STA4502 MiniProject

Zhenan Fan Student# 1000233151
Yeming Wen Student# 1000523882

March 3, 2016

We are applying MCMC algorithm to Bayesian linear regression in the context of polynomial fitting problem. In particular, we are interested in the predictive distribution. In this mini-project, we first define an arbitrary nonlinear function

$$f(x) = \frac{1}{6}(3\sin(2(x/3+1)^2) + 6\cos(2(x/3+1)^2) + 8)$$

Then we generate 301 data points from $f(x)$ with noise as our training data. Our goal is to use Bayesian linear regression to predict $\hat{f}(x^*)$ and then we can compare it to $f(x^*)$ to see how well our algorithm did.
In this project we assume $p(t|x, w, \beta) \sim N(t|\sum_{j=0}^{D} w_j x^j, \beta^{-1})$, which means we want to fit a polynomial to the training data set. Also, we give a normal prior to the weights $p(w) \sim N(w|0, aI)$. Then we can write down the predictive distribution, let D denote the training data set, x^* denote the given input, and t^* is our prediction,

$$p(t^*|x^*, D) = \int\int\int p(t^*|x^*, w, \beta)p(w|D, a, \beta)p(a, \beta|D)dwdad\beta$$

We can consider $p(t^*|x^*, w, \beta)$ as the likelihood function and $p(w|D, a, \beta)$ as the posterior distribution.
In general, let M be the number of iterations in MCMC algorithm. To simplify the model, we assume positive $f(x)$, then we only have to consider positive t^*. Also, we generate an alphalist and a betalist with normal distribution with mean 0 and corresponding training data variance. So within one MCMC iteration, we treat a, β as constants, so $p(a, \beta|D)$ is removed, which implies,

$$p(t^*|x^*, D) = \int p(t^*|x^*, w, \beta)p(w|D, a, \beta)dw$$

.
Next we are interested in the mean of predictive distribution. So we want to compute

$$\begin{aligned} E(t^*) &= \int t^* p(t^*|x^*, D)dt^* && (1)\\ &= \int\int t^* p(t^*|x^*, w, \beta)p(w|D, a, \beta)dwdt^* && (2)\\ &= \int\int e^{t^*} t^* p(t^*|x^*, w, \beta)p(w|D, a, \beta)e^{-t^*}dwdt^* && (3)\end{aligned}$$

In this model, we fit a 5-degree polynomial and try to predict values for 20 new inputs, so we have the following likelihood function

$$p(t|x, w, \beta) \sim N(t|\sum_{j=0}^{5} w_j x^j, \beta^{-1})$$

Also we let $\pi = p(w|D, a, \beta)$, which it is the posterior distribution of w. i.e.

$$\pi(w) \propto p(w|a, \beta) * p(target|input, w, a, \beta)$$

To be consistent to the notation in class, we set $h(t^*, w) = e^{t^*} t^* p(t^*|x^*, w, \beta)$, where $p(t^*|x^*, w, \beta)$ is the likelihood function for new input.
In the MCMC algorithm, we first initialize w according to the prior, which it is a 6-dimensional vector. Next

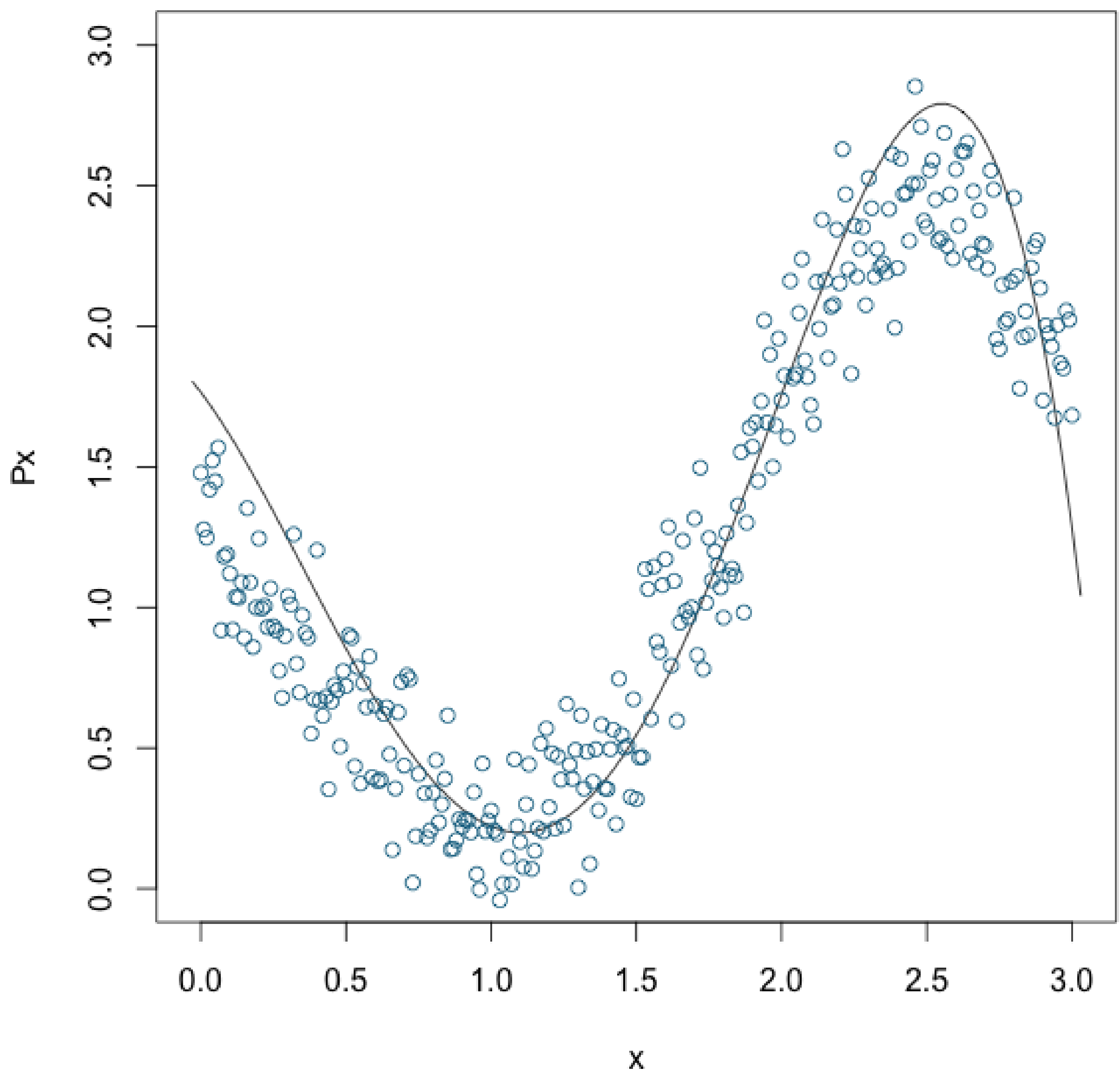
we ran something similar to Metropolis algorithm. Specifically, we propose a new vector w', and accept it with probability $\frac{\pi(w')}{\pi(w)}$. Then with this w (it's w' if we accept otherwise it is the w from last iteration), and sampling t^* from $exp(1)$, we can compute $h(t^*, w, x)$ within this iteration, where x is the new input value and we have 20 of them.
Repeat the above procedure M times, the mean of hlist after burn-in is our final prediction for 20 new inputs. Also we keep track of the value of w, so we can write down our fitted polynomial.

Listing 1: Random Walk Metropolis R code

```
#Target polynomial:

f = function(x) {(3*sin(2*(x/3+1)^2) + 6*cos(2*(x/3+1)^2) + 8 )/6}
input = seq(from=0, to=3, by=0.01)
y = f(input)
target = y + rnorm(301,0,0.2)
plot(input,target,col='deepskyblue4',xlab='x',main='Observed data')

# Define function for varfact
varfact <- function(series) { 2 * sum(acf(series, plot=FALSE)$acf) - 1 }

D = 6

noise = sd(target)

# points want to extimates
xstarlist = runif(20)

# exact values
ylist = f(xstarlist)

# return polynomial with coefficients in w
poly = function(w,x,D){
  s = 0
  for (i in 1:D){
    s = s + w[i]*x^(i-1)
  }
  return(s)
}

h = function(t,w,beta,x){
  return(exp(t)*t*dnorm(t, mean = poly(w,x,D), sd = 1/beta))
}

logmultinorm = function(w,alpha,D){
  log((2*pi)^(-0.5*D)*(1/alpha)^(-0.5*D)*exp(-0.5*alpha*sum(w^2)))
}

logg = function(w,input,target,alpha,beta,D){
  s = 0
  for (i in 1:length(target)){
    s = s + dnorm(target[i],poly(w,input[i],D),1/beta,log = TRUE)
  }
  return(s + logmultinorm(w,alpha,D))
}
```

```
M = 110000   # run length
B = 10000   # amount of burn-in
alphalist = abs(rnorm(M,0,noise))
betalist = abs(1/rnorm(M,0,noise))
tlist = rexp(M)
# for keeping track of values
wmatrix = matrix(rep(0,M*D),nrow = D,ncol = M)
hmatrix = matrix(rep(0,M*20),nrow = 20,ncol = M)
numaccept = 0
# overdispersed starting distribution (dim=D)
W = rnorm(D,0,1/alphalist[1])
sigma = 0.5   # proposal scaling


for (i in 1:M) {
  Y = W + sigma * rnorm(D)   # proposal value (dim=D)
  U = runif(1)   # for accept/reject
  a = logg(Y,input,target,alphalist[i],betalist[i],D) -
  logg(W,input,target,alphalist[i],betalist[i],D)   # for accept/reject
  if (log(U) < a) {
    W = Y   # accept proposal
    numaccept = numaccept + 1
  }
  for (j in 1:D){
    wmatrix[j,i] = W[j]
  }
  for (k in 1:20){
    hmatrix[k,i] = h(tlist[i],W,betalist[i],xstarlist[k]);
  }
}

for (k in 1:20){
  estimate = mean(hmatrix[k,(B+1):M])
  iidse =  sd(hmatrix[k,(B+1):M]) / sqrt(M-B)
  se = iidse*sqrt(varfact(hmatrix[k,(B+1):M]) )
  cat("Estimate for x = ", xstarlist[k], "=" , estimate, ",
  True value (f(x))= ", ylist[k], "\n")
  cat("approximate 95% confidence interval is (", estimate - 1.96 * se, ",",
      estimate + 1.96 * se, ")\n\n")
}

w = wmatrix[,M]
predictpoly = polynomial(coef = w)


plot(predictpoly,xlim = c(0,3),ylim = c(0,3))
points(input,target,type = "p", col='deepskyblue4',xlab='x',main='Observed data')
```

And we get the output



Listing 2: R output

```
Estimate for x =  0.160577 is 1.419294 , True value (f(x))=  1.127248
approximate 95% confidence interval is ( 1.391167 , 1.44742 )

Estimate for x =  0.9899601 is 0.4158945 , True value (f(x))=  0.2178462
approximate 95% confidence interval is ( 0.4081417 , 0.4236472 )

Estimate for x =  0.453312 is 0.9170817 , True value (f(x))=  0.6876926
approximate 95% confidence interval is ( 0.9031105 , 0.9310528 )
```

```
Estimate for x =  0.5914694 is 0.7158354 , True value (f(x))=  0.5068513
approximate 95% confidence interval is ( 0.7059315 , 0.7257393 )

Estimate for x =  0.7428511 is 0.5504592 , True value (f(x))=  0.3479855
approximate 95% confidence interval is ( 0.5426297 , 0.5582888 )

Estimate for x =  0.464768 is 0.9075627 , True value (f(x))=  0.671718
approximate 95% confidence interval is ( 0.8920921 , 0.9230332 )

Estimate for x =  0.07897275 is 1.522164 , True value (f(x))=  1.252638
approximate 95% confidence interval is ( 1.49407 , 1.550257 )

Estimate for x =  0.6387615 is 0.6542706 , True value (f(x))=  0.4520785
approximate 95% confidence interval is ( 0.645401 , 0.6631402 )

Estimate for x =  0.5817941 is 0.7331457 , True value (f(x))=  0.5185618
approximate 95% confidence interval is ( 0.7220487 , 0.7442426 )

Estimate for x =  0.7518658 is 0.5440949 , True value (f(x))=  0.3401651
approximate 95% confidence interval is ( 0.5359097 , 0.5522802 )

Estimate for x =  0.08717507 is 1.513882 , True value (f(x))=  1.24011
approximate 95% confidence interval is ( 1.482046 , 1.545719 )

Estimate for x =  0.9182943 is 0.4404978 , True value (f(x))=  0.2361555
approximate 95% confidence interval is ( 0.4326342 , 0.4483613 )

Estimate for x =  0.3694 is 1.046509 , True value (f(x))=  0.8087238
approximate 95% confidence interval is ( 1.030208 , 1.06281 )

Estimate for x =  0.5742994 is 0.7416084 , True value (f(x))=  0.5277443
approximate 95% confidence interval is ( 0.7308318 , 0.752385 )

Estimate for x =  0.6752285 is 0.6152825 , True value (f(x))=  0.4128695
approximate 95% confidence interval is ( 0.6068049 , 0.6237601 )

Estimate for x =  0.8706393 is 0.4635071 , True value (f(x))=  0.2575953
approximate 95% confidence interval is ( 0.4556642 , 0.4713501 )

Estimate for x =  0.2876286 is 1.197784 , True value (f(x))=  0.9316992
approximate 95% confidence interval is ( 1.177637 , 1.21793 )

Estimate for x =  0.674066 is 0.6179598 , True value (f(x))=  0.4140761
approximate 95% confidence interval is ( 0.6088086 , 0.6271109 )

Estimate for x =  0.7465105 is 0.5465784 , True value (f(x))=  0.3447863
approximate 95% confidence interval is ( 0.5390394 , 0.5541173 )

Estimate for x =  0.1921243 is 1.367434 , True value (f(x))=  1.078529
approximate 95% confidence interval is ( 1.341655 , 1.393213 )

estimate polynomial :
1.76558 - 1.370776*x - 1.774254*x^2 + 1.83905*x^3 - 0.1733876*x^4 -
0.06588073*x^5
```

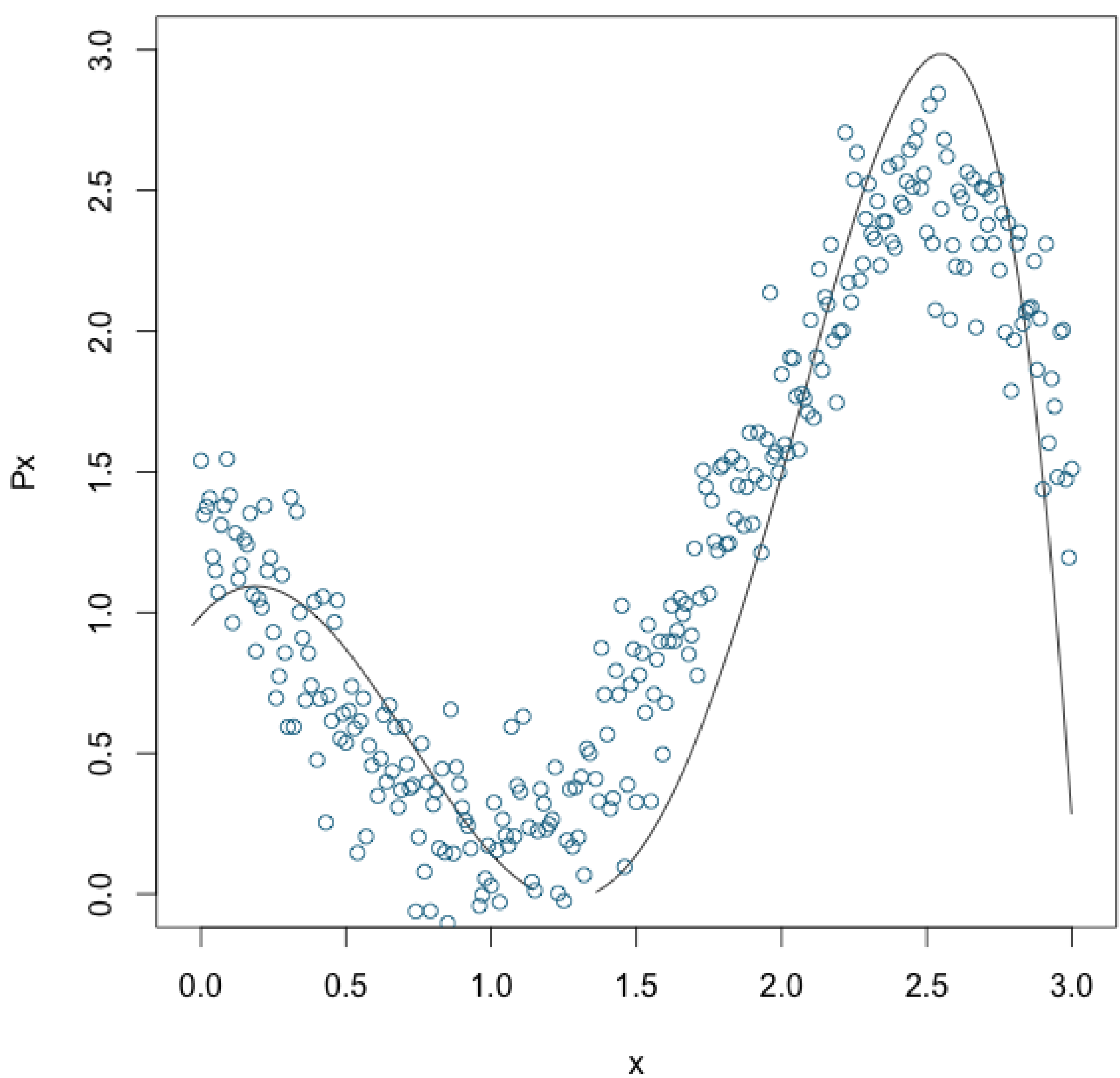
Here is the code for Variable-At-A-Time Algorithm

Listing 3: Variable-At-A-Time R code

```
#Target polynomial:

f = function(x) {(3*sin(2*(x/3+1)^2) + 6*cos(2*(x/3+1)^2) + 8 )/6}
input = seq(from=0, to=3, by=0.01)
y = f(input)
target = y + rnorm(301,0,0.2)
plot(input,target,col='deepskyblue4',xlab='x',main='Observed data')

# Define function for varfact
varfact <- function(series) { 2 * sum(acf(series, plot=FALSE)$acf) - 1 }

D = 6

noise = sd(target)

# points want to extimates
xstarlist = runif(20,0,3)

# exact values
ylist = f(xstarlist)

# return polynomial with coefficients in w
poly = function(w,x,D){
  s = 0
  for (i in 1:D){
    s = s + w[i]*x^(i-1)
  }
  return(s)
}

h = function(t,w,beta,x){
  return(exp(t)*t*dnorm(t, mean = poly(w,x,D), sd = 1/beta))
}

logmultinorm = function(w,alpha,D){
  log((2*pi)^(-0.5*D)*(1/alpha)^(-0.5*D)*exp(-0.5*alpha*sum(w^2)))
}

logg = function(w,input,target,alpha,beta,D){
  s = 0
  for (i in 1:length(target)){
    s = s + dnorm(target[i],poly(w,input[i],D),1/beta,log = TRUE)
  }
  return(s + logmultinorm(w,alpha,D))
}


M = 110000  # run length
B = 10000  # amount of burn-in
alphalist = abs(rnorm(M,0,noise))
betalist = abs(1/rnorm(M,0,noise))
```

```
tlist = rexp(M)
# for keeping track of values
wmatrix = matrix(rep(0,M*D),nrow = D,ncol = M)
hmatrix = matrix(rep(0,M*20),nrow = 20,ncol = M)
numaccept = 0
# overdispersed starting distribution (dim=D)
W = rnorm(D,0,1/alphalist[1])
sigma = 0.1   # proposal scaling


for (i in 1:M) {
  coord = floor( runif(1,1,D+1) )  # uniform on {1,2,...,D}
  Y = W
  Y[coord] = W[coord] + sigma * rnorm(1)  # proposal
  U = runif(1)  # for accept/reject
  a = logg(Y,input,target,alphalist[i],betalist[i],D) -
  logg(W,input,target,alphalist[i],betalist[i],D)  # for accept/reject
  if (log(U) < a) {
    W = Y  # accept proposal
    numaccept = numaccept + 1
  }
  for (j in 1:D){
    wmatrix[j,i] = W[j]
  }
  for (k in 1:20){
    hmatrix[k,i] = h(tlist[i],W,betalist[i],xstarlist[k]);
  }
}

for (k in 1:20){
  estimate = mean(hmatrix[k,(B+1):M])
  iidse =  sd(hmatrix[k,(B+1):M]) / sqrt(M-B)
  se = iidse*sqrt(varfact(hmatrix[k,(B+1):M]) )
  cat("Estimate for x = ", xstarlist[k], "is" , estimate, ",
  True value (f(x)) is ", ylist[k], "\n")
  cat("approximate 95% confidence interval is (", estimate - 1.96 * se, ",",
     estimate + 1.96 * se, ")\n\n")
}
cat("acceptance rate = ", numaccept/M )

w = wmatrix[,M]
predictpoly = polynomial(coef = w)


plot(predictpoly,xlim = c(0,3),ylim = c(0,3))
points(input,target,type = "p", col='deepskyblue4',xlab='x',main='Observed data')
```

And we get the output



Listing 4: R output

```
Estimate for x =  2.157489 is 2.180967 , True value (f(x)) is  2.083078
approximate 95% confidence interval is ( 2.132901 , 2.229033 )

Estimate for x =  2.911417 is 1.408798 , True value (f(x)) is  1.919722
approximate 95% confidence interval is ( 1.383038 , 1.434559 )

Estimate for x =  1.663045 is 0.6688371 , True value (f(x)) is  0.9562295
approximate 95% confidence interval is ( 0.6610177 , 0.6766564 )
```

```
Estimate for x =  2.69406 is 2.91572 , True value (f(x)) is  2.336078
approximate 95% confidence interval is ( 2.816937 , 3.014502 )

Estimate for x =  2.895752 is 1.550337 , True value (f(x)) is  1.958341
approximate 95% confidence interval is ( 1.525694 , 1.57498 )

Estimate for x =  1.133888 is 0.3645747 , True value (f(x)) is  0.2359126
approximate 95% confidence interval is ( 0.3601866 , 0.3689627 )

Estimate for x =  0.3164865 is 1.082079 , True value (f(x)) is  0.8878831
approximate 95% confidence interval is ( 1.067628 , 1.096529 )

Estimate for x =  2.358848 is 2.861278 , True value (f(x)) is  2.377627
approximate 95% confidence interval is ( 2.77013 , 2.952425 )

Estimate for x =  2.11409 is 2.048102 , True value (f(x)) is  1.997377
approximate 95% confidence interval is ( 2.000968 , 2.095236 )

Estimate for x =  0.452603 is 1.012806 , True value (f(x)) is  0.6886861
approximate 95% confidence interval is ( 0.9994188 , 1.026194 )

Estimate for x =  0.525778 is 0.9565627 , True value (f(x)) is  0.5893972
approximate 95% confidence interval is ( 0.9426317 , 0.9704938 )

Estimate for x =  1.305032 is 0.3401239 , True value (f(x)) is  0.3593653
approximate 95% confidence interval is ( 0.3359197 , 0.344328 )

Estimate for x =  1.179608 is 0.348833 , True value (f(x)) is  0.2578536
approximate 95% confidence interval is ( 0.3445658 , 0.3531002 )

Estimate for x =  1.859953 is 1.147757 , True value (f(x)) is  1.414501
approximate 95% confidence interval is ( 1.13232 , 1.163194 )

Estimate for x =  1.335083 is 0.3455914 , True value (f(x)) is  0.392645
approximate 95% confidence interval is ( 0.3413017 , 0.349881 )

Estimate for x =  0.7642032 is 0.6878164 , True value (f(x)) is  0.3297966
approximate 95% confidence interval is ( 0.6795576 , 0.6960753 )

Estimate for x =  1.723669 is 0.7998767 , True value (f(x)) is  1.091986
approximate 95% confidence interval is ( 0.7891555 , 0.8105979 )

Estimate for x =  1.312721 is 0.3411793 , True value (f(x)) is  0.3675551
approximate 95% confidence interval is ( 0.3369647 , 0.3453939 )

Estimate for x =  1.833726 is 1.072244 , True value (f(x)) is  1.351405
approximate 95% confidence interval is ( 1.055814 , 1.088674 )

Estimate for x =  1.533271 is 0.4733783 , True value (f(x)) is  0.6935359
approximate 95% confidence interval is ( 0.4678844 , 0.4788721 )

estimate polynomial :
0.9899345 + 1.126159*x - 3.178382*x^2 + 0.4105512*x^3 + 1.108255*x^4 -
0.3141828*x^5
```