CPSC 302 - Assignment 3

Tristan Rice, q7w9a, 25886145

1. Linear Algebra

1.a

$$||A||_1 = max_j(a+b, b+a+b, \ldots, b+a) = a+2b$$
$$||A||_\infty = max_j(a+b, b+a+b, \ldots, b+a) = a+2b$$

$$||A||_2 = \sqrt{\lambda_{max}(A^T A)} = \sigma_{max}(A)$$

The largest singular value is the square root of the largets eigen vector.

$$\lambda_{max} = a+2b$$
$$\sigma_{max} = \sqrt{a+2b}$$

$$||A||_2 = \sqrt{a+2b}$$

1.b

The matrix A is defined to be symmetric.

Claim: If A is strictly diagonally dominant then it is symmetric positive definite.

$$v \in \mathbb{R}^n$$

$$Av = \begin{bmatrix} av_1 + bv_2 \\ bv_1 + av_2 + bv_3 \\ bv_2 + av_3 + bv_4 \\ \ldots \\ bv_{n-2} + av_{n-1} + bv_n \\ bv_{n-1} + av_n \end{bmatrix}$$

$$v^T Av = v_1(av_1 + bv_2) + v_2(bv_1 + av_2 + bv_3) + v_3(bv_2 + av_3 + bv_4) + \ldots v_{n-1}(bv_{n-2} + av_{n-1} + bv_n) + v_n(bv_{n-1} + av_n)$$

$$v^T Av = av_1^2 + bv_1v_2 + bv_1v_2 + av_2^2 + bv_3v_2 + bv_2v_3 + av_3^2 + bv_4v_3 + \ldots bv_{n-2}v_{n-1} + av_{n-1}^2 + bv_nv_{n-1} + bv_{n-1}v_n + av_n^2$$

$$v^T Av = av_1^2 + 2bv_1v_2 + av_2^2 + 2bv_2v_3 + av_3^2 + 2bv_3v_4 + \ldots av_{n-1}^2 + 2bv_{n-1}v_n + av_n^2$$

Since A is strictly diagonally dominant, that means $a > 2b$. Thus, $v^T Av > 0$ and A is symmetric positive definite.

1.c

We first need to figure out for which j is the maximum and minimum of $\cos(\frac{\pi j}{n+1}$. a, b are both positive so they don't affect the sign.

$$\cos(\frac{\pi}{n+1}) \approx 1$$
$$\cos(\frac{n\pi}{n+1}) \approx -1$$

$$\max_{j=1,\ldots,n} = a + 2b$$
$$\min_{j=1,\ldots,n} = a - 2b$$

$$\kappa_2(A) = \frac{a+2b}{a-2b}$$

2. Tridiagonal Systems of Equations

2.a

Seems to work just fine.

2.b

We find $||v - u||_\infty = 2.0562 * 10^{-5}$. The computation of v matches the solution that matlab's operator returns.

3. Cholesky Decomposition

3.a

If you set

$$A = \begin{bmatrix}1\end{bmatrix}, \alpha = -1, B = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$$

$$\begin{bmatrix}0 & 1\end{bmatrix} B \begin{bmatrix}0\\1\end{bmatrix} = -1$$

We see that the scalar α is negative, it violates the definition of positive definite: $z^T M z$ is positive for every non-zero column vector z. Thus α must be positive.

If we construct the reverse

$$A = \begin{bmatrix}-1\end{bmatrix}, \alpha = 1, B = \begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}$$

$$\begin{bmatrix}1 & 0\end{bmatrix} B \begin{bmatrix}1\\0\end{bmatrix} = -1$$

We see that if A is not positive definite B is not positive definite. Thus, A must be positive definite.

3.b

$$A = \begin{bmatrix} A & a \\ a^T & \alpha \end{bmatrix}$$

Let's define G to be the Cholesky decomposition of A.

We want to find a new factorization of B in the form

$$\begin{bmatrix} G & 0 \\ h^T & i \end{bmatrix}$$

$$\begin{bmatrix} A & a \\ a^T & \alpha \end{bmatrix} = \begin{bmatrix} G & 0 \\ h^T & i \end{bmatrix} \begin{bmatrix} G^T & h \\ 0 & i \end{bmatrix} = \begin{bmatrix} GG^T & Gh \\ G^T h^T & h^T h + i^2 \end{bmatrix}$$

Thus,

$$a = Gh, \alpha = h^T h + i^2$$

Solving for h,

$$h = G^{-1} a$$

Solving for i,

$$i^2 = \alpha - (G^{-1}a)^T (G^{-1}a)$$
$$i = \sqrt{\alpha - (G^{-1}a)^T G^{-1} a}$$

Thus, the Cholesky factorization of B is,

$$chol(B) = \begin{bmatrix} G & 0 \\ (G^{-1}a)^T & \sqrt{\alpha - (G^{-1}a)^T G^{-1} a} \end{bmatrix}$$

3.c

$$B = \begin{bmatrix} 4 & 2 & 2 \\ 2 & 6 & 2 \\ 2 & 2 & 5 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 1 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 0 & 0 \\ 1 & \sqrt{5} & 0 \\ 1 & \frac{1}{\sqrt{5}} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{19}{5} \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \\ 0 & \sqrt{5} & \frac{1}{\sqrt{5}} \\ 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 0 & 0 \\ 1 & \sqrt{5} & 0 \\ 1 & \frac{1}{\sqrt{5}} & \sqrt{\frac{19}{5}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \\ 0 & \sqrt{5} & \frac{1}{\sqrt{5}} \\ 0 & 0 & \sqrt{\frac{19}{5}} \end{bmatrix}$$

$$C = \begin{bmatrix} 2 & 0 & 0 \\ 1 & \sqrt{5} & 0 \\ 1 & \frac{1}{\sqrt{5}} & \sqrt{\frac{19}{5}} \end{bmatrix}$$

4. LDL^T Decomposition

$$A = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 3 & 0 \\ 1 & 0 & 2 \end{bmatrix}$$

$$U = \begin{bmatrix} 4 & -1 & 1 \\ 0 & \frac{11}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{7}{4} \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{4} & 1 & 0 \\ \frac{1}{4} & 0 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 4 & -1 & 1 \\ 0 & \frac{11}{4} & \frac{1}{4} \\ 0 & 0 & \frac{19}{11} \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{4} & 1 & 0 \\ \frac{1}{4} & \frac{1}{11} & 1 \end{bmatrix}$$

$$U = DL^T$$

$$D = \begin{bmatrix} 4 & 0 & 0 \\ 0 & \frac{11}{4} & \frac{1}{4} \\ 0 & 0 & \frac{19}{11} \end{bmatrix}$$

5. Hessenberg Matrix

5.a

```
A =

   1   4   2   3   9
   3   4   1   7   9
   0   2   3   4   9
   0   0   1   3   4
   0   0   0   4   5

L =

   1.00000   0.00000   0.00000   0.00000   0.00000
   3.00000   1.00000   0.00000   0.00000   0.00000
   0.00000  -0.25000   1.00000   0.00000   0.00000
   0.00000   0.00000   0.57143   1.00000   0.00000
   0.00000   0.00000   0.00000   4.00000   1.00000

U =

    1.00000    4.00000    2.00000    3.00000    9.00000
    0.00000   -8.00000   -5.00000   -2.00000  -18.00000
    0.00000    0.00000    1.75000    3.50000    4.50000
    0.00000    0.00000    0.00000    1.00000    1.42857
    0.00000    0.00000    0.00000    0.00000   -0.71429

L * U =

   1   4   2   3   9
   3   4   1   7   9
   0   2   3   4   9
   0   0   1   3   4
   0   0   0   4   5
```

5.b

L has 1 values along the diagonal, and first subdiagonal is non-zero, but everything else is zero.

5.c

$$Ax = b, A = LU, LUx = b$$

If we solve this by first doing the LU decomposition as specified above and then doing backwards substitution and then forward subsitution we see that:

- LU decomposition $= O(n^2)$
- Forward subsitution with $L = O(n)$ since there's only one element other than the pivot per column.
- Backwards subsitution with $U = O(n^2)$

Thus, the number of operations is $= O(n^2)$.

5.d

The sparsity pattern changes with partial pivoting. We can see this if we use MatLab's `lu` function.

```
[L, U] = lu(A)

L =

   0.33333   1.00000   0.00000   0.00000   0.00000
   1.00000   0.00000   0.00000   0.00000   0.00000
   0.00000   0.75000   1.00000   0.00000   0.00000
   0.00000   0.00000   0.57143   0.25000   1.00000
   0.00000   0.00000   0.00000   1.00000   0.00000

U =

   3.00000   4.00000   1.00000   7.00000   9.00000
   0.00000   2.66667   1.66667   0.66667   6.00000
   0.00000   0.00000   1.75000   3.50000   4.50000
   0.00000   0.00000   0.00000   4.00000   5.00000
   0.00000   0.00000   0.00000   0.00000   0.17857
```

Since partial pivoting operates row by row and there's only one entry below the pivot per column that means each row only can be switched with the one below it. Thus, there can be some 1 values on the diagonal above the middle.

```
% 2.a

% The input matrix is n x n. b is in R^n
n = 10000;
% construct the 3 diagaonals
d1 = - (2:n);
d2 = 3 * (1:n);
d3 = - (1:n-1);
% pick b to be something
b = 1:n;

x = solvetridiagonal(b, d1, d2, d3);

% 2.b



% n is number of steps
n = 100;
% h is step size
h = 1/n;

% create the tridiagonal matrix representing our problem
d1 = repmat(-1/h^2, 1, n-1);
% change the last value on the lower diagonal
d1(1, n-1) = -2/h^2;
d2 = repmat(2/h^2, 1, n);
d3 = repmat(-1/h^2, 1, n-1);

% create the b values
b = g((1:n) * h);

% solve for v of Dv=b
v = solvetridiagonal(b, d1, d2, d3);

% check against matlab
d = full(gallery('tridiag', n, -1/h^2, 2/h^2, -1/h^2));
% change the last value on the lower diagonal
d(n, n-1) = -2/h^2;
vright = (d\(b'))';
% likely will vary slightly, but not greatly
assert(norm(v - vright, 2) < 1e-10);


% compute actual answer
u = sin(pi/2 * (1:n) * h);

% compute norm

norm(v'-u', inf)
norm(vright'-u', inf)

% Define the function g from the textbook.
function gt = g(t)
  gt = (pi/2)^2*sin(pi/2 * t);
end

% solvetridiagonal solves the tridiagonal matrix formed by the lower
% diagonal d1, middle diagonal d2, upper diagonal d3 and b.
% Dx = b
function x = solvetridiagonal(b, d1, d2, d3)
  % n is the length of the center diagonal
  n = length(d2);

  % Scan down to reduce the lower diagonal to zero and the pivots to 1.
  for i = 1:n
    % Rescale the row so the pivot is 1.
    a = d2(i);
    d2(i) = d2(i) / a;
    if i < n
      d3(i) = d3(i) / a;
```

```
    end
    b(i) = b(i) / a;

    % Cancel out all non zero elements in the column except for the pivot.
    if i < n
      factor = -d1(i);
      d1(i) = d1(i) + factor;
      d2(i+1) = d2(i+1) + factor * d3(i);
      b(i+1) = b(i+1) + factor * b(i);
    end
  end

  % Scan back upwards and reduce the upper diagonal to zero and thus solving for x.
  for i = flip(2:n)
    factor = -d3(i-1);
    d3(i-1) = d3(i-1) + factor;
    b(i-1) = b(i-1) + factor * b(i);
  end

  % Finally set x = b
  x = b;
end
```

```
0;

function [L, U] = hessenberg(A)
  n = rows(A);
  % A is closest to U
  U = A;
  L = eye(n);

  for i = 1:(n-1)
    % Calculate the factor to multiply to the ith row before adding to the ith+1
    % row.
    factor = -U(i+1, i)/U(i, i);
    L(i+1, i) = -factor;
    % For a given row, there is only one element below the pivot by definition.
    % Thus we only need to update a single row.
    for j = i:n
      U(i+1, j) += U(i, j) * factor;
    end
  end
end

A = [1 4 2 3 9; 3 4 1 7 9; 0 2 3 4 9; 0 0 1 3 4; 0 0 0 4 5]
[L, U] = hessenberg(A)

% verify our output is correct
newA = L * U
assert(newA == A)
```