```
Alexander Winkles Coding Project 4

Here is the code used:

function [y, list]=nspline(xknot,fvalue,x)
%This is a matlab program to find a natural cubic spline. In the arguments in
%y=nspline(xknot,fvalue,x), xknot is a sequence of knots, fvalue is a
%sequence of function values, and x is a point or a sequence of points to be
%evaluated. Output y gives the value of the natural cubic spline at x.
%This is written by Dr. Ming-Jun Lai, Department of Mathematics, University
%of Georgia in 1994. See Dr. Lai for any question.
n=length(xknot);
l=length(fvalue);
if n~=l
   disp('The number of knots is not the same as the number of ')
   disp('function values.')
   return;
else
  h=zeros(size([1:n-1]));
 for i=1:n-1
  h(1,i)=xknot(1,i+1)-xknot(1,i);
 end;
 A=zeros(size([1:n]'*[1:n]));
 A(1,1)=1;A(n,n)=1;
 for i=1:n-2
  A(i+1,i)=h(1,i);
 end;
 for i=3:n
  A(i-1,i)=h(1,i-1);
 end;
 for i=2:n-1
  A(i,i)=2*(h(1,i)+h(1,i-1));
 end;

 bb=zeros(size([1:n]'));
 for i=2:n-1
  bb(i,1)=3*(fvalue(1,i+1)-fvalue(1,i))/h(1,i)...
          -3*(fvalue(1,i)-fvalue(1,i-1))/h(1,i-1);
 end;
 c=inv(A)*bb;
 for i=1:n-1
 b(1,i)=(fvalue(1,i+1)-fvalue(1,i))/h(1,i)-h(1,i)*(2*c(i,1)+c(i+1,1))/3;
 d(1,i)=(c(i+1,1)-c(i,1))/(3*h(1,i));
 end;
 m=length(x);
 for i=1:m
  for j=1:n-1
   if xknot(1,j)<=x(1,i) & x(1,i)<=xknot(1,j+1)
   y(1,i)=fvalue(1,j)+b(1,j)*(x(1,i)-xknot(1,j))...
         +c(j,1)*(x(1,i)-xknot(1,j))^2+d(1,j)*(x(1,i)-xknot(1,j))^3;
   end;
  end;
 end;
end;
```

```
 list=zeros(size([1:n-1]'*[1 2 3 4]));
 for i=1:n-1
 list(i,4)=fvalue(1,i); list(i,3)=b(1,i); list(i,2)=c(i,1); list(i,1)=d(1,i);
 end;
% list

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Problem 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x = [0.9,1.3,1.9,2.1,2.6,3.0,3.9,4.4,4.7,5.0,6.0,7.0,8.0,9.2,10.5,11.3,11.6,12.0,12.6,13.0,13.3];
y = [1.3,1.5,1.85,2.1,2.6,2.7,2.4,2.15,2.05,2.1,2.25,2.3,2.25,1.95,1.4,0.9,0.7,0.6,0.5,0.4,0.25];
r = 0.9:0.01:13.3;
s = zeros(1,length(r));
for i = 1:length(r)
s(1,i) = nspline(x,y,r(1,i));
end;
plot(x,y,'+',r,s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Problem 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

f =@(x) 1./(1+25*x.^2)

f =

    @(x)1./(1+25*x.^2)

x = -1 : 0.5 : 1;
y = f(x);
r = -1:0.01:1;
s = zeros(1,length(r));
for i=1:length(r)
s(1,i) = nspline(x,y,r(1,i));
end;
plot(x,y,'+',r,s)

x = -1 : 0.25 : 1;
y = f(x);
s = zeros(1,length(r));
for i=1:length(r)
s(1,i) = nspline(x,y,r(1,i));
end;
plot(x,y,'+',r,s)

x = -1:0.2:1;
y = f(x);
s = zeros(1,length(r));
for i=1:length(r)
s(1,i) = nspline(x,y,r(1,i));
end;
plot(x,y,'+',r,s)
```

```
x = -1:(1/7):1;
y = f(x);
s = zeros(1,length(r));
for i=1:length(r)
s(1,i) = nspline(x,y,r(1,i));
end;
plot(x,y,'+',r,s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Problem 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

f =@(x) exp(x);
x = 0 : 0.5 : 5;;
y = f(x);
s = zeros(1,length(r));
r = 0 : 0.01 : 5;
s = zeros(1,length(r));
for i=1:length(r)
s(1,i) = nspline(x,y,r(1,i),exp(0),exp(5));
end;
plot(x,y,'+',r,s)
```

Figure 1: Duck Approximation

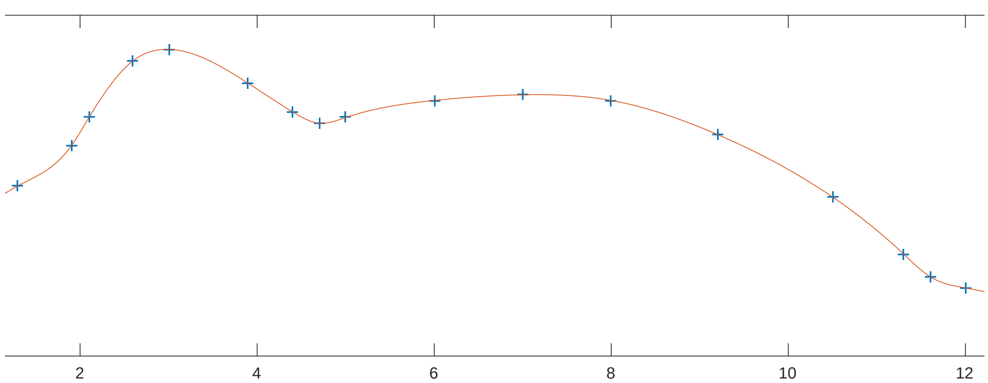

Figure 2: Runge function with n = 5

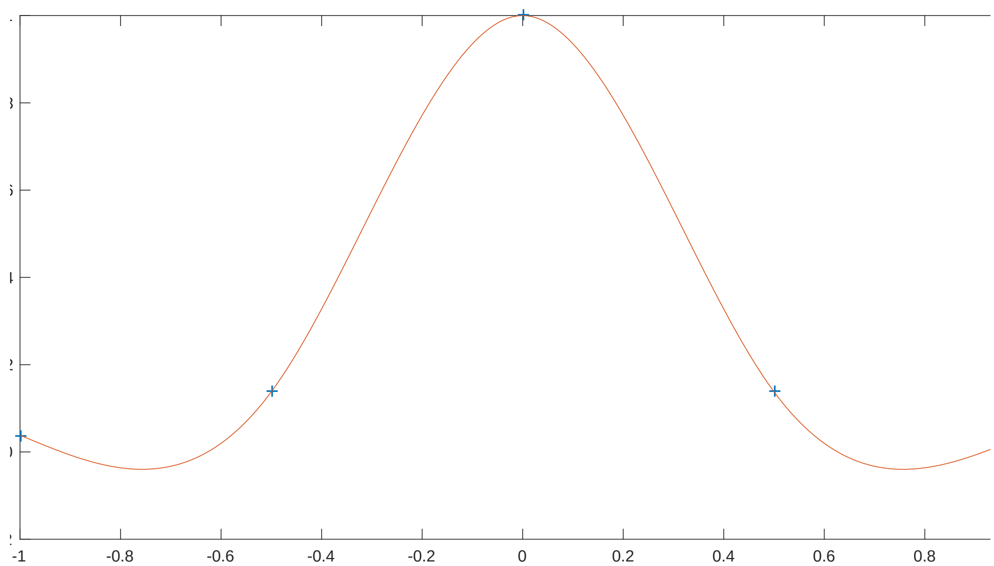

Figure 3: Runge function with n = 9

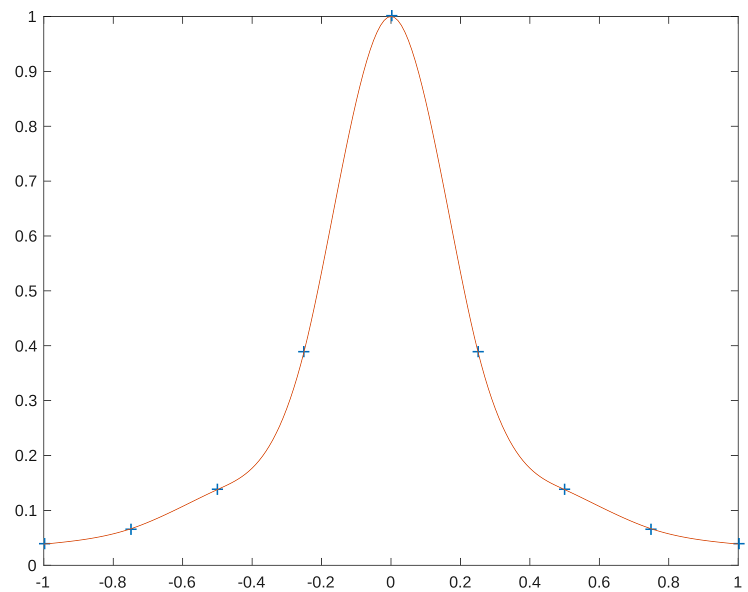

Figure 4: Runge function with n = 11

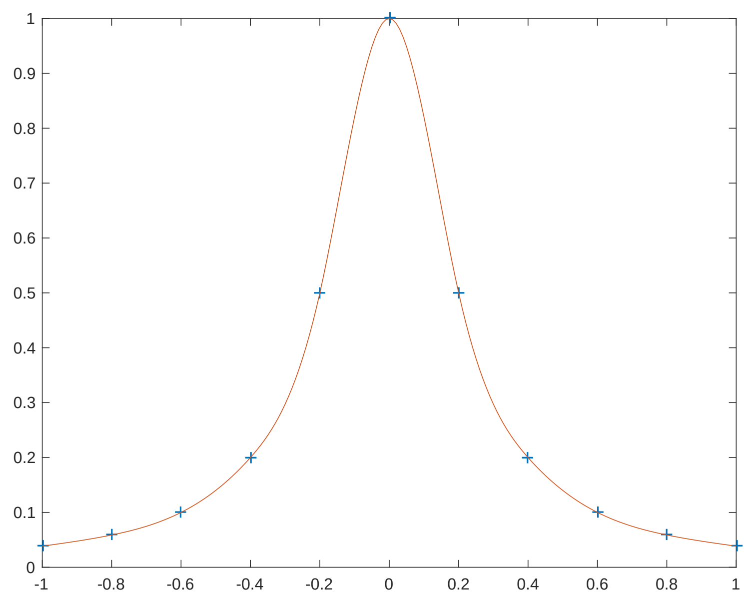

Figure 5: Runge function with n = 15

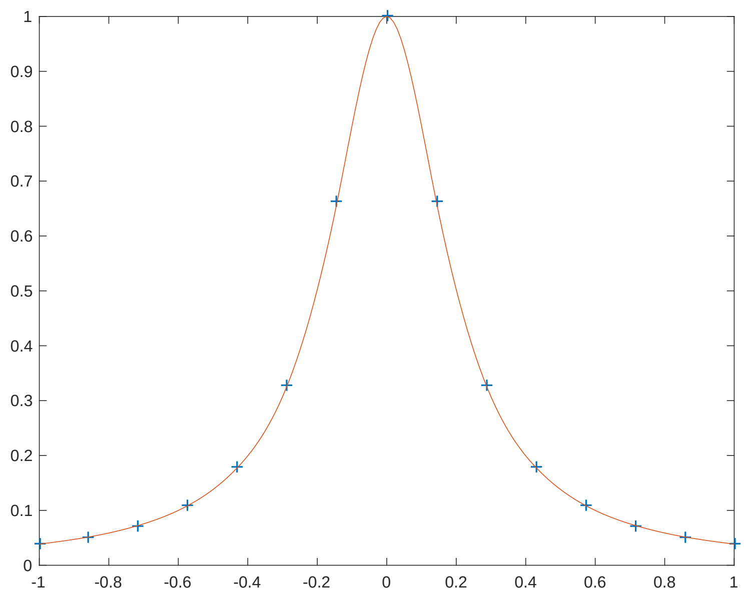

Figure 6: Spline interpolation of e^x

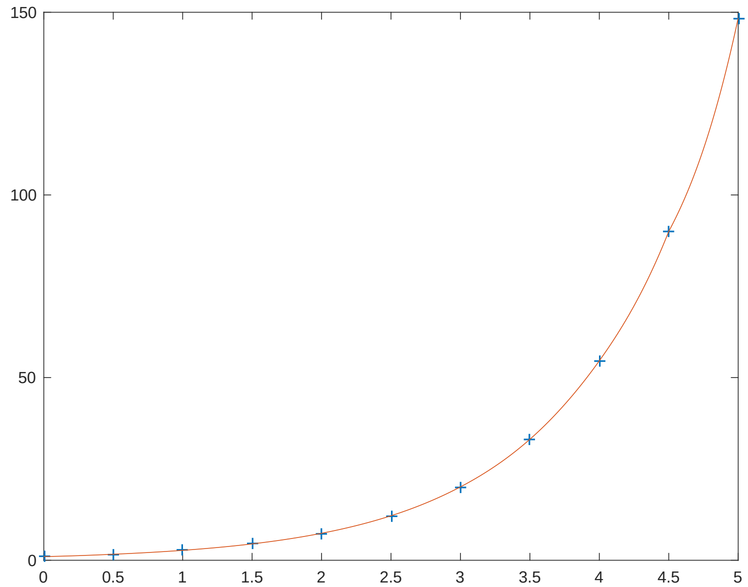

Figure 7: Comparison of e^x and its spline