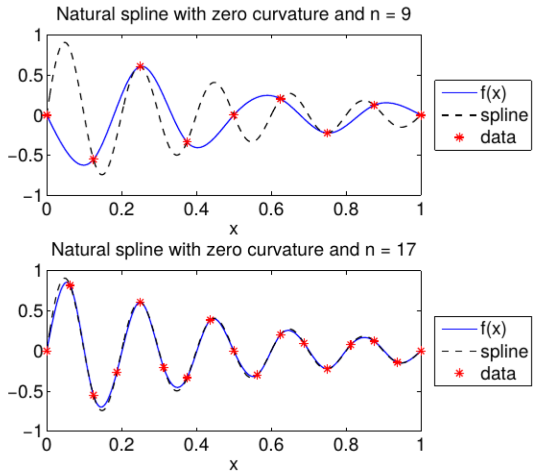
1. The requested function is as follows.

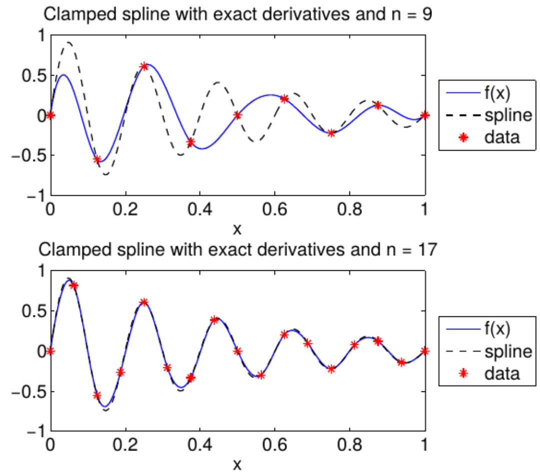
```
%  Returns slope values defining a clamped/natural spine, based on
%  input type = C/N. (x,y) are data values, and alfa and beta are
%  first/second derivative values at the endpoints x_1 and x_n.
%  function s=GetSplineSlopes(x,y,alfa,beta,type)
   function s=GetSplineSlopes(x,y,alfa,beta,type);
n=length(x);
b=zeros(n,1);

% Vectors e,d,f define a tridiagonal matrix A, but
% e(1) and f(n) are wasted space that is not used.
e=zeros(n,1); d=zeros(n,1); f=zeros(n,1);

% Get Dx values and first order divided differences.
for i=1:n-1
    Dx(i)   = x(i+1)-x(i);
    ddiv(i) = (y(i+1)-y(i))/Dx(i);
end

% Build up rows 2 to n-1 (same for all splines).
for i=2:n-1
    b(i)=3*(Dx(i-1)*ddiv(i) + Dx(i)*ddiv(i-1));
    e(i) = Dx(i); d(i) = 2*(Dx(i)+Dx(i-1)); f(i) = Dx(i-1);
end

switch type, % See class notes on WebCT for the formulas used here.
  case 'C',
     % Rows 1 and n particular for clamped spline.
     d(1) = 1; b(1) = alfa;
     d(n) = 1; b(n) = beta;
  case 'N',
     % Rows 1 and n particular for natural spline.
     d(1) = 2; f(1) = 1; b(1)=3*ddiv(1)  -Dx(1)  *alfa;
     e(n) = 1; d(n) = 2; b(n)=3*ddiv(n-1)+Dx(n-1)*beta;
  otherwise,
     error=('Variable type not C or N')
end

% System is tridiagonal, so use tridiagonal solve.
% (Note: TriDiLU, LBiDiSol, UBiDiSol from WebCT.)
[l, u] = TriDiLU(d,e,f);
tmp = LBiDiSol(l,b);
s   = UBiDiSol(u,f,tmp);
```

2. A script which generates the requested plots follows.

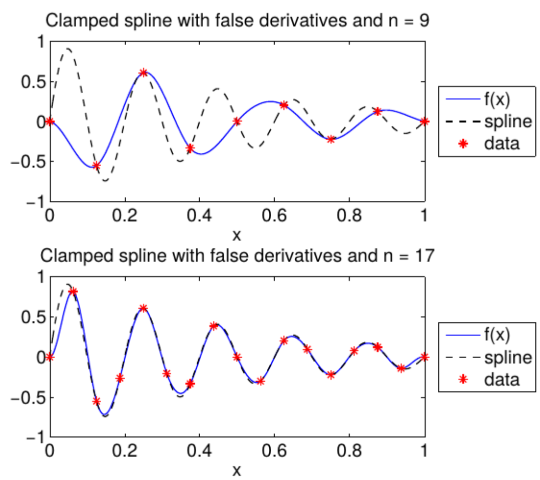
```
% Script: Set9Problem2
% Makes plots for Problem 2 of Homework Set 9.

parts = ['a','b','c'];
ns = [9 17];
for j = 1:length(parts)
part = parts(j);
for k = 1:length(ns)
n = ns(k);
switch n,
   case 9,
     figrow = 1;
   case 17,
     figrow = 2;
   otherwise,
     error('Problem 2 requires n equal to 9 or 17')
end
x = transpose(linspace(0,1,n));
y = exp(-2*x).*sin(10*pi*x);
switch part
   case 'a',
       % Zero curvature conditions/natural spline.
       alfa = 0; beta = 0;
       s = GetSplineSlopes(x,y,alfa,beta,'N');
       fignum = 1;
```



(a) Plot for Problem 2a.



(b) Plot for Problem 2b.



(c) Plot for Problem 2c.

Figure 1: Plots for Problem 2a,b,c.

Multiplication of both sides by $\Delta x_i \Delta x_{i+1}$ then yields

$$\Delta x_{i+1}\left(s_i + 2s_{i+1} - 3y[x_i, x_{i+1}]\right) = \Delta x_i\left(-s_{i+2} - 2s_{i+1} + 3y[x_{i+1}, x_{i+2}]\right).$$

Finally, moving all s_k terms to the left side and all divided differences to the right, we arrive at

$$\Delta x_{i+1} s_i + 2(\Delta x_{i+1} + \Delta x_i)s_{i+1} + \Delta x_i s_{i+2} = 3y[x_i, x_{i+1}]\Delta_{i+1} + 3y[x_{i+1}, x_{i+2}]\Delta x_i.$$

This equation holds for $i = 1, \ldots, N-1$, and in all constitute rows 2 through $N-1$ of the linear systems which determines the $\boldsymbol{s}$ which defines the clamped spline. We complete the system by adding $s_1 = \alpha$ as the 1st row and $s_N = \beta$ as the nth row.

4. The required script follows. Its output is shown in Fig. 2.

```
%  Script: Set9Problem4
%  Data from the problem statement follows.
```

```
x=transpose([4.7 4.0 4.6 2.3 2.3 4.7 4.1 8.3 6.0 8.6 7.8 5.4 5.4 4.7]);
y=transpose([1.3 1.9 2.5 3.1 3.9 3.1 5.8 5.8 3.0 2.9 2.0 2.4 1.5 1.3]);
n=length(x);

%  Set up arc length based on piecewise linear interpolant.
lambda=zeros(n,1);
lambda(1)=0;
for i=2:n
    lambda(i)=lambda(i-1)+sqrt((x(i)-x(i-1))^2 + (y(i)-y(i-1))^2);
end

% Array for dense evaluation.
L=transpose(linspace(min(lambda),max(lambda),500));

% Zero curvature conditions.
alfax = 0; betax = 0; alfay = 0; betay = 0;

sx = GetSplineSlopes(lambda,x,alfax,betax,'N');
Cx = pwchermite_coeffs(lambda,x,sx);
X  = eval_pwpoly(lambda,Cx,L);
sy = GetSplineSlopes(lambda,y,alfay,betay,'N');
Cy = pwchermite_coeffs(lambda,y,sy);
Y  = eval_pwpoly(lambda,Cy,L);
figure(4); explot(X,Y,'b-',x,y,'r*')
title('Zero curvature natural spline'); xlabel('x'); ylabel('y')
saveas(gcf,'UpsideDownGhostSpline.eps','epsc')
```

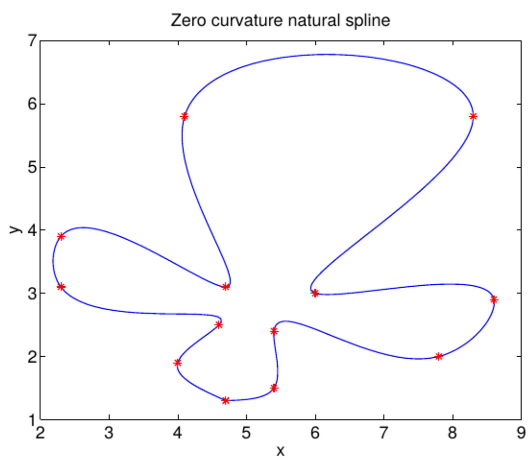


Figure 2: Plot for Problem 4.

1. The following script sets up and solves both described least-squares problems. Its output also follows.

```
%  Script: Set10Problem1
ns = [6 8]; H  = hilb(10);
for k = 1:length(ns);
    n = ns(k);
    A = H(:,1:n);  AtA = A'*A;
    c = ones(n,1); b = A*c; Atb = A'*b;
    x = AtA\Atb;  % Solve the normal equations
    rel_ferr = norm(x-c,inf);  % Note forward err same as relative forward err.
    rel_berr = norm(Atb-AtA*x,inf)/norm(Atb,inf);
    kappa = cond(AtA,inf);
    disp(['                                                         '])
    disp(['Choice of n:               ',num2str(n)                   ])
    disp(['Forward error in inf norm:',num2str(rel_ferr,'%0.5g')         ])
    disp(['Error magnification:       ',num2str(rel_ferr/rel_berr,'%0.5g')])
    disp(['kappa_inf of A^tA:         ',num2str(kappa,'%0.5g')           ])
end
```

```
>> format compact; warning off MATLAB:nearlySingularMatrix; Set10Problem1

Choice of n:              6
Forward error in inf norm:9.3258e-05
Error magnification:      4.6915e+11
kappa_inf of A^tA:        1.2002e+13

Choice of n:              8
Forward error in inf norm:2.1503
Error magnification:      1.2344e+16
kappa_inf of A^tA:        2.204e+17
```

For (a) $n = 6$, the numerical solution is correct to about 4 digits, and the infinity-norm condition number $\kappa_\infty(A^T A) \simeq \mathsf{1.2002e+13}$. If $\kappa_\infty(B) \simeq 10^k$, then, when *numerically* solving $B\mathbf{x} = \mathbf{g}$, we may lose k digits relative to the 16 or so in double precision (see Sauer, page 95). That is relative forward errors of size $\varepsilon_{\text{mach}} \cdot \kappa_\infty(B)$ are possible. Therefore, we expect to have a solution accurate to 3 digits, and the actual solution is a bit better. The error magnification is less than the condition number (the upper bound). For (b) $n = 8$, the numerical solution has no correct digits, nor are we guaranteed any since $\kappa_\infty(A^T A) \simeq \mathsf{2.204e+17}$.

2. The matrix is $A = [\mathbf{a}_1, \mathbf{a}_2]$, where

$$\mathbf{a}_1 = \begin{pmatrix} -4 \\ -2 \\ 4 \end{pmatrix}, \qquad \mathbf{a}_2 = \begin{pmatrix} -4 \\ 7 \\ -5 \end{pmatrix}.$$

We will first construct a "thin decomposition"

$$[\mathbf{a}_1, \mathbf{a}_2] = [\mathbf{q}_1, \mathbf{q}_2] \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix},$$

which is equivalent to the equations

$$\mathbf{a}_1 = r_{11}\mathbf{q}_1, \qquad \mathbf{a}_2 = r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2.$$

From the first equation, $r_{11} = \|\mathbf{a}_1\|_2 = 6$, which yields

$$\mathbf{q}_1 = \frac{\mathbf{a}_1}{r_{11}} = \begin{pmatrix} -\frac{2}{3} \\ -\frac{1}{3} \\ \frac{2}{3} \end{pmatrix}.$$

Rewrite the second equation as $r_{22}\mathbf{q}_2 = \mathbf{a}_2 - r_{12}\mathbf{q}_1$. Using $\mathbf{q}_1^T\mathbf{q}_2 = 0$, we get $r_{12} = \mathbf{q}_1^T\mathbf{a}_2 = \frac{8}{3} - \frac{7}{3} - \frac{10}{3} = -3$, and

$$r_{22}\mathbf{q}_2 = \mathbf{a}_2 - (\mathbf{q}_1^T\mathbf{a}_2)\mathbf{q}_1 = \begin{pmatrix} -4 \\ 7 \\ -5 \end{pmatrix} + 3\begin{pmatrix} -\frac{2}{3} \\ -\frac{1}{3} \\ \frac{2}{3} \end{pmatrix} = \begin{pmatrix} -6 \\ 6 \\ -3 \end{pmatrix}, \qquad r_{22} = \left\| \begin{pmatrix} -6 \\ 6 \\ -3 \end{pmatrix} \right\|_2 = 9$$

Finally,

$$\mathbf{q}_2 = \frac{1}{r_{22}} \begin{pmatrix} -6 \\ 6 \\ -3 \end{pmatrix} = \begin{pmatrix} -\frac{2}{3} \\ \frac{2}{3} \\ -\frac{1}{3} \end{pmatrix},$$

and our "thin decomposition" is then

$$\begin{pmatrix} -4 & -4 \\ -2 & 7 \\ 4 & -5 \end{pmatrix} = \begin{pmatrix} -\frac{2}{3} & -\frac{2}{3} \\ -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix} \begin{pmatrix} 6 & -3 \\ 0 & 9 \end{pmatrix}$$

To get a "thick" QR decomposition, where Q is orthogonal, we construct a third unit vector $\mathbf{q}_3$ which is orthogonal to $\mathbf{q}_1$ and $\mathbf{q}_2$. The cross product affords one construction,

$$\mathbf{q}_3 = \mathbf{q}_1 \times \mathbf{q}_2 = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ -\frac{2}{3} & -\frac{1}{3} & \frac{2}{3} \\ -\frac{2}{3} & \frac{2}{3} & -\frac{1}{3} \end{vmatrix} = -\frac{1}{3}\mathbf{i} - \frac{2}{3}\mathbf{j} - \frac{2}{3}\mathbf{k} = \begin{pmatrix} -\frac{1}{3} \\ -\frac{2}{3} \\ -\frac{2}{3} \end{pmatrix}.$$

Therefore,

$$\begin{pmatrix} -4 & -4 \\ -2 & 7 \\ 4 & -5 \end{pmatrix} = [\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3] \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} -\frac{2}{3} & -\frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} & -\frac{2}{3} \end{pmatrix} \begin{pmatrix} 6 & -3 \\ 0 & 9 \\ 0 & 0 \end{pmatrix}.$$

Using MATLAB® , we find the following decomposition.

```
>> format compact
>> A = [-4 -4; -2 7; 4 -5];
>> [Q R] = qr(A);
>> rats(A)
ans =
      -4             -4
      -2              7
       4             -5
>> rats(Q)
ans =
     -2/3           2/3            1/3
     -1/3          -2/3            2/3
      2/3           1/3            2/3
>> rats(R)
ans =
       6             -3
       0             -9
       0              0
```

This differs slightly from our hand-constructed decomposition, but both are valid QR decompositions. Indeed, since $[\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3]$ is an orthogonal matrix, so is $[\mathbf{q}_1, -\mathbf{q}_2, -\mathbf{q}_3]$.

4. Using the QR decomposition from above, we see that the least squares problem to solve is

$$\begin{pmatrix} -\frac{2}{3} & -\frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} & -\frac{2}{3} \end{pmatrix} \begin{pmatrix} 6 & -3 \\ 0 & 9 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 9 \\ 0 \end{pmatrix}.$$

Therefore, since Q is orthogonal

$$\begin{pmatrix} 6 & -3 \\ 0 & 9 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -\frac{2}{3} & -\frac{1}{3} & \frac{2}{3} \\ -\frac{2}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{2}{3} & -\frac{2}{3} \end{pmatrix} \begin{pmatrix} 3 \\ 9 \\ 0 \end{pmatrix} = \begin{pmatrix} -5 \\ 4 \\ -7 \end{pmatrix}.$$

The least-squares solution $\mathbf{x}_{LS}$ therefore has components $x_2 = \frac{4}{9}$ and $x_1 = \frac{1}{6}(-5 + 3x_2) = \frac{1}{6}(-5 + \frac{4}{3}) = -\frac{11}{18}$. By inspection the length of the minimum residual is $\|\mathbf{r}\|_2 = \|Q^T\mathbf{b} - R\mathbf{x}_{LS}\|_2 = 7$.