HW 5 Solutions

1. Here is the script which performs the iteration. Note that we have not coded a general Gauss-Seidel iteration here, rather the specific iteration for the particular sparse system at hand.

```
%  Set7Problem1: Script for Homework Set 7, Problem 1.
toler  = 1e-6;
n      = 100;
xexact = ones(n,1);
b      = ones(n,1); b(1) = 2; b(n) = 2;
x      = zeros(n,1);
ferr   = 1;
niter  = 0;

% Gauss-Seidel iteration for first tridiagonal system on p.122 of Sauer.
while ferr >= toler
     x(1) = (b(1) + x(2))/3;
     for j = 2:n-1
         x(j) = (b(j) + x(j-1) + x(j+1))/3;
     end
     x(n) = (b(n) + x(n-1))/3;
     niter = niter + 1;
     ferr = norm(x-xexact,inf);
end

% Compute residual and its norm without forming matrix.
r = zeros(n,1);
r(1) = b(1)-3*x(1)+x(2);
for k = 2:n-1
    r(k) = b(k) + x(k-1) - 3*x(k) + x(k-1);
end
r(n) = b(n)+x(n-1)-3*x(n);
berr = norm(r,inf);

% display output.
disp(['                                                   '])
disp(['       Gauss-Seidel                                '])
disp(['-------------------------------                    '])
disp(['n                    = ',num2str(n)                 ])
disp(['toler                = ',num2str(toler)             ])
disp(['iterations           = ',num2str(niter)             ])
disp(['backward error       = ',num2str(berr,'%4.2e')])
disp(['forward error        = ',num2str(ferr,'%4.2e')])
```

Running the script, we find the following output from the MATLAB® command line.

```
        Gauss-Seidel
-------------------------------
n                    = 100
toler                = 1e-06
iterations           = 20
backward error       = 1.19e-06
forward error        = 9.54e-07
```

2. Rearranged to be strictly diagonally dominant, the system is

$$\begin{aligned} 4u + 3w &= 0 \\ u + 4v &= 5 \\ v + 2w &= 2, \end{aligned}$$

assuming the variables are ordered as u then v then w. The Gauss-Seidel method is then

$$\begin{aligned} u^{(k+1)} &= -\tfrac{3}{4}w^{(k)} \\ v^{(k+1)} &= \tfrac{5}{4} - \tfrac{1}{4}u^{(k+1)} \\ w^{(k+1)} &= 1 - \tfrac{1}{2}v^{(k+1)}. \end{aligned}$$

With $(u^{(0)}, v^{(0)}, w^{(0)}) = (0, 0, 0)$, we find $(u^{(1)}, v^{(1)}, w^{(1)}) = (0, \frac{5}{4}, \frac{3}{8})$ and $(u^{(2)}, v^{(2)}, w^{(2)}) = (-\frac{9}{32}, \frac{169}{128}, \frac{87}{256})$.

3. For question 1, assume the matrix vector multiplication $\mathbf{y} = A\mathbf{x}$ is coded as follows.

```
y(1) = 3*x(1) - 1*x(2);                    % 3 flops
for k = 2:n-1
    y(k) = 3*x(k) - 1*x(k-1) - 1*x(k+1); % 5 flops per k
end
y(n) = 3*x(n) - 1*x(n-1)                   % 3 flops
```

Then the first entry of $A\mathbf{x}$ takes 3 FLOPS to compute (2 multiplications, 1 addition). Each of the middle $n-2$ entries takes 5 FLOPS (3 multiplications, 2 additions). The last entry takes 3 FLOPS (2 multiplications, 1 addition). So the total count is

$$5(n-2)+6 = 5n-4 = O(n).$$

However, this counting assumes that we view the multiplications by 1 as true flops. If $\mathbf{y} = A\mathbf{x}$ is (more reasonably) coded as

```
y(1) = 3*x(1) - x(2);                  % 2 flops
for k = 2:n-1
    y(k) = 3*x(k) - x(k-1) - x(k+1);  % 3 flops per k
end
y(n) = 3*x(n) - x(n-1)                 % 2 flops
```

then the counting is $3n-2$ total flops, again $O(n)$. For question 2, each entry of $B\mathbf{x}$ takes $n-1$ additions and n multiplications to compute (the cost of a "dot product"). There are n entries. So the total count is

$$2n^2 - n = O(n^2).$$

4. For **(a)** we require $y_k = p(x_k)$ for $k = 1, 2, \ldots, n$, that is

$$c_1 + c_2 x_k + c_3 x_k^2 + \cdots + c_n x_k^{n-1} = y_k, \qquad \text{for } k = 1, 2, \ldots, n.$$

As matrix multiplication between a row and column vector, the last equation reads

$$(1, x_k, x_k^2, \cdots, x_k^{n-1}) \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix} = y_k, \qquad \text{for } k = 1, 2, \ldots, n.$$

Collecting all rows into a matrix system, we then get $V\mathbf{c} = \mathbf{y}$, where

$$V = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix}$$

is the *Vandermonde matrix.* For **(b)** the following MATLAB® function and script perform the required task.

```
% Returns expansion coefficients that solve the n-node interpolation
% problem using a monomial basis.
%     Inputs: x (vector of nodes), y (vector of samples)
%     Output: c = V\y, where is V is the Vandermonde matrix.
% Can uncomment last lines to optionally display n and cond(V,inf).
% function c = interpvandmon(x,y)
  function c = interpvandmon(x,y)
  n = length(y); c = zeros(n,1); V = zeros(n,n);
  for k = 1:n
      V(:,k) = x.^(k-1);
  end
  c = V\y;
  disp(['n = ',num2str(n,'%02.0f'),' and cond(V,inf)=',num2str(cond(V,inf),'%1.4e')])
```

```
% Script: Set7Problem4b
% Makes plot and output required by Problem 4(b) of Homework Set 7.

% Data, nodes, and function values.
D4 = [[0 1]; [1 4]; [2 1]; [3 1]]; x = D4(:,1); y = D4(:,2);

% Form and solve Vandermonde monomial system.
c = interpvandmon(x,y);

% Dense set of points for plotting.
z = transpose(linspace(-0.5,3.5,500));

% Use Horner's to evaluate interpolating polynomial on z, and then plot p vs z.
p = c(1) + z.*(c(2) + z.*(c(3) + z*c(4)));
explot(z,p)
axis([-0.5 3.5 -6 6])
hold on
title('y = p(x) where p(x) interpolates data (0 1) (1 4) (2 1) (3 1)')
xlabel('x')
explot(x,y,'r.')
saveas(gcf,'Set7Problem4b.eps','epsc')
```

The output from the script is shown in Fig. 1, and indeed the plot is identical to Fig. 1, page 4, from the lecture `interp1`. For **(c)** we use the display option in `interpvandmon.m` to view the relevant condition

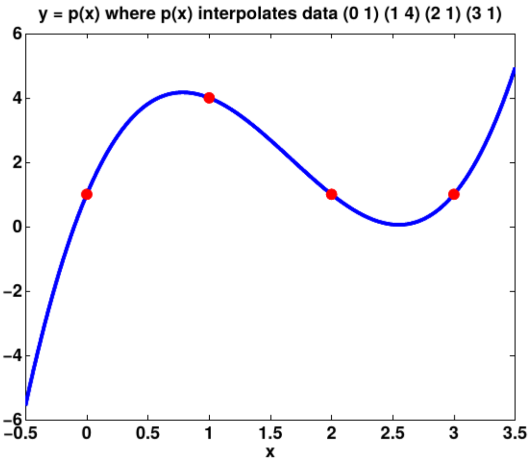


Figure 1: Plot for Problem 4(b).

numbers. The following script gives the results.

```
%  Set7Problem4c: Script for Homework Set 7, Problem 4(c).
format compact
warning('off','MATLAB:nearlySingularMatrix') % Turn off warning generated by large condition numbers.
ns = [5 10 15 20];
for k = 1:length(ns)
    n = ns(k);
    x = transpose(linspace(0,1,n));
    y = ones(n,1);                          % Note sample y values not used.
    c = interpvandmon(x,y);                 % Must uncomment last "display" line in interpvandmon.
end
```

Output from MATLAB® command line:

```
>> Set7Problem4c
n = 05 and cond(V,inf)=1.7067e+03
n = 10 and cond(V,inf)=4.8184e+07
n = 15 and cond(V,inf)=1.6055e+12
n = 20 and cond(V,inf)=5.0058e+16
```

HW 5 Solutions Part II

1. Given $D_3 = \{(0,-2),(2,1),(4,4)\}$, the nodal points are $x_1 = 0, x_2 = 2, x_3 = 4$ and the Lagrange basis is

$$\begin{aligned} L_1(x) &= \frac{(x-2)(x-4)}{(0-2)(0-4)} = +\frac{1}{8}(x-2)(x-4) \\ L_2(x) &= \frac{(x-0)(x-4)}{(2-0)(2-4)} = -\frac{1}{4}x(x-4) \\ L_3(x) &= \frac{(x-0)(x-2)}{(4-0)(4-2)} = +\frac{1}{8}x(x-2). \end{aligned}$$

Therefore, the interpolating polynomial is

$$\begin{aligned} p(x) &= -2 \cdot L_1(x) + 1 \cdot L_2(x) + 4 \cdot L_3(x) \\ &= -\frac{1}{4}(x-2)(x-4) - \frac{1}{4}x(x-4) + \frac{1}{2}x(x-2). \end{aligned}$$

Collecting like powers, we then find

$$\begin{aligned} p(x) &= -\frac{1}{4}(x-2)(x-4) - \frac{1}{4}x(x-4) + \frac{1}{2}x(x-2) \\ &= -\frac{1}{4}(x^2-6x+8) - \frac{1}{4}(x^2-4x) + \frac{1}{2}(x^2-2x) \\ &= -\frac{1}{4}(x^2-6x+8+x^2-4x-2x^2+4x) \\ &= -\frac{1}{4}(-6x+8) \\ &= +\frac{1}{2}(3x-4). \end{aligned}$$

To construct the same polnomial via the Newton method, we form the following divided difference table.

```
0|-2
 |     >  3/2
2| 1             >  0
 |     >  3/2
4| 4
```

So the polynomial is

$$p(x) = -2 \cdot \phi_1(x) + \frac{3}{2} \cdot \phi_2(x) + 0 \cdot \phi_3(x) = -2 + \frac{3}{2}(x-0) = \frac{1}{2}(3x-4), \tag{1}$$

the same as before.

2. The given and written functions are as follows.

```
  function c=interpnewt(x,y)
% function c=interpnewt(x,y)
% computes coefficients c of Newton interpolant through (x_k,y_k), k=1:length(x)
n=length(x);
for k=1:n-1
   y(k+1:n)=(y(k+1:n)-y(k))./(x(k+1:n)-x(k));
end
c=y;
```

```
  function p = hornernewt(c,x,z)
% function p = hornernewt(c,x,z)
% Uses Horner method to evaluate in nested form a polynomial defined
% by coefficients c and shifts x. Polynomial is evaluated at z.
n = length(c); % 1 + degree of polynomial.
p = c(n);
for k = n-1:-1:1
    p = p.*(z-x(k))+c(k);
end
```

These functions are used by the next script to make the plot of $L_3(x)$ shown in Fig. 1

```
% Script: Set8Problem2
% Makes plot required by Problem 2 of Homework Set 8.

% Interpolation points and data.
x = transpose([1:11]);
y = zeros(11,1); y(3) = 1;

% Get expansion coefficients of interpolating polynomial expressed in Newton basis.
c = interpnewt(x,y);

% Make a plot of the interpolating polynomial.
z = transpose(linspace(1,11,500));
p = hornernewt(c,x,z);
explot(z,p)
hold on
explot(x,y,'rx')
title('Lagrange basis function L_{3}(x)')
xlabel('x')
axis tight;
saveas(gcf,'Set8Problem2.eps','epsc') % Save figure as an eps.
```

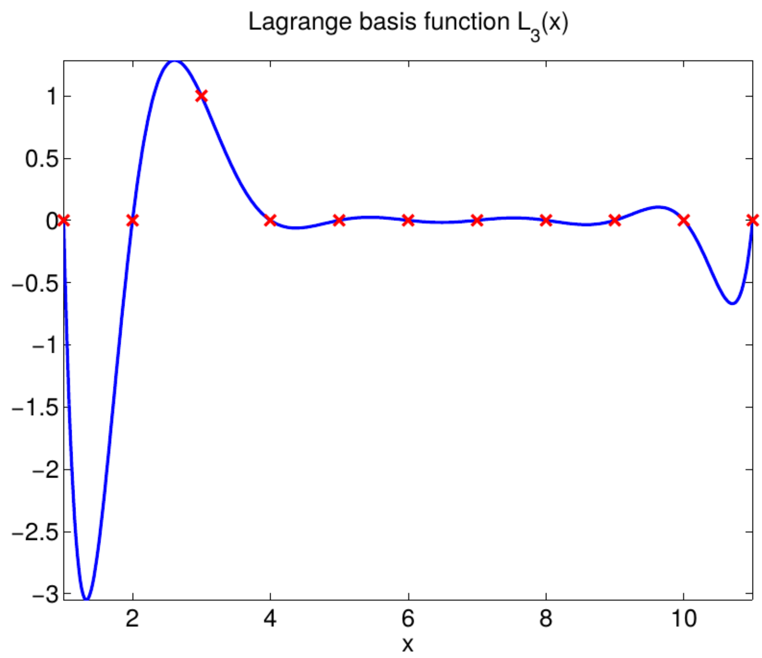


Figure 1: Plot for Problem 2.

3. The following script makes either plots for the uniform points or the Chebyshev points, based on flag `pts`.

```
% Script: Set8Problem3
% Makes plots for Problem 3 of Homework Set 8. Choose Uniform or Chebyshev points based on following flag.
%pts = 'Uniform';
pts = 'Chebyshev';

n = 11;
switch pts
  case 'Uniform'
    x = transpose(-4+8*[0:n-1]/(n-1));
  case 'Chebyshev'
    x = transpose(4*cos(pi*(2*[1:n]-1)/(2*n)));
  otherwise
    'No points selected.'
end

% Get y samples and construct coefficients for polynomial with respect to Newton basis.
y = 1./(x.^2+1);
c = interpnewt(x,y);

% Get arrays for plotting.
z = transpose(linspace(-4,4,500));
p = hornernewt(c,x,z);
g = ones(size(z))/factorial(n);
for k = 1:n
    g = g.*(z-x(k));
end

% Make the plots.
figure(4); clf;
subplot(3,1,1)
explot(z,p,'b-',x,y,'rx')
title([pts,' interpolant p(x) for 1/(x^2+1)'])
subplot(3,1,2)
explot(z,1./(z.^2+1)-p,'b-',x,zeros(size(x)),'rx')
title('Signed error 1/(x^2+1) - p(x)')
subplot(3,1,3)
explot(z,g,'b-',x,zeros(size(x)),'rx')
title('g(x) = (1/n!)\Pi_{k=1}^n (x-x_k)')
xlabel('x')
switch pts
  case 'Uniform'
    saveas(gcf,'Set8Problem3_unif.eps','epsc')
  case 'Chebyshev'
    saveas(gcf,'Set8Problem3_cheb.eps','epsc')
end
```

Two plots in Fig. 2 depict the results. For each case, we see at least qualitatively that $g(x)$ and the error $e(x) = f(x) - p(x)$ have the same shape. They are not exactly the same, of course, because $f^{(11)}(c_x)$ is not a constant function and $e(x) = g(x)f^{(11)}(c_x)$. (Note that $c_x = c(x)$ for the Chebyshev points would be different function than the c_x for the uniform points.) Also from the plots, it is evident that the magnitude $|f^{(11)}(c_x)|$ must be about 10^3 on the interval, since the y-scale of the plot for $e(x)$ is about this factor larger than the y-scale for the $g(x)$ plot (both for the uniform and Chebyshev examples).

4. The nth derivatives of $f(x) = 1/(x+5)$ is

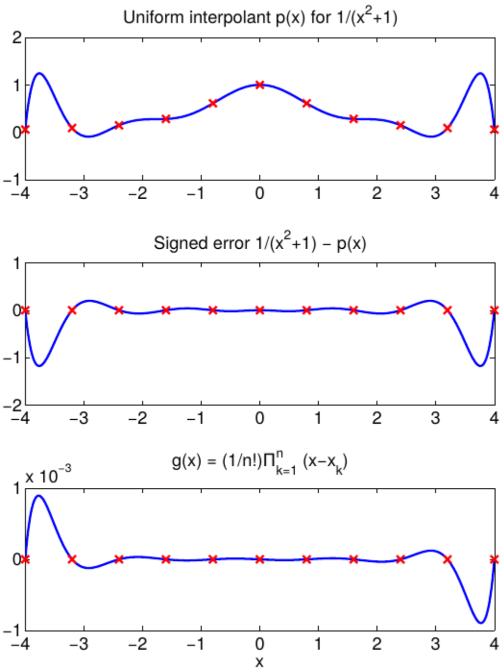
$$f^{(n)}(x) = \frac{(-1)^n n!}{(x+5)^{n+1}}, \tag{2}$$

as is easily checked by induction. Therefore, using the error formula for polynomial interpolation, we find

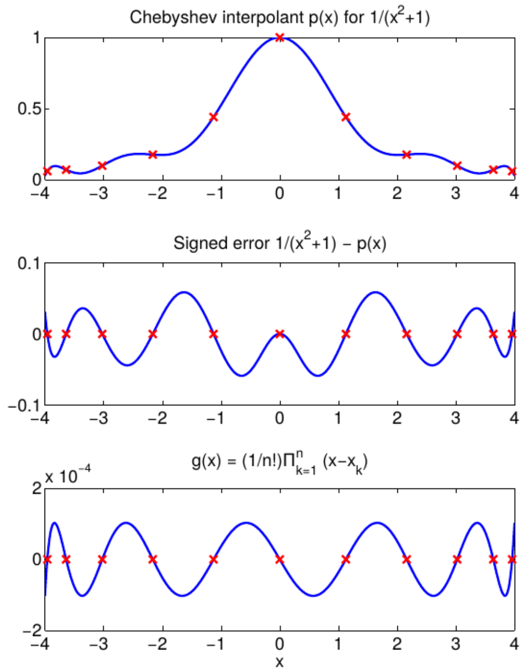
$$\frac{1}{x+5} - p_5(x) = \frac{(x-0)(x-2)(x-4)(x-6)(x-8)(x-10)}{n!} \cdot \frac{(-1)^6 6!}{(c_x+5)^7},$$

where $c_x \in [0, 10]$. Now, the worst case estimate for the magnitude of the 6th derivative is clearly

$$|f^{(6)}(c_x)| = \frac{6!}{(c_x+5)^7} \le \frac{6!}{5^7}, \qquad \text{for } c_x \in [0, 10]. \tag{3}$$



(a) Plots for uniform points.



(b) Plots for Chebyshev points.

Figure 2: Plots for Problem 2.

Therefore, we have the general bound

$$\left|\frac{1}{x+5} - p_5(x)\right| \leq \frac{|(x-0)(x-2)(x-4)(x-6)(x-8)(x-10)|}{5^7},$$

which may be applied to the points in question. Specifically, for $x = 1$, $f(x) = \frac{1}{6}$ and

$$\begin{aligned}\left|\frac{1}{6} - p_5(1)\right| &\leq \frac{|(1-0)(1-2)(1-4)(1-6)(1-8)(1-10)|}{5^7} \\ &= \frac{1 \cdot 1 \cdot 3 \cdot 5 \cdot 7 \cdot 9}{5^7} \\ &= \frac{945}{78125} = \frac{189}{15625} \simeq 1.2096 \times 10^{-2}.\end{aligned}$$

For $x = 5$, $f(x) = \frac{1}{10}$ and

$$\begin{aligned}\left|\frac{1}{10} - p_5(5)\right| &\leq \frac{|(5-0)(5-2)(5-4)(5-6)(5-8)(5-10)|}{5^7} \\ &= \frac{5 \cdot 3 \cdot 1 \cdot 1 \cdot 3 \cdot 5}{5^7} \\ &= \frac{9}{5^5} = \frac{9}{3125} \simeq 2.8800 \times 10^{-3}.\end{aligned}$$