Abelardo Riojas
ISC 4220C Spring 2020
Professor Shanbhag

Lab 4 Report

1. The Hilbert matrix is defined by $a_{ij} = 1/(i + j - 1)$.
   - Construct a 4×4 Hilbert matrix, and find its condition number using the row-sum norm

	1	2	3	4
1	1	0.5000	0.3333	0.2500
2	0.5000	0.3333	0.2500	0.2000
3	0.3333	0.2500	0.2000	0.1667
4	0.2500	0.2000	0.1667	0.1429

     $||A|| = 1 + ½ + ⅓ + ¼ = 2.0833...$
     $||A^{-1}|| = 240 + 2700 + 6480 + 4200 = 13620$
     cond(A) = (2.08333)*(13620) = 28375
   - How many digits of precision will be lost due to ill-conditioning on a double precision machine?

     On a double precision machine with t = 16, the solution is only valid to 16 - $\log_{10}$(cond(A)) = 11.54 ~ 11 digits so 5 digits of precision are lost.
   - Scale each row of the matrix by dividing each row by its largest element. Repeat the steps above for this scaled matrix.

     $||A_{scaled}|| = 1 + 8/10 + ⅔ + 4/7 = 3.03809524…$
     $||A^{-1}_{scaled}|| = 240 + 1350 + 2160 + 1050 = 4800$
     cond(A_{scaled}) = 14582.8
     On a double precision machine with t = 16, the solution is only valid to 16 - $\log_{10}$(cond(A_{scaled})) = 11.83 ~ 12 digits so 4 digits of precision are lost.
2. True or False: If ||A|| is small, then the condition number of A is small. Explain.
   False. Matrices can have large norms of their inverses, which would lead to a large condition number.
3. True or False: The condition numbers of A and A^{-1} are the same. Explain.
   True. Since the condition number of A is $||A|| * ||A^{-1}||$, the condition number of A^{-1} would be $||A^{-1}|| * ||A^{-1-1}||$. A^{-1-1} is just A. Meaning the equivalence holds by the commutative property.

Use the Matlab intrinsic function `[L, U, P] = lu(A)` to compute the matrix inverse of

$$\mathbf{A} = \begin{bmatrix} 6 & 15 & 55 \\ 15 & 55 & 225 \\ 55 & 225 & 979 \end{bmatrix}.$$

```
A = [6 15 55; 15 55 255; 55 255 979];

[L, U, P] = lu(A);

Ainv = zeros(3,3);
I = eye(3,3);
for i = 1:3
    Ainv(:,i) = (inv(P)*L*U)\I(:,i);
end
Ainv*A
```

Result:

```
ans =

   1.000000000000000   0.000000000000002   0.000000000000007
  -0.000000000000000   1.000000000000000                   0
   0.000000000000000  -0.000000000000000   1.000000000000000
```

(ii) (from **2015 Exam**) The following system of equations arises in circuit modeling:

$$\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{bmatrix} \mathbf{V} = \begin{bmatrix} -2.01 \\ -0.01 \\ -0.01 \\ -0.01 \\ -0.01 \\ -3.01 \end{bmatrix}$$

We want to solve the problem using (a) Gauss-Seidel, and (b) successive over-relaxation (SOR), with an initial guess of $\mathbf{V}^{(0)} = [0\ 0\ 0\ 0\ 0\ 0]^T$.

Stop the iterations once the convergence criterion $||\mathbf{V}^{(k+1)} - \mathbf{V}^{(k)}|| \le 10^{-5}||\mathbf{V}^{(k+1)}||$ is satisfied.

Show the first three iterations using Gauss-Seidel, including the value of the relative error $||V^{(k+1)} - V^{(k)}||/||V^{(k+1)}||$.

x_{k+1}	relative error	iter
1.005000000000000 0.507500000000000 0.258750000000000 0.124375000000000 0.067187500000000 1.538593750000000	1	1
1.258750000000000 0.763750000000000 0.449062500000000 0.253125000000000 0.900859375000000 1.955429687500000	0.385705658202481	2
1.386875000000000 0.922968750000000 0.593046875000000 0.741953125000000 1.353691406250000 2.181845703125000	0.233042012327099	3

How many iterations of Gauss-Siedel are required to solve the system?

My code took 58 iterations of G-S to solve this system.

For the following values of ω, report the number of iterations required for SOR to converge: (a) 1.2, (b) 1.4, and (c) 1.6.

ω = 1.2, iter = 38
ω = 1.4, iter = 18
ω = 1.6, iter = 28

```
A = [
    -2 1 0 0 0 0;
    1 -2 1 0 0 0;
    0 1 -2 1 0 0;
    0 0 1 -2 1 0;
    0 0 0 1 -2 1;
    0 0 0 0 1 -2;];

v = [-2.01; -.01; -.01; .01; -.01; -3.01];
v0 = zeros(6,1);
b = zeros(6,1);

L = zeros(6,6);
U = zeros(6,6);
D = zeros(6,6);

for i = 1:6
    for j = 1:6
        if i == j
            D(i,j) = A(i,j);
        elseif i < j
            U(i,j) = A(i,j);
        else
            L(i,j) = A(i,j);
        end
    end
end
 %gauss sidel
flag = 0;
xk = v0;
iter = 0;
while flag ~= 1
    xkp1 = inv((L+D))*(v - U*xk)
    relerror = norm(xkp1 - xk)/norm(xkp1)
    if norm(xkp1 - xk) <= .00001
        flag = 1
    end
    xk = xkp1;
    iter = iter+1
end
```

```
%%SOR
flag = 0;
xk = v0;
iter = 0;
w = 1.6;


while flag ~= 1
    iter = iter+1
    xkp1 = w*v - (w*U + (w-1)*D)*xk;
    xkp1 = inv(D+ w*L)*xkp1;
    if norm(xkp1 - xk) <= .00001
        flag = 1;
    end
    xk = xkp1;
end
```