AM117 Sample Solutions HW #4

Free Advice from the TA:

- When you're counting operations of some algorithm, you need to show code (be it Matlab or Pseudo-) that matches your counts. Otherwise there's no telling where your results came from.

- When you find a result using facts from the book, then always mention what you are using.

  For example, it will in general not be enough to say "This matrix is s.p.d." even if that is so. You need to give a reason why you believe so.

- Be very aware of what a *necessary* and what a *sufficient* condition is. You might want to review what you learnt in calculus about conditions for local extrema of functions to help you along with that.

  In particular, the theorem on p.213 in the book gives *necessary* conditions that *need* to be true for every spd matrix. That means, if they are not satisfied, then it is safe to conclude that the matrix is not spd. But even if all the conditions are true, the matrix may still not be spd: the conditions are *not sufficient*. (that is, unless you can point to a proof of that fact.)

- When counting operations, try not to initialize variables with zero and then add in a loop. If you do, the first operation is always "0 + something", which could just as well be handled by an assignment without any arithmetic. If you do implement and count "0 + ⋯" as such, you will generally miss the operation count that the book gives.

- For the Cholesky back-/forward-substitution step, realize that a transpose costs n^2 operations, even if they're not arithmetic ones. The goal of the problem was to write an algorithm that avoids this transpose, which *nobody* did.

1. Problem 2, p. 180

a)

$$\left\|(3,-5,\sqrt{2})^T\right\|_{l^2} = \sqrt{3^2+5^2+\sqrt{2}^2} = \sqrt{9+25+2} = \sqrt{36} = 6,$$

$$\left\|(3,-5,\sqrt{2})^T\right\|_{l^\infty} = \max\{|3|,|-5|,|\sqrt{2}|\} = 5.$$

b)

$$\left\|(2,1,-3,4)^T\right\|_{l^2} = \sqrt{4+1+9+16} = \sqrt{30} \approx 5.477.$$

$$\left\|(2,1,-3,4)^T\right\|_{l^\infty} = 4.$$

c)

$$\left\|(4,8,-1)^T\right\|_{l^2} = \sqrt{16+64+1} = \sqrt{81} = 9.$$

$$\left\|(4,8,-1)^T\right\|_{l^\infty} = 8.$$

d)

$$\left\|(-2\sqrt{3},-6,4,2)^T\right\|_{l^2} = \sqrt{12+36+16+4} = \sqrt{68} \approx 8.246.$$

$$\left\|(-2\sqrt{3},-6,4,2)^T\right\|_{l^\infty} = 6.$$

e)

$$\left\|(e,\pi,-1)^T\right\|_{l^2} = \sqrt{e^2+\pi^2+1} \approx 4.273.$$

$$\left\|(e,\pi,-1)^T\right\|_{l^2} = \pi.$$

2. Problem 6, p.180

a) Let

$$A = \begin{pmatrix} 5 & -4 \\ -1 & 7 \end{pmatrix}.$$

Let's do the straightforward thing first: $\|A\|_{l^\infty} = \max\{5+4,\, 1+7\} = 9$. To calculate $\|A\|_{l^2}$, we need a couple more steps. First, we need A^TA:

$$A^TA = \begin{pmatrix} 26 & -27 \\ -27 & 65 \end{pmatrix}.$$

Next, we need the eigenvalues of A^TA. To that end, we write down the charateristic polynomial of A:

$$\det(A^TA - \lambda I) = \begin{vmatrix} 26-\lambda & -27 \\ -27 & 65-\lambda \end{vmatrix} = (26-\lambda)(65-\lambda) - 27^2 = 0.$$

The solutions of that equation are

$$\lambda_{1,2}(A^TA) = \frac{91 \pm 3\sqrt{493}}{2}.$$

Thus,

$$\rho(A^TA) = \max\{|\lambda_1|, |\lambda_2|\} = \frac{91+3\sqrt{493}}{2} \approx 78.805,$$

and we obtain $\|A\|_{l^2} = \sqrt{\rho(A^TA)} \approx 8.877$.

b) Let

$$A := \begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix}, A^TA = \begin{pmatrix} 17 & 11 \\ 11 & 13 \end{pmatrix}, \lambda_{1,2}(A^TA) = 15 \pm 5\sqrt{5}.$$

Thus $\|A\|_{l^2} = \sqrt{15+5\sqrt{5}} \approx 5.1167$. Slightly more straightforward, $\|A\|_{l^\infty} = 4+2 = 6$.

c) Let

$$A := \begin{pmatrix} 4 & -1 & -2 \\ 1 & 2 & -3 \\ 0 & 0 & 4 \end{pmatrix}, A^TA = \begin{pmatrix} 17 & -2 & -11 \\ -2 & 5 & -4 \\ -11 & -4 & 29 \end{pmatrix}.$$

Here, obtaining the roots of

$$\det(A^TA - \lambda I) = -\lambda^3 + 51l^2 - 582\lambda + 1296 = 0$$

analytically really does not help much–I also don't really know how to do it. Sure, I can let a computer algebra system loose on it, but that only results in huge terms that are not necessarily helpful. So, we'll compute these eigenvalues and obtain:

```
A =
```

```
  4  -1  -2
  1  2  -3
  0  0  4

AtA =

  17  -2  -11
  -2   5  -4
  -11  -4  29

eigv =

    2.9411
   12.3350
   35.7239

normA =  5.9769
For comparison: norm(A,2)=5.97695
```

using

```
A = [4,-1,-2;1,2,-3;0,0,4]
AtA = A'*A
eigv = eig(AtA)
normA = max(sqrt(eigv))
fprintf('For comparison: norm(A,2)=%g', norm(A,2))
```

Further, it is not hard to see that $\|A\|_{l^\infty} = 7$.

d) Again, we will simply compute the eigenvalues and the norm:

```
A =

  2  1  0
  -1  2  -1
  -3  4  -4

AtA =

  14  -12  13
  -12  21  -18
  13  -18  17

eigv =

    0.34596
    5.02064
   46.63339

normA =  6.8289
For comparison: norm(A,2)=6.82886
```

Straightforwardly, $\|A\|_{l^\infty} = 11$.

3. Problem 1, p. 201

A few helpful preliminaries before we start: There are n^2 entries in a square matrix–easy, right? But how many entries are in a triangular matrix including the diagonal? The answer is $n^2/2 + n/2$, and this is simply one way of writing Gauss's sum formula

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2},$$

which is easily shown by induction. (If you're not absolutely sure how this would work, please go ahead and do it. It's really quick.) If you don't count the diagonal in a triangular matrix, you get n less entries, i.e. $n^2/2 - n/2$.

a) The quickest way to the goal is realizing that the only difference between Gaussian Elimination for LU and Gaussian Elimination for solving linear systems is that Gauss LU does not have to update the right-hand side. That saves one subtraction and one multiplication per inner loop, of which there is one for each entry below the diagonal, i.e. $n^2/2 - n/2$, as we learned above. All in all, we save $n^2 - n$ operations. p. 155 in the book shows that Gauss for solving takes

$$\frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n$$

operations, so Gauss for LU takes

$$\frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n - (n^2 - n) = \frac{2}{3}n^3 - \frac{1}{2}n^2 - \frac{1}{6}n,$$

which is the value we're looking to prove.

There is of course also the long way around involving operation-counting and induction proofs, if you like long-winded error-prone methods.

b) It is shown in the book that backsubstitution with a general triangular matrix (such as the result of Gaussian elimination) takes n^2 operations (cf. p. 156). Doing this twice would result in $2n^2$ operations, but that does not take into account that either L or U can be chosen so that they have ones on the diagonal. Inspection of the pseudocode on p. 155f yields that that saves us n divisions (the first and last line of the code, respectively). Altogether we get an operation count of $2n^2 - n$, as claimed.

c) What we are asked to do is essentially provide an operation count for matrix multiplication. Here's some quick and dirty code:

```
A = [1,2,3;4,5,6;7,8,9];
b = [1;2;3];
c = zeros(3,1);
n = 3;

for i = 1:n
  c(i) = A(i,1)*b(1);
  for j = 2:n
    c(i) = c(i) + A(i,j)*b(j);
  end
end

Ab = A*b
c
```

Each outer loop has $n - 1$ additions and n multiplications, so $2n - 1$ operations total. The outer loop is run n times, so we have

$$n(2n-1) = 2n^2 - n$$

operations, as claimed.

4. Problem 1, p. 221

The following table summarizes the results:

Matrix	Strictly diag. dom.	Sym. pos. def.
a)	✓	✓ (diag.dom.)
b)	×	× (not sym.)
c)	×	× ($a_{2,1}^2 > a_{1,1}a_{2,2}$)
d)	×	✓ (see below)
e)	✓	× (not sym.)
f)	✓	× ($a_{3,3} < 0$)

The Theorem on p. 213 has many useful criteria that allow us to quickly conclude that a matrix is *not* positive definite–the remarks in parentheses refer to the condition that is violated. Similarly, the corollary on p. 214 is a quick positive result, which we can use in part a). This leaves us with only one matrix to check explicitly: the one in d). For this matrix, we'll solve

$$\begin{pmatrix} l_{1,1} & & \\ l_{2,1} & l_{2,2} & \\ l_{3,1} & l_{3,2} & l_{3,3} \end{pmatrix}\begin{pmatrix} l_{1,1} & l_{2,1} & l_{3,1} \\ & l_{2,2} & l_{3,2} \\ & & l_{3,3} \end{pmatrix}=\begin{pmatrix} 4 & -2 & 2 \\ -2 & 6 & 4 \\ 2 & 4 & 7 \end{pmatrix}$$

by direct factorization. We obtain

$$\begin{pmatrix} l_{1,1} & & \\ l_{2,1} & l_{2,2} & \\ l_{3,1} & l_{3,2} & l_{3,3} \end{pmatrix}=\begin{pmatrix} 2 & & \\ -1 & \sqrt{5} & \\ 1 & \sqrt{5} & 1 \end{pmatrix}.$$

Since the Cholesky decomposition works if and only if a matrix is s.p.d., we may conclude that this is the case for the matrix in d).

5. Problem 11, p.221

Consider the following (already annotated) code for the Cholesky decomposition:

```
function L = mycholesky(A)
  [n, dummy] = size(A);

  % coded up directly from p. 216

  for k=1:n
    diag_squared = A(k,k);

    % 2(k-1) operations
    for j=1:k-1
      diag_squared = diag_squared - L(k,j)^2;
    end

    L(k,k) = sqrt(diag_squared);

    for i=k+1:n
      entry = A(i,k);

      % 2(k-1) operations
      for j=1:k-1
```

```
        entry = entry - L(i,j)*L(k,j);
      end

      % 1 operation
      L(i,k) = entry/L(k,k);
    end
  end
end
```

According to it, we have

$$\sum_{k=1}^{n}\left[2(k-1)+\sum_{i=k+1}^{n}[2(k-1)+1]\right]$$

operations. (As a help when doing problems like these: You might want to figure out how summing works in your favorite computer algebra system. That way, after you obtain the non-closed form term above, you can have the computer verify that it indeed does sum to the right expression before you waste time proving something that's wrong. Note: Turning in computer output does not count as a proof. This is just meant to help you not waste time. Note 2: For example, in Maxima you type

```
nusum(2*(k-1)+(n-(k+1)+1)*(2*(k-1)+1), k, 1, n);
```

to sum the above expression and it tells you that we're on the right track.)

We can prove this now by massaging the formula into the right shape:

$$\begin{aligned}
&\sum_{k=1}^{n}\left[2(k-1)+\sum_{i=k+1}^{n}[2(k-1)+1]\right]\\
=\;&\sum_{k=1}^{n}[2(k-1)+(n-(k+1)+1)[2(k-1)+1]]\\
=\;&\sum_{k=1}^{n}[2(k-1)+(n-k))[2(k-1)+1]]\\
=\;&\sum_{k=1}^{n}\left[n(2k-1)-2k^2+3k-2\right]\\
=\;&2n\sum_{k=1}^{n}k-n^2-2\sum_{k=1}^{n}k^2+3\sum_{k=1}^{n}k-2n\\
=\;&2n\frac{n(n+1)}{2}-n^2-2\frac{n(n+1)(2n+1)}{6}+3\frac{n(n+1)}{2}-2n\\
=\;&\frac{2}{3}n^3+\frac{1}{2}n^2-5n,
\end{aligned}$$

where we've used Gauss's sum formula

$$\sum_{k=1}^{n}k=\frac{n(n+1)}{2}$$

and the fact that

$$\sum_{k=1}^{n}k^2=\frac{n(n+1)(2n+1)}{6},$$

which you can either look up or easily prove by induction. Further, there is one square root per diagonal entry of L, so that we end up with n square roots.

6. Problem 12, p. 221

a) Consider that we're trying to solve $L\,L^T\boldsymbol{x}=\boldsymbol{b}$, or rather, $L(L^T\boldsymbol{x})=\boldsymbol{b}$ and note that just like for LU, the solving can be broken up in two separate subtasks, namely

$$\begin{aligned}\text{Solve } L\boldsymbol{y} &= \boldsymbol{b},\\ \text{Solve } L^T\boldsymbol{x} &= \boldsymbol{y}.\end{aligned}$$

We assume that $L\in\mathbb{R}^{n\times n}$ is a given lower-triangular matrix that satisfies $L\,L^T=A$ for some matrix A for which we're solving $A\boldsymbol{x}=\boldsymbol{b}$. The first solve can be performed as

$$y_i=\left(b_i-\sum_{j=1}^{i-1}L_{i,j}y_j\right)/L_{i,i}\quad(i=1,...,n),$$

and the second solve works on $\boldsymbol{y}$ and yields

$$\begin{aligned}x_i &= \left(y_i-\sum_{j=i+1}^{n}L^T_{i,j}y_j\right)/L_{i,i}\quad(i=n,...,1)\\ &= \left(y_i-\sum_{j=i+1}^{n}L_{j,i}y_j\right)/L_{i,i}\quad(i=n,...,1).\end{aligned}$$

Note that in the second solve, the vector needs to be filled in backwards (i.e. from n to 1, in the same order as for a regular Gauss backsubstitution).

b) Following Problem 3b), it is not hard to see that the operation count for two backsubstitutions including the diagonal is $2n^2$. (also cf. p. 156)

7. Problem 13, p. 221

Code:

```
function p7
  disp('--------------------------------')
  disp('Part a)')
  disp('--------------------------------')
  A = [16, -28, 0;-28, 53,10;0,10,29];
  b = [8, -2, 38]';
  solution = cholesky_solve(A, b)
  disp('--------------------------------')
  disp('Part b)')
  disp('--------------------------------')
  A = [9/4,3,3/2;3,25/4,7/2;3/2,7/2,17/4];
  b = [3,1,9]';
  solution = cholesky_solve(A, b)
  disp('--------------------------------')
  disp('Part c)')
  disp('--------------------------------')
  A = [4,-2,-2,0;-2,5,1,-2;-2,1,10,3;0,-2,3,18];
  b = [4,-4,4,-13]';
  solution = cholesky_solve(A, b)
  disp('--------------------------------')
  disp('Part d)')
  disp('--------------------------------')
  A = [1,-2,3,-2;-2,20,-2,8;3,-2,11,-5;-2,8,-5,9];
```

```
  b = [15,-12,56,-35]';
  solution = cholesky_solve(A, b)
end

function sol = cholesky_solve(A, b)
  A
  L = mycholesky(A)

  [n, dummy] = size(A);

  % backsubsitution for L (lower)
  tempsol = zeros(n,1);
  for row = 1:n
    tempsol(row) = b(row) - L(row,1:row-1)*tempsol(1:row-1);
    tempsol(row) = tempsol(row) / L(row,row);
  end

  % backsubsitution for L' (upper)
  sol = zeros(n,1);
  for step = 0:n-1
    row = n-step;
    sol(row) = tempsol(row) - L(row+1:n,row)'*sol(row+1:n);
    sol(row) = sol(row) / L(row,row);
  end

  residual = A*sol - b
end

function L = mycholesky(A)
  [n, dummy] = size(A);

  % coded up directly from p. 216

  for k=1:n
    diag_squared = A(k,k);

    for j=1:k-1
      diag_squared = diag_squared - L(k,j)^2;
    end

    L(k,k) = sqrt(diag_squared);

    for i=k+1:n
      entry = A(i,k);

      for j=1:k-1
        entry = entry - L(i,j)*L(k,j);
      end

      L(i,k) = entry/L(k,k);
    end
  end

  if (norm(A-L*L',2) > 1e-13)
```

```
    error('Cholesky did not work out.')
  end
end
```

Data:

```
------------------------------
Part a)
------------------------------
A =

  16  -28   0
  -28  53  10
   0  10  29

L =

  4  0  0
  -7  2  0
  0  5  2

residual =

  0
  0
  0

solution =

  -3
  -2
  2

------------------------------
Part b)
------------------------------
A =

   2.2500   3.0000   1.5000
   3.0000   6.2500   3.5000
   1.5000   3.5000   4.2500

L =

   1.50000   0.00000   0.00000
   2.00000   1.50000   0.00000
   1.00000   1.00000   1.50000

residual =

  0
  0
  0
```

```
solution =

  4
  -4
  4

------------------------------
Part c)
------------------------------
A =

   4  -2  -2   0
  -2   5   1  -2
  -2   1  10   3
   0  -2   3  18

L =

  2  0  0  0
  -1  2  0  0
  -1  0  3  0
  0  -1  1  4

residual =

  0
  0
  0
  0

solution =

  1
  -1
  1
  -1

------------------------------
Part d)
------------------------------
A =

   1  -2   3  -2
  -2  20  -2   8
   3  -2  11  -5
  -2   8  -5   9

L =

  1  0  0  0
  -2  4  0  0
  3  1  1  0
  -2  1  0  2
```

```
residual =

  0
  0
  0
  0

solution =

  -9.062500
   0.093750
   6.500000
  -2.375000
```