HW#4 Part I

Solutions

1. Here is a sequence of row operations on $[A|\mathbf{b}]$ achieving upper triangular form.

$$\left[\begin{array}{rrr|r} 2 & 1 & -4 & -7 \\ 1 & -1 & 1 & -2 \\ -1 & 3 & -2 & 6 \end{array}\right] \xrightarrow[R3\to R3+\boxed{\frac{1}{2}}R1]{R2\to R2-\boxed{\frac{1}{2}}R1} \left[\begin{array}{rrr|r} 2 & 1 & -4 & -7 \\ 0 & -\frac{3}{2} & 3 & \frac{3}{2} \\ 0 & \frac{7}{2} & -4 & \frac{5}{2} \end{array}\right] \xrightarrow{R3\to R3+\boxed{\frac{7}{3}}R2} \left[\begin{array}{rrr|r} 2 & 1 & -4 & -7 \\ 0 & -\frac{3}{2} & 3 & \frac{3}{2} \\ 0 & 0 & 3 & 6 \end{array}\right].$$

Now by backward substitution, $x_3 = 2$, $-3x_2 = 3 - 6x_3 \implies x_2 = 3$, and $2x_1 = -7 - x_2 + 4x_3 \implies -1$. Therefore, $\mathbf{x} = (-1, 3, 2)^T$.

2. The total work for Gaussian elimination $\sim 2n^3/3$. If we triple n, that is send $n \to 3n$, then the total work changes as $2n^3/3 \to 2(3n)^3/3 = 27 \cdot (2n^3/3)$. That is, it increases by a factor of 27.

3. The requested MATLAB® functions are as follows.

```
  function [L,U]=naivege(A)
% function [L,U]=naivege(A)
% Computes the LU factorization without pivoting. A,L,U are n-by-n
% matrices with A=LU, L is unit lower triangular and U is upper triangular.
[n,n]=size(A);
for k=1:n-1                          % run over all columns k except last
  for j=k+1:n
    A(j,k)=A(j,k)/A(k,k);            % compute multipliers
    for l=k+1:n
      A(j,l)=A(j,l)-A(j,k)*A(k,l);   % eliminate entries
    end
  end
end
L=tril(A,-1)+eye(n,n); U=triu(A);
```

```
  function x=LTriSol(L,b)
% function x=LTriSol(L,b)
% Solves the system Lx=b, where L is unit lower triangular n-by-n, b is n-by-1
n=length(b);
x=zeros(n,1);
x(1)=b(1);
for j=2:n
  x(j) = b(j)-L(j,1:j-1)*x(1:j-1);
end
```

```
  function x=UTriSol(U,b)
% function x=UTriSol(U,b)
% Solves the system Ux=b, where U is upper triangular n-by-n, b is n-by-1
n=length(b);
x=zeros(n,1);
x(n)=b(n)/U(n,n);
for j=n-1:-1:1
  x(j) = (b(j)-U(j,j+1:n)*x(j+1:n))/U(j,j);
end
```

In the command line we then verify that the solution agrees with that found by hand in **1.**

```
>> format compact
>> A = [2 1 -4; 1 -1 1; -1 3 -2]; b = transpose([-7 -2 6]);
>> [L U] = naivege(A);
>> y = LTriSol(L,b);
>> x = UTriSol(U,y)
x =
    -1
     3
     2
```

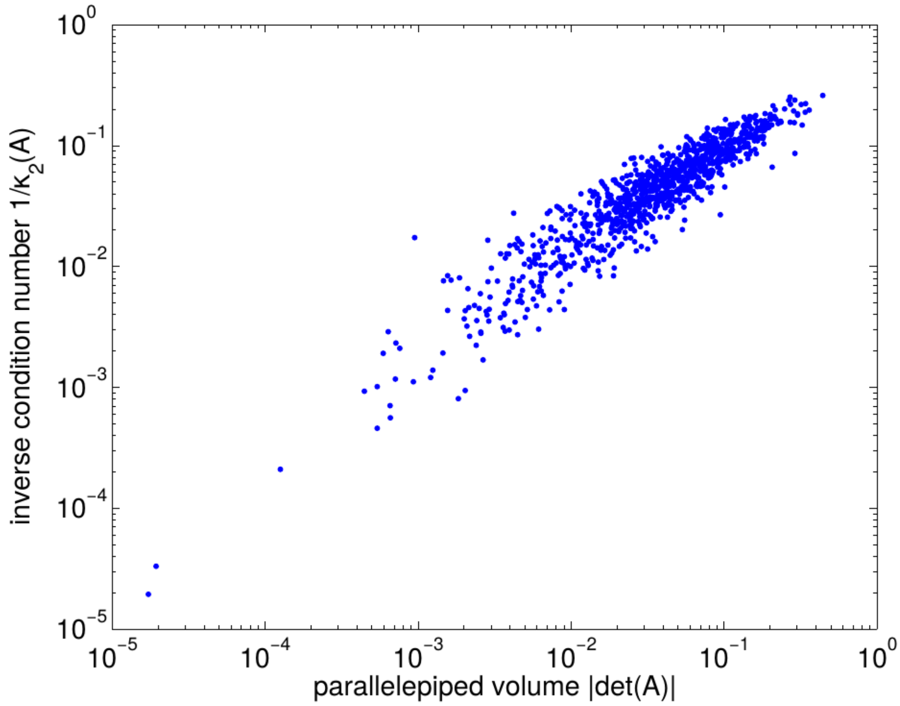
4. The script given below makes the required plot and generates the following output:

```
>> Set5Problem4
min volume = 5.4007e-05 and corresponding cond(A) = 1.0403e+04
max volume = 4.7534e-01 and corresponding cond(A) = 2.9605e+00
```

The line `A = A*diag(1./sqrt(sum(A.*A)))` in the script normalizes the columns of A (in the 2-norm). Indeed, the innermost operation `A.*A` is clearly componentwise squaring, whereas the next `sum` operation sums `A.*A` over each column. So `sqrt(sum(A.*A))` is a row vector whose entries correspond to the (Pythagorean or 2-norm) lengths of the columns of `A`. Notice that we compute `absDetA` $= |\det A|$ directly, since to get $\det A$ we would also need to compute $\det P = \pm 1$ (since P is a permutation matrix). The figure depicts the abscissa $|\det A|$ versus the ordinate $1/\kappa_2(A)$ (reciprocal two-norm condition number), and it suggests $\kappa_2(A) \sim |\det A|^{-1}$, at least for the class of matrices considered here. Therefore, when the determinant becomes small in absolute value, the condition number becomes large. **Please note that, unfortunately, due to a typo, the problem requested a plot of $\det A$ (no absolute value) versus $1/\kappa_2(A)$.**

```
% Script: Set4Problem4
% Makes plot and output required by Problem 4 of Homework Set 5.

ktotal = 1000;
% Integers kminvol and kmaxvol will keep of where min/max volume occurs.
minvol = 1e20; kminvol = 1; maxvol = 0; kmaxvol = 1;

% Preallocation of necessary memory.
volumes = zeros(ktotal,1); invcnds = zeros(ktotal,1);

for k = 1:ktotal
    A = rand(4);
    invcnds(k) = 1/cond(A);
    A = A*diag(1./sqrt(sum(A.*A)));
    [L U P] = lu(A);                % Note detA = det(P)*det(U)
    absDetA  = abs(prod(diag(U)));  % where detP = 1 or -1.
    volumek = absDetA;
    volumes(k) = volumek;
    if volumek > maxvol             % Simple minded approach here.
       maxvol = volumek;
       kmaxvol = k;
    end
    if volumek < minvol
       minvol  = volumek;
       kminvol = k;
    end
end

% Strings for output.
minVstr      = num2str(minvol,'%1.4e');
condAminVstr = num2str(1/invcnds(kminvol),'%1.4e');
maxVstr      = num2str(maxvol,'%1.4e');
condAmaxVstr = num2str(1/invcnds(kmaxvol),'%1.4e');

disp(['min volume = ',minVstr,' and corresponding cond(A) = ',condAminVstr])
```

```
disp(['max volume = ',maxVstr,' and corresponding cond(A) = ',condAmaxVstr])
exloglog(volumes,invcnds,'.')
ylabel('inverse condition number 1/\kappa_2(A)')
xlabel('parallelepiped volume |det(A)|')

% Save figure as an eps.
saveas(gcf,'Set5Problem4.eps','epsc')
```



1. For part **(a)** the magnitude $\|\mathbf{b} - A\mathbf{x}_c\|_\infty$ of the residual in the infinity norm is

$$\left\| \begin{pmatrix} 3 \\ 6.01 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ 2 & 4.01 \end{pmatrix} \begin{pmatrix} -10 \\ 6 \end{pmatrix} \right\|_\infty = \left\| \begin{pmatrix} 3 \\ 6.01 \end{pmatrix} - \begin{pmatrix} 2 \\ 4.06 \end{pmatrix} \right\|_\infty = \left\| \begin{pmatrix} 1 \\ 1.95 \end{pmatrix} \right\|_\infty = 1.95.$$

Clearly $\|\mathbf{b}\|_\infty = 6.01$, whence

$$\frac{\|\mathbf{b} - A\mathbf{x}_c\|_\infty}{\|\mathbf{b}\|_\infty} = \frac{1.95}{6.01} = \frac{195}{601} \simeq 0.3245.$$

The exact solution is clearly $\mathbf{x}_{\text{exact}} = (1,1)^T$, so the relative forward error is

$$\frac{\|\mathbf{x}_c - \mathbf{x}_{\text{exact}}\|_\infty}{\|\mathbf{x}_{\text{exact}}\|_\infty} = \left\| \begin{pmatrix} -10 \\ 6 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\|_\infty = \left\| \begin{pmatrix} -11 \\ 5 \end{pmatrix} \right\|_\infty = 11.$$

Therefore, we have **error magnification:** $11/(195/601) = 6611/195 \simeq 33.9$. For part **(c)** $\|\mathbf{b} - A\mathbf{x}_c\|_\infty$ is

$$\left\| \begin{pmatrix} 3 \\ 6.01 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ 2 & 4.01 \end{pmatrix} \begin{pmatrix} -600 \\ 301 \end{pmatrix} \right\|_\infty = \left\| \begin{pmatrix} 3 \\ 6.01 \end{pmatrix} - \begin{pmatrix} 2 \\ 7.01 \end{pmatrix} \right\|_\infty = \left\| \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\|_\infty = 1.$$

Since $\|\mathbf{b}\|_\infty = 6.01$ as before, the relative backward error is

$$\frac{\|\mathbf{b} - A\mathbf{x}_c\|_\infty}{\|\mathbf{b}\|_\infty} = \frac{1}{6.01} = \frac{100}{601} \simeq 0.1664.$$

Now the relative forward error is

$$\frac{\|\mathbf{x}_c - \mathbf{x}_{\text{exact}}\|_\infty}{\|\mathbf{x}_{\text{exact}}\|_\infty} = \left\| \begin{pmatrix} -600 \\ 301 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\|_\infty = \left\| \begin{pmatrix} -601 \\ 300 \end{pmatrix} \right\|_\infty = 601.$$

Therefore, we have **error magnification:** $601/(100/601) = 601^2/100 = 3612.01$. For part **(e)**, notice

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 4.01 \end{pmatrix} \implies A^{-1} = \frac{1}{4.01 - 4} \begin{pmatrix} 4.01 & -2 \\ -2 & 1 \end{pmatrix} = \begin{pmatrix} 401 & -200 \\ -200 & 100 \end{pmatrix}.$$

By inspection then, $\|A\|_\infty = 6.01$, $\|A^{-1}\|_\infty = 601$, so $\kappa_\infty(A) = \|A\|_\infty \cdot \|A^{-1}\|_\infty = 6.01 \cdot 601 = 3612.01$. Evidently, the error magnification in **(c)** realizes the largest possible value (the condition number).

2. The following MATLAB® function and script perform the required task.

```
% Cauchy matrix based on random vector rand(n,1).
% function C = CauchyMatrix(n);
  function C = CauchyMatrix(n);
  xi = repmat(rand(n,1),[1 n]);
  C = 1./(xi + transpose(xi));
```

```
% Script: Set6Problem2, CS/Math 375, UNM Spring 2011
% Makes plot and output required by Problem 2 of Homework Set 6.
ns    = [4 8 12 16];
ferrs = zeros(size(ns));      % Preallocate arrays for n,
berrs = zeros(size(ns));      % forward/backward errors,
conds = zeros(size(ns));      % and condition number.
for k = 1:length(ns)
    n        = ns(k);                   % Get n
    zexact   = ones(n,1);               % Get exact solution.
    C        = CauchyMatrix(n);         % Create matrix.
    conds(k) = cond(C,inf);             % Compute cond. num.
    b        = C*zexact;                % Create Rhs.
```

```
    zc       = C\b;                           % Solve system!
    ferrs(k) = norm(zexact-zc,inf);           % Compute errors, ||zexact||_inf = 1.
    berrs(k) = norm(b-C*zc,inf)/norm(b,inf);
end
% Make table of results.
FID = fopen('Set6Problem2-Table','w');
fprintf(FID,'-----------------------------------------------------------\n');
fprintf(FID,'|  n  |  f.errors  |  b.errors  |  m.factor  |  cond.num  |\n');
fprintf(FID,'-----------------------------------------------------------\n');
for k = 1:length(ns)
  fprintf(FID,'| %3.0f | %1.4e | %1.4e | %1.4e | %1.4e |\n', ...
          ns(k),ferrs(k),berrs(k),ferrs(k)/berrs(k),conds(k));
end
fprintf(FID,'-----------------------------------------------------------\n');
fprintf(FID,'f.errors, b.errors: relative forward and backward errors\n');
fprintf(FID,'m.factor, cond.num: magnification factor and infinity norm condition number\n');
```

The output table is as follows.

```
-----------------------------------------------------------
|  n  |  f.errors  |  b.errors  |  m.factor  |  cond.num  |
-----------------------------------------------------------
|   4 | 4.2077e-14 | 4.0078e-17 | 1.0499e+03 | 4.7000e+03 |
|   8 | 1.2445e-03 | 3.6004e-17 | 3.4565e+13 | 7.0556e+13 |
|  12 | 3.8399e+00 | 1.0968e-16 | 3.5011e+16 | 1.2123e+18 |
|  16 | 6.1381e+02 | 5.3444e-15 | 1.1485e+17 | 3.4508e+18 |
-----------------------------------------------------------
f.errors, b.errors: relative forward and backward errors
m.factor, cond.num: magnification factor and infinity norm condition number
```

Discussion. The condition number $\kappa_\infty(C)$ is always greater than the magnification factor, consistent with its interpretation as the largest possible magnification factor over all right-hand sides $\mathbf{b}$. Nevertheless, for these solves the magnification factor is close to the largest possible one, and becomes very large with increased n. For n large $\mathbf{z}_c$ is of poor quality (large forward error), despite yielding (as column three of the table shows) a residual $\mathbf{r} = \mathbf{b} - C\mathbf{z}_c$ which has a maximum component $\|\mathbf{r}\|_\infty \simeq \|\mathbf{b}\|_\infty \varepsilon_{\text{mach}} = O(\varepsilon_{\text{mach}})$ that is nearly machine precision in size ($\|\mathbf{b}\|_\infty$ is typically about 10^2 at most). Notice also for $n = 4, 8$ that the number of correct digits in the forward error is roughly $\log_{10}(1/\varepsilon_{\text{mach}}) - \log_{10}\kappa_\infty(C) \simeq 16 - \log_{10}\kappa_\infty(C)$, as expected. For n large C is extremely ill-conditioned, and the numerical solution for $n = 12, 16$ has no correct digits.