Jacob Manning HW2

1. see code
Forward error is always worse
GE w/o pivoting was always the worst.
GEPP and QR were very stable
Crammer's rule and A^{-1} were less stable

2. $A\hat{v} + \Delta A\hat{v} = \hat{\lambda}\hat{v} \Rightarrow A\hat{v} - \hat{\lambda}\hat{v} = \Delta A\hat{v} \Rightarrow$

$\|A\hat{v} - \hat{\lambda}\hat{v}\|_2 = \|\Delta A\hat{v}\|_2 \leq \|\Delta A\|_2 \|\hat{v}\|_2 \Rightarrow$

$$\frac{\|A\hat{v} - \hat{\lambda}\hat{v}\|_2}{\|\hat{v}\|_2} \leq \|\Delta A\|_2$$

$$\Delta A = \frac{(\hat{\lambda}\hat{v} - A\hat{v})\hat{v}^T}{\hat{v}^T\hat{v}}$$ Note, ΔA is a rank 1 matrix let's check

$$\|\Delta A\|_2 = \frac{1}{\|\hat{v}\|_2^2}\|(\hat{\lambda}\hat{v} - A\hat{v})\hat{v}^T\|_2$$ By a theorem given in class it follows

$$= \frac{1}{\|\hat{v}\|_2^2}\|\hat{\lambda}\hat{v} - A\hat{v}\|_2\|\hat{v}\|_2$$

$$= \frac{\|\hat{\lambda}\hat{v} - A\hat{v}\|_2}{\|\hat{v}\|_2} \checkmark$$

3. Let a_k be the row containing the pivot in each case.

$$\begin{bmatrix} a_1 \\ \vdots \\ a_k \\ \vdots \\ a_n \end{bmatrix}$$

Scale each $a_{kk} = 1$, Each element $a_{kj} \leq a_{kk} \therefore \left|\frac{a_{kj}}{a_{kk}}\right| \leq 1$
At worse case in each step $\left|\frac{a_{kj}}{a_{kk}}\right| = 1$ so the growth at each k-step would be equal to 2.

3cont After $n-1$ elementation steps of this worse case scenario $p_n = \prod_{i=1}^{n-1} 2 = 2^{n-1}$

see code

There were no points that $p_n > \sqrt{n}$. This is unlikely.

4a. $A = R_1^T A_1 R_1$ is a spd matrix by prop of spd matrices, we know a product of spd matrices is spd. It follows R_1^T, A_1, and R_1 are all spd. Similarly, a principle submatrix of a spd matrix is spd. Thus, $K - \frac{ww^T}{a_{11}}$ is spd

$$\|A\|_2 = \sup_{\|x\|=1} \|Ax\|_2 = \sup_{\|x\|=1} \|R^T R x\|_2 = \sup_{\|x\|=1} \langle R^T R X, X \rangle$$
$$= \sup_{\|x\|=1} X^T R^T R x = \sup_{\|x\|=1} \|Rx\|_2^2 = \|R\|_2^2$$

$\therefore \|R\|_2 = \sqrt{\|A\|_2}$. This is because $\|R\|_2^2 \sim O(\|A\|_2)$

Thus $\frac{\|\Delta A\|_2}{\|R\|_2^2} = \frac{\|\Delta A\|_2}{\|A\|_2}$ so this leads to backward stability

b. Since A is cdd $|C_{ii}| \geq |V_i| + \sum_{j \neq i} |C_{ij}|$.

Consider $\sum_{j \neq i} |C_{ji} - \frac{V_i W_i}{\alpha}| \leq \sum_{j \neq i} |C_{ji} + |V_i| - \frac{V_i W_i}{\alpha}|$

$\leq \sum_{j \neq i} |C_{ji}| + ||V_i| - \frac{V_i W_i}{\alpha}|$

$\leq \sum_{j \neq i} |C_{ji}| + |V_i|(|1 - \frac{|W_i|}{\alpha}|)$

Since A is cdd $1 - \frac{|W|}{|\alpha|} > 0$ $\leq |C_{ii}| - |\frac{VW^T}{\alpha}|$

$\leq |C_{ii} - \frac{VW^T}{\alpha}|$ $\boxtimes$

Thus no pivoting is necessary

```
%problem 1
list=[];

n=9;
list=[list lisn(n)];

n=19;
list=[list lisn(n)];

n=29;
list=[list lisn(n)];

n=39;
list=[list lisn(n)];

%parts a-d
disp(list(1:4,:))

%part e
disp(list(5,:))

%problem 3
for j = 5:9
    i = 0;
    n = 2^j;

    for k = 1:1000
        Ap=randn(n,n)/sqrt(n);
        ak = luFactor(Ap);
        am = max(abs(Ap(:)));
        p = ak/am;

        if p > sqrt(n)
            i=i+1;
        end
    end

    disp(i/k)

end

function l=lisn(n)
    l=zeros(5,2);
    u=linspace(-1,1,n+1);
    A=vander(u);
    x=repelem(1,n+1)';
    b=A*x;
    [Q,R]=qr(A);

    x1=A\b;
    x2=R\(Q'*b);
    x3=cram(b,A);
```

```
    x4=inv(A)*b;
    x5=GEnoP(A, b);

    [fe1,be1]=err(x1,x,b,A);
    [fe2,be2]=err(x2,x,b,A);
    [fe3,be3]=err(x3,x,b,A);
    [fe4,be4]=err(x4,x,b,A);
    [fe5,be5]=err(x5,x,b,A);

    l(1,:)=[fe1,be1];
    l(2,:)=[fe2,be2];
    l(3,:)=[fe3,be3];
    l(4,:)=[fe4,be4];
    l(5,:)=[fe5,be5];
end

function [fe, be]=err(xh,x,b,A)
    fe=norm(xh-x)/norm(x);
    be=norm(b-A*xh)/(norm(A)*norm(xh));
end

function X=cram(b,A)
    % Determinant of coefficient matrix
    det_A = det(A);

    % Solution vector
    X = zeros(size(b));

    % Cramer's rule
    for i = 1:size(A, 1)
        % Replace the ith column of A with B
        A_i = A;
        A_i(:, i) = b;

        % Calculate the determinant of A_i
        det_A_i = det(A_i);

        % Calculate the ith element of X
        X(i) = det_A_i / det_A;
    end
end

function x = GEnoP(A, b)
    B = [A, b];
    [n, m] = size(B);
    % Start with Forward sub
    for i = 1:n
        B(i, :) = B(i, :) ./ B(i, i);
        for k = i+1:n
            B(k, :) = (-B(i, :) * B(k, i)) + B(k, :);
        end
    end
    % Back Substitution
    for j = n-1:-1:1
```

```
        for z = j+1:1:n
            B(j, :) = (-B(z, :) * B(j, z)) + B(j, :);
        end
    end
    x = B(:, end);
end

function ak = luFactor(A)
    % Get the size of A
    [n, ~] = size(A);

    % Initialize L, U, and P
    L = eye(n);
    U = A;
    P = eye(n);
    ak = 0;

    for k = 1:n-1
        % Find the maximum element in the current column
        [~, i] = max(abs(U(k:n, k)));
        if max(abs(U(k:n, k))) > ak
            ak = max(abs(U(k:n, k)));
        end
        i = i + k - 1;

        % Swap rows i and k in U, and record the permutation in P
        U([k, i], :) = U([i, k], :);
        P([k, i], :) = P([i, k], :);
    end
end
```

```
Warning: Matrix is close to singular or badly scaled. Results may be
inaccurate.
RCOND =  1.443640e-19.
Warning: Matrix is close to singular or badly scaled. Results may be
inaccurate.
RCOND =  1.443640e-19.
  Columns 1 through 3

   0.000000000000045   0.000000000000000   0.000000004655333
   0.000000000000046   0.000000000000000   0.000000000379934
   0.000000000000030   0.000000000000010   0.000000000658849
   0.000000000000080   0.000000000000016   0.000000001934529

  Columns 4 through 6

   0.000000000000000   0.000124504487067   0.000000000000000
   0.000000000000000   0.000211291350386   0.000000000000000
   0.000000000104980   0.000045119848938   0.000011784214817
   0.000000000057591   0.000090467439630   0.000004510925470

  Columns 7 through 8

   1.979754795496287   0.000000000000000
```

```
   4.401487120361307   0.000000000000000
   2.189867236640283   0.298812855759016
   2.413950523261177   0.188973324482738

   1.0e+27 *

  Columns 1 through 3

   0.000000000000000   0.000000000000000   0.000000000000000

  Columns 4 through 6

   0.000000000000000   0.000000000005040   0.000000000000000

  Columns 7 through 8

   3.653782428916880   0.000000000000000

     0

     0

     0

     0

     0
```

Published with MATLAB® R2023b