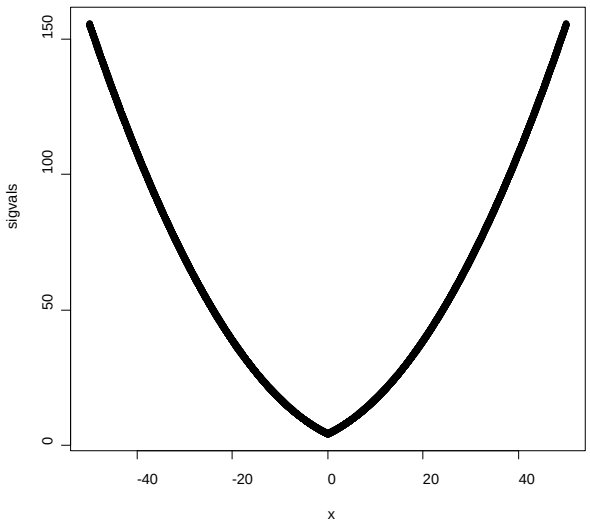
Accepted 27588 out of 100000 (27.588%).
final beta = 0.714842
alpha(0.000000) is 0.323469
alpha(2.000000) is 0.179157
alpha(5.000000) is 0.038691
alpha(10.000000) is 0.004204
alpha(20.000000) is 0.000200
alpha(50.000000) is 0.000000
alpha(100.000000) is 0.000000
alpha(150.000000) is 0.000000
it takes 185.402000 seconds to run the adaptation algorithm
it takes 80.801000 seconds to compute g and sigma
it takes 20.724000 seconds to generate the first Markov Chain
it takes 28.117000 seconds to generate the second Markov Chain
it takes 20.586000 seconds to generate the third Markov Chain
it takes 20.432000 seconds to generate the fourth Markov Chain
it takes 20.293000 seconds to generate the fifth Markov Chain
it takes 1514.924000 seconds to test the local acceptance

[1] 10.26327
[1] 0.01055025
[1] 0.3960828

2.1.2.9 case 27 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 10$



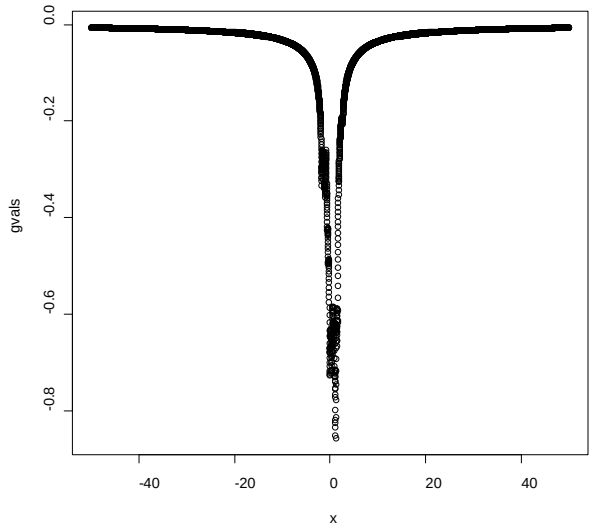


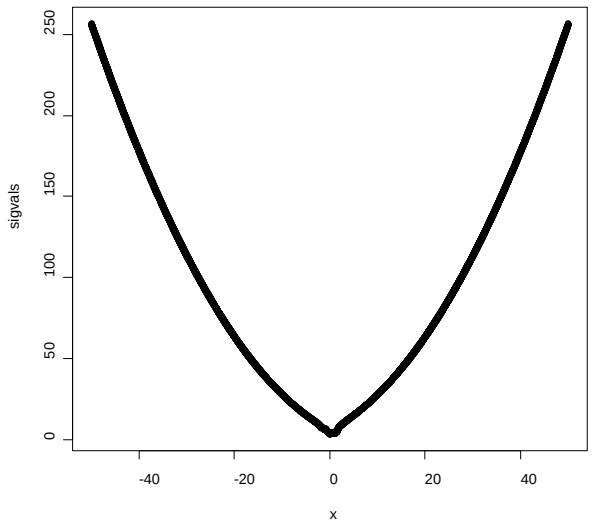
Accepted 23321 out of 100000 (23.321%).
final beta = 1.470430
alpha(0.000000) is 0.201310
alpha(2.000000) is 0.301568
alpha(5.000000) is 0.366086
alpha(10.000000) is 0.380836
alpha(20.000000) is 0.352299
alpha(50.000000) is 0.239894
alpha(100.000000) is 0.155501
alpha(150.000000) is 0.105800
it takes 185.475000 seconds to run the adaptation algorithm
it takes 82.646000 seconds to compute g and sigma
it takes 18.919000 seconds to generate the first Markov Chain
it takes 18.652000 seconds to generate the second Markov Chain
it takes 18.690000 seconds to generate the third Markov Chain
it takes 18.640000 seconds to generate the fourth Markov Chain
it takes 18.789000 seconds to generate the fifth Markov Chain
it takes 1089.644000 seconds to test the local acceptance

[1] 7.028814
[1] 0.008970417
[1] 0.5260427

2.1.2.10 case 28 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 0.1$



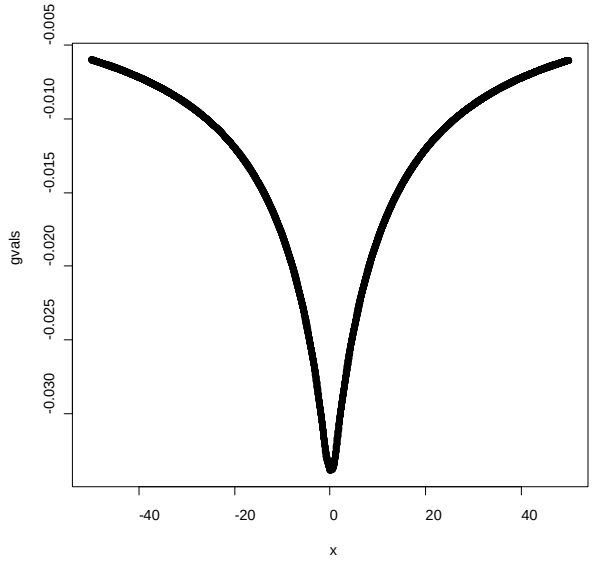


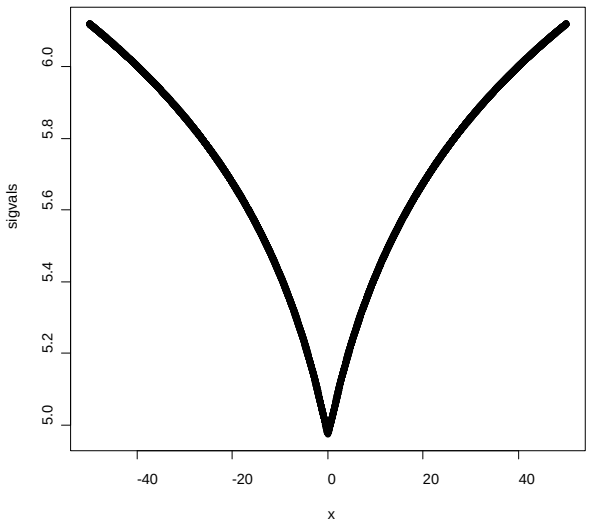
Accepted 21876 out of 100000 (21.876%).
final beta = 1.970691
alpha(0.000000) is 0.202246
alpha(2.000000) is 0.235476
alpha(5.000000) is 0.255892
alpha(10.000000) is 0.268970
alpha(20.000000) is 0.234585
alpha(50.000000) is 0.152896
alpha(100.000000) is 0.091300
alpha(150.000000) is 0.063500
it takes 195.322000 seconds to run the adaptation algorithm
it takes 86.348000 seconds to compute g and sigma
it takes 25.162000 seconds to generate the first Markov Chain
it takes 25.307000 seconds to generate the second Markov Chain
it takes 25.162000 seconds to generate the third Markov Chain
it takes 24.111000 seconds to generate the fourth Markov Chain
it takes 22.829000 seconds to generate the fifth Markov Chain
it takes 1155.381000 seconds to test the local acceptance

[1] 8.145908
[1] 0.009549854
[1] 0.4514627

2.1.2.11 case 29 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth

$b_n = 10$

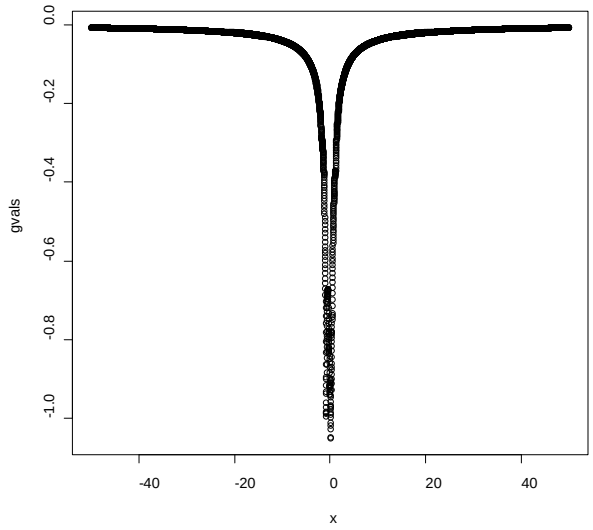


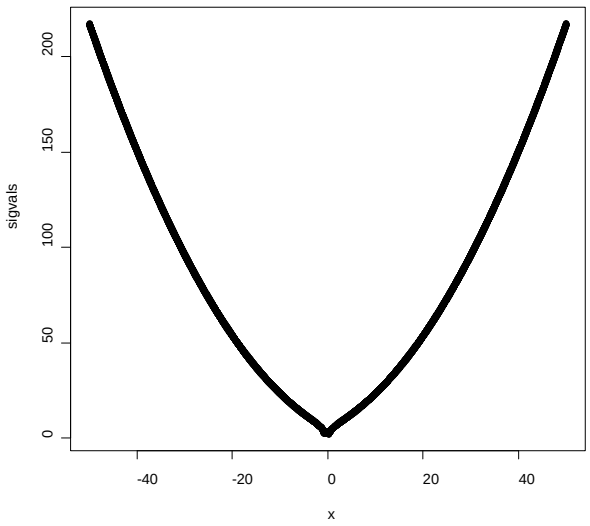


Accepted 24039 out of 100000 (24.039%).
final beta = 1.638337
alpha(0.000000) is 0.194215
alpha(2.000000) is 0.331993
alpha(5.000000) is 0.488551
alpha(10.000000) is 0.496526
alpha(20.000000) is 0.499359
alpha(50.000000) is 0.505963
alpha(100.000000) is 0.508336
alpha(150.000000) is 0.507511
it takes 186.879000 seconds to run the adaptation algorithm
it takes 83.355000 seconds to compute g and sigma
it takes 20.056000 seconds to generate the first Markov Chain
it takes 19.361000 seconds to generate the second Markov Chain
it takes 19.535000 seconds to generate the third Markov Chain
it takes 19.817000 seconds to generate the fourth Markov Chain
it takes 19.696000 seconds to generate the fifth Markov Chain
it takes 858.514000 seconds to test the local acceptance

[1] 6.165894
[1] 0.008328171
[1] 0.5697796

2.1.2.12 case 30 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$



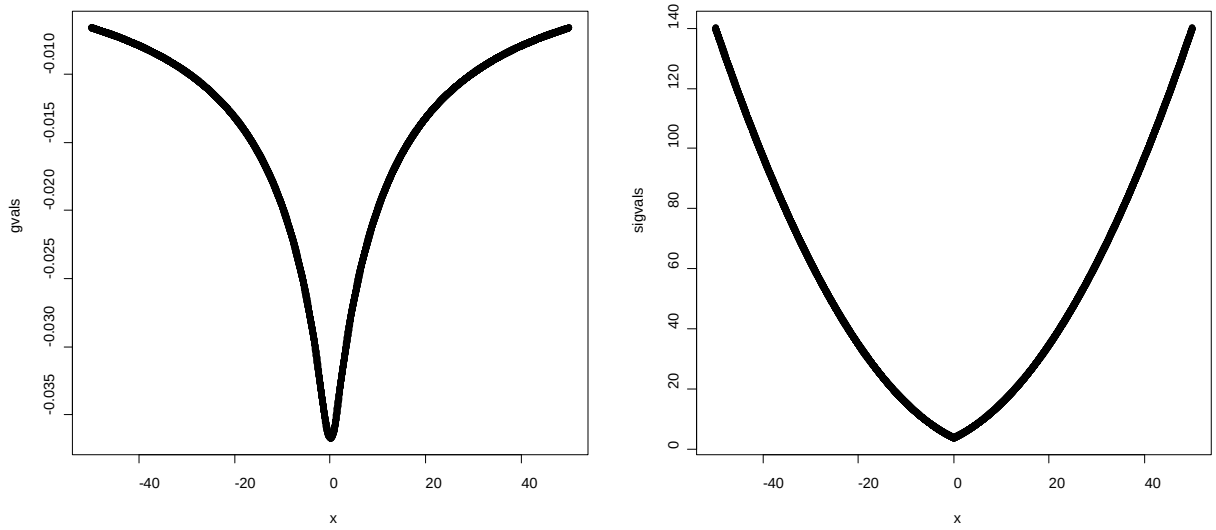


Accepted 26457 out of 100000 (26.457%).
final beta = 1.805474
alpha(0.000000) is 0.269674
alpha(2.000000) is 0.249986
alpha(5.000000) is 0.293759
alpha(10.000000) is 0.307941
alpha(20.000000) is 0.279936
alpha(50.000000) is 0.176204
alpha(100.000000) is 0.109100
alpha(150.000000) is 0.075300
it takes 194.183000 seconds to run the adaptation algorithm
it takes 86.661000 seconds to compute g and sigma
it takes 24.157000 seconds to generate the first Markov Chain
it takes 23.538000 seconds to generate the second Markov Chain
it takes 23.525000 seconds to generate the third Markov Chain
it takes 23.575000 seconds to generate the fourth Markov Chain
it takes 23.875000 seconds to generate the fifth Markov Chain
it takes 1177.590000 seconds to test the local acceptance

[1] 6.940457
[1] 0.008799531
[1] 0.4994858

2.1.2.13 case 31 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 10$

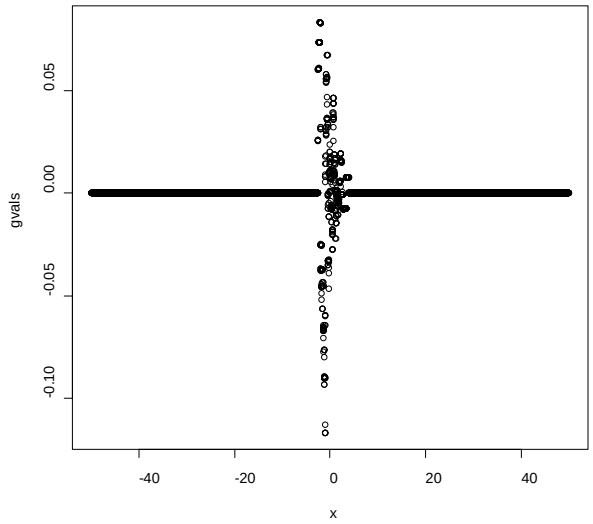


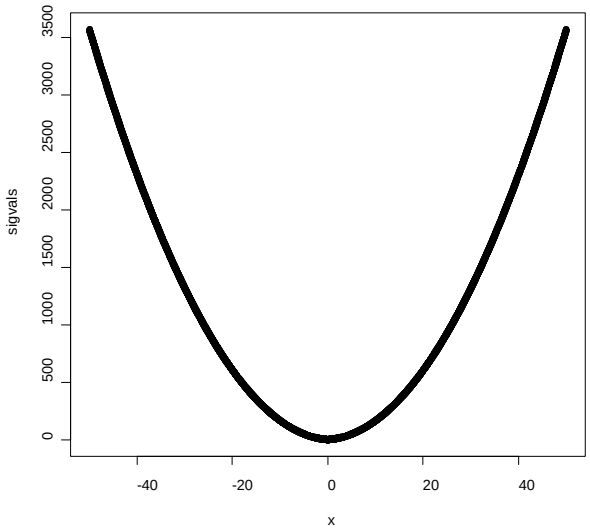
Accepted 25724 out of 100000 (25.724%).
final beta = 1.365579
alpha(0.000000) is 0.221199
alpha(2.000000) is 0.320933
alpha(5.000000) is 0.391188
alpha(10.000000) is 0.401987
alpha(20.000000) is 0.378950
alpha(50.000000) is 0.264811
alpha(100.000000) is 0.163500
alpha(150.000000) is 0.120201
it takes 195.386000 seconds to run the adaptation algorithm
it takes 87.578000 seconds to compute g and sigma
it takes 24.496000 seconds to generate the first Markov Chain
it takes 24.224000 seconds to generate the second Markov Chain
it takes 24.354000 seconds to generate the third Markov Chain
it takes 24.327000 seconds to generate the fourth Markov Chain
it takes 24.485000 seconds to generate the fifth Markov Chain
it takes 1079.685000 seconds to test the local acceptance

[1] 6.246366
[1] 0.008281501
[1] 0.5653669

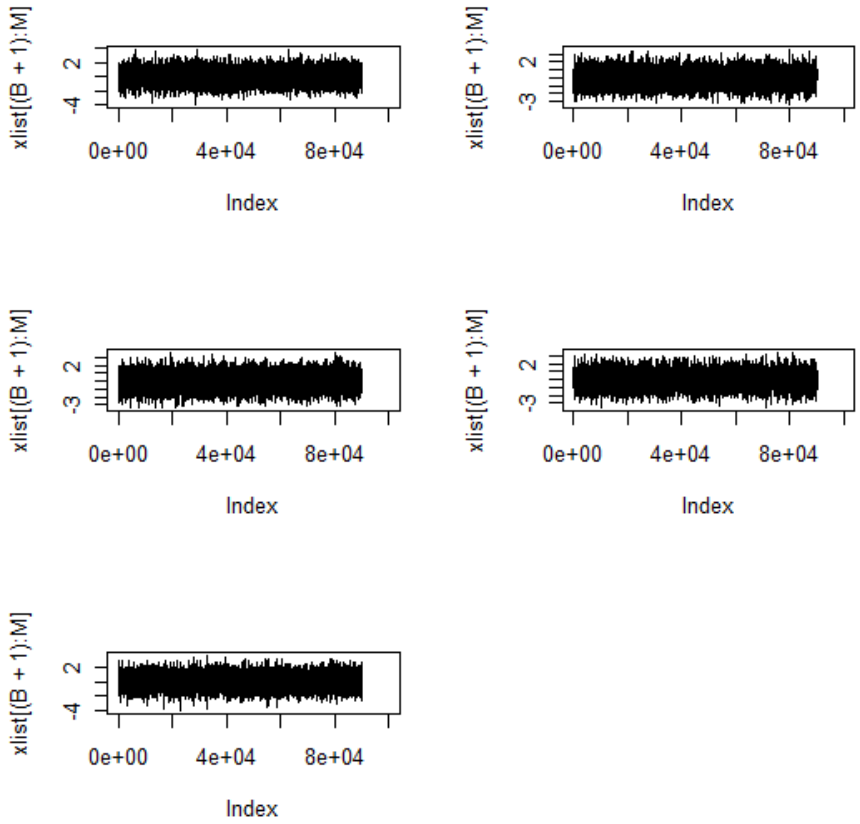
2.1.3 kernel function $K(x)=\begin{cases}0,\mid x\mid \geq 2*width\\ -1*hegith, width<\mid x\mid <2*width\\ 1*height,\mid x\mid \leq width\end{cases}$

2.1.3.1 case 32 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $height=0.5, \text{width}=0.5, C=1, \gamma=2$ with fixed bandwidth $b_n=1$





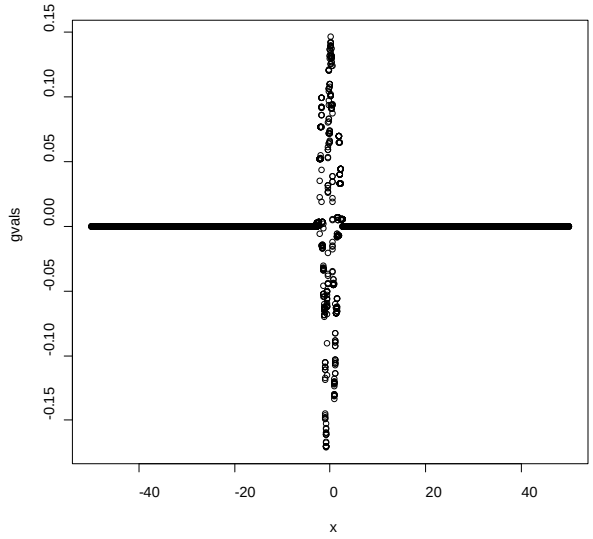
Accepted 23241 out of 100000 (23.241%).
final beta = 0.317424
alpha(0.000000) is 0.294893
alpha(2.000000) is 0.148025
alpha(5.000000) is 0.085389
alpha(10.000000) is 0.049931
alpha(20.000000) is 0.026705
alpha(50.000000) is 0.012373
alpha(100.000000) is 0.006300
alpha(150.000000) is 0.003200
it takes 124.035000 seconds to run the adaptation algorithm
it takes 54.897000 seconds to compute g and sigma
it takes 18.362000 seconds to generate the first Markov Chain
it takes 18.330000 seconds to generate the second Markov Chain
it takes 18.314000 seconds to generate the third Markov Chain
it takes 18.892000 seconds to generate the fourth Markov Chain
it takes 18.439000 seconds to generate the fifth Markov Chain
it takes 972.911000 seconds to test the local acceptance

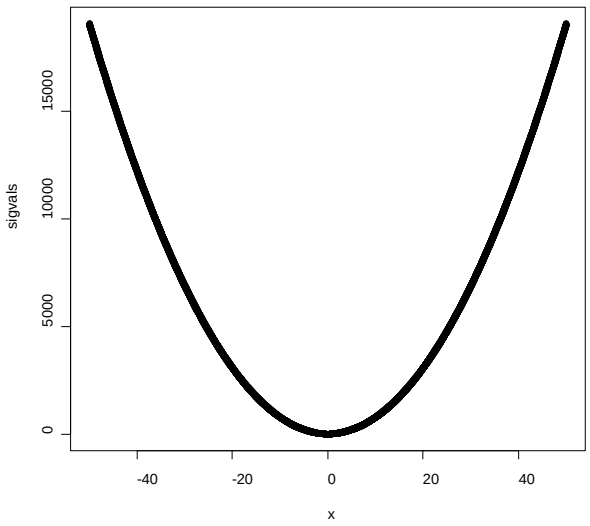



```
>    source("xlist1");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 11.21697
[1] 0.01119531
[1] 0.3480791
>    source("xlist2");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 11.01346
[1] 0.01102779
[1] 0.3393188
>    source("xlist3");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 10.73923
[1] 0.01097638
[1] 0.3521589
>    source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 11.00212
[1] 0.01111038
[1] 0.3414209
>    source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
```

[1] 10.74114
[1] 0.01082565
[1] 0.3407953

2.1.3.2 case 33 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$

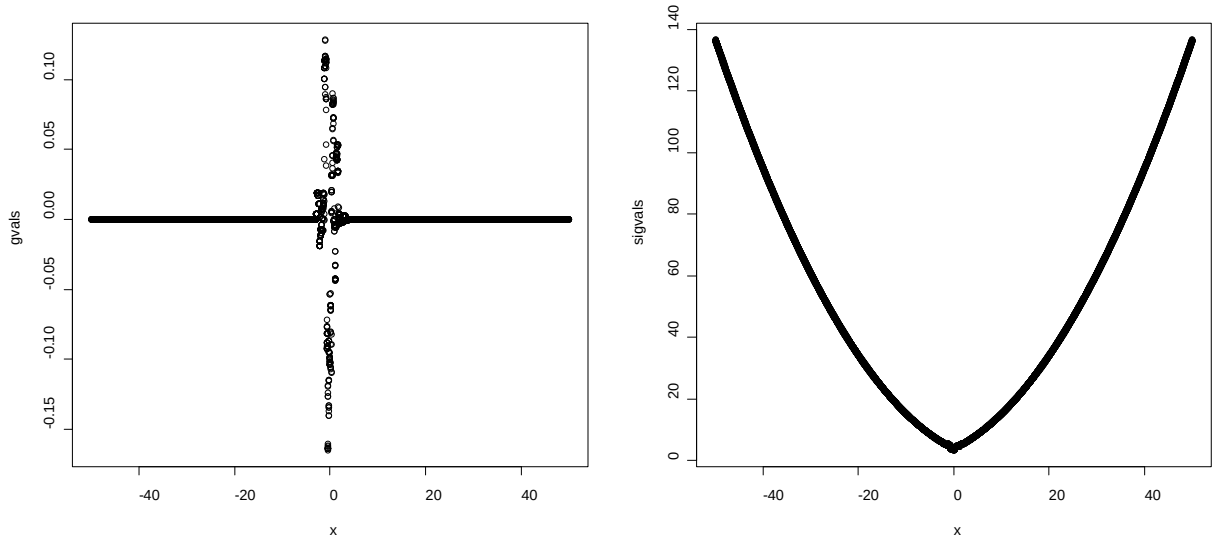




Accepted 23196 out of 100000 (23.196%).
final beta = -2.577837
alpha(0.000000) is 0.648993
alpha(2.000000) is 0.050936
alpha(5.000000) is 0.019636
alpha(10.000000) is 0.012347
alpha(20.000000) is 0.004800
alpha(50.000000) is 0.002700
alpha(100.000000) is 0.001300
alpha(150.000000) is 0.000500
it takes 128.638000 seconds to run the adaptation algorithm
it takes 56.695000 seconds to compute g and sigma
it takes 29.910000 seconds to generate the first Markov Chain
it takes 30.054000 seconds to generate the second Markov Chain
it takes 28.054000 seconds to generate the third Markov Chain
it takes 29.111000 seconds to generate the fourth Markov Chain
it takes 32.262000 seconds to generate the fifth Markov Chain
it takes 1103.764000 seconds to test the local acceptance

[1] 27.30234
[1] 0.01750065
[1] 0.1734251

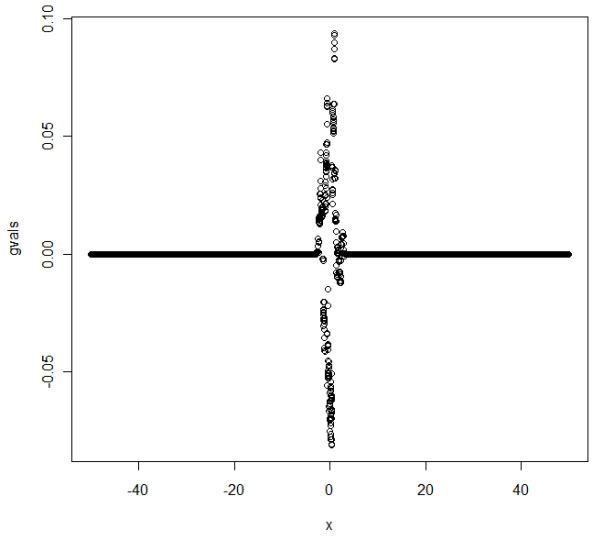
2.1.3.3 case 34 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

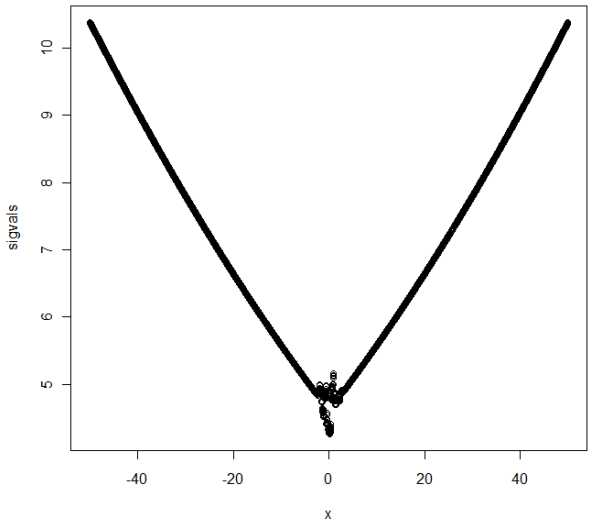


Accepted 25450 out of 100000 (25.450%).
final beta = 1.333309
alpha(0.000000) is 0.230635
alpha(2.000000) is 0.321085
alpha(5.000000) is 0.392833
alpha(10.000000) is 0.407148
alpha(20.000000) is 0.374904
alpha(50.000000) is 0.263909
alpha(100.000000) is 0.164500
alpha(150.000000) is 0.118700
it takes 129.268000 seconds to run the adaptation algorithm
it takes 56.751000 seconds to compute g and sigma
it takes 19.518000 seconds to generate the first Markov Chain
it takes 19.386000 seconds to generate the second Markov Chain
it takes 19.124000 seconds to generate the third Markov Chain
it takes 19.121000 seconds to generate the fourth Markov Chain
it takes 19.110000 seconds to generate the fifth Markov Chain
it takes 724.909000 seconds to test the local acceptance

[1] 6.348009
[1] 0.008311205
[1] 0.5478521
> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 6.251577
[1] 0.008308611
[1] 0.5388973

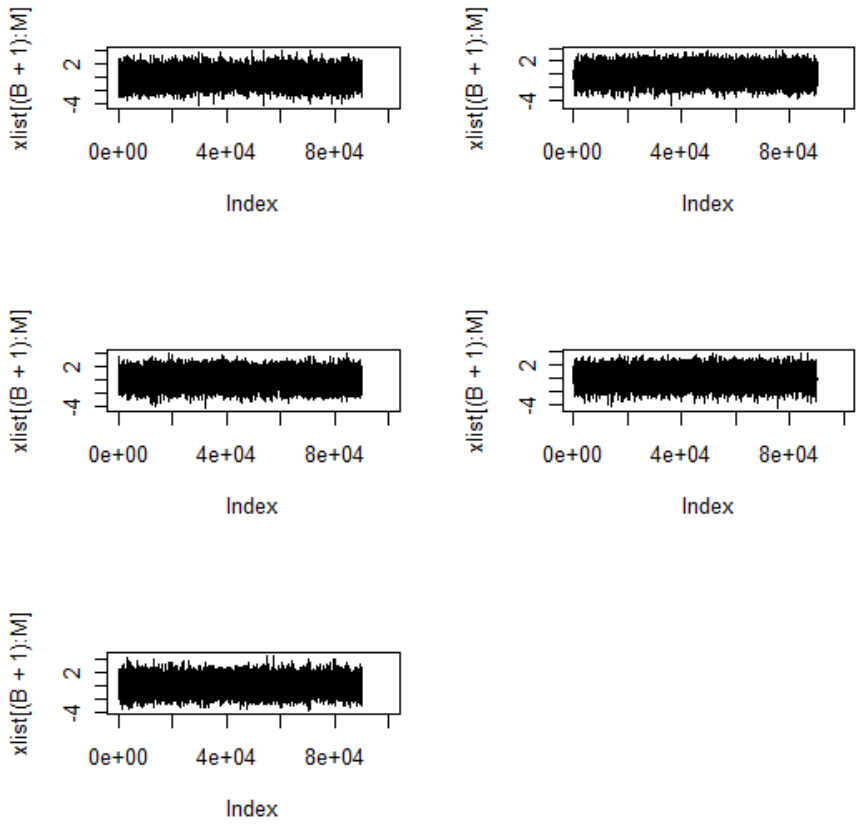
2.1.3.4 case 35 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$





Accepted 25285 out of 100000 (25.285%).
final beta = 1.528678
alpha(0.000000) is 0.211206
alpha(2.000000) is 0.355405
alpha(5.000000) is 0.499102
alpha(10.000000) is 0.505651
alpha(20.000000) is 0.502005
alpha(50.000000) is 0.502010
alpha(100.000000) is 0.497914
alpha(150.000000) is 0.508727
it takes 129.425000 seconds to run the adaptation algorithm
it takes 57.572000 seconds to compute g and sigma
it takes 19.149000 seconds to generate the first Markov Chain
it takes 18.673000 seconds to generate the second Markov Chain
it takes 19.200000 seconds to generate the third Markov Chain
it takes 18.676000 seconds to generate the fourth Markov Chain

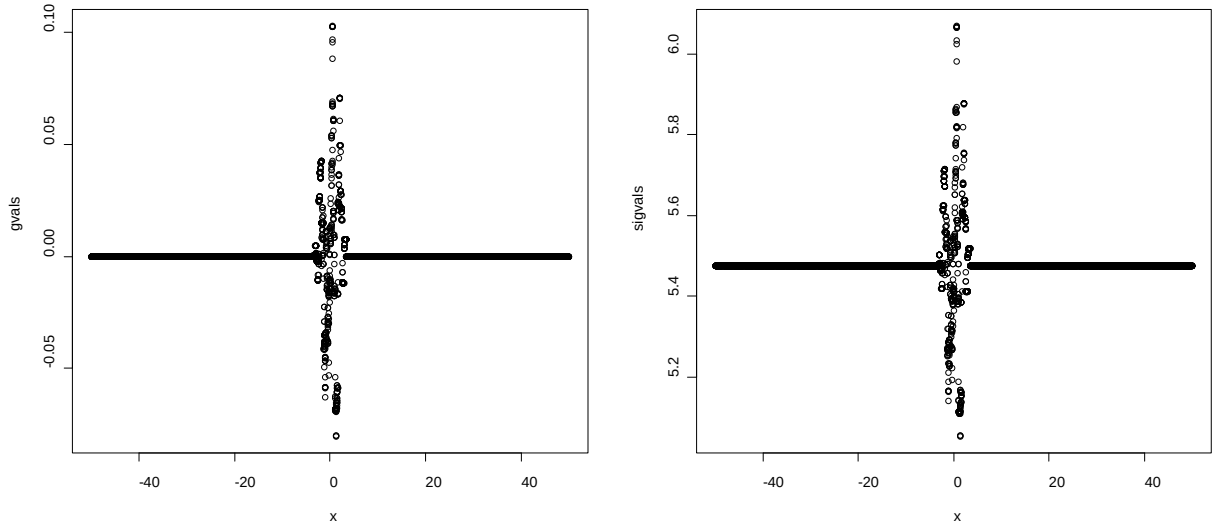
it takes 18.608000 seconds to generate the fifth Markov Chain
it takes 586.823000 seconds to test the local acceptance




```
>    source("xlist1");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 5.714605
[1] 0.007964564
[1] 0.5832376
>    source("xlist2");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 6.093879
[1] 0.008183764
[1] 0.5765046
>    source("xlist3");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 6.027745
[1] 0.008146571
[1] 0.5731739
>    source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 5.876299
[1] 0.008100663
[1] 0.5905011
```

> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 6.161878
[1] 0.008257061
[1] 0.5782485

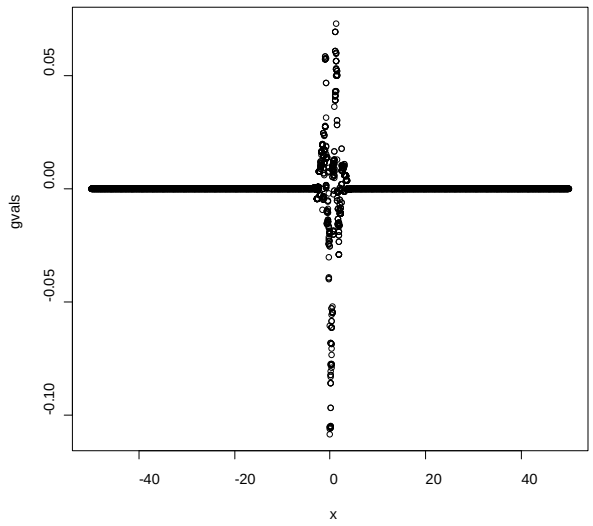
2.1.3.5 case 36 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0, \gamma = 2$ with fixed bandwidth $b_n = 1$

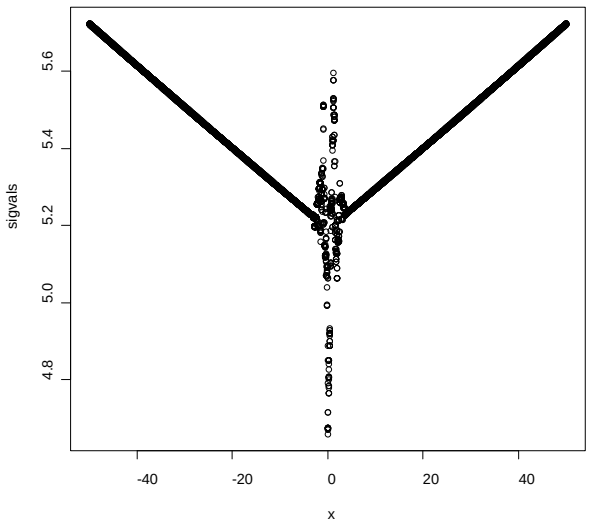


Accepted 22366 out of 100000 (22.366%).
final beta = 1.700274
alpha(0.000000) is 0.180320
alpha(2.000000) is 0.318118
alpha(5.000000) is 0.483259
alpha(10.000000) is 0.498034
alpha(20.000000) is 0.502090
alpha(50.000000) is 0.506030
alpha(100.000000) is 0.508454
alpha(150.000000) is 0.500686
it takes 127.853000 seconds to run the adaptation algorithm
it takes 56.083000 seconds to compute g and sigma
it takes 19.049000 seconds to generate the first Markov Chain
it takes 18.718000 seconds to generate the second Markov Chain
it takes 18.813000 seconds to generate the third Markov Chain
it takes 18.679000 seconds to generate the fourth Markov Chain
it takes 18.793000 seconds to generate the fifth Markov Chain
it takes 584.083000 seconds to test the local acceptance

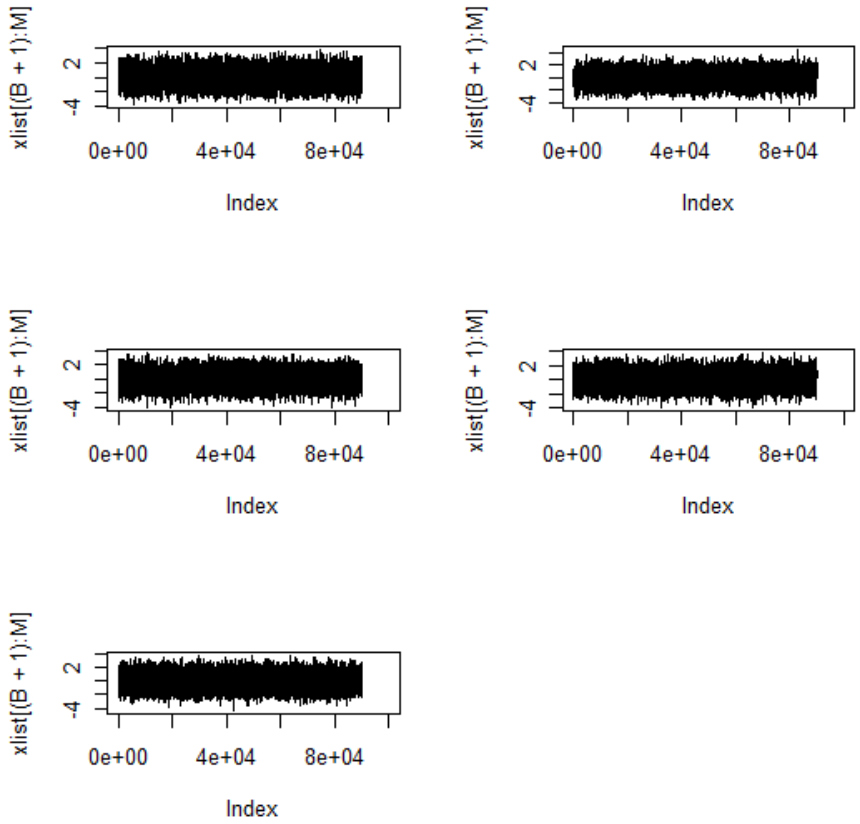
[1] 6.861094
[1] 0.008786732
[1] 0.5409318

2.1.3.6 case 37 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.001, \gamma = 2$ with fixed bandwidth $b_n = 1$





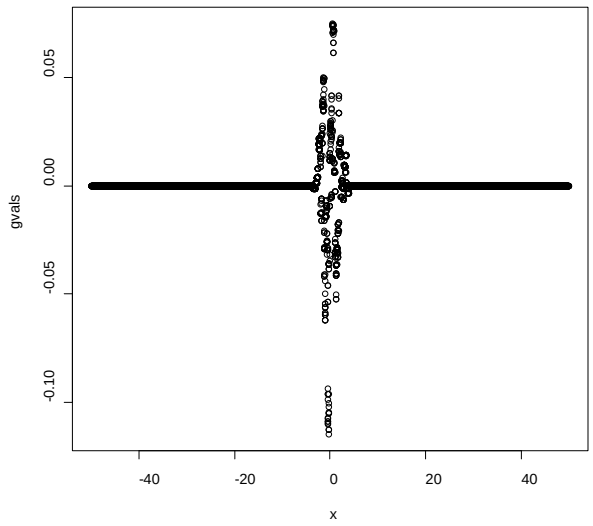
Accepted 23124 out of 100000 (23.124%).
final beta = 1.646878
alpha(0.000000) is 0.194871
alpha(2.000000) is 0.334081
alpha(5.000000) is 0.494616
alpha(10.000000) is 0.501907
alpha(20.000000) is 0.503077
alpha(50.000000) is 0.499887
alpha(100.000000) is 0.495402
alpha(150.000000) is 0.496685
it takes 127.753000 seconds to run the adaptation algorithm
it takes 56.661000 seconds to compute g and sigma
it takes 19.610000 seconds to generate the first Markov Chain
it takes 19.572000 seconds to generate the second Markov Chain
it takes 19.451000 seconds to generate the third Markov Chain
it takes 20.123000 seconds to generate the fourth Markov Chain
it takes 19.345000 seconds to generate the fifth Markov Chain
it takes 589.808000 seconds to test the local acceptance

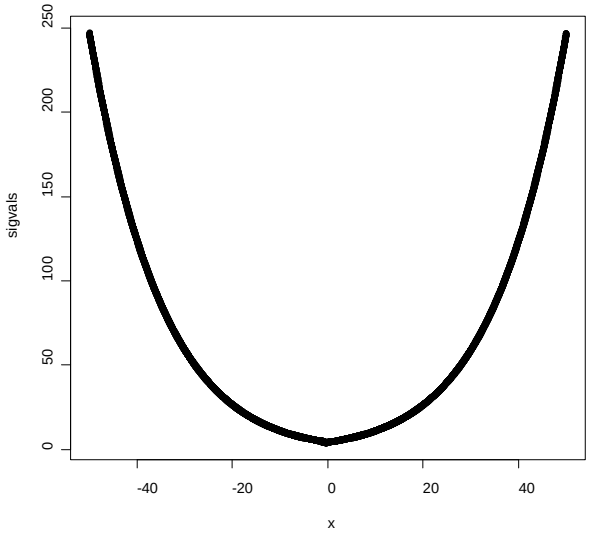



```
>    source("xlist1");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 6.480792
[1] 0.008498191
[1] 0.5554767
>    source("xlist2");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 6.665416
[1] 0.008673713
[1] 0.5492348
>    source("xlist3");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 6.420364
[1] 0.008417142
[1] 0.5508303
>    source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 6.372943
[1] 0.008327294
[1] 0.5473217
>    source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
```

[1] 6.769786
[1] 0.008624476
[1] 0.5400489

2.1.3.7 case 38 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.001, \gamma = 10$ with fixed bandwidth $b_n = 1$

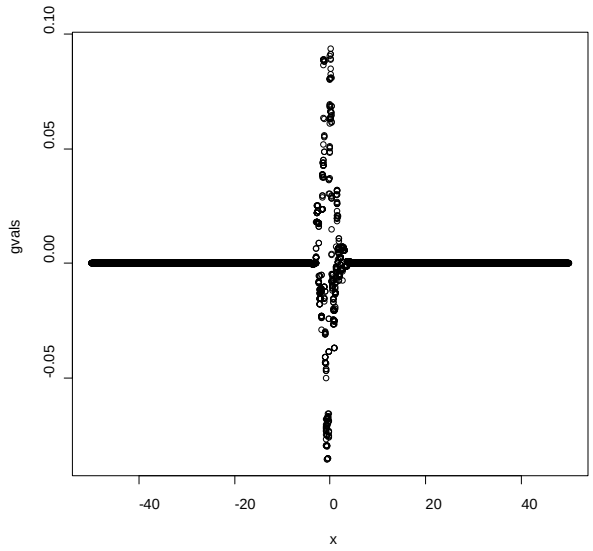


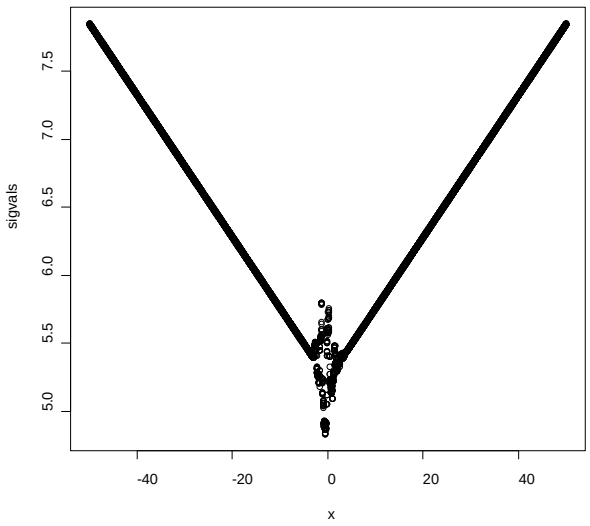


Accepted 25144 out of 100000 (25.144%).
final beta = 1.454878
alpha(0.000000) is 0.211201
alpha(2.000000) is 0.331006
alpha(5.000000) is 0.434881
alpha(10.000000) is 0.469318
alpha(20.000000) is 0.432653
alpha(50.000000) is 0.158708
alpha(100.000000) is 0.018400
alpha(150.000000) is 0.002500
it takes 131.014000 seconds to run the adaptation algorithm
it takes 58.790000 seconds to compute g and sigma
it takes 21.423000 seconds to generate the first Markov Chain
it takes 19.720000 seconds to generate the second Markov Chain
it takes 19.325000 seconds to generate the third Markov Chain
it takes 19.330000 seconds to generate the fourth Markov Chain
it takes 20.172000 seconds to generate the fifth Markov Chain
it takes 794.188000 seconds to test the local acceptance

[1] 6.5259
[1] 0.008477031
[1] 0.55597

2.1.3.8 case 39 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.001, \gamma = 1$ with fixed bandwidth $b_n = 1$

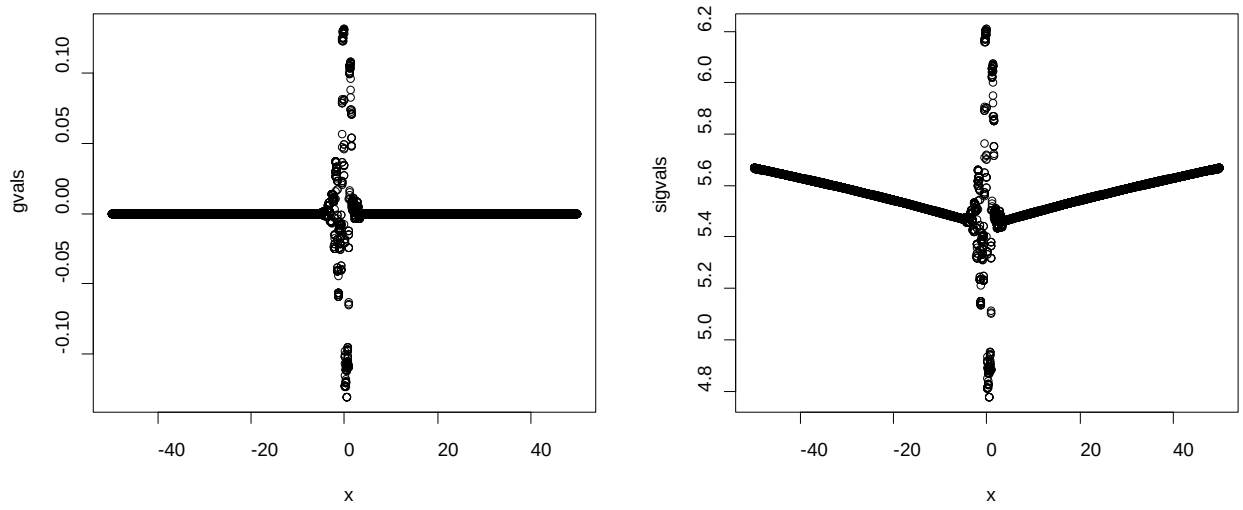




Accepted 23332 out of 100000 (23.332%).
final beta = 1.655068
alpha(0.000000) is 0.188214
alpha(2.000000) is 0.329689
alpha(5.000000) is 0.483880
alpha(10.000000) is 0.511928
alpha(20.000000) is 0.509062
alpha(50.000000) is 0.497378
alpha(100.000000) is 0.497785
alpha(150.000000) is 0.505373
it takes 127.757000 seconds to run the adaptation algorithm
it takes 56.383000 seconds to compute g and sigma
it takes 19.888000 seconds to generate the first Markov Chain
it takes 19.055000 seconds to generate the second Markov Chain
it takes 18.871000 seconds to generate the third Markov Chain
it takes 19.311000 seconds to generate the fourth Markov Chain
it takes 19.177000 seconds to generate the fifth Markov Chain
it takes 584.332000 seconds to test the local acceptance

[1] 6.692964
[1] 0.008586954
[1] 0.5457672

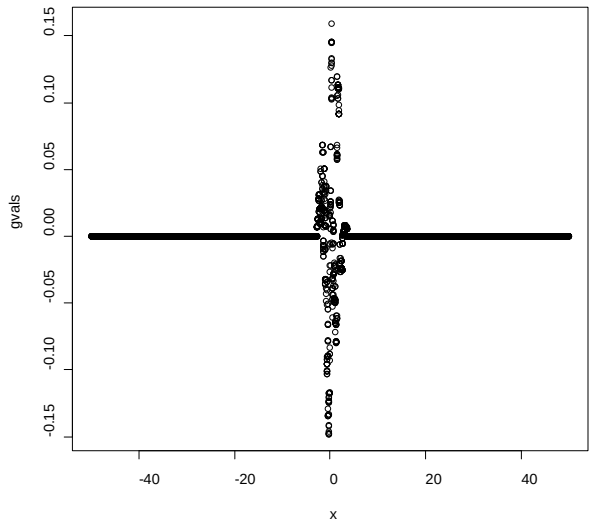
2.1.3.9 case 40 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.001, \gamma = 0.1$ with fixed bandwidth $b_n = 1$

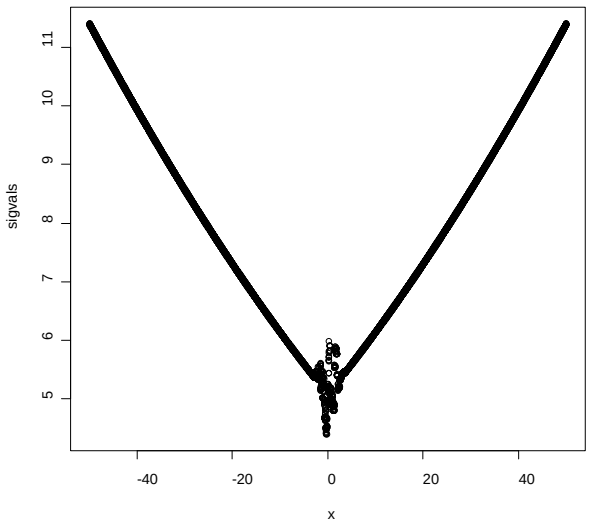


Accepted 22634 out of 100000 (22.634%).
final beta = 1.694196
alpha(0.000000) is 0.175360
alpha(2.000000) is 0.328895
alpha(5.000000) is 0.477720
alpha(10.000000) is 0.503992
alpha(20.000000) is 0.499353
alpha(50.000000) is 0.508652
alpha(100.000000) is 0.512111
alpha(150.000000) is 0.501107
it takes 129.093000 seconds to run the adaptation algorithm
it takes 60.973000 seconds to compute g and sigma
it takes 19.711000 seconds to generate the first Markov Chain
it takes 19.439000 seconds to generate the second Markov Chain
it takes 19.064000 seconds to generate the third Markov Chain
it takes 19.393000 seconds to generate the fourth Markov Chain
it takes 19.296000 seconds to generate the fifth Markov Chain
it takes 585.210000 seconds to test the local acceptance

[1] 6.49034
[1] 0.008560577
[1] 0.5352067

2.1.3.10 case 41 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 1, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

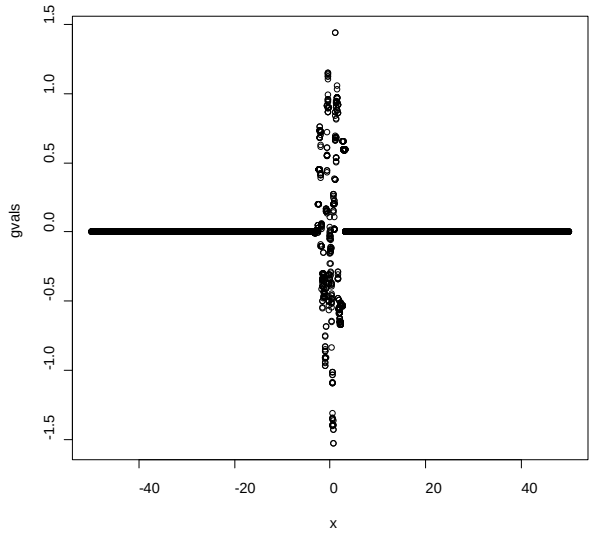


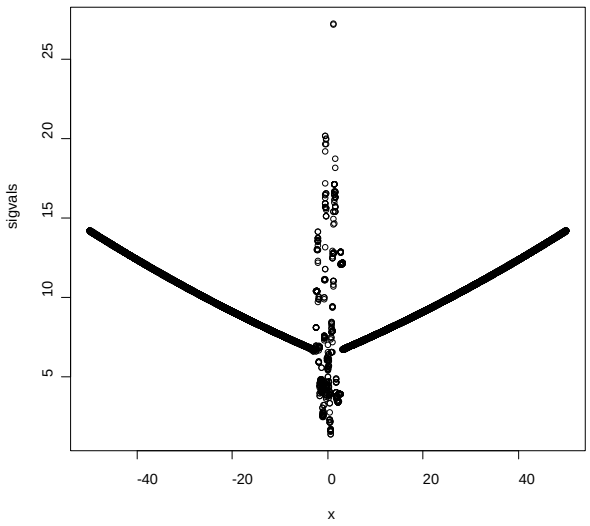


Accepted 23484 out of 100000 (23.484%).
final beta = 1.623149
alpha(0.000000) is 0.187799
alpha(2.000000) is 0.320460
alpha(5.000000) is 0.476159
alpha(10.000000) is 0.510392
alpha(20.000000) is 0.499119
alpha(50.000000) is 0.498086
alpha(100.000000) is 0.494173
alpha(150.000000) is 0.499675
it takes 128.783000 seconds to run the adaptation algorithm
it takes 56.547000 seconds to compute g and sigma
it takes 19.107000 seconds to generate the first Markov Chain
it takes 19.310000 seconds to generate the second Markov Chain
it takes 18.989000 seconds to generate the third Markov Chain
it takes 19.172000 seconds to generate the fourth Markov Chain
it takes 19.425000 seconds to generate the fifth Markov Chain
it takes 586.787000 seconds to test the local acceptance

[1] 6.555601
[1] 0.008636471
[1] 0.5541553

2.1.3.11 case 42 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 10, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

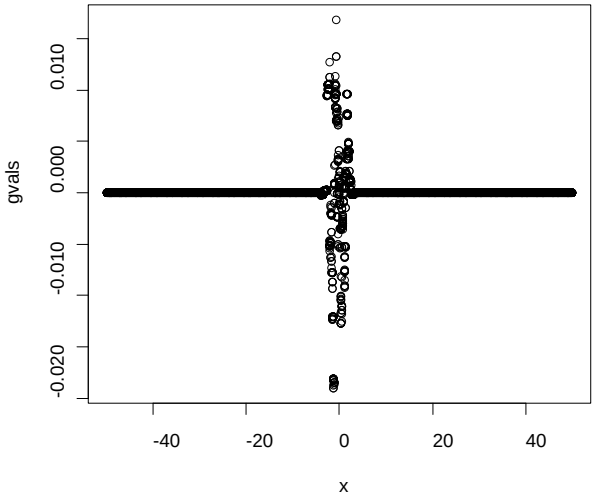


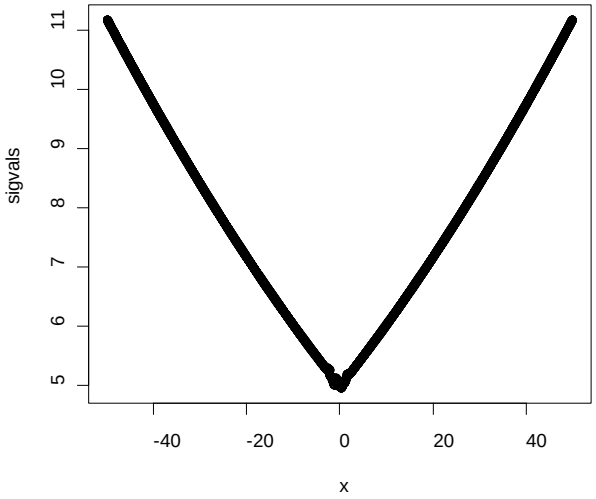


Accepted 18745 out of 100000 (18.745%).
final beta = 1.842263
alpha(0.000000) is 0.161404
alpha(2.000000) is 0.399803
alpha(5.000000) is 0.440082
alpha(10.000000) is 0.507989
alpha(20.000000) is 0.491224
alpha(50.000000) is 0.503609
alpha(100.000000) is 0.500292
alpha(150.000000) is 0.502577
it takes 129.356000 seconds to run the adaptation algorithm
it takes 57.235000 seconds to compute g and sigma
it takes 20.125000 seconds to generate the first Markov Chain
it takes 19.813000 seconds to generate the second Markov Chain
it takes 19.562000 seconds to generate the third Markov Chain
it takes 19.695000 seconds to generate the fourth Markov Chain
it takes 20.045000 seconds to generate the fifth Markov Chain
it takes 585.469000 seconds to test the local acceptance

[1] 9.308285
[1] 0.01020134
[1] 0.4585032

2.1.3.12 case 43 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.1, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

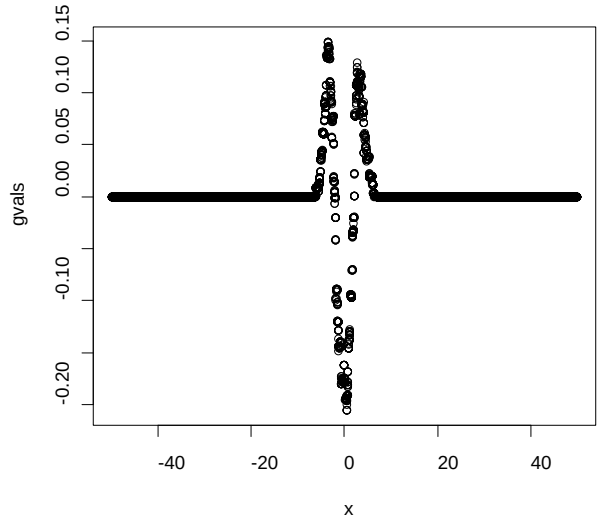


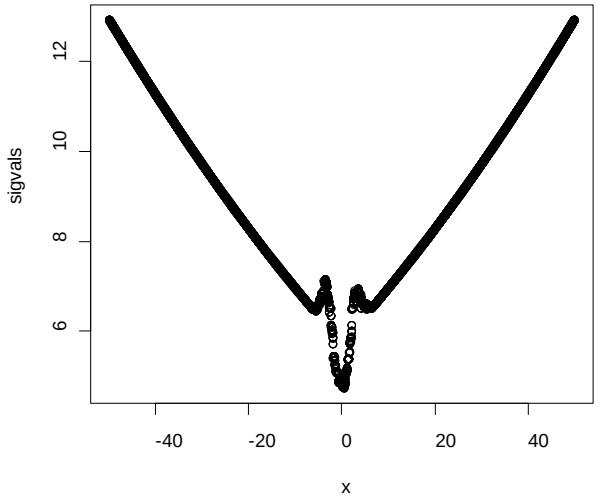


Accepted 23679 out of 100000 (23.679%).
final beta = 1.603976
alpha(0.000000) is 0.188531
alpha(2.000000) is 0.337021
alpha(5.000000) is 0.483636
alpha(10.000000) is 0.499772
alpha(20.000000) is 0.499243
alpha(50.000000) is 0.506060
alpha(100.000000) is 0.504384
alpha(150.000000) is 0.504052
it takes 130.732000 seconds to run the adaptation algorithm
it takes 57.978000 seconds to compute g and sigma
it takes 18.671000 seconds to generate the first Markov Chain
it takes 18.633000 seconds to generate the second Markov Chain
it takes 18.680000 seconds to generate the third Markov Chain
it takes 18.546000 seconds to generate the fourth Markov Chain
it takes 18.661000 seconds to generate the fifth Markov Chain
it takes 578.604000 seconds to test the local acceptance

[1] 6.726921
[1] 0.008570708
[1] 0.5399683
> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 6.362033
[1] 0.008291113
[1] 0.5503619

2.1.3.13 case 44 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 2, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$



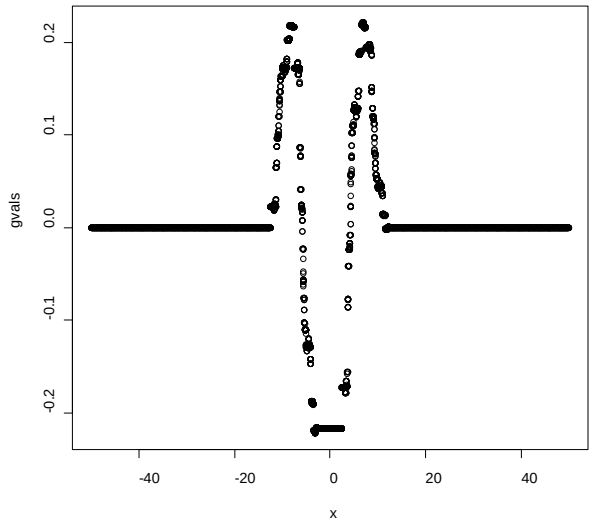


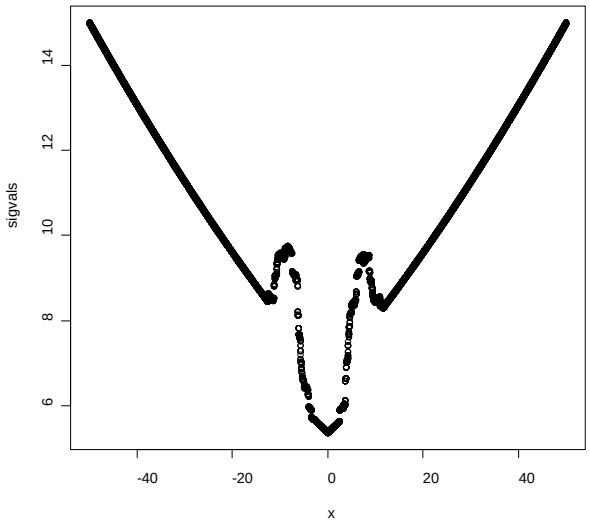
Accepted 23426 out of 100000 (23.426%).
final beta = 1.749096
alpha(0.000000) is 0.192279
alpha(2.000000) is 0.300504
alpha(5.000000) is 0.453942
alpha(10.000000) is 0.503413
alpha(20.000000) is 0.495909
alpha(50.000000) is 0.507192
alpha(100.000000) is 0.495004
alpha(150.000000) is 0.503508
it takes 129.124000 seconds to run the adaptation algorithm
it takes 57.509000 seconds to compute g and sigma
it takes 18.836000 seconds to generate the first Markov Chain
it takes 19.084000 seconds to generate the second Markov Chain
it takes 20.564000 seconds to generate the third Markov Chain
it takes 18.880000 seconds to generate the fourth Markov Chain

it takes 20.031000 seconds to generate the fifth Markov Chain
it takes 594.385000 seconds to test the local acceptance

[1] 0.5300015
> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 7.392876
[1] 0.009022338
[1] 0.5201393

2.1.3.14 case 45 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

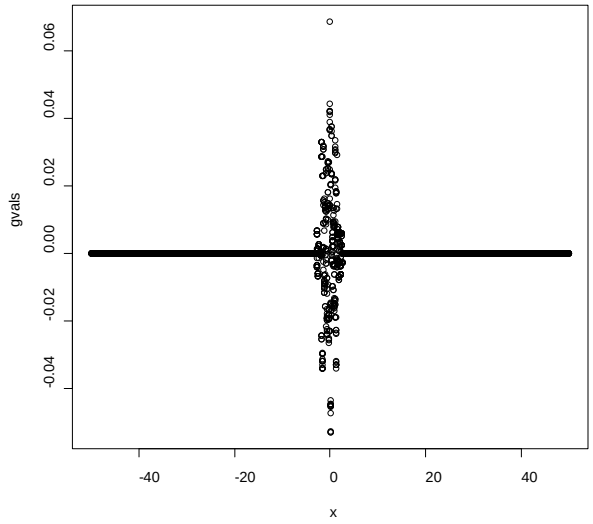


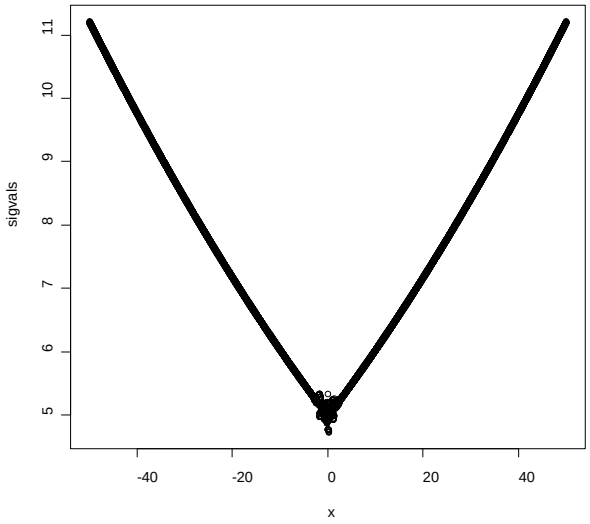


Accepted 21954 out of 100000 (21.954%).
final beta = 1.896775
alpha(0.000000) is 0.182106
alpha(2.000000) is 0.324927
alpha(5.000000) is 0.394897
alpha(10.000000) is 0.492385
alpha(20.000000) is 0.502033
alpha(50.000000) is 0.493569
alpha(100.000000) is 0.500820
alpha(150.000000) is 0.501123
it takes 130.563000 seconds to run the adaptation algorithm
it takes 57.790000 seconds to compute g and sigma
it takes 19.697000 seconds to generate the first Markov Chain
it takes 19.753000 seconds to generate the second Markov Chain
it takes 19.598000 seconds to generate the third Markov Chain
it takes 19.523000 seconds to generate the fourth Markov Chain
it takes 19.212000 seconds to generate the fifth Markov Chain
it takes 603.258000 seconds to test the local acceptance

[1] 6.854895
[1] 0.008702906
[1] 0.5164709

2.1.3.15 case 46 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.1, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

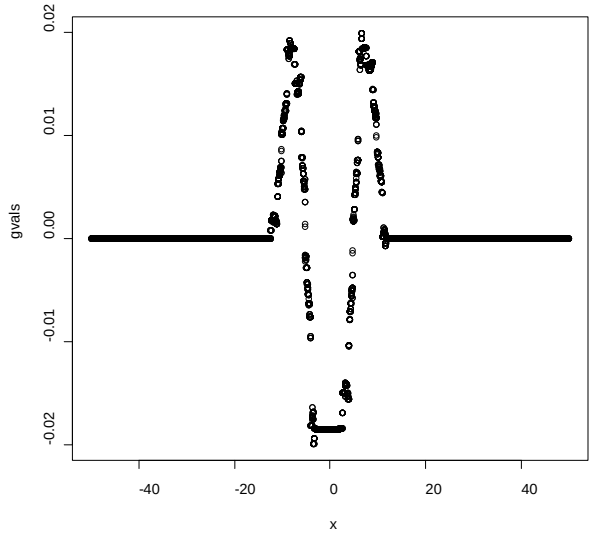


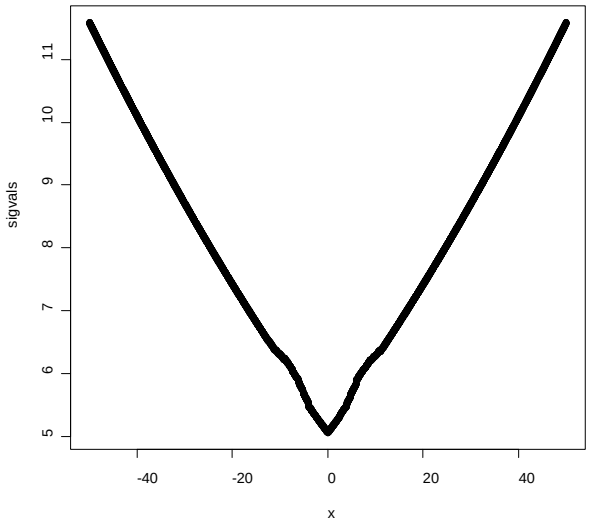


Accepted 23642 out of 100000 (23.642%).
final beta = 1.605517
alpha(0.000000) is 0.192827
alpha(2.000000) is 0.325493
alpha(5.000000) is 0.478442
alpha(10.000000) is 0.512192
alpha(20.000000) is 0.499076
alpha(50.000000) is 0.493956
alpha(100.000000) is 0.503451
alpha(150.000000) is 0.504400
it takes 131.982000 seconds to run the adaptation algorithm
it takes 58.176000 seconds to compute g and sigma
it takes 20.263000 seconds to generate the first Markov Chain
it takes 20.133000 seconds to generate the second Markov Chain
it takes 20.427000 seconds to generate the third Markov Chain
it takes 20.049000 seconds to generate the fourth Markov Chain
it takes 20.327000 seconds to generate the fifth Markov Chain
it takes 605.936000 seconds to test the local acceptance

[1] 6.081314
[1] 0.008252063
[1] 0.5518468

2.1.3.16 case 47 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 10$

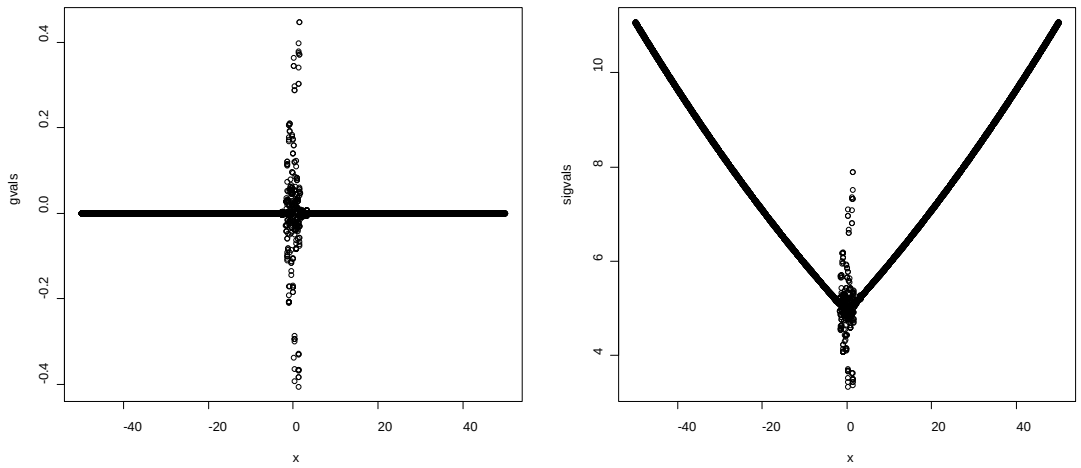




Accepted 23590 out of 100000 (23.590%).
final beta = 1.639301
alpha(0.000000) is 0.191540
alpha(2.000000) is 0.335518
alpha(5.000000) is 0.479633
alpha(10.000000) is 0.498527
alpha(20.000000) is 0.500063
alpha(50.000000) is 0.505558
alpha(100.000000) is 0.500621
alpha(150.000000) is 0.493689
it takes 133.815000 seconds to run the adaptation algorithm
it takes 59.140000 seconds to compute g and sigma
it takes 18.822000 seconds to generate the first Markov Chain
it takes 19.763000 seconds to generate the second Markov Chain
it takes 19.456000 seconds to generate the third Markov Chain
it takes 18.831000 seconds to generate the fourth Markov Chain
it takes 19.578000 seconds to generate the fifth Markov Chain
it takes 600.070000 seconds to test the local acceptance

[1] 6.359084
[1] 0.00842308
[1] 0.554771

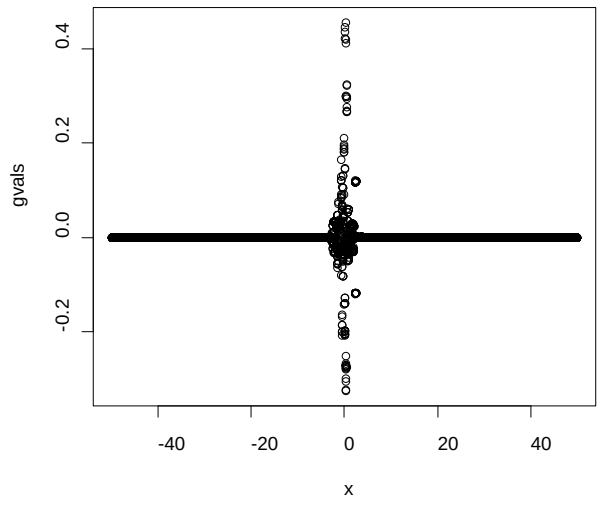
2.1.3.17 case 48 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 0.1$

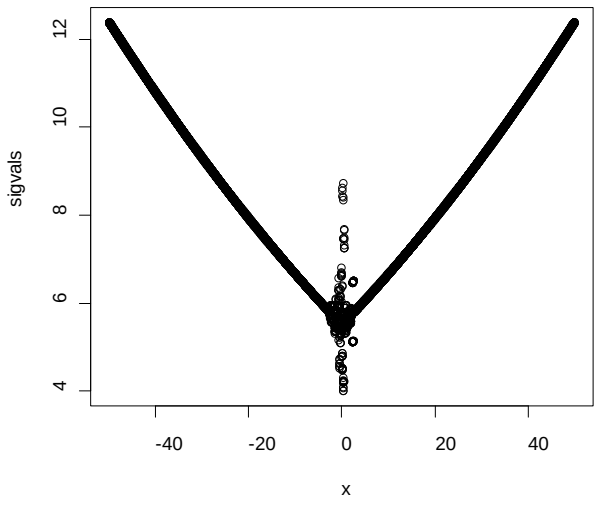


Accepted 24144 out of 100000 (24.144%).
final beta = 1.592917
alpha(0.000000) is 0.198912
alpha(2.000000) is 0.328186
alpha(5.000000) is 0.474185
alpha(10.000000) is 0.494897
alpha(20.000000) is 0.509700
alpha(50.000000) is 0.493201
alpha(100.000000) is 0.508407
alpha(150.000000) is 0.497111
it takes 132.193000 seconds to run the adaptation algorithm
it takes 58.991000 seconds to compute g and sigma
it takes 20.215000 seconds to generate the first Markov Chain
it takes 19.654000 seconds to generate the second Markov Chain
it takes 19.407000 seconds to generate the third Markov Chain
it takes 19.431000 seconds to generate the fourth Markov Chain
it takes 19.357000 seconds to generate the fifth Markov Chain
it takes 609.803000 seconds to test the local acceptance

```
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008251732
> asjd(xlist)
[1] 0.543797
>
> source("xlist4")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 6.212332
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008317304
> asjd(xlist)
[1] 0.5649309
>
> source("xlist5")
> plot(xlist[(B+1):M],type="l")
> varfact(xlist[(B+1):M]);
[1] 6.130305
> sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
[1] 0.008166715
> asjd(xlist)
[1] 0.5469031
```

2.1.3.18 case 49 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 2$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$



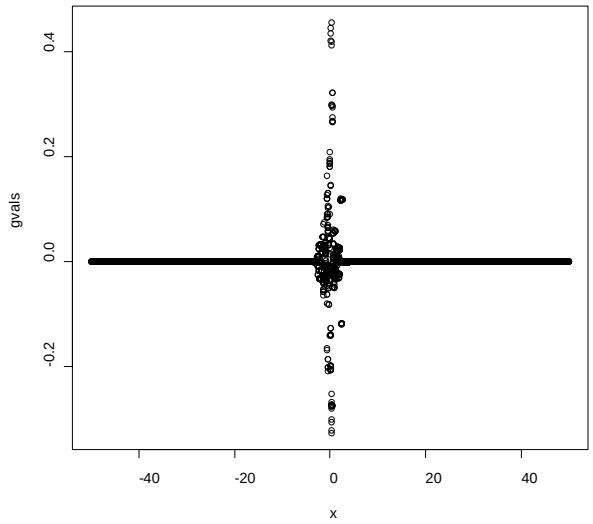


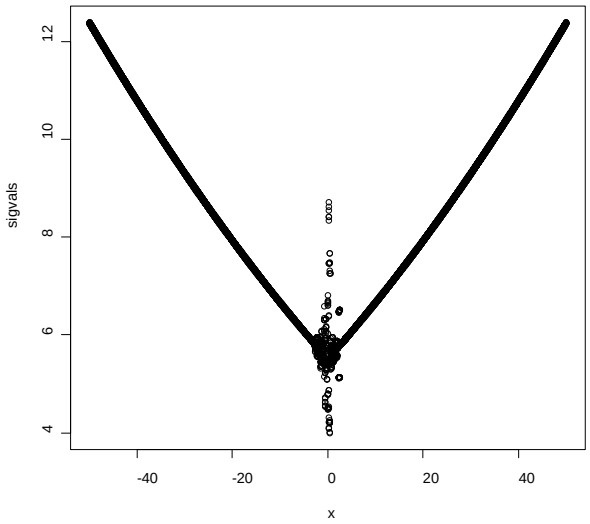
Accepted 21609 out of 100000 (21.609%).
final beta = 1.705975

alpha(0.000000) is 0.168346
alpha(2.000000) is 0.303514
alpha(5.000000) is 0.473132
alpha(10.000000) is 0.508745
alpha(20.000000) is 0.500702
alpha(50.000000) is 0.502543
alpha(100.000000) is 0.501443
alpha(150.000000) is 0.500437
it takes 132.324000 seconds to run the adaptation algorithm
it takes 58.739000 seconds to compute g and sigma
it takes 19.708000 seconds to generate the first Markov Chain
it takes 19.571000 seconds to generate the second Markov Chain
it takes 19.439000 seconds to generate the third Markov Chain
it takes 19.542000 seconds to generate the fourth Markov Chain
it takes 19.544000 seconds to generate the fifth Markov Chain
it takes 599.769000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 7.150534
[1] 0.008775849
[1] 0.5047632
> source("xlist3");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) /
sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 7.179354
[1] 0.008915502
[1] 0.5114743
> source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) /
sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 7.156782
[1] 0.008810378
[1] 0.5108781
> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) /
sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 7.01146
[1] 0.008889637
[1] 0.5306071

2.1.3.19 case 50 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$





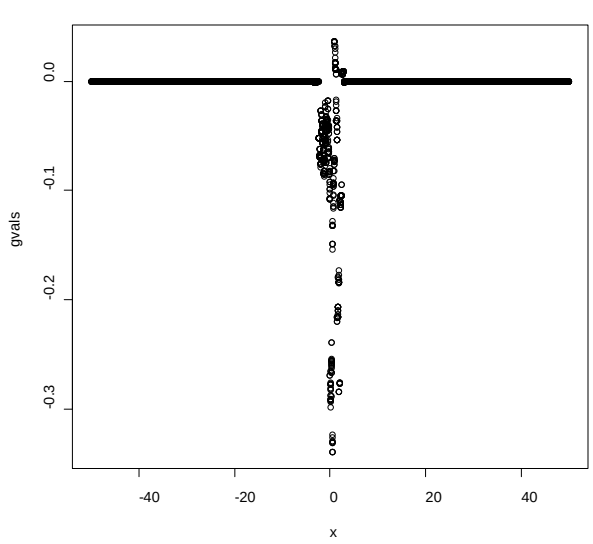
Accepted 21609 out of 100000 (21.609%).
final beta = 1.705975
alpha(0.000000) is 0.168346
alpha(2.000000) is 0.303514
alpha(5.000000) is 0.473132

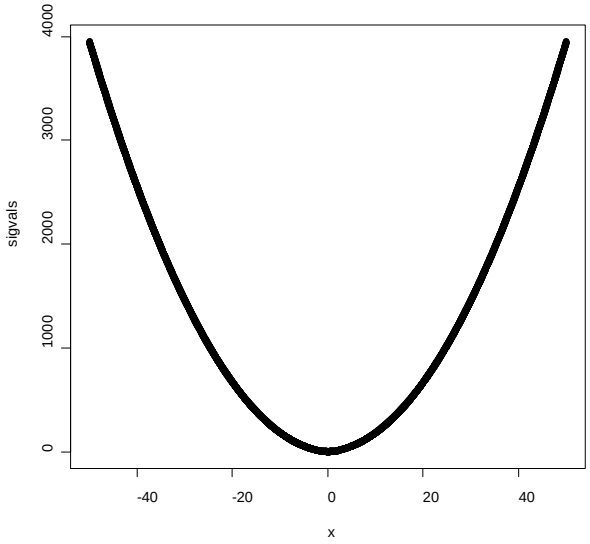
alpha(10.000000) is 0.508745
alpha(20.000000) is 0.500702
alpha(50.000000) is 0.502543
alpha(100.000000) is 0.501443
alpha(150.000000) is 0.500437
it takes 132.324000 seconds to run the adaptation algorithm
it takes 58.739000 seconds to compute g and sigma
it takes 19.708000 seconds to generate the first Markov Chain
it takes 19.571000 seconds to generate the second Markov Chain
it takes 19.439000 seconds to generate the third Markov Chain
it takes 19.542000 seconds to generate the fourth Markov Chain
it takes 19.544000 seconds to generate the fifth Markov Chain
it takes 599.769000 seconds to test the local acceptance

```
[1] 0.5047632
>    source("xlist3");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 7.179354
[1] 0.008915502
[1] 0.5114743
>    source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 7.156782
[1] 0.008810378
[1] 0.5108781
>    source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 7.01146
[1] 0.008889637
[1] 0.5306071
```

2.1.4 kernel function $K(x)=\begin{cases}0, |x|\geq 2*width \quad or \quad |x|\leq width\\ 1, width<|x|<2*width\end{cases}$

2.1.4.1 case 51 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\text{width}=0.5, C=1, \gamma=2$ with fixed bandwidth $b_n=1$

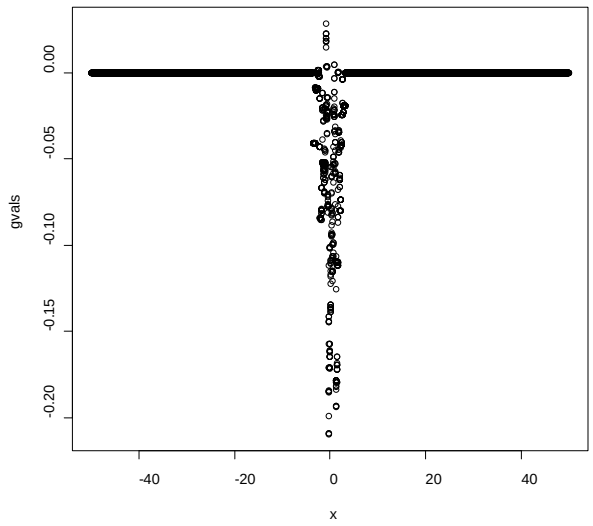


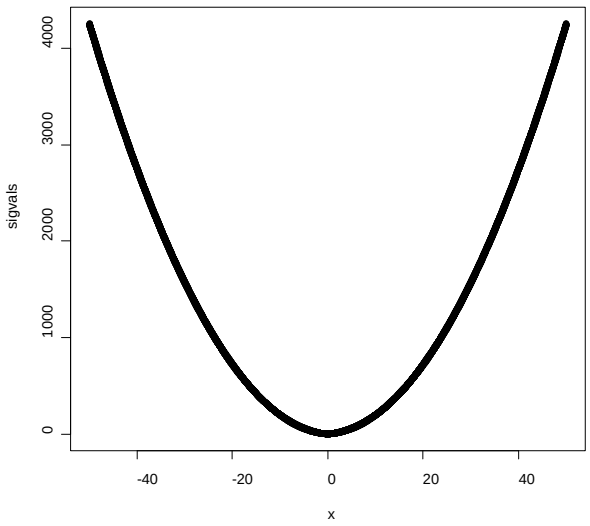


Accepted 23009 out of 100000 (23.009%).
final beta = 0.417902
alpha(0.000000) is 0.310259
alpha(2.000000) is 0.143592
alpha(5.000000) is 0.076589
alpha(10.000000) is 0.041575
alpha(20.000000) is 0.022942
alpha(50.000000) is 0.009200
alpha(100.000000) is 0.005400
alpha(150.000000) is 0.004200
it takes 125.540000 seconds to run the adaptation algorithm
it takes 55.805000 seconds to compute g and sigma
it takes 19.754000 seconds to generate the first Markov Chain
it takes 21.302000 seconds to generate the second Markov Chain
it takes 22.193000 seconds to generate the third Markov Chain
it takes 20.612000 seconds to generate the fourth Markov Chain
it takes 20.616000 seconds to generate the fifth Markov Chain
it takes 974.826000 seconds to test the local acceptance

[1] 11.34705
[1] 0.01121087
[1] 0.3325641

2.1.4.2 case 52 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$

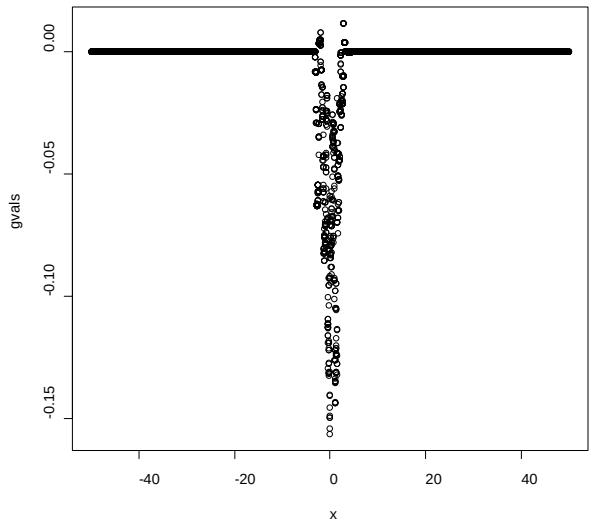


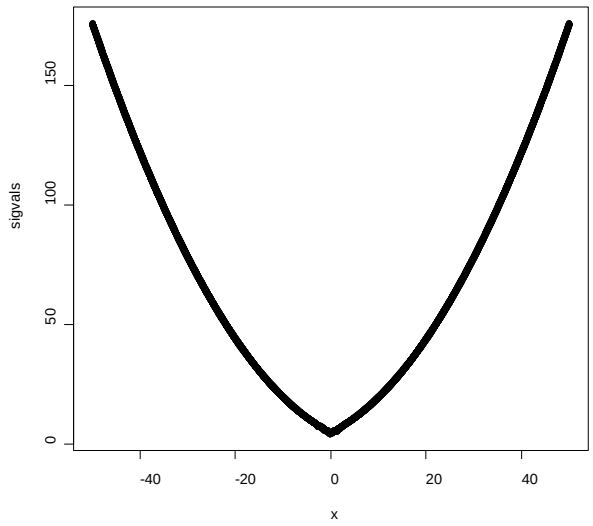


Accepted 20742 out of 100000 (20.742%).
final beta = 0.491977
alpha(0.000000) is 0.275429
alpha(2.000000) is 0.134558
alpha(5.000000) is 0.065966
alpha(10.000000) is 0.044212
alpha(20.000000) is 0.024001
alpha(50.000000) is 0.009100
alpha(100.000000) is 0.004300
alpha(150.000000) is 0.003400
it takes 128.005000 seconds to run the adaptation algorithm
it takes 56.269000 seconds to compute g and sigma
it takes 19.186000 seconds to generate the first Markov Chain
it takes 19.712000 seconds to generate the second Markov Chain
it takes 20.762000 seconds to generate the third Markov Chain
it takes 21.651000 seconds to generate the fourth Markov Chain
it takes 20.180000 seconds to generate the fifth Markov Chain
it takes 1010.090000 seconds to test the local acceptance

[1] 12.73849
[1] 0.01200023
[1] 0.3157405

2.1.4.3 case 53 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$



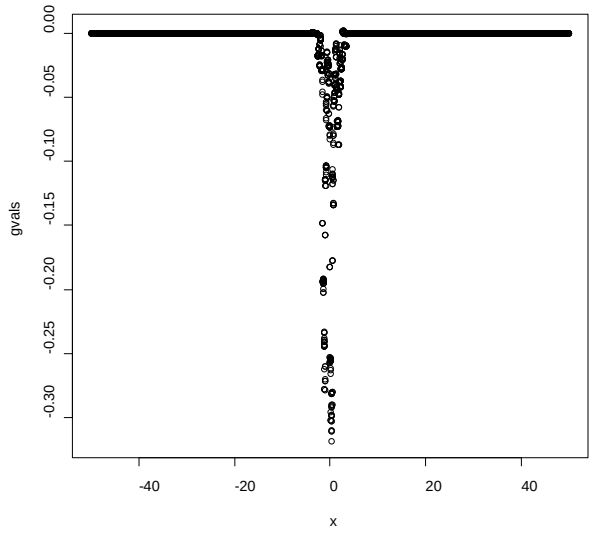


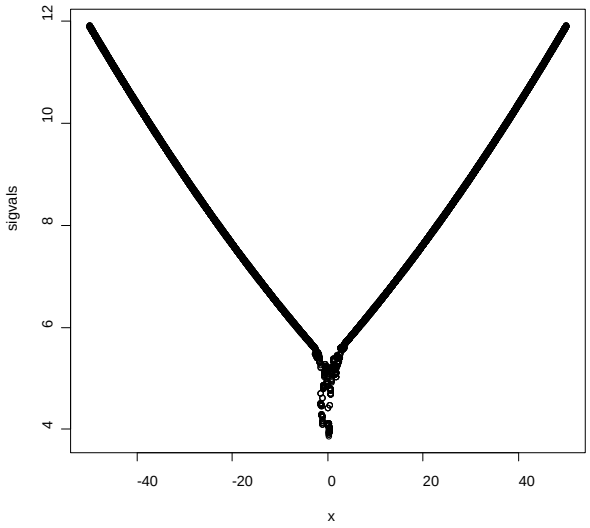
Accepted 21407 out of 100000 (21.407%).
final beta = 1.586745
alpha(0.000000) is 0.186088
alpha(2.000000) is 0.255791
alpha(5.000000) is 0.322271
alpha(10.000000) is 0.361781
alpha(20.000000) is 0.319420
alpha(50.000000) is 0.215913
alpha(100.000000) is 0.130600
alpha(150.000000) is 0.098400
it takes 133.242000 seconds to run the adaptation algorithm
it takes 59.019000 seconds to compute g and sigma
it takes 22.075000 seconds to generate the first Markov Chain
it takes 21.186000 seconds to generate the second Markov Chain
it takes 20.902000 seconds to generate the third Markov Chain
it takes 20.804000 seconds to generate the fourth Markov Chain
it takes 22.036000 seconds to generate the fifth Markov Chain
it takes 774.597000 seconds to test the local acceptance

[1] 7.425369
[1] 0.00926486
[1] 0.4934043

2.1.4.4 case 54 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 0.01, \gamma = 2$ with fixed bandwidth

$b_n = 1$

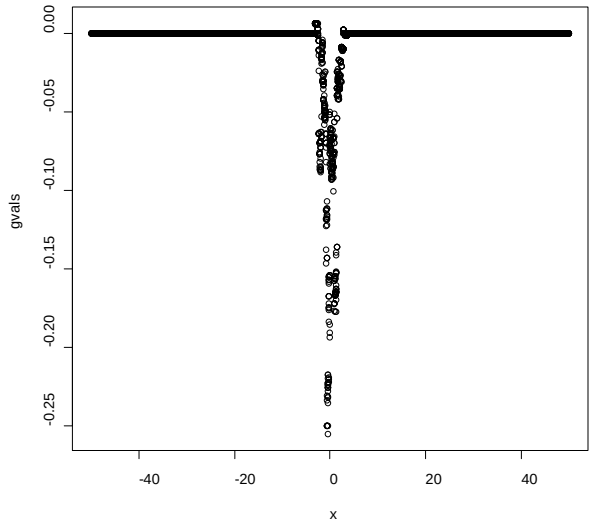


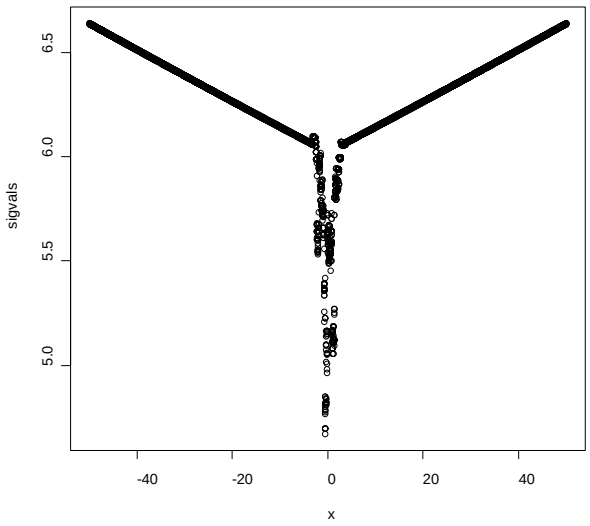


Accepted 24563 out of 100000 (24.563%).
final beta = 1.666639
alpha(0.000000) is 0.204343
alpha(2.000000) is 0.324662
alpha(5.000000) is 0.465485
alpha(10.000000) is 0.501068
alpha(20.000000) is 0.505233
alpha(50.000000) is 0.498960
alpha(100.000000) is 0.511536
alpha(150.000000) is 0.503191
it takes 126.672000 seconds to run the adaptation algorithm
it takes 55.830000 seconds to compute g and sigma
it takes 20.221000 seconds to generate the first Markov Chain
it takes 19.321000 seconds to generate the second Markov Chain
it takes 19.575000 seconds to generate the third Markov Chain
it takes 19.669000 seconds to generate the fourth Markov Chain
it takes 19.634000 seconds to generate the fifth Markov Chain
it takes 576.363000 seconds to test the local acceptance

[1] 6.352072
[1] 0.008441499
[1] 0.5698832

2.1.4.5 case 55 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 1, C = 0.001, \gamma = 2$ with fixed bandwidth $b_n = 1$

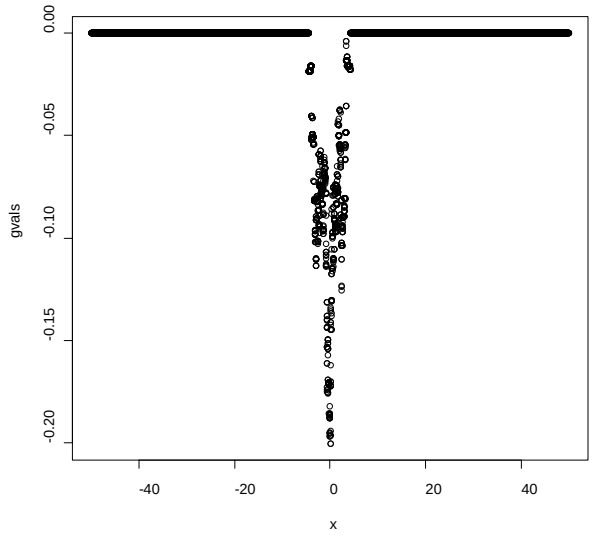


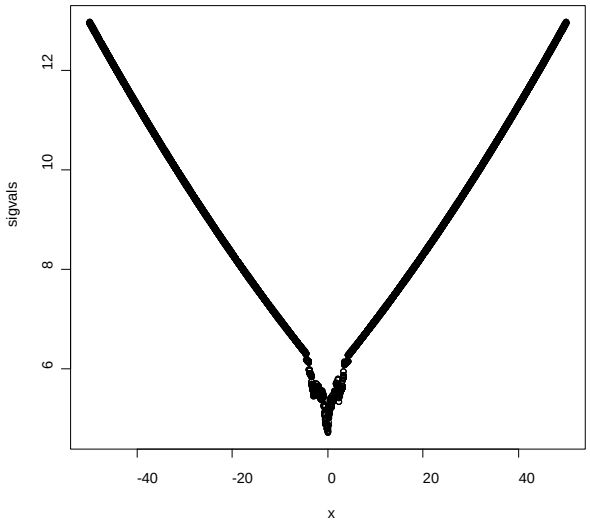


Accepted 21862 out of 100000 (21.862%).
final beta = 1.795190
alpha(0.000000) is 0.178309
alpha(2.000000) is 0.313602
alpha(5.000000) is 0.465026
alpha(10.000000) is 0.507124
alpha(20.000000) is 0.504982
alpha(50.000000) is 0.506570
alpha(100.000000) is 0.500667
alpha(150.000000) is 0.491467
it takes 126.739000 seconds to run the adaptation algorithm
it takes 55.053000 seconds to compute g and sigma
it takes 19.893000 seconds to generate the first Markov Chain
it takes 19.991000 seconds to generate the second Markov Chain
it takes 20.178000 seconds to generate the third Markov Chain
it takes 20.074000 seconds to generate the fourth Markov Chain
it takes 20.152000 seconds to generate the fifth Markov Chain
it takes 580.130000 seconds to test the local acceptance

[1] 7.055358
[1] 0.008838765
[1] 0.5103956

2.1.4.6 case 56 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 1, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

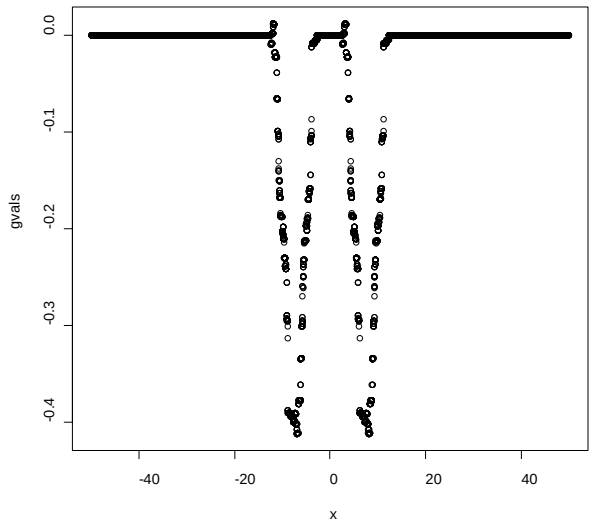


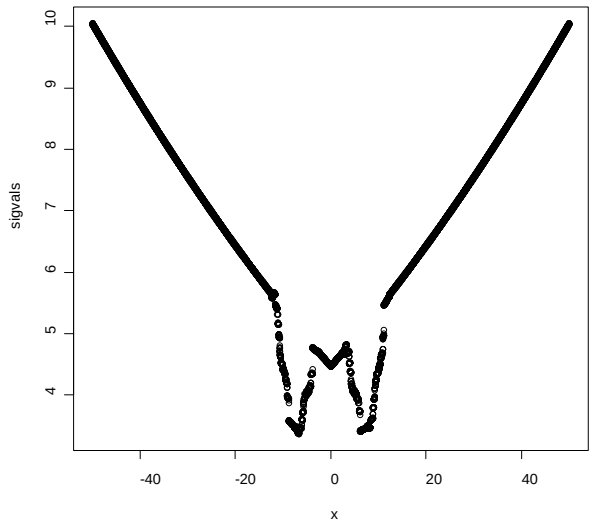


Accepted 22481 out of 100000 (22.481%).
final beta = 1.752170
alpha(0.000000) is 0.189223
alpha(2.000000) is 0.310525
alpha(5.000000) is 0.459372
alpha(10.000000) is 0.497289
alpha(20.000000) is 0.503854
alpha(50.000000) is 0.495202
alpha(100.000000) is 0.494385
alpha(150.000000) is 0.509056
it takes 128.270000 seconds to run the adaptation algorithm
it takes 56.474000 seconds to compute g and sigma
it takes 19.944000 seconds to generate the first Markov Chain
it takes 19.749000 seconds to generate the second Markov Chain
it takes 21.216000 seconds to generate the third Markov Chain
it takes 20.799000 seconds to generate the fourth Markov Chain
it takes 20.124000 seconds to generate the fifth Markov Chain
it takes 582.432000 seconds to test the local acceptance

[1] 7.293104
[1] 0.008969298
[1] 0.5260217

2.1.4.7 case 57 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$



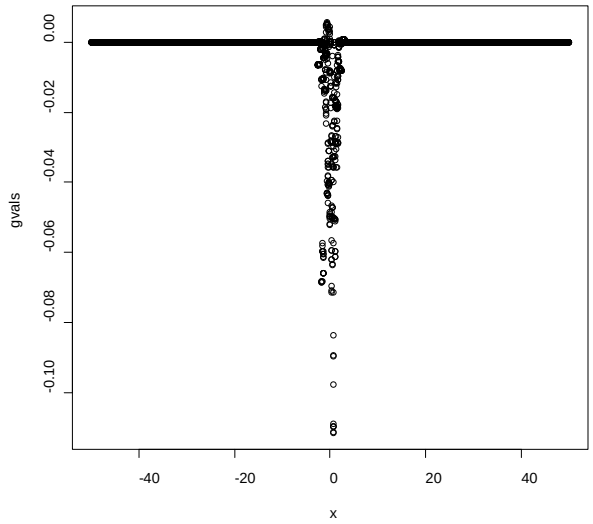


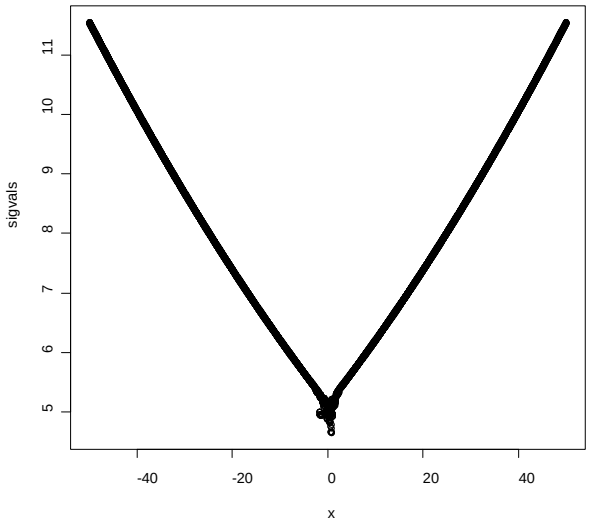
Accepted 26365 out of 100000 (26.365%).
final beta = 1.496543
alpha(0.000000) is 0.216441
alpha(2.000000) is 0.368902
alpha(5.000000) is 0.500734
alpha(10.000000) is 0.507066
alpha(20.000000) is 0.502065
alpha(50.000000) is 0.502358
alpha(100.000000) is 0.495112
alpha(150.000000) is 0.503168
it takes 124.340000 seconds to run the adaptation algorithm
it takes 56.677000 seconds to compute g and sigma
it takes 21.173000 seconds to generate the first Markov Chain
it takes 20.264000 seconds to generate the second Markov Chain
it takes 19.534000 seconds to generate the third Markov Chain
it takes 19.489000 seconds to generate the fourth Markov Chain
it takes 18.931000 seconds to generate the fifth Markov Chain
it takes 569.717000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 5.500506
[1] 0.007780254
[1] 0.5968249

2.1.4.8 case 58 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.1, C = 0.01, \gamma = 2$ with fixed bandwidth

$b_n = 1$

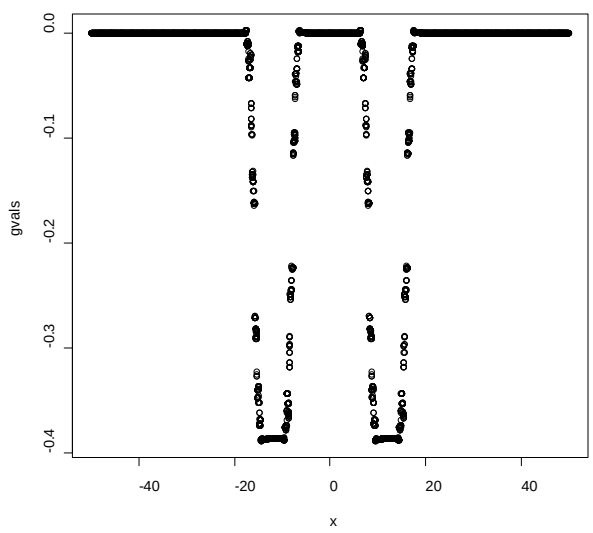


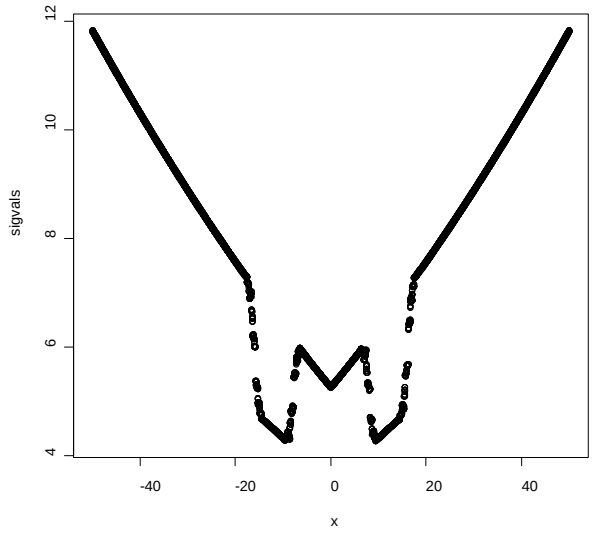


Accepted 23471 out of 100000 (23.471%).
final beta = 1.635035
alpha(0.000000) is 0.196509
alpha(2.000000) is 0.329970
alpha(5.000000) is 0.479467
alpha(10.000000) is 0.512654
alpha(20.000000) is 0.507827
alpha(50.000000) is 0.496350
alpha(100.000000) is 0.489645
alpha(150.000000) is 0.503596
it takes 128.387000 seconds to run the adaptation algorithm
it takes 57.261000 seconds to compute g and sigma
it takes 22.145000 seconds to generate the first Markov Chain
it takes 21.970000 seconds to generate the second Markov Chain
it takes 21.670000 seconds to generate the third Markov Chain
it takes 22.460000 seconds to generate the fourth Markov Chain
it takes 22.477000 seconds to generate the fifth Markov Chain
it takes 589.629000 seconds to test the local acceptance

[1] 6.456669
[1] 0.008401081
[1] 0.5444502

2.1.4.9 case 59 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 8, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

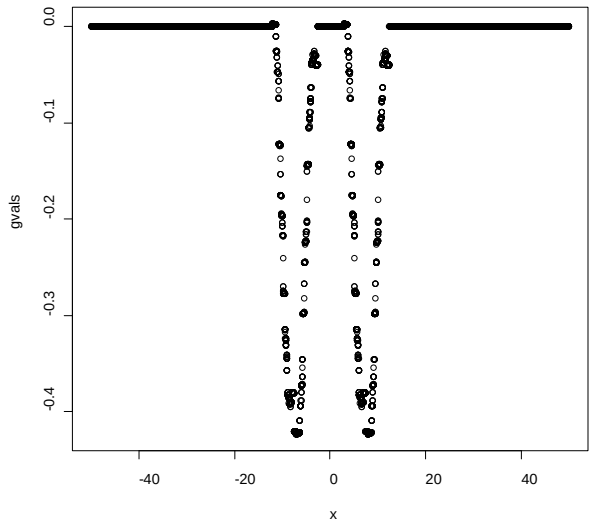


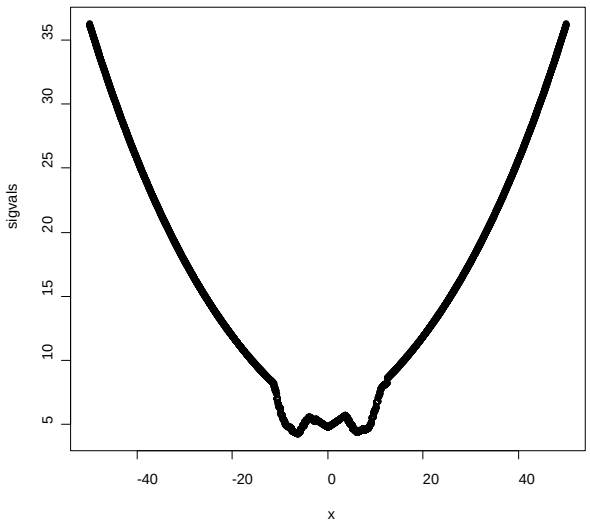


Accepted 22642 out of 100000 (22.642%).
final beta = 1.660210
alpha(0.000000) is 0.182785
alpha(2.000000) is 0.313792
alpha(5.000000) is 0.478720
alpha(10.000000) is 0.516151
alpha(20.000000) is 0.495757
alpha(50.000000) is 0.495627
alpha(100.000000) is 0.510013
alpha(150.000000) is 0.506006
it takes 123.308000 seconds to run the adaptation algorithm
it takes 54.821000 seconds to compute g and sigma
it takes 19.924000 seconds to generate the first Markov Chain
it takes 19.878000 seconds to generate the second Markov Chain
it takes 19.394000 seconds to generate the third Markov Chain
it takes 19.481000 seconds to generate the fourth Markov Chain
it takes 20.128000 seconds to generate the fifth Markov Chain
it takes 574.279000 seconds to test the local acceptance

> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 6.965559
[1] 0.008812787
[1] 0.5334517

2.1.4.10 case 60 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 5$ with fixed bandwidth $b_n = 1$



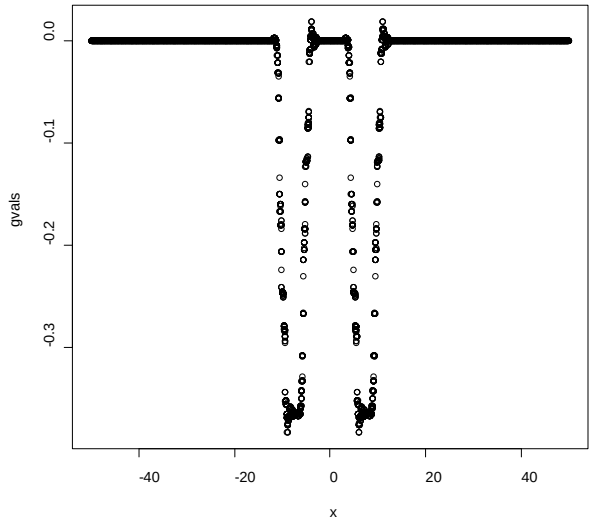


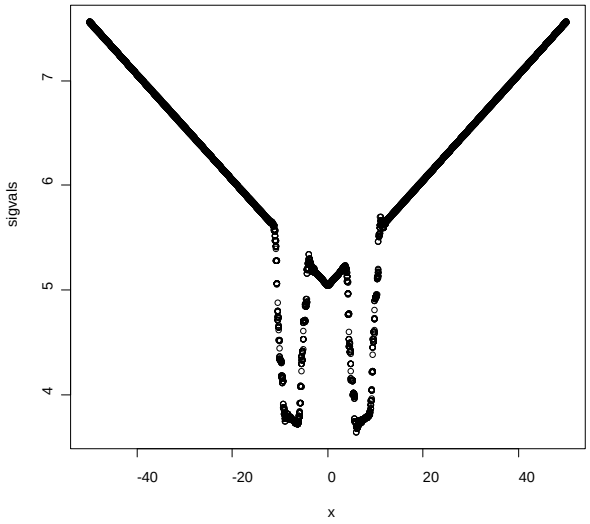
Accepted 23976 out of 100000 (23.976%).
final beta = 1.563580
alpha(0.000000) is 0.199347
alpha(2.000000) is 0.334377
alpha(5.000000) is 0.500748
alpha(10.000000) is 0.507236
alpha(20.000000) is 0.505732
alpha(50.000000) is 0.493323
alpha(100.000000) is 0.399116
alpha(150.000000) is 0.235500
it takes 124.660000 seconds to run the adaptation algorithm
it takes 55.283000 seconds to compute g and sigma
it takes 19.535000 seconds to generate the first Markov Chain
it takes 19.427000 seconds to generate the second Markov Chain
it takes 19.173000 seconds to generate the third Markov Chain
it takes 19.934000 seconds to generate the fourth Markov Chain
it takes 20.513000 seconds to generate the fifth Markov Chain
it takes 594.727000 seconds to test the local acceptance

[1] 6.651483
[1] 0.008605778
[1] 0.5450531

2.1.4.11 case 61 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 1$ with fixed bandwidth

$b_n = 1$



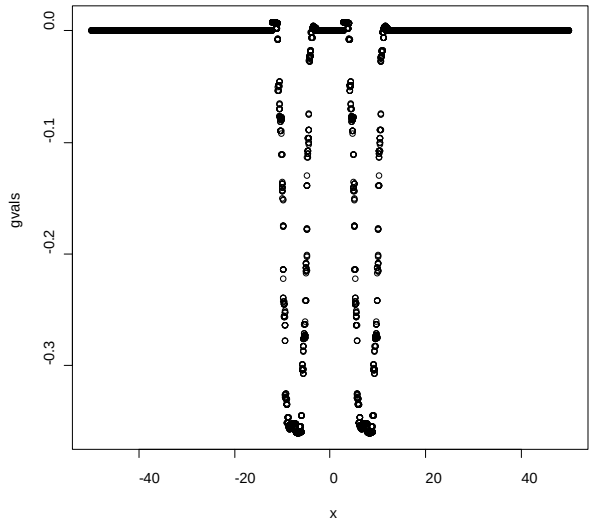


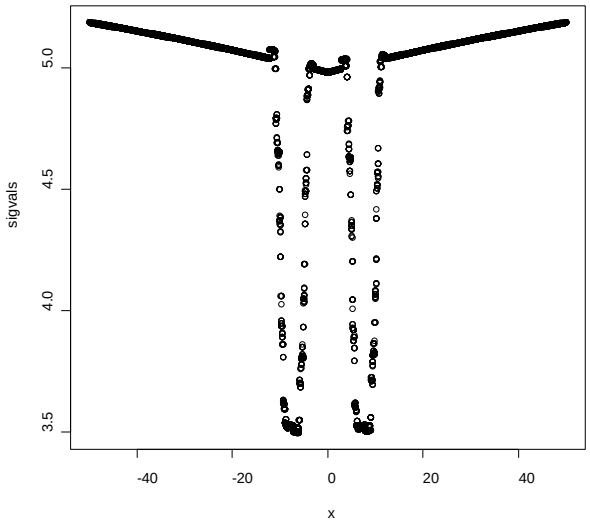
Accepted 23675 out of 100000 (23.675%).
final beta = 1.617569
alpha(0.000000) is 0.197561
alpha(2.000000) is 0.334814
alpha(5.000000) is 0.509056
alpha(10.000000) is 0.508438
alpha(20.000000) is 0.499158
alpha(50.000000) is 0.497928
alpha(100.000000) is 0.501279
alpha(150.000000) is 0.493803
it takes 125.480000 seconds to run the adaptation algorithm
it takes 56.040000 seconds to compute g and sigma
it takes 19.422000 seconds to generate the first Markov Chain
it takes 19.479000 seconds to generate the second Markov Chain
it takes 19.599000 seconds to generate the third Markov Chain
it takes 20.212000 seconds to generate the fourth Markov Chain
it takes 19.031000 seconds to generate the fifth Markov Chain
it takes 590.678000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 6.598833
[1] 0.008443037
[1] 0.5469699

2.1.4.12 case 62 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 0.1$ with fixed bandwidth

$b_n = 1$



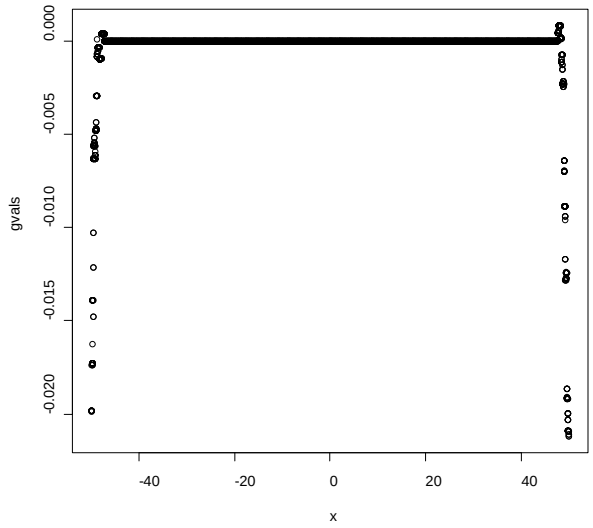


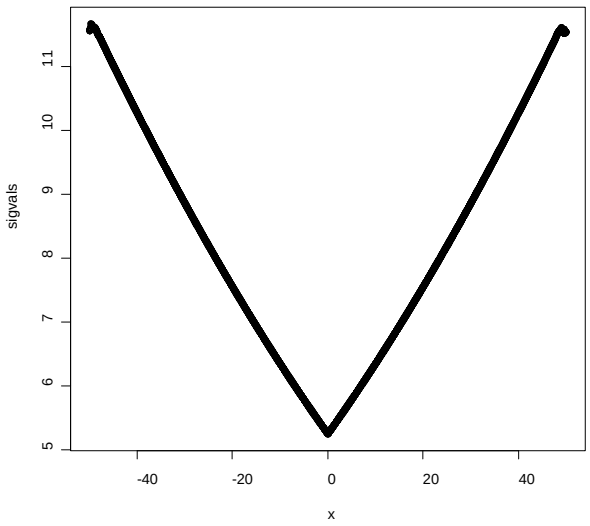
Accepted 24397 out of 100000 (24.397%).
final beta = 1.605496
alpha(0.000000) is 0.196312
alpha(2.000000) is 0.351285
alpha(5.000000) is 0.508944
alpha(10.000000) is 0.503029
alpha(20.000000) is 0.505562
alpha(50.000000) is 0.500858
alpha(100.000000) is 0.498800
alpha(150.000000) is 0.498894
it takes 124.458000 seconds to run the adaptation algorithm
it takes 55.248000 seconds to compute g and sigma
it takes 19.847000 seconds to generate the first Markov Chain
it takes 18.661000 seconds to generate the second Markov Chain
it takes 18.698000 seconds to generate the third Markov Chain
it takes 19.318000 seconds to generate the fourth Markov Chain
it takes 20.375000 seconds to generate the fifth Markov Chain
it takes 595.427000 seconds to test the local acceptance

[1] 6.12154
[1] 0.00825666
[1] 0.5646845

2.1.4.13 case 63 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth

$b_n = 10$



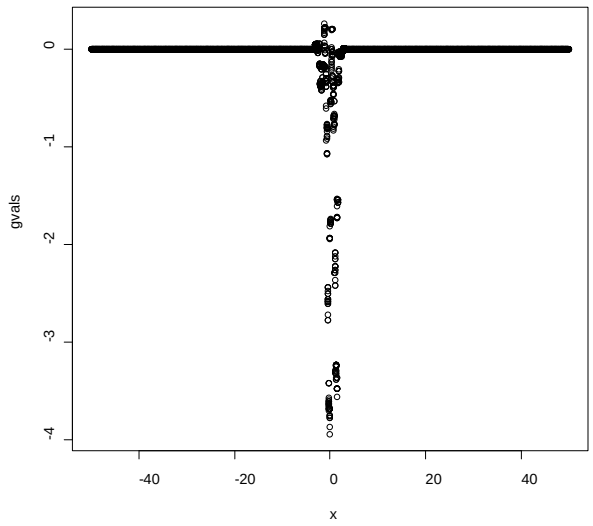


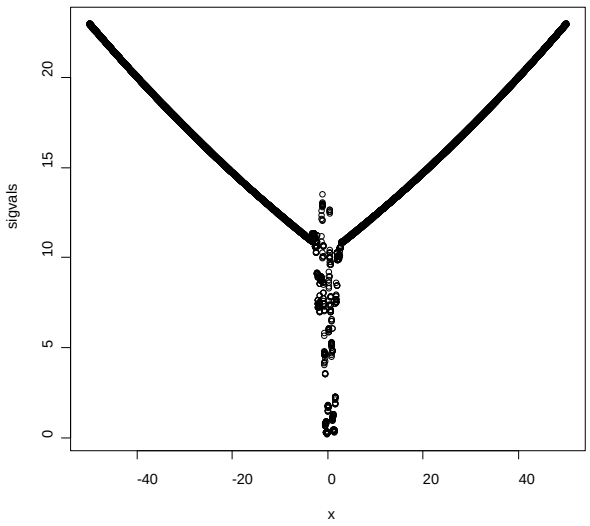
Accepted 22622 out of 100000 (22.622%).
final beta = 1.657513
alpha(0.000000) is 0.183999
alpha(2.000000) is 0.323600
alpha(5.000000) is 0.475474
alpha(10.000000) is 0.506252
alpha(20.000000) is 0.512030
alpha(50.000000) is 0.506490
alpha(100.000000) is 0.493719
alpha(150.000000) is 0.506028
it takes 130.990000 seconds to run the adaptation algorithm
it takes 57.461000 seconds to compute g and sigma
it takes 20.162000 seconds to generate the first Markov Chain
it takes 20.553000 seconds to generate the second Markov Chain
it takes 20.230000 seconds to generate the third Markov Chain
it takes 20.700000 seconds to generate the fourth Markov Chain
it takes 20.905000 seconds to generate the fifth Markov Chain
it takes 601.853000 seconds to test the local acceptance

[1] 6.666041
[1] 0.008627651
[1] 0.541695

2.1.4.14 case 64 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth

$b_n = 0.1$

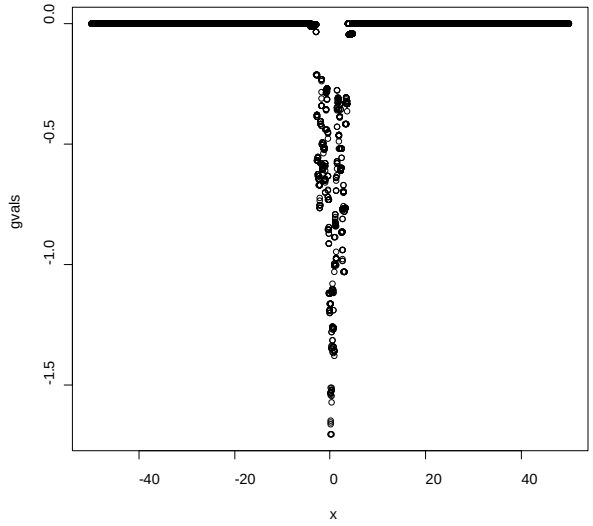


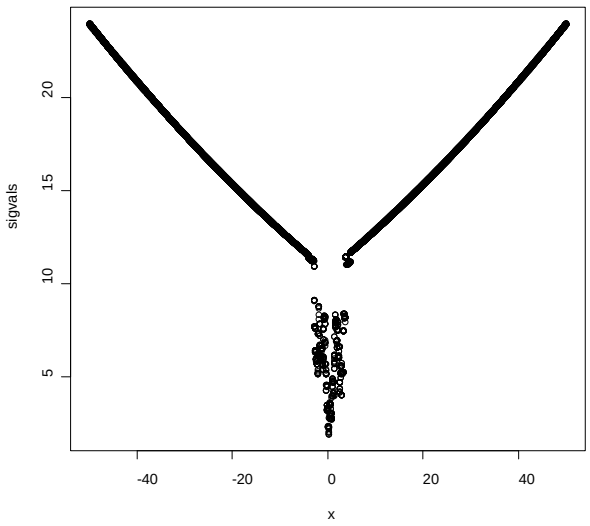


Accepted 27016 out of 100000 (27.016%).
final beta = 2.323557
alpha(0.000000) is 0.292087
alpha(2.000000) is 0.172451
alpha(5.000000) is 0.303265
alpha(10.000000) is 0.435934
alpha(20.000000) is 0.496122
alpha(50.000000) is 0.502316
alpha(100.000000) is 0.500706
alpha(150.000000) is 0.497121
it takes 128.956000 seconds to run the adaptation algorithm
it takes 57.127000 seconds to compute g and sigma
it takes 18.625000 seconds to generate the first Markov Chain
it takes 18.546000 seconds to generate the second Markov Chain
it takes 18.632000 seconds to generate the third Markov Chain
it takes 18.589000 seconds to generate the fourth Markov Chain
it takes 18.534000 seconds to generate the fifth Markov Chain
it takes 575.384000 seconds to test the local acceptance

[1] 13.51092
[1] 0.01197586
[1] 0.2913418

2.1.4.15 case 65 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

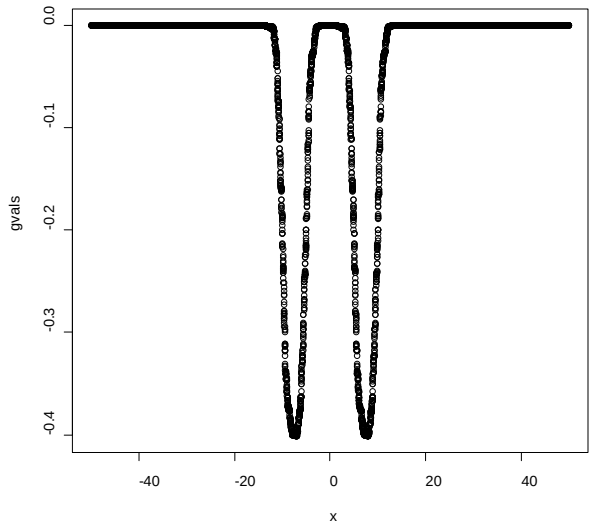


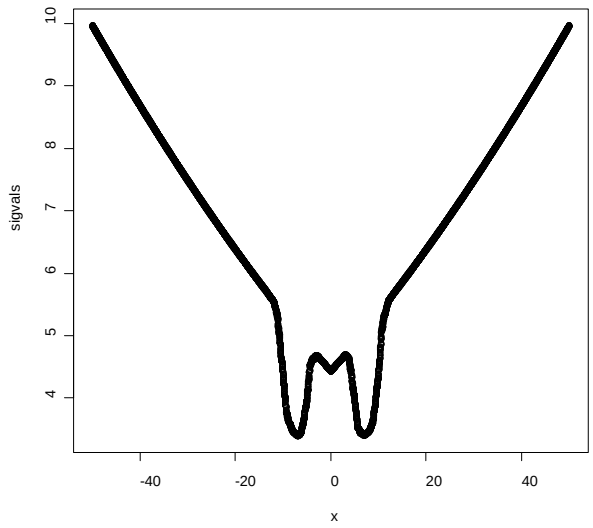


Accepted 22627 out of 100000 (22.627%).
final beta = 2.364856
alpha(0.000000) is 0.213898
alpha(2.000000) is 0.256772
alpha(5.000000) is 0.316926
alpha(10.000000) is 0.449290
alpha(20.000000) is 0.500463
alpha(50.000000) is 0.504012
alpha(100.000000) is 0.496350
alpha(150.000000) is 0.500614
it takes 128.110000 seconds to run the adaptation algorithm
it takes 56.517000 seconds to compute g and sigma
it takes 18.665000 seconds to generate the first Markov Chain
it takes 18.569000 seconds to generate the second Markov Chain
it takes 18.552000 seconds to generate the third Markov Chain
it takes 18.615000 seconds to generate the fourth Markov Chain
it takes 18.494000 seconds to generate the fifth Markov Chain
it takes 582.322000 seconds to test the local acceptance

[1] 7.970058
[1] 0.009514803
[1] 0.4789362

2.1.4.16 case 66 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$





Accepted 26560 out of 100000 (26.560%).
final beta = 1.488413
alpha(0.000000) is 0.220851
alpha(2.000000) is 0.372854
alpha(5.000000) is 0.510782
alpha(10.000000) is 0.508244
alpha(20.000000) is 0.506292
alpha(50.000000) is 0.509547
alpha(100.000000) is 0.504294
alpha(150.000000) is 0.500956
it takes 125.194000 seconds to run the adaptation algorithm
it takes 55.229000 seconds to compute g and sigma
it takes 19.441000 seconds to generate the first Markov Chain
it takes 19.359000 seconds to generate the second Markov Chain
it takes 19.212000 seconds to generate the third Markov Chain
it takes 19.485000 seconds to generate the fourth Markov Chain
it takes 19.515000 seconds to generate the fifth Markov Chain
it takes 566.340000 seconds to test the local acceptance

```
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 5.940533
[1] 0.008155242
[1] 0.610068
```

2.3.5 constant $\sigma(x) = C$

The first run:

```
> cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
> cat("acceptance rate =", num/M, "\n");
acceptance rate = 0.23481
>
> varfact <- function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
> varf = varfact(xlist[(B+1):M]);
> se = sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
> val = 0
> for(i in (B+1):M )
+    val = val +(xlist[i]-xlist[i-1])^2
> asjd = val/(M-B)
>
> varf;
[1] 6.032679
> se;
[1] 0.008154245
> asjd;
[1] 0.5538456
```

The second run:

```
> cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
> cat("acceptance rate =", num/M, "\n");
acceptance rate = 0.23477
>
> varfact <- function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
> varf = varfact(xlist[(B+1):M]);
> se = sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
> val = 0
> for(i in (B+1):M )
+    val = val +(xlist[i]-xlist[i-1])^2
> asjd = val/(M-B)
>
> varf;
[1] 6.428601
> se;
[1] 0.008419095
> asjd;
```

```
[1] 0.5549895
```

The third run:

```
> cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
> cat("acceptance rate =", num/M, "\n");
acceptance rate = 0.23366
>
> varfact <- function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
> varf = varfact(xlist[(B+1):M]);
> se = sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
> val = 0
> for(i in (B+1):M )
+    val = val +(xlist[i]-xlist[i-1])^2
> asjd = val/(M-B)
>
> varf;
[1] 6.010923
> se;
[1] 0.008097415
> asjd;
[1] 0.5370982
```

The fourth run:

```
> cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
> cat("acceptance rate =", num/M, "\n");
acceptance rate = 0.23494
>
> varfact <- function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
> varf = varfact(xlist[(B+1):M]);
> se = sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
> val = 0
> for(i in (B+1):M )
+    val = val +(xlist[i]-xlist[i-1])^2
> asjd = val/(M-B)
>
> varf;
[1] 6.143806
> se;
[1] 0.008276198
```

```
> asjd;
[1] 0.5699829
```

The fifth run:

```
> cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
> cat("acceptance rate =", num/M, "\n");
acceptance rate = 0.23071
>
> varfact <- function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
> varf = varfact(xlist[(B+1):M]);
> se = sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );
> val = 0
> for(i in (B+1):M )
+    val = val +(xlist[i]-xlist[i-1])^2
> asjd = val/(M-B)
>
> varf;
[1] 6.654467
> se;
[1] 0.008502017
> asjd;
[1] 0.5473398
```

2.1.6 varfact comparison

kernel function $K(x)=e^{-\frac{\lvert x\rvert^{\alpha_1}}{\alpha_2}}$									
α_1	C	γ	η_n	b_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	$\frac{1}{(n+5)^{0.5}}$	1	11.35909	12.05121	11.45263	10.70036	10.27269
2					12.87526	12.985	13.3621	12.5798	11.6263
10					11.95971	11.1817	12.009	11.1779	12.3533
0.1					12.4793	12.1969	12.509	12.4791	13.202
0.5					10.85975	9.60599	10.4611	11.1237	10.3478
2	2				15.7421	15.26345	15.33123	15.21730	14.3668
	10				25.09042	30.3672	26.7781	28.74199	28.38106
	0.1				6.533223	6.42219	6.045298	6.83526	6.91115
	0.5				9.801138	9.868752	10.06519	10.29492	10.2363
	0.01				6.521881	6.590561	7.095979	5.971771	6.151567
	0.1	10			12.03479	11.71392	10.21051	10.77967	10.77254
		0.5			6.6764022	6.618857	6.412747	6.739889	6.701922
		1			7.184957	7.440486	7.480085	7.304401	8.057888
		0.1			7.54	7.650029	6.898279	7.697511	7.061558
		2			7.007102	7.612522	7.921151	7.508852	7.411572
			$\frac{1}{(n+5)^{0.8}}$	10	6.689204	6.323629	6.023003	6.549738	6.704092
				0.1	7.638526	7.725935	8.350977	7.279248	7.577468
				$\frac{1}{n^{0.2}}$	7.705133	7.408075	7.525139	7.189149	7.331579

The best case is:

case 8 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=2,\alpha_2=1,C=0.1,\gamma=2$ with fixed bandwidth $b_n=1$

kernel function $K(x)=\frac{1}{1+\alpha_1\lvert x\rvert^{\alpha_2}}$									
α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	10.69007	10.81582	11.26665	11.75963	10.69007
10					10.281	11.16265	11.00003	10.15537	12.06744
0.1					12.01854	14.03712	12.95212	12.25375	12.01854
1	10				26.62614	27.94747	28.96005	30.58577	26.62614
	0.1				7.397457	7.050885	6.675549	7.357388	7.397457
	0.01				6.515635	7.225982	6.216325	6.595525	6.515635
	0.1	1			7.185476	6.92573	6.877931	7.298966	7.185476
		10			9.680703	9.388705	10.12533	11.79035	9.680703
		2	10		6.725084	6.679757	6.763915	7.555752	6.725084
			0.1		7.909221	7.823185	8.079191	8.019257	7.909221
		0.1	10		6.306029	6.253697	5.868809	6.429638	6.306029
		2	$\frac{1}{n^{0.2}}$		6.861169	7.47218	7.089262	7.616252	6.861169

			10	$\frac{1}{(n+5)^{0.8}}$	6.258227	5.779044	5.963561	6.285545	6.258227

The best case is:

case 25 $\frac{1}{(n+5)^{0.8}}$ $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 10$

kernel function $K(x) = \begin{cases} 0, \mid x \mid \geq 2 * width \\ -1 * hegith, width < \mid x \mid < 2 * width \\ 1 * height, \mid x \mid \leq width \end{cases}$

C	γ	height	width	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	0.5	0.5	1	$\frac{1}{(n+5)^{0.5}}$	11.21697	11.01346	10.73923	11.00212	10.74114
10						26.36981	27.70317	25.53938	24.13856	27.30234
0.1						6.388425	6.44163	6.432831	6.348009	6.251577
0.01						5.714605	6.093879	6.027745	6.093879	6.161878
0						6.769783	6.339817	7.0314341	6.552781	6.861094
0.001						6.480792	6.6654161	6.420364	6.372943	6.769786
0.01	10					5.84989	6.229419	5.915948	6.139981	6.5259
	1					6.338966	6.507104	5.854828	5.9643191	6.692964
	0.1					6.755861	6.746315	6.918906	6.680906	6.49034
	2	1				6.53588	5.974104	6.804957	6.391518	6.555601
		10				9.536619	9.63294	9.340004	9.086103	9.308285
		0.1				6.388736	5.793102	6.651792	6.726921	6.362033
		0.5	2			6.989622	6.410413	6.045172	6.436296	7.392876
			5			6.83254	6.885814	6.679356	6.848712	6.854895
			0.1			6.610762	5.995149	6.148899	6.304812	6.081314
			0.5	10		6.07822	6.375568	6.301751	6.438769	6.359084
				0.1		6.460936	6.708183	6.249199	6.212332	6.130305
				$\frac{1}{n^{0.2}}$		6.5368	7.150534	7.179354	7.156782	7.01146
				1	$\frac{1}{(n+5)^{0.8}}$	6.5368	7.150534	7.179354	7.156782	7.01146

The best case is:

case 38 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.01, \gamma = 10$ with fixed bandwidth $b_n = 1$

Kernel function $K(x) = \begin{cases} 0, \mid x \mid \geq 2 * width \quad or \quad \mid x \mid \leq width \\ 1, width < \mid x \mid < 2 * width \end{cases}$

C	width	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	0.5	2	1	$\frac{1}{(n+5)^{0.5}}$	11.8298	11.3176	11.76312	11.11506	11.34705
10					12.68587	12.2002	12.52693	12.71785	12.73849
0.1					6.968693	7.780381	7.12373	6.84579	7.425369

0.01					6.325229	6.238655	6.788932	6.642527	6.352072
0.001					6.632203	6.651379	6.905611	6.9814	7.055358
	1				7.228149	6.63247	6.819531	6.440049	7.293104
	5				5.72553	5.958359	5.350588	5.822084	5.500506
	0.1				6.305438	6.342151	6.781963	6.660485	6.456669
	8				7.512562	6.160492	6.884363	6.907896	6.965559
		5			6.199165	6.789928	6.931863	6.463562	6.651483
		1			6.551915	6.891536	6.335435	6.279825	6.598833
0.01		0.1			5.794013	6.270467	6.320622	6.509231	6.12154
			10		6.704157	6.904249	6.711725	6.449955	6.666041
	5		0.1		14.51732	13.42962	13.60299	13.92906	13.51092
		2	$\frac{1}{n^{0.2}}$		7.597946	7.456866	7.910384	7.821029	7.970058
			1	$\frac{1}{(n+5)^{0.8}}$	5.377442	5.680714	5.344205	5.464543	5.940533

The best case is:

case 57 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

constant $\sigma(x) = C$				
First run	Second run	Third run	Fourth run	Fifth run
6.032679	6.428601	6.010923	6.143806	6.654467

2.1.7 variance comparison

kernel function $K(x)=e^{-\frac{\lvert x\rvert^{\alpha_1}}{\alpha_2}}$									
α_1	C	γ	η_n	b_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	$\frac{1}{(n+5)^{0.5}}$	1	0.01122983	0.01172226	0.01136355	0.01089844	0.01078979
2					0.0118463	0.0120133	0.0122284	0.0116674	0.0111596
10					0.011360	0.0111945	0.0117056	0.0110526	0.0115645
0.1					0.0117178	0.0114601	0.0118072	0.0118785	0.0121816
0.5					0.0110567	0.0104420	0.0109040	0.0109978	0.0107297
2	2				0.0133701	0.0133216	0.0130189	0.0130471	0.0124461
	10				0.0165678	0.0185194	0.0171344	0.0175987	0.0175510
	0.1				0.0086311	0.0083967	0.0081392	0.0087408	0.0086971
	0.5				0.0102999	0.0103018	0.0107431	0.0105740	0.010764
	0.01				0.00852	0.0085	0.00899	0.00812	0.00832
	0.1	10			0.0114	0.01135	0.01074	0.01113	0.01087
		0.5			0.00871	0.00862	0.00835	0.00864	0.0087
		1			0.00892	0.00901	0.0092	0.0091	0.00958
		0.1			0.00909	0.00922	0.00868	0.00919	0.00885
		2			0.00892	0.00919	0.00935	0.00916	0.009
				10	0.00863	0.00828	0.00813	0.00844	0.00855
				0.1	0.00931	0.00922	0.00956	0.0089	0.00915
			$\frac{1}{(n+5)^{0.8}}$	$\frac{1}{n^{0.2}}$	0.0093	0.00907	0.00906	0.00892	0.00883

The best case is:

case 8 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=2,\alpha_2=1,C=0.1,\gamma=2$ with fixed bandwidth $b_n=1$

kernel function $K(x)=\frac{1}{1+\alpha_1\lvert x\rvert^{\alpha_2}}$									
α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	0.01080392	0.01088505	0.01107906	0.01174345	0.01129308
10					0.0107729	0.01107101	0.01115669	0.01045884	0.01162421
0.1					0.0114314	0.01282429	0.01201123	0.0115313	0.01155291
1	10				0.01681819	0.01778243	0.01751706	0.01876566	0.01656713
	0.1				0.009064997	0.008835046	0.008520924	0.009064476	0.008521907
	0.01				0.008606985	0.008944422	0.008234797	0.008487706	0.008421049
	0.1	1			0.00892049	0.008758606	0.00868717	0.008971944	0.009006958
		10			0.01042457	0.01022211	0.01077376	0.01165331	0.01055025
		2	10		0.008629022	0.008642	0.008714656	0.009189865	0.008970417
			0.1		0.009410435	0.009324346	0.009459222	0.00931274	0.009549854
		0.1	10		0.008361173	0.008262193	0.008130423	0.008468772	0.008328171
		2	$\frac{1}{n^{0.2}}$		0.00870455	0.00917211	0.00879794	0.00920209	0.00879953

			10	$\frac{1}{(n+5)^{0.8}}$	0.00827424	0.00808351	0.00806594	0.00835063	0.00828150

The best case is:

case 25 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 1$ with fixed bandwidth $b_n = 1$

kernel function $K(x) = \begin{cases} 0, \mid x \mid \geq 2 * width \\ -1 * hegith, width < \mid x \mid < 2 * width \\ 1 * height, \mid x \mid \leq width \end{cases}$

C	γ	height	width	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	0.5	0.5	1	$\frac{1}{(n+5)^{0.5}}$	0.01119531	0.01102779	0.01097638	0.01111038	0.01082565
10						0.01708748	0.01719778	0.01633589	0.01621996	0.01750065
0.1						0.00832147	0.00850810	0.00846442	0.00831120	0.00830861
0.01						0.00796456	0.00818376	0.0081465	0.00818376	0.00825706
0						0.00873404	0.00844347	0.00890642	0.00872350	0.00878673
0.001						0.00849819	0.00867371	0.00841714	0.00832729	0.00862447
0.01	10					0.00805243	0.00827595	0.00817826	0.00814704	0.00847703
	1					0.0084309	0.0085183	0.00791077	0.00811096	0.00858695
	0.1					0.0086998	0.00870021	0.00871831	0.00871517	0.00856057
	2	1				0.00857281	0.00813515	0.00881743	0.00841922	0.00863647
		10				0.01022875	0.01028774	0.0103866	0.00999755	0.01020134
		0.1				0.00847153	0.00808108	0.00859385	0.00857070	0.00829111
		0.5	2			0.00883611	0.00846877	0.00813382	0.00846513	0.00902233
			5			0.5247985	0.5225287	0.5246484	0.5300015	0.5201393
			0.1			0.00867055	0.00873511	0.00868802	0.00864770	0.0087029
			0.5	10		0.00858705	0.00829408	0.00832144	0.00814989	0.00825206
				0.1		0.0081262	0.00833905	0.0082930	0.0084702	0.0084230
				$\frac{1}{n^{0.2}}$		0.00843647	0.00861249	0.0083173	0.0081667	0.00825173
				1	$\frac{1}{(n+5)^{0.8}}$	0.00856975	0.00877584	0.00891550	0.00881037	0.00888963

The best case is:

case 38 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.001, \gamma = 10$ with fixed bandwidth $b_n = 1$

Kernel function $K(x) = \begin{cases} 0, \mid x \mid \geq 2 * width \quad or \quad \mid x \mid \leq width \\ 1, width < \mid x \mid < 2 * width \end{cases}$

C	width	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	0.5	2	1	$\frac{1}{(n+5)^{0.5}}$	0.01145177	0.01115331	0.01128486	0.01087948	0.01121087
10					0.01205766	0.01178141	0.01198769	0.01184585	0.01200023
0.1					0.008753066	0.009387937	0.00893782	0.008729863	0.00926486
0.01					0.008346846	0.008280165	0.008669679	0.008634343	0.008441499

0.001					0.008636908	0.00860409	0.008797365	0.008743271	0.008838765
0.01	1				0.008857288	0.008613555	0.008615749	0.008473382	0.008969298
	5				0.00792509	0.008171163	0.007620757	0.008002132	0.007780254
	0.1				0.008363918	0.008285638	0.008547047	0.008700804	0.008401081
	8				0.009170728	0.008328777	0.008684492	0.008594844	0.008812787
	5	5			0.008384142	0.008724913	0.008774852	0.008414547	0.008605778
		1			0.008558539	0.008789663	0.008396568	0.008364616	0.008443037
		0.1			0.008001875	0.008395209	0.00845136	0.008474585	0.00825666
		2	10		0.00857308	0.008749343	0.008612675	0.00850364	0.008627651
			0.1		0.01280593	0.01232926	0.01235656	0.01234854	0.0119758
			$\frac{1}{n^{0.2}}$		0.009259425	0.009196404	0.009356108	0.009326126	0.009514803
			1	$\frac{1}{(n+5)^{0.8}}$	0.00780106	0.007956023	0.007668444	0.007867795	0.008155242

The best case is:

case 57 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$

constant $\sigma(x) = C$				
First run	Second run	Third run	Fourth run	Fifth run
0.008154245	0.008419095	0.008097415	0.008276198	0.008502017

2.1.8 comparison of average squared jump distance

kernel function $K(x)=e^{-\frac{\lvert x\rvert^{\alpha_1}}{\alpha_2}}$									
α_1	C	γ	η_n	b_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	$\frac{1}{(n+5)^{0.5}}$	1	0.3482317	0.3378794	0.3383667	0.3463196	0.33542
2					0.3020888	0.3106892	0.3051376	0.3061098	0.309262
10					0.3301311	0.3358699	0.3388592	0.3208399	0.3275879
0.1					0.2985020	0.3010016	0.3049649	0.3073156	0.303518
0.5					0.3499498	0.3523485	0.3627218	0.3406725	0.3623652
2	2				0.270866	0.2717923	0.2693725	0.2657615	0.2595380
	10				0.1661419	0.1613741	0.166271	0.1561579	0.1614916
	0.1				0.545549	0.5252874	0.5203305	0.5352214	0.5301068
	0.5				0.3768554	0.3621124	0.3841471	0.3711383	0.3784802
	0.01				0.486626	0.47211	0.49826	0.49613	0.49295
	0.1	10			0.35078	0.35504	0.37335	0.36722	0.35932
		0.5			0.52328	0.51853	0.51545	0.52934	0.52331
		1			0.48663	0.47211	0.49826	0.49613	0.49295
		0.1			0.48265	0.48159	0.48307	0.47591	0.4898
		2			0.494592	0.48125	0.47849	0.47118	0.477
			$\frac{1}{(n+5)^{0.8}}$	10	0.534894	0.52604	0.52935	0.51976	0.51584
				0.1	0.458733	0.45319	0.45694	0.44814	0.45777
				$\frac{1}{n^{0.2}}$	0.486247	0.48948	0.48803	0.48851	0.48079

The best case is:

case 8 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=2,\alpha_2=1,C=0.1,\gamma=2$ with fixed bandwidth $b_n=1$

kernel function $K(x)=\frac{1}{1+\alpha_1\lvert x\rvert^{\alpha_2}}$									
α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	0.3387809	0.3446142	0.3314934	0.3495083	0.3478964
10					0.3506148	0.3349281	0.342898	0.3357567	0.333423
0.1					0.3169546	0.3205971	0.3261828	0.3104705	0.3238721
1	10				0.159876	0.1661034	0.1576704	0.1617906	0.1643911
	0.1				0.4906771	0.4946082	0.4807281	0.4897404	0.4765741
	0.01				0.5466972	0.5388166	0.5297919	0.5527384	0.5429365
	0.1	1			0.5146352	0.5145474	0.5087591	0.4962806	0.5071021
		10			0.4082753	0.4054954	0.4134897	0.410045	0.3960828
		2	10		0.5297551	0.5084722	0.5256346	0.5121834	0.5260427
			0.1		0.447282	0.437752	0.4453548	0.43688	0.4514627
		0.1	10		0.5614798	0.5495616	0.5841707	0.5634893	0.5697796
		2	$\frac{1}{n^{0.2}}$		0.498937	0.5065231	0.485246	0.488642	0.4994858

			10	$\frac{1}{(n+5)^{0.8}}$	0.5487379	0.5700657	0.5551277	0.5553396	0.5653669

The best case is:

case 29 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth $b_n = 10$

kernel function $K(x) = \begin{cases} 0, \mid x \mid \geq 2 * width \\ -1 * hegith, width < \mid x \mid < 2 * width \\ 1 * height, \mid x \mid \leq width \end{cases}$

C	γ	height	width	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	0.5	0.5	1	$\frac{1}{(n+5)^{0.5}}$	0.3480791	0.3393188	0.3521589	0.3414209	0.3407953
10						0.1717606	0.1693742	0.169111	0.1736697	0.1734251
0.1						0.5400879	0.5459329	0.5532259	0.5478521	0.5388973
0.01						0.5832376	0.5765046	0.5731739	0.5765046	0.5782485
0						0.5437982	0.5434355	0.5379284	0.5601083	0.5409318
0.001						0.5554767	0.5492348	0.5508303	0.5473217	0.5400489
0.01	10					0.5757201	0.5650518	0.5851384	0.5462827	0.555975
	1					0.5539996	0.5513252	0.52823	0.5416636	0.5457672
	0.1					0.531748	0.5351088	0.5204629	0.5416667	0.5352067
	2	1				0.5418091	0.536561	0.5630413	0.548019	0.5541553
		10				0.4327383	0.4517592	0.4476535	0.4391682	0.4585032
		0.1				0.5548144	0.572752	0.5579829	0.5399683	0.5503619
		0.5	2			0.5230771	0.5227794	0.5236362	0.5200925	0.5164709
			5			0.5615815	0.5478925	0.5628432	0.5608967	0.5518468
			0.1			0.5533189	0.5475685	0.5356417	0.5613171	0.554771
			0.5	10		0.5306976	0.5047632	0.5114743	0.510878	0.5306071
				0.1		0.5306976	0.5047632	0.5114743	0.5108781	0.5306071
				$\frac{1}{n^{0.2}}$		0.3480791	0.3393188	0.3521589	0.3414209	0.3407953
				1	$\frac{1}{(n+5)^{0.8}}$	0.1717606	0.1693742	0.169111	0.1736697	0.1734251

The best case is:

case 38 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $height = 0.5, \text{width} = 0.5, C = 0.001, \gamma = 10$ with fixed bandwidth $b_n = 1$

Kernel function $K(x) = \begin{cases} 0, \mid x \mid \geq 2 * width \quad or \quad \mid x \mid \leq width \\ 1, width < \mid x \mid < 2 * width \end{cases}$

C	width	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	0.5	2	1	$\frac{1}{(n+5)^{0.5}}$	0.316954	0.3105255	0.3226398	0.3064963	0.3157405
10					0.4793217	0.4780706	0.4868769	0.4790695	0.4934043
0.1					0.5616479	0.562497	0.5484761	0.5583827	0.5698832
0.01					0.5399391	0.5072552	0.5181693	0.5078605	0.5103956

0.001					0.5201501	0.5321433	0.5283818	0.5313193	0.5260217
0.01	1				0.5872723	0.0605619	0.5860981	0.5982348	0.5968249
	5				0.5499262	0.5435791	0.5322315	0.5504899	0.5444502
	0.1				0.537842	0.546856	0.5303485	0.5117296	0.5334517
	8				0.5624894	0.5548582	0.5508696	0.538276	0.5450531
	5	5			0.5510387	0.5559884	0.5598121	0.5568019	0.5469699
		1			0.5684341	0.5686782	0.5763505	0.5646907	0.5646845
		0.1			0.5350034	0.5295421	0.5335563	0.5449261	0.541695
		2	10		0.3080717	0.3065731	0.316869	0.3040506	0.2913418
			0.1		0.4785666	0.475891	0.4682705	0.4743092	0.4789362
			$\frac{1}{n^{0.2}}$		0.6113059	0.5972358	0.6040591	0.6167911	0.610068
			1	$\frac{1}{(n+5)^{0.8}}$	0.316954	0.3105255	0.3226398	0.3064963	0.3157405

The best case is:

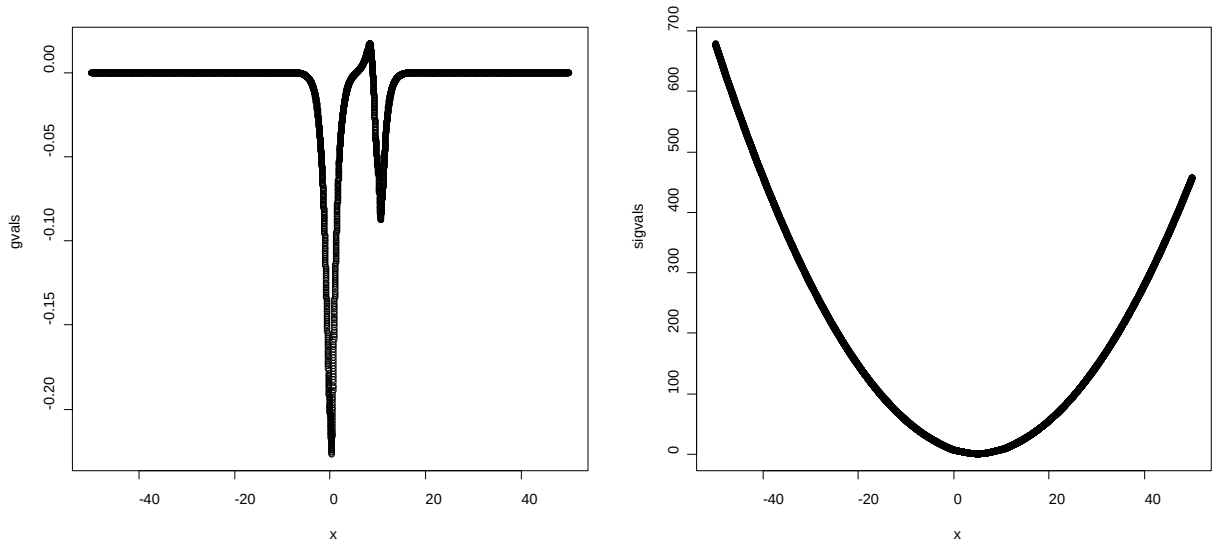
case 65 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

constant $\sigma(x) = C$				
First run	Second run	Third run	Fourth run	Fifth run
0.5538456	0.5549895	0.5370982	0.5699829	0.5473398

2.2 Example 2: mixture of two normal distributions in R^1

2.2.1 kernel function $K(x) = e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

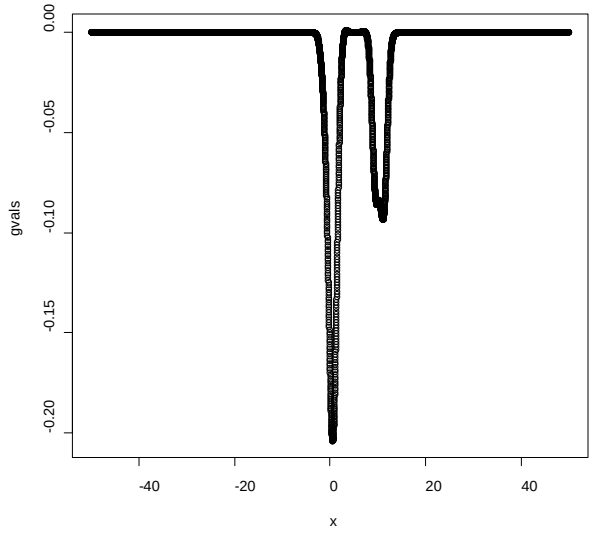
2.2.1.1 case 67 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

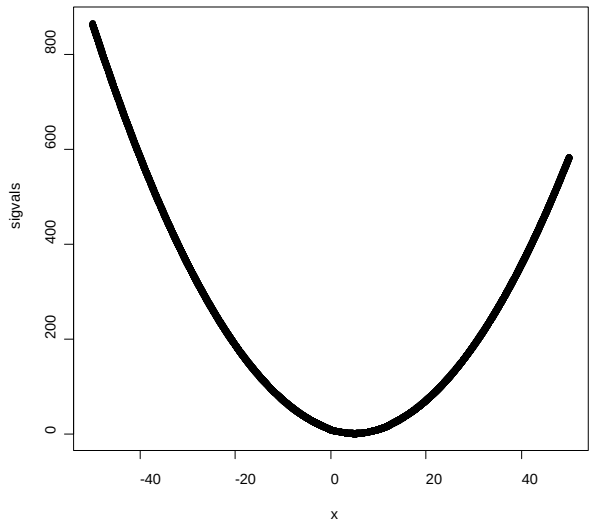


Accepted 23354 out of 100000 (23.354%).
final beta = -1.530327
alpha(5.000000) is 0.981209
alpha(7.000000) is 0.595073
alpha(10.000000) is 0.183040
alpha(20.000000) is 0.182034
alpha(30.000000) is 0.125577
alpha(50.000000) is 0.081246
alpha(70.000000) is 0.054118
alpha(120.000000) is 0.032300
alpha(150.000000) is 0.024200
it takes 211.785000 seconds to run the adaptation algorithm
it takes 93.757000 seconds to compute g and sigma
it takes 2380.770000 seconds to generate the first Markov Chain
it takes 2437.619000 seconds to generate the second Markov Chain
it takes 2437.086000 seconds to generate the third Markov Chain
it takes 2367.840000 seconds to generate the fourth Markov Chain
it takes 2366.400000 seconds to generate the fifth Markov Chain
it takes 3924.334000 seconds to test the local acceptance

[1] 17.14759
[1] 0.07038708
[1] 5.778095

2.2.1.2 case 68 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

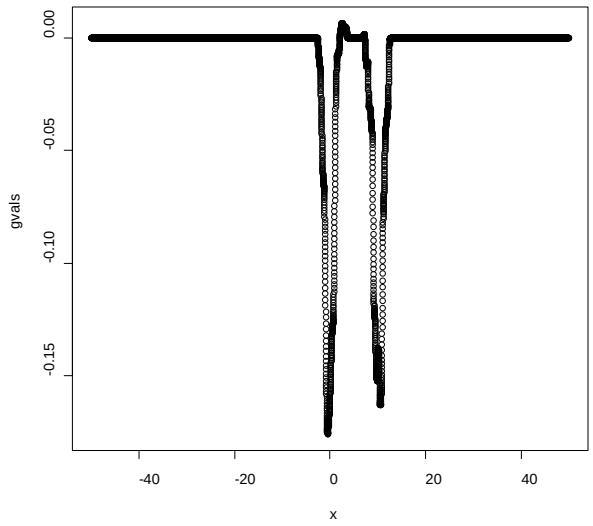


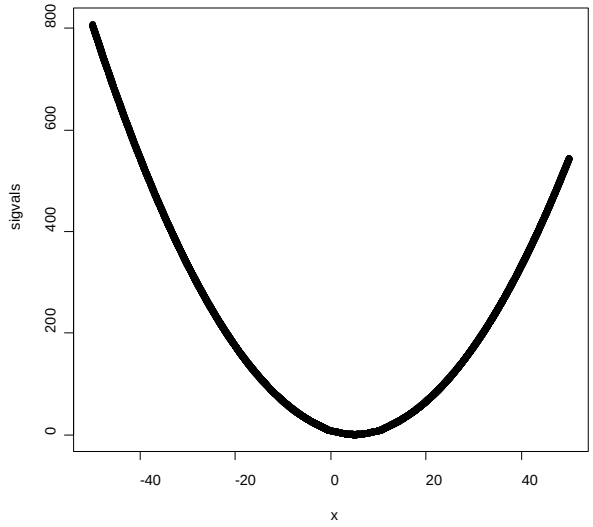


Accepted 21366 out of 100000 (21.366%).
final beta = -1.287503
alpha(5.000000) is 0.989037
alpha(7.000000) is 0.623838
alpha(10.000000) is 0.167483
alpha(20.000000) is 0.152847
alpha(30.000000) is 0.104381
alpha(50.000000) is 0.053026
alpha(70.000000) is 0.042200
alpha(120.000000) is 0.024900
alpha(150.000000) is 0.020700
it takes 213.984000 seconds to run the adaptation algorithm
it takes 140.214000 seconds to compute g and sigma
it takes 21.567000 seconds to generate the first Markov Chain
it takes 22.135000 seconds to generate the second Markov Chain
it takes 21.171000 seconds to generate the third Markov Chain
it takes 21.428000 seconds to generate the fourth Markov Chain
it takes 21.561000 seconds to generate the fifth Markov Chain
it takes 3266.257000 seconds to test the local acceptance

[1] 13.98678
[1] 0.06355741
[1] 6.943634

2.2.1.3 case 69 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

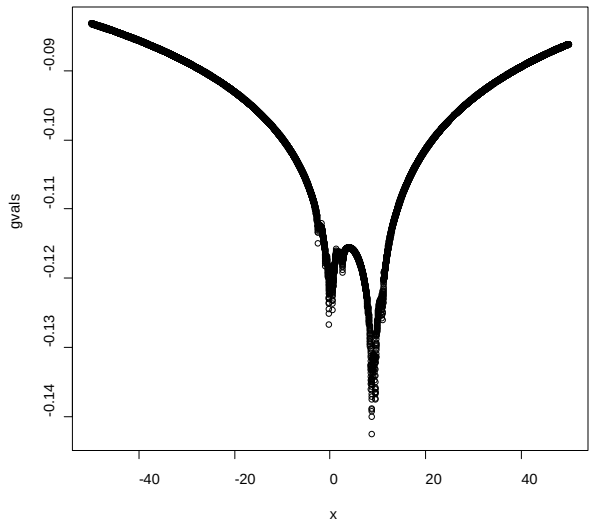


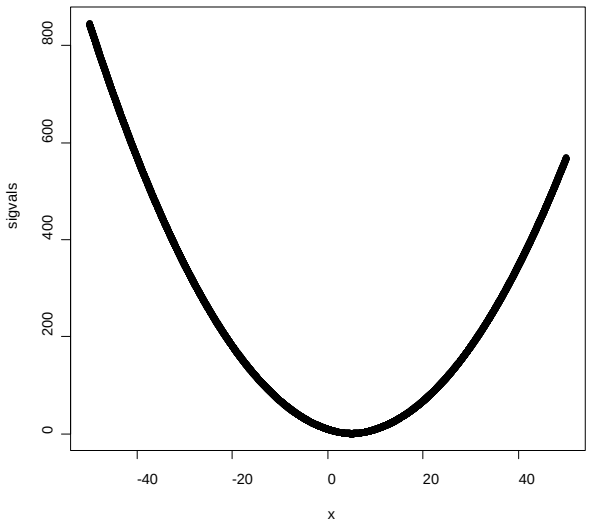


Accepted 21953 out of 100000 (21.953%).
final beta = -1.356750
alpha(5.000000) is 0.988818
alpha(7.000000) is 0.610090
alpha(10.000000) is 0.176504
alpha(20.000000) is 0.159430
alpha(30.000000) is 0.108443
alpha(50.000000) is 0.058606
alpha(70.000000) is 0.043800
alpha(120.000000) is 0.026200
alpha(150.000000) is 0.023200
it takes 286.940000 seconds to run the adaptation algorithm
it takes 135.983000 seconds to compute g and sigma
it takes 21.773000 seconds to generate the first Markov Chain
it takes 21.542000 seconds to generate the second Markov Chain
it takes 22.802000 seconds to generate the third Markov Chain
it takes 21.843000 seconds to generate the fourth Markov Chain
it takes 21.417000 seconds to generate the fifth Markov Chain
it takes 2777.904000 seconds to test the local acceptance

[1] 15.06675
[1] 0.06591678
[1] 6.664103

.2.1.4 case 70 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$



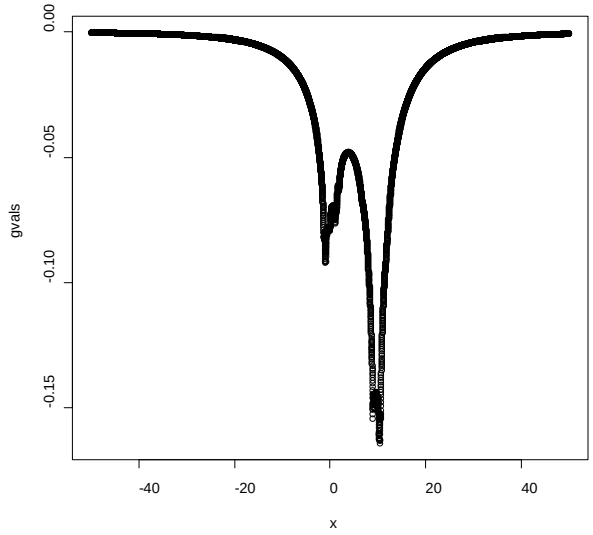


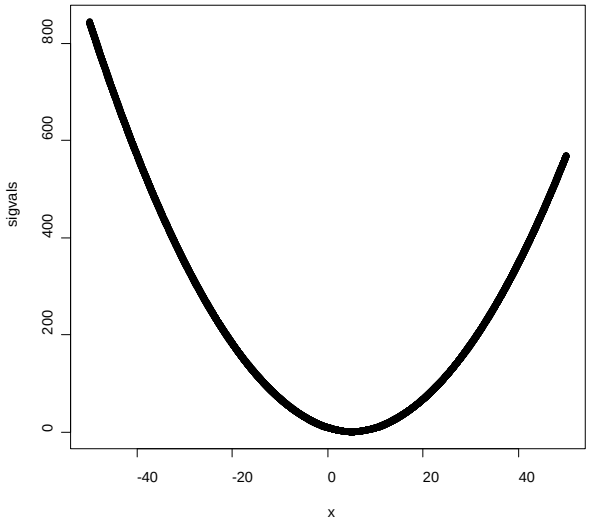
Accepted 21238 out of 100000 (21.238%).
final beta = -1.227692
alpha(5.000000) is 0.988853
alpha(7.000000) is 0.600787
alpha(10.000000) is 0.173634
alpha(20.000000) is 0.155803
alpha(30.000000) is 0.103307
alpha(50.000000) is 0.061476
alpha(70.000000) is 0.044889
alpha(120.000000) is 0.023100
alpha(150.000000) is 0.020700
it takes 238.775000 seconds to run the adaptation algorithm
it takes 104.928000 seconds to compute g and sigma
it takes 23.019000 seconds to generate the first Markov Chain
it takes 22.909000 seconds to generate the second Markov Chain
it takes 22.931000 seconds to generate the third Markov Chain
it takes 23.262000 seconds to generate the fourth Markov Chain
it takes 23.417000 seconds to generate the fifth Markov Chain
it takes 2066.982000 seconds to test the local acceptance

[1] 13.59459
[1] 0.06266335
[1] 7.059283

2.2.1.5 case 71 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.5, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

$b_n = 1$



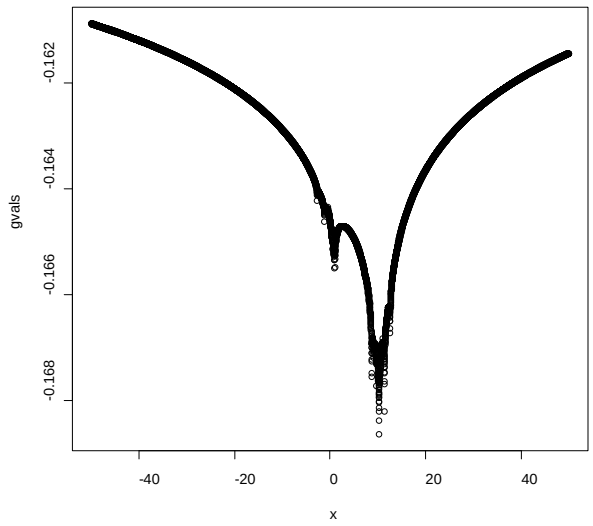


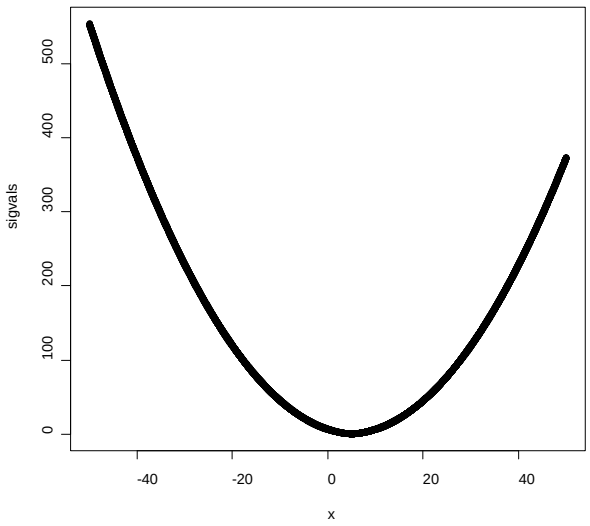
Accepted 21238 out of 100000 (21.238%).
final beta = -1.227692
alpha(5.000000) is 0.988853
alpha(7.000000) is 0.600787
alpha(10.000000) is 0.173634
alpha(20.000000) is 0.155803
alpha(30.000000) is 0.103307
alpha(50.000000) is 0.061476
alpha(70.000000) is 0.044889
alpha(120.000000) is 0.023100
alpha(150.000000) is 0.020700
it takes 238.775000 seconds to run the adaptation algorithm
it takes 104.928000 seconds to compute g and sigma
it takes 23.019000 seconds to generate the first Markov Chain
it takes 22.909000 seconds to generate the second Markov Chain
it takes 22.931000 seconds to generate the third Markov Chain
it takes 23.262000 seconds to generate the fourth Markov Chain
it takes 23.417000 seconds to generate the fifth Markov Chain
it takes 2066.982000 seconds to test the local acceptance

[1] 13.72507
[1] 0.06299655
[1] 6.884881

2.2.1.6 case 72 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.01, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

$b_n = 1$



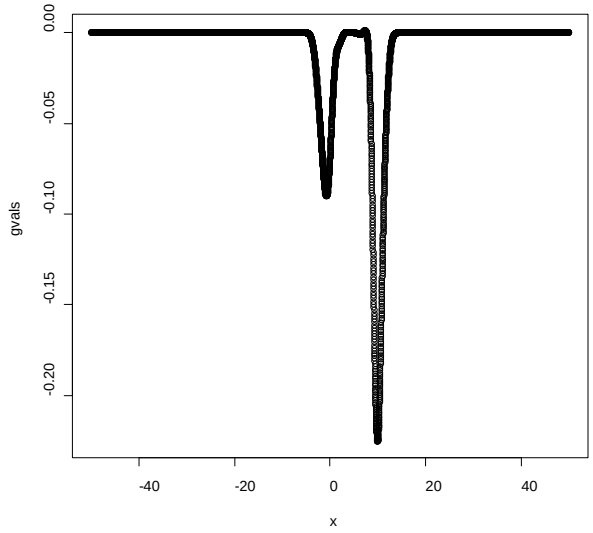


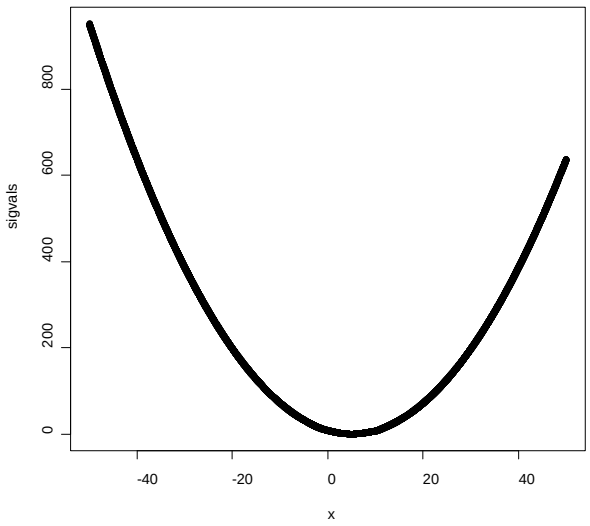
Accepted 24931 out of 100000 (24.931%).
final beta = -1.573366
alpha(5.000000) is 0.988528
alpha(7.000000) is 0.600182
alpha(10.000000) is 0.210625
alpha(20.000000) is 0.218254
alpha(30.000000) is 0.156754
alpha(50.000000) is 0.093247
alpha(70.000000) is 0.066904
alpha(120.000000) is 0.039000
alpha(150.000000) is 0.030100
it takes 227.191000 seconds to run the adaptation algorithm
it takes 102.547000 seconds to compute g and sigma
it takes 21.391000 seconds to generate the first Markov Chain
it takes 20.529000 seconds to generate the second Markov Chain
it takes 20.712000 seconds to generate the third Markov Chain
it takes 22.904000 seconds to generate the fourth Markov Chain
it takes 20.531000 seconds to generate the fifth Markov Chain
it takes 1969.599000 seconds to test the local acceptance

[1] 21.60776
[1] 0.0791383
[1] 4.975623

2.2.1.7 case 73 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 10, \gamma = 2$ with fixed bandwidth

$b_n = 1$



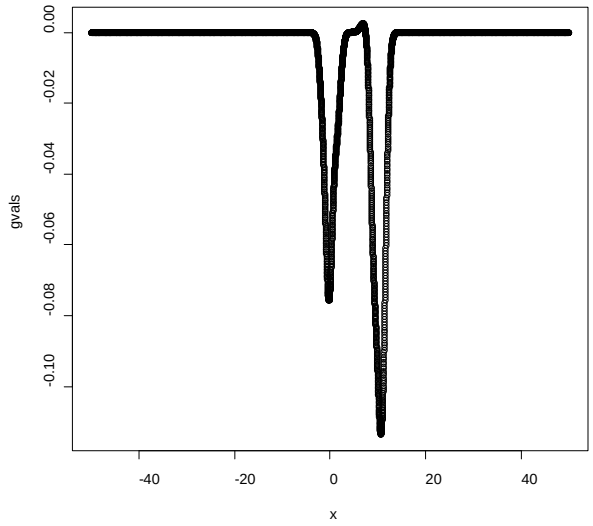


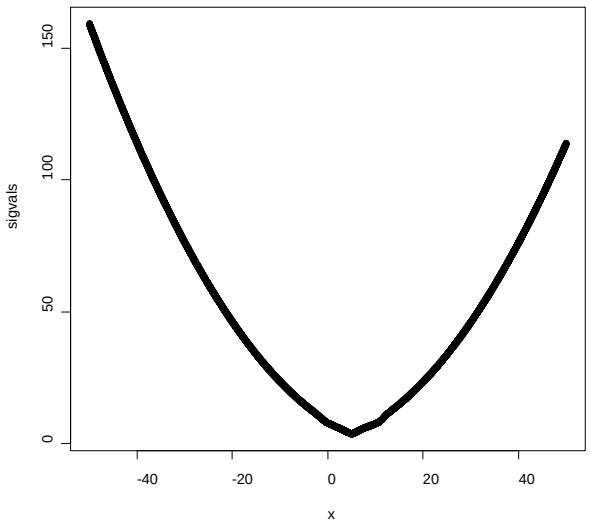
Accepted 23009 out of 100000 (23.009%).
final beta = -5.765632
alpha(5.000000) is 0.958174
alpha(7.000000) is 0.597633
alpha(10.000000) is 0.198928
alpha(20.000000) is 0.139649
alpha(30.000000) is 0.087800
alpha(50.000000) is 0.055351
alpha(70.000000) is 0.038322
alpha(120.000000) is 0.023300
alpha(150.000000) is 0.016500
it takes 220.284000 seconds to run the adaptation algorithm
it takes 139.986000 seconds to compute g and sigma
it takes 20.836000 seconds to generate the first Markov Chain
it takes 20.884000 seconds to generate the second Markov Chain
it takes 20.730000 seconds to generate the third Markov Chain
it takes 20.737000 seconds to generate the fourth Markov Chain
it takes 20.583000 seconds to generate the fifth Markov Chain
it takes 3317.109000 seconds to test the local acceptance

[1] 18.24566
[1] 0.07269454
[1] 5.745729

2.2.1.8 case 74 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 1$



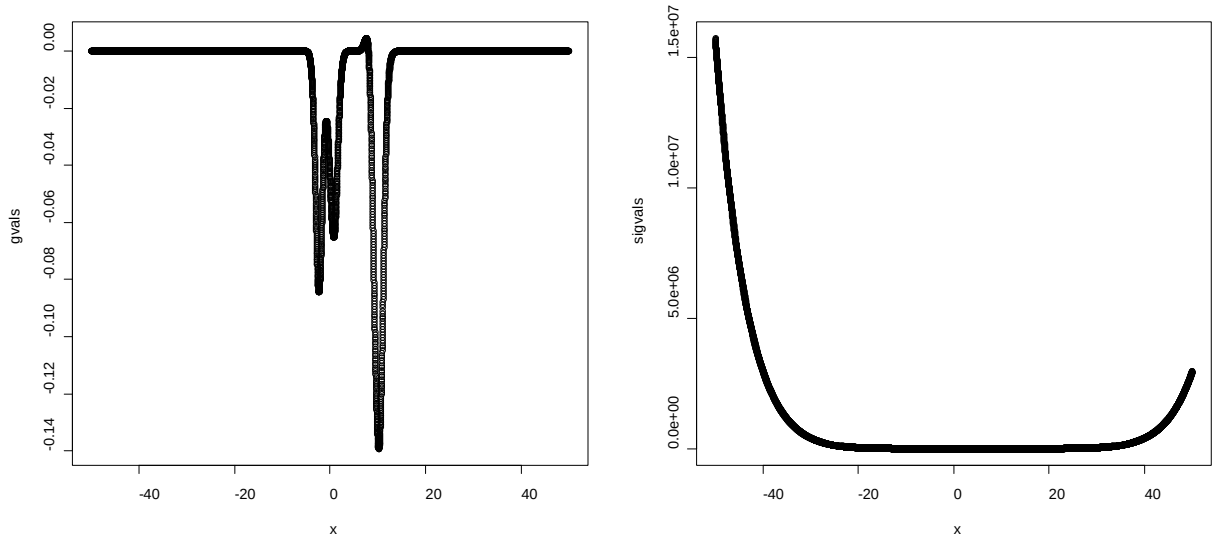


Accepted 23165 out of 100000 (23.165%).
final beta = 1.326566
alpha(5.000000) is 0.994617
alpha(7.000000) is 0.626367
alpha(10.000000) is 0.179817
alpha(20.000000) is 0.407404
alpha(30.000000) is 0.364035
alpha(50.000000) is 0.281372
alpha(70.000000) is 0.229441
alpha(120.000000) is 0.146200
alpha(150.000000) is 0.125800
it takes 225.062000 seconds to run the adaptation algorithm
it takes 147.438000 seconds to compute g and sigma
it takes 21.056000 seconds to generate the first Markov Chain
it takes 21.647000 seconds to generate the second Markov Chain
it takes 21.773000 seconds to generate the third Markov Chain
it takes 21.205000 seconds to generate the fourth Markov Chain
it takes 21.122000 seconds to generate the fifth Markov Chain
it takes 2874.449000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 12.56142
[1] 0.06021042
[1] 7.484434

2.2.1.9 case 75 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 10$ with fixed bandwidth

$b_n = 1$

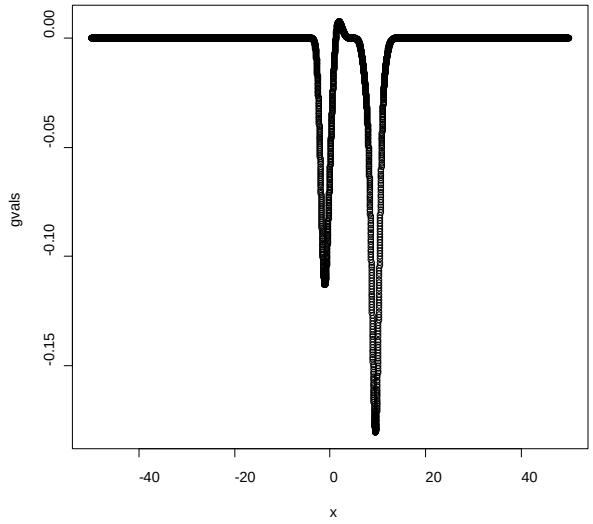


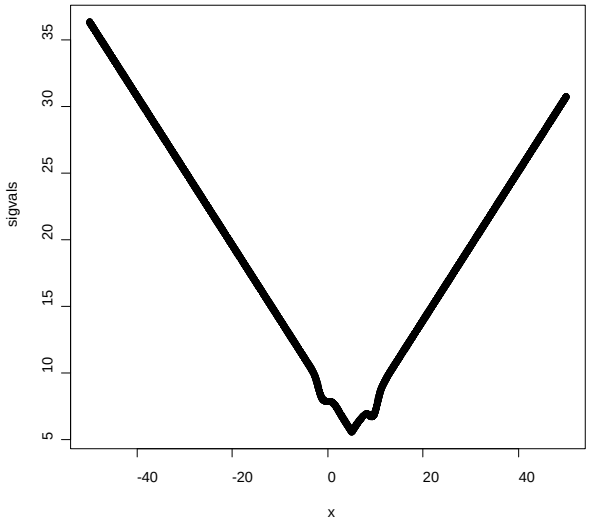
Accepted 23299 out of 100000 (23.299%).
final beta = -2.146768
alpha(5.000000) is 0.998253
alpha(7.000000) is 0.659461
alpha(10.000000) is 0.201391
alpha(20.000000) is 0.008940
alpha(30.000000) is 0.000600
alpha(50.000000) is 0.000000
alpha(70.000000) is 0.000000
alpha(120.000000) is 0.000000
alpha(150.000000) is 0.000000
it takes 217.680000 seconds to run the adaptation algorithm
it takes 145.329000 seconds to compute g and sigma
it takes 27.535000 seconds to generate the first Markov Chain
it takes 29.751000 seconds to generate the second Markov Chain
it takes 27.858000 seconds to generate the third Markov Chain
it takes 28.172000 seconds to generate the fourth Markov Chain
it takes 29.909000 seconds to generate the fifth Markov Chain
it takes 3552.758000 seconds to test the local acceptance

[1] 33.06702
[1] 0.09774311
[1] 3.609716

2.2.1.10 case 76 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 1$ with fixed bandwidth

$b_n = 1$



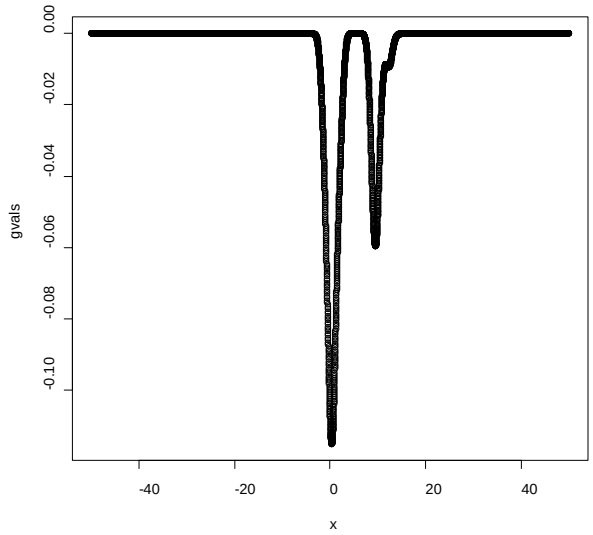


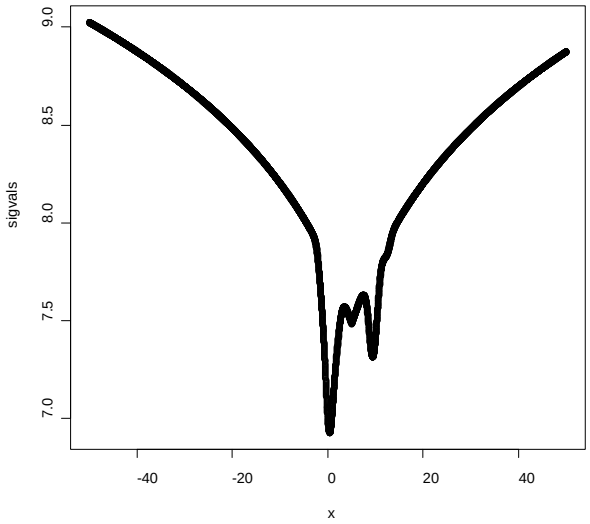
Accepted 23278 out of 100000 (23.278%).
final beta = 1.720646
alpha(5.000000) is 0.932807
alpha(7.000000) is 0.579465
alpha(10.000000) is 0.189430
alpha(20.000000) is 0.478537
alpha(30.000000) is 0.497733
alpha(50.000000) is 0.497044
alpha(70.000000) is 0.499786
alpha(120.000000) is 0.510260
alpha(150.000000) is 0.494803
it takes 219.482000 seconds to run the adaptation algorithm
it takes 145.520000 seconds to compute g and sigma
it takes 22.165000 seconds to generate the first Markov Chain
it takes 21.901000 seconds to generate the second Markov Chain
it takes 21.597000 seconds to generate the third Markov Chain
it takes 21.378000 seconds to generate the fourth Markov Chain
it takes 22.742000 seconds to generate the fifth Markov Chain
it takes 2409.464000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 12.76278
[1] 0.06078702
[1] 7.467558

2.2.1.11 case 77 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth

$b_n = 1$



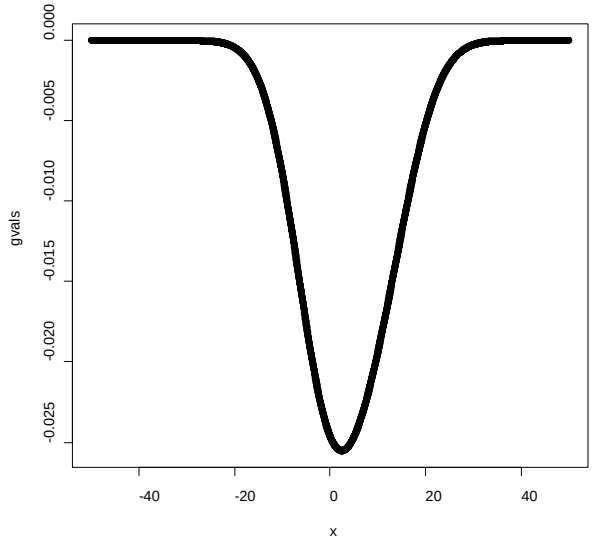


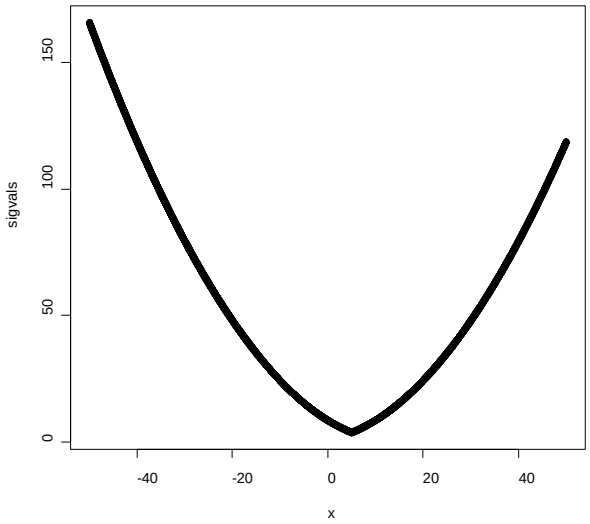
Accepted 23701 out of 100000 (23.701%).
final beta = 2.012694
alpha(5.000000) is 0.817731
alpha(7.000000) is 0.540472
alpha(10.000000) is 0.186645
alpha(20.000000) is 0.505608
alpha(30.000000) is 0.501515
alpha(50.000000) is 0.504638
alpha(70.000000) is 0.499089
alpha(120.000000) is 0.501855
alpha(150.000000) is 0.504430
it takes 487.270000 seconds to run the adaptation algorithm
it takes 277.239000 seconds to compute g and sigma
it takes 51.754000 seconds to generate the first Markov Chain
it takes 53.247000 seconds to generate the second Markov Chain
it takes 54.769000 seconds to generate the third Markov Chain
it takes 53.697000 seconds to generate the fourth Markov Chain
it takes 50.302000 seconds to generate the fifth Markov Chain
it takes 4301.456000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 14.24509
[1] 0.06419357
[1] 6.900112

2.2.1.12 case 78 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 10$



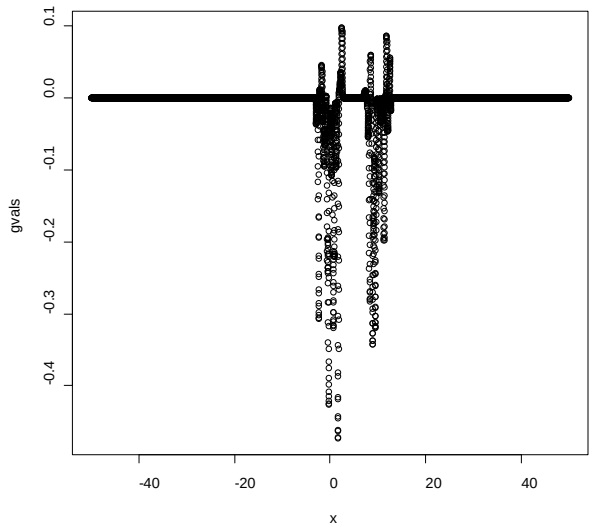


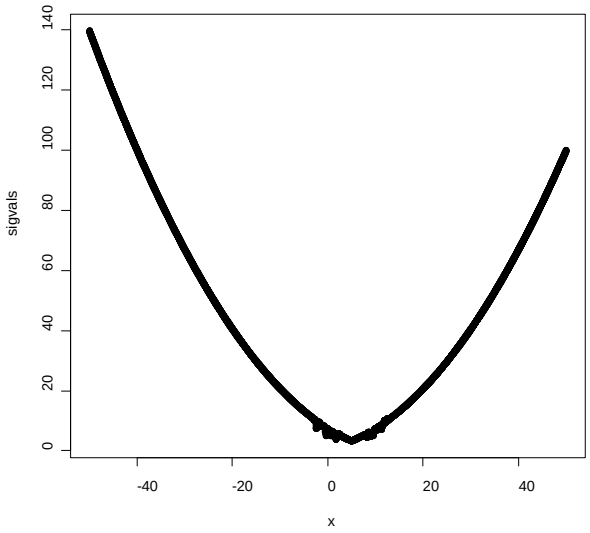
Accepted 22066 out of 100000 (22.066%).
final beta = 1.366644
alpha(5.000000) is 0.993489
alpha(7.000000) is 0.616531
alpha(10.000000) is 0.177149
alpha(20.000000) is 0.392194
alpha(30.000000) is 0.338101
alpha(50.000000) is 0.271826
alpha(70.000000) is 0.228907
alpha(120.000000) is 0.147800
alpha(150.000000) is 0.124100
it takes 225.032000 seconds to run the adaptation algorithm
it takes 100.500000 seconds to compute g and sigma
it takes 21.698000 seconds to generate the first Markov Chain
it takes 21.345000 seconds to generate the second Markov Chain
it takes 21.250000 seconds to generate the third Markov Chain
it takes 21.398000 seconds to generate the fourth Markov Chain
it takes 22.055000 seconds to generate the fifth Markov Chain
it takes 2007.569000 seconds to test the local acceptance

[1] 12.88139
[1] 0.06114331
[1] 7.709572

2.2.1.13 case 79 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 0.1$

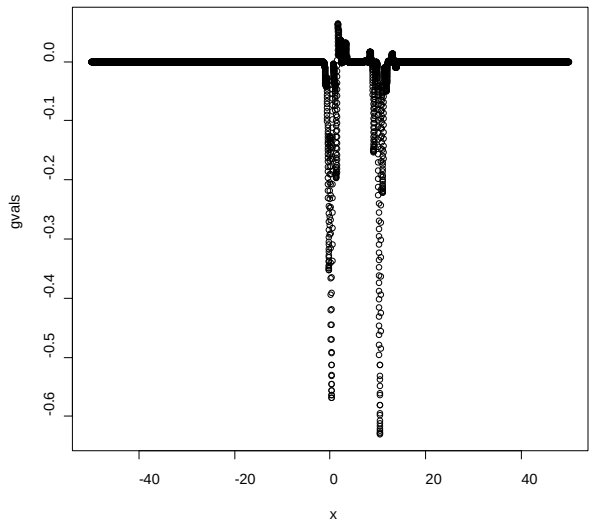


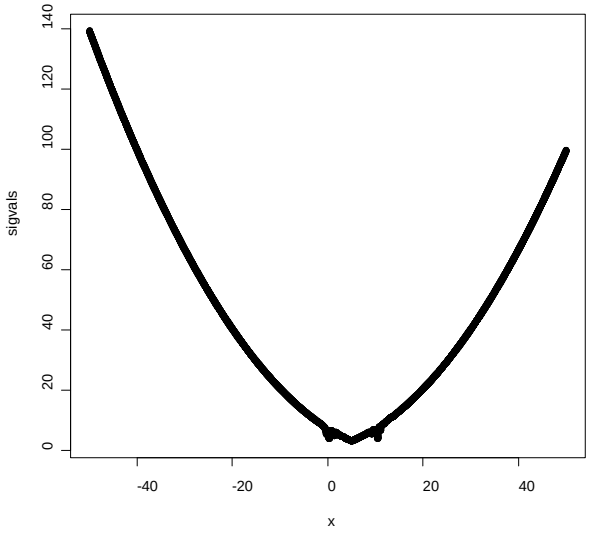


Accepted 24301 out of 100000 (24.301%).
final beta = 1.194554
alpha(5.000000) is 0.998548
alpha(7.000000) is 0.641290
alpha(10.000000) is 0.196131
alpha(20.000000) is 0.437494
alpha(30.000000) is 0.399110
alpha(50.000000) is 0.316721
alpha(70.000000) is 0.254963
alpha(120.000000) is 0.171400
alpha(150.000000) is 0.141227
it takes 351.445000 seconds to run the adaptation algorithm
it takes 153.841000 seconds to compute g and sigma
it takes 21.685000 seconds to generate the first Markov Chain
it takes 21.738000 seconds to generate the second Markov Chain
it takes 21.278000 seconds to generate the third Markov Chain
it takes 21.452000 seconds to generate the fourth Markov Chain
it takes 21.341000 seconds to generate the fifth Markov Chain
it takes 2306.552000 seconds to test the local acceptance

[1] 15.5557
[1] 0.06697404
[1] 6.533062

2.2.1.14 case 80 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$



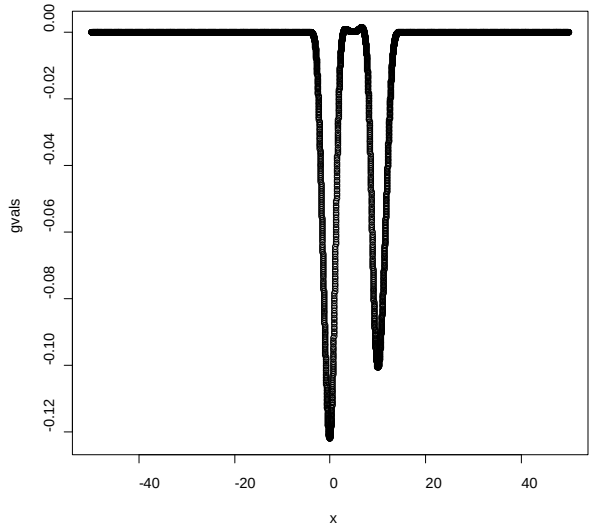


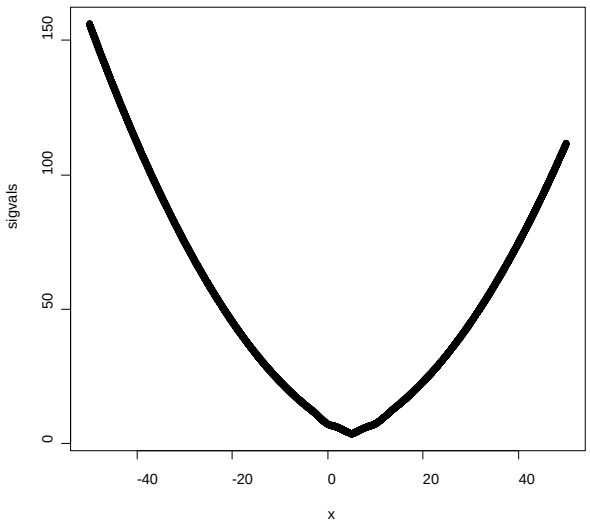
Accepted 24208 out of 100000 (24.208%).
final beta = 1.192546
alpha(5.000000) is 0.998591
alpha(7.000000) is 0.640761
alpha(10.000000) is 0.194188
alpha(20.000000) is 0.425004
alpha(30.000000) is 0.392819
alpha(50.000000) is 0.317502
alpha(70.000000) is 0.260490
alpha(120.000000) is 0.171910
alpha(150.000000) is 0.144109
it takes 324.573000 seconds to run the adaptation algorithm
it takes 153.297000 seconds to compute g and sigma
it takes 20.367000 seconds to generate the first Markov Chain
it takes 20.281000 seconds to generate the second Markov Chain
it takes 20.337000 seconds to generate the third Markov Chain
it takes 20.305000 seconds to generate the fourth Markov Chain
it takes 20.278000 seconds to generate the fifth Markov Chain
it takes 2252.503000 seconds to test the local acceptance

[1] 17.61966
[1] 0.0711586
[1] 6.218763

2.2.1.15 case 81 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 10$



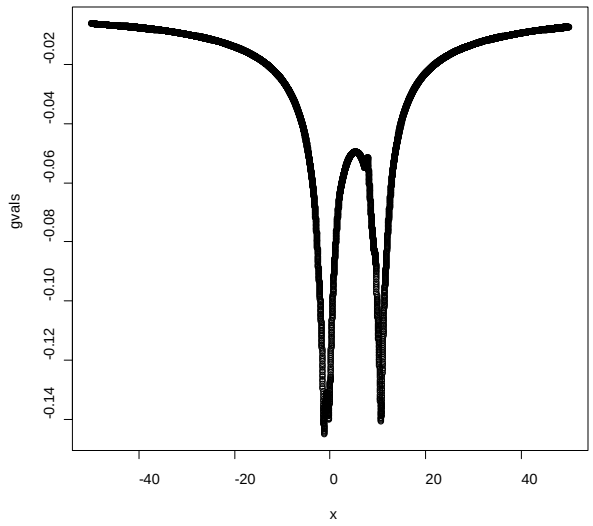


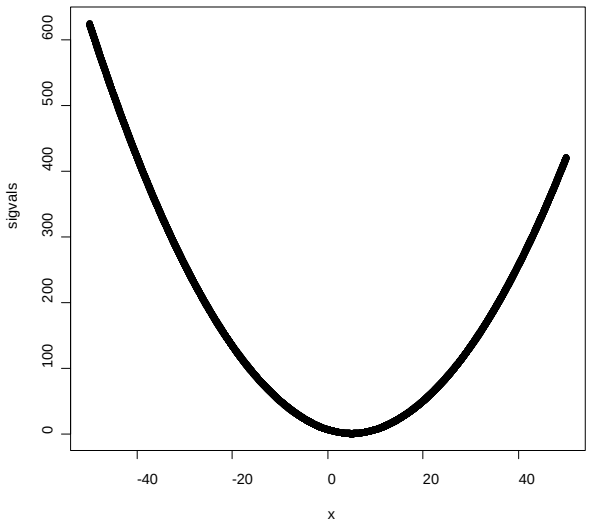
Accepted 23329 out of 100000 (23.329%).
final beta = 1.307548
alpha(5.000000) is 0.994511
alpha(7.000000) is 0.618129
alpha(10.000000) is 0.182081
alpha(20.000000) is 0.409533
alpha(30.000000) is 0.366793
alpha(50.000000) is 0.289425
alpha(70.000000) is 0.239334
alpha(120.000000) is 0.158200
alpha(150.000000) is 0.128500
it takes 214.443000 seconds to run the adaptation algorithm
it takes 139.279000 seconds to compute g and sigma
it takes 20.489000 seconds to generate the first Markov Chain
it takes 20.609000 seconds to generate the second Markov Chain
it takes 20.549000 seconds to generate the third Markov Chain
it takes 20.468000 seconds to generate the fourth Markov Chain
it takes 20.479000 seconds to generate the fifth Markov Chain
it takes 2795.277000 seconds to test the local acceptance

[1] 13.18471
[1] 0.06168104
[1] 7.440597

2.2.2 kernel function $K(x)=\frac{1}{1+\alpha_1 |x|^{\alpha_2}}$

2.2.2.1 case 90 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=1,\alpha_2=1,C=1,\gamma=2$ with fixed bandwidth $b_n=1$

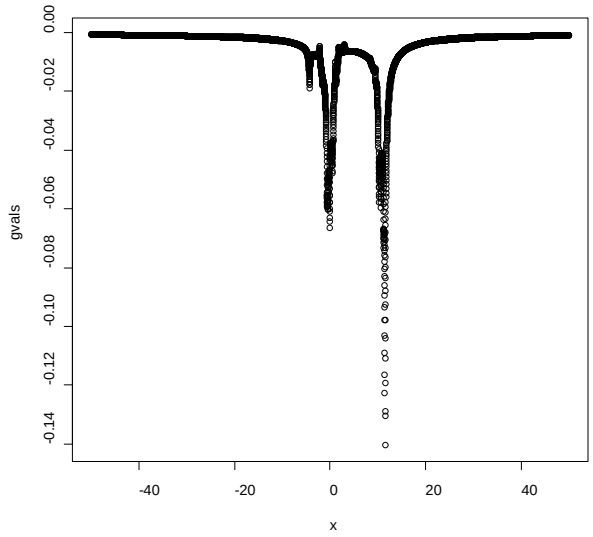


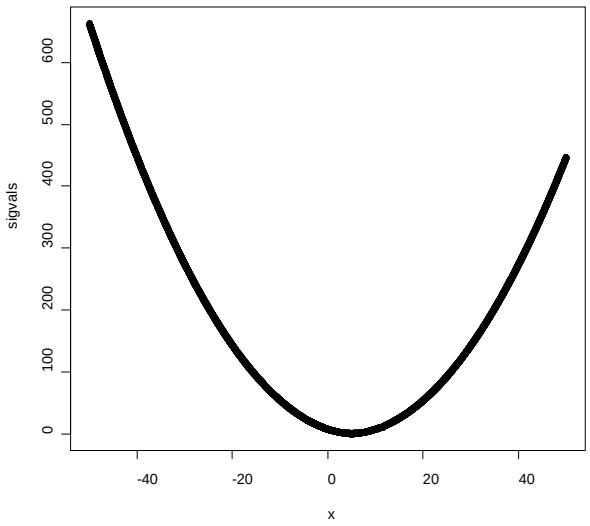


Accepted 24720 out of 100000 (24.720%).
final beta = -1.607328
alpha(5.000000) is 0.988139
alpha(7.000000) is 0.589771
alpha(10.000000) is 0.200291
alpha(20.000000) is 0.196613
alpha(30.000000) is 0.137781
alpha(50.000000) is 0.081525
alpha(70.000000) is 0.057500
alpha(120.000000) is 0.032600
alpha(150.000000) is 0.024600
it takes 184.334000 seconds to run the adaptation algorithm
it takes 81.864000 seconds to compute g and sigma
it takes 21.676000 seconds to generate the first Markov Chain
it takes 20.450000 seconds to generate the second Markov Chain
it takes 20.362000 seconds to generate the third Markov Chain
it takes 20.683000 seconds to generate the fourth Markov Chain
it takes 20.657000 seconds to generate the fifth Markov Chain
it takes 1592.689000 seconds to test the local acceptance

[1] 19.49482
[1] 0.07497625
[1] 5.323529

2.2.2.2 case 91 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$



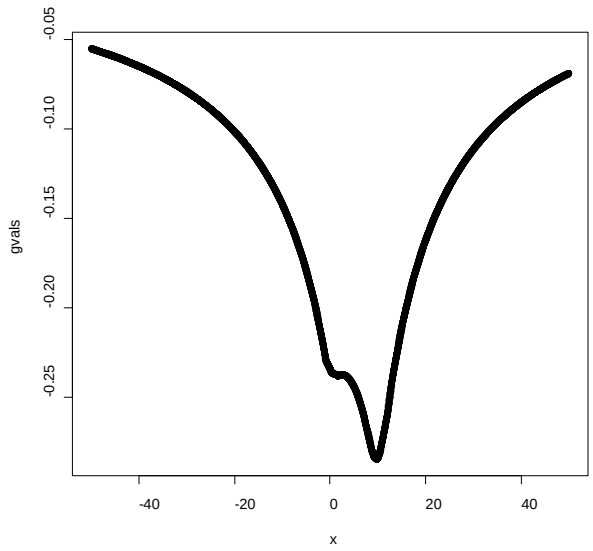


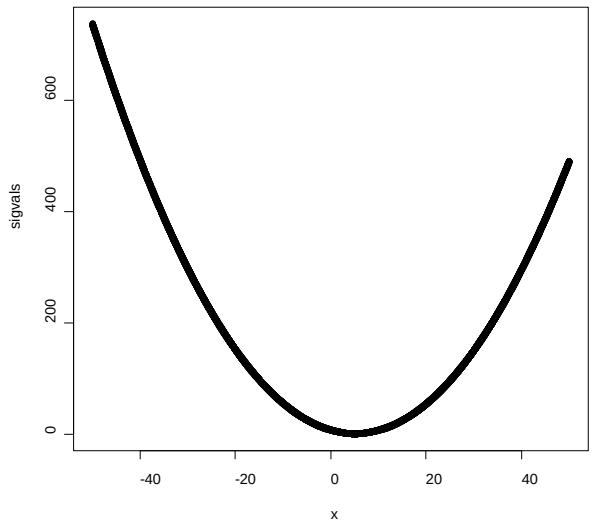
Accepted 23284 out of 100000 (23.284%).
final beta = -1.553623
alpha(5.000000) is 0.988174
alpha(7.000000) is 0.596101
alpha(10.000000) is 0.194054
alpha(20.000000) is 0.188999
alpha(30.000000) is 0.127711
alpha(50.000000) is 0.076924
alpha(70.000000) is 0.056700
alpha(120.000000) is 0.034100
alpha(150.000000) is 0.024600
it takes 185.151000 seconds to run the adaptation algorithm
it takes 82.135000 seconds to compute g and sigma
it takes 21.671000 seconds to generate the first Markov Chain
it takes 21.784000 seconds to generate the second Markov Chain
it takes 21.923000 seconds to generate the third Markov Chain
it takes 22.090000 seconds to generate the fourth Markov Chain
it takes 22.497000 seconds to generate the fifth Markov Chain
it takes 1604.363000 seconds to test the local acceptance

[1] 16.85428
[1] 0.06995134
[1] 6.095268

2.2.2.3 case 92 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

$b_n = 1$

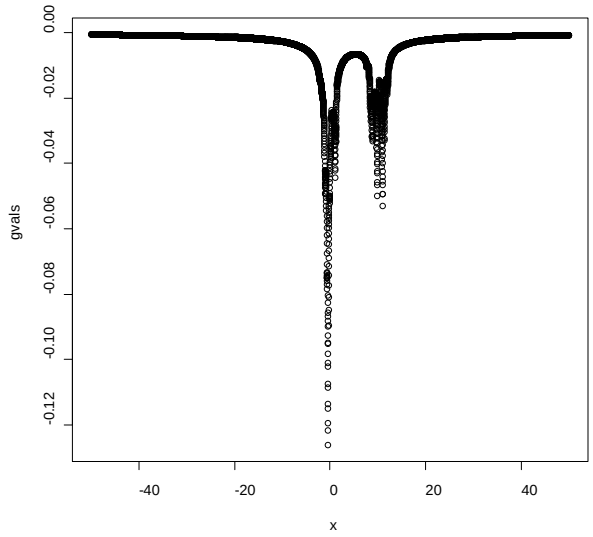


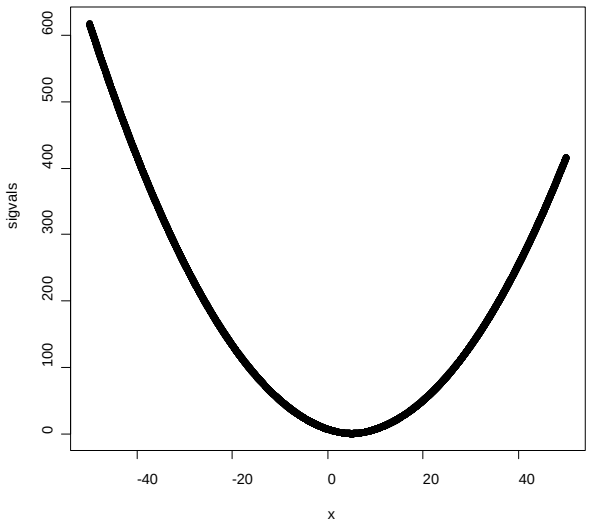


Accepted 23990 out of 100000 (23.990%).
final beta = -1.392714
alpha(5.000000) is 0.988405
alpha(7.000000) is 0.593630
alpha(10.000000) is 0.195274
alpha(20.000000) is 0.185996
alpha(30.000000) is 0.121447
alpha(50.000000) is 0.068600
alpha(70.000000) is 0.049300
alpha(120.000000) is 0.028100
alpha(150.000000) is 0.022400
it takes 179.813000 seconds to run the adaptation algorithm
it takes 80.313000 seconds to compute g and sigma
it takes 21.353000 seconds to generate the first Markov Chain
it takes 21.448000 seconds to generate the second Markov Chain
it takes 21.540000 seconds to generate the third Markov Chain
it takes 21.411000 seconds to generate the fourth Markov Chain
it takes 21.412000 seconds to generate the fifth Markov Chain
it takes 1617.949000 seconds to test the local acceptance

[1] 18.48608
[1] 0.07306009
[1] 5.840868

2.2.2.4 case 93 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 12, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

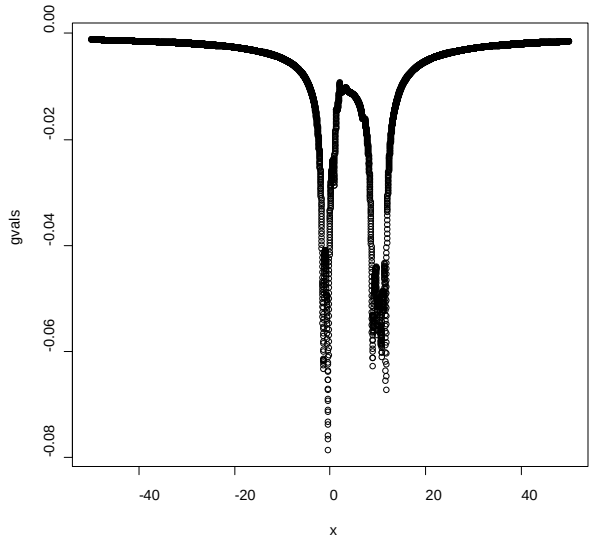


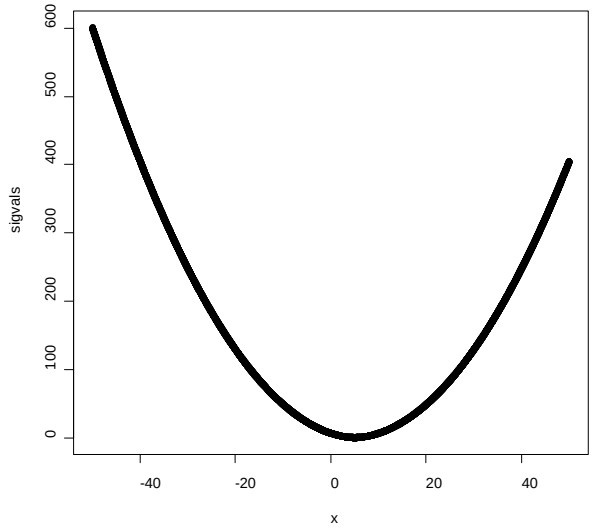


Accepted 24077 out of 100000 (24.077%).
final beta = -1.624033
alpha(5.000000) is 0.988294
alpha(7.000000) is 0.593646
alpha(10.000000) is 0.198721
alpha(20.000000) is 0.199731
alpha(30.000000) is 0.138526
alpha(50.000000) is 0.078896
alpha(70.000000) is 0.059563
alpha(120.000000) is 0.033700
alpha(150.000000) is 0.027705
it takes 184.236000 seconds to run the adaptation algorithm
it takes 80.468000 seconds to compute g and sigma
it takes 21.800000 seconds to generate the first Markov Chain
it takes 21.435000 seconds to generate the second Markov Chain
it takes 22.060000 seconds to generate the third Markov Chain
it takes 22.133000 seconds to generate the fourth Markov Chain
it takes 21.635000 seconds to generate the fifth Markov Chain
it takes 1601.408000 seconds to test the local acceptance

[1] 19.10133
[1] 0.07426885
[1] 5.761007

2.2.2.5 case 94 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 5, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$



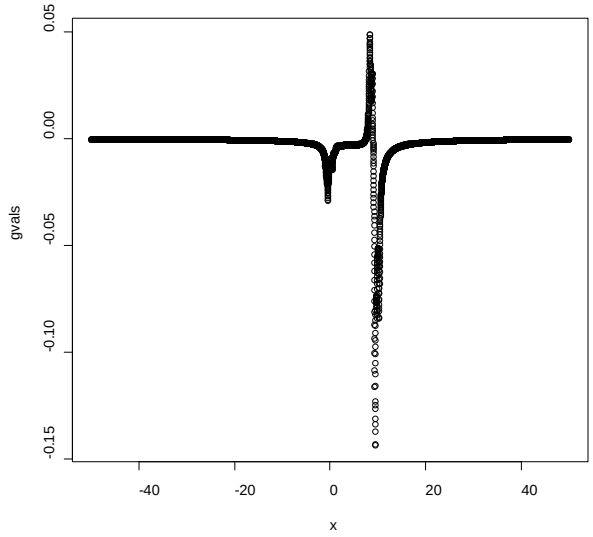


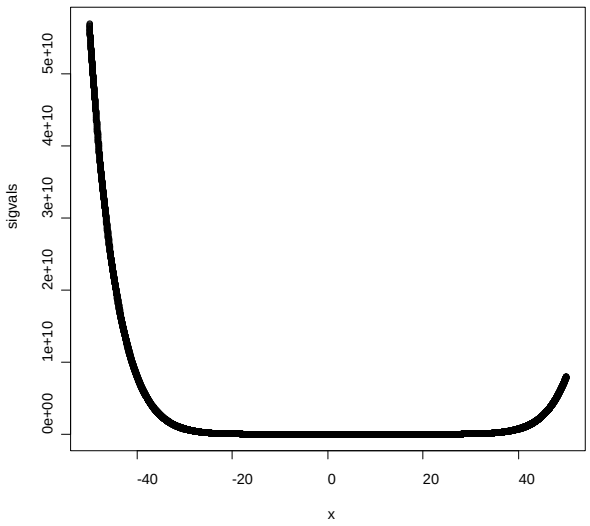
Accepted 24465 out of 100000 (24.465%).
final beta = -1.651342
alpha(5.000000) is 0.988520
alpha(7.000000) is 0.585530
alpha(10.000000) is 0.197157
alpha(20.000000) is 0.200606
alpha(30.000000) is 0.136503
alpha(50.000000) is 0.085026
alpha(70.000000) is 0.054800
alpha(120.000000) is 0.034400
alpha(150.000000) is 0.026400
it takes 182.533000 seconds to run the adaptation algorithm
it takes 79.414000 seconds to compute g and sigma
it takes 20.642000 seconds to generate the first Markov Chain
it takes 20.342000 seconds to generate the second Markov Chain
it takes 20.352000 seconds to generate the third Markov Chain
it takes 20.449000 seconds to generate the fourth Markov Chain
it takes 20.339000 seconds to generate the fifth Markov Chain
it takes 1575.653000 seconds to test the local acceptance

[1] 19.09234
[1] 0.07416466
[1] 5.474942

2.2.2.6 case 95 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 10$ with fixed bandwidth

$b_n = 1$



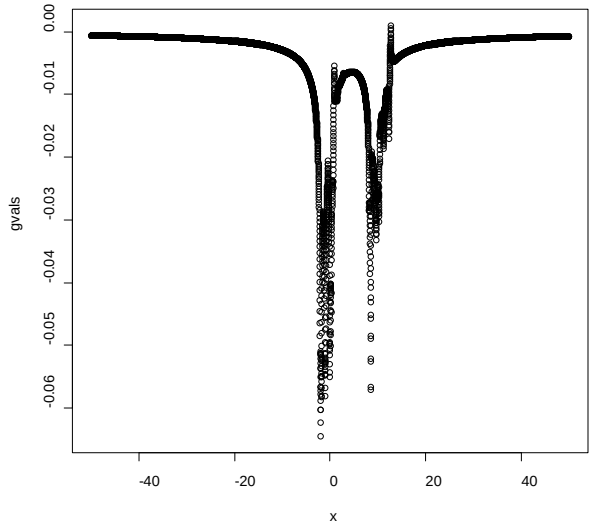


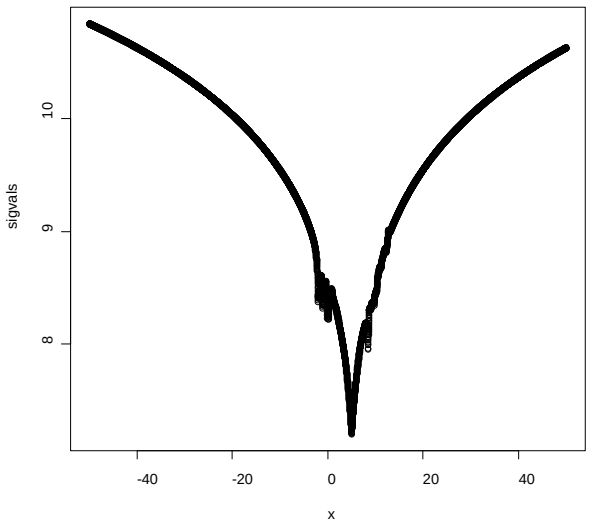
Accepted 18062 out of 100000 (18.062%).
final beta = -15.487329
alpha(5.000000) is 0.975604
alpha(7.000000) is 0.970986
alpha(10.000000) is 0.099708
alpha(20.000000) is 0.000000
alpha(30.000000) is 0.000000
alpha(50.000000) is 0.000000
alpha(70.000000) is 0.000000
alpha(120.000000) is 0.000000
alpha(150.000000) is 0.000000
it takes 184.158000 seconds to run the adaptation algorithm
it takes 80.077000 seconds to compute g and sigma
it takes 129.847000 seconds to generate the first Markov Chain
it takes 106.905000 seconds to generate the second Markov Chain
it takes 114.789000 seconds to generate the third Markov Chain
it takes 111.836000 seconds to generate the fourth Markov Chain
it takes 122.212000 seconds to generate the fifth Markov Chain
it takes 1797.306000 seconds to test the local acceptance

[1] 53.60444
[1] 0.1244969
[1] 1.709541

2.2.2.7 case 96 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 0.1$ with fixed bandwidth

$b_n = 1$



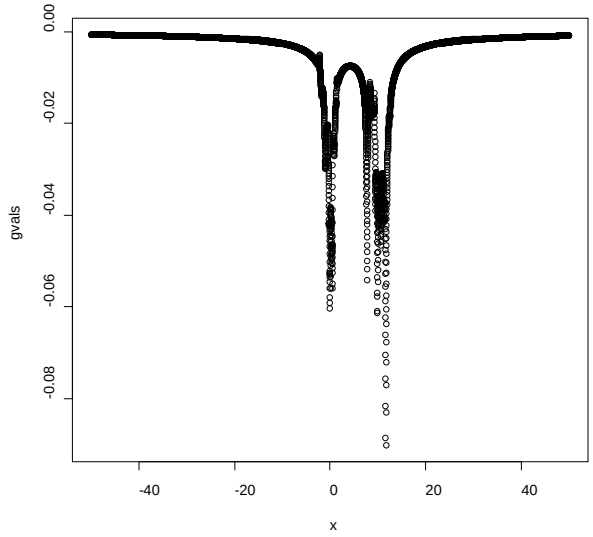


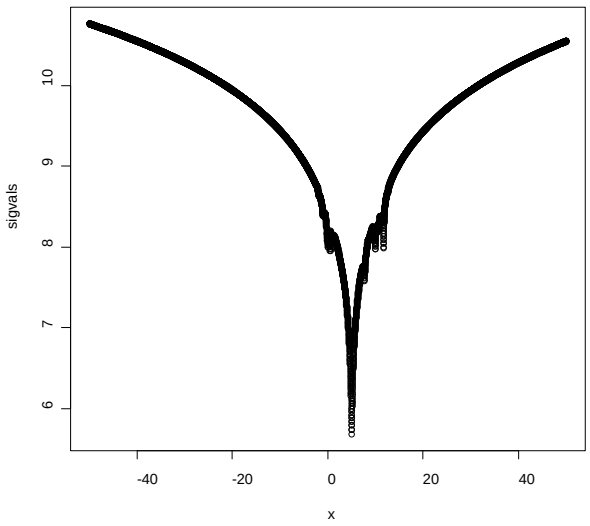
Accepted 22236 out of 100000 (22.236%).
final beta = 1.981254
alpha(5.000000) is 0.837339
alpha(7.000000) is 0.525632
alpha(10.000000) is 0.178894
alpha(20.000000) is 0.504550
alpha(30.000000) is 0.500767
alpha(50.000000) is 0.498991
alpha(70.000000) is 0.497244
alpha(120.000000) is 0.500871
alpha(150.000000) is 0.500862
it takes 180.159000 seconds to run the adaptation algorithm
it takes 80.338000 seconds to compute g and sigma
it takes 20.996000 seconds to generate the first Markov Chain
it takes 20.246000 seconds to generate the second Markov Chain
it takes 20.903000 seconds to generate the third Markov Chain
it takes 20.337000 seconds to generate the fourth Markov Chain
it takes 20.582000 seconds to generate the fifth Markov Chain
it takes 1096.667000 seconds to test the local acceptance

[1] 12.47803
[1] 0.05992151
[1] 7.639173

2.2.2.8 case 97 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 10$ with fixed bandwidth

$b_n = 1$



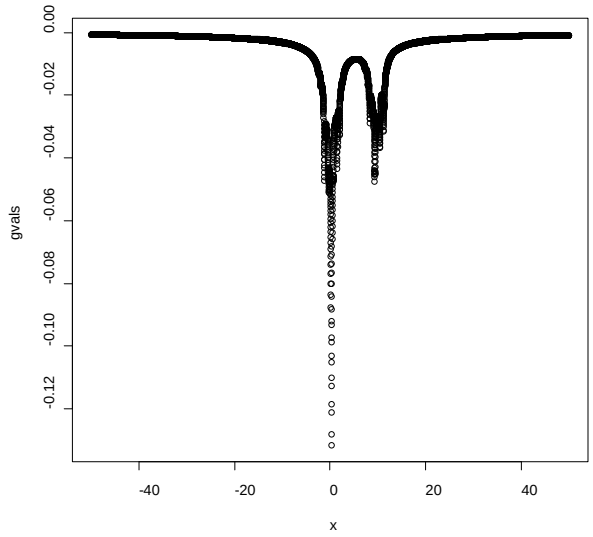


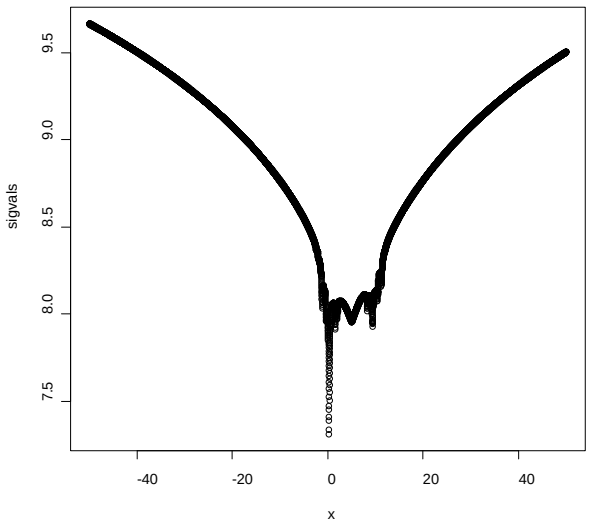
Accepted 22723 out of 100000 (22.723%).
final beta = 2.082104
alpha(5.000000) is 0.000000
alpha(7.000000) is 0.000000
alpha(10.000000) is 0.000000
alpha(20.000000) is 0.000000
alpha(30.000000) is 0.000000
alpha(50.000000) is 0.018600
alpha(70.000000) is 0.129300
alpha(120.000000) is 0.286300
alpha(150.000000) is 0.341000
it takes 193.949000 seconds to run the adaptation algorithm
it takes 85.107000 seconds to compute g and sigma
it takes 22.242000 seconds to generate the first Markov Chain
it takes 22.288000 seconds to generate the second Markov Chain
it takes 22.125000 seconds to generate the third Markov Chain
it takes 22.425000 seconds to generate the fourth Markov Chain
it takes 22.492000 seconds to generate the fifth Markov Chain
it takes 814.063000 seconds to test the local acceptance

[1] 12.94998
[1] 0.06111524
[1] 7.64341

2.2.2.9 case 98 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 10$



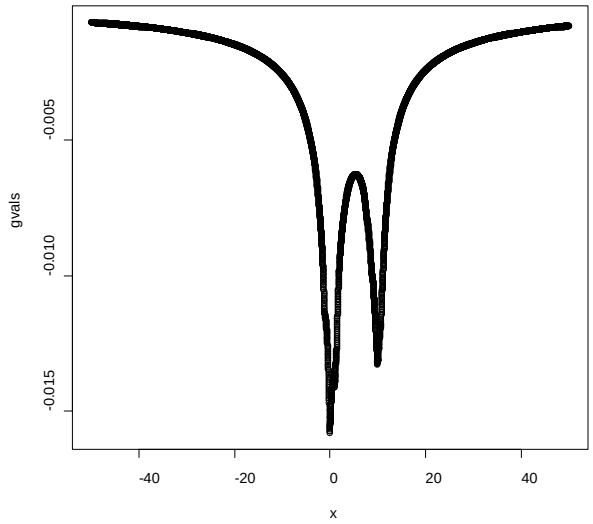


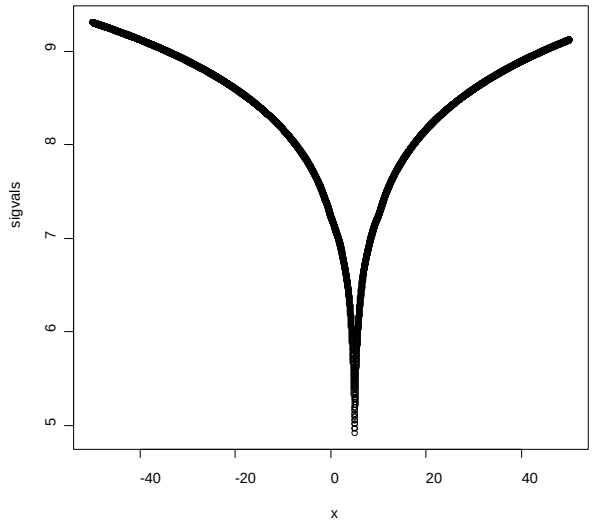
Accepted 24628 out of 100000 (24.628%).
final beta = 1.935983
alpha(5.000000) is 0.856700
alpha(7.000000) is 0.567400
alpha(10.000000) is 0.195400
alpha(20.000000) is 0.497600
alpha(30.000000) is 0.502600
alpha(50.000000) is 0.495200
alpha(70.000000) is 0.510000
alpha(120.000000) is 0.498000
alpha(150.000000) is 0.506600
it takes 180.883000 seconds to run the adaptation algorithm
it takes 79.484000 seconds to compute g and sigma
it takes 20.451000 seconds to generate the first Markov Chain
it takes 20.749000 seconds to generate the second Markov Chain
it takes 20.666000 seconds to generate the third Markov Chain
it takes 20.354000 seconds to generate the fourth Markov Chain
it takes 20.373000 seconds to generate the fifth Markov Chain
it takes 771.898000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 13.93196
[1] 0.06357011
[1] 7.406249

2.2.2.10 case 99 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 0.1$



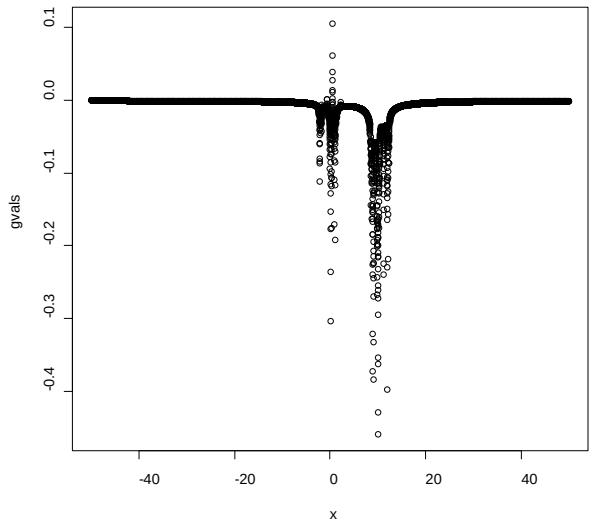


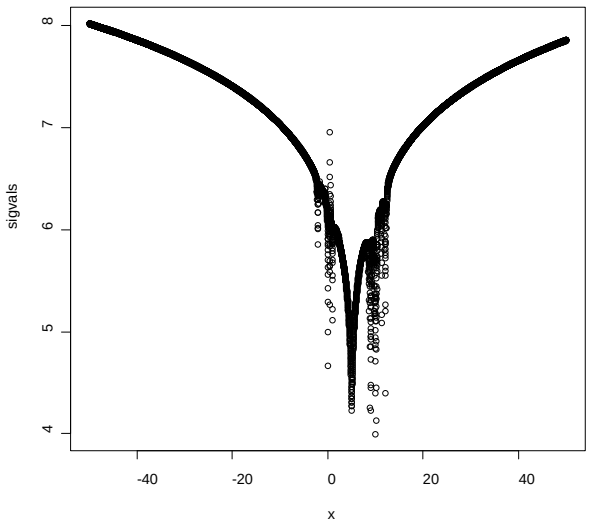
Accepted 24127 out of 100000 (24.127%).
final beta = 1.600481
alpha(5.000000) is 0.959075
alpha(7.000000) is 0.577982
alpha(10.000000) is 0.189727
alpha(20.000000) is 0.504383
alpha(30.000000) is 0.501884
alpha(50.000000) is 0.493078
alpha(70.000000) is 0.497578
alpha(120.000000) is 0.491683
alpha(150.000000) is 0.498875
it takes 189.887000 seconds to run the adaptation algorithm
it takes 82.323000 seconds to compute g and sigma
it takes 20.289000 seconds to generate the first Markov Chain
it takes 21.285000 seconds to generate the second Markov Chain
it takes 20.836000 seconds to generate the third Markov Chain
it takes 21.327000 seconds to generate the fourth Markov Chain
it takes 20.622000 seconds to generate the fifth Markov Chain
it takes 1120.381000 seconds to test the local acceptance

[1] 14.33295
[1] 0.06422718
[1] 7.005803

2.2.2.11 case 100 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 0.1$ with fixed bandwidth

$b_n = 10$



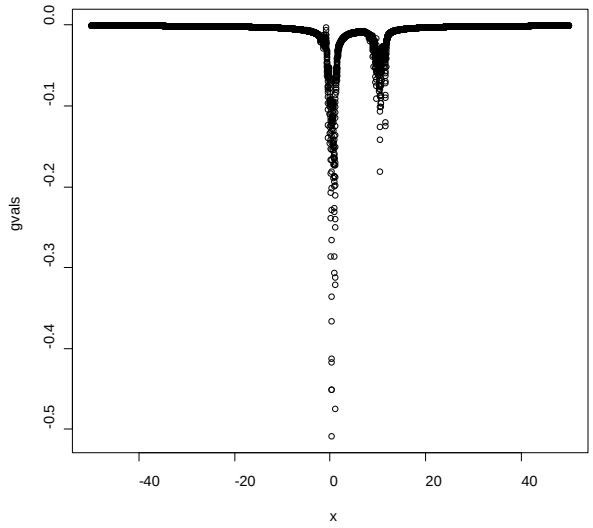


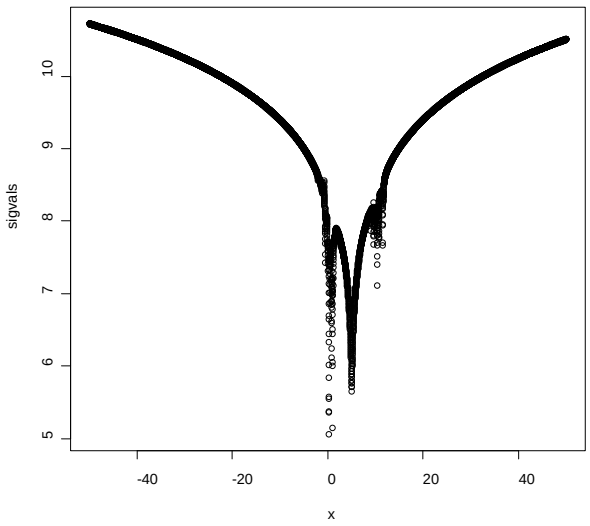
Accepted 25905 out of 100000 (25.905%).
final beta = 1.450997
alpha(5.000000) is 0.982768
alpha(7.000000) is 0.614188
alpha(10.000000) is 0.204066
alpha(20.000000) is 0.511236
alpha(30.000000) is 0.501774
alpha(50.000000) is 0.494540
alpha(70.000000) is 0.499826
alpha(120.000000) is 0.498532
alpha(150.000000) is 0.499842
it takes 185.462000 seconds to run the adaptation algorithm
it takes 81.620000 seconds to compute g and sigma
it takes 20.542000 seconds to generate the first Markov Chain
it takes 21.623000 seconds to generate the second Markov Chain
it takes 20.950000 seconds to generate the third Markov Chain
it takes 21.644000 seconds to generate the fourth Markov Chain
it takes 21.796000 seconds to generate the fifth Markov Chain
it takes 1133.939000 seconds to test the local acceptance

[1] 18.55894
[1] 0.07316876
[1] 5.515589

2.2.2.12 case 101 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with decreasing bandwidth

$b_n = \frac{1}{n^{0.2}}$



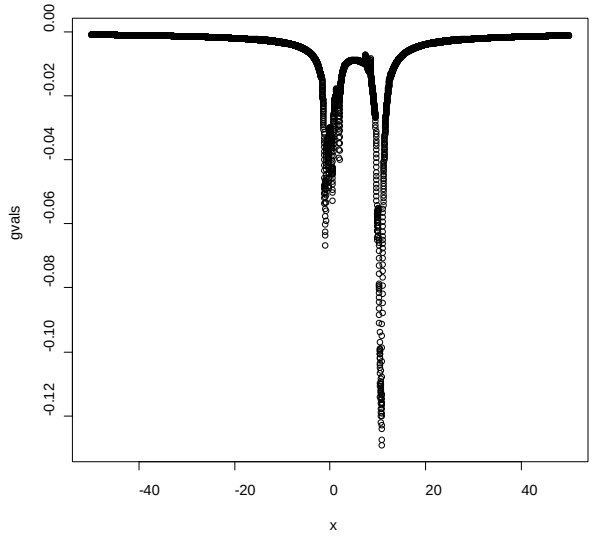


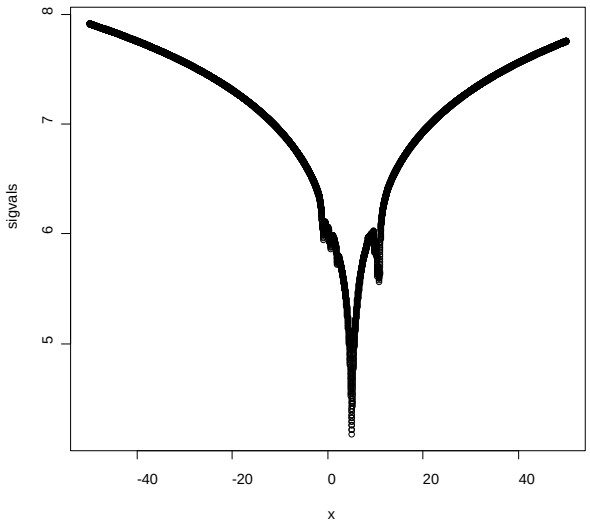
Accepted 23002 out of 100000 (23.002%).
final beta = 1.742155
alpha(5.000000) is 0.925394
alpha(7.000000) is 0.534543
alpha(10.000000) is 0.179065
alpha(20.000000) is 0.500864
alpha(30.000000) is 0.495748
alpha(50.000000) is 0.495755
alpha(70.000000) is 0.497514
alpha(120.000000) is 0.498657
alpha(150.000000) is 0.500898
it takes 186.083000 seconds to run the adaptation algorithm
it takes 82.053000 seconds to compute g and sigma
it takes 22.213000 seconds to generate the first Markov Chain
it takes 21.654000 seconds to generate the second Markov Chain
it takes 22.906000 seconds to generate the third Markov Chain
it takes 23.243000 seconds to generate the fourth Markov Chain
it takes 23.168000 seconds to generate the fifth Markov Chain
it takes 1133.776000 seconds to test the local acceptance

[1] 14.00296
[1] 0.06365483
[1] 7.400967

2.2.2.13 case 102 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

$b_n = 10$



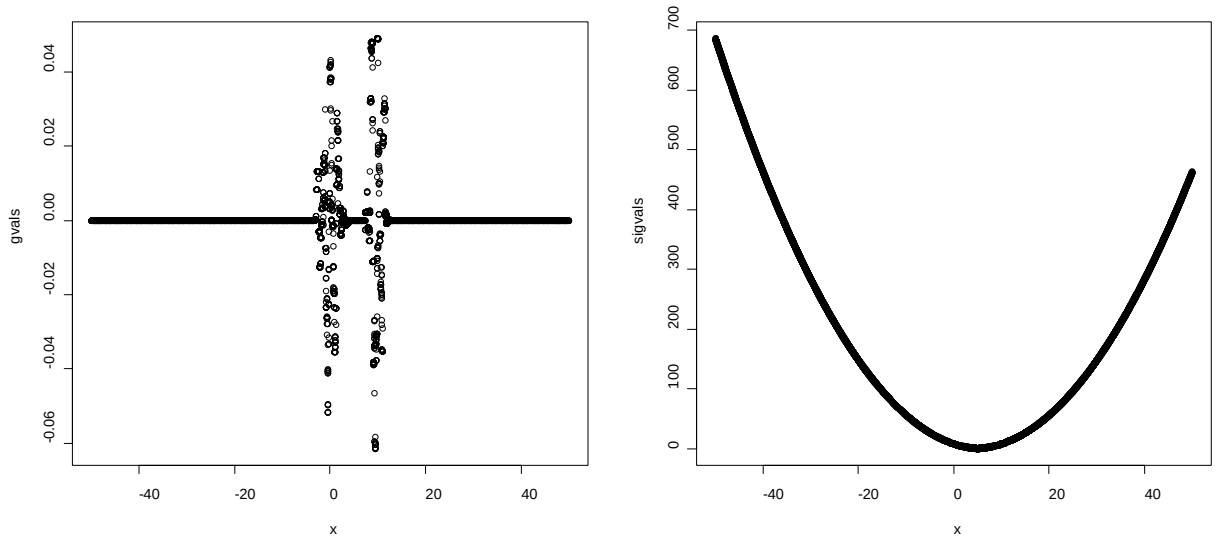


Accepted 23672 out of 100000 (23.672%).
final beta = 1.658811
alpha(5.000000) is 0.945195
alpha(7.000000) is 0.568038
alpha(10.000000) is 0.189494
alpha(20.000000) is 0.501942
alpha(30.000000) is 0.496667
alpha(50.000000) is 0.496623
alpha(70.000000) is 0.501563
alpha(120.000000) is 0.490541
alpha(150.000000) is 0.492890
it takes 181.352000 seconds to run the adaptation algorithm
it takes 78.941000 seconds to compute g and sigma
it takes 20.257000 seconds to generate the first Markov Chain
it takes 20.210000 seconds to generate the second Markov Chain
it takes 20.094000 seconds to generate the third Markov Chain
it takes 20.147000 seconds to generate the fourth Markov Chain
it takes 20.194000 seconds to generate the fifth Markov Chain
it takes 1087.941000 seconds to test the local acceptance

[1] 18.03951
[1] 0.07210694
[1] 5.518752

2.2.3 kernel function $K(x)=\begin{cases}0, \mid x \mid \geq 2*width \\ -1*hegith, width < \mid x \mid < 2*width \\ 1*height, \mid x \mid \leq width\end{cases}$

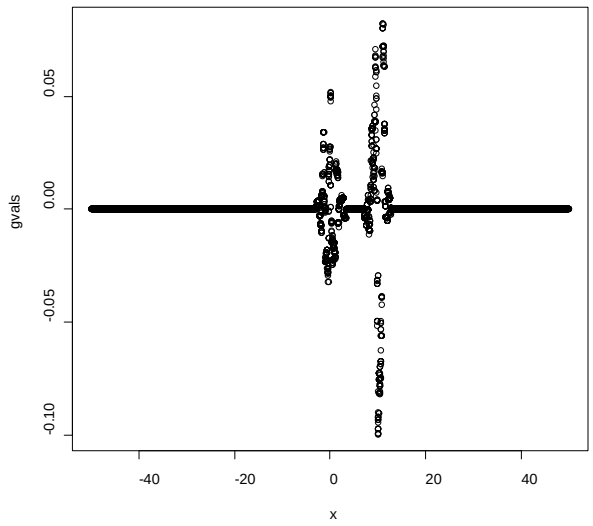
2.2.3.1 case 103 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C=1, \gamma=2, \text{height}=0.5, \text{width}=0.5,$ with fixed bandwidth $b_n = 1$

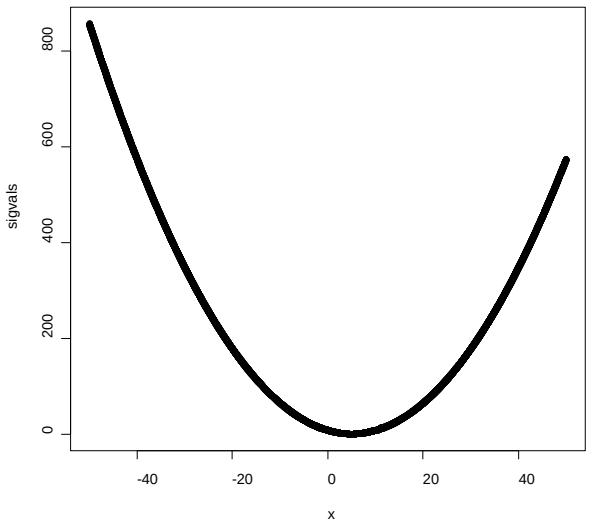


Accepted 23171 out of 100000 (23.171%).
final beta = -1.519856
alpha(5.000000) is 0.988000
alpha(7.000000) is 0.603800
alpha(10.000000) is 0.185000
alpha(20.000000) is 0.182000
alpha(30.000000) is 0.125600
alpha(50.000000) is 0.072000
alpha(70.000000) is 0.051400
alpha(120.000000) is 0.029100
alpha(150.000000) is 0.023900
it takes 125.983000 seconds to run the adaptation algorithm
it takes 33.358000 seconds to compute g and sigma
it takes 18.496000 seconds to generate the first Markov Chain
it takes 18.395000 seconds to generate the second Markov Chain
it takes 19.412000 seconds to generate the third Markov Chain
it takes 18.896000 seconds to generate the fourth Markov Chain
it takes 19.008000 seconds to generate the fifth Markov Chain
it takes 721.432000 seconds to test the local acceptance

[1] 15.87749
[1] 0.06777017
[1] 6.794295

2.2.3.2 case 104 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 10, \gamma = 2, \text{height} = 0.5, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

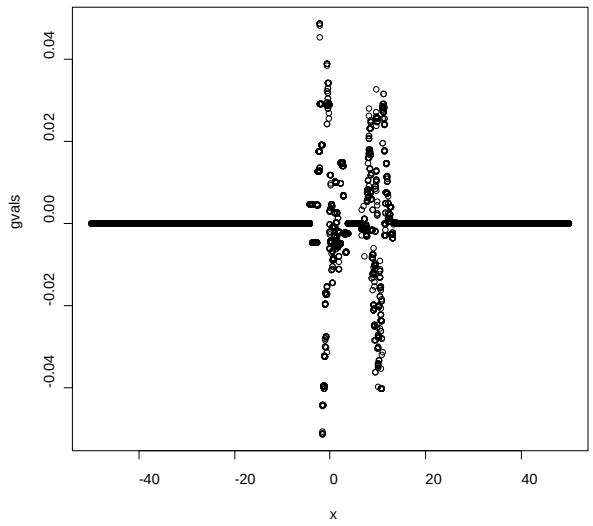


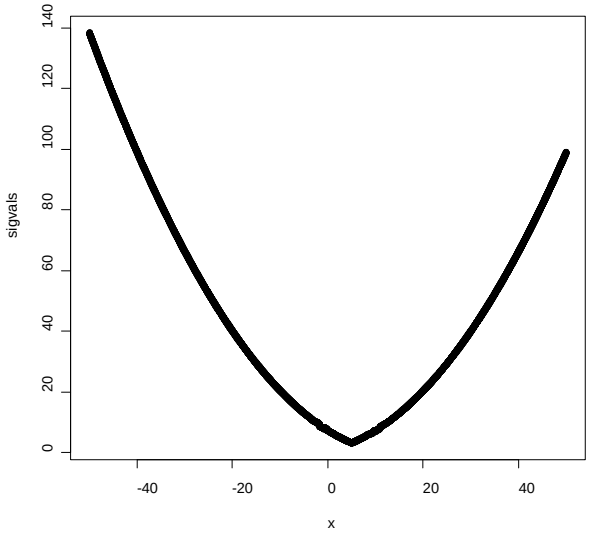


Accepted 23433 out of 100000 (23.433%).
final beta = -5.869167
alpha(5.000000) is 0.956600
alpha(7.000000) is 0.610600
alpha(10.000000) is 0.193700
alpha(20.000000) is 0.154200
alpha(30.000000) is 0.101700
alpha(50.000000) is 0.061300
alpha(70.000000) is 0.043000
alpha(120.000000) is 0.023500
alpha(150.000000) is 0.018200
it takes 126.170000 seconds to run the adaptation algorithm
it takes 33.355000 seconds to compute g and sigma
it takes 18.941000 seconds to generate the first Markov Chain
it takes 19.131000 seconds to generate the second Markov Chain
it takes 18.991000 seconds to generate the third Markov Chain
it takes 18.368000 seconds to generate the fourth Markov Chain
it takes 18.620000 seconds to generate the fifth Markov Chain
it takes 728.738000 seconds to test the local acceptance

[1] 18.30191
[1] 0.07264643
[1] 5.623555

2.2.3.3 case 105 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 2, \text{height} = 0.5, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

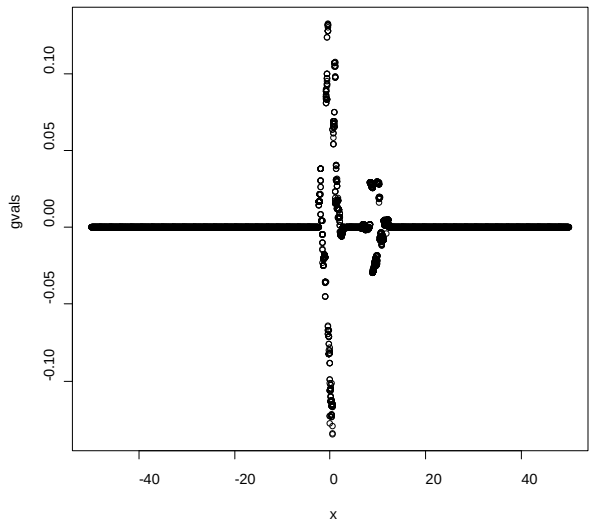


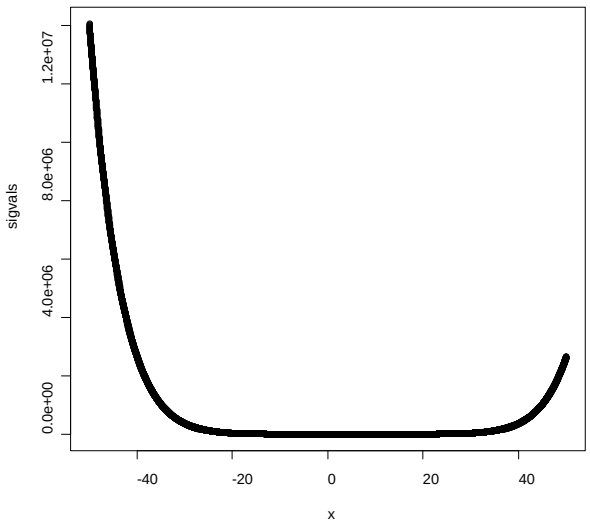


Accepted 24212 out of 100000 (24.212%).
final beta = 1.185992
alpha(5.000000) is 0.998500
alpha(7.000000) is 0.641800
alpha(10.000000) is 0.193100
alpha(20.000000) is 0.437000
alpha(30.000000) is 0.398400
alpha(50.000000) is 0.316700
alpha(70.000000) is 0.265200
alpha(120.000000) is 0.171600
alpha(150.000000) is 0.146700
it takes 124.856000 seconds to run the adaptation algorithm
it takes 33.247000 seconds to compute g and sigma
it takes 14.799000 seconds to generate the first Markov Chain
it takes 15.013000 seconds to generate the second Markov Chain
it takes 14.871000 seconds to generate the third Markov Chain
it takes 15.018000 seconds to generate the fourth Markov Chain
it takes 14.761000 seconds to generate the fifth Markov Chain
it takes 681.928000 seconds to test the local acceptance

[1] 13.42696
[1] 0.06233687
[1] 7.217294

2.2.3.4 case 106 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 10, \text{height} = 0.5, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

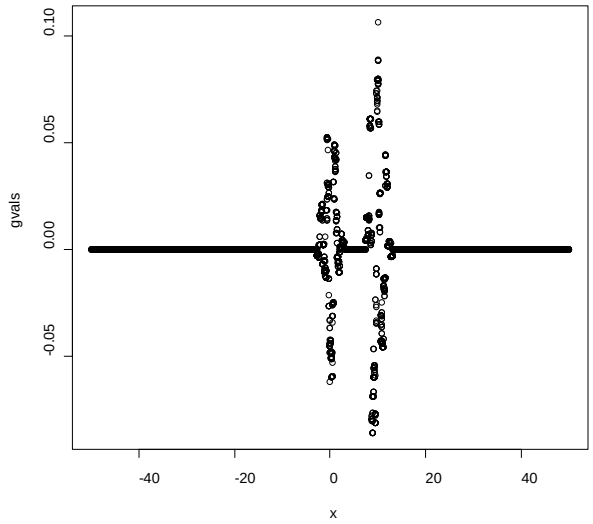


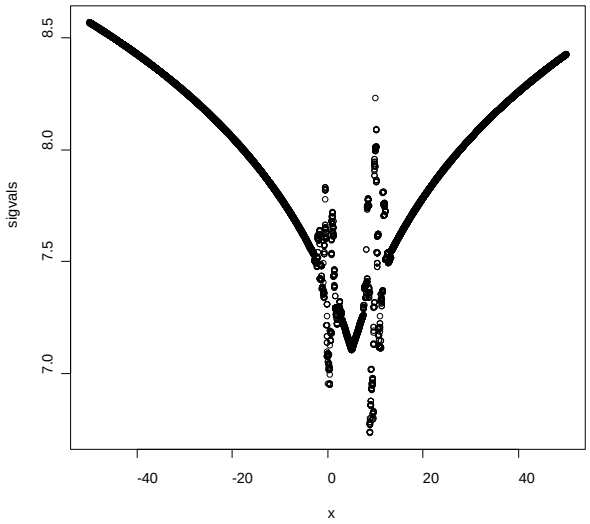


Accepted 23920 out of 100000 (23.920%).
final beta = -2.260178
alpha(5.000000) is 0.998600
alpha(7.000000) is 0.675300
alpha(10.000000) is 0.198500
alpha(20.000000) is 0.011200
alpha(30.000000) is 0.000700
alpha(50.000000) is 0.000000
alpha(70.000000) is 0.000000
alpha(120.000000) is 0.000000
alpha(150.000000) is 0.000000
it takes 126.420000 seconds to run the adaptation algorithm
it takes 33.691000 seconds to compute g and sigma
it takes 26.633000 seconds to generate the first Markov Chain
it takes 26.667000 seconds to generate the second Markov Chain
it takes 25.153000 seconds to generate the third Markov Chain
it takes 24.161000 seconds to generate the fourth Markov Chain
it takes 23.908000 seconds to generate the fifth Markov Chain
it takes 763.257000 seconds to test the local acceptance

[1] 36.67233
[1] 0.1030286
[1] 3.271804

2.2.3.5 case 107 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 0.5, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

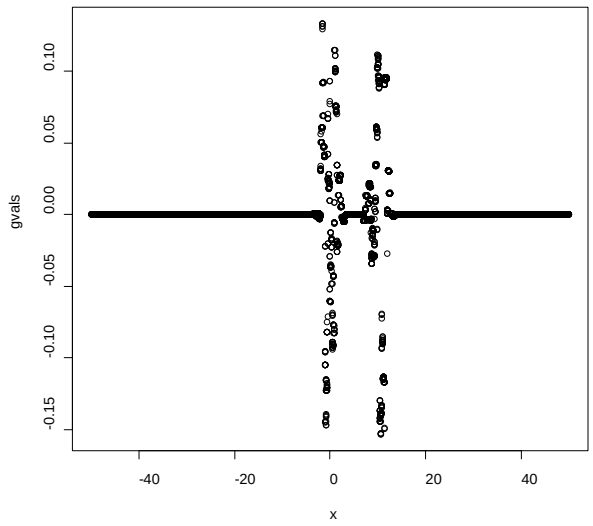


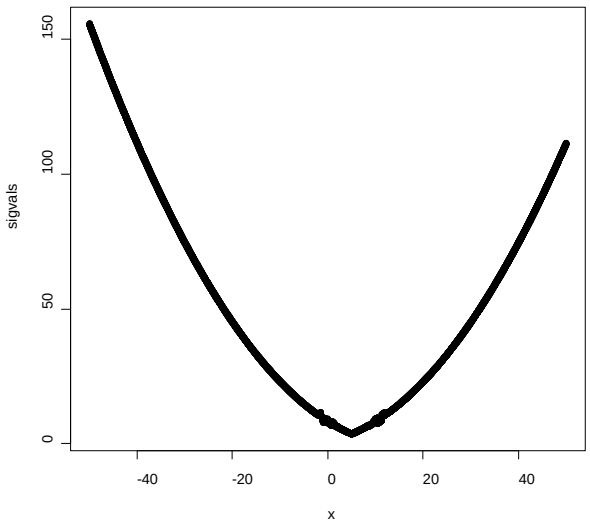


Accepted 24069 out of 100000 (24.069%).
final beta = 1.960744
alpha(5.000000) is 0.845900
alpha(7.000000) is 0.563800
alpha(10.000000) is 0.187600
alpha(20.000000) is 0.502400
alpha(30.000000) is 0.498400
alpha(50.000000) is 0.490100
alpha(70.000000) is 0.501200
alpha(120.000000) is 0.499200
alpha(150.000000) is 0.495100
it takes 126.576000 seconds to run the adaptation algorithm
it takes 33.654000 seconds to compute g and sigma
it takes 14.539000 seconds to generate the first Markov Chain
it takes 14.438000 seconds to generate the second Markov Chain
it takes 14.539000 seconds to generate the third Markov Chain
it takes 14.475000 seconds to generate the fourth Markov Chain
it takes 14.386000 seconds to generate the fifth Markov Chain
it takes 688.012000 seconds to test the local acceptance

[1] 13.54641
[1] 0.06246479
[1] 7.10335

2.2.3.6 case 108 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 2, \text{height} = 1, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

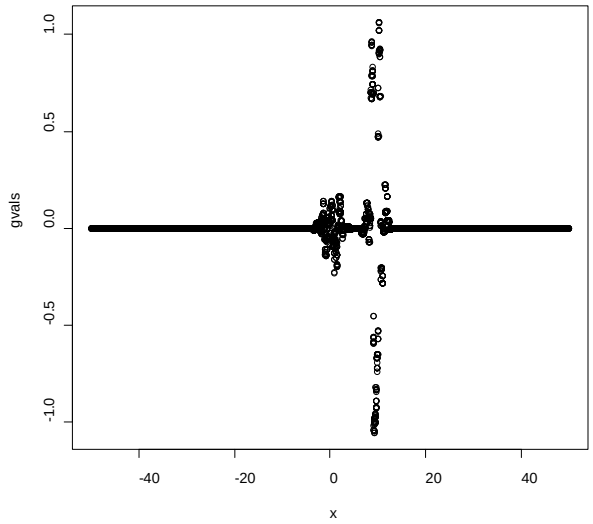


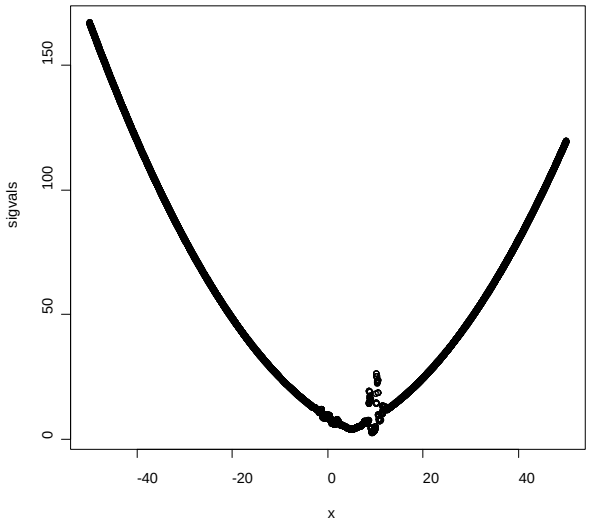


Accepted 22858 out of 100000 (22.858%).
final beta = 1.305092
alpha(5.000000) is 0.994700
alpha(7.000000) is 0.623500
alpha(10.000000) is 0.175000
alpha(20.000000) is 0.406200
alpha(30.000000) is 0.354900
alpha(50.000000) is 0.291400
alpha(70.000000) is 0.231900
alpha(120.000000) is 0.156500
alpha(150.000000) is 0.128000
it takes 128.046000 seconds to run the adaptation algorithm
it takes 34.032000 seconds to compute g and sigma
it takes 17.171000 seconds to generate the first Markov Chain
it takes 16.964000 seconds to generate the second Markov Chain
it takes 16.993000 seconds to generate the third Markov Chain
it takes 17.088000 seconds to generate the fourth Markov Chain
it takes 17.015000 seconds to generate the fifth Markov Chain
it takes 696.429000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 13.20776
[1] 0.06187482
[1] 7.630405

2.2.3.7 case 109 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 2, \text{height} = 5, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

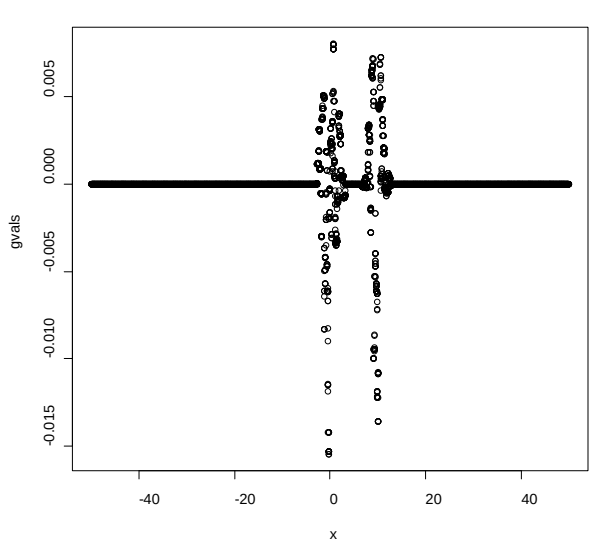


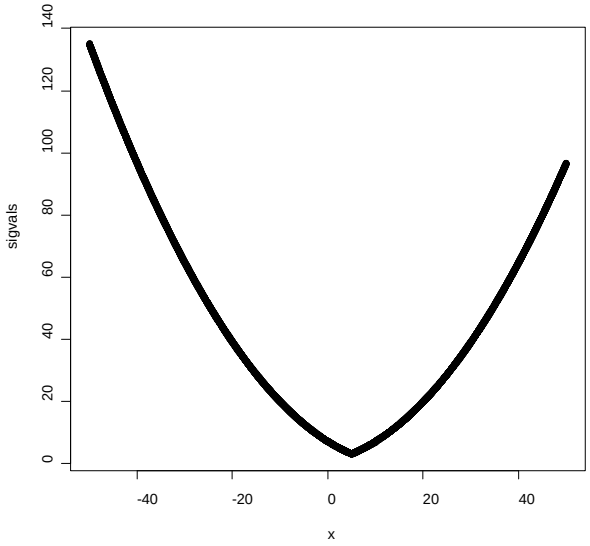


Accepted 19990 out of 100000 (19.990%).
final beta = 1.374566
alpha(5.000000) is 0.990800
alpha(7.000000) is 0.611700
alpha(10.000000) is 0.175300
alpha(20.000000) is 0.396700
alpha(30.000000) is 0.356900
alpha(50.000000) is 0.280000
alpha(70.000000) is 0.224500
alpha(120.000000) is 0.146700
alpha(150.000000) is 0.114400
it takes 126.505000 seconds to run the adaptation algorithm
it takes 33.891000 seconds to compute g and sigma
it takes 28.960000 seconds to generate the first Markov Chain
it takes 30.148000 seconds to generate the second Markov Chain
it takes 29.466000 seconds to generate the third Markov Chain
it takes 30.076000 seconds to generate the fourth Markov Chain
it takes 29.000000 seconds to generate the fifth Markov Chain
it takes 698.858000 seconds to test the local acceptance

[1] 19.00901
[1] 0.07394524
[1] 6.026763

2.2.3.8 case 110 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 2, \text{height} = 0.1, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

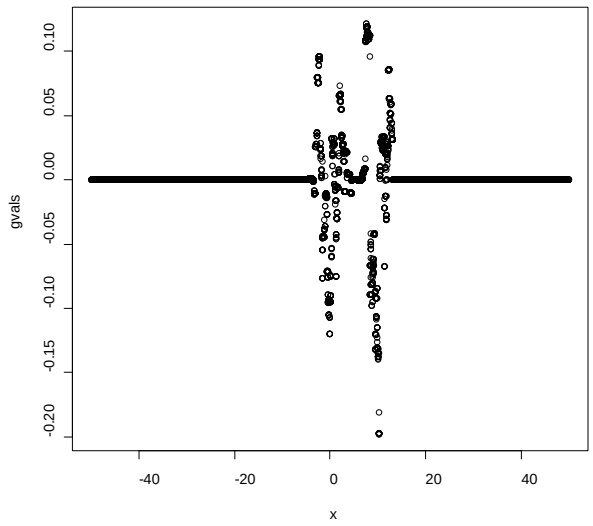


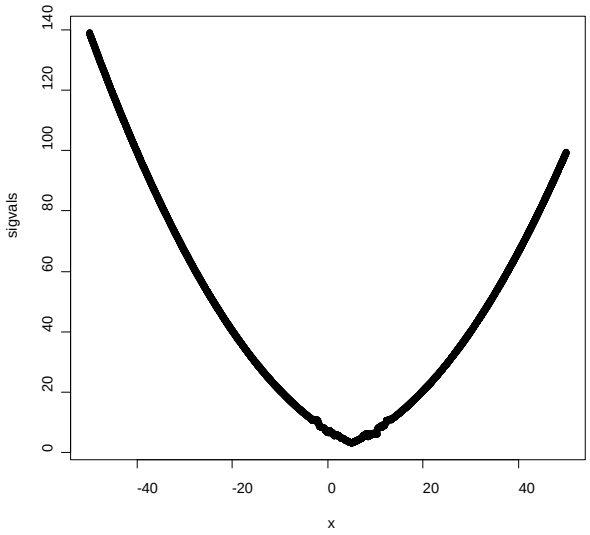


Accepted 24162 out of 100000 (24.162%).
final beta = 1.162301
alpha(5.000000) is 0.999400
alpha(7.000000) is 0.644200
alpha(10.000000) is 0.195200
alpha(20.000000) is 0.443100
alpha(30.000000) is 0.397500
alpha(50.000000) is 0.326200
alpha(70.000000) is 0.264600
alpha(120.000000) is 0.169000
alpha(150.000000) is 0.150200
it takes 128.729000 seconds to run the adaptation algorithm
it takes 34.300000 seconds to compute g and sigma
it takes 14.885000 seconds to generate the first Markov Chain
it takes 15.071000 seconds to generate the second Markov Chain
it takes 15.083000 seconds to generate the third Markov Chain
it takes 15.030000 seconds to generate the fourth Markov Chain
it takes 15.525000 seconds to generate the fifth Markov Chain
it takes 687.539000 seconds to test the local acceptance

[1] 14.73459
[1] 0.0653481
[1] 6.995348

2.2.3.9 case 111 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 1, \gamma = 0.1, \text{height} = 1, \text{width} = 1,$ with fixed bandwidth $b_n = 1$

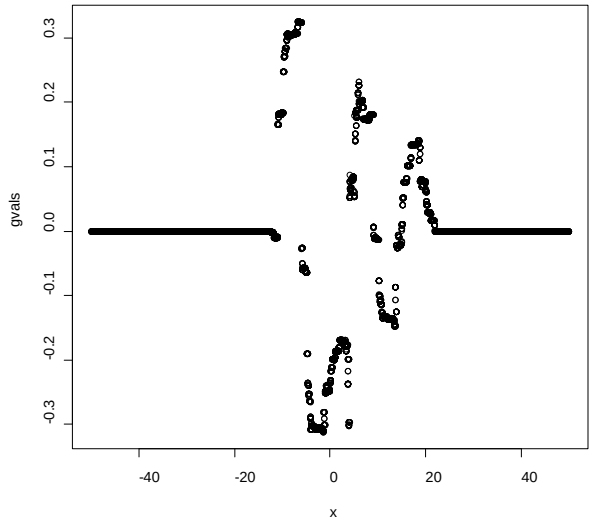


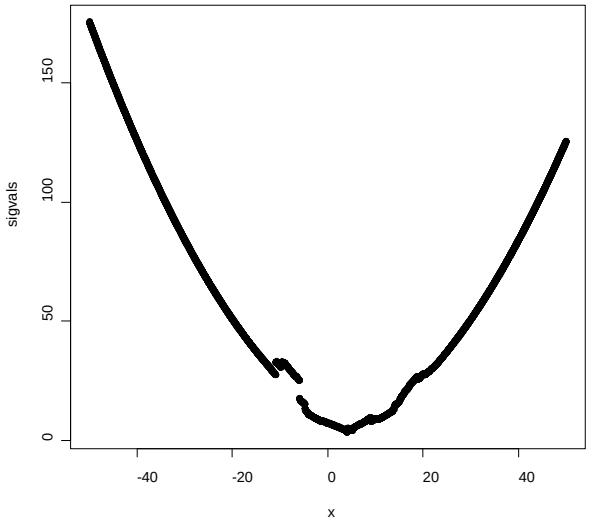


Accepted 24070 out of 100000 (24.070%).
final beta = 1.190445
alpha(5.000000) is 0.998200
alpha(7.000000) is 0.637100
alpha(10.000000) is 0.194100
alpha(20.000000) is 0.431700
alpha(30.000000) is 0.396000
alpha(50.000000) is 0.313700
alpha(70.000000) is 0.260200
alpha(120.000000) is 0.172400
alpha(150.000000) is 0.146100
it takes 130.494000 seconds to run the adaptation algorithm
it takes 34.635000 seconds to compute g and sigma
it takes 15.504000 seconds to generate the first Markov Chain
it takes 15.325000 seconds to generate the second Markov Chain
it takes 15.827000 seconds to generate the third Markov Chain
it takes 15.820000 seconds to generate the fourth Markov Chain
it takes 15.180000 seconds to generate the fifth Markov Chain
it takes 673.514000 seconds to test the local acceptance

[1] 14.21126
[1] 0.0639442
[1] 7.098321

2.2.3.10 case 112 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 5,$ with fixed bandwidth $b_n = 1$

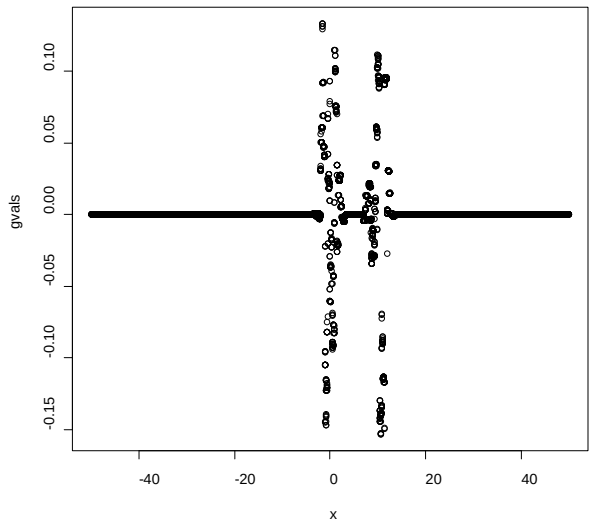


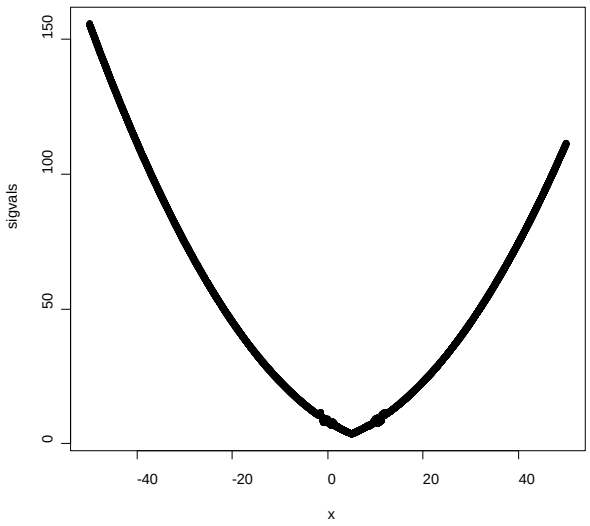


Accepted 22764 out of 100000 (22.764%).
final beta = 1.423704
alpha(5.000000) is 0.979500
alpha(7.000000) is 0.562100
alpha(10.000000) is 0.171100
alpha(20.000000) is 0.362800
alpha(30.000000) is 0.331700
alpha(50.000000) is 0.256200
alpha(70.000000) is 0.207300
alpha(120.000000) is 0.138500
alpha(150.000000) is 0.112700
it takes 124.856000 seconds to run the adaptation algorithm
it takes 33.271000 seconds to compute g and sigma
it takes 17.093000 seconds to generate the first Markov Chain
it takes 16.845000 seconds to generate the second Markov Chain
it takes 16.925000 seconds to generate the third Markov Chain
it takes 17.035000 seconds to generate the fourth Markov Chain
it takes 16.817000 seconds to generate the fifth Markov Chain
it takes 682.008000 seconds to test the local acceptance

[1] 13.13645
[1] 0.06167256
[1] 7.480724

2.2.3.11 case 113 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with fixed bandwidth $b_n = 1$

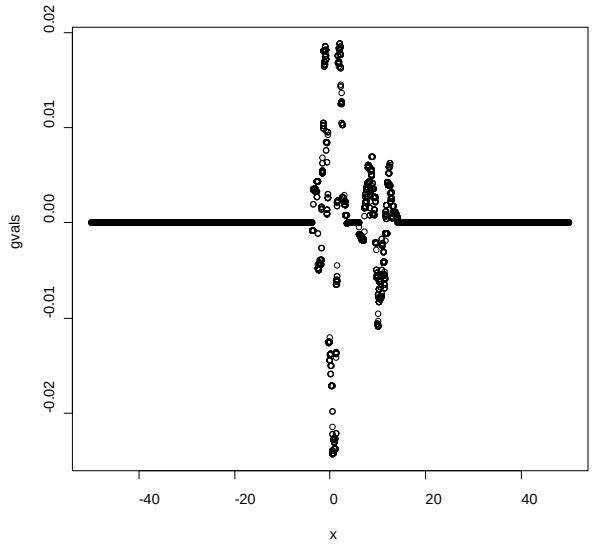


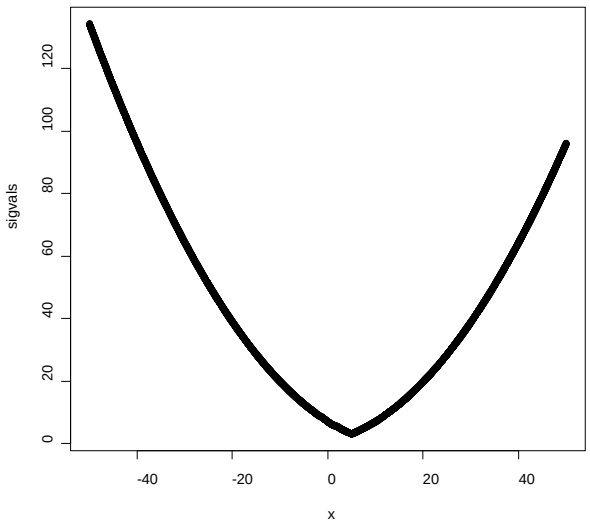


Accepted 22512 out of 100000 (22.512%).
final beta = 1.354120
alpha(5.000000) is 0.992200
alpha(7.000000) is 0.613300
alpha(10.000000) is 0.175600
alpha(20.000000) is 0.391400
alpha(30.000000) is 0.355300
alpha(50.000000) is 0.277000
alpha(70.000000) is 0.221500
alpha(120.000000) is 0.153500
alpha(150.000000) is 0.121700
it takes 124.137000 seconds to run the adaptation algorithm
it takes 33.218000 seconds to compute g and sigma
it takes 96.215000 seconds to generate the first Markov Chain
it takes 17.237000 seconds to generate the second Markov Chain
it takes 16.931000 seconds to generate the third Markov Chain
it takes 17.485000 seconds to generate the fourth Markov Chain
it takes 17.579000 seconds to generate the fifth Markov Chain
it takes 675.932000 seconds to test the local acceptance

[1] 12.13842
[1] 0.05919573
[1] 7.954234

2.2.3.12 case 114 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with fixed bandwidth $b_n = 10$

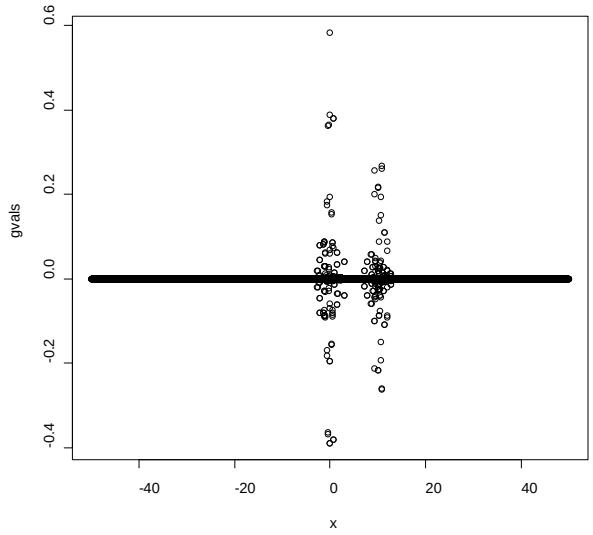


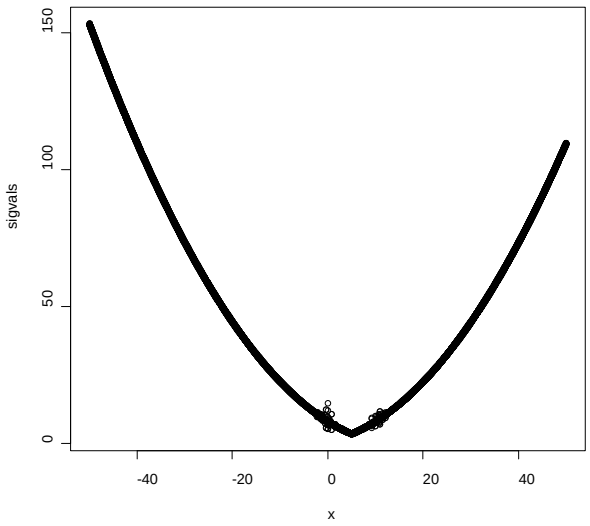


Accepted 24426 out of 100000 (24.426%).
final beta = 1.156102
alpha(5.000000) is 0.999300
alpha(7.000000) is 0.649700
alpha(10.000000) is 0.199600
alpha(20.000000) is 0.443400
alpha(30.000000) is 0.398200
alpha(50.000000) is 0.324700
alpha(70.000000) is 0.264000
alpha(120.000000) is 0.177500
alpha(150.000000) is 0.150800
it takes 129.592000 seconds to run the adaptation algorithm
it takes 34.583000 seconds to compute g and sigma
it takes 15.233000 seconds to generate the first Markov Chain
it takes 15.353000 seconds to generate the second Markov Chain
it takes 15.480000 seconds to generate the third Markov Chain
it takes 15.380000 seconds to generate the fourth Markov Chain
it takes 15.365000 seconds to generate the fifth Markov Chain
it takes 710.588000 seconds to test the local acceptance

[1] 13.40427
[1] 0.06225008
[1] 6.997632

2.2.3.13 case 115 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with fixed bandwidth $b_n = 0.1$

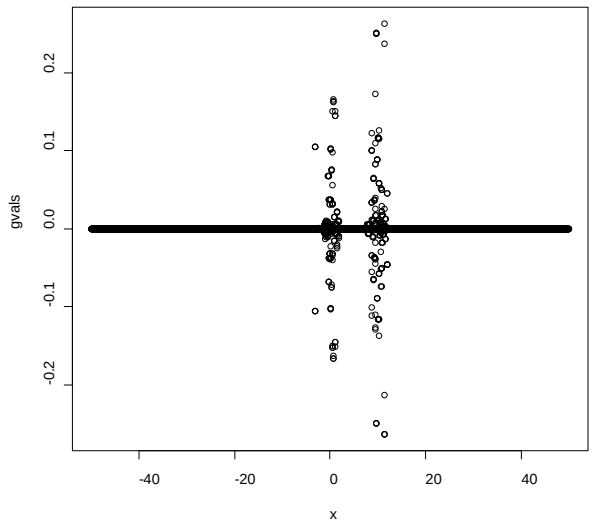


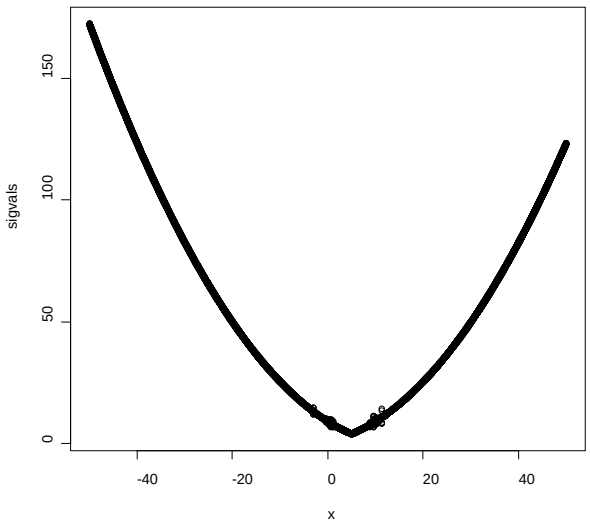


Accepted 22790 out of 100000 (22.790%).
final beta = 1.287911
alpha(5.000000) is 0.996700
alpha(7.000000) is 0.633100
alpha(10.000000) is 0.183000
alpha(20.000000) is 0.406000
alpha(30.000000) is 0.363800
alpha(50.000000) is 0.288600
alpha(70.000000) is 0.233700
alpha(120.000000) is 0.157000
alpha(150.000000) is 0.137300
it takes 131.432000 seconds to run the adaptation algorithm
it takes 35.259000 seconds to compute g and sigma
it takes 17.457000 seconds to generate the first Markov Chain
it takes 17.422000 seconds to generate the second Markov Chain
it takes 17.078000 seconds to generate the third Markov Chain
it takes 17.222000 seconds to generate the fourth Markov Chain
it takes 17.383000 seconds to generate the fifth Markov Chain
it takes 695.595000 seconds to test the local acceptance

[1] 11.9384
[1] 0.05869727
[1] 7.548351

2.2.3.14 case 116 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

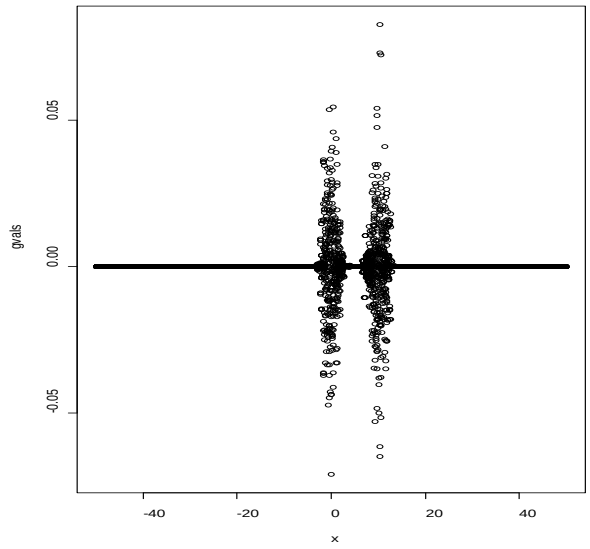


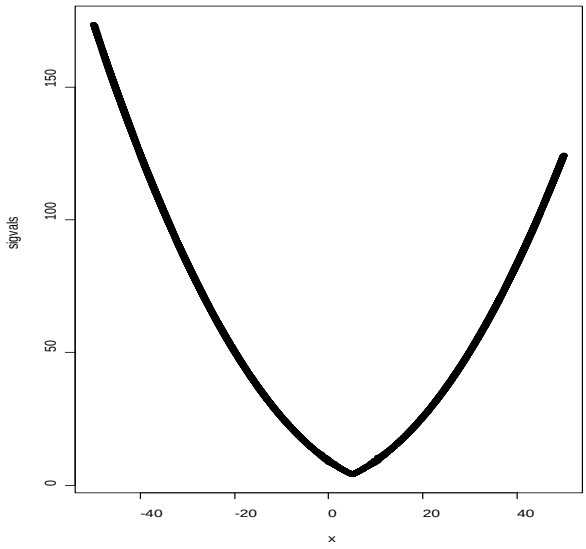


Accepted 21764 out of 100000 (21.764%).
final beta = 1.405223
alpha(5.000000) is 0.990500
alpha(7.000000) is 0.604200
alpha(10.000000) is 0.169900
alpha(20.000000) is 0.384300
alpha(30.000000) is 0.342200
alpha(50.000000) is 0.272700
alpha(70.000000) is 0.209200
alpha(120.000000) is 0.138600
alpha(150.000000) is 0.113700
it takes 127.433000 seconds to run the adaptation algorithm
it takes 34.299000 seconds to compute g and sigma
it takes 19.226000 seconds to generate the first Markov Chain
it takes 19.014000 seconds to generate the second Markov Chain
it takes 19.349000 seconds to generate the third Markov Chain
it takes 20.065000 seconds to generate the fourth Markov Chain
it takes 21.261000 seconds to generate the fifth Markov Chain
it takes 716.572000 seconds to test the local acceptance

[1] 11.16328
[1] 0.05691963
[1] 8.158163

2.2.3.15 case 117 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$





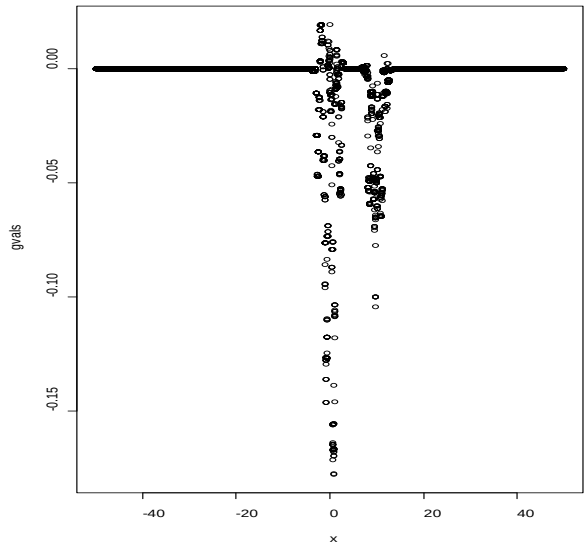
Accepted 21518 out of 100000 (21.518%).
final beta = 1.413473
alpha(5.000000) is 0.988200
alpha(7.000000) is 0.611800
alpha(10.000000) is 0.168800
alpha(20.000000) is 0.380700
alpha(30.000000) is 0.341300
alpha(50.000000) is 0.269300
alpha(70.000000) is 0.220400
alpha(120.000000) is 0.138000
alpha(150.000000) is 0.110700
it takes 129.851000 seconds to run the adaptation algorithm
it takes 34.370000 seconds to compute g and sigma
it takes 19.528000 seconds to generate the first Markov Chain
it takes 20.341000 seconds to generate the second Markov Chain
it takes 19.515000 seconds to generate the third Markov Chain
it takes 19.079000 seconds to generate the fourth Markov Chain
it takes 19.145000 seconds to generate the fifth Markov Chain
it takes 713.662000 seconds to test the local acceptance

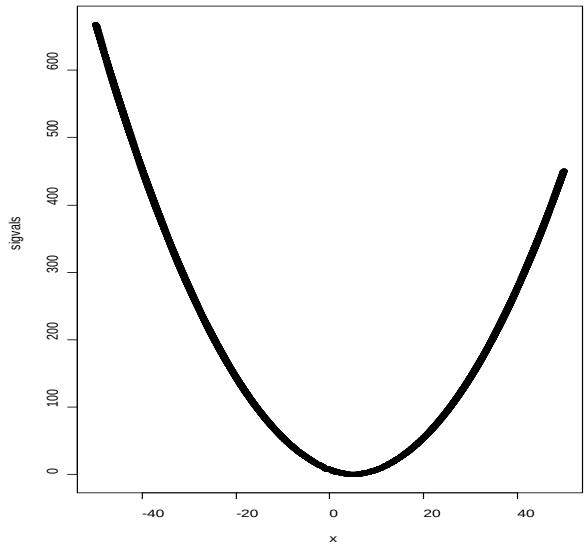
[1] 12.90081
[1] 0.0611555
[1] 7.84211

2.2.4 kernel function $K(x)=\begin{cases}0, |x|\geq 2*width \quad or \quad |x|\leq width\\ 1, width<|x|<2*width\end{cases}$

2.2.4.1 case 119 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $C=1,\gamma=0.5,\text{width}=0.5,$ with fixed bandwidth

$b_n=1$

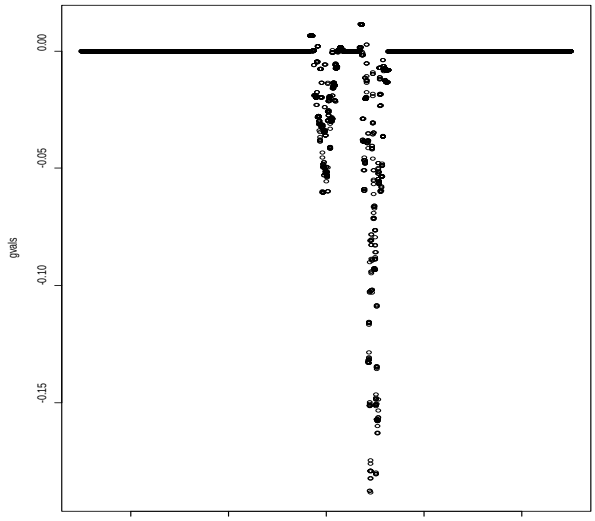


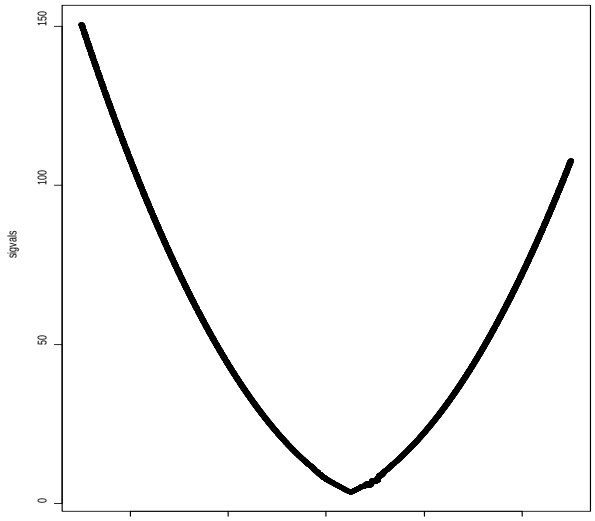


Accepted 23479 out of 100000 (23.479%).
final beta = -1.546046
alpha(5.000000) is 0.979400
alpha(7.000000) is 0.598700
alpha(10.000000) is 0.189100
alpha(20.000000) is 0.184800
alpha(30.000000) is 0.132600
alpha(50.000000) is 0.076700
alpha(70.000000) is 0.051700
alpha(120.000000) is 0.029300
alpha(150.000000) is 0.027400
it takes 309.700000 seconds to run the adaptation algorithm
it takes 84.090000 seconds to compute g and sigma
it takes 126.250000 seconds to generate the first Markov Chain
it takes 127.470000 seconds to generate the second Markov Chain
it takes 127.730000 seconds to generate the third Markov Chain
it takes 127.230000 seconds to generate the fourth Markov Chain
it takes 127.370000 seconds to generate the fifth Markov Chain
it takes 1860.250000 seconds to test the local acceptance

[1] 17.38122
[1] 0.07095952
[1] 6.096523

2.2.4.2 case 120 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 1, \gamma = 0.5, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$



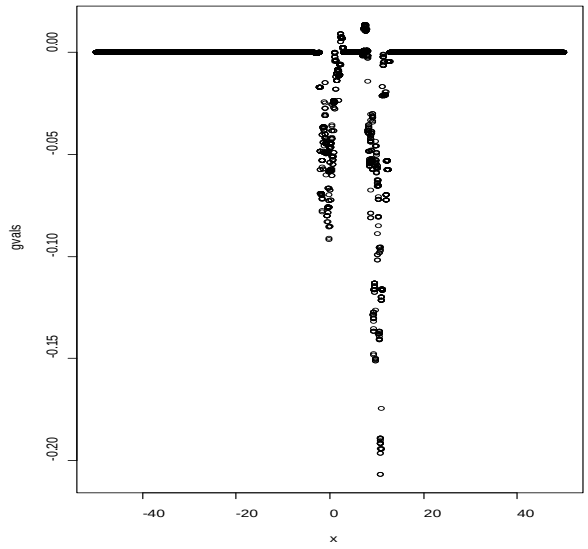


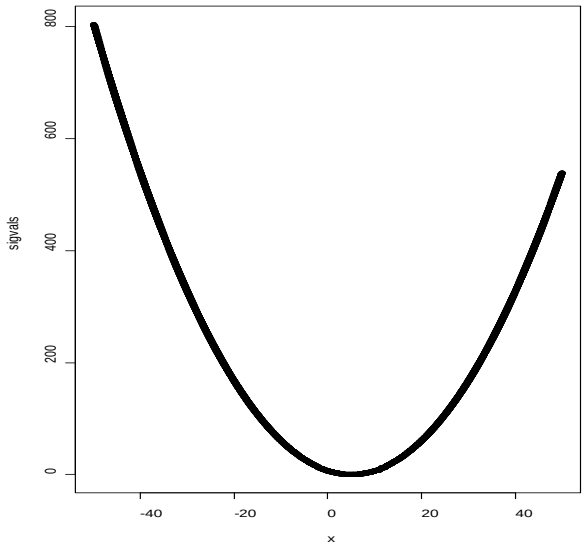
Accepted 23535 out of 100000 (23.535%).
final beta = 1.271283
alpha(5.000000) is 0.995900
alpha(7.000000) is 0.619400
alpha(10.000000) is 0.199300
alpha(20.000000) is 0.416000
alpha(30.000000) is 0.376300
alpha(50.000000) is 0.289800
alpha(70.000000) is 0.240800
alpha(120.000000) is 0.164500
alpha(150.000000) is 0.133500
it takes 310.460000 seconds to run the adaptation algorithm
it takes 84.390000 seconds to compute g and sigma
it takes 121.920000 seconds to generate the first Markov Chain
it takes 121.700000 seconds to generate the second Markov Chain
it takes 122.100000 seconds to generate the third Markov Chain
it takes 121.570000 seconds to generate the fourth Markov Chain
it takes 121.940000 seconds to generate the fifth Markov Chain
it takes 1754.020000 seconds to test the local acceptance

[1] 13.80102
[1] 0.0632373
[1] 7.232841

2.2.4.3 case 121 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 1, \gamma = 0.5, \text{width} = 0.5,$ with fixed bandwidth

$b_n = 1$



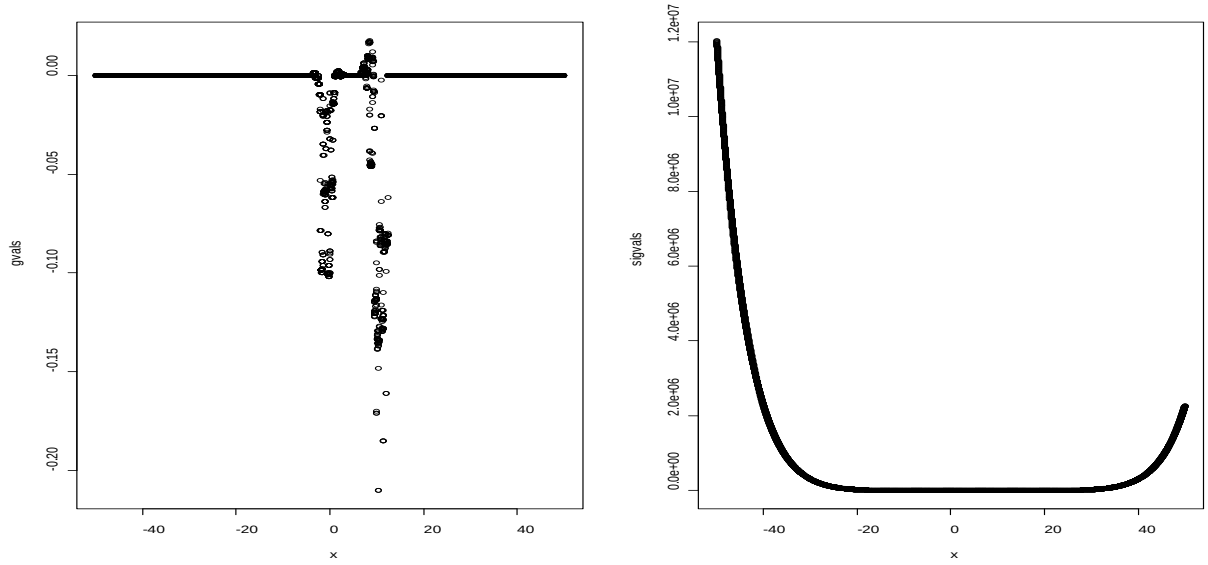


Accepted 24444 out of 100000 (24.444%).
final beta = -5.933267
alpha(5.000000) is 0.951800
alpha(7.000000) is 0.614000
alpha(10.000000) is 0.207400
alpha(20.000000) is 0.164300
alpha(30.000000) is 0.113200
alpha(50.000000) is 0.067500
alpha(70.000000) is 0.043800
alpha(120.000000) is 0.027100
alpha(150.000000) is 0.020600
it takes 309.940000 seconds to run the adaptation algorithm
it takes 84.140000 seconds to compute g and sigma
it takes 125.430000 seconds to generate the first Markov Chain
it takes 124.300000 seconds to generate the second Markov Chain
it takes 125.330000 seconds to generate the third Markov Chain
it takes 124.340000 seconds to generate the fourth Markov Chain
it takes 123.770000 seconds to generate the fifth Markov Chain
it takes 1865.350000 seconds to test the local acceptance

[1] 22.79518
[1] 0.08128286
[1] 4.885462

2.2.4.4 case 121 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 10, \text{width} = 0.5,$ with fixed bandwidth

$b_n = 1$

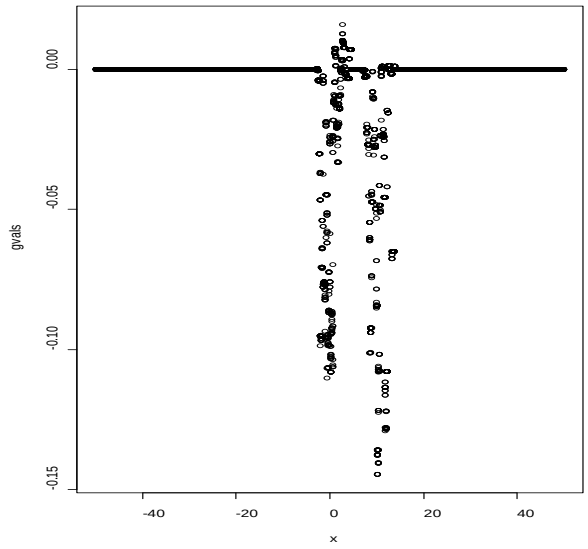


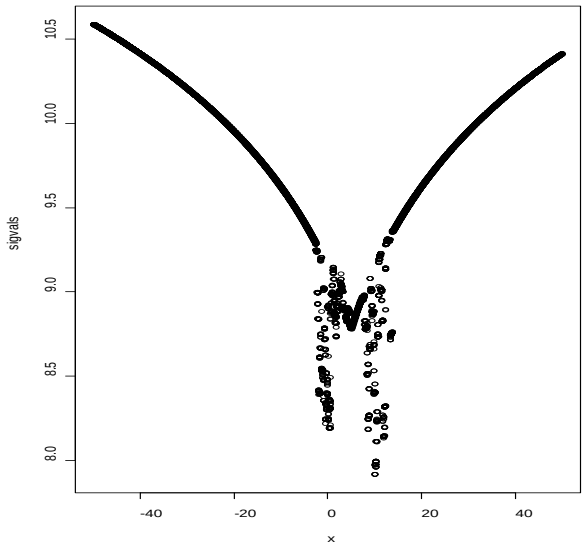
Accepted 27356 out of 100000 (27.356%).
final beta = -2.414094
alpha(5.000000) is 0.992400
alpha(7.000000) is 0.689500
alpha(10.000000) is 0.218500
alpha(20.000000) is 0.011800
alpha(30.000000) is 0.000600
alpha(50.000000) is 0.000000
alpha(70.000000) is 0.000000
alpha(120.000000) is 0.000000
alpha(150.000000) is 0.000000
it takes 309.330000 seconds to run the adaptation algorithm
it takes 83.960000 seconds to compute g and sigma
it takes 127.880000 seconds to generate the first Markov Chain
it takes 131.110000 seconds to generate the second Markov Chain
it takes 129.140000 seconds to generate the third Markov Chain
it takes 129.650000 seconds to generate the fourth Markov Chain
it takes 129.490000 seconds to generate the fifth Markov Chain
it takes 1955.590000 seconds to test the local acceptance

[1] 48.46781
[1] 0.1181587
[1] 2.552771

2.2.4.5 case 122 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{width} = 0.5,$ with fixed bandwidth

$b_n = 1$



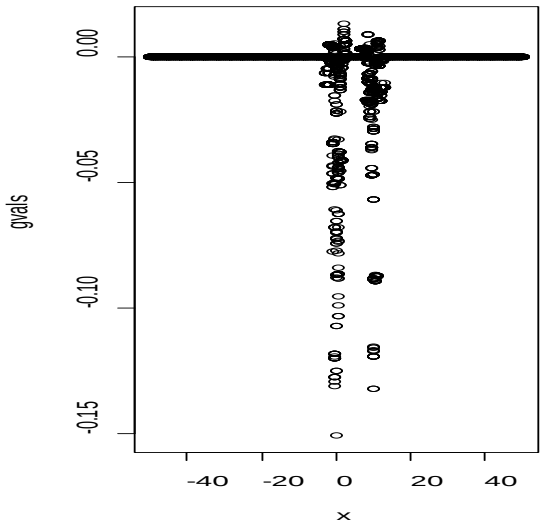


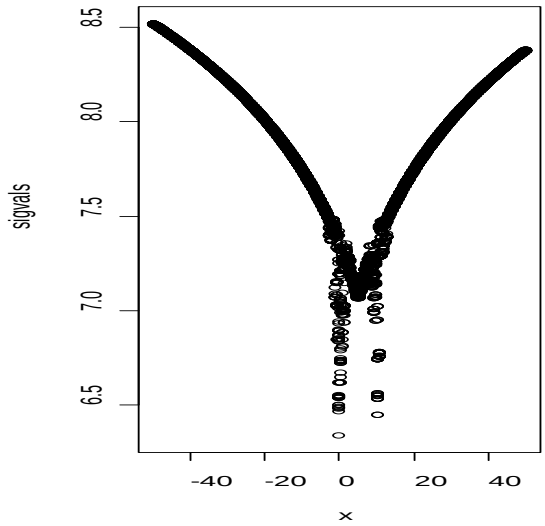
Accepted 21973 out of 100000 (21.973%).
final beta = 2.172493
alpha(5.000000) is 0.743600
alpha(7.000000) is 0.486600
alpha(10.000000) is 0.171900
alpha(20.000000) is 0.507400
alpha(30.000000) is 0.504300
alpha(50.000000) is 0.506300
alpha(70.000000) is 0.504300
alpha(120.000000) is 0.504100
alpha(150.000000) is 0.499500
it takes 309.810000 seconds to run the adaptation algorithm
it takes 84.110000 seconds to compute g and sigma
it takes 124.310000 seconds to generate the first Markov Chain
it takes 124.430000 seconds to generate the second Markov Chain
it takes 124.950000 seconds to generate the third Markov Chain
it takes 124.400000 seconds to generate the fourth Markov Chain
it takes 124.030000 seconds to generate the fifth Markov Chain
it takes 1775.810000 seconds to test the local acceptance

[1] 12.94547
[1] 0.06114334
[1] 7.74818

2.2.4.6 case 123 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{width} = 1,$ with fixed bandwidth

$b_n = 1$

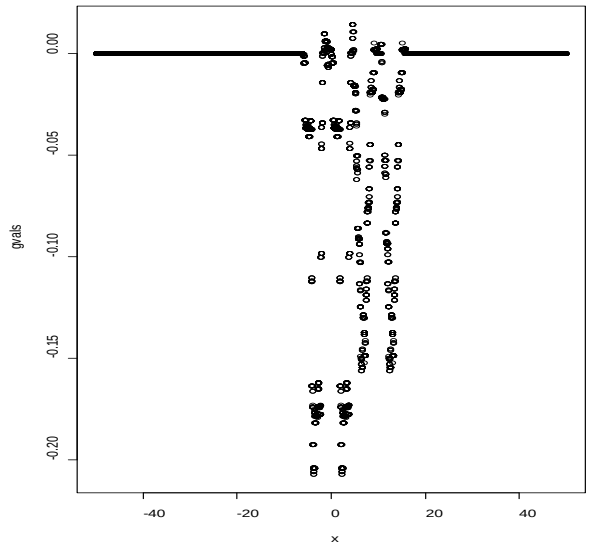


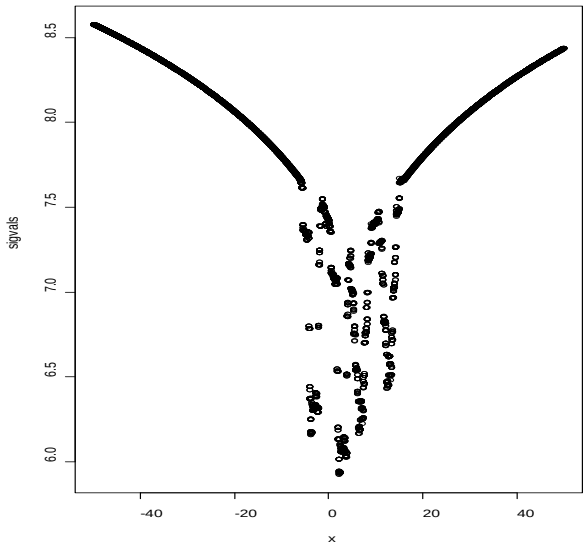


[1] 14.42627
[1] 0.06452843
[1] 7.019186

2.2.4.7 case 124 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{width} = 2,$ with fixed bandwidth

$b_n = 1$



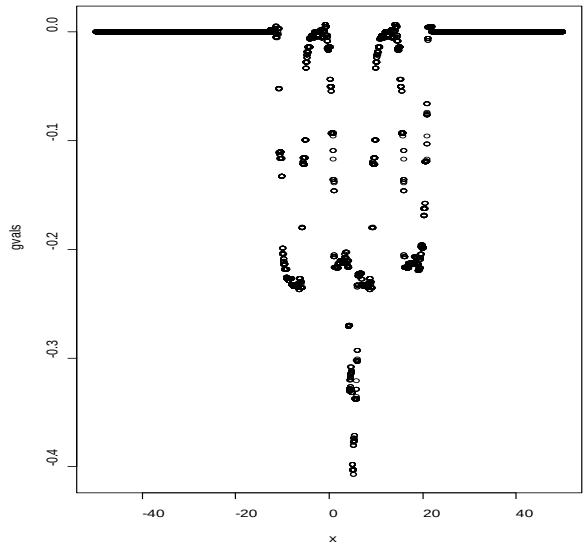


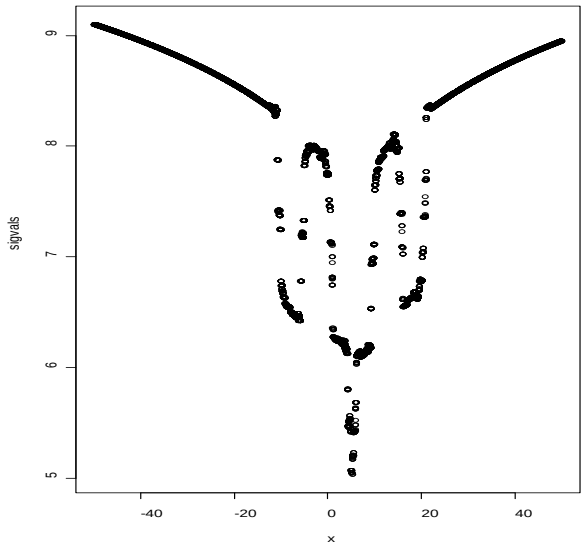
Accepted 24217 out of 100000 (24.217%).
final beta = 1.961981
alpha(5.000000) is 0.855500
alpha(7.000000) is 0.584500
alpha(10.000000) is 0.185600
alpha(20.000000) is 0.505800
alpha(30.000000) is 0.494700
alpha(50.000000) is 0.499600
alpha(70.000000) is 0.494600
alpha(120.000000) is 0.501400
alpha(150.000000) is 0.501700
it takes 307.750000 seconds to run the adaptation algorithm
it takes 84.090000 seconds to compute g and sigma
it takes 118.580000 seconds to generate the first Markov Chain
it takes 118.320000 seconds to generate the second Markov Chain
it takes 118.810000 seconds to generate the third Markov Chain
it takes 118.700000 seconds to generate the fourth Markov Chain
it takes 118.490000 seconds to generate the fifth Markov Chain
it takes 1760.380000 seconds to test the local acceptance

[1] 14.49698
[1] 0.06464175
[1] 6.796207

2.2.4.8 case 125 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{width} = 5,$ with fixed bandwidth

$b_n = 1$



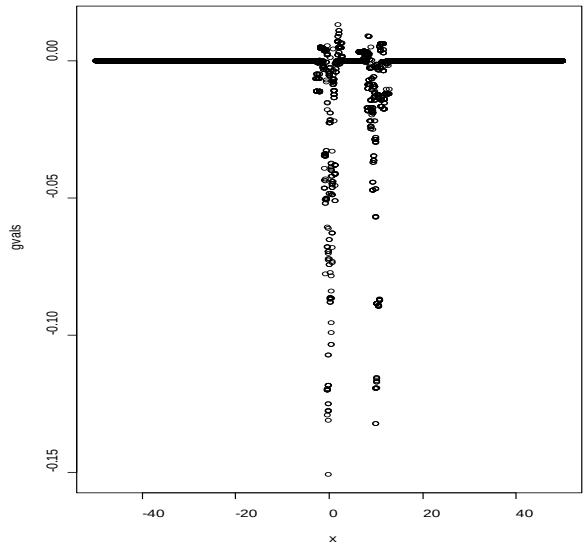


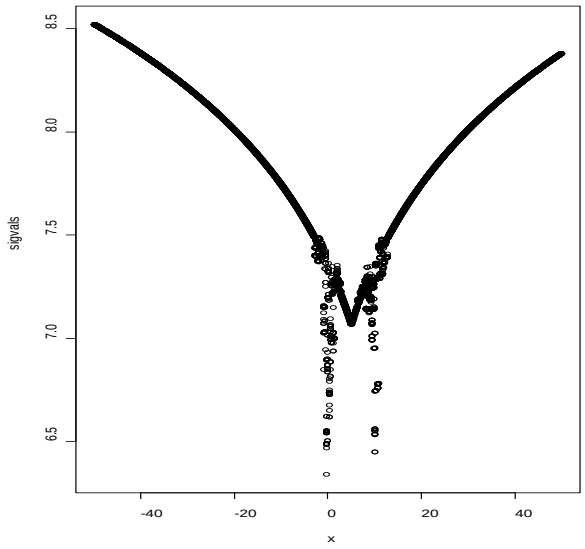
Accepted 24435 out of 100000 (24.435%).
final beta = 2.021228
alpha(5.000000) is 0.953600
alpha(7.000000) is 0.595700
alpha(10.000000) is 0.194000
alpha(20.000000) is 0.508900
alpha(30.000000) is 0.505300
alpha(50.000000) is 0.498600
alpha(70.000000) is 0.496300
alpha(120.000000) is 0.496100
alpha(150.000000) is 0.495600
it takes 307.200000 seconds to run the adaptation algorithm
it takes 83.720000 seconds to compute g and sigma
it takes 119.290000 seconds to generate the first Markov Chain
it takes 118.780000 seconds to generate the second Markov Chain
it takes 119.190000 seconds to generate the third Markov Chain
it takes 119.890000 seconds to generate the fourth Markov Chain
it takes 119.000000 seconds to generate the fifth Markov Chain
it takes 1754.180000 seconds to test the local acceptance

[1] 14.63965
[1] 0.06480795
[1] 7.008607

2.2.4.9 case 126 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{width} = 0.2,$ with fixed bandwidth

$b_n = 1$

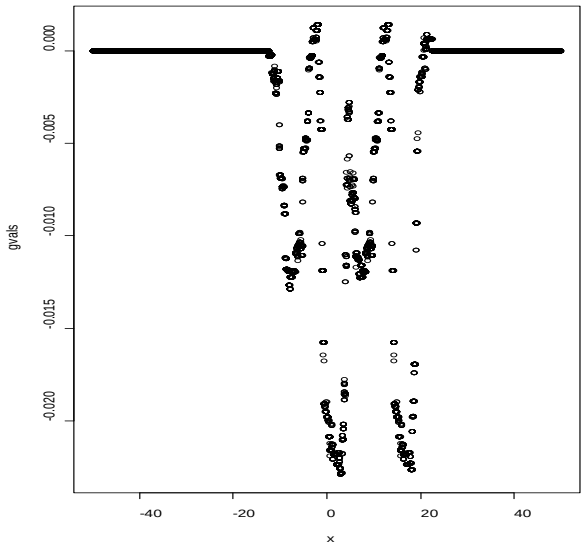


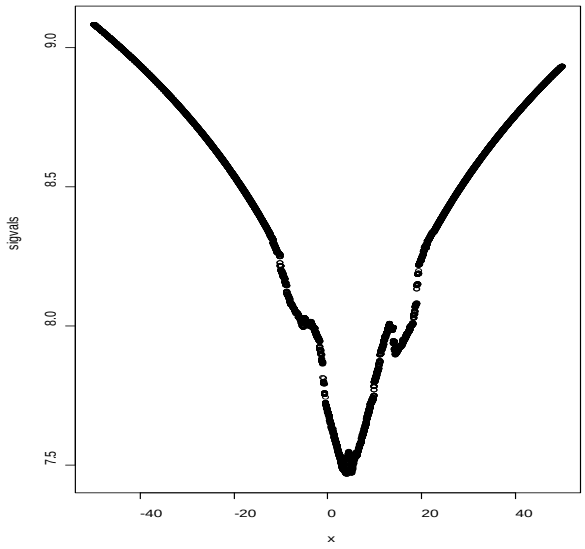


[1] 15.07854
[1] 0.06592749
[1] 6.8665

2.2.4.10 case 127 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{width} = 0.2,$ with fixed bandwidth

$b_n = 10$



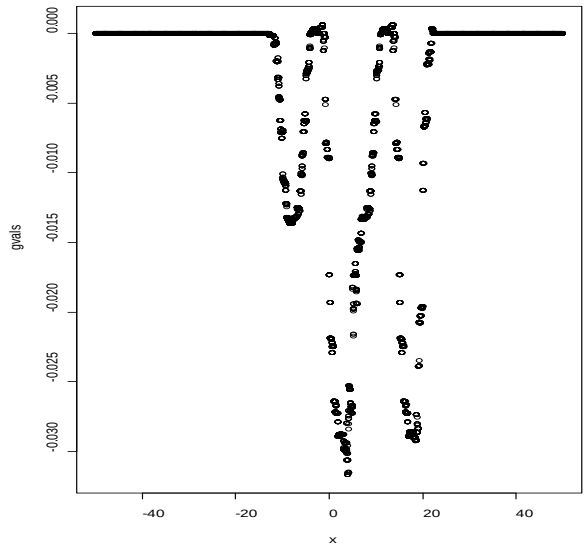


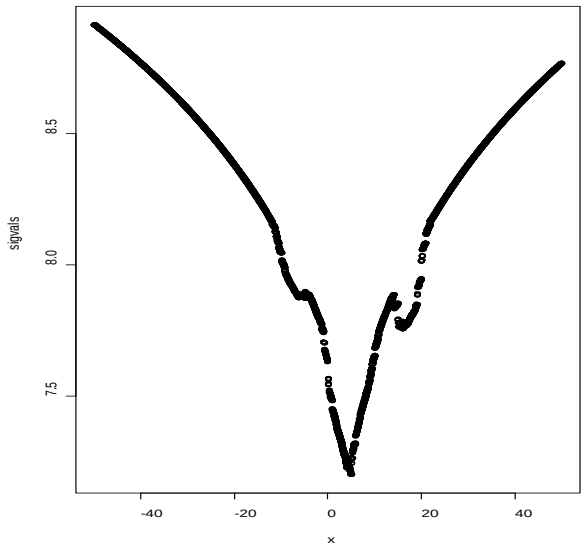
```
> asjd = function(x){
+ val = 0
+ for(i in (B+1):M )
+    val = val +(xlist[i]-xlist[i-1])^2
+ return (val/(M-B))
+ }
> L = length(gvals)
> x = seq(-50,50,length = L)
> M = 10^5
> B = 10^4
> varfact = function(xxx) { 2 * sum(acf(xxx, plot=FALSE)$acf) - 1 }
> asjd = function(x){
+ val = 0
+ for(i in (B+1):M )
+    val = val +(xlist[i]-xlist[i-1])^2
+ return (val/(M-B))
+ }
>
>
>
>
> pdf(file = ifelse(T, "trace plot.pdf","trace plot%03d.pdf"),width=10,height=5,pointsize = 1/200)
> par(mfrow=c(3,2))
>    source("xlist1");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 14.10879
[1] 0.0637883
[1] 7.253909
>    source("xlist2");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 13.49518
[1] 0.06256611
[1] 7.351082
>    source("xlist3");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 13.22538
[1] 0.06178781
[1] 7.209034
>    source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    / sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 13.95147
[1] 0.06346684
[1] 7.0977
```

> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 13.37783
[1] 0.06227059
[1] 7.294676

2.2.4.11 case 128 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{width} = 0.2,$ with fixed bandwidth

$b_n = 0.1$

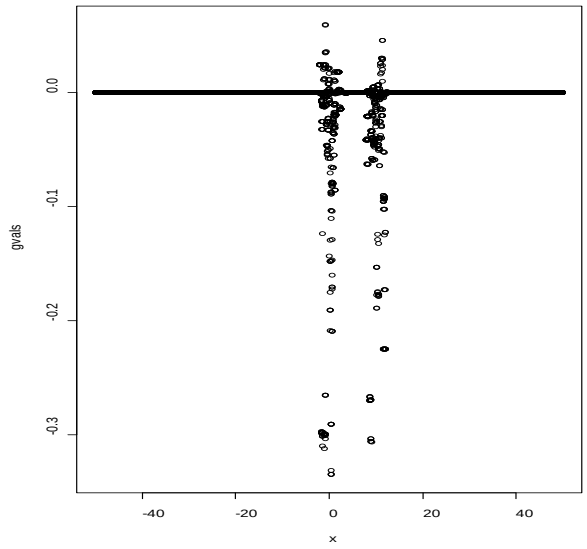


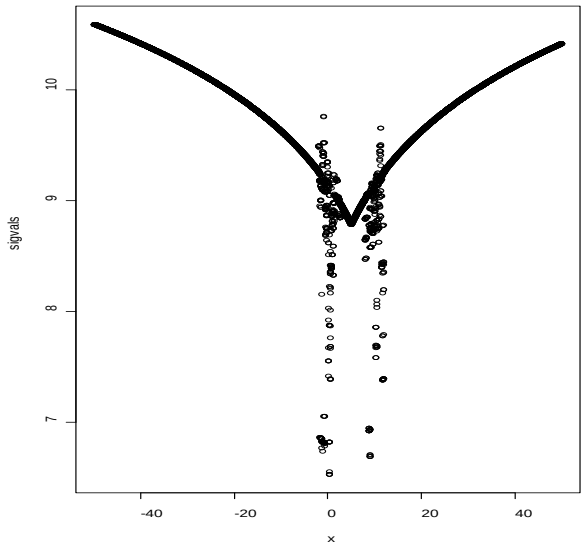


Accepted 23425 out of 100000 (23.425%).
final beta = 2.000723
alpha(5.000000) is 0.835800
alpha(7.000000) is 0.543300
alpha(10.000000) is 0.191800
alpha(20.000000) is 0.509000
alpha(30.000000) is 0.506600
alpha(50.000000) is 0.507000
alpha(70.000000) is 0.499800
alpha(120.000000) is 0.503600
alpha(150.000000) is 0.497900
it takes 307.160000 seconds to run the adaptation algorithm
it takes 83.710000 seconds to compute g and sigma
it takes 119.950000 seconds to generate the first Markov Chain
it takes 118.830000 seconds to generate the second Markov Chain
it takes 119.610000 seconds to generate the third Markov Chain
it takes 119.170000 seconds to generate the fourth Markov Chain
it takes 118.640000 seconds to generate the fifth Markov Chain
it takes 1751.980000 seconds to test the local acceptance

[1] 13.83392
[1] 0.06320622
[1] 7.1248

2.2.4.12 case 127 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{width} = 0.2,$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

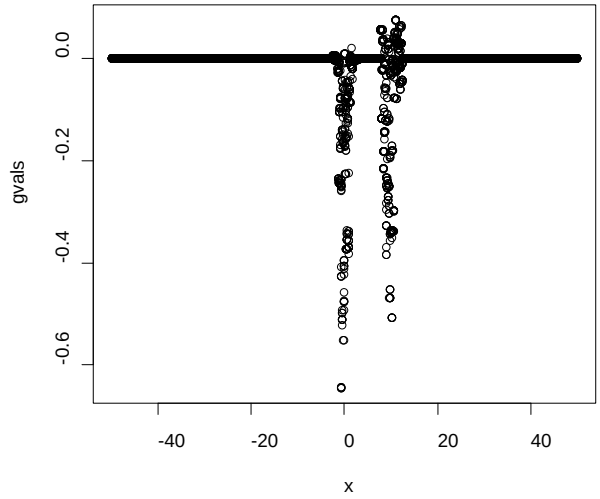


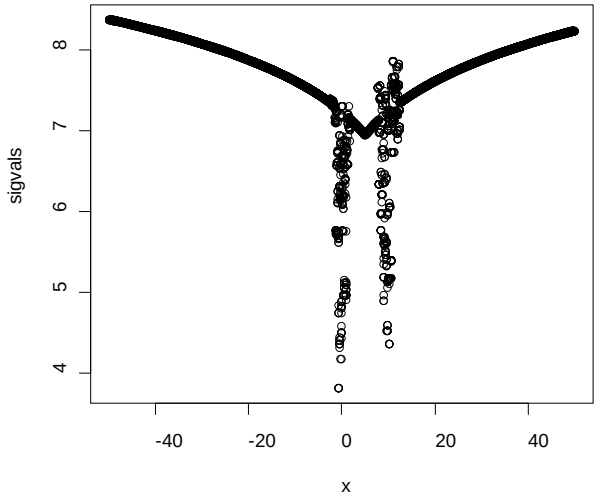


[1] 14.64450
[1] 0.0651862
[1] 7.411176

2.2.4.13 case 128 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $C = 0.1, \gamma = 0.1, \text{width} = 0.2,$ with fixed bandwidth

$b_n = 1$





Accepted 24581 out of 100000 (24.581%).
final beta = 1.938198
alpha(5.000000) is 0.851400
alpha(7.000000) is 0.553300
alpha(10.000000) is 0.197900
alpha(20.000000) is 0.503200
alpha(30.000000) is 0.494000
alpha(50.000000) is 0.498300
alpha(70.000000) is 0.496000
alpha(120.000000) is 0.496600
alpha(150.000000) is 0.512100
it takes 312.800000 seconds to run the adaptation algorithm
it takes 83.730000 seconds to compute g and sigma
it takes 117.950000 seconds to generate the first Markov Chain
it takes 118.380000 seconds to generate the second Markov Chain
it takes 118.000000 seconds to generate the third Markov Chain
it takes 117.490000 seconds to generate the fourth Markov Chain
it takes 117.520000 seconds to generate the fifth Markov Chain
it takes 1752.870000 seconds to test the local acceptance

[1] 18.1577
[1] 0.07247686
[1] 5.867065

2.2.5 constant $\sigma(x) = C$

```
> cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
> cat("acceptance rate =", num/M, "\n");
acceptance rate = 0.23382
>
> varf;
[1] 13.13618
> se;
[1] 0.0616654
> asjd;
[1] 7.204595


> cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
> cat("acceptance rate =", num/M, "\n");
acceptance rate = 0.23263
>
> varf;
[1] 13.87016
> se;
[1] 0.06330728
> asjd;
[1] 7.282846

> cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
> cat("acceptance rate =", num/M, "\n");
acceptance rate = 0.23461
>
> varf;
[1] 13.78088
> se;
[1] 0.06301751
> asjd;
[1] 7.396178


> cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
> cat("acceptance rate =", num/M, "\n");
acceptance rate = 0.23545
```

```
> varf;
[1] 13.15203
> se;
[1] 0.0615862
> asjd;
[1] 7.319729

> cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
> cat("acceptance rate =", num/M, "\n");
acceptance rate = 0.23389
>
> varf;
[1] 14.7803
> se;
[1] 0.0654612
> asjd;
[1] 7.25605


> cat("ran Metropolis algorithm for", M, "iterations, with burn-in", B, "\n");
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
> cat("acceptance rate =", num/M, "\n");
acceptance rate = 0.23557
>
> varf;
[1] 13.29707
> se;
[1] 0.06180272
> asjd;
[1] 7.169893
```

2.2.6 varfact comparison

kernel function $K(x)=e^{-\frac{\lvert x\rvert^{\alpha_1}}{\alpha_2}}$									
α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	17.69815	17.96696	16.85364	17.94989	17.14759
2					14.95411	13.80018	13.74683	14.68709	13.98678
10					14.46429	15.19155	14.90729	14.72928	15.06675
0.1					14.34042	13.87193	13.79645	13.24493	13.59459
0.5					14.83536	15.16301	14.53647	13.99793	13.72507
0.01					22.38718	22.21759	21.64512	21.04189	21.60776
2	10				20.01951	18.42564	17.85715	18.41037	18.24566
	0.1				13.7	13.29679	12.96888	13.27147	12.56142
		10			32.15388	32.82322	33.58812	32.00728	33.06702
		1			13.00467	14.39601	13.2234	13.0975	12.76278
		0.1			13.76117	14.83305	15.0663	14.10183	14.24509
		2	10		12.86394	12.7031	11.99687	12.00228	12.88139
			0.1		15.77916	14.91464	14.67844	16.82218	15.5557
			$\frac{1}{n^{0.2}}$		16.43819	16.60408	17.4054	17.26619	17.61966
			10	$\frac{1}{(n+5)^{0.8}}$	13.23106	13.0639	13.65704	13.23585	13.18471

The best case is;

case 78 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=2,\alpha_2=1,C=0.1,\gamma=2$ with fixed bandwidth $b_n=10$

kernel function $K(x)=\frac{1}{1+\alpha_1\lvert x\rvert^{\alpha_2}}$									
α_1	γ	C	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	1	1	$\frac{1}{(n+5)^{0.5}}$	20.27012	20.11195	21.12319	19.69325	19.49482
10					15.69775	16.31847	17.38448	15.9035	16.85428
0.1					17.99677	19.78633	17.97754	19.07744	18.48608
12					19.31375	18.98969	18.15367	19.02709	19.10133
5					20.26718	19.14768	20.01333	19.97645	19.09234
10	10				55.52678	52.85685	52.491	54.99053	53.60444
	0.1				13.12142	12.64089	13.28427	12.47632	12.47803
		10			13.58733	12.41142	12.40737	13.24543	12.94998
		0.1			13.04691	13.63237	13.14097	12.45919	13.93196
		10	10		15.32114	14.14721	14.81707	14.93572	14.33295
			0.1		18.64611	17.79232	19.85046	18.25231	18.55894

			$\frac{1}{n^{0.2}}$		13.61764	13.98172	13.41386	13.11681	14.00296
			1	$\frac{1}{(n+5)^{0.8}}$	18.34272	18.0786	17.18455	19.03875	18.03951

The best case is :

case 97 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.1$ with fixed bandwidth $b_n = 1$

kernel function $K(x) = \begin{cases} 0, & |x| \geq 2 * width \\ -1 * hegith, & width < |x| < 2 * width \\ 1 * height, & |x| \leq width \end{cases}$

C	γ	height	width	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	0.5	0.5	1	$\frac{1}{(n+5)^{0.5}}$	15.50201	16.26951	17.4331	15.55692	15.87749
10						17.98869	19.55673	19.11706	19.4903	18.30191
0.1						13.16841	12.91911	14.08071	13.82063	13.42696
	10					36.04216	36.61575	35.55663	36.17935	36.67233
	0.1					13.99116	14.12284	15.72336	13.55838	13.54641
	2	1				12.50317	12.30035	13.56912	12.80086	13.20776
		5				18.81126	18.54136	18.27606	19.26981	19.00901
		0.1				14.13331	14.61382	14.34998	13.46336	14.73459
	0.1	1	1			15.09972	14.44022	14.08979	14.83034	14.21126
			5			12.90002	13.21733	13.23861	13.22283	13.13645
			0.1	1		12.57327	12.491	13.2023	12.02869	12.13842
				10		13.50934	14.19841	14.90362	14.71978	13.40427
				0.1		12.42918	12.50971	12.60511	13.22798	11.9384
				$\frac{1}{n^{0.2}}$		11.79056	12.93844	12.33559	12.7613	11.16328
					$\frac{1}{(n+5)^{0.8}}$	12.55904	12.68916	12.20365	12.05565	12.90081

The best case is:

case 116 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

kernel function $K(x) = \begin{cases} 0, & |x| \geq 2 * width \quad or \quad |x| \leq width \\ 1, & width < |x| < 2 * width \end{cases}$

C	γ	width	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	0.5	1	$\frac{1}{(n+5)^{0.5}}$	16.51565	18.22744	15.92406	17.40298	17.38122
0.1					12.6270	12.55610	14.12581	14.04699	13.80102
10					22.08531	22.36662	24.28318	22.38431	22.79518
0.1	10				46.47284	48.2707	47.34358	46.03923	48.46781
	0.1				13.11484	13.85565	12.77984	13.13842	12.94547
		1			13.23159	14.77552	13.67518	15.07019	14.42627
		2			15.42502	14.14825	14.85676	14.78062	14.49698
		5			14.72672	14.49256	12.60517	13.93011	14.63965
		0.2			14.26320	14.07368	15.16668	15.36069	15.07854
			10		14.10879	13.49518	13.22538	13.95147	13.37783
			0.1		13.29121	13.61542	14.84861	13.48883	13.83392
			$\frac{1}{n^{0.2}}$		13.37948	13.37175	13.69730	13.51921	14.64450

The best case is:

case 122 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{width} = 0.5,$ with fixed bandwidth $b_n = 1$

constant $\sigma(x) = C$				
First run	Second run	Third run	Fourth run	Fifth run
13.13618	13.87016	13.78088	13.15203	13.29707

2.2.7 variance comparison

kernel function $K(x)=e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

α_1	γ	C	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	0.07154074	0.07193655	0.06989788	0.07191396	0.07038708
2					0.06550641	0.06316496	0.0626794	0.06511746	0.06355741
10					0.06475452	0.06649223	0.06553578	0.0652558	0.06591678
0.1					0.06437798	0.0632377	0.06322152	0.06182315	0.06266335
0.5					0.06575282	0.06612688	0.06453477	0.06354389	0.06299655
0.01					0.08035613	0.07999806	0.07908512	0.07824674	0.0791383
2	10				0.07581159	0.07303472	0.07170715	0.07310784	0.07269454
	0.1				0.06287379	0.0619756	0.06102925	0.0619696	0.06021042
		10			0.09599056	0.09769438	0.09826665	0.09655463	0.09774311
		1			0.0613665	0.06454725	0.06190249	0.06136325	0.06078702
		0.1			0.06305008	0.06538476	0.06595366	0.06393927	0.06419357
		2	10		0.06090368	0.06061328	0.05894096	0.05876169	0.06114331
			0.1		0.06755124	0.06559679	0.06504136	0.0696462	0.06697404
			$\frac{1}{n^{0.2}}$		0.06887529	0.06929928	0.07082739	0.07069858	0.0711586
			10	$\frac{1}{(n+5)^{0.8}}$	0.06171668	0.06159626	0.0628181	0.06180789	0.06168104

The best case is:

case 78 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=2,\alpha_2=1,C=0.1,\gamma=2$ with fixed bandwidth $b_n=10$

kernel function $K(x)=\frac{1}{1+\alpha_1|x|^{\alpha_2}}$

α_1	γ	C	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	1	1	$\frac{1}{(n+5)^{0.5}}$	0.07629325	0.07587907	0.0781154	0.07529194	0.07497625
10					0.06719478	0.06871731	0.07076993	0.06748606	0.06995134
0.1					0.07226338	0.07567992	0.07210705	0.07443589	0.07306009
12					0.07443621	0.07443067	0.07245077	0.07427455	0.07426885
5					0.07666722	0.07458284	0.07591355	0.07602797	0.07416466
10	10				0.1277001	0.1222414	0.1220896	0.1253721	0.1244969
	0.1				0.06150226	0.06049887	0.06188916	0.05988082	0.05992151
		10			0.06273084	0.05980796	0.05994844	0.06195356	0.06111524
		0.1			0.06129223	0.06265302	0.06158764	0.06001449	0.06357011
		10	10		0.06658314	0.06392398	0.06548828	0.06572278	0.06422718
			0.1		0.07344185	0.07158091	0.07582998	0.07245237	0.07316876

			$\frac{1}{n^{0.2}}$		0.06273164	0.06365028	0.06226108	0.06163139	0.06365483
			1	$\frac{1}{(n+5)^{0.8}}$	0.07275788	0.07213117	0.07042167	0.07417965	0.07210694

The best case is :

case 97 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.1$ with fixed bandwidth $b_n = 1$

kernel function $K(x) = \begin{cases} 0, & \mid x \mid \geq 2 * width \\ -1 * hegith, & width < \mid x \mid < 2 * width \\ 1 * height, & \mid x \mid \leq width \end{cases}$

C	γ	height	width	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	0.5	0.5	1	$\frac{1}{(n+5)^{0.5}}$	0.06687103	0.06847918	0.07115063	0.06699158	0.06777017
10						0.07202369	0.07508933	0.07414543	0.07515094	0.07264643
0.1						0.0616654	0.060983	0.0637165	0.06336811	0.06233687
	10					0.1019297	0.102977	0.1014364	0.1026182	0.1030286
	0.1					0.06356368	0.06403037	0.06759348	0.06273767	0.06246479
	2	1				0.06018968	0.05974201	0.06253642	0.060799	0.06187482
		5				0.07375515	0.07327441	0.07266018	0.07462279	0.07394524
		0.1				0.06394895	0.06489215	0.0642523	0.06229267	0.0653481
		1	1			0.06595247	0.06471726	0.06365214	0.0653969	0.0639442
			5			0.06086061	0.06180042	0.06187954	0.06189745	0.06167256
			0.1			0.06029126	0.05995545	0.06172432	0.05899454	0.05919573
0.1				10		0.06261301	0.06407868	0.06556438	0.06525447	0.06225008
				0.1		0.05999715	0.06028388	0.06027195	0.06199946	0.05869727
				$\frac{1}{n^{0.2}}$		0.05833108	0.06115153	0.05970294	0.06067416	0.05691963
					$\frac{1}{(n+5)^{0.8}}$	0.06024648	0.06057204	0.05938889	0.05899644	0.0611555

The best case is:

case 116 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

kernel function $K(x)=\begin{cases}0, |x|\geq 2*width \quad or \quad |x|\leq width \\ 1, width<|x|<2*width\end{cases}$

C	γ	width	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	0.5	1	$\frac{1}{(n+5)^{0.5}}$	0.0688377	0.07281504	0.06764052	0.07086825	0.07095952
0.1					0.06047511	0.0601648	0.06383409	0.06352232	0.0632373
10					0.07988052	0.08018424	0.0835769	0.08041703	0.08128286
0.1	10				0.1156726	0.1176081	0.1167444	0.1153350	0.1181587
	0.1				0.06155224	0.06305464	0.06067722	0.06163425	0.06114334
		1			0.06183885	0.06549107	0.06257762	0.0660431	0.06452843
		2			0.0667545	0.06391155	0.0653105	0.06523758	0.06464175
		5			0.06525439	0.06470269	0.0602786	0.06347932	0.06480795
		0.2			0.06396429	0.06371747	0.06614052	0.06661003	0.06592749
			10		0.0637883	0.06256611	0.06178781	0.06346684	0.06227059
			0.1		0.06208175	0.06271901	0.06542974	0.06230577	0.06320622
			$\frac{1}{n^{0.2}}$		0.06218174	0.06223508	0.06291217	0.06254374	0.0651862

The best case is:

case 127 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $C=0.1, \gamma=0.1, \text{width}=0.2,$ with decreasing bandwidth $b_n=\frac{1}{n^{0.2}}$

constant $\sigma(x)=C$

First run	Second run	Third run	Fourth run	Fifth run
0.061665	0.06330728	0.06301751	0.0615862	0.0654612

2.2.8 comparison of average squared jump distance

kernel function $K(x)=e^{-\frac{\lvert x\rvert^{\alpha_1}}{\alpha_2}}$									
α_1	γ	C	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	5.873551	5.901538	5.852935	5.774491	5.77809
2					6.773307	6.938589	6.937717	7.121168	6.943634
10					6.928128	6.721215	6.860596	6.818162	6.664103
0.1					7.004704	6.809467	6.970135	7.051681	7.059283
0.5					6.972854	6.879271	6.854267	6.893609	6.884881
0.01					5.091557	4.989655	4.981934	5.225652	4.975623
2	10				5.651396	5.739741	5.687458	5.634909	5.745729
	0.1				7.69529	7.737276	7.478679	7.668143	7.484434
		10			3.69852	3.628897	3.603881	3.616932	3.609716
		1			7.389716	7.380399	7.403763	7.493079	7.467558
		0.1			7.01781	6.925817	7.223606	7.042545	6.900112
		2	10		7.728795	7.719727	8.015229	7.980781	7.709572
			0.1		6.412002	6.535136	6.507305	6.503781	6.533062
			$\frac{1}{n^{0.2}}$		6.213535	6.22061	6.262421	6.301681	6.218763
			10	$\frac{1}{(n+5)^{0.8}}$	7.503375	7.499092	7.287635	7.384808	7.440597

The best case is;

case 78 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=2,\alpha_2=1,C=0.1,\gamma=2$ with fixed bandwidth $b_n=10$

kernel function $K(x)=\frac{1}{1+\alpha_1\lvert x\rvert^{\alpha_2}}$									
α_1	γ	C	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	1	1	$\frac{1}{(n+5)^{0.5}}$	5.308739	5.376076	5.25213	5.306378	5.323529
10					6.192003	6.254055	5.980946	6.28541	6.095268
0.1					5.777566	5.582482	5.701381	5.649991	5.840868
12					5.630077	5.722548	5.749735	5.689602	5.761007
5					5.512693	5.530929	5.461332	5.455003	5.474942
10	10				1.660918	1.65942	1.720822	1.617675	1.709541
	0.1				7.655875	7.656696	7.482752	7.638246	7.639173
		10			7.445668	7.493031	7.48323	7.579445	7.64341
		0.1			7.437534	7.532984	7.525821	7.588561	7.406249
		10	10		6.988029	6.857484	6.945544	6.965335	7.005803
			0.1		5.491011	5.335033	5.349837	5.463103	5.515589

			$\frac{1}{n^{0.2}}$		7.379925	7.319542	7.527431	7.500853	7.400967
			1	$\frac{1}{(n+5)^{0.8}}$	5.593592	5.552727	5.602003	5.673047	5.518752

The best case is :

case 97 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.1$ with fixed bandwidth $b_n = 1$

kernel function $K(x) = \begin{cases} 0, & |x| \geq 2 * width \\ -1 * hegith, & width < |x| < 2 * width \\ 1 * height, & |x| \leq width \end{cases}$

C	γ	height	width	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	0.5	0.5	1	$\frac{1}{(n+5)^{0.5}}$	6.344871	6.382293	6.453518	6.396776	6.794295
10						5.634184	5.595427	5.579323	5.695321	5.623555
0.1						7.445216	7.253714	7.196647	7.182623	7.217294
	10					3.294306	3.369713	3.355714	3.43006	3.271804
	0.1					7.065596	6.907671	7.017294	6.967323	7.10335
	2	1				7.621703	7.488763	7.694096	7.716375	7.630405
		5				6.122762	5.888783	6.063521	5.941985	6.026763
		0.1				6.942994	7.18058	7.090891	7.165429	6.995348
		1	1			7.100944	7.212488	7.00521	7.118056	7.098321
			5			7.628372	7.578396	7.483249	7.48983	7.480724
			0.1			8.049105	7.920465	7.715893	7.935231	7.954234
				10		7.23596	7.043965	6.929782	6.959583	6.997632
				0.1		7.776327	7.678879	7.621687	7.649541	7.548351
				$\frac{1}{n^{0.2}}$		8.03459	8.031316	7.766907	7.8188	8.158163
					$\frac{1}{(n+5)^{0.8}}$	7.752874	8.032417	8.022584	8.001408	7.84211

The best case is:

case 116 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $C = 0.1, \gamma = 0.1, \text{height} = 1, \text{width} = 0.1,$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

kernel function $K(x)=\begin{cases}0, \lvert x\rvert \geq 2*width \quad or \quad \lvert x\rvert \leq width \\ 1, width < \lvert x\rvert < 2*width\end{cases}$									
C	γ	width	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	2	0.5	1	$\frac{1}{(n+5)^{0.5}}$	6.012644	5.993335	6.183591	5.966074	6.096523
0.1					7.403707	7.4661	7.376348	7.236378	7.232841
10					5.024682	4.924248	4.869332	4.788359	4.885462
0.1	10				2.656734	2.535808	2.508755	2.621832	2.552771
	0.1				7.522012	7.63438	7.592666	7.63681	7.74818
		1			6.954202	6.927666	7.030507	6.864217	7.019186
		2			6.956227	6.856488	6.83983	6.806807	6.8665
		5			6.7738	6.86222	6.809493	6.83525	6.796207
		0.2			6.944817	7.050115	7.056753	7.206524	7.008607
			10		7.253909	7.351082	7.209034	7.0977	7.294676
			0.1		7.361709	7.091562	7.160905	7.246703	7.1248
			$\frac{1}{n^{0.2}}$		7.598452	7.649363	7.587748	7.615007	7.411176

The best case is:

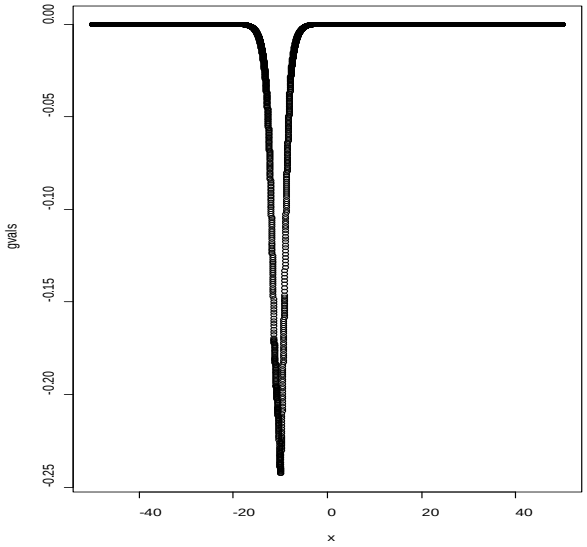
case 122 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $C=0.1, \gamma=0.1, \text{width}=0.5,$ with fixed bandwidth $b_n=1$

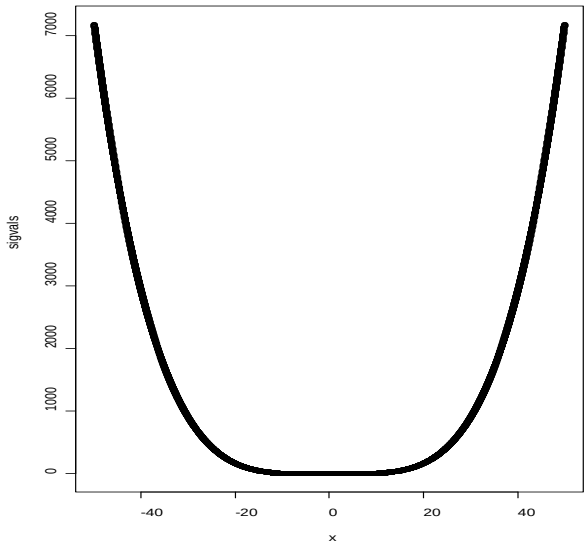
constant $\sigma(x)=C$				
First run	Second run	Third run	Fourth run	Fifth run
7.204595	7.282846	7.396178	7.31972	7.25605

2.3 Example 3: mixture of three normal distributions in R^1

2.3.1 kernel function $K(x) = e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

2.3.1.1 case 129 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

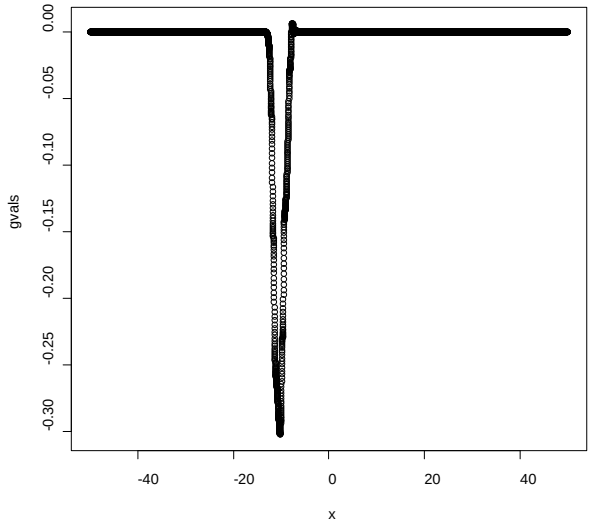


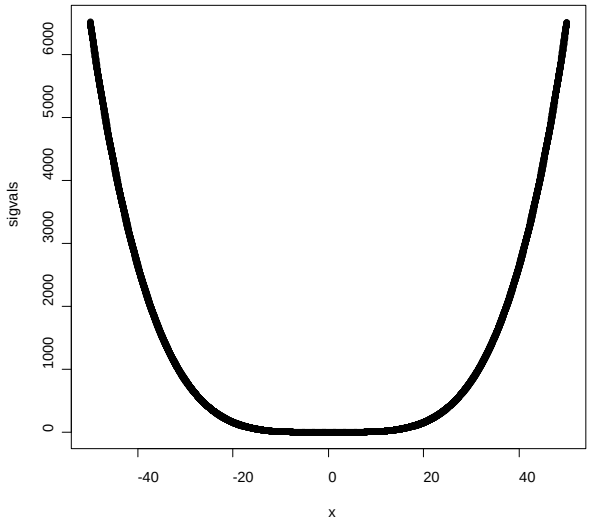


Accepted 76483 out of 100000 (76.483%).
final beta = -6.748936
alpha(0.000000) is 0.765200
alpha(2.000000) is 0.727800
alpha(5.000000) is 0.958900
alpha(8.000000) is 0.590400
alpha(10.000000) is 0.155000
alpha(13.000000) is 0.121000
alpha(15.000000) is 0.100400
alpha(20.000000) is 0.058200
alpha(30.000000) is 0.018300
alpha(50.000000) is 0.004300
it takes 447.440000 seconds to run the adaptation algorithm
it takes 198.950000 seconds to compute g and sigma
it takes 166.460000 seconds to generate the first Markov Chain
it takes 166.820000 seconds to generate the second Markov Chain
it takes 168.250000 seconds to generate the third Markov Chain
it takes 169.860000 seconds to generate the fourth Markov Chain
it takes 166.760000 seconds to generate the fifth Markov Chain
it takes 1317.690000 seconds to test the local acceptance

[1] 0.2709634

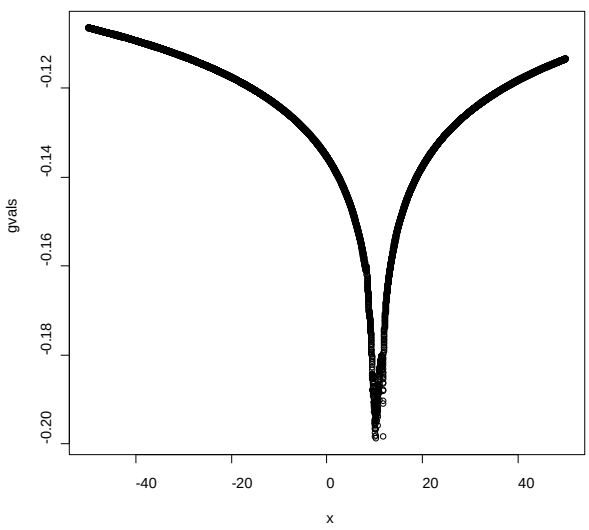
2.3.1.2 case 130 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

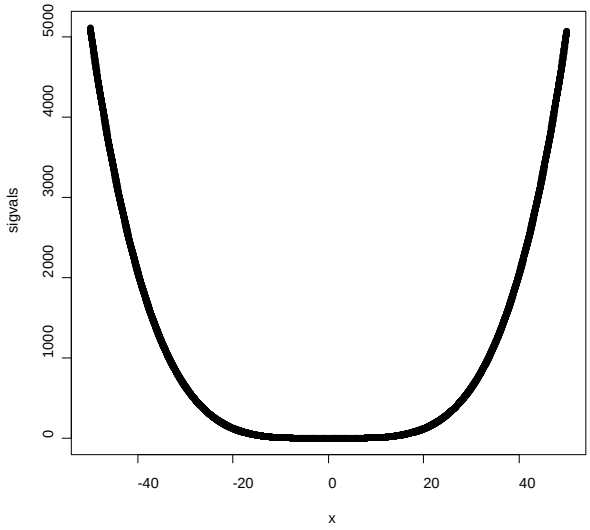




Accepted 23986 out of 100000 (23.986%).
final beta = -6.847073
alpha(0.000000) is 0.786900
alpha(2.000000) is 0.745000
alpha(5.000000) is 0.953800
alpha(8.000000) is 0.605300
alpha(10.000000) is 0.166700
alpha(13.000000) is 0.119100
alpha(15.000000) is 0.105400
alpha(20.000000) is 0.064500
alpha(30.000000) is 0.021300
alpha(50.000000) is 0.005200
it takes 770.470000 seconds to run the adaptation algorithm
it takes 353.150000 seconds to compute g and sigma
it takes 167.060000 seconds to generate the first Markov Chain
it takes 167.030000 seconds to generate the second Markov Chain
it takes 165.370000 seconds to generate the third Markov Chain
it takes 170.350000 seconds to generate the fourth Markov Chain
it takes 164.040000 seconds to generate the fifth Markov Chain
it takes 1752.270000 seconds to test the local acceptance

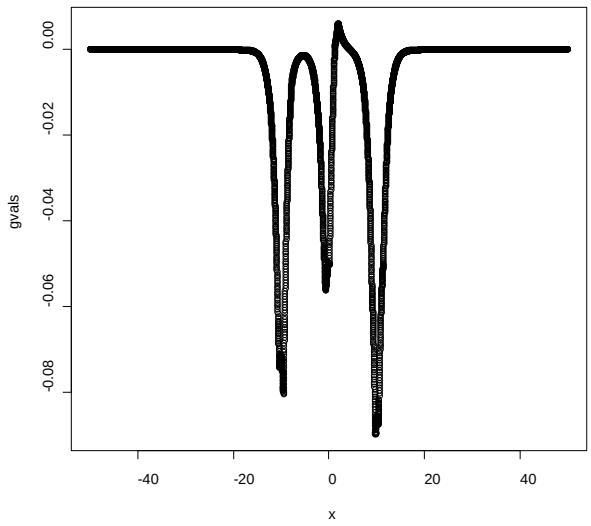
2.3.1.3 case 131 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

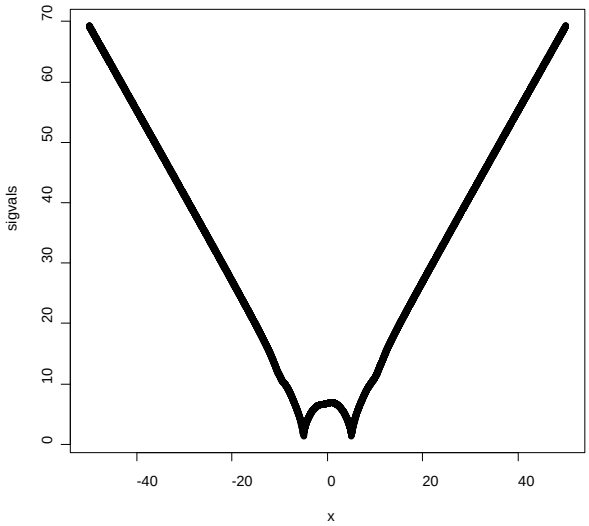




Accepted 83273 out of 100000 (83.273%).
final beta = -6.981961
alpha(0.000000) is 0.855400
alpha(2.000000) is 0.781100
alpha(5.000000) is 0.956300
alpha(8.000000) is 0.638800
alpha(10.000000) is 0.224100
alpha(13.000000) is 0.139900
alpha(15.000000) is 0.134600
alpha(20.000000) is 0.079600
alpha(30.000000) is 0.028900
alpha(50.000000) is 0.005400
it takes 648.690000 seconds to run the adaptation algorithm
it takes 283.420000 seconds to compute g and sigma
it takes 165.590000 seconds to generate the first Markov Chain
it takes 169.070000 seconds to generate the second Markov Chain
it takes 165.930000 seconds to generate the third Markov Chain
it takes 167.700000 seconds to generate the fourth Markov Chain
it takes 166.860000 seconds to generate the fifth Markov Chain
it takes 1386.410000 seconds to test the local acceptance

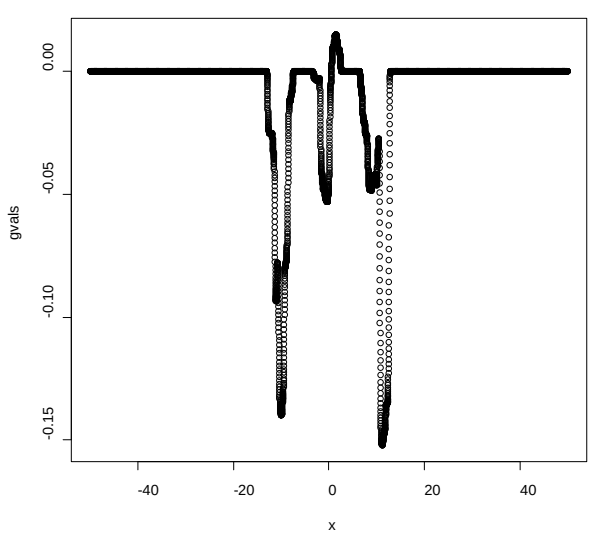
2.3.1.4 case 132 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

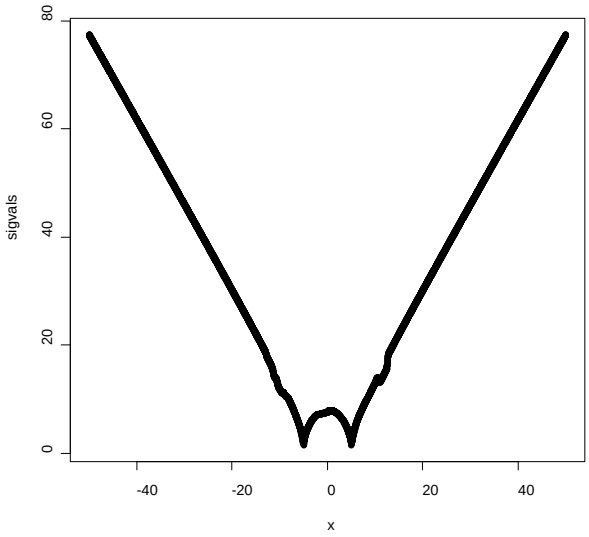




Accepted 24965 out of 100000 (24.965%).
final beta = 0.330946
alpha(0.000000) is 0.238200
alpha(2.000000) is 0.473400
alpha(5.000000) is 0.976900
alpha(8.000000) is 0.389100
alpha(10.000000) is 0.174500
alpha(13.000000) is 0.311900
alpha(15.000000) is 0.397100
alpha(20.000000) is 0.426900
alpha(30.000000) is 0.431100
alpha(50.000000) is 0.425900
it takes 446.820000 seconds to run the adaptation algorithm
it takes 197.800000 seconds to compute g and sigma
it takes 166.480000 seconds to generate the first Markov Chain
it takes 166.300000 seconds to generate the second Markov Chain
it takes 167.630000 seconds to generate the third Markov Chain
it takes 170.040000 seconds to generate the fourth Markov Chain
it takes 167.520000 seconds to generate the fifth Markov Chain
it takes 701.980000 seconds to test the local acceptance

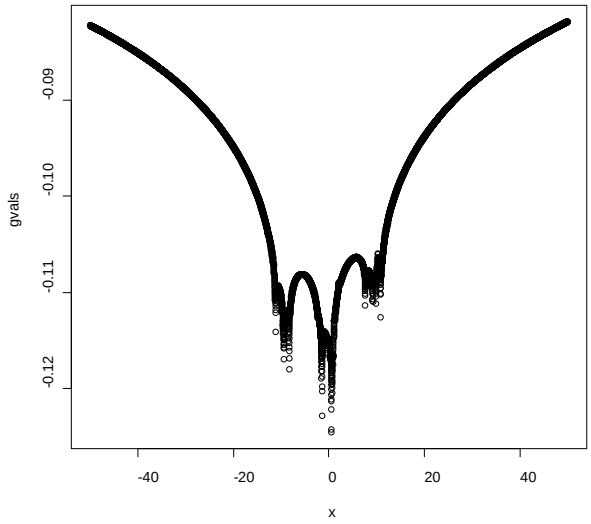
2.3.1.5 case 131 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

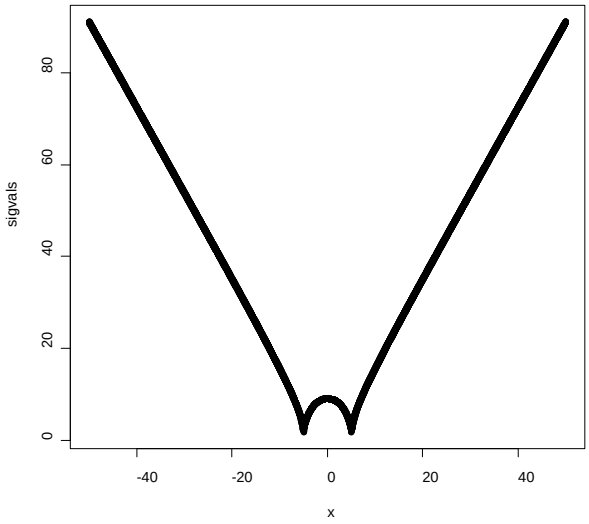




Accepted 24286 out of 100000 (24.286%).
final beta = 0.441691
alpha(0.000000) is 0.218800
alpha(2.000000) is 0.458700
alpha(5.000000) is 0.977800
alpha(8.000000) is 0.373500
alpha(10.000000) is 0.168400
alpha(13.000000) is 0.313600
alpha(15.000000) is 0.386900
alpha(20.000000) is 0.409000
alpha(30.000000) is 0.406700
alpha(50.000000) is 0.398200
it takes 776.670000 seconds to run the adaptation algorithm
it takes 347.250000 seconds to compute g and sigma
it takes 168.790000 seconds to generate the first Markov Chain
it takes 169.200000 seconds to generate the second Markov Chain
it takes 169.140000 seconds to generate the third Markov Chain
it takes 170.260000 seconds to generate the fourth Markov Chain
it takes 170.010000 seconds to generate the fifth Markov Chain
it takes 1136.300000 seconds to test the local acceptance

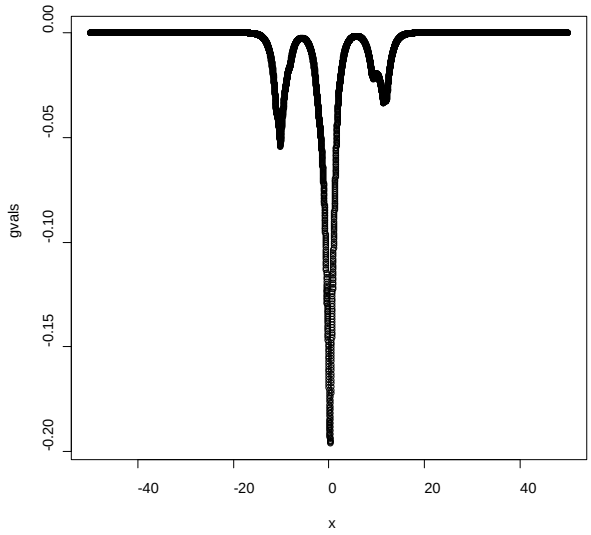
2.3.1.6 case 132 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

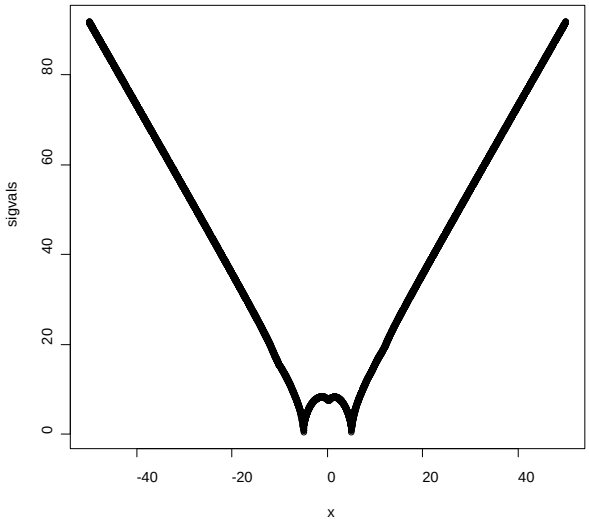




Accepted 22621 out of 100000 (22.621%).
final beta = 0.687387
alpha(0.000000) is 0.187300
alpha(2.000000) is 0.435400
alpha(5.000000) is 0.983100
alpha(8.000000) is 0.348400
alpha(10.000000) is 0.155500
alpha(13.000000) is 0.290400
alpha(15.000000) is 0.359100
alpha(20.000000) is 0.373600
alpha(30.000000) is 0.361100
alpha(50.000000) is 0.370800
it takes 650.950000 seconds to run the adaptation algorithm
it takes 282.750000 seconds to compute g and sigma
it takes 177.640000 seconds to generate the first Markov Chain
it takes 178.130000 seconds to generate the second Markov Chain
it takes 178.750000 seconds to generate the third Markov Chain
it takes 178.560000 seconds to generate the fourth Markov Chain
it takes 178.550000 seconds to generate the fifth Markov Chain
it takes 1000.330000 seconds to test the local acceptance

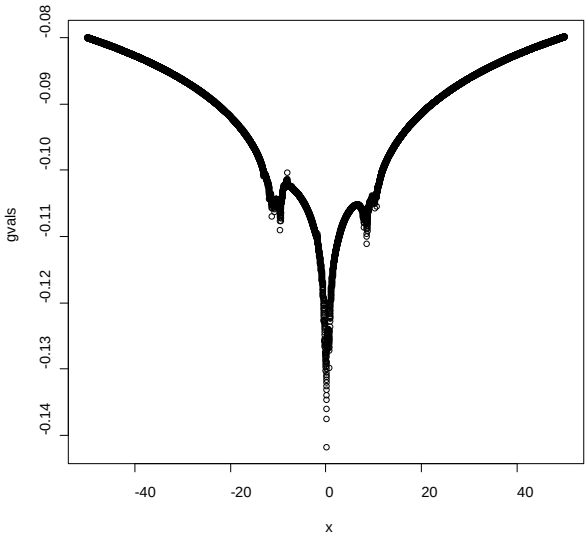
2.3.1.7 case 133 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

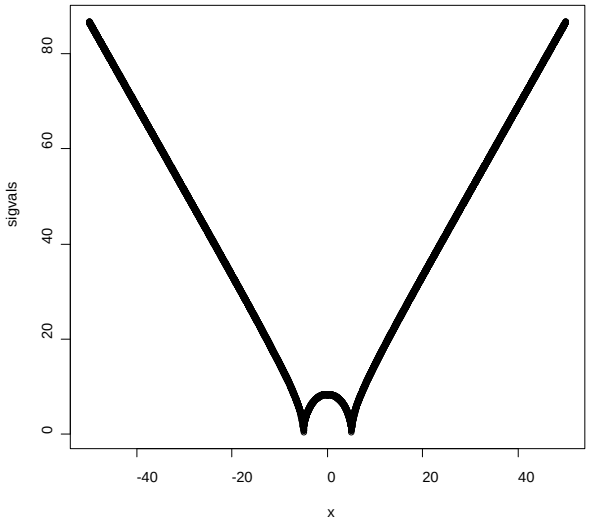




Accepted 22575 out of 100000 (22.575%).
final beta = -0.538530
alpha(0.000000) is 0.207800
alpha(2.000000) is 0.433300
alpha(5.000000) is 0.696900
alpha(8.000000) is 0.341900
alpha(10.000000) is 0.166300
alpha(13.000000) is 0.286300
alpha(15.000000) is 0.354600
alpha(20.000000) is 0.365400
alpha(30.000000) is 0.367900
alpha(50.000000) is 0.357700
it takes 447.840000 seconds to run the adaptation algorithm
it takes 200.430000 seconds to compute g and sigma
it takes 175.050000 seconds to generate the first Markov Chain
it takes 174.760000 seconds to generate the second Markov Chain
it takes 174.250000 seconds to generate the third Markov Chain
it takes 174.060000 seconds to generate the fourth Markov Chain
it takes 174.670000 seconds to generate the fifth Markov Chain
it takes 761.780000 seconds to test the local acceptance

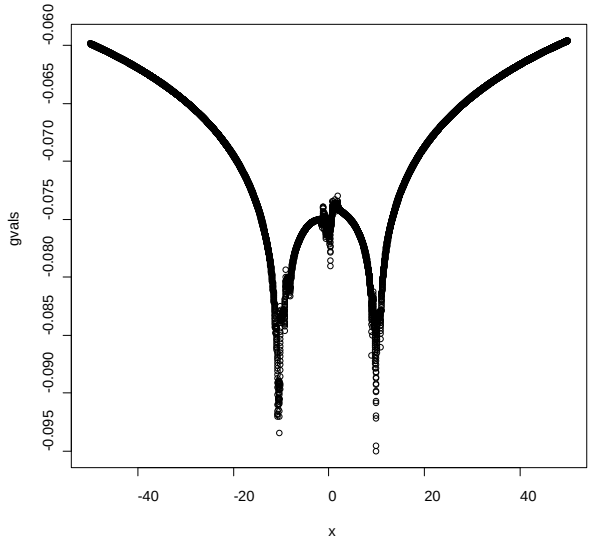
2.3.1.8 case 134 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

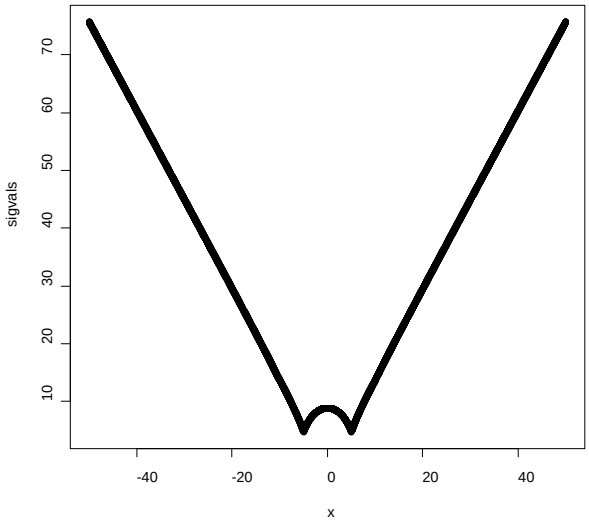




Accepted 22849 out of 100000 (22.849%).
final beta = -0.517135
alpha(0.000000) is 0.203900
alpha(2.000000) is 0.448400
alpha(5.000000) is 0.684000
alpha(8.000000) is 0.364200
alpha(10.000000) is 0.161600
alpha(13.000000) is 0.296000
alpha(15.000000) is 0.367600
alpha(20.000000) is 0.380500
alpha(30.000000) is 0.377800
alpha(50.000000) is 0.370000
it takes 649.150000 seconds to run the adaptation algorithm
it takes 283.370000 seconds to compute g and sigma
it takes 174.430000 seconds to generate the first Markov Chain
it takes 174.900000 seconds to generate the second Markov Chain
it takes 174.820000 seconds to generate the third Markov Chain
it takes 176.090000 seconds to generate the fourth Markov Chain
it takes 176.120000 seconds to generate the fifth Markov Chain
it takes 989.210000 seconds to test the local acceptance

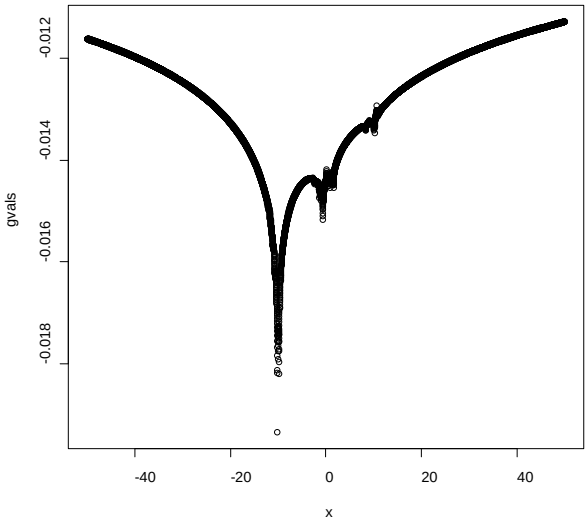
2.3.1.9 case 135 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

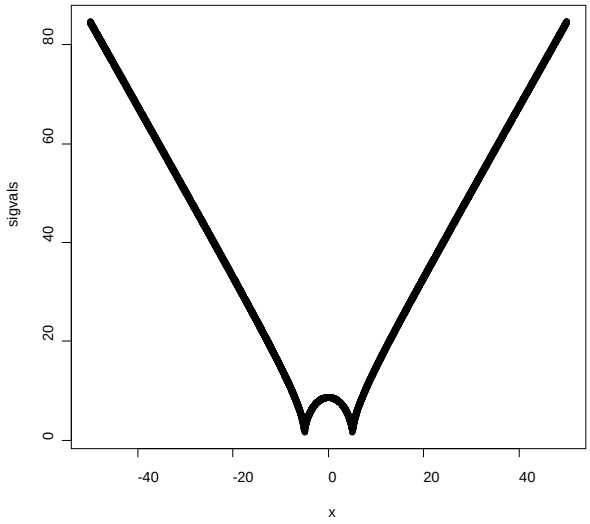




Accepted 23337 out of 100000 (23.337%).
final beta = 1.628756
alpha(0.000000) is 0.206800
alpha(2.000000) is 0.429500
alpha(5.000000) is 0.983000
alpha(8.000000) is 0.358700
alpha(10.000000) is 0.175500
alpha(13.000000) is 0.313000
alpha(15.000000) is 0.411500
alpha(20.000000) is 0.411400
alpha(30.000000) is 0.407100
alpha(50.000000) is 0.409500
it takes 232.112000 seconds to run the adaptation algorithm
it takes 102.358000 seconds to compute g and sigma
it takes 21.231000 seconds to generate the first Markov Chain
it takes 20.655000 seconds to generate the second Markov Chain
it takes 21.185000 seconds to generate the third Markov Chain
it takes 22.052000 seconds to generate the fourth Markov Chain
it takes 22.464000 seconds to generate the fifth Markov Chain
it takes 302.099000 seconds to test the local acceptance

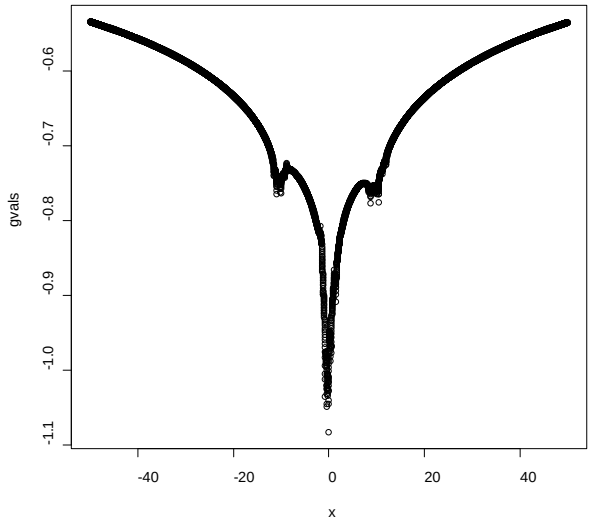
2.3.1.10 case 136 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 10$

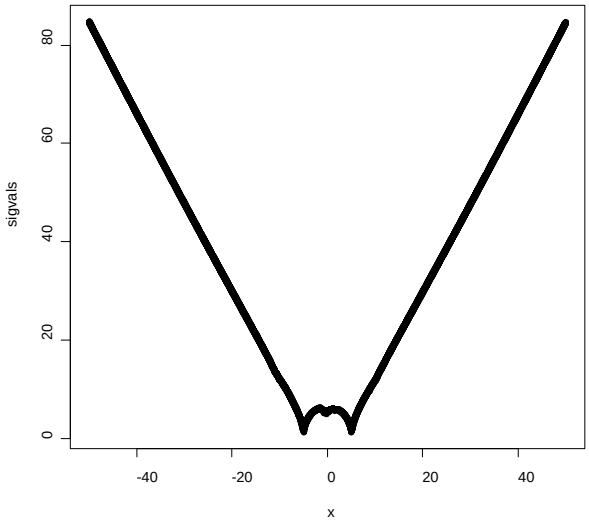




Accepted 22834 out of 100000 (22.834%).
final beta = 0.542365
alpha(0.000000) is 0.203200
alpha(2.000000) is 0.442500
alpha(5.000000) is 0.984500
alpha(8.000000) is 0.351700
alpha(10.000000) is 0.167100
alpha(13.000000) is 0.296200
alpha(15.000000) is 0.366500
alpha(20.000000) is 0.392700
alpha(30.000000) is 0.379800
alpha(50.000000) is 0.378700
it takes 640.290000 seconds to run the adaptation algorithm
it takes 283.020000 seconds to compute g and sigma
it takes 175.850000 seconds to generate the first Markov Chain
it takes 177.050000 seconds to generate the second Markov Chain
it takes 175.730000 seconds to generate the third Markov Chain
it takes 176.160000 seconds to generate the fourth Markov Chain
it takes 175.780000 seconds to generate the fifth Markov Chain
it takes 984.570000 seconds to test the local acceptance

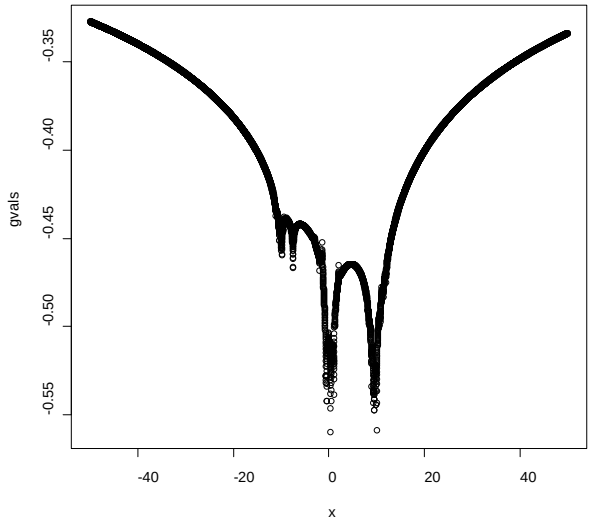
2.3.1.11 case 137 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 0.1$

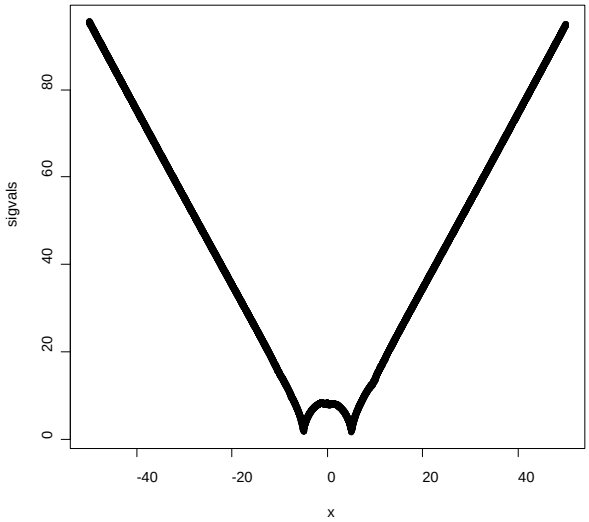




Accepted 24776 out of 100000 (24.776%).
final beta = 1.066797
alpha(0.000000) is 0.255000
alpha(2.000000) is 0.473500
alpha(5.000000) is 0.983600
alpha(8.000000) is 0.387300
alpha(10.000000) is 0.153900
alpha(13.000000) is 0.305800
alpha(15.000000) is 0.392600
alpha(20.000000) is 0.409300
alpha(30.000000) is 0.385600
alpha(50.000000) is 0.380900
it takes 235.762000 seconds to run the adaptation algorithm
it takes 101.070000 seconds to compute g and sigma
it takes 20.240000 seconds to generate the first Markov Chain
it takes 20.960000 seconds to generate the second Markov Chain
it takes 20.410000 seconds to generate the third Markov Chain
it takes 20.560000 seconds to generate the fourth Markov Chain
it takes 20.140000 seconds to generate the fifth Markov Chain
it takes 300.521000 seconds to test the local acceptance

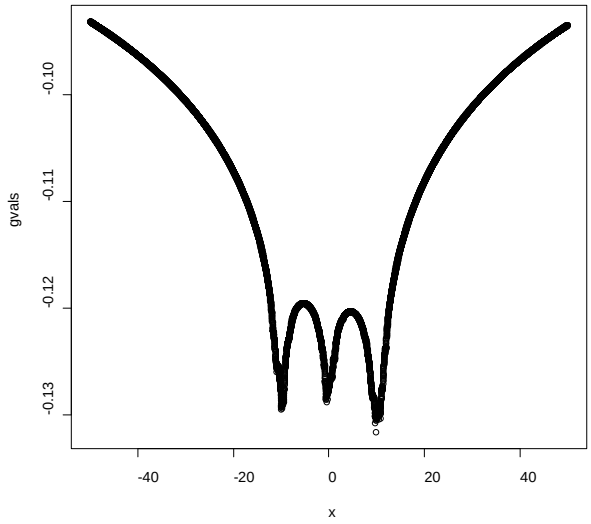
2.3.1.12 case 138 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 0.1, \gamma = 0.5$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

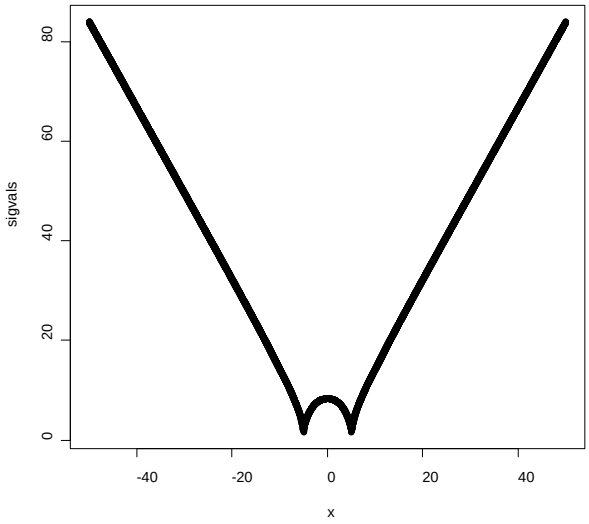




Accepted 23303 out of 100000 (23.303%).
final beta = 0.979695
alpha(0.000000) is 0.207000
alpha(2.000000) is 0.447200
alpha(5.000000) is 0.986800
alpha(8.000000) is 0.363500
alpha(10.000000) is 0.163200
alpha(13.000000) is 0.295000
alpha(15.000000) is 0.365900
alpha(20.000000) is 0.375900
alpha(30.000000) is 0.371600
alpha(50.000000) is 0.346100
it takes 240.848000 seconds to run the adaptation algorithm
it takes 109.387000 seconds to compute g and sigma
it takes 24.340000 seconds to generate the first Markov Chain
it takes 24.380000 seconds to generate the second Markov Chain
it takes 24.550000 seconds to generate the third Markov Chain
it takes 24.620000 seconds to generate the fourth Markov Chain
it takes 24.690000 seconds to generate the fifth Markov Chain
it takes 336.289000 seconds to test the local acceptance

2.3.1.13 case 139 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$



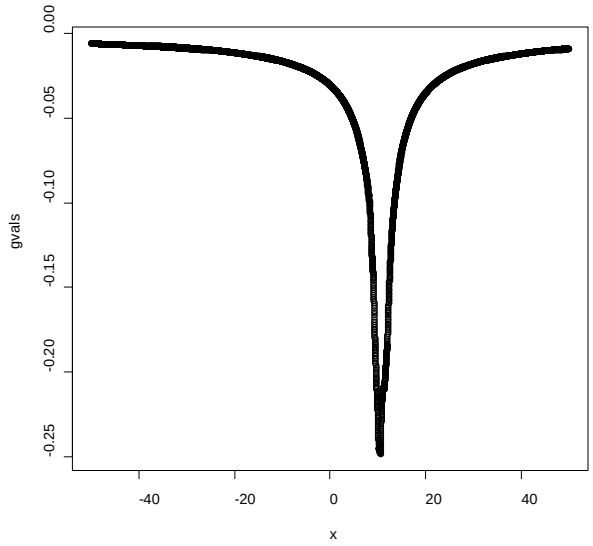


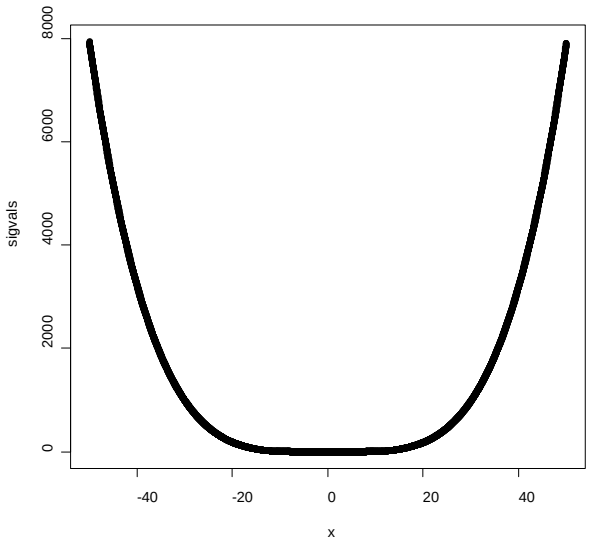
Accepted 23368 out of 100000 (23.368%).
final beta = 0.616456
alpha(0.000000) is 0.203600
alpha(2.000000) is 0.443700
alpha(5.000000) is 0.986100
alpha(8.000000) is 0.365900
alpha(10.000000) is 0.164400
alpha(13.000000) is 0.301400
alpha(15.000000) is 0.384300
alpha(20.000000) is 0.399100
alpha(30.000000) is 0.388300
alpha(50.000000) is 0.382800
it takes 232.327000 seconds to run the adaptation algorithm
it takes 100.786000 seconds to compute g and sigma
it takes 24.283000 seconds to generate the first Markov Chain
it takes 21.371000 seconds to generate the second Markov Chain
it takes 21.644000 seconds to generate the third Markov Chain
it takes 22.590000 seconds to generate the fourth Markov Chain
it takes 21.119000 seconds to generate the fifth Markov Chain
it takes 290.380000 seconds to test the local acceptance

[1] 13.34282
[1] 0.09514207
[1] 17.27245

2.3.2 kernel function $K(x) = \frac{1}{1+\alpha_1 |x|^{\alpha_2}}$

2.3.2.1 case 140 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$



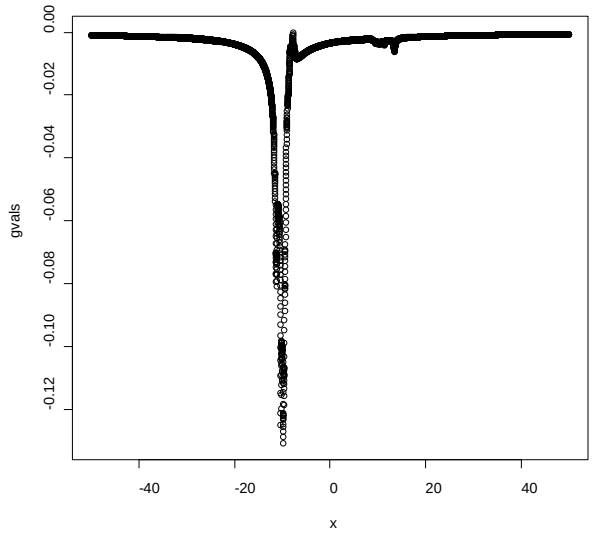


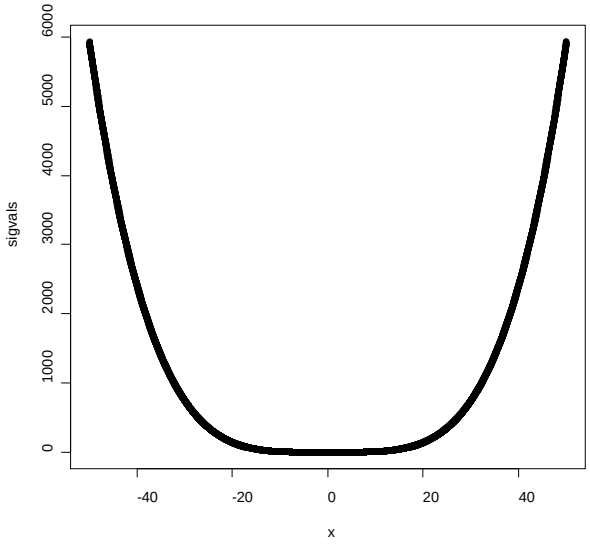
Accepted 74753 out of 100000 (74.753%).
final beta = -6.643285
alpha(0.000000) is 0.751500
alpha(2.000000) is 0.723000
alpha(5.000000) is 0.956800
alpha(8.000000) is 0.597100
alpha(10.000000) is 0.175300
alpha(13.000000) is 0.129200
alpha(15.000000) is 0.103600
alpha(20.000000) is 0.056700
alpha(30.000000) is 0.016700
alpha(50.000000) is 0.003300
it takes 357.200000 seconds to run the adaptation algorithm
it takes 156.500000 seconds to compute g and sigma
it takes 168.160000 seconds to generate the first Markov Chain
it takes 168.450000 seconds to generate the second Markov Chain
it takes 166.270000 seconds to generate the third Markov Chain
it takes 167.860000 seconds to generate the fourth Markov Chain
it takes 166.480000 seconds to generate the fifth Markov Chain
it takes 893.900000 seconds to test the local acceptance

[1] 13.66704
[1] 0.01207831
[1] 0.3014526

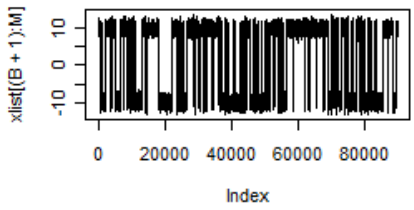
2.3.2.2 case 141 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

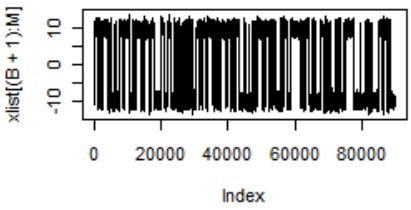
$b_n = 1$

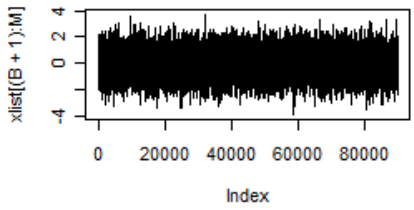


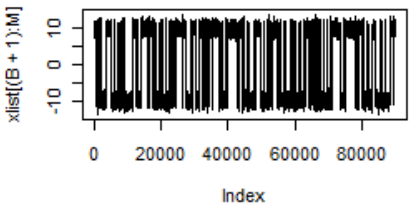


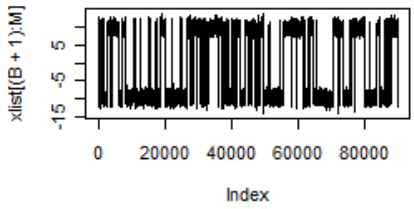
Accepted 24059 out of 100000 (24.059%).
final beta = -6.939904
alpha(0.000000) is 0.813200
alpha(2.000000) is 0.753900
alpha(5.000000) is 0.954500
alpha(8.000000) is 0.617100
alpha(10.000000) is 0.191500
alpha(13.000000) is 0.134500
alpha(15.000000) is 0.126900
alpha(20.000000) is 0.065300
alpha(30.000000) is 0.023900
alpha(50.000000) is 0.005600
it takes 357.490000 seconds to run the adaptation algorithm
it takes 156.390000 seconds to compute g and sigma
it takes 171.120000 seconds to generate the first Markov Chain
it takes 166.660000 seconds to generate the second Markov Chain
it takes 166.270000 seconds to generate the third Markov Chain
it takes 167.590000 seconds to generate the fourth Markov Chain
it takes 164.220000 seconds to generate the fifth Markov Chain
it takes 862.580000 seconds to test the local acceptance









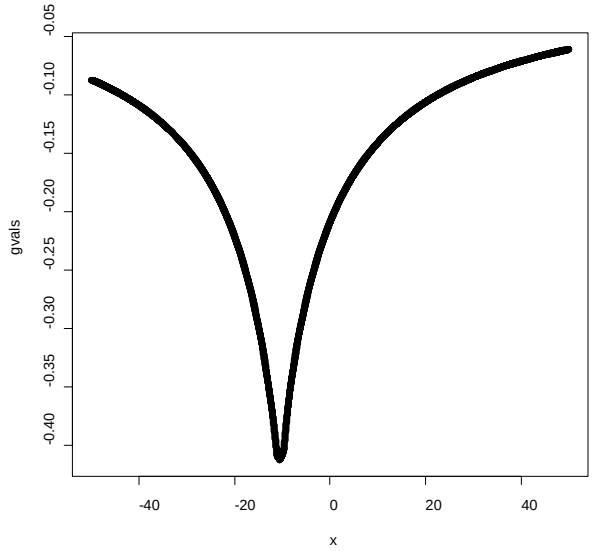


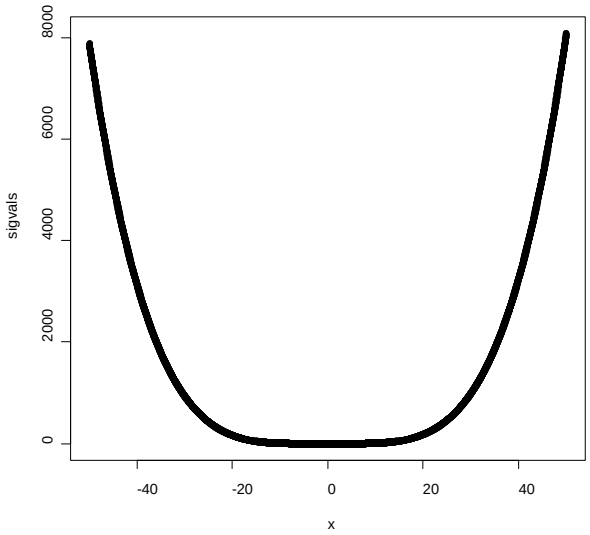

```
>    source("xlist1");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 90.40186
[1] 0.3162226
[1] 1.229918
>    source("xlist2");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 89.71504
[1] 0.3175898
[1] 1.441011
>    source("xlist3");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 19.61744
[1] 0.01470637
[1] 0.2134324
>    source("xlist4");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
sqrt(M-B) * sqrt( varfact(xlist[(B+1):M]) );asjd(xlist[(B+1):M]);
[1] 90.32034
[1] 0.3185251
[1] 1.313225
>    source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M])    /
```

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 91.49129
[1] 0.3193592
[1] 1.163729

2.3.2.3 case 142 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth

$b_n = 1$



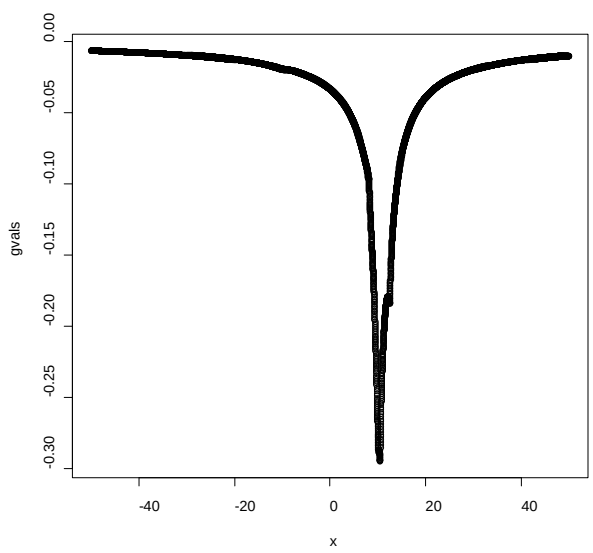


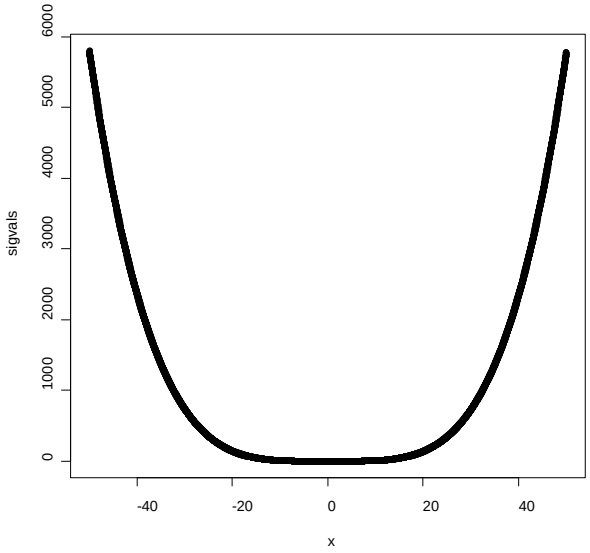
Accepted 21928 out of 100000 (21.928%).
final beta = -6.569201
alpha(0.000000) is 0.776600
alpha(2.000000) is 0.722100
alpha(5.000000) is 0.956800
alpha(8.000000) is 0.590700
alpha(10.000000) is 0.151900
alpha(13.000000) is 0.113300
alpha(15.000000) is 0.104100
alpha(20.000000) is 0.050200
alpha(30.000000) is 0.018000
alpha(50.000000) is 0.003600
it takes 350.290000 seconds to run the adaptation algorithm
it takes 154.400000 seconds to compute g and sigma
it takes 167.000000 seconds to generate the first Markov Chain
it takes 165.600000 seconds to generate the second Markov Chain
it takes 163.820000 seconds to generate the third Markov Chain
it takes 166.220000 seconds to generate the fourth Markov Chain
it takes 169.510000 seconds to generate the fifth Markov Chain
it takes 886.600000 seconds to test the local acceptance

[1] 88.28712
[1] 0.3145201
[1] 1.458424

2.3.2.4 case 143 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth

$b_n = 1$



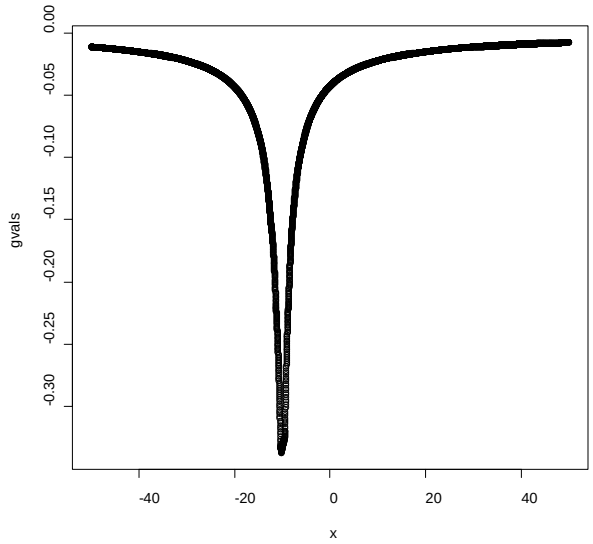


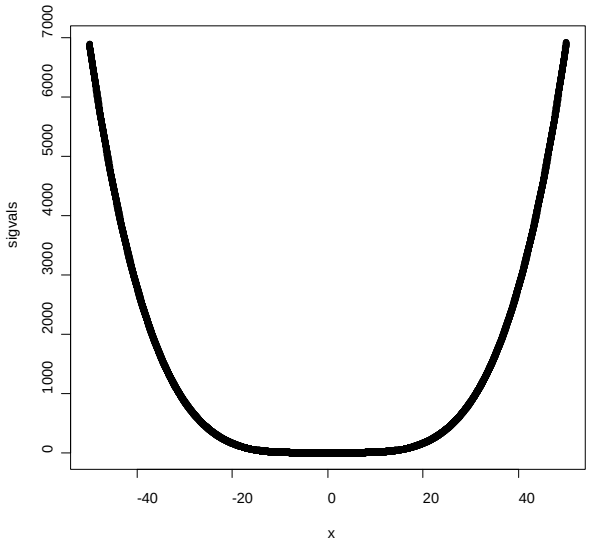
Accepted 21960 out of 100000 (21.960%).
final beta = -2.358082
alpha(0.000000) is 0.653400
alpha(2.000000) is 0.683000
alpha(5.000000) is 0.981500
alpha(8.000000) is 0.573800
alpha(10.000000) is 0.187400
alpha(13.000000) is 0.148500
alpha(15.000000) is 0.124500
alpha(20.000000) is 0.071500
alpha(30.000000) is 0.023900
alpha(50.000000) is 0.006600
it takes 350.510000 seconds to run the adaptation algorithm
it takes 154.900000 seconds to compute g and sigma
it takes 164.410000 seconds to generate the first Markov Chain
it takes 166.610000 seconds to generate the second Markov Chain
it takes 167.720000 seconds to generate the third Markov Chain
it takes 166.330000 seconds to generate the fourth Markov Chain
it takes 168.090000 seconds to generate the fifth Markov Chain
it takes 851.800000 seconds to test the local acceptance

[1] 90.71062
[1] 0.3180162
[1] 1.260599

2.3.2.5 case 144 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 10, \gamma = 2$ with fixed bandwidth

$b_n = 1$

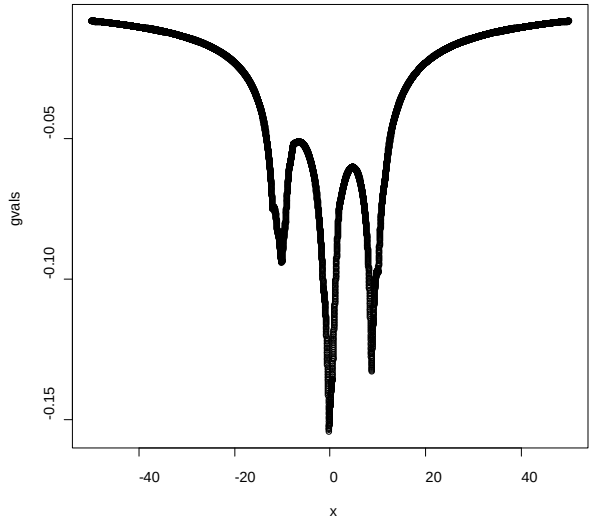


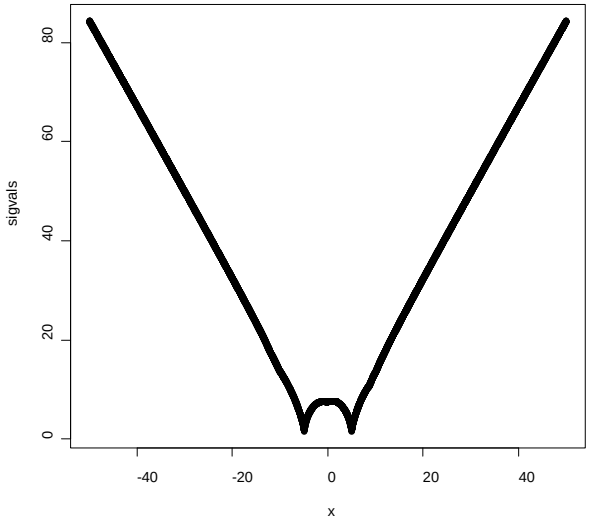


Accepted 23476 out of 100000 (23.476%).
final beta = -11.382114
alpha(0.000000) is 0.806700
alpha(2.000000) is 0.745700
alpha(5.000000) is 0.713100
alpha(8.000000) is 0.611800
alpha(10.000000) is 0.161700
alpha(13.000000) is 0.118300
alpha(15.000000) is 0.104900
alpha(20.000000) is 0.057000
alpha(30.000000) is 0.021400
alpha(50.000000) is 0.004700
it takes 356.180000 seconds to run the adaptation algorithm
it takes 156.340000 seconds to compute g and sigma
it takes 163.020000 seconds to generate the first Markov Chain
it takes 164.080000 seconds to generate the second Markov Chain
it takes 161.600000 seconds to generate the third Markov Chain
it takes 164.910000 seconds to generate the fourth Markov Chain
it takes 168.950000 seconds to generate the fifth Markov Chain
it takes 884.960000 seconds to test the local acceptance

> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 89.95243
[1] 0.3179866
[1] 1.358456

2.3.2.6 case 145 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$



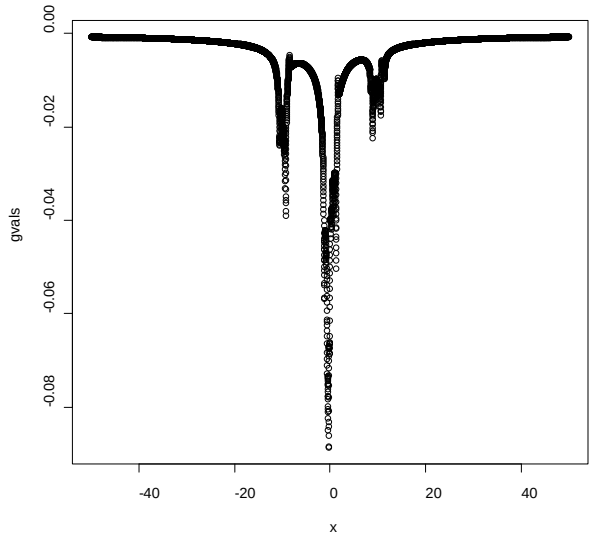


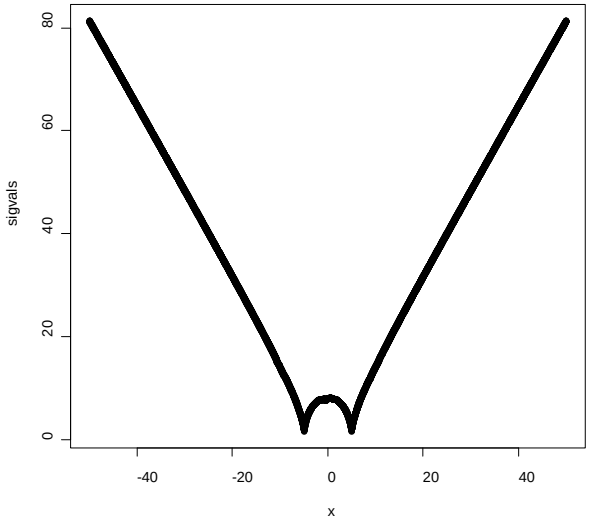
Accepted 23749 out of 100000 (23.749%).
final beta = 0.535915
alpha(0.000000) is 0.215100
alpha(2.000000) is 0.457000
alpha(5.000000) is 0.980300
alpha(8.000000) is 0.371700
alpha(10.000000) is 0.161000
alpha(13.000000) is 0.310200
alpha(15.000000) is 0.366000
alpha(20.000000) is 0.391400
alpha(30.000000) is 0.383000
alpha(50.000000) is 0.372500
it takes 356.300000 seconds to run the adaptation algorithm
it takes 156.880000 seconds to compute g and sigma
it takes 170.430000 seconds to generate the first Markov Chain
it takes 169.350000 seconds to generate the second Markov Chain
it takes 168.660000 seconds to generate the third Markov Chain
it takes 169.640000 seconds to generate the fourth Markov Chain
it takes 168.940000 seconds to generate the fifth Markov Chain

it takes 624.960000 seconds to test the local acceptance

> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 14.27748
[1] 0.09757993
[1] 16.54765

2.3.2.7 case 146 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$



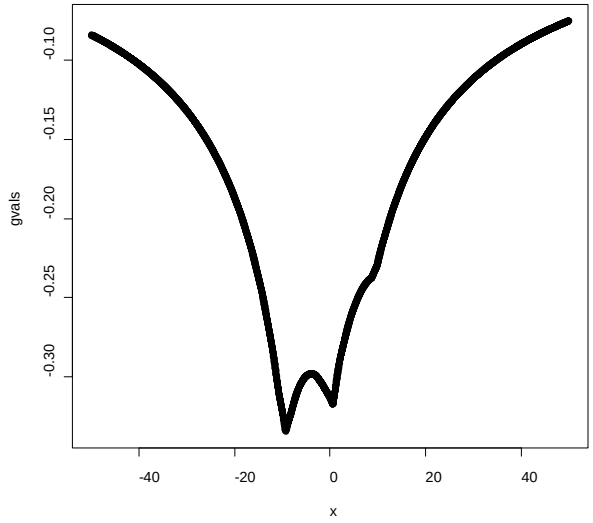


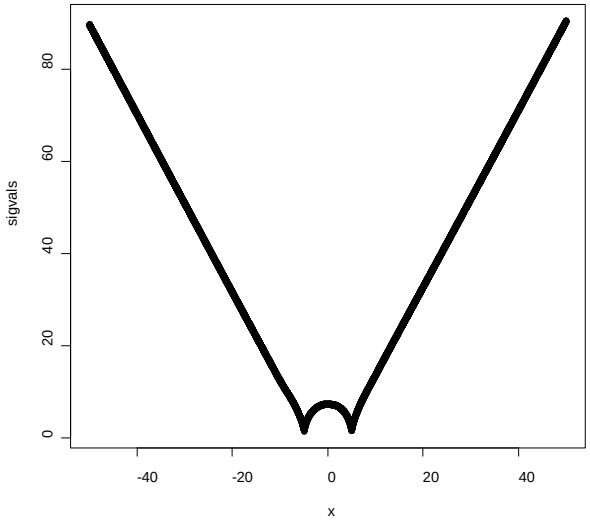
Accepted 23602 out of 100000 (23.602%).
final beta = 0.491872
alpha(0.000000) is 0.205200
alpha(2.000000) is 0.461500
alpha(5.000000) is 0.981800
alpha(8.000000) is 0.360800
alpha(10.000000) is 0.163000
alpha(13.000000) is 0.301700
alpha(15.000000) is 0.375700
alpha(20.000000) is 0.390900
alpha(30.000000) is 0.383200
alpha(50.000000) is 0.380400
it takes 356.260000 seconds to run the adaptation algorithm
it takes 157.310000 seconds to compute g and sigma
it takes 170.510000 seconds to generate the first Markov Chain
it takes 169.670000 seconds to generate the second Markov Chain
it takes 169.950000 seconds to generate the third Markov Chain
it takes 170.300000 seconds to generate the fourth Markov Chain
it takes 169.670000 seconds to generate the fifth Markov Chain

it takes 619.850000 seconds to test the local acceptance

> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 12.43203
[1] 0.09186996
[1] 17.23872

2.3.2.8 case 147 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.1, \alpha_2 = 1, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$



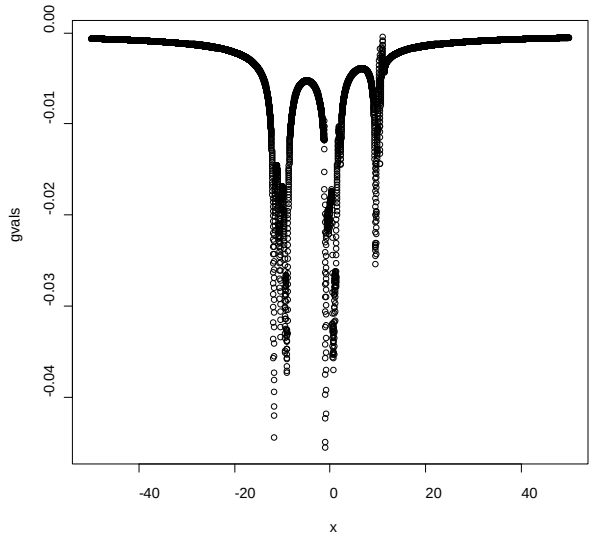


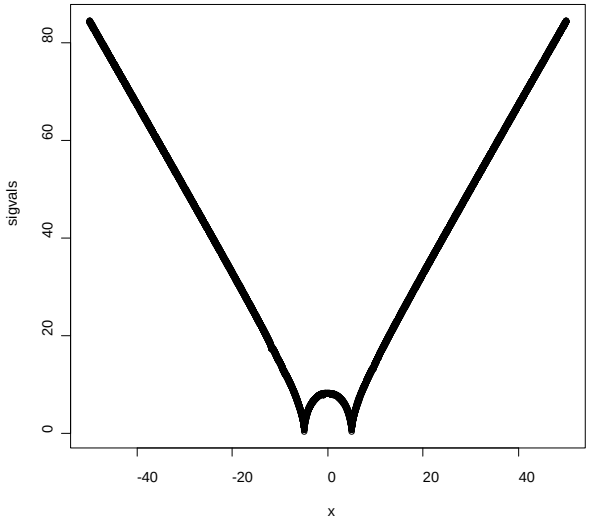
Accepted 24104 out of 100000 (24.104%).
final beta = 0.672420
alpha(0.000000) is 0.219700
alpha(2.000000) is 0.474000
alpha(5.000000) is 0.985900
alpha(8.000000) is 0.374900
alpha(10.000000) is 0.157300
alpha(13.000000) is 0.307600
alpha(15.000000) is 0.369800
alpha(20.000000) is 0.379100
alpha(30.000000) is 0.373500
alpha(50.000000) is 0.361500
it takes 187.283000 seconds to run the adaptation algorithm
it takes 82.121000 seconds to compute g and sigma
it takes 20.090000 seconds to generate the first Markov Chain
it takes 20.182000 seconds to generate the second Markov Chain
it takes 20.524000 seconds to generate the third Markov Chain
it takes 20.467000 seconds to generate the fourth Markov Chain
it takes 20.692000 seconds to generate the fifth Markov Chain

it takes 249.071000 seconds to test the local acceptance

> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 14.78932
[1] 0.09995486
[1] 16.03785

2.3.2.9 case 148 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 1$





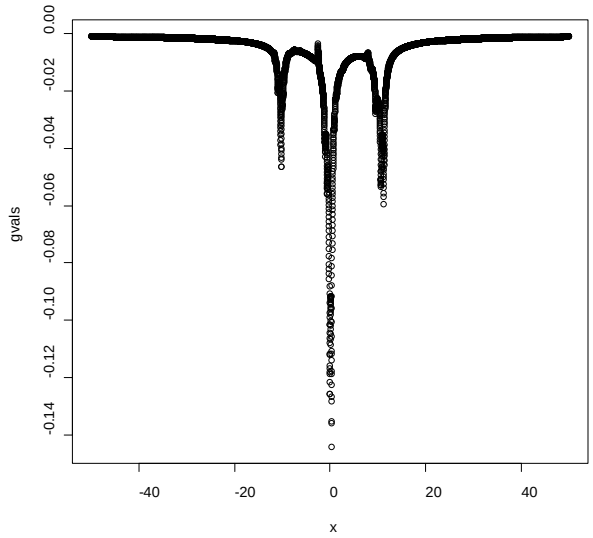
Accepted 22987 out of 100000 (22.987%).
final beta = -0.620667
alpha(0.000000) is 0.209300
alpha(2.000000) is 0.449600
alpha(5.000000) is 0.679100
alpha(8.000000) is 0.361700
alpha(10.000000) is 0.167800
alpha(13.000000) is 0.298600
alpha(15.000000) is 0.364600
alpha(20.000000) is 0.388100
alpha(30.000000) is 0.380300
alpha(50.000000) is 0.387100
it takes 357.170000 seconds to run the adaptation algorithm
it takes 156.510000 seconds to compute g and sigma
it takes 170.650000 seconds to generate the first Markov Chain
it takes 170.480000 seconds to generate the second Markov Chain
it takes 170.280000 seconds to generate the third Markov Chain
it takes 171.030000 seconds to generate the fourth Markov Chain
it takes 170.120000 seconds to generate the fifth Markov Chain

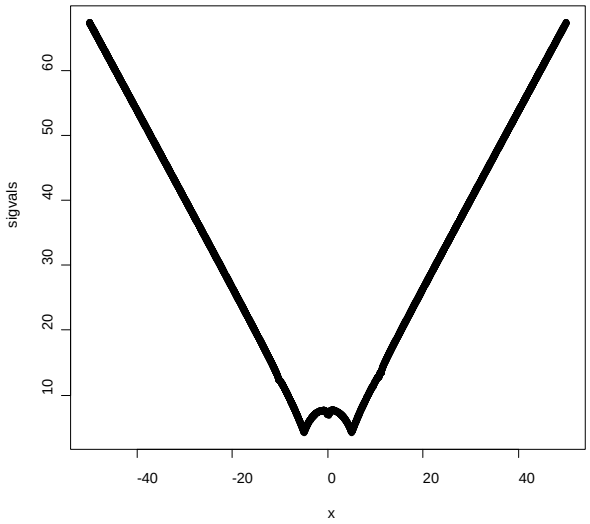
it takes 621.790000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 13.72235
[1] 0.09650097
[1] 17.49019

2.3.2.10 case 149 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 0.1, \gamma = 0.5$ with fixed bandwidth

$b_n = 1$

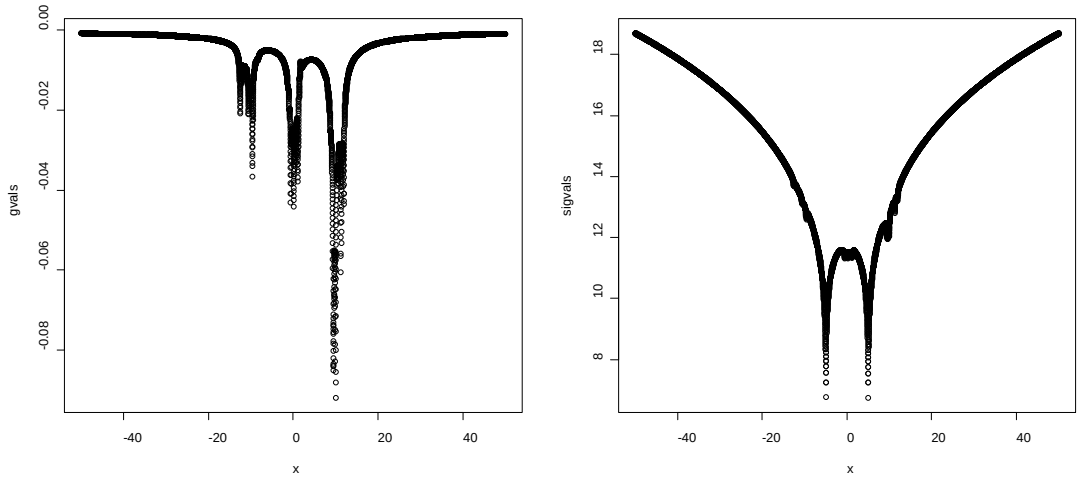




Accepted 24399 out of 100000 (24.399%).
final beta = 1.453206
alpha(0.000000) is 0.224300
alpha(2.000000) is 0.455900
alpha(5.000000) is 0.993000
alpha(8.000000) is 0.378800
alpha(10.000000) is 0.174700
alpha(13.000000) is 0.326800
alpha(15.000000) is 0.426700
alpha(20.000000) is 0.438900
alpha(30.000000) is 0.429700
alpha(50.000000) is 0.440100
it takes 352.330000 seconds to run the adaptation algorithm
it takes 154.950000 seconds to compute g and sigma
it takes 167.720000 seconds to generate the first Markov Chain
it takes 167.820000 seconds to generate the second Markov Chain
it takes 167.570000 seconds to generate the third Markov Chain
it takes 166.890000 seconds to generate the fourth Markov Chain
it takes 166.760000 seconds to generate the fifth Markov Chain
it takes 580.080000 seconds to test the local acceptance

[1] 14.91846
[1] 0.1007374
[1] 15.86291

2.3.2.11 case 150 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.1$ with fixed bandwidth $b_n = 1$

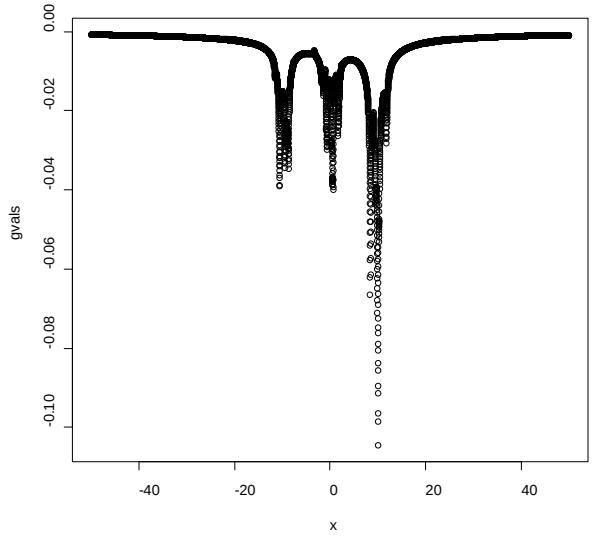


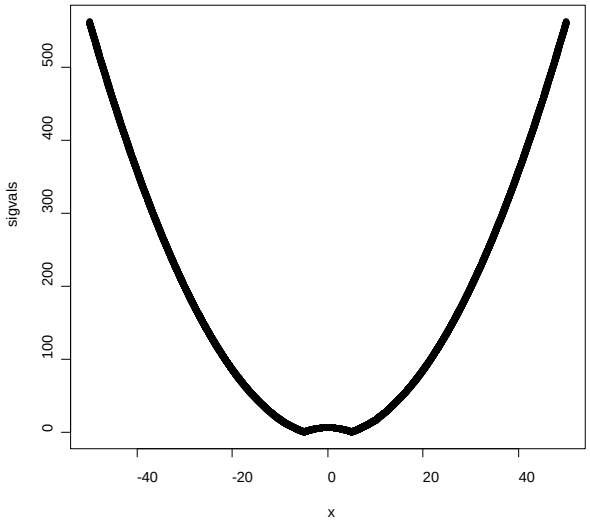
Accepted 22156 out of 100000 (22.156%).
final beta = 1.917426
alpha(0.000000) is 0.170300
alpha(2.000000) is 0.378600
alpha(5.000000) is 0.930700
alpha(8.000000) is 0.344400
alpha(10.000000) is 0.172300
alpha(13.000000) is 0.358100
alpha(15.000000) is 0.487300
alpha(20.000000) is 0.500900
alpha(30.000000) is 0.501700
alpha(50.000000) is 0.493700
it takes 353.150000 seconds to run the adaptation algorithm
it takes 155.440000 seconds to compute g and sigma
it takes 167.430000 seconds to generate the first Markov Chain
it takes 167.340000 seconds to generate the second Markov Chain
it takes 168.300000 seconds to generate the third Markov Chain
it takes 168.280000 seconds to generate the fourth Markov Chain
it takes 168.090000 seconds to generate the fifth Markov Chain
it takes 525.460000 seconds to test the local acceptance

[1] 13.31069
[1] 0.09509448
[1] 17.21339

2.3.2.12 case 151 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 1$ with fixed bandwidth

$b_n = 1$



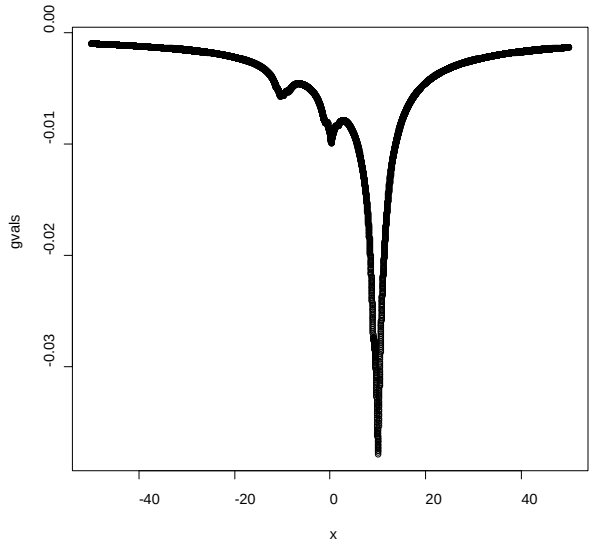


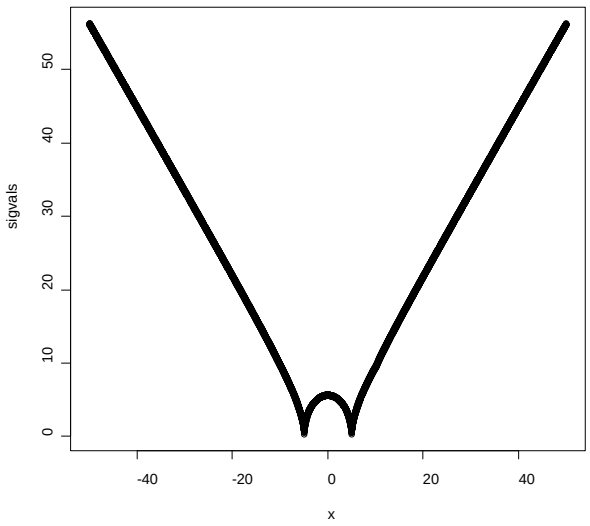
Accepted 23422 out of 100000 (23.422%).
final beta = -3.783332
alpha(0.000000) is 0.253600
alpha(2.000000) is 0.486500
alpha(5.000000) is 0.645400
alpha(8.000000) is 0.368100
alpha(10.000000) is 0.137900
alpha(13.000000) is 0.200000
alpha(15.000000) is 0.203000
alpha(20.000000) is 0.171300
alpha(30.000000) is 0.113100
alpha(50.000000) is 0.067000
it takes 358.220000 seconds to run the adaptation algorithm
it takes 157.160000 seconds to compute g and sigma
it takes 182.650000 seconds to generate the first Markov Chain
it takes 184.390000 seconds to generate the second Markov Chain
it takes 182.440000 seconds to generate the third Markov Chain
it takes 182.340000 seconds to generate the fourth Markov Chain
it takes 181.730000 seconds to generate the fifth Markov Chain
it takes 837.150000 seconds to test the local acceptance

[1] 16.61734
[1] 0.1063952
[1] 14.43249

2.3.2.13 case 152 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth

$b_n = 10$



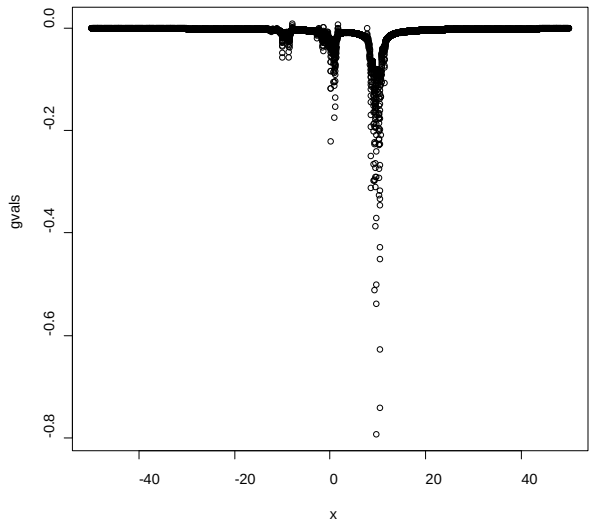


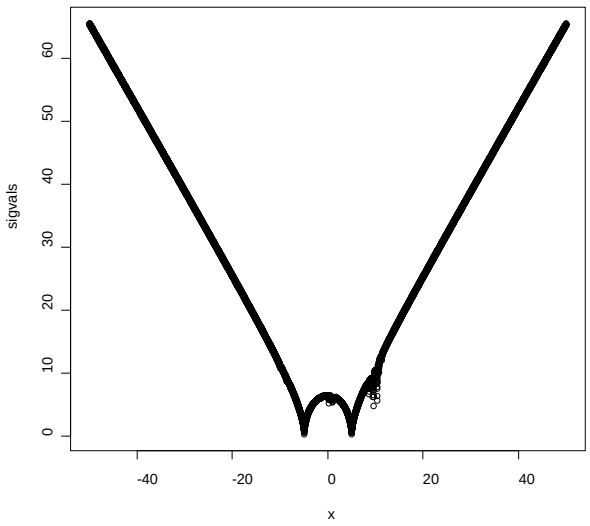
Accepted 25983 out of 100000 (25.983%).
final beta = -1.028352
alpha(0.000000) is 0.259200
alpha(2.000000) is 0.476600
alpha(5.000000) is 0.779900
alpha(8.000000) is 0.416200
alpha(10.000000) is 0.164800
alpha(13.000000) is 0.326700
alpha(15.000000) is 0.421800
alpha(20.000000) is 0.451900
alpha(30.000000) is 0.460200
alpha(50.000000) is 0.465600
it takes 193.936000 seconds to run the adaptation algorithm
it takes 85.283000 seconds to compute g and sigma
it takes 19.154000 seconds to generate the first Markov Chain
it takes 18.866000 seconds to generate the second Markov Chain
it takes 20.767000 seconds to generate the third Markov Chain
it takes 20.124000 seconds to generate the fourth Markov Chain
it takes 21.762000 seconds to generate the fifth Markov Chain
it takes 222.388000 seconds to test the local acceptance

[1] 21.61043
[1] 0.1214091
[1] 10.9672

2.3.2.14 case 153 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth

$b_n = 0.1$



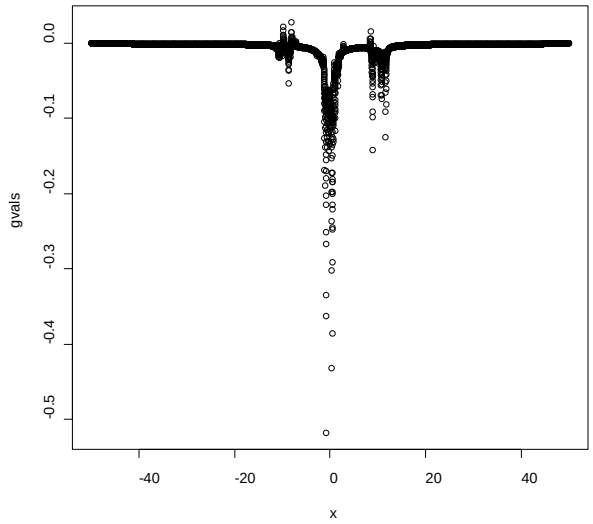


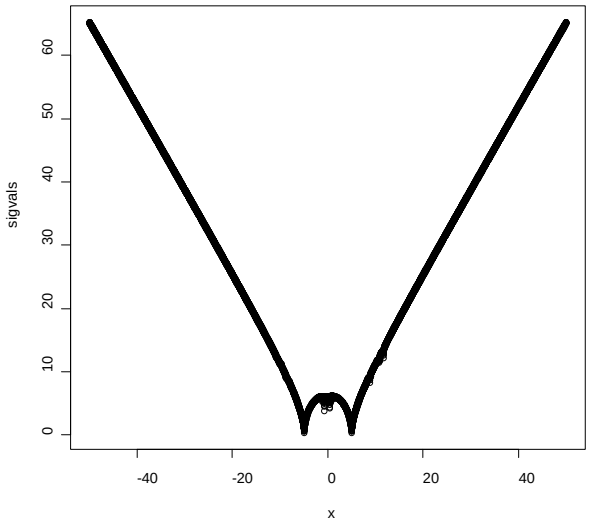
Accepted 25310 out of 100000 (25.310%).
final beta = -0.875086
alpha(0.000000) is 0.237000
alpha(2.000000) is 0.485000
alpha(5.000000) is 0.802500
alpha(8.000000) is 0.398100
alpha(10.000000) is 0.171400
alpha(13.000000) is 0.324000
alpha(15.000000) is 0.399000
alpha(20.000000) is 0.435900
alpha(30.000000) is 0.436800
alpha(50.000000) is 0.447100
it takes 189.983000 seconds to run the adaptation algorithm
it takes 85.461000 seconds to compute g and sigma
it takes 24.926000 seconds to generate the first Markov Chain
it takes 23.326000 seconds to generate the second Markov Chain
it takes 23.988000 seconds to generate the third Markov Chain
it takes 23.716000 seconds to generate the fourth Markov Chain
it takes 21.556000 seconds to generate the fifth Markov Chain
it takes 229.869000 seconds to test the local acceptance

[1] 17.48015
[1] 0.108967
[1] 13.33216

2.3.2.15 case 154 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ withdecreasing

bandwidth $b_n = \frac{1}{n^{0.2}}$



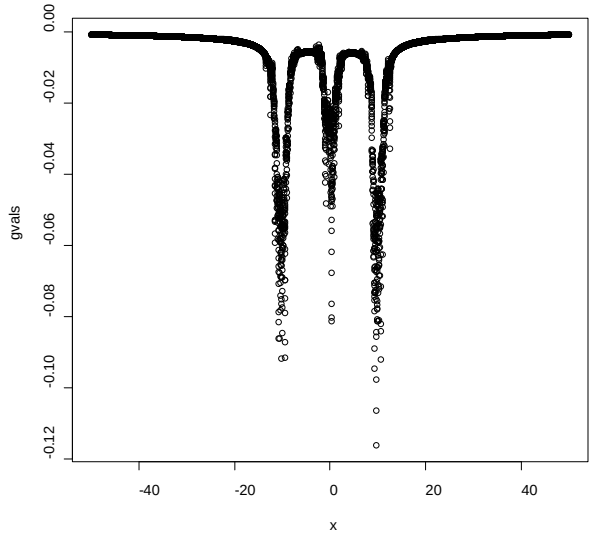


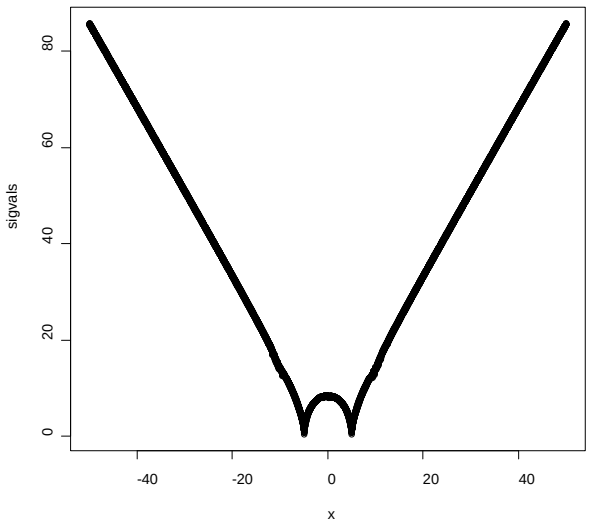
Accepted 25287 out of 100000 (25.287%).
final beta = -0.880212
alpha(0.000000) is 0.254100
alpha(2.000000) is 0.477500
alpha(5.000000) is 0.794300
alpha(8.000000) is 0.402300
alpha(10.000000) is 0.168100
alpha(13.000000) is 0.317600
alpha(15.000000) is 0.402900
alpha(20.000000) is 0.435400
alpha(30.000000) is 0.436600
alpha(50.000000) is 0.436100
it takes 188.427000 seconds to run the adaptation algorithm
it takes 82.681000 seconds to compute g and sigma
it takes 20.371000 seconds to generate the first Markov Chain
it takes 20.669000 seconds to generate the second Markov Chain
it takes 20.434000 seconds to generate the third Markov Chain
it takes 20.132000 seconds to generate the fourth Markov Chain
it takes 19.589000 seconds to generate the fifth Markov Chain
it takes 236.470000 seconds to test the local acceptance

[1] 17.77595
[1] 0.1097439
[1] 13.55263

2.3.2.16 case 155 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ fixed bandwidth

$b_n = 1$





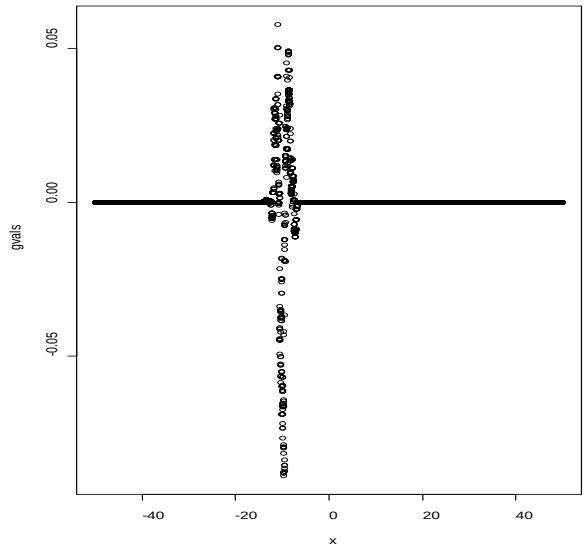
Accepted 23144 out of 100000 (23.144%).
final beta = -0.606829
alpha(0.000000) is 0.202900
alpha(2.000000) is 0.448100
alpha(5.000000) is 0.840800
alpha(8.000000) is 0.360600
alpha(10.000000) is 0.159400
alpha(13.000000) is 0.293900
alpha(15.000000) is 0.365500
alpha(20.000000) is 0.387600
alpha(30.000000) is 0.374600
alpha(50.000000) is 0.378500
it takes 187.910000 seconds to run the adaptation algorithm
it takes 83.228000 seconds to compute g and sigma
it takes 23.281000 seconds to generate the first Markov Chain
it takes 24.219000 seconds to generate the second Markov Chain
it takes 22.092000 seconds to generate the third Markov Chain
it takes 22.606000 seconds to generate the fourth Markov Chain
it takes 23.064000 seconds to generate the fifth Markov Chain
it takes 261.017000 seconds to test the local acceptance

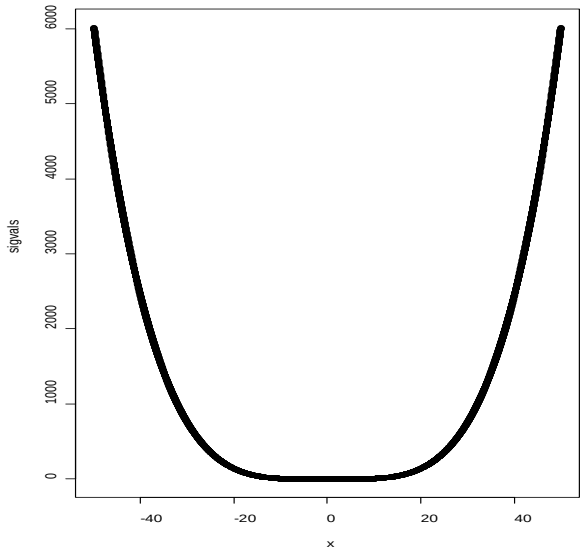
[1] 12.96691
[1] 0.09388677
[1] 17.26632

2.3.3 kernel function $K(x)=\begin{cases}0,\mid x\mid\geq 2*width\\ -1*hegith, width<\mid x\mid<2*width\\ 1*height,\mid x\mid\leq width\end{cases}$

2.3.3.1 case 149 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=10,\alpha_2=1,C=0.1,\gamma=0.5$ with fixed bandwidth

$b_n=1$

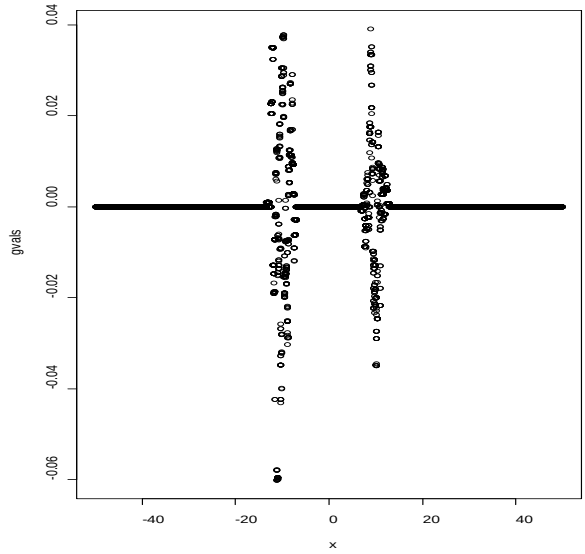


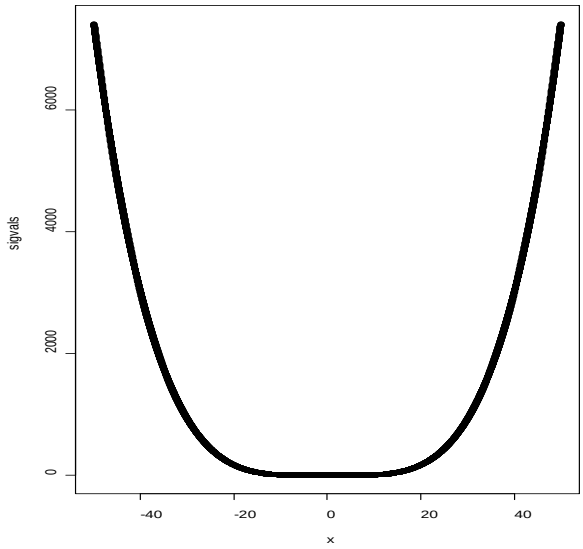


Accepted 23385 out of 100000 (23.385%).
final beta = -6.925807
alpha(0.000000) is 0.811400
alpha(2.000000) is 0.742800
alpha(5.000000) is 0.952600
alpha(8.000000) is 0.614000
alpha(10.000000) is 0.182200
alpha(13.000000) is 0.132100
alpha(15.000000) is 0.114700
alpha(20.000000) is 0.063800
alpha(30.000000) is 0.025300
alpha(50.000000) is 0.005000
it takes 346.490000 seconds to run the adaptation algorithm
it takes 152.520000 seconds to compute g and sigma
it takes 166.510000 seconds to generate the first Markov Chain
it takes 172.820000 seconds to generate the second Markov Chain
it takes 163.730000 seconds to generate the third Markov Chain
it takes 164.330000 seconds to generate the fourth Markov Chain
it takes 169.000000 seconds to generate the fifth Markov Chain
it takes 840.990000 seconds to test the local acceptance

[1] 90.39445
[1] 0.3176161
[1] 1.277381

2.3.3.2 case 157 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.5, \text{width} = 0.5, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$

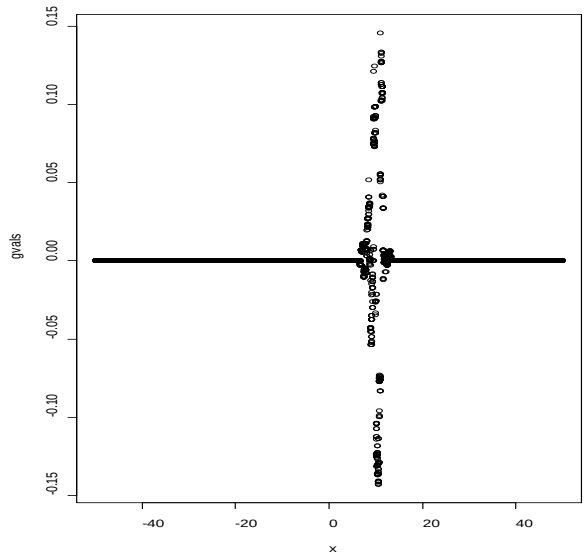


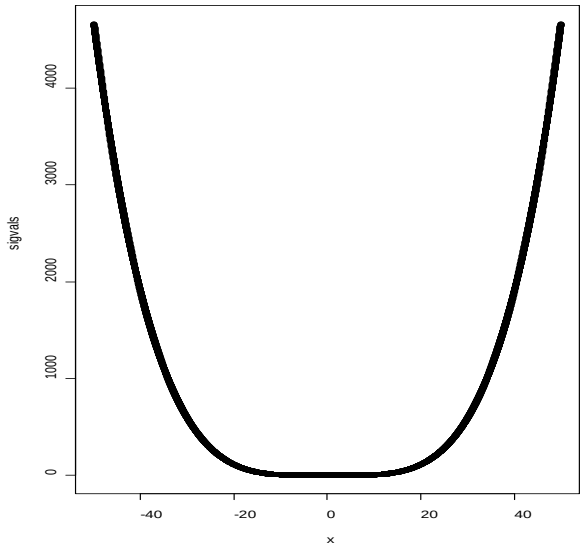


Accepted 20323 out of 100000 (20.323%).
final beta = -11.322095
alpha(0.000000) is 0.782200
alpha(2.000000) is 0.738500
alpha(5.000000) is 0.712800
alpha(8.000000) is 0.590300
alpha(10.000000) is 0.155300
alpha(13.000000) is 0.124000
alpha(15.000000) is 0.107600
alpha(20.000000) is 0.057300
alpha(30.000000) is 0.020600
alpha(50.000000) is 0.004700
it takes 346.640000 seconds to run the adaptation algorithm
it takes 152.770000 seconds to compute g and sigma
it takes 167.790000 seconds to generate the first Markov Chain
it takes 166.350000 seconds to generate the second Markov Chain
it takes 166.360000 seconds to generate the third Markov Chain
it takes 167.700000 seconds to generate the fourth Markov Chain
it takes 168.230000 seconds to generate the fifth Markov Chain
it takes 865.340000 seconds to test the local acceptance

[1] 82.33719
[1] 0.3038594
[1] 2.291764

2.3.3.3 case 158 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.5, \text{width} = 0.5, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

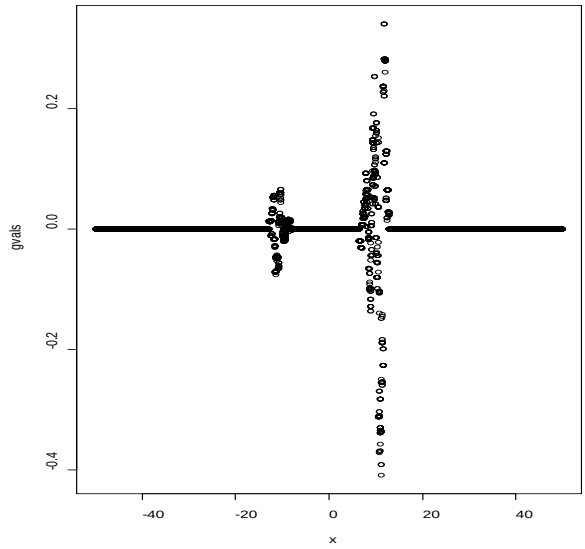


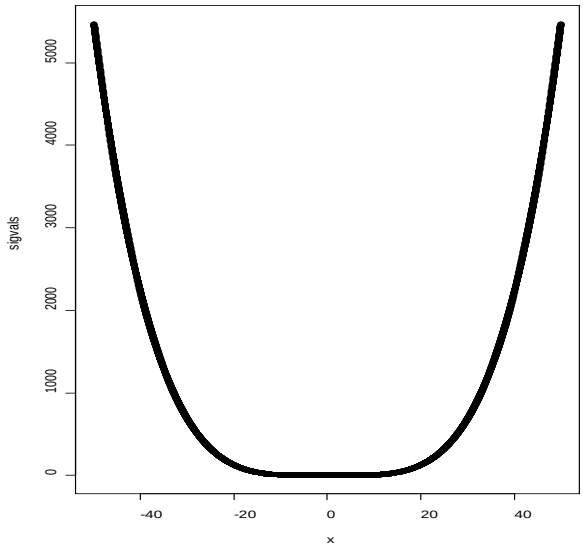


Accepted 23903 out of 100000 (23.903%).
final beta = -2.583150
alpha(0.000000) is 0.722500
alpha(2.000000) is 0.700400
alpha(5.000000) is 0.981800
alpha(8.000000) is 0.596600
alpha(10.000000) is 0.195900
alpha(13.000000) is 0.150100
alpha(15.000000) is 0.134500
alpha(20.000000) is 0.085400
alpha(30.000000) is 0.029800
alpha(50.000000) is 0.006800
it takes 347.020000 seconds to run the adaptation algorithm
it takes 152.780000 seconds to compute g and sigma
it takes 168.540000 seconds to generate the first Markov Chain
it takes 167.300000 seconds to generate the second Markov Chain
it takes 166.200000 seconds to generate the third Markov Chain
it takes 171.660000 seconds to generate the fourth Markov Chain
it takes 166.300000 seconds to generate the fifth Markov Chain
it takes 817.780000 seconds to test the local acceptance

[1] 91.6572
[1] 0.3206371
[1] 1.161233

2.3.3.4 case 159 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 2, \text{width} = 0.5, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

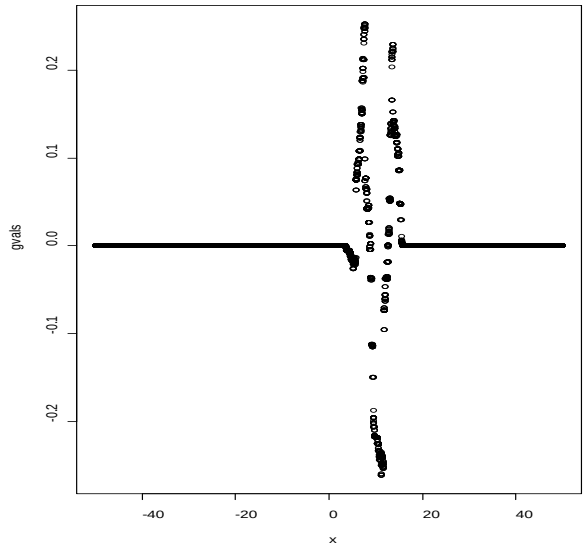


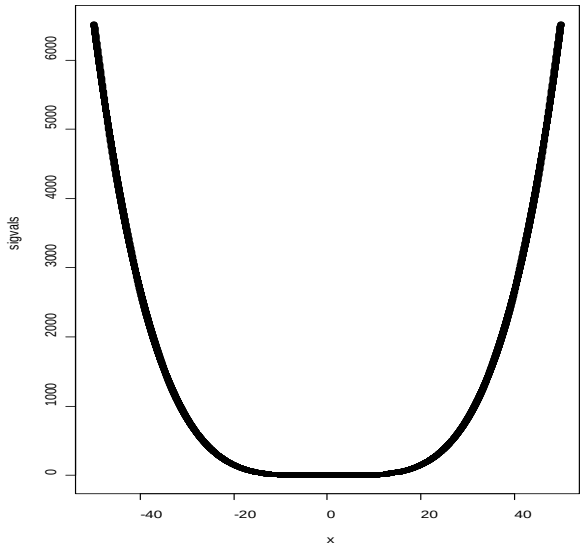


Accepted 81480 out of 100000 (81.480%).
final beta = -7.021089
alpha(0.000000) is 0.839500
alpha(2.000000) is 0.759000
alpha(5.000000) is 0.955800
alpha(8.000000) is 0.628500
alpha(10.000000) is 0.194600
alpha(13.000000) is 0.135300
alpha(15.000000) is 0.129300
alpha(20.000000) is 0.068600
alpha(30.000000) is 0.021900
alpha(50.000000) is 0.005500
it takes 346.280000 seconds to run the adaptation algorithm
it takes 152.760000 seconds to compute g and sigma
it takes 163.380000 seconds to generate the first Markov Chain
it takes 168.860000 seconds to generate the second Markov Chain
it takes 171.100000 seconds to generate the third Markov Chain
it takes 166.650000 seconds to generate the fourth Markov Chain
it takes 166.330000 seconds to generate the fifth Markov Chain
it takes 829.620000 seconds to test the local acceptance

[1] 21.36030
[1] 0.01514145
[1] 0.1902173

2.3.3.5 case 160 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.5, \text{width} = 2, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

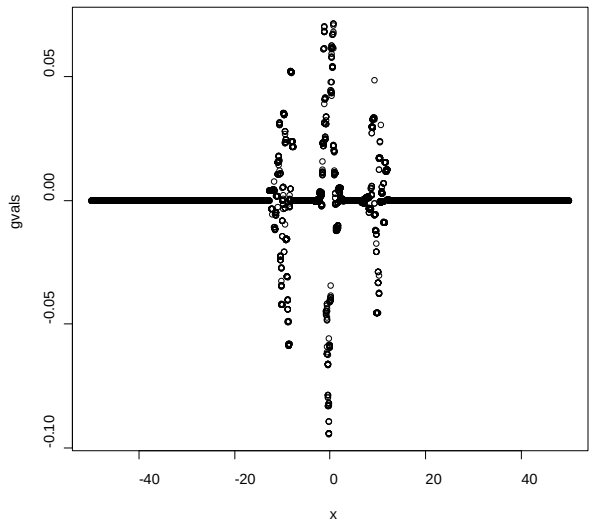


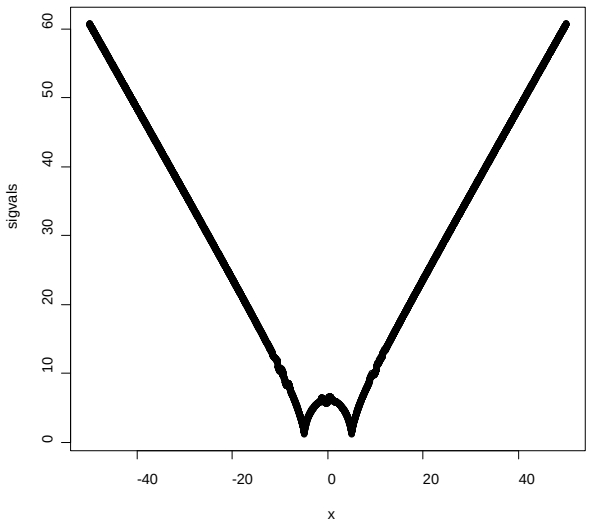


Accepted 22930 out of 100000 (22.930%).
final beta = -6.845094
alpha(0.000000) is 0.788500
alpha(2.000000) is 0.738200
alpha(5.000000) is 0.955200
alpha(8.000000) is 0.590000
alpha(10.000000) is 0.195100
alpha(13.000000) is 0.119600
alpha(15.000000) is 0.102500
alpha(20.000000) is 0.060000
alpha(30.000000) is 0.023800
alpha(50.000000) is 0.004400
it takes 340.490000 seconds to run the adaptation algorithm
it takes 152.290000 seconds to compute g and sigma
it takes 166.240000 seconds to generate the first Markov Chain
it takes 166.180000 seconds to generate the second Markov Chain
it takes 172.370000 seconds to generate the third Markov Chain
it takes 166.020000 seconds to generate the fourth Markov Chain
it takes 166.830000 seconds to generate the fifth Markov Chain
it takes 857.250000 seconds to test the local acceptance

[1] 91.28385
[1] 0.3193566
[1] 1.143155

2.3.3.6 case 161 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.5, \text{width} = 0.5, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

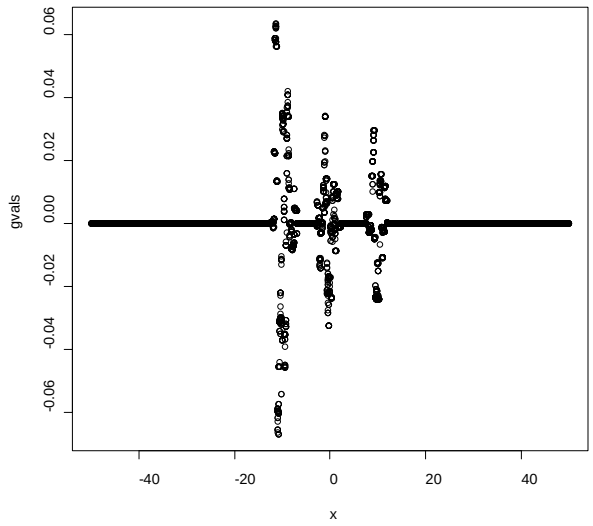


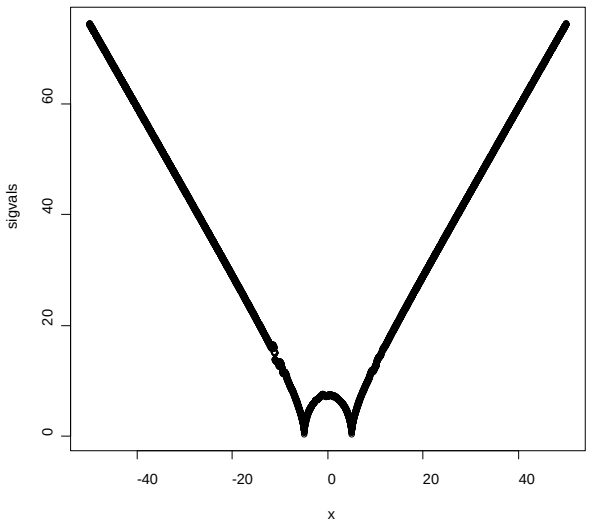


Accepted 25782 out of 100000 (25.782%).
final beta = 0.199733
alpha(0.000000) is 0.251500
alpha(2.000000) is 0.478800
alpha(5.000000) is 0.974700
alpha(8.000000) is 0.393500
alpha(10.000000) is 0.162700
alpha(13.000000) is 0.315100
alpha(15.000000) is 0.416800
alpha(20.000000) is 0.452300
alpha(30.000000) is 0.442600
alpha(50.000000) is 0.449500
it takes 347.050000 seconds to run the adaptation algorithm
it takes 152.430000 seconds to compute g and sigma
it takes 166.980000 seconds to generate the first Markov Chain
it takes 166.170000 seconds to generate the second Markov Chain
it takes 166.680000 seconds to generate the third Markov Chain
it takes 167.440000 seconds to generate the fourth Markov Chain
it takes 165.880000 seconds to generate the fifth Markov Chain
it takes 561.600000 seconds to test the local acceptance

[1] 17.85295
[1] 0.1088777
[1] 13.29712

2.3.3.7 case 162 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.5, \text{width} = 0.5, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

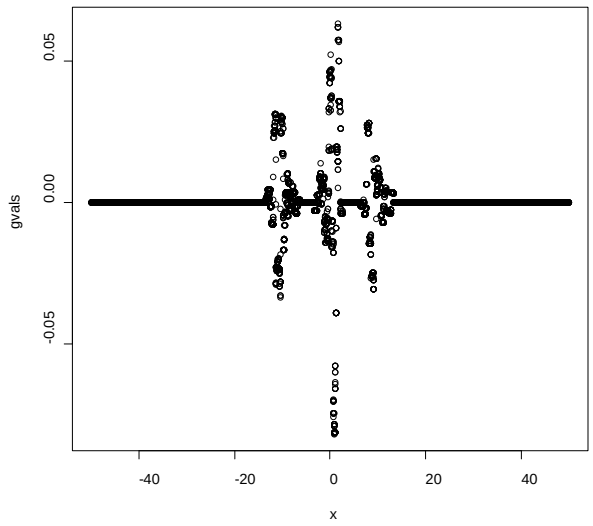


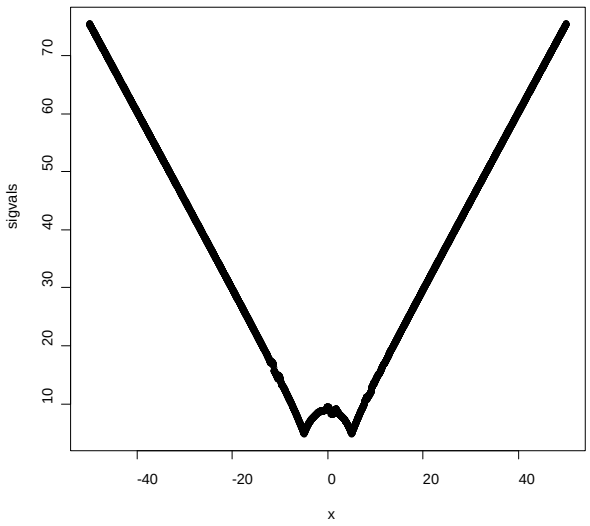


Accepted 24413 out of 100000 (24.413%).
final beta = -0.748534
alpha(0.000000) is 0.225100
alpha(2.000000) is 0.463600
alpha(5.000000) is 0.665100
alpha(8.000000) is 0.384400
alpha(10.000000) is 0.169500
alpha(13.000000) is 0.307900
alpha(15.000000) is 0.385300
alpha(20.000000) is 0.420700
alpha(30.000000) is 0.418600
alpha(50.000000) is 0.408900
it takes 347.780000 seconds to run the adaptation algorithm
it takes 152.450000 seconds to compute g and sigma
it takes 168.000000 seconds to generate the first Markov Chain
it takes 168.100000 seconds to generate the second Markov Chain
it takes 167.180000 seconds to generate the third Markov Chain
it takes 168.710000 seconds to generate the fourth Markov Chain
it takes 167.550000 seconds to generate the fifth Markov Chain
it takes 585.160000 seconds to test the local acceptance

[1] 13.60182
[1] 0.09651798
[1] 16.46537

2.3.3.8 case 163 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.5, \text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

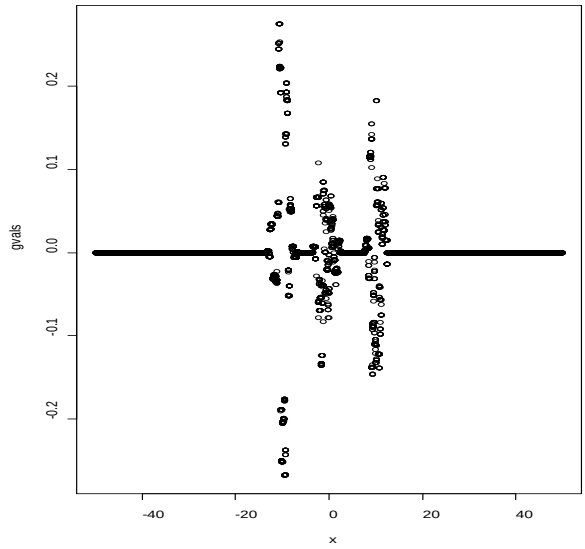


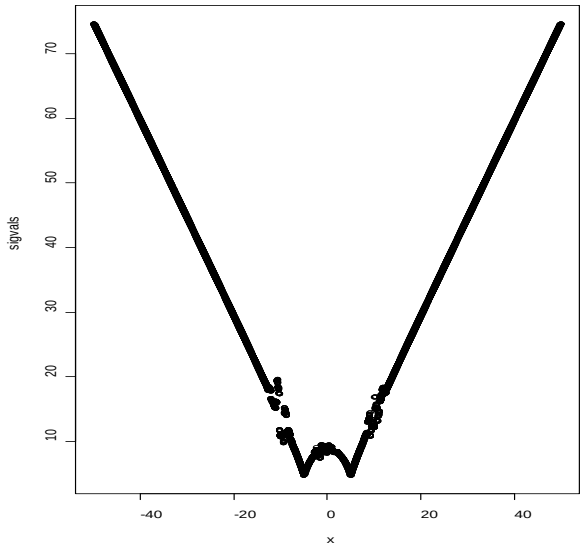


Accepted 23030 out of 100000 (23.030%).
final beta = 1.566764
alpha(0.000000) is 0.196400
alpha(2.000000) is 0.438500
alpha(5.000000) is 0.984500
alpha(8.000000) is 0.361400
alpha(10.000000) is 0.169300
alpha(13.000000) is 0.303900
alpha(15.000000) is 0.397200
alpha(20.000000) is 0.406300
alpha(30.000000) is 0.407600
alpha(50.000000) is 0.404800
it takes 347.430000 seconds to run the adaptation algorithm
it takes 152.300000 seconds to compute g and sigma
it takes 169.060000 seconds to generate the first Markov Chain
it takes 169.210000 seconds to generate the second Markov Chain
it takes 169.940000 seconds to generate the third Markov Chain
it takes 168.840000 seconds to generate the fourth Markov Chain
it takes 168.600000 seconds to generate the fifth Markov Chain
it takes 593.080000 seconds to test the local acceptance

[1] 12.99476
[1] 0.09387773
[1] 17.25407

2.3.3.9 case 164 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 2, \text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

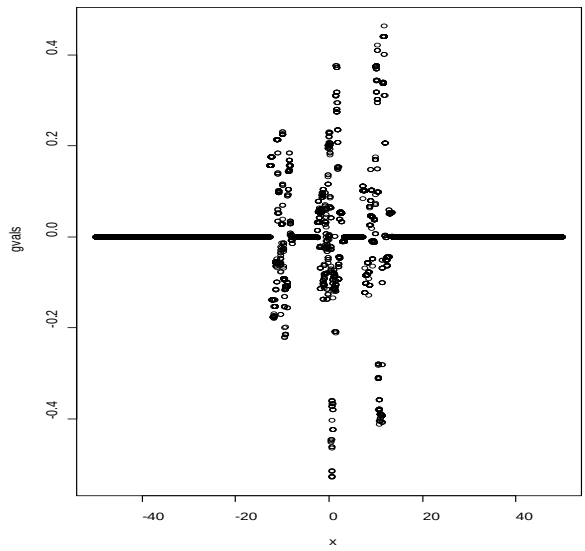


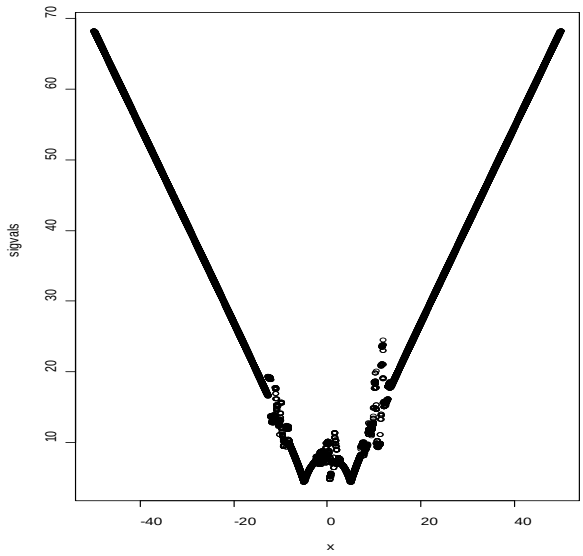


Accepted 23148 out of 100000 (23.148%).
final beta = 1.555008
alpha(0.000000) is 0.197200
alpha(2.000000) is 0.436800
alpha(5.000000) is 0.985200
alpha(8.000000) is 0.354100
alpha(10.000000) is 0.163900
alpha(13.000000) is 0.315100
alpha(15.000000) is 0.414400
alpha(20.000000) is 0.416100
alpha(30.000000) is 0.405600
alpha(50.000000) is 0.405300
it takes 346.860000 seconds to run the adaptation algorithm
it takes 152.730000 seconds to compute g and sigma
it takes 169.540000 seconds to generate the first Markov Chain
it takes 170.330000 seconds to generate the second Markov Chain
it takes 169.700000 seconds to generate the third Markov Chain
it takes 170.450000 seconds to generate the fourth Markov Chain
it takes 169.770000 seconds to generate the fifth Markov Chain
it takes 590.270000 seconds to test the local acceptance

[1] 13.87151
[1] 0.09678516
[1] 17.20643

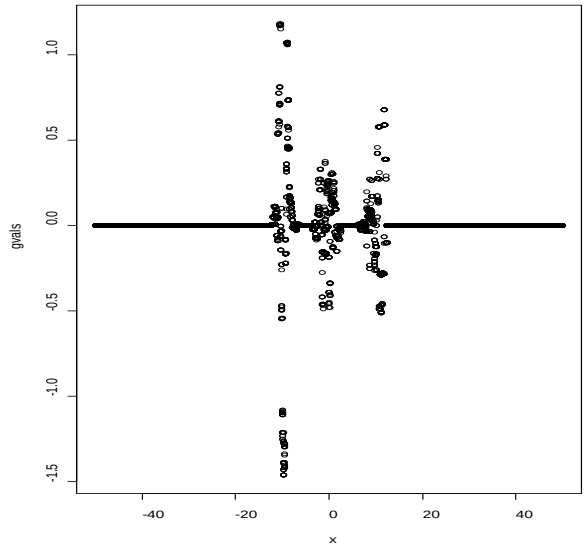
2.3.3.10 case 165 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 5, \text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

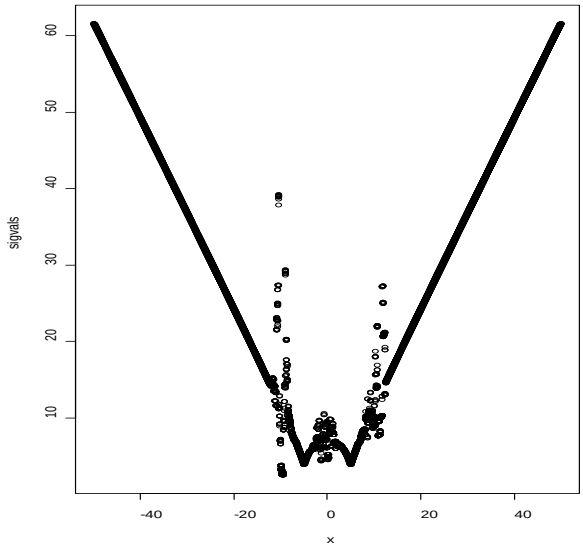




Accepted 23545 out of 100000 (23.545%).
final beta = 1.465999
alpha(0.000000) is 0.197500
alpha(2.000000) is 0.460100
alpha(5.000000) is 0.990300
alpha(8.000000) is 0.384100
alpha(10.000000) is 0.161600
alpha(13.000000) is 0.315100
alpha(15.000000) is 0.423100
alpha(20.000000) is 0.435000
alpha(30.000000) is 0.436200
alpha(50.000000) is 0.425000
it takes 346.630000 seconds to run the adaptation algorithm
it takes 152.570000 seconds to compute g and sigma
it takes 169.800000 seconds to generate the first Markov Chain
it takes 169.690000 seconds to generate the second Markov Chain
it takes 169.740000 seconds to generate the third Markov Chain
it takes 169.530000 seconds to generate the fourth Markov Chain
it takes 168.990000 seconds to generate the fifth Markov Chain
it takes 577.110000 seconds to test the local acceptance rate

2.3.3.11 case 166 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 10, \text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

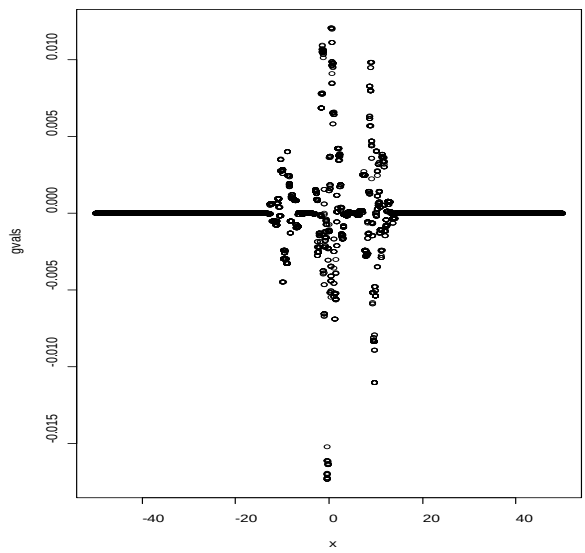


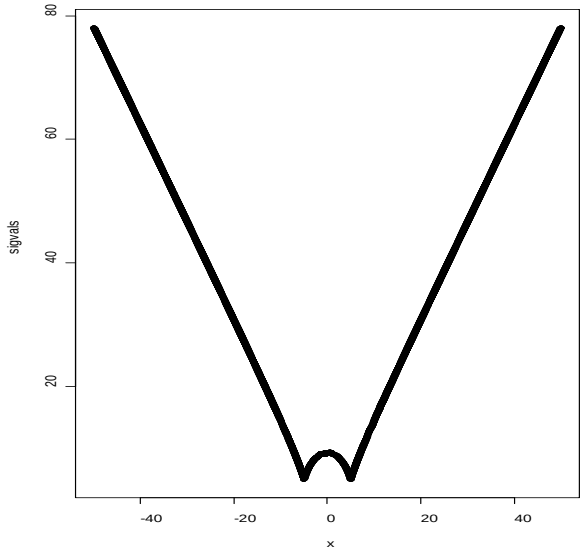


Accepted 22643 out of 100000 (22.643%).
final beta = 1.363053
alpha(0.000000) is 0.212000
alpha(2.000000) is 0.456200
alpha(5.000000) is 0.994700
alpha(8.000000) is 0.386900
alpha(10.000000) is 0.169600
alpha(13.000000) is 0.329200
alpha(15.000000) is 0.430400
alpha(20.000000) is 0.446300
alpha(30.000000) is 0.447600
alpha(50.000000) is 0.443200
it takes 346.650000 seconds to run the adaptation algorithm
it takes 152.580000 seconds to compute g and sigma
it takes 177.310000 seconds to generate the first Markov Chain
it takes 177.070000 seconds to generate the second Markov Chain
it takes 176.700000 seconds to generate the third Markov Chain
it takes 176.140000 seconds to generate the fourth Markov Chain
it takes 177.500000 seconds to generate the fifth Markov Chain
it takes 563.200000 seconds to test the local acceptance

[1] 21.28494
[1] 0.1196420
[1] 12.15185

2.3.3.12 case 167 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

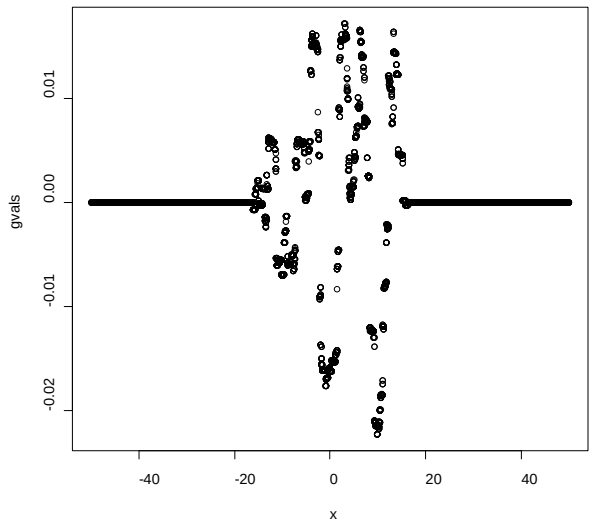


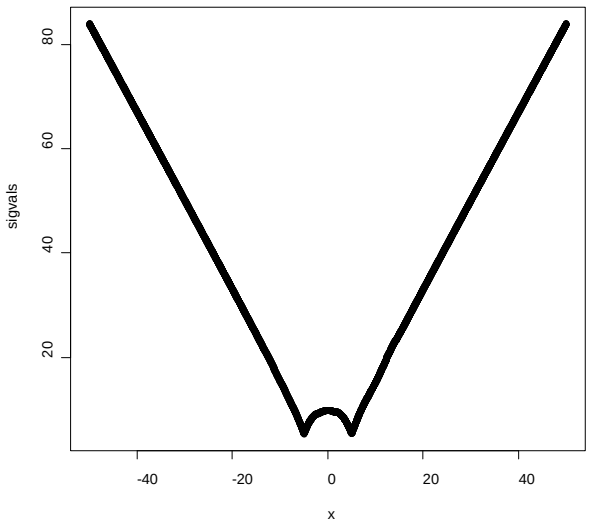


Accepted 22697 out of 100000 (22.697%).
final beta = 1.599916
alpha(0.000000) is 0.185500
alpha(2.000000) is 0.432600
alpha(5.000000) is 0.980600
alpha(8.000000) is 0.360300
alpha(10.000000) is 0.157000
alpha(13.000000) is 0.309100
alpha(15.000000) is 0.398400
alpha(20.000000) is 0.410200
alpha(30.000000) is 0.395600
alpha(50.000000) is 0.392800
it takes 347.310000 seconds to run the adaptation algorithm
it takes 152.460000 seconds to compute g and sigma
it takes 169.840000 seconds to generate the first Markov Chain
it takes 169.340000 seconds to generate the second Markov Chain
it takes 170.130000 seconds to generate the third Markov Chain
it takes 169.930000 seconds to generate the fourth Markov Chain
it takes 170.120000 seconds to generate the fifth Markov Chain
it takes 597.480000 seconds to test the local acceptance

[1] 13.29028
[1] 0.09469267
[1] 17.58271

2.3.3.13 case 168 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

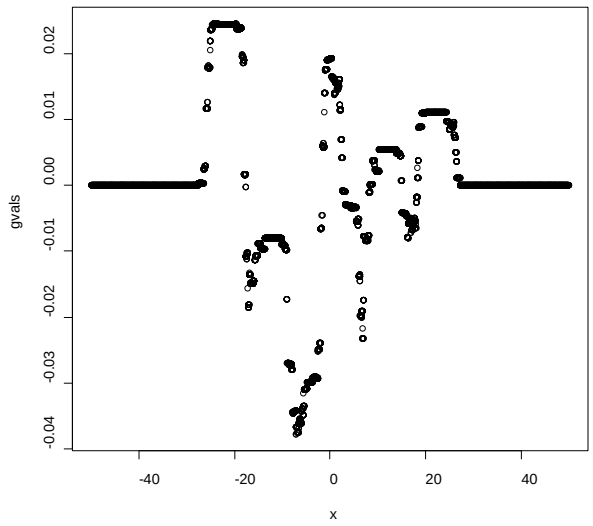


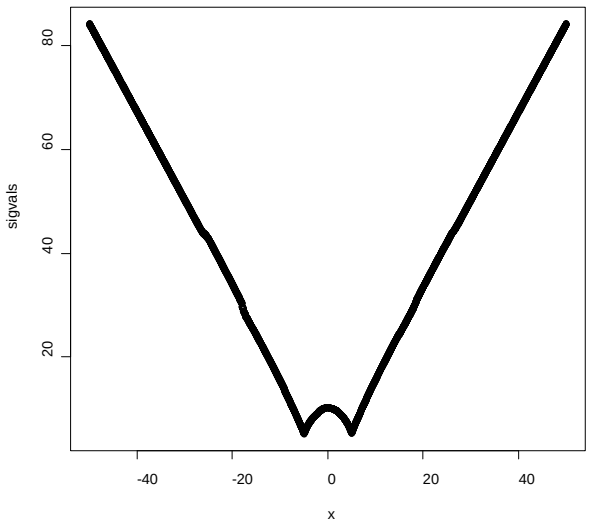


Accepted 21952 out of 100000 (21.952%).
final beta = 1.672013
alpha(0.000000) is 0.184000
alpha(2.000000) is 0.418100
alpha(5.000000) is 0.968800
alpha(8.000000) is 0.343500
alpha(10.000000) is 0.158900
alpha(13.000000) is 0.300400
alpha(15.000000) is 0.388000
alpha(20.000000) is 0.388200
alpha(30.000000) is 0.379900
alpha(50.000000) is 0.380900
it takes 344.410000 seconds to run the adaptation algorithm
it takes 152.050000 seconds to compute g and sigma
it takes 172.150000 seconds to generate the first Markov Chain
it takes 173.150000 seconds to generate the second Markov Chain
it takes 170.720000 seconds to generate the third Markov Chain
it takes 171.580000 seconds to generate the fourth Markov Chain
it takes 171.530000 seconds to generate the fifth Markov Chain
it takes 609.770000 seconds to test the local acceptance

[1] 13.51672
[1] 0.09560705
[1] 17.78677

2.3.3.14 case 169 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 8, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

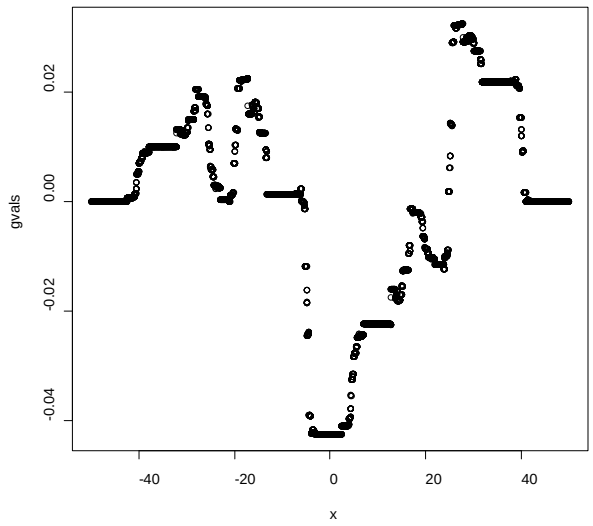


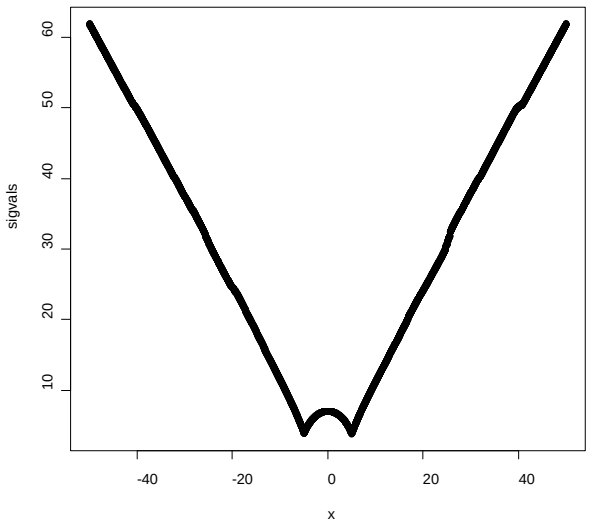


Accepted 21813 out of 100000 (21.813%).
final beta = 1.675306
alpha(0.000000) is 0.176600
alpha(2.000000) is 0.413300
alpha(5.000000) is 0.969100
alpha(8.000000) is 0.342100
alpha(10.000000) is 0.154200
alpha(13.000000) is 0.297700
alpha(15.000000) is 0.386400
alpha(20.000000) is 0.385700
alpha(30.000000) is 0.384000
alpha(50.000000) is 0.385500
it takes 336.020000 seconds to run the adaptation algorithm
it takes 150.700000 seconds to compute g and sigma
it takes 172.710000 seconds to generate the first Markov Chain
it takes 172.260000 seconds to generate the second Markov Chain
it takes 171.770000 seconds to generate the third Markov Chain
it takes 173.580000 seconds to generate the fourth Markov Chain
it takes 172.910000 seconds to generate the fifth Markov Chain
it takes 609.330000 seconds to test the local acceptance

[1] 13.63757
[1] 0.09651497
[1] 17.84017

2.3.3.15 case 170 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 15, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

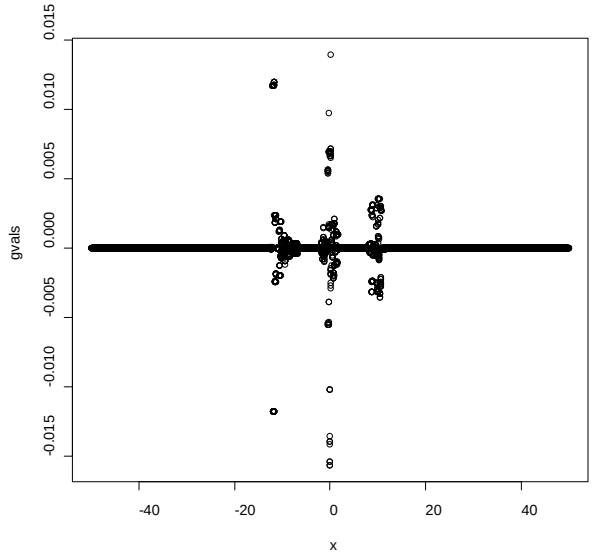


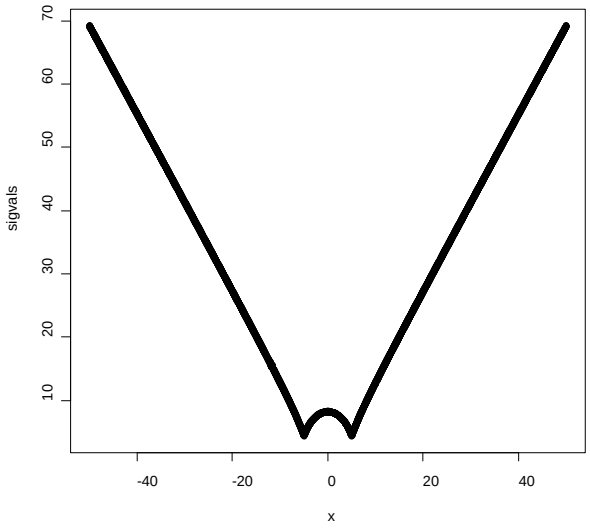


Accepted 24968 out of 100000 (24.968%).
final beta = 1.368241
alpha(0.000000) is 0.237200
alpha(2.000000) is 0.467000
alpha(5.000000) is 0.996200
alpha(8.000000) is 0.390800
alpha(10.000000) is 0.173500
alpha(13.000000) is 0.325900
alpha(15.000000) is 0.432200
alpha(20.000000) is 0.452500
alpha(30.000000) is 0.452300
alpha(50.000000) is 0.439200
it takes 330.360000 seconds to run the adaptation algorithm
it takes 148.840000 seconds to compute g and sigma
it takes 166.510000 seconds to generate the first Markov Chain
it takes 166.620000 seconds to generate the second Markov Chain
it takes 166.570000 seconds to generate the third Markov Chain
it takes 167.160000 seconds to generate the fourth Markov Chain
it takes 166.690000 seconds to generate the fifth Markov Chain
it takes 563.370000 seconds to test the local acceptance

[1] 15.62307
[1] 0.1033918
[1] 14.97829

2.3.3.16 case 171 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 0.1, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

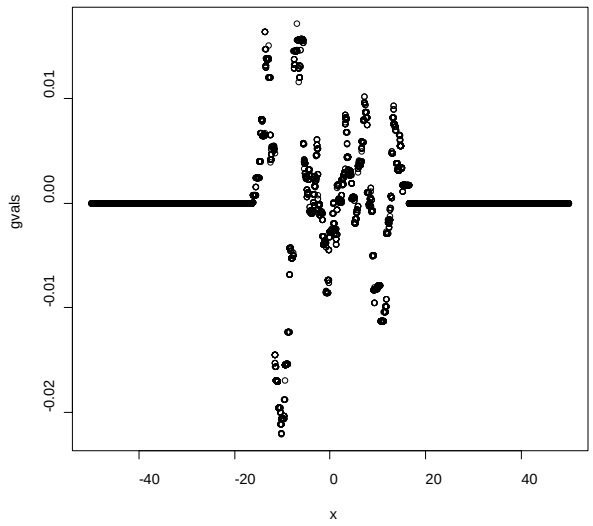


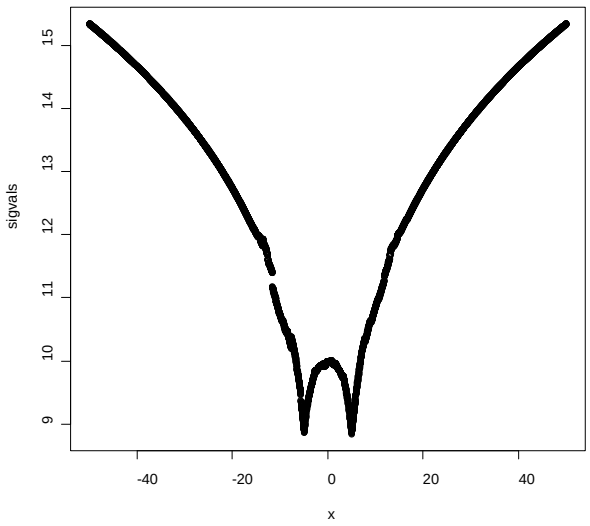


Accepted 24069 out of 100000 (24.069%).
final beta = 1.479692
alpha(0.000000) is 0.205700
alpha(2.000000) is 0.444900
alpha(5.000000) is 0.989800
alpha(8.000000) is 0.365500
alpha(10.000000) is 0.167200
alpha(13.000000) is 0.322200
alpha(15.000000) is 0.422300
alpha(20.000000) is 0.426700
alpha(30.000000) is 0.419800
alpha(50.000000) is 0.428600
it takes 348.040000 seconds to run the adaptation algorithm
it takes 152.510000 seconds to compute g and sigma
it takes 167.270000 seconds to generate the first Markov Chain
it takes 167.570000 seconds to generate the second Markov Chain
it takes 167.280000 seconds to generate the third Markov Chain
it takes 167.430000 seconds to generate the fourth Markov Chain
it takes 167.320000 seconds to generate the fifth Markov Chain
it takes 576.120000 seconds to test the local acceptance

[1] 13.0853
[1] 0.09413097
[1] 16.67941

2.3.3.17 case 172 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.1$ with fixed bandwidth $b_n = 1$

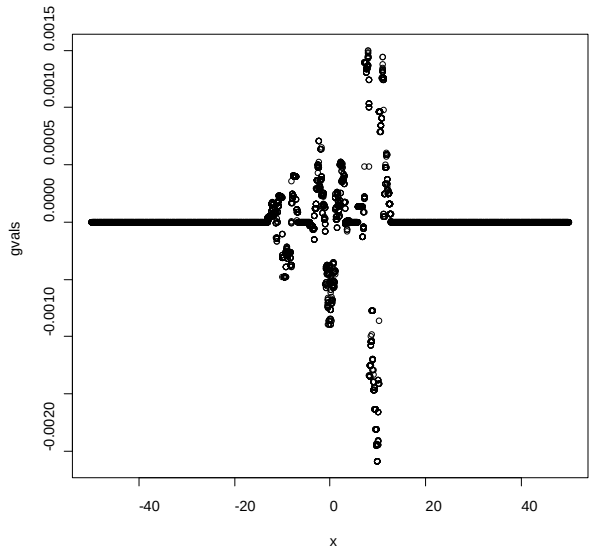


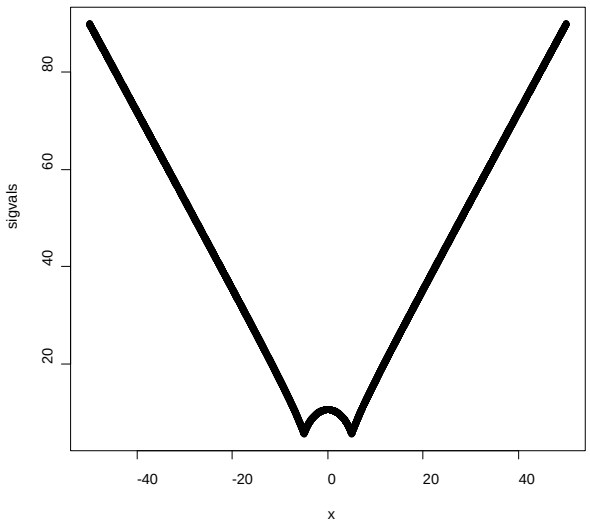


Accepted 23706 out of 100000 (23.706%).
final beta = 2.179023
alpha(0.000000) is 0.189700
alpha(2.000000) is 0.407500
alpha(5.000000) is 0.867700
alpha(8.000000) is 0.363500
alpha(10.000000) is 0.178000
alpha(13.000000) is 0.375300
alpha(15.000000) is 0.498500
alpha(20.000000) is 0.508500
alpha(30.000000) is 0.499000
alpha(50.000000) is 0.503400
it takes 131.989000 seconds to run the adaptation algorithm
it takes 56.976000 seconds to compute g and sigma
it takes 19.196000 seconds to generate the first Markov Chain
it takes 19.213000 seconds to generate the second Markov Chain
it takes 19.843000 seconds to generate the third Markov Chain
it takes 19.346000 seconds to generate the fourth Markov Chain
it takes 20.059000 seconds to generate the fifth Markov Chain
it takes 143.342000 seconds to test the local acceptance

[1] 14.62442
[1] 0.09965965
[1] 15.71839

2.3.3.18 case 173 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 10$

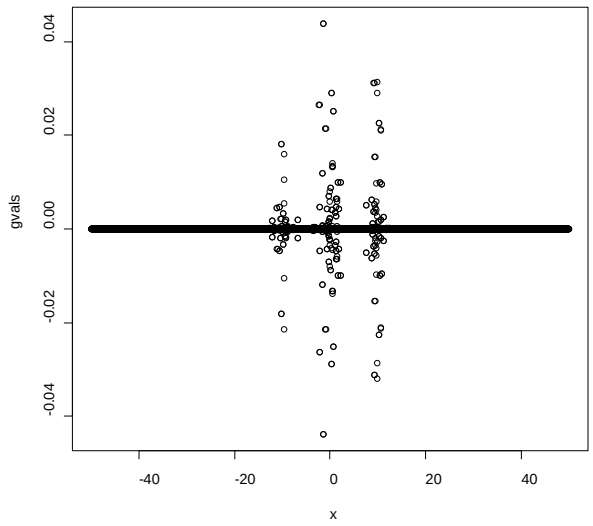


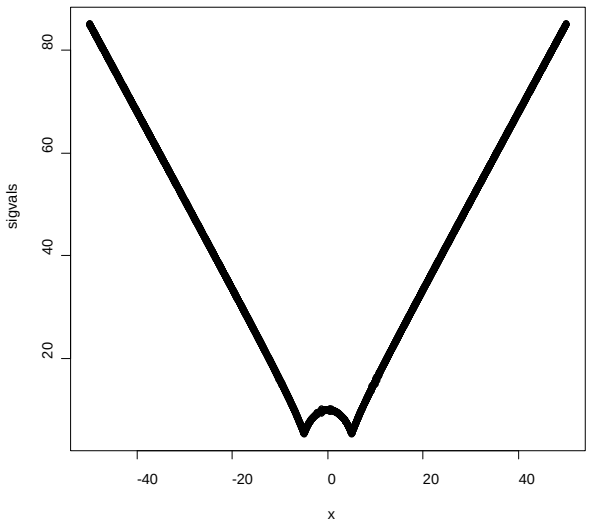


Accepted 21044 out of 100000 (21.044%).
final beta = 1.740088
alpha(0.000000) is 0.167200
alpha(2.000000) is 0.396400
alpha(5.000000) is 0.960700
alpha(8.000000) is 0.323100
alpha(10.000000) is 0.158800
alpha(13.000000) is 0.281300
alpha(15.000000) is 0.374900
alpha(20.000000) is 0.364600
alpha(30.000000) is 0.374800
alpha(50.000000) is 0.362200
it takes 345.360000 seconds to run the adaptation algorithm
it takes 151.990000 seconds to compute g and sigma
it takes 175.400000 seconds to generate the first Markov Chain
it takes 175.550000 seconds to generate the second Markov Chain
it takes 175.420000 seconds to generate the third Markov Chain
it takes 175.200000 seconds to generate the fourth Markov Chain
it takes 175.200000 seconds to generate the fifth Markov Chain
it takes 623.140000 seconds to test the local acceptance

[1] 12.96732
[1] 0.0933517
[1] 17.73759

2.3.3.19 case 174 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 0.1$

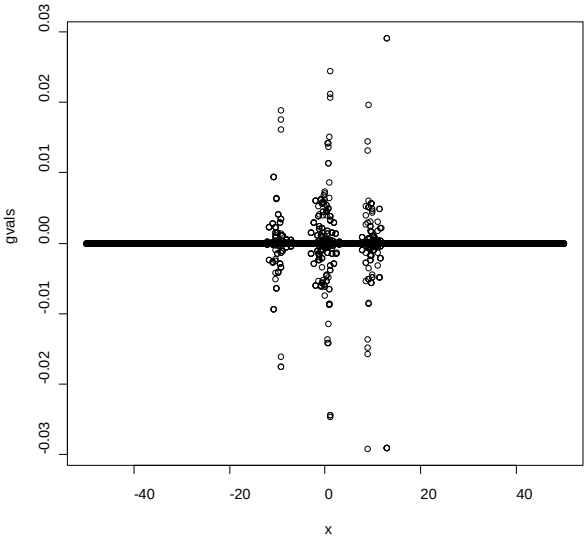


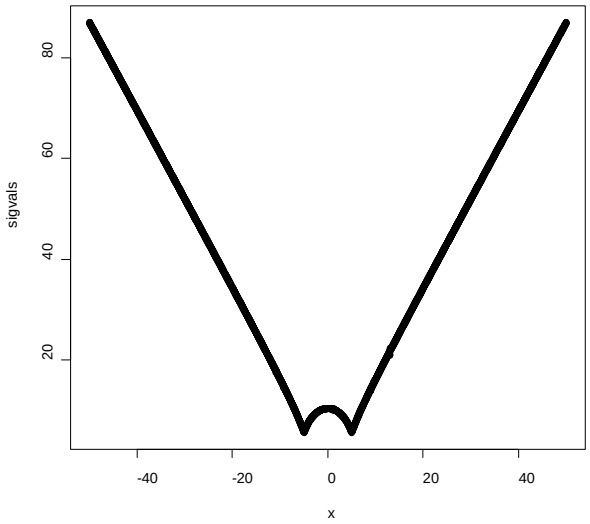


Accepted 21641 out of 100000 (21.641%).
final beta = 1.685472
alpha(0.000000) is 0.176700
alpha(2.000000) is 0.411100
alpha(5.000000) is 0.969100
alpha(8.000000) is 0.340200
alpha(10.000000) is 0.158400
alpha(13.000000) is 0.295200
alpha(15.000000) is 0.388400
alpha(20.000000) is 0.386200
alpha(30.000000) is 0.389100
alpha(50.000000) is 0.378400
it takes 348.360000 seconds to run the adaptation algorithm
it takes 152.300000 seconds to compute g and sigma
it takes 173.130000 seconds to generate the first Markov Chain
it takes 173.530000 seconds to generate the second Markov Chain
it takes 172.380000 seconds to generate the third Markov Chain
it takes 171.800000 seconds to generate the fourth Markov Chain
it takes 172.980000 seconds to generate the fifth Markov Chain
it takes 612.070000 seconds to test the local acceptance

[1] 12.68868
[1] 0.09229896
[1] 17.97203

2.3.3.20 case 175 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

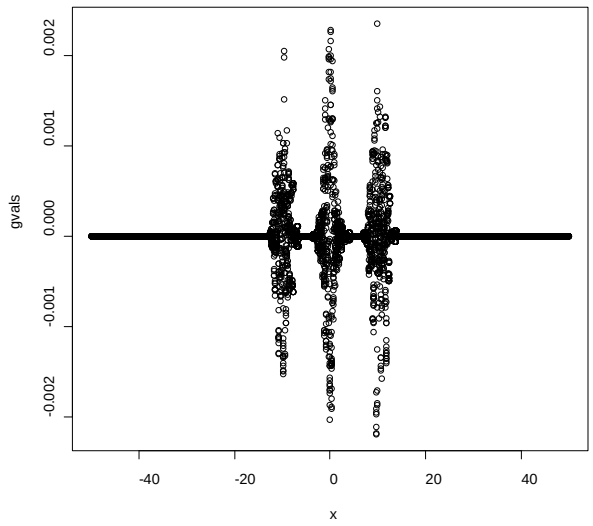


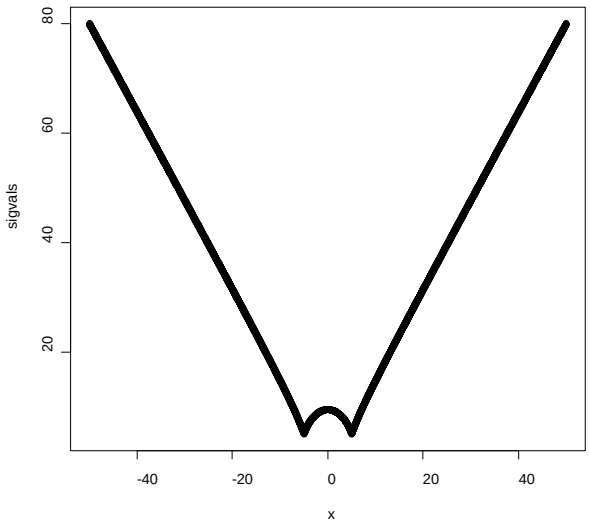


Accepted 21390 out of 100000 (21.390%).
final beta = 1.708766
alpha(0.000000) is 0.163300
alpha(2.000000) is 0.404300
alpha(5.000000) is 0.966800
alpha(8.000000) is 0.344400
alpha(10.000000) is 0.157300
alpha(13.000000) is 0.298100
alpha(15.000000) is 0.369800
alpha(20.000000) is 0.371100
alpha(30.000000) is 0.378700
alpha(50.000000) is 0.382100
it takes 348.370000 seconds to run the adaptation algorithm
it takes 152.520000 seconds to compute g and sigma
it takes 173.190000 seconds to generate the first Markov Chain
it takes 174.010000 seconds to generate the second Markov Chain
it takes 173.840000 seconds to generate the third Markov Chain
it takes 172.650000 seconds to generate the fourth Markov Chain
it takes 173.240000 seconds to generate the fifth Markov Chain
it takes 615.190000 seconds to test the local acceptance

[1] 12.13882
[1] 0.09072717
[1] 18.01603

2.3.3.21 case 176 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$



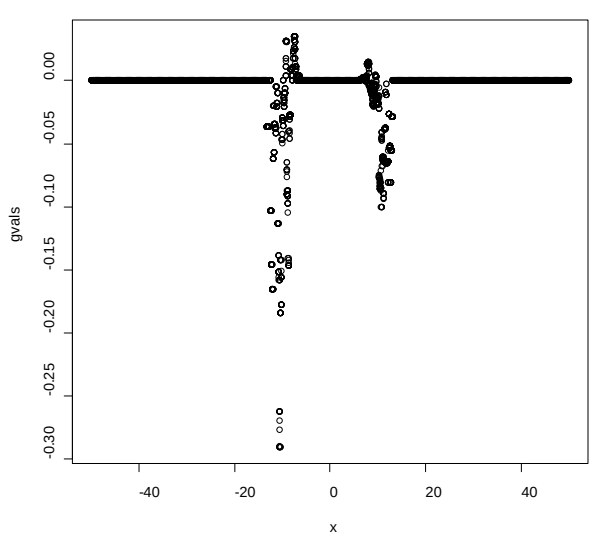


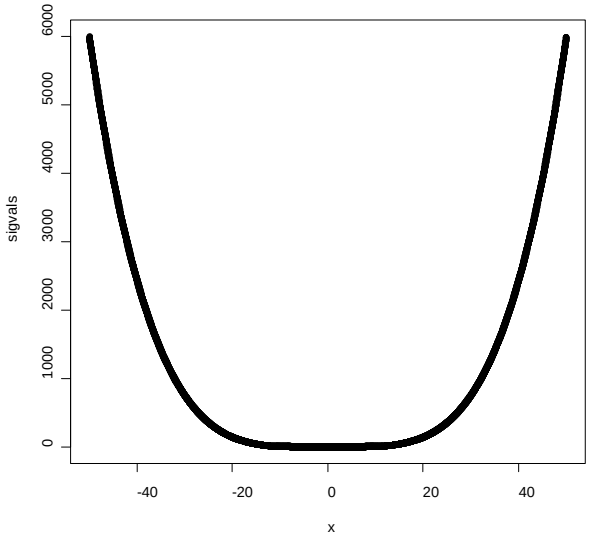
Accepted 22542 out of 100000 (22.542%).
final beta = 1.624557
alpha(0.000000) is 0.194300
alpha(2.000000) is 0.423500
alpha(5.000000) is 0.975500
alpha(8.000000) is 0.349000
alpha(10.000000) is 0.161200
alpha(13.000000) is 0.310700
alpha(15.000000) is 0.401300
alpha(20.000000) is 0.394200
alpha(30.000000) is 0.396300
alpha(50.000000) is 0.403000
it takes 705.820000 seconds to run the adaptation algorithm
it takes 308.400000 seconds to compute g and sigma
it takes 175.690000 seconds to generate the first Markov Chain
it takes 175.800000 seconds to generate the second Markov Chain
it takes 175.520000 seconds to generate the third Markov Chain
it takes 175.250000 seconds to generate the fourth Markov Chain
it takes 175.900000 seconds to generate the fifth Markov Chain
it takes 1031.100000 seconds to test the local acceptance

[1] 12.53176
[1] 0.09117011
[1] 17.39284

2.3.4 kernel function $K(x)=\begin{cases}0, |x|\geq 2*width \quad or \quad |x|\leq width\\ 1, width<|x|<2*width\end{cases}$

2.3.4.1 case 177 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\text{width}=0.5, C=1, \gamma=2$ with fixed bandwidth $b_n=1$

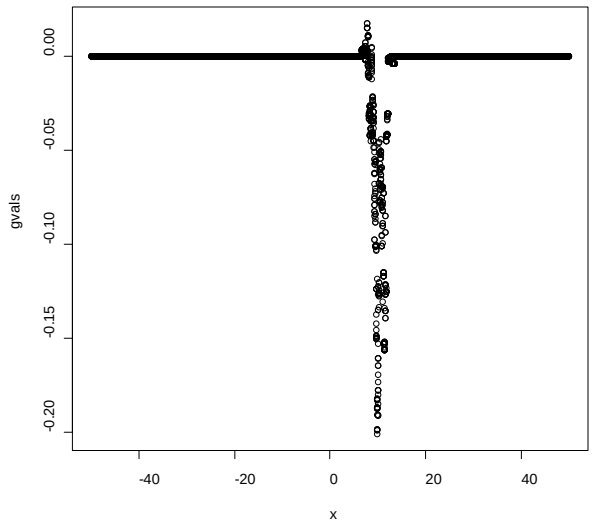


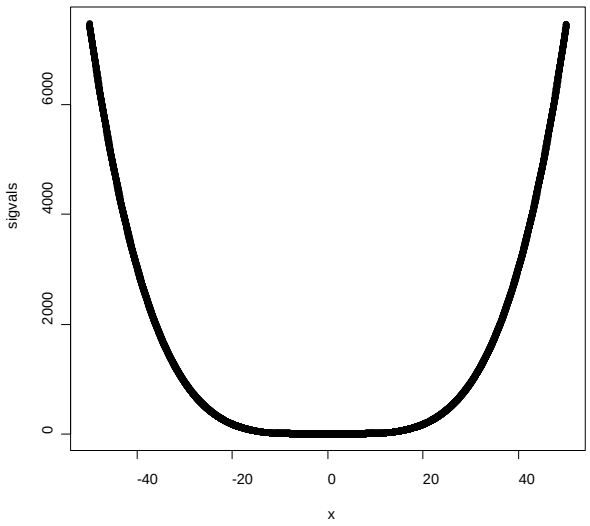


Accepted 24315 out of 100000 (24.315%).
final beta = -6.929724
alpha(0.000000) is 0.813800
alpha(2.000000) is 0.753800
alpha(5.000000) is 0.955900
alpha(8.000000) is 0.620100
alpha(10.000000) is 0.186500
alpha(13.000000) is 0.128900
alpha(15.000000) is 0.113300
alpha(20.000000) is 0.067100
alpha(30.000000) is 0.023400
alpha(50.000000) is 0.005600
it takes 125.482000 seconds to run the adaptation algorithm
it takes 55.504000 seconds to compute g and sigma
it takes 18.709000 seconds to generate the first Markov Chain
it takes 19.010000 seconds to generate the second Markov Chain
it takes 19.325000 seconds to generate the third Markov Chain
it takes 18.488000 seconds to generate the fourth Markov Chain
it takes 18.261000 seconds to generate the fifth Markov Chain
it takes 264.813000 seconds to test the local acceptance

[1] 91.28675
[1] 0.3161601
[1] 1.140038

2.3.4.2 case 178 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 10, \gamma = 2$ with fixed bandwidth $b_n = 1$



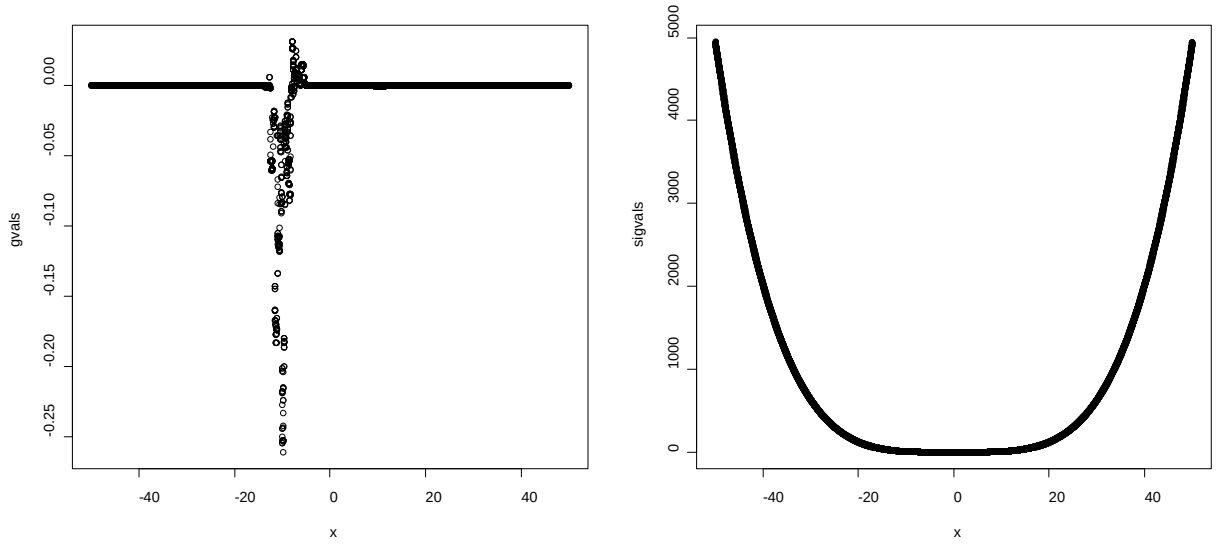


Accepted 77934 out of 100000 (77.934%).
final beta = -11.312342
alpha(0.000000) is 0.777300
alpha(2.000000) is 0.738200
alpha(5.000000) is 0.711200
alpha(8.000000) is 0.594200
alpha(10.000000) is 0.167200
alpha(13.000000) is 0.120000
alpha(15.000000) is 0.099900
alpha(20.000000) is 0.057700
alpha(30.000000) is 0.017300
alpha(50.000000) is 0.004100
it takes 124.417000 seconds to run the adaptation algorithm
it takes 55.553000 seconds to compute g and sigma
it takes 20.010000 seconds to generate the first Markov Chain
it takes 19.438000 seconds to generate the second Markov Chain
it takes 18.998000 seconds to generate the third Markov Chain
it takes 19.289000 seconds to generate the fourth Markov Chain
it takes 19.039000 seconds to generate the fifth Markov Chain
it takes 274.555000 seconds to test the local acceptance

[1] 17.12798
[1] 0.01378514
[1] 0.2614159

2.3.4.3 case 179 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 0.1, \gamma = 2$ with fixed bandwidth

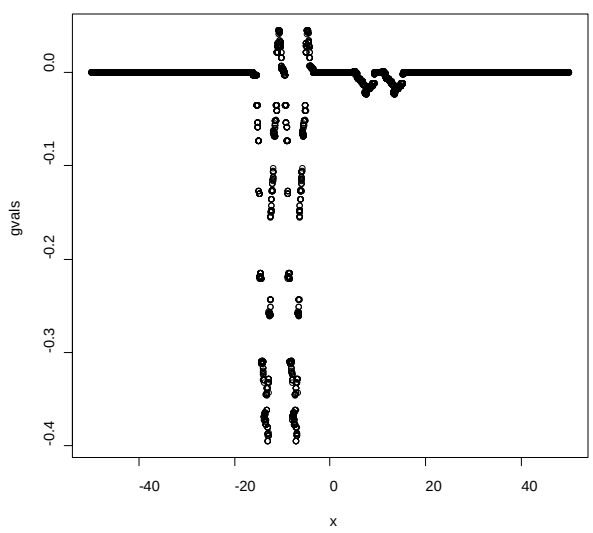
$b_n = 1$

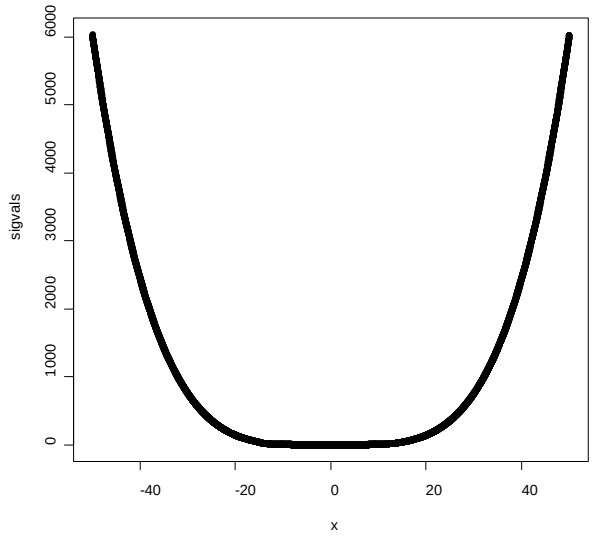


Accepted 23497 out of 100000 (23.497%).
final beta = -2.523769
alpha(0.000000) is 0.708600
alpha(2.000000) is 0.683200
alpha(5.000000) is 0.989400
alpha(8.000000) is 0.587200
alpha(10.000000) is 0.179800
alpha(13.000000) is 0.142000
alpha(15.000000) is 0.132400
alpha(20.000000) is 0.080800
alpha(30.000000) is 0.028700
alpha(50.000000) is 0.007200
it takes 124.323000 seconds to run the adaptation algorithm
it takes 55.545000 seconds to compute g and sigma
it takes 19.766000 seconds to generate the first Markov Chain
it takes 19.247000 seconds to generate the second Markov Chain
it takes 19.919000 seconds to generate the third Markov Chain
it takes 19.192000 seconds to generate the fourth Markov Chain
it takes 20.575000 seconds to generate the fifth Markov Chain
it takes 260.328000 seconds to test the local acceptance

[1] 91.63516
[1] 0.3196954
[1] 1.21477

2.3.4.4 case 180 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 2, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

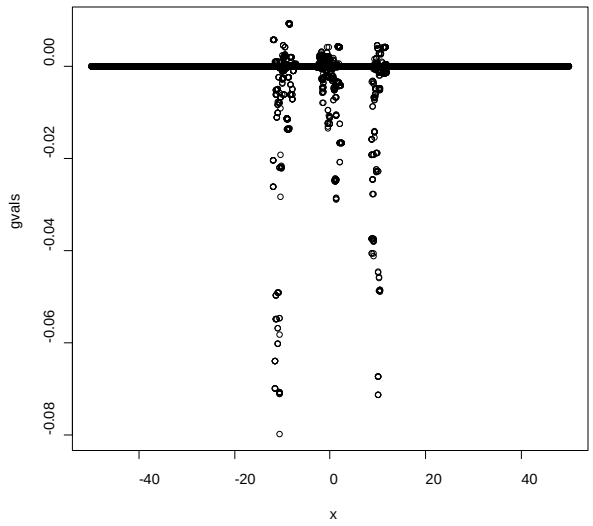


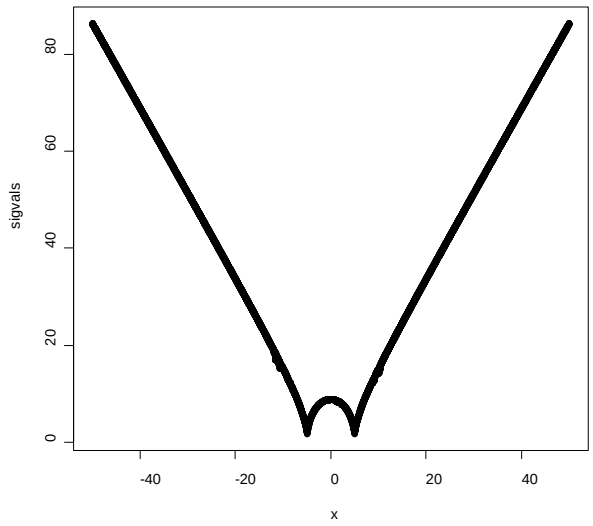


Accepted 23808 out of 100000 (23.808%).
final beta = -6.923705
alpha(0.000000) is 0.816100
alpha(2.000000) is 0.754200
alpha(5.000000) is 0.954400
alpha(8.000000) is 0.617700
alpha(10.000000) is 0.178500
alpha(13.000000) is 0.133000
alpha(15.000000) is 0.116000
alpha(20.000000) is 0.066300
alpha(30.000000) is 0.023700
alpha(50.000000) is 0.005400
it takes 124.565000 seconds to run the adaptation algorithm
it takes 55.222000 seconds to compute g and sigma
it takes 19.552000 seconds to generate the first Markov Chain
it takes 18.824000 seconds to generate the second Markov Chain
it takes 18.883000 seconds to generate the third Markov Chain
it takes 18.746000 seconds to generate the fourth Markov Chain
it takes 19.450000 seconds to generate the fifth Markov Chain
it takes 265.105000 seconds to test the local acceptance

[1] 88.68445
[1] 0.3080952
[1] 1.326619

2.3.4.5 case 181 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$



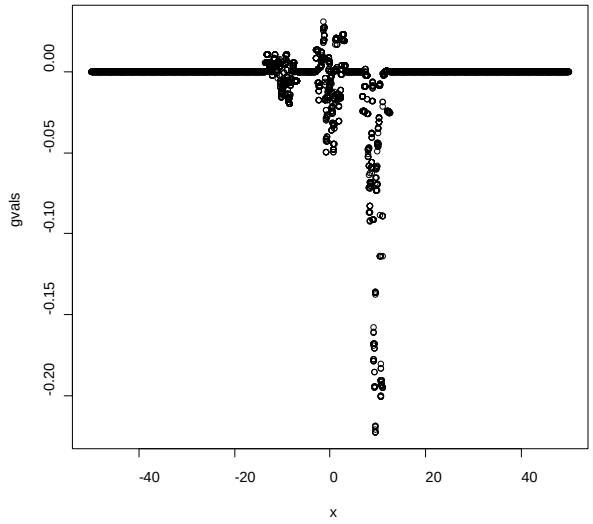


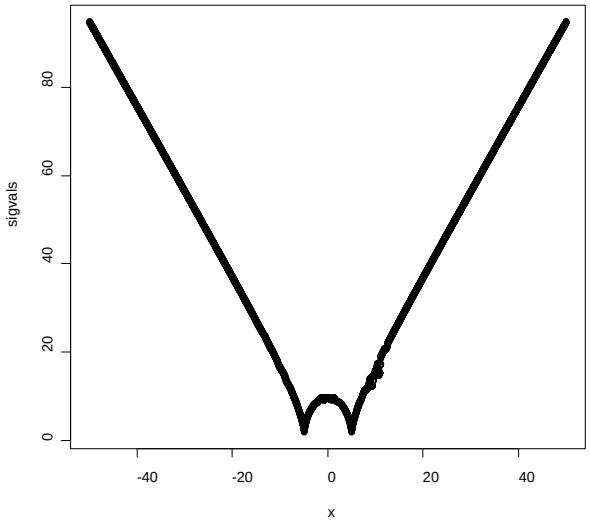
Accepted 22533 out of 100000 (22.533%).
final beta = 0.550883
alpha(0.000000) is 0.203900
alpha(2.000000) is 0.442800
alpha(5.000000) is 0.988000
alpha(8.000000) is 0.358500
alpha(10.000000) is 0.163500
alpha(13.000000) is 0.291900
alpha(15.000000) is 0.362400
alpha(20.000000) is 0.385800
alpha(30.000000) is 0.384800
alpha(50.000000) is 0.371000
it takes 124.380000 seconds to run the adaptation algorithm
it takes 55.442000 seconds to compute g and sigma
it takes 20.841000 seconds to generate the first Markov Chain
it takes 21.312000 seconds to generate the second Markov Chain
it takes 21.030000 seconds to generate the third Markov Chain
it takes 20.597000 seconds to generate the fourth Markov Chain
it takes 21.002000 seconds to generate the fifth Markov Chain
it takes 179.464000 seconds to test the local acceptance

[1] 12.88688
[1] 0.09355521
[1] 17.49032

2.3.4.6 case 182 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 1, \gamma = 0.5$ with fixed bandwidth

$b_n = 1$



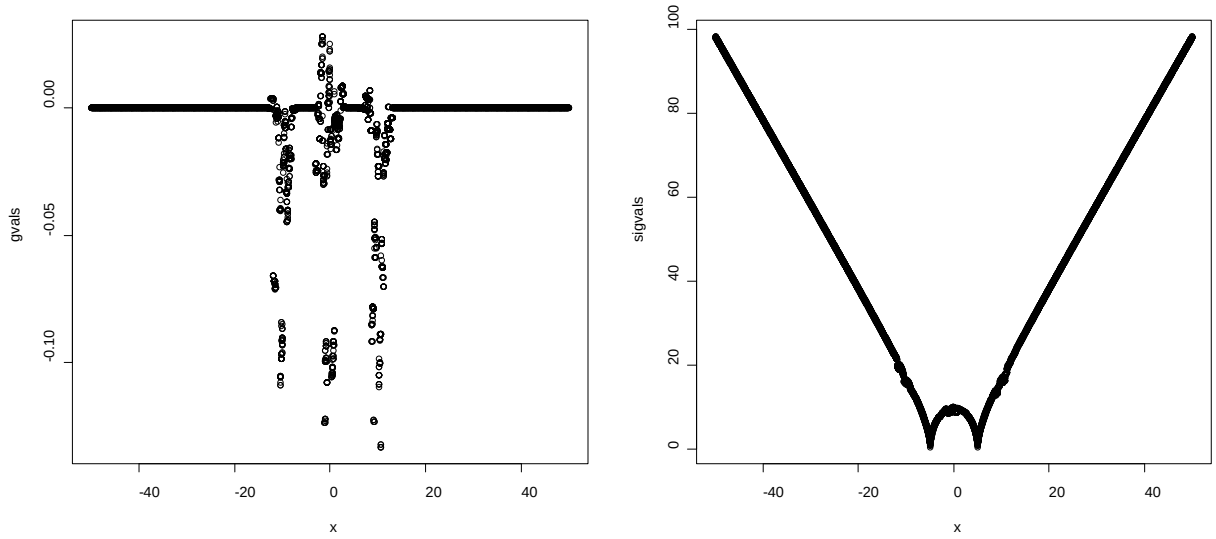


Accepted 21843 out of 100000 (21.843%).
final beta = 0.644714
alpha(0.000000) is 0.180400
alpha(2.000000) is 0.421000
alpha(5.000000) is 0.989900
alpha(8.000000) is 0.352300
alpha(10.000000) is 0.160300
alpha(13.000000) is 0.285800
alpha(15.000000) is 0.348500
alpha(20.000000) is 0.356200
alpha(30.000000) is 0.348300
alpha(50.000000) is 0.359800
it takes 123.740000 seconds to run the adaptation algorithm
it takes 54.826000 seconds to compute g and sigma
it takes 21.707000 seconds to generate the first Markov Chain
it takes 22.307000 seconds to generate the second Markov Chain
it takes 22.245000 seconds to generate the third Markov Chain
it takes 22.370000 seconds to generate the fourth Markov Chain
it takes 21.890000 seconds to generate the fifth Markov Chain
it takes 186.859000 seconds to test the local acceptance

[1] 12.92349
[1] 0.09353589
[1] 17.78533

2.3.4.7 case 183 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 10, \gamma = 0.5$ with fixed bandwidth

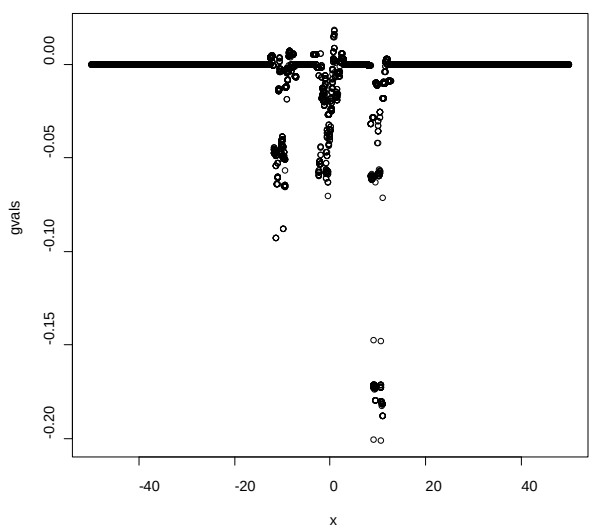
$b_n = 1$

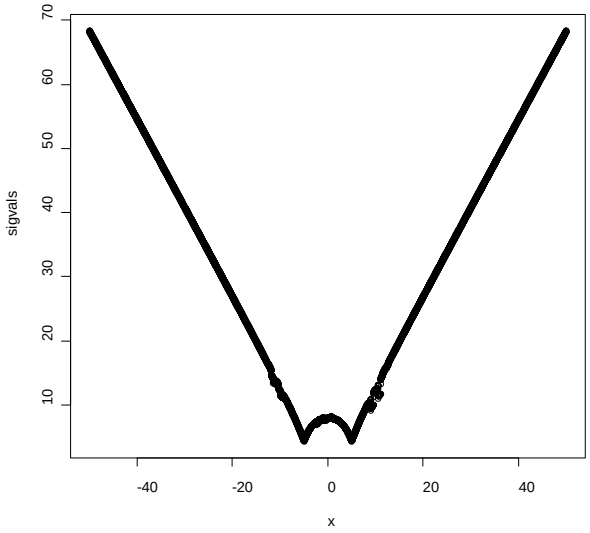


Accepted 21715 out of 100000 (21.715%).
final beta = -0.470452
alpha(0.000000) is 0.182300
alpha(2.000000) is 0.417700
alpha(5.000000) is 0.854200
alpha(8.000000) is 0.343400
alpha(10.000000) is 0.157600
alpha(13.000000) is 0.283400
alpha(15.000000) is 0.336300
alpha(20.000000) is 0.341500
alpha(30.000000) is 0.349800
alpha(50.000000) is 0.338600
it takes 123.765000 seconds to run the adaptation algorithm
it takes 54.681000 seconds to compute g and sigma
it takes 21.766000 seconds to generate the first Markov Chain
it takes 22.020000 seconds to generate the second Markov Chain
it takes 21.988000 seconds to generate the third Markov Chain
it takes 22.060000 seconds to generate the fourth Markov Chain
it takes 22.170000 seconds to generate the fifth Markov Chain
it takes 185.693000 seconds to test the local acceptance

> source("xlist5");plot(xlist[(B+1):M],type="l");varfact(xlist[(B+1):M]);sd(xlist[(B+1):M]) / sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 12.94596
[1] 0.09434861
[1] 18.22428

2.3.4.8 case 184 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 0.1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$



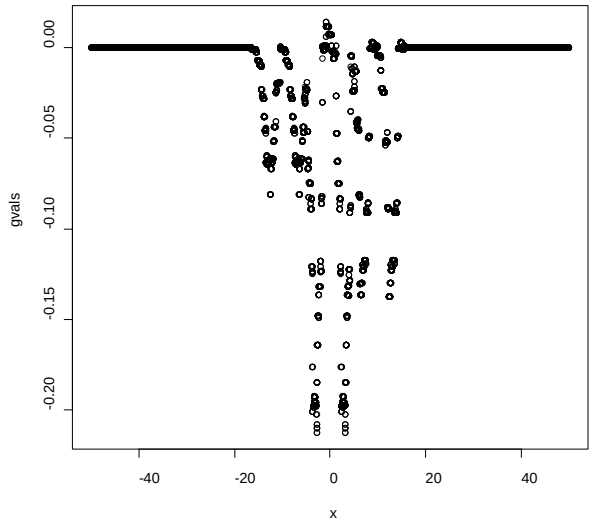


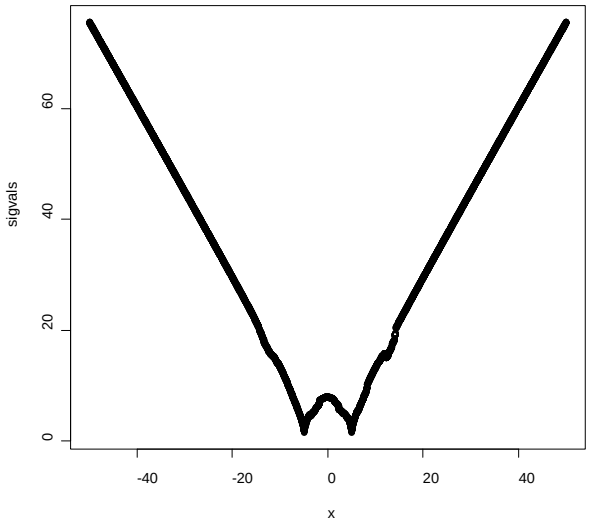
Accepted 24373 out of 100000 (24.373%).
final beta = 1.466264
alpha(0.000000) is 0.210200
alpha(2.000000) is 0.455500
alpha(5.000000) is 0.990300
alpha(8.000000) is 0.371200
alpha(10.000000) is 0.176600
alpha(13.000000) is 0.324300
alpha(15.000000) is 0.422000
alpha(20.000000) is 0.431700
alpha(30.000000) is 0.422100
alpha(50.000000) is 0.429400
it takes 124.755000 seconds to run the adaptation algorithm
it takes 55.057000 seconds to compute g and sigma
it takes 19.595000 seconds to generate the first Markov Chain
it takes 20.079000 seconds to generate the second Markov Chain
it takes 20.256000 seconds to generate the third Markov Chain
it takes 19.634000 seconds to generate the fourth Markov Chain
it takes 19.287000 seconds to generate the fifth Markov Chain

it takes 165.799000 seconds to test the local acceptance

sqrt(M-B) * sqrt(varfact(xlist[(B+1):M]));asjd(xlist[(B+1):M]);
[1] 14.67285
[1] 0.09916782
[1] 16.07759

2.3.4.9 case 185 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 2, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

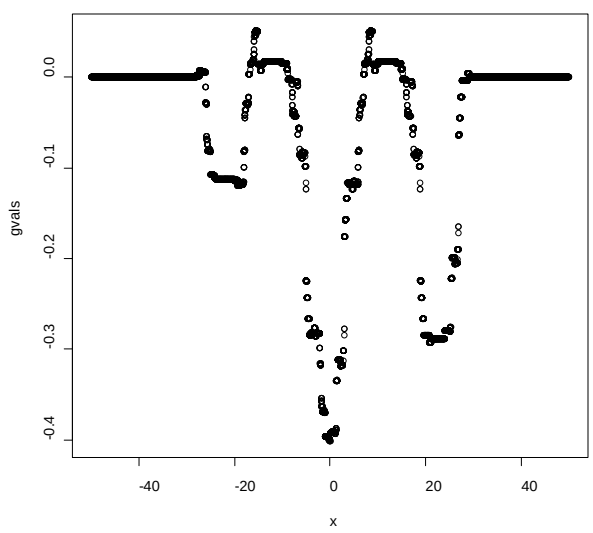


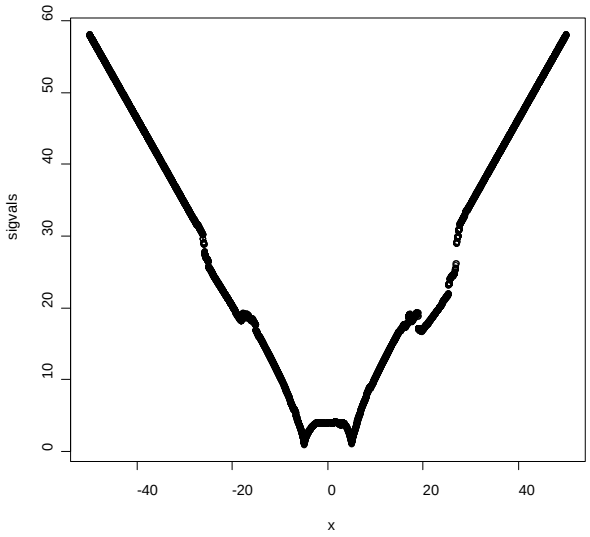


Accepted 24236 out of 100000 (24.236%).
final beta = 0.419881
alpha(0.000000) is 0.225300
alpha(2.000000) is 0.469700
alpha(5.000000) is 0.985200
alpha(8.000000) is 0.385800
alpha(10.000000) is 0.166800
alpha(13.000000) is 0.313200
alpha(15.000000) is 0.379700
alpha(20.000000) is 0.408500
alpha(30.000000) is 0.409600
alpha(50.000000) is 0.403300
it takes 124.924000 seconds to run the adaptation algorithm
it takes 55.695000 seconds to compute g and sigma
it takes 19.998000 seconds to generate the first Markov Chain
it takes 19.831000 seconds to generate the second Markov Chain
it takes 19.483000 seconds to generate the third Markov Chain
it takes 21.227000 seconds to generate the fourth Markov Chain
it takes 22.338000 seconds to generate the fifth Markov Chain
it takes 169.441000 seconds to test the local acceptance

[1] 14.55493
[1] 0.09914065
[1] 16.28342

2.3.4.10 case 186 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 8, C = 1, \gamma = 0.5$ with fixed bandwidth $b_n = 1$



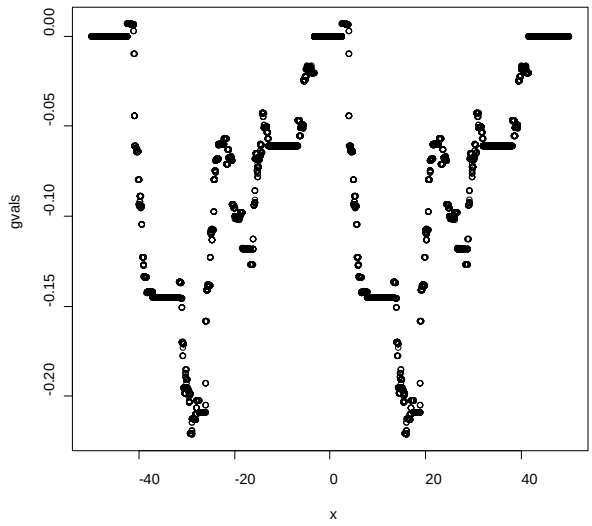


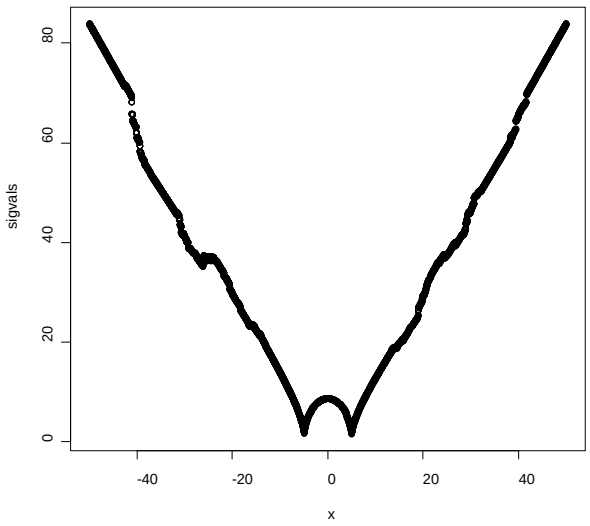
Accepted 24475 out of 100000 (24.475%).
final beta = 0.154851
alpha(0.000000) is 0.285300
alpha(2.000000) is 0.494800
alpha(5.000000) is 0.980600
alpha(8.000000) is 0.380900
alpha(10.000000) is 0.131100
alpha(13.000000) is 0.302800
alpha(15.000000) is 0.395800
alpha(20.000000) is 0.480100
alpha(30.000000) is 0.460000
alpha(50.000000) is 0.446200
it takes 124.161000 seconds to run the adaptation algorithm
it takes 55.555000 seconds to compute g and sigma
it takes 19.451000 seconds to generate the first Markov Chain
it takes 19.152000 seconds to generate the second Markov Chain
it takes 19.122000 seconds to generate the third Markov Chain
it takes 19.160000 seconds to generate the fourth Markov Chain
it takes 19.256000 seconds to generate the fifth Markov Chain
it takes 155.375000 seconds to test the local acceptance

[1] 29.62488
[1] 0.1409916
[1] 7.644844

2.3.4.11 case 187 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 15, C = 1, \gamma = 0.5$ with fixed bandwidth

$b_n = 1$



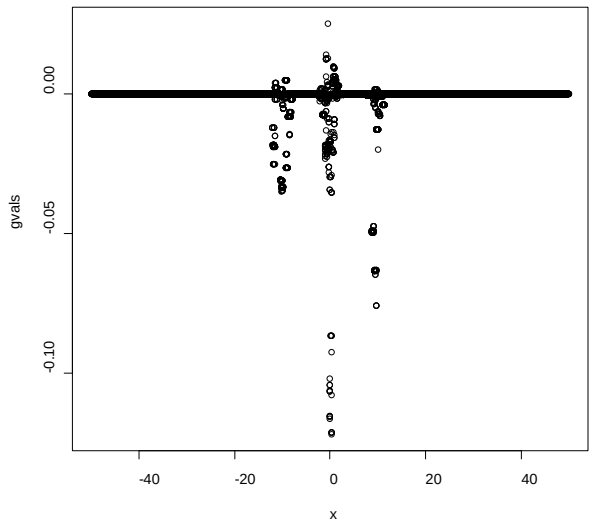


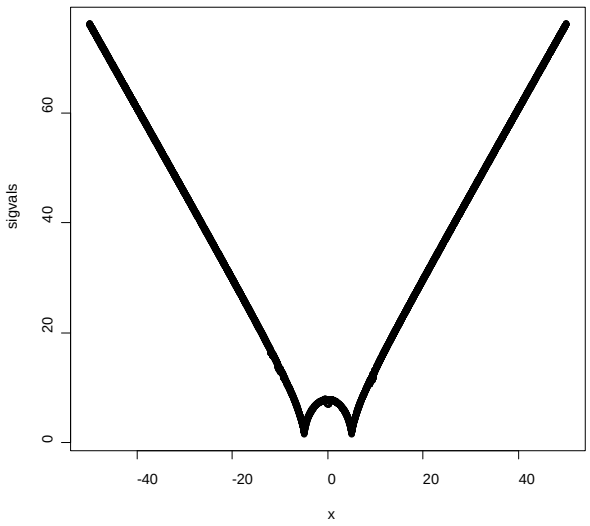
Accepted 23608 out of 100000 (23.608%).
final beta = 0.522466
alpha(0.000000) is 0.205600
alpha(2.000000) is 0.439000
alpha(5.000000) is 0.985900
alpha(8.000000) is 0.384500
alpha(10.000000) is 0.174000
alpha(13.000000) is 0.318700
alpha(15.000000) is 0.409000
alpha(20.000000) is 0.417200
alpha(30.000000) is 0.402000
alpha(50.000000) is 0.386800
it takes 125.527000 seconds to run the adaptation algorithm
it takes 56.435000 seconds to compute g and sigma
it takes 21.357000 seconds to generate the first Markov Chain
it takes 22.341000 seconds to generate the second Markov Chain
it takes 20.670000 seconds to generate the third Markov Chain
it takes 19.801000 seconds to generate the fourth Markov Chain
it takes 19.573000 seconds to generate the fifth Markov Chain
it takes 171.328000 seconds to test the local acceptance

[1] 13.02177
[1] 0.09372085
[1] 16.78987

2.3.4.12 case 188 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.1, C = 1, \gamma = 0.5$ with fixed bandwidth

$b_n = 1$



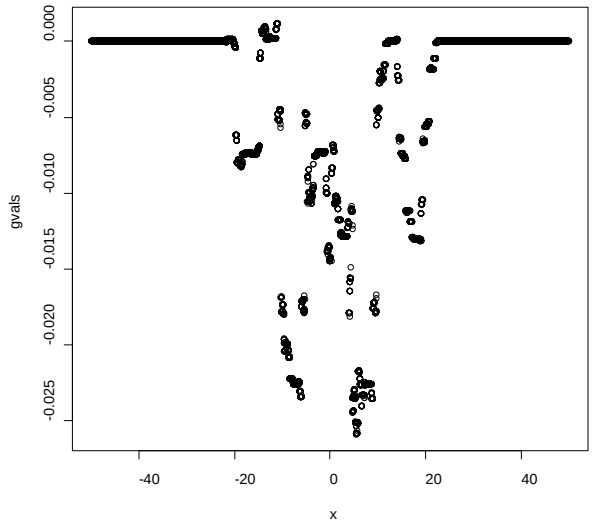


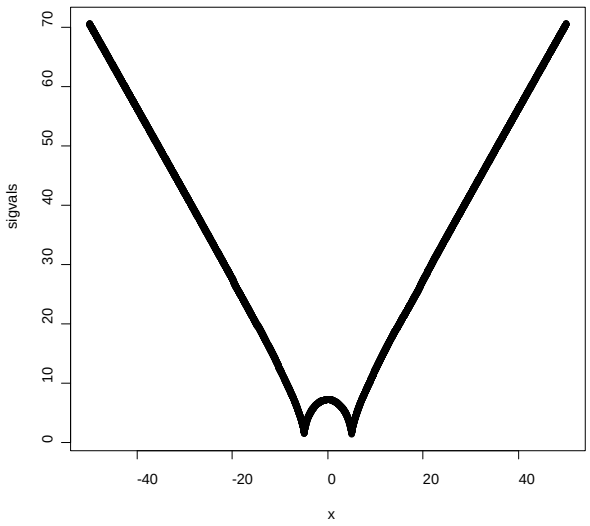
Accepted 23881 out of 100000 (23.881%).
final beta = 0.426753
alpha(0.000000) is 0.217900
alpha(2.000000) is 0.453400
alpha(5.000000) is 0.985100
alpha(8.000000) is 0.373600
alpha(10.000000) is 0.163700
alpha(13.000000) is 0.308000
alpha(15.000000) is 0.388000
alpha(20.000000) is 0.410700
alpha(30.000000) is 0.401100
alpha(50.000000) is 0.408400
it takes 126.962000 seconds to run the adaptation algorithm
it takes 56.884000 seconds to compute g and sigma
it takes 20.672000 seconds to generate the first Markov Chain
it takes 19.828000 seconds to generate the second Markov Chain
it takes 19.207000 seconds to generate the third Markov Chain
it takes 20.404000 seconds to generate the fourth Markov Chain
it takes 20.557000 seconds to generate the fifth Markov Chain
it takes 172.155000 seconds to test the local acceptance

[1] 14.18767
[1] 0.09807043
[1] 16.43629

2.3.4.13 case 190 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 1, \gamma = 0.5$ with fixed bandwidth

$b_n = 10$



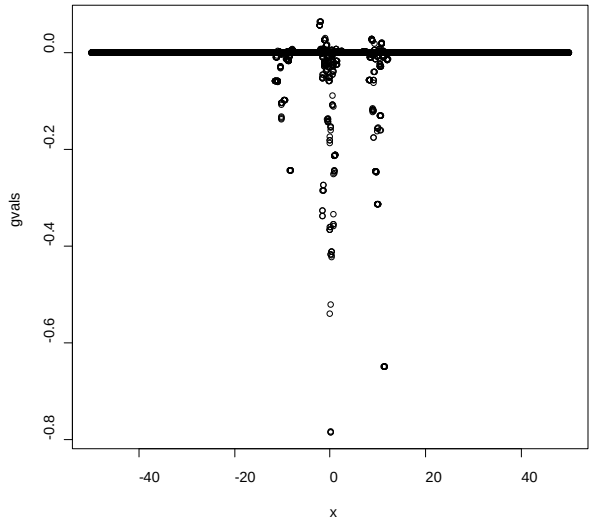


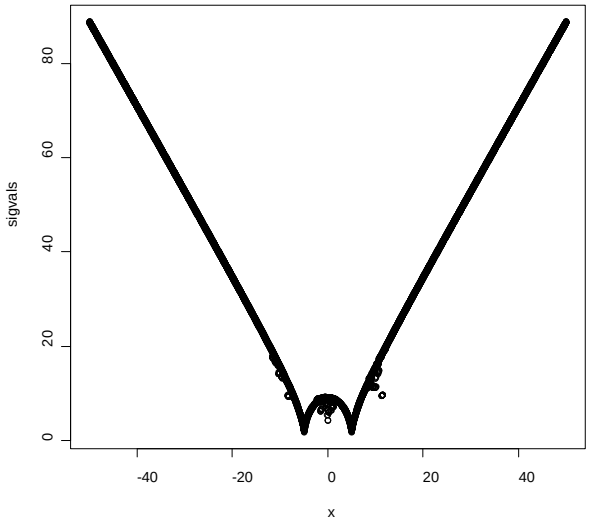
Accepted 24986 out of 100000 (24.986%).
final beta = 0.348742
alpha(0.000000) is 0.230200
alpha(2.000000) is 0.466600
alpha(5.000000) is 0.986900
alpha(8.000000) is 0.387100
alpha(10.000000) is 0.173500
alpha(13.000000) is 0.312200
alpha(15.000000) is 0.406000
alpha(20.000000) is 0.421100
alpha(30.000000) is 0.419100
alpha(50.000000) is 0.425200
it takes 128.936000 seconds to run the adaptation algorithm
it takes 58.141000 seconds to compute g and sigma
it takes 19.376000 seconds to generate the first Markov Chain
it takes 20.517000 seconds to generate the second Markov Chain
it takes 21.659000 seconds to generate the third Markov Chain
it takes 21.121000 seconds to generate the fourth Markov Chain
it takes 20.021000 seconds to generate the fifth Markov Chain
it takes 172.470000 seconds to test the local acceptance

[1] 14.28311
[1] 0.09859633
[1] 15.87459

2.3.4.14 case 191 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 1, \gamma = 0.5$ with fixed bandwidth

$b_n = 0.1$



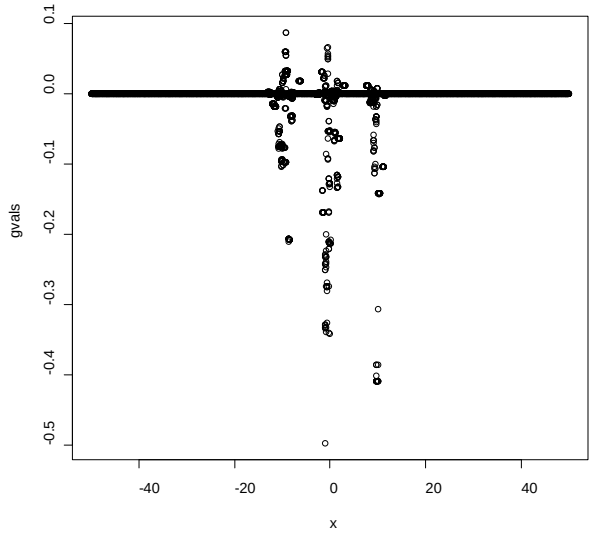


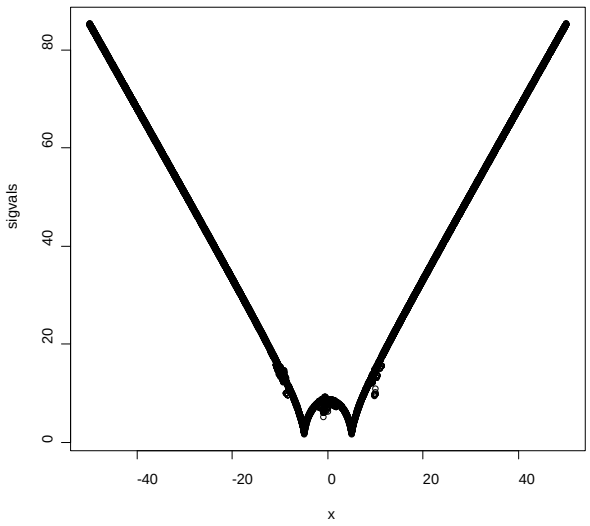
Accepted 22412 out of 100000 (22.412%).
final beta = 0.580059
alpha(0.000000) is 0.192800
alpha(2.000000) is 0.430700
alpha(5.000000) is 0.987700
alpha(8.000000) is 0.353300
alpha(10.000000) is 0.162300
alpha(13.000000) is 0.295400
alpha(15.000000) is 0.371700
alpha(20.000000) is 0.374300
alpha(30.000000) is 0.376500
alpha(50.000000) is 0.372300
it takes 129.581000 seconds to run the adaptation algorithm
it takes 57.591000 seconds to compute g and sigma
it takes 21.501000 seconds to generate the first Markov Chain
it takes 21.500000 seconds to generate the second Markov Chain
it takes 21.806000 seconds to generate the third Markov Chain
it takes 21.157000 seconds to generate the fourth Markov Chain
it takes 21.346000 seconds to generate the fifth Markov Chain
it takes 187.114000 seconds to test the local acceptance

[1] 13.28312
[1] 0.09494838
[1] 17.80852

2.3.4.15 case 192 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 0.5, C = 1, \gamma = 0.5$ with decreasing bandwidth

$b_n = \frac{1}{n^{0.2}}$



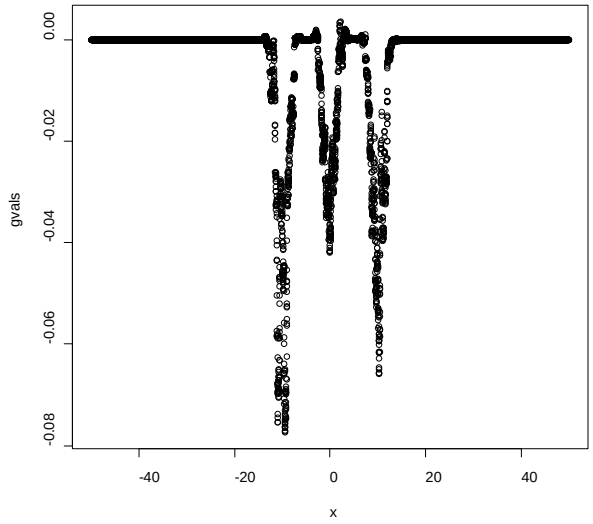


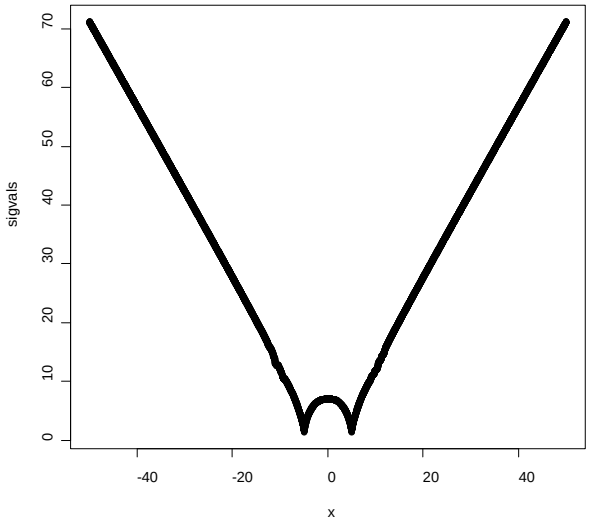
Accepted 23078 out of 100000 (23.078%).
final beta = 0.539463
alpha(0.000000) is 0.205900
alpha(2.000000) is 0.454600
alpha(5.000000) is 0.986500
alpha(8.000000) is 0.355300
alpha(10.000000) is 0.160800
alpha(13.000000) is 0.296500
alpha(15.000000) is 0.365300
alpha(20.000000) is 0.382200
alpha(30.000000) is 0.379000
alpha(50.000000) is 0.373800
it takes 129.994000 seconds to run the adaptation algorithm
it takes 57.687000 seconds to compute g and sigma
it takes 20.977000 seconds to generate the first Markov Chain
it takes 21.497000 seconds to generate the second Markov Chain
it takes 23.012000 seconds to generate the third Markov Chain
it takes 21.125000 seconds to generate the fourth Markov Chain
it takes 20.858000 seconds to generate the fifth Markov Chain
it takes 186.591000 seconds to test the local acceptance

[1] 12.70254
[1] 0.09291843
[1] 17.29666

2.3.4.16 case 193 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\text{width} = 0.5, C = 1, \gamma = 0.5$ with fixed bandwidth

$b_n = 1$





Accepted 24964 out of 100000 (24.964%).
final beta = 0.357821
alpha(0.000000) is 0.238000
alpha(2.000000) is 0.464800
alpha(5.000000) is 0.984500
alpha(8.000000) is 0.392800
alpha(10.000000) is 0.164700
alpha(13.000000) is 0.311300
alpha(15.000000) is 0.394600
alpha(20.000000) is 0.421600
alpha(30.000000) is 0.425200
alpha(50.000000) is 0.421500
it takes 124.575000 seconds to run the adaptation algorithm
it takes 55.922000 seconds to compute g and sigma
it takes 19.610000 seconds to generate the first Markov Chain
it takes 19.339000 seconds to generate the second Markov Chain
it takes 19.417000 seconds to generate the third Markov Chain
it takes 19.444000 seconds to generate the fourth Markov Chain
it takes 19.706000 seconds to generate the fifth Markov Chain
it takes 163.433000 seconds to test the local acceptance

[1] 15.27249
[1] 0.101742
[1] 15.44641

2.3.5 constant $\sigma(x) = C$

```
> source("R1： sigma=C")
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
acceptance rate = 0.2342
varfact is about 14.15845
standard error is about 0.09767042
average squared jump distance is about 15.54288
```

2.3.6 varfact comparison

kernel function $K(x)=e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	16.26846	14.57988	16.42912	86.86853	15.50366
10					18.74016	18.43029	90.07437	91.53203	90.90502
0.1					94.37575	93.97776	94.89696	93.97518	27.86239
1		0.5			15.41126	16.27198	15.48657	15.75304	16.19362
10					14.65439	15.18358	13.33354	14.90325	14.06277
0.1					13.48694	12.5845	13.65186	13.35255	12.4874
1	10				12.64651	12.96737	12.65248	13.67789	12.82574
0.1					12.92113	13.57182	14.02518	14.24815	13.62849
	0.1				13.43403	13.49145	13.51024	13.90427	13.35211
	1		10		13.31778	13.00656	12.33834	13.99267	13.74377
			0.1		17.80311	15.6759	16.52688	16.10023	17.91262
			$\frac{1}{n^{0.2}}$		13.21089	12.84258	13.86328	13.80823	14.55637
			1	$\frac{1}{(n+5)^{0.8}}$	14.00937	13.22654	12.34942	13.38838	13.34282

The best case is:

case 133 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=1,\alpha_2=1,C=10,\gamma=0.5$ with fixed bandwidth $b_n=1$

kernel function $K(x)=\frac{1}{1+\alpha_1|x|^{\alpha_2}}$

α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	83.99893	84.04238	83.67897	82.42829	13.66704
10					90.40186	89.71504	19.61744	90.32034	91.49129
0.1					86.46781	17.73197	87.82512	16.51479	88.28712
1	0.1				89.07015	9.307463	9.667522	88.64032	90.71062
	10				89.72487	89.13216	89.81371	88.89575	89.95243
	1	0.5			13.68547	13.03615	14.32362	14.0087	14.27748
10					14.44041	13.52041	13.55925	13.76832	12.43203
0.1					15.82955	15.23739	14.52674	15.13927	14.78932
10	10				13.3392	12.24885	13.38762	13.53404	13.72235
	0.1				15.14332	15.4522	14.88309	15.50852	14.91846
	10	0.1			13.4966	12.38203	14.30898	14.33035	13.31069
		1			16.47299	15.41183	15.58338	16.51207	16.61734

		0.5	10		20.44234	21.46451	19.50774	21.96715	21.61043
			0.1		17.52911	18.1089	17.64972	17.20674	17.48015
			$\frac{1}{n^{0.2}}$		17.33197	17.97494	17.86626	16.96093	17.77595
				$\frac{1}{(n+5)^{0.8}}$	13.86666	13.45143	13.59058	13.00682	12.96691

The best case is

case 148 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

kernel function $K(x) = \begin{cases} 0, \mid x \mid \geq 2 * width \\ -1 * hegith, width < \mid x \mid < 2 * width \\ 1 * height, \mid x \mid \leq width \end{cases}$

height	width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	0.5	1	0.5	1	$\frac{1}{(n+5)^{0.5}}$	17.77124	17.56957	17.15754	18.7177	17.85295
		10				14.5163	13.97646	14.53291	15.26343	13.60182
		0.1				14.47679	13.60182	12.84939	13.38512	12.99476
2		0.1				13.44326	14.87757	12.65287	13.89331	13.87151
5						14.77333	15.53957	15.15622	14.71878	14.70888
10						22.74932	21.23517	21.51277	22.1724	21.28494
0.1						13.12834	12.62207	13.73153	13.44643	13.29028
	2					12.32118	12.50984	12.6972	13.12091	13.51672
	8					13.52936	12.9048	12.80928	12.16642	13.63757
	15					15.11477	15.55721	15.568	16.00962	15.62307
	0.1					13.27735	13.54215	14.14683	13.06055	13.0853
	2		0.1			14.92786	15.62123	15.38949	16.17938	14.62442
			0.5	10		12.76971	12.85744	12.76055	12.98399	12.96732
				0.1		13.06958	13.49213	13.10409	13.37818	12.68868
				$\frac{1}{n^{0.2}}$		12.83056	12.70031	13.2164	12.35389	12.13882
				1	$\frac{1}{(n+5)^{0.8}}$	13.6103	13.25533	12.66344	12.985	12.53176

The best case is :

case 173 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with fixed bandwidth

$b_n = 10$

kernel function $K(x)=\begin{cases}0, |x|\geq 2*width \quad or \quad |x|\leq width \\ 1, width<|x|<2*width\end{cases}$

width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	1	2	1	$\frac{1}{(n+5)^{0.5}}$	21.4667	91.91048	91.56507	91.52051	91.28675
	10				83.076957	17.25225	84.51131	16.42809	17.12798
	0.1				91.2014	91.82045	91.54555	12.36587	91.63516
2	1				89.2247	19.79872	88.87049	20.77564	88.68445
0.1					12.98	13.09344	12.8819	12.88677	12.88688
0.5	1	0.5	1		12.43654	12.98527	12.4032	12.94385	12.92349
	10				13.1587	13.1709	12.11507	13.44184	12.94596
	0.1				15.58158	14.4539	15.0466	14.11616	14.67285
2	1				14.21562	14.09779	14.64326	15.09419	14.55493
8					29.03847	29.74645	30.26822	31.29497	29.62488
15					13.74205	14.75585	13.3168	13.31694	13.02177
0.1					14.67465	14.47777	14.09109	4.44332	14.18767
0.5			10		14.74344	13.17465	14.85009	13.66765	14.28311
			0.1		12.6542	13.1784	13.38414	12.82934	13.28312
			$\frac{1}{n^{0.2}}$		13.92314	13.61308	13.15229	13.74245	12.70254
			1	$\frac{1}{(n+5)^{0.8}}$	14.60853	14.70849	5.70829	15.25032	15.27249

The best case is:

case 182 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\text{width}=0.5, C=1, \gamma=0.5$ with fixed bandwidth $b_n=1$

constant $\sigma(x)=C$				
First run	Second run	Third run	Fourth run	Fifth run
14.15845	15.31856	13.73391	15.47598	14.98496

2.3.7 variance comparison

kernel function $K(x)=e^{-\frac{\lvert x\rvert^{\alpha_1}}{\alpha_2}}$									
α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	0.01353855	0.01258219	0.01373153	0.3112750	0.01314821
10					0.01458137	0.01433285	0.3162902	0.3189283	0.3148806
0.1					0.3227189	0.324662	0.3258069	0.3235092	0.01766048
1		0.5			0.1014592	0.1061646	0.1022247	0.1033099	0.1044669
10					0.1005178	0.1018522	0.0955402	0.09975748	0.096773
0.1					0.09539856	0.09189677	0.09668277	0.09525747	0.09103119
1	10				0.09292519	0.09330761	0.0922181	0.09732262	0.09345714
0.1					0.09362208	0.09749066	0.09749066	0.09815398	0.09637123
	0.1				0.09595994	0.09619656	0.09564208	0.09727296	0.09567496
	1		10		0.09477128	0.09377085	0.09106809	0.09720666	0.09702575
			0.1		0.1095804	0.1027753	0.1056355	0.1045836	0.1104789
0.1			$\frac{1}{n^{0.2}}$		0.09481027	0.09343687	0.09703812	0.0966789	0.09964849
			1	$\frac{1}{(n+5)^{0.8}}$	0.09799623	0.09448611	0.09128248	0.09492446	0.09514207

The best case is:

case 133 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=1,\alpha_2=1,C=10,\gamma=0.5$ with fixed bandwidth $b_n=1$

kernel function $K(x)=\frac{1}{1+\alpha_1\lvert x\rvert^{\alpha_2}}$									
α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	0.3070454	0.3068906	0.3052713	0.3048297	0.01207831
10					0.3162226	0.3175898	0.01470637	0.3185251	0.3193592
0.1					0.3120735	0.01419517	0.3107338	0.01337576	0.3145201
1	0.1				0.3149874	0.0102625	0.01038378	0.3151718	0.3180162
	10				0.3138015	0.3144775	0.3119324	0.3141539	0.3179866
	1	0.5			0.09629609	0.09412796	0.09878587	0.0973144	0.09757993
10					0.09918275	0.09574896	0.09617135	0.09636164	0.0918699
0.1					0.1044797	0.1014555	0.09926426	0.1026024	0.09995486
10	10				0.0948684	0.09043256	0.09575366	0.09641464	0.09650097
	0.1				0.101807	0.102022	0.1004197	0.1020636	0.1007374
	10	0.1			0.09532997	0.09173752	0.09862858	0.0989875	0.09509448
		1			0.1062357	0.1024435	0.102358	0.1059048	0.1063952
		0.5	10		0.1185137	0.1212922	0.1148601	0.1225326	0.1214091

			0.1		0.1094693	0.1108987	0.1093262	0.1075092	0.108967
			$\frac{1}{n^{0.2}}$		0.1097906	0.1107658	0.1109778	0.1064364	0.1097439
				$\frac{1}{(n+5)^{0.8}}$	0.09686088	0.09565276	0.09584571	0.09404183	0.09388677

The best case is

case 148 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

kernel function $K(x) = \begin{cases} 0, \mid x \mid \geq 2 * width \\ -1 * hegith, width < \mid x \mid < 2 * width \\ 1 * height, \mid x \mid \leq width \end{cases}$

height	width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	0.5	1	0.5	1	$\frac{1}{(n+5)^{0.5}}$	0.1096651	0.1084375	0.1067751	0.1127673	0.1088777
		10				0.0987999	0.0980836	0.0997633	0.1016252	0.0965179
		0.1				0.0983502	0.0958647	0.0936723	0.0953152	0.0938777
2		0.1				0.0952151	0.1007342	0.0924407	0.0971953	0.0967851
5						0.0995773	0.1026557	0.1013624	0.1004220	0.098975
10						0.1249001	0.1207218	0.1206259	0.1231427	0.1196420
0.1						0.0943379	0.0925746	0.0955971	0.0950350	0.0946926
	2					0.0906348	0.0923965	0.0928345	0.0951008	0.0956070
	8					0.0966672	0.0930716	0.0933293	0.0910570	0.0965149
	15					0.100082	0.102526	0.1019249	0.1035333	0.1033918
	0.1					0.0940607	0.0956745	0.0976864	0.0942801	0.0941309
	2		0.1			0.1004776	0.1030413	0.1023623	0.1049767	0.0996596
			0.5	10		0.0929315	0.0933425	0.0922779	0.0940506	0.0933517
				0.1		0.0942665	0.0954377	0.0946791	0.095681	0.0922989
				$\frac{1}{n^{0.2}}$		0.0930883	0.0931727	0.093848	0.091052	0.090727
				1	$\frac{1}{(n+5)^{0.8}}$	0.096987	0.095062	0.093127	0.093600	0.091170

The best case is:

case 175 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with decreasing bandwidth $b_n = \frac{1}{n^{0.2}}$

kernel function $K(x)=\begin{cases}0, |x|\geq 2*width \quad or \quad |x|\leq width \\ 1, width<|x|<2*width\end{cases}$

width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	1	2	1	$\frac{1}{(n+5)^{0.5}}$	0.01560605	0.3208122	0.3201209	0.3172555	0.3161601
	10				0.3043341	0.01413039	0.3057916	0.01335597	0.01378514
	0.1				0.319614	0.3210132	0.3201003	0.01201751	0.3196954
2	1				0.316767	0.01488253	0.3156092	0.01514421	0.3080952
0.1					0.09400533	0.09445313	0.09332865	0.09327825	0.09355521
0.5	1	0.5	1		0.09160858	0.09325567	0.09206486	0.09338538	0.09353589
	10				0.0943911	0.09383944	0.09069657	0.0952836	0.09434861
	0.1				0.102896	0.09876851	0.1006781	0.09787487	0.09916782
2	1				0.09793323	0.09871597	0.09948697	0.1016156	0.09914065
8					0.1399522	0.1428308	0.1440682	0.1472225	0.1409916
15					0.09698082	0.1003973	0.09533199	0.09510083	0.09372085
0.1					0.09981102	0.09970097	0.09808324	0.09874253	0.09807043
0.5			10		0.1004022	0.09453277	0.1004871	0.09610355	0.09859633
			0.1		0.0927094	0.09423476	0.09581283	0.09348766	0.09494838
			$\frac{1}{n^{0.2}}$		0.09759395	0.09547936	0.09430047	0.09667705	0.09291843
			1	$\frac{1}{(n+5)^{0.8}}$	0.09948617	0.09938801	0.1030033	0.1019863	0.101742

The best case is:

case 182 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\text{width}=0.5, C=1, \gamma=0.5$ with fixed bandwidth $b_n=1$

constant $\sigma(x)=C$				
First run	Second run	Third run	Fourth run	Fifth run
0.09767042	0.1011924	0.09713788	0.09842107	0.1022126

2.3.8 comparison of average jump distance

kernel function $K(x)=e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	0.2751228	0.2759929	0.2706779	1.680425	0.2709634
10					0.2420872	0.2420623	1.300179	1.100248	1.137486
0.1					0.7802203	0.8506236	0.75010	0.8316456	0.1633702
1		0.5			14.88102	14.89564	14.8591	15.10428	14.77388
10					16.15489	15.7693	16.31433	16.00171	15.95296
0.1					17.58205	17.69769	17.8748	17.63134	17.75754
1	10				17.64668	17.55073	17.72917	17.68229	17.58393
0.1	10				17.24616	17.43554	17.11712	17.62766	17.20509
	0.1				17.65325	17.40023	17.13826	17.25142	17.47388
			10		17.16678	17.39617	17.29623	17.32788	17.61989
			0.1		14.17684	14.39708	14.29359	14.24914	14.03917
			$\frac{1}{n^{0.2}}$		17.09166	17.4916	17.58568	17.48781	16.96503
				$\frac{1}{(n+5)^{0.8}}$	17.29917	17.58498	17.3033	17.30672	17.27245

The best case is:

case 133 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\alpha_1=1,\alpha_2=1,C=10,\gamma=0.5$ with fixed bandwidth $b_n=1$

kernel function $K(x)=\frac{1}{1+\alpha_1|x|^{\alpha_2}}$

α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	1	2	1	$\frac{1}{(n+5)^{0.5}}$	2.032223	2.016114	2.033322	2.244812	0.3014526
10					1.229918	1.441011	0.2134324	1.313225	1.163729
0.1					1.775863	0.2643402	1.454854	0.2695992	1.458424
1	0.1				1.453122	0.4300739	0.4270017	1.449158	1.260599
	10				1.264062	1.348775	1.324255	1.389254	1.358456
10	1	0.5			16.8956	17.15056	16.62733	16.56032	16.54765
					16.91554	17.05386	17.0554	17.05604	17.23872
0.1					15.85114	16.04918	16.2265	16.07591	16.03785
10	10				17.2243	17.46561	17.20456	17.48782	17.49019
	0.1				16.09596	15.91488	16.15975	15.87687	15.86291
	10	0.1			17.01713	17.33785	16.88911	17.58669	17.21339
		1			14.94124	14.47357	14.76298	14.378	14.43249
		0.5	10		11.12058	11.09729	11.1061	11.01476	10.9672

			0.1		13.53036	13.3978	13.39257	13.43201	13.33216
			$\frac{1}{n^{0.2}}$		13.75155	13.59613	13.75362	13.54406	13.55263
				$\frac{1}{(n+5)^{0.8}}$	17.50773	17.1296	16.98451	17.70879	17.26632

The best case is

case 148 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 10, \alpha_2 = 1, C = 10, \gamma = 0.5$ with fixed bandwidth $b_n = 1$

kernel function $K(x) = \begin{cases} 0, \mid x \mid \geq 2 * width \\ -1 * hegith, width < \mid x \mid < 2 * width \\ 1 * height, \mid x \mid \leq width \end{cases}$

height	width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	0.5	1	0.5	1	$\frac{1}{(n+5)^{0.5}}$	13.26237	13.30066	13.43912	13.43278	13.29712
		10				15.89375	16.29958	16.33701	16.20252	16.46537
		0.1				17.25006	17.29043	17.44449	17.52067	17.25407
2		0.1				16.61506	16.81390	16.96739	17.08910	17.20643
5						15.34476	15.23875	14.98588	15.59716	14.76257
10						12.40918	12.45344	12.24459	12.17052	12.15185
0.1						17.56846	17.56847	17.56094	17.46663	17.58271
	2					17.96944	18.08717	18.11135	17.80244	17.78677
	8					17.80607	17.92915	17.32039	18.16933	17.84017
	15					14.65539	14.81321	14.85105	15.0683	14.97829
	0.1					16.80683	16.78677	16.69167	16.58419	16.67941
	2		0.1			15.7515	15.63679	15.41598	15.38731	15.71839
			0.5	10		18.04347	18.16659	17.67161	17.90958	17.73759
				0.1		17.56144	17.92892	17.91349	17.91258	17.97203
				$\frac{1}{n^{0.2}}$		17.79253	18.07391	17.8165	18.141	18.01603
				1	$\frac{1}{(n+5)^{0.8}}$	17.93693	17.97735	17.67513	17.42695	17.39284

The best case is :

case 173 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{height} = 0.1, \text{width} = 2, C = 0.1, \gamma = 0.5$ with fixed bandwidth

$b_n = 10$

kernel function $K(x)=\begin{cases}0, |x|\geq 2*width \quad or \quad |x|\leq width \\ 1, width<|x|<2*width\end{cases}$

width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	1	2	1	$\frac{1}{(n+5)^{0.5}}$	0.2160827	1.134826	1.111249	1.120923	1.140038
	10				2.037578	0.2655736	2.010462	0.2639856	0.2614159
	0.1				1.176037	1.131612	1.176397	0.3714259	1.21477
2	1				1.329289	0.2200012	1.344635	0.2176641	1.326619
0.1					17.5467	17.79094	18.01757	17.60527	17.49032
0.5	1	0.5	1		17.93946	17.42754	17.92938	17.68764	17.78533
	10				17.75638	17.64809	17.8715	17.66988	18.22428
	0.1				15.48933	15.75263	15.99588	16.07236	16.07759
2	1				16.3643	16.21187	16.19518	16.21535	16.28342
8					7.740157	7.842474	7.7877	7.708015	7.644844
15					16.73765	16.26973	16.97077	7.01989	16.78987
0.1					16.32375	16.6559	16.6806	16.37162	16.43629
0.5			10		15.59574	15.70382	15.66388	15.9781	15.87459
			0.1		17.38151	17.41642	17.40737	17.41414	17.80852
			$\frac{1}{n^{0.2}}$		16.799	16.79686	17.50548	16.96549	17.29666
			1	$\frac{1}{(n+5)^{0.8}}$	5.57927	15.45931	15.21297	15.65399	15.44641

The best case is:

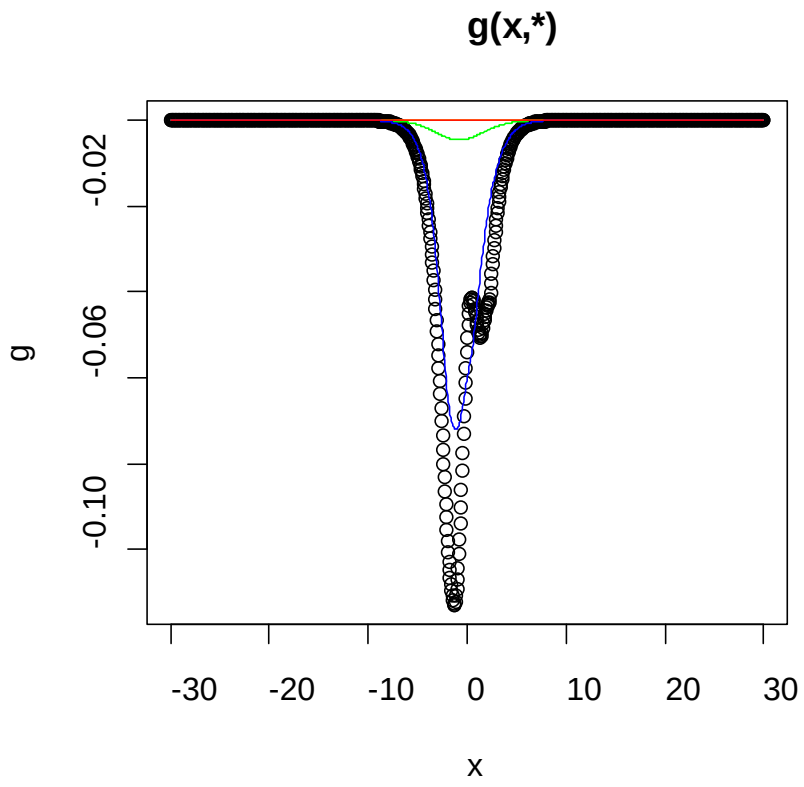
case 182 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $\text{width}=0.5, C=1, \gamma=0.5$ with fixed bandwidth $b_n=1$

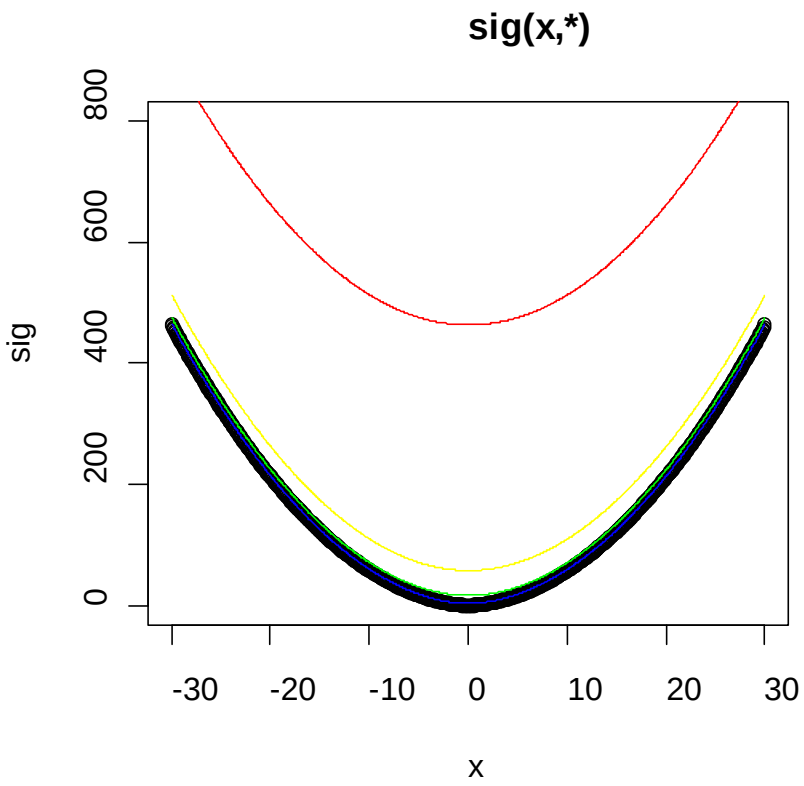
constant $\sigma(x)=C$				
First run	Second run	Third run	Fourth run	Fifth run
15.54288	15.35994	15.284	15.54276	15.54288

2.4 Example 4: normal distribution in R^2

2.4.1 kernel function $K(x) = e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

2.4.1.1 case 194 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$



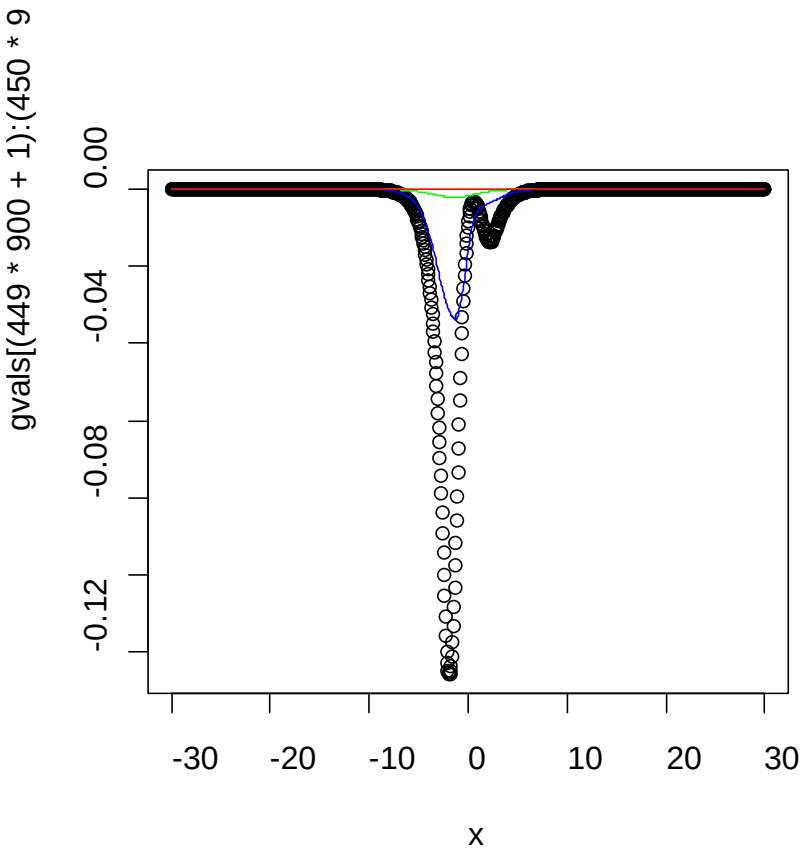


Without

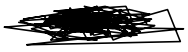
approximation:
alpha(0.000000,0.000000) is about 0.758800
alpha(0.000000,2.000000) is about 0.138100
alpha(5.000000,5.000000) is about 0.024000
alpha(2.000000,10.000000) is about 0.015000
alpha(10.000000,10.000000) is about 0.007900
alpha(20.000000,50.000000) is about 0.000700
alpha(50.000000,50.000000) is about 0.000200
it takes 296.759000 seconds to run the adaptation algorithm
it takes 10001.982000 seconds to plot g and sigma
it takes 3674.517000 seconds to generate the first Markov Chain
it takes 3806.229000 seconds to generate the second Markov Chain
it takes 3813.156000 seconds to generate the third Markov Chain
it takes 3638.899000 seconds to generate the fourth Markov Chain
it takes 3640.485000 seconds to generate the fifth Markov Chain
it takes 2712.386000 seconds to test the local acceptance

```
> source("xlist3");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 17.56787
[1] 0.01787614
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4192436
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 15.05131
[1] 0.01619204
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4235149
> source("xlist4");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 16.71129
[1] 0.01723975
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4241305
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 16.05023
[1] 0.0166278
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4103297
> source("xlist5");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 15.7971
[1] 0.01652789
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4156999
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 15.84213
[1] 0.01660798
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4247686
```

With approximation:

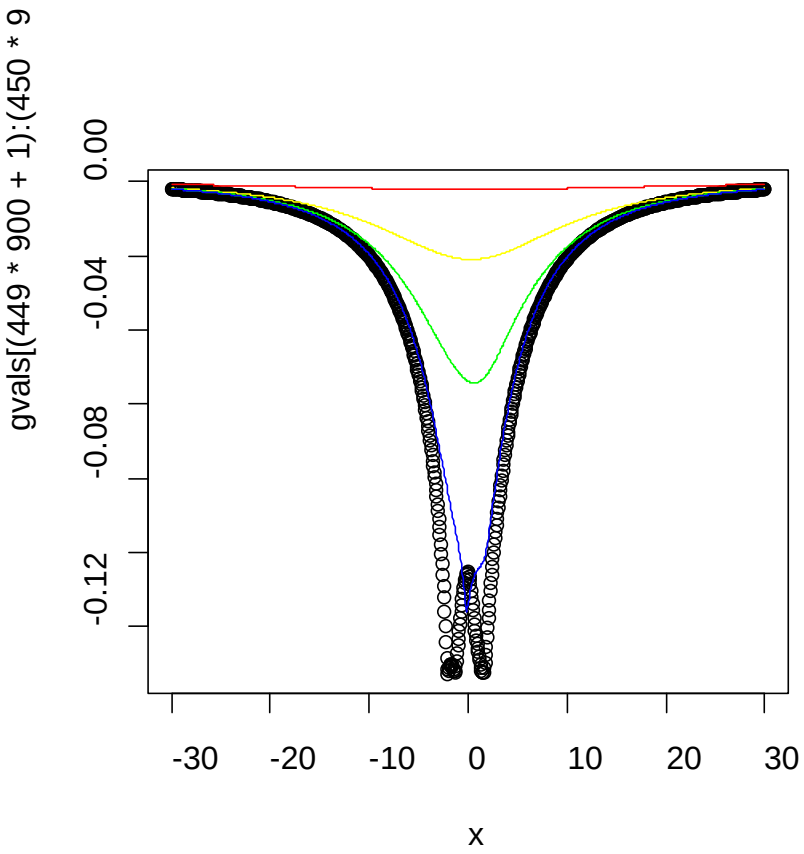


alpha(0.000000,0.000000) is about 0.677500
alpha(0.000000,2.000000) is about 0.137900
alpha(5.000000,5.000000) is about 0.025100
alpha(2.000000,10.000000) is about 0.011400
alpha(10.000000,10.000000) is about 0.007900
alpha(20.000000,50.000000) is about 0.000900
alpha(50.000000,50.000000) is about 0.000400
it takes 310.314000 seconds to run the adaptation algorithm
it takes 123.634000 seconds to compute g and sigma
it takes 9939.710000 seconds to plot g and sigma
it takes 512.742000 seconds to generate the first Markov Chain
it takes 471.932000 seconds to generate the second Markov Chain
it takes 496.096000 seconds to generate the third Markov Chain
it takes 514.006000 seconds to generate the fourth Markov Chain
it takes 468.624000 seconds to generate the fifth Markov Chain
it takes 2030.625000 seconds to test the local acceptance



```
> source("xlist3");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 17.83584
[1] 0.01795715
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.37771
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 18.94229
[1] 0.01861971
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.3819818
> source("xlist4");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 19.21391
[1] 0.0187782
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.3715975
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 17.64572
[1] 0.01804834
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.382282
> source("xlist5");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 18.30911
[1] 0.01796832
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.3769988
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 18.34555
[1] 0.01780736
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.3663442
```

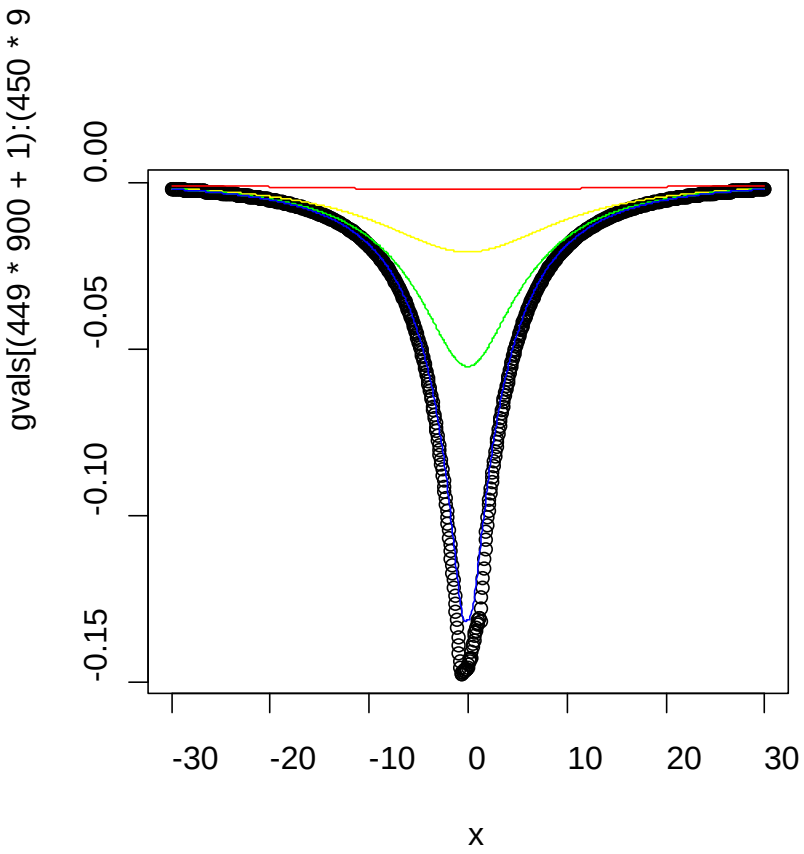
2.4.1.2 case 195 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.5, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$



alpha(0.000000,0.000000) is about 0.743500
alpha(0.000000,2.000000) is about 0.140300
alpha(5.000000,5.000000) is about 0.025400
alpha(2.000000,10.000000) is about 0.013000
alpha(10.000000,10.000000) is about 0.008200
alpha(20.000000,50.000000) is about 0.000200
alpha(50.000000,50.000000) is about 0.000400
it takes 697.040000 seconds to run the adaptation algorithm
it takes 278.120000 seconds to compute g and sigma
it takes 1394.863520 seconds to plot g and sigma
it takes 2056.040000 seconds to generate the first Markov Chain
it takes 2026.420000 seconds to generate the second Markov Chain
it takes 2127.810000 seconds to generate the third Markov Chain
it takes 2047.300000 seconds to generate the fourth Markov Chain
it takes 2019.330000 seconds to generate the fifth Markov Chain
it takes 775.512704 seconds to test the local acceptance

```
> source("xlist3");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 16.17414
[1] 0.01657182
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.3948022
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 17.14889
[1] 0.01751794
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.3982419
> source("xlist4");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 18.61931
[1] 0.01846755
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.3851333
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 16.61003
[1] 0.01718756
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.3953794
> source("xlist5");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 14.81985
[1] 0.01588587
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4055987
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 15.54527
[1] 0.01654854
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4082785
```

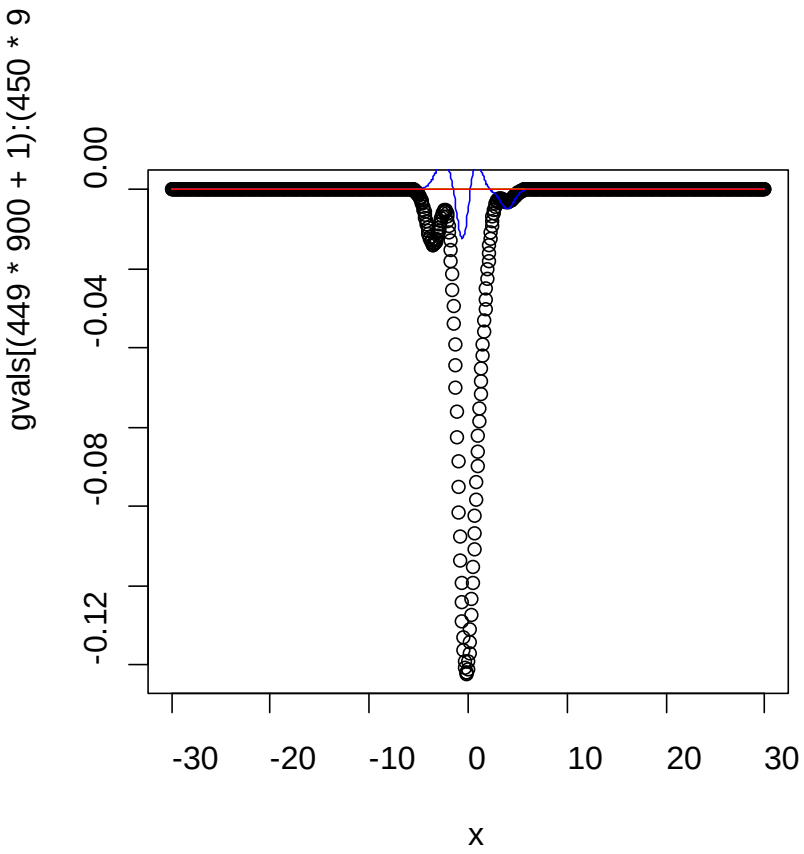
2.4.1.3 case 196 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 0.5, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$



alpha(0.000000,0.000000) is about 0.215700
alpha(0.000000,2.000000) is about 0.285000
alpha(5.000000,5.000000) is about 0.328900
alpha(2.000000,10.000000) is about 0.335800
alpha(10.000000,10.000000) is about 0.319200
alpha(20.000000,50.000000) is about 0.157600
alpha(50.000000,50.000000) is about 0.124300
it takes 308.418000 seconds to run the adaptation algorithm
it takes 120.640000 seconds to compute g and sigma
it takes 9777.420000 seconds to plot g and sigma
it takes 233.015000 seconds to generate the first Markov Chain
it takes 236.232000 seconds to generate the second Markov Chain
it takes 230.846000 seconds to generate the third Markov Chain
it takes 233.644000 seconds to generate the fourth Markov Chain
it takes 234.809000 seconds to generate the fifth Markov Chain
it takes 1859.186000 seconds to test the local acceptance

```
> source("xlist3");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.026836
[1] 0.009977178
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4939001
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.258773
[1] 0.01015497
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4959621
> source("xlist4");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.63622
[1] 0.01044986
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4918952
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.127733
[1] 0.01004813
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4806261
> source("xlist5");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 7.844774
[1] 0.009936305
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.491927
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.082055
[1] 0.01011846
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4961528
>
```

2.4.1.4 case 197 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 2, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 1$

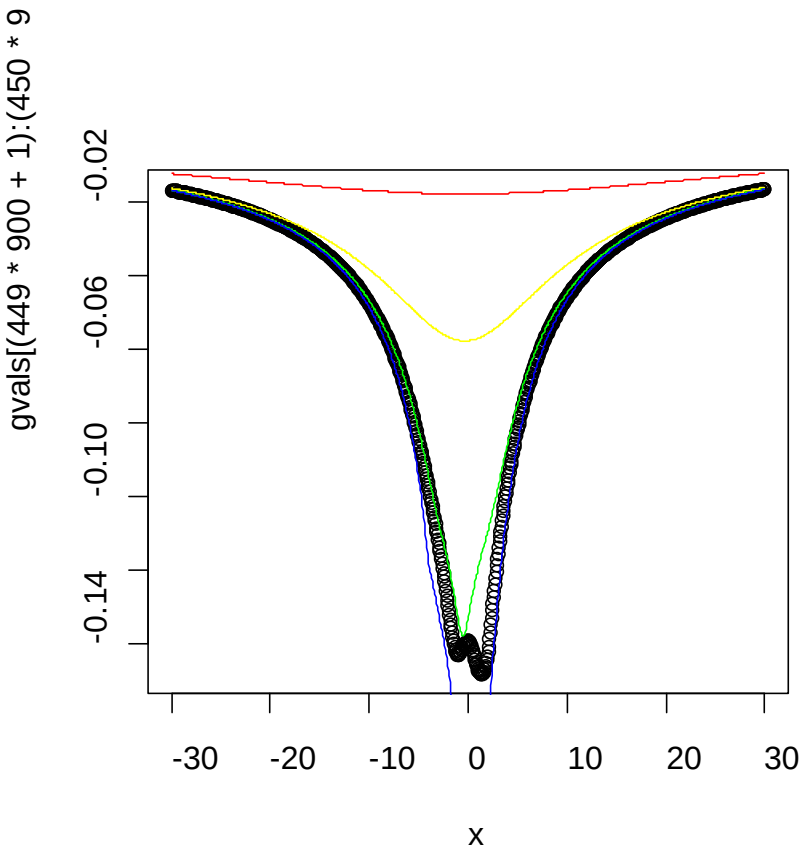


alpha(0.000000,0.000000) is about 0.213600
alpha(0.000000,2.000000) is about 0.263100
alpha(5.000000,5.000000) is about 0.324800
alpha(2.000000,10.000000) is about 0.329200
alpha(10.000000,10.000000) is about 0.324500
alpha(20.000000,50.000000) is about 0.155400
alpha(50.000000,50.000000) is about 0.121100
it takes 692.970000 seconds to run the adaptation algorithm
it takes 275.750000 seconds to compute g and sigma
it takes -1745.982960 seconds to plot g and sigma
it takes 1540.860000 seconds to generate the first Markov Chain
it takes 1553.610000 seconds to generate the second Markov Chain
it takes 1544.950000 seconds to generate the third Markov Chain
it takes 1550.080000 seconds to generate the fourth Markov Chain
it takes 1543.870000 seconds to generate the fifth Markov Chain
it takes 1162.082704 seconds to test the local acceptance

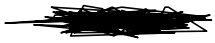
```
> source("xlist3");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.949513
[1] 0.01073386
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4865919
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.598656
[1] 0.01047538
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4712849
> source("xlist4");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.588107
[1] 0.01039507
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4599503
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.953894
[1] 0.01072307
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4720166
> source("xlist5");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.759423
[1] 0.01047106
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4650114
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 7.537465
[1] 0.009703281
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.480265
>
```

2.4.2 kernel function $K(x) = \frac{1}{1+\alpha_1 |x|^{\alpha_2}}$

2.4.2.1 case 198 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 1, \gamma = 2$ with fixed bandwidth $b_n = 1$

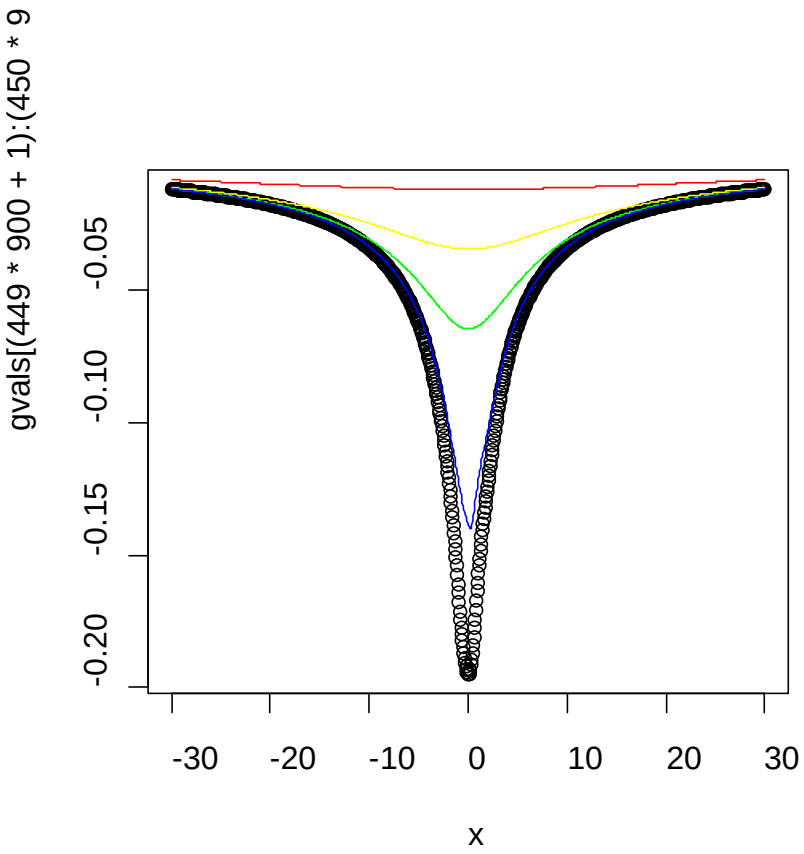


alpha(0.000000,0.000000) is about 0.812400
alpha(0.000000,2.000000) is about 0.183100
alpha(5.000000,5.000000) is about 0.032400
alpha(2.000000,10.000000) is about 0.016800
alpha(10.000000,10.000000) is about 0.010700
alpha(20.000000,50.000000) is about 0.000800
alpha(50.000000,50.000000) is about 0.000600
it takes 265.875000 seconds to run the adaptation algorithm
it takes 104.690000 seconds to compute g and sigma
it takes 8658.581000 seconds to plot g and sigma
it takes 358.066000 seconds to generate the first Markov Chain
it takes 358.640000 seconds to generate the second Markov Chain
it takes 385.815000 seconds to generate the third Markov Chain
it takes 362.398000 seconds to generate the fourth Markov Chain
it takes 385.542000 seconds to generate the fifth Markov Chain
it takes 1678.442000 seconds to test the local acceptance



```
> source("xlist3");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 12.73097
[1] 0.01494266
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4921363
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 13.90548
[1] 0.01589283
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.5000742
> source("xlist4");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 11.93283
[1] 0.01434084
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4873498
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 14.43801
[1] 0.01607662
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.481112
> source("xlist5");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 12.03846
[1] 0.01445098
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4984666
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 13.71947
[1] 0.01538246
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4771416
```

2.4.2.2 case 199 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 1$ with fixed bandwidth $b_n = 1$



alpha(0.000000,0.000000) is about 0.180900
alpha(0.000000,2.000000) is about 0.282200
alpha(5.000000,5.000000) is about 0.399500
alpha(2.000000,10.000000) is about 0.409400
alpha(10.000000,10.000000) is about 0.430200
alpha(20.000000,50.000000) is about 0.451200
alpha(50.000000,50.000000) is about 0.446700
it takes 269.569000 seconds to run the adaptation algorithm
it takes 106.705000 seconds to compute g and sigma
it takes 8641.609000 seconds to plot g and sigma
it takes 234.730000 seconds to generate the first Markov Chain
it takes 230.138000 seconds to generate the second Markov Chain
it takes 232.516000 seconds to generate the third Markov Chain
it takes 234.278000 seconds to generate the fourth Markov Chain
it takes 233.943000 seconds to generate the fifth Markov Chain
it takes 1645.604000 seconds to test the local acceptance

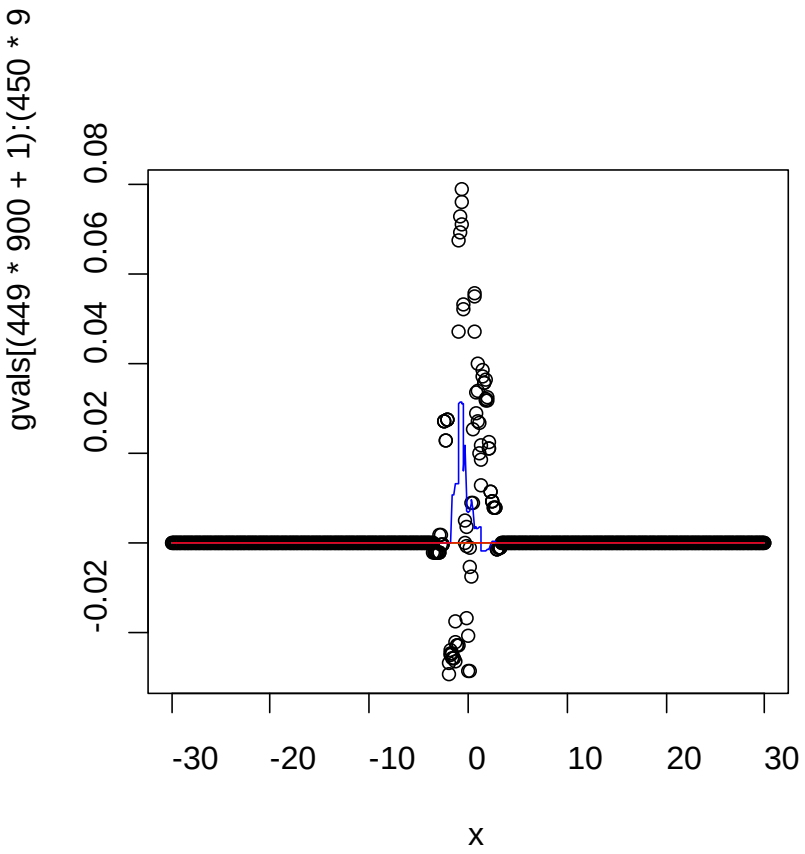
```
> source("xlist3");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 7.61042
[1] 0.009621487
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4751286
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.295657
[1] 0.009891014
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4542424
> source("xlist4");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.137364
[1] 0.009998282
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4802455
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.042155
[1] 0.009880959
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4680676
> source("xlist5");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.286704
[1] 0.009931457
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4713538
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.348549
[1] 0.01007295
```

2.4.2.3 case 200 $\eta_n = \frac{1}{(n+5)^{0.8}}$, $\alpha_1 = 1, \alpha_2 = 1, C = 0.1, \gamma = 2$ with fixed bandwidth $b_n = 10$

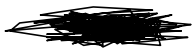
```
> source("xlist3");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.845103
[1] 0.0103805
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.45205
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.806404
[1] 0.0105059
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4572161
> source("xlist4");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 9.134059
[1] 0.01065294
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4562965
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.585028
[1] 0.0103154
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4593464
> source("xlist5");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.98815
[1] 0.01067445
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4683815
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.692522
[1] 0.01038471
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4767837
```

2.4.3 kernel function $K(x)=\begin{cases}0,|x|\geq 2*width\\ -1*hegith, width<|x|<2*width\\ 1*height,|x|\leq width\end{cases}$

2.4.3.1 case 201 $\eta_n=\frac{1}{(n+5)^{0.5}}$, $height=0.5, \text{width}=0.5, C=0.01, \gamma=10$ with fixed bandwidth $b_n=1$



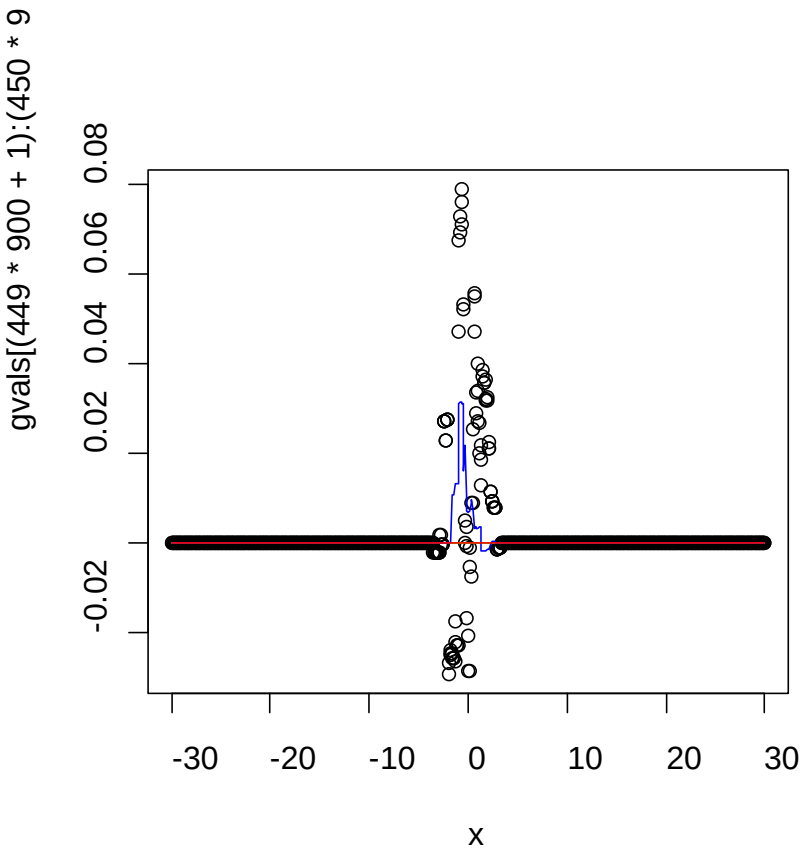
alpha(0.000000,0.000000) is about 0.172500
alpha(0.000000,0.000000) is about 0.168700
alpha(5.000000,5.000000) is about 0.401600
alpha(10.000000,10.000000) is about 0.395500
alpha(20.000000,50.000000) is about 0.054200
it takes 205.869000 seconds to run the adaptation algorithm
it takes 79.808000 seconds to compute g and sigma
it takes -1.#QNAN0 seconds to plot g and sigma
it takes 198.873000 seconds to generate the first Markov Chain
it takes 204.658000 seconds to generate the second Markov Chain
it takes 202.383000 seconds to generate the third Markov Chain
it takes 199.687000 seconds to generate the fourth Markov Chain
it takes 198.928000 seconds to generate the fifth Markov Chain
it takes 756.626000 seconds to test the local acceptance



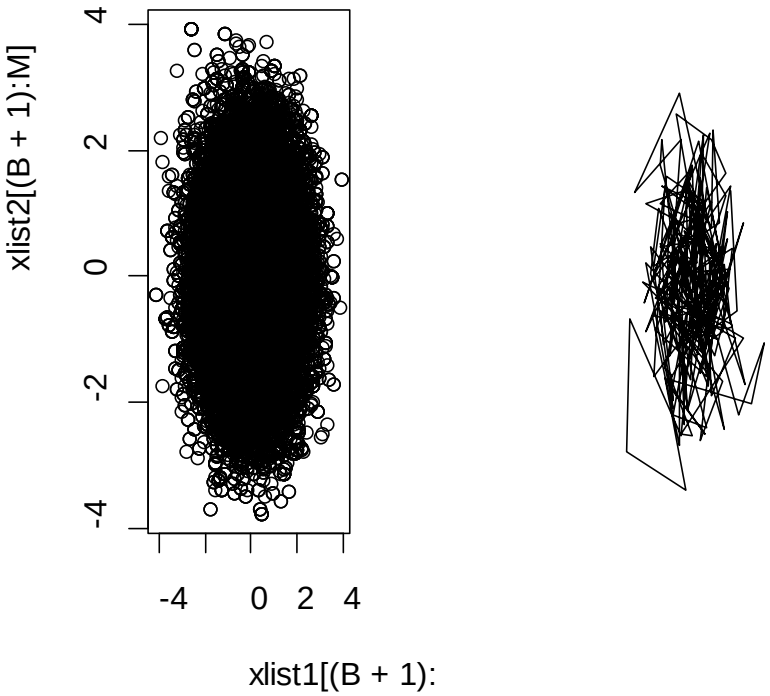
```
> source("xlist3");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.50804
[1] 0.009935555
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4494896
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.373991
[1] 0.00991694
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4568538
> source("xlist4");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.242958
[1] 0.009808818
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4652114
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.407656
[1] 0.00984394
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4524555
> source("xlist5");plot(xlist1[(B+1):M],xlist2[(B+1):M],type="l");
> varfact(xlist1[(B+1):M]);sd(xlist1[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist1[(B+1):M]) );
[1] 8.01933
[1] 0.009759364
> val = 0;for(i in (B+1):M ) val = val +(xlist1[i]-xlist1[i-1])^2;(val/(M-B));
[1] 0.4576324
> varfact(xlist2[(B+1):M]);sd(xlist2[(B+1):M]) / sqrt(M-B) * sqrt( varfact(xlist2[(B+1):M]) );
[1] 8.055316
[1] 0.009937325
> val = 0;for(i in (B+1):M ) val = val +(xlist2[i]-xlist2[i-1])^2;(val/(M-B));
[1] 0.4613729
```

2.4.4 kernel function $K(x) = \begin{cases} 0, & |x| \geq 2 * width \quad or \quad |x| \leq width \\ 1, & width < |x| < 2 * width \end{cases}$

2.4.4.1 case 202 $\eta_n = \frac{1}{(n+5)^{0.5}}$, $\text{width} = 5, C = 0.01, \gamma = 2$ with fixed bandwidth $b_n = 1$



alpha(0.000000,0.000000) is about 0.127900
alpha(0.000000,2.000000) is about 0.276100
alpha(5.000000,5.000000) is about 0.466900
alpha(2.000000,10.000000) is about 0.458900
alpha(10.000000,10.000000) is about 0.461400
alpha(20.000000,50.000000) is about 0.471300
alpha(50.000000,50.000000) is about 0.475300
it takes 203.986000 seconds to run the adaptation algorithm
it takes 79.703000 seconds to compute g and sigma
it takes -1.#QNAN0 seconds to plot g and sigma
it takes 236.103000 seconds to generate the first Markov Chain
it takes 234.442000 seconds to generate the second Markov Chain
it takes 234.676000 seconds to generate the third Markov Chain
it takes 236.654000 seconds to generate the fourth Markov Chain
it takes 237.002000 seconds to generate the fifth Markov Chain
it takes 1278.837000 seconds to test the local acceptance


xlist2[(B + 1):M]
4
2
0
-2
-4
-4
0
2
4
xlist1[(B + 1):

acceptance rate = 0.23388
varfact is about 8.07853 and 8.904434
standard error is about 0.00939504 and 0.009958031
average squared jump distance is about 0.4340354 and 0.4327815
> source("R1 sigma=C")
ran Metropolis algorithm for 1e+05 iterations, with burn-in 10000
acceptance rate = 0.23235
varfact is about 8.492082 and 8.658637
standard error is about 0.009631347 and 0.009772716
average squared jump distance is about 0.4279432 and 0.4182256

2.4.6 varfact comparison

Kernel function : $K(x)=e^{-\frac{\lvert x\rvert^{\alpha_1}}{\alpha_2}}$									
α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	0.1	2	1	$\frac{1}{(n+5)^{0.5}}$	8.011433	8.120745	8.026836	8.63622	7.844774
					8.390189	8.684449	8.258773	8.127733	8.08205

Kernel function : $K(x)=\frac{1}{1+\alpha_1 \lvert x\rvert^{\alpha_2}}$									
α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	0.1	1	1	$\frac{1}{(n+5)^{0.5}}$	9.01875	8.05305	7.61042	8.137364	8.28670
					7.950148	8.35461	8.295657	8.042155	8.348549

Kernel function : $K(x)=\begin{cases}0, \lvert x\rvert \geq 2*width \\ -1*hegith, width<\lvert x\rvert<2*width \\ 1*height, \lvert x\rvert \leq width\end{cases}$										
height	width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	0.5	0.01	10	1	$\frac{1}{(n+5)^{0.5}}$	7.741678	7.79554	8.50804	8.24295	8.0193
						8.519492	7.76981	8.373991	8.407656	8.05531

Kernel function : $K(x)=\begin{cases}0, \lvert x\rvert \geq 2*width \quad or \quad \lvert x\rvert \leq width \\ 1, width<\lvert x\rvert<2*width\end{cases}$									
width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
5	0.01	2	1	$\frac{1}{(n+5)^{0.5}}$	9.507272	8.887635	8.88124	8.759379	8.603718
					9.231252	8.916839	9.160737	9.014931	9.049856

constant $\sigma(x) = C$				
8.584404	8.331632	8.68917	8.07853	8.492082
8.08258	8.4440	8.334455	8.904434	8.6586

2.4.7 variance

Kernel function : $K(x)=e^{-\frac{\lvert x\rvert^{\alpha_1}}{\alpha_2}}$									
α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	0.1	2	1	$\frac{1}{(n+5)^{0.5}}$	0.00996478	0.01011733	0.009977178	0.01044986	0.009936305
					0.01034323	0.01059126	0.01015497	0.01004813	0.01011846

Kernel function : $K(x)=\frac{1}{1+\alpha_1 \lvert x\rvert^{\alpha_2}}$									
α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	0.1	1	1	$\frac{1}{(n+5)^{0.5}}$	0.01053137	0.009726586	0.009621487	0.009998282	0.00993145
					0.009772318	0.009948752	0.009891014	0.009880959	0.01007295

Kernel function : $K(x)=\begin{cases}0, \lvert x\rvert \geq 2*width \\ -1*hegith, width<\lvert x\rvert<2*width \\ 1*height, \lvert x\rvert \leq width\end{cases}$										
height	width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	0.5	0.01	10	1	$\frac{1}{(n+5)^{0.5}}$	0.009485893	0.009529391	0.009935555	0.009808818	0.009759364
						0.01000965	0.009559542	0.00991694	0.00984394	0.009937325

Kernel function : $K(x)=\begin{cases}0, \lvert x\rvert \geq 2*width \quad or \quad \lvert x\rvert \leq width \\ 1, width<\lvert x\rvert<2*width\end{cases}$									
width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
5	0.01	2	1	$\frac{1}{(n+5)^{0.5}}$	0.01031362	0.009982451	0.01007545	0.009845031	0.00987189
					0.0101037	0.01016246	0.01006309	0.01013637	0.01010916

constant $\sigma(x) = C$				
0.00972953	0.0097068	0.009780478	0.00939504 an	0.009631347
0.0094686	0.009739506	0.00954663	0.009958031	0.009772716

2.4.8 comparison of average squared jump distance

Kernel function : $K(x) = e^{-\frac{|x|^{\alpha_1}}{\alpha_2}}$

α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	0.1	2	1	$\frac{1}{(n+5)^{0.5}}$	0.506856	0.504129	0.493900	0.491895	0.491927
					0.506880	0.50175	0.495962	0.480626	0.496152

Kernel function : $K(x) = \frac{1}{1+\alpha_1 \mid x \mid^{\alpha_2}}$

α_1	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
1	0.1	1	1	$\frac{1}{(n+5)^{0.5}}$	0.458961	0.461018	0.4751	0.480245	0.47135
					0.466943	0.467047	0.4542	0.4680	0.45683

Kernel function : $K(x) = \begin{cases} 0, \mid x \mid \geq 2 * width \\ -1 * hegith, width < \mid x \mid < 2 * width \\ 1 * height, \mid x \mid \leq width \end{cases}$

height	width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
0.5	0.5	0.01	10	1	$\frac{1}{(n+5)^{0.5}}$	0.463001	0.46015	0.449489	0.465211	0.457632
						0.461200	0.465179	0.456853	0.452455	0.461372

Kernel function : $K(x) = \begin{cases} 0, \mid x \mid \geq 2 * width \quad or \quad \mid x \mid \leq width \\ 1, width < \mid x \mid < 2 * width \end{cases}$

width	C	γ	b_n	η_n	First run	Second run	Third run	Fourth run	Fifth run
5	0.01	2	1	$\frac{1}{(n+5)^{0.5}}$	0.394871	0.406117	0.404594	0.396869	0.414291
					0.40248	0.421151	0.404090	0.40975	0.413094

constant $\sigma(x) = C$				
0.4365735	0.4408152	0.421703	0.4340354	0.4279432
0.4396845	0.4289132	0.4226133	0.4327815	0.4182256