Textbook problems to be done by hand:

- 1.4.21

We get 
\[
R = \begin{bmatrix}
4 & 1 & 2 & 1 \\
0 & 3 & 2 & 1 \\
0 & 0 & 2 & 3 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]
This means 
\[
R^T R x = b
\]
Then we get 
\[
R x = (R^T)^{-1} b = \begin{bmatrix}
8 \\
6 \\
5 \\
1 \\
\end{bmatrix}
\]
and 
\[
x = R^{-1} ((R^T)^{-1} b) = \begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
\end{bmatrix}
\]

- 1.4.54

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22} \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\end{bmatrix} > 0
\]
for any nonzero vector \( x \) because \( A \) is positive definite. So let’s choose a particular vector where \( x_2 = 0 \). Then we get 
\[
0 < \begin{bmatrix}
x_1^T \\
0 \\
\end{bmatrix}
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22} \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
0 \\
\end{bmatrix} = x_1^T A_{11} x_1
\]
for an arbitrary \( x_1 \). But that means \( A_{11} \) is p.d. Same can be done for \( A_{22} \).

- 1.4.56

Check that \( B \) is p.d. by considering a nonzero vector \( y \) and the multiplication 
\[
y^T B y = y^T X^T A X y = (X y)^T A (X y).
\]
Since \( X \) is nonsingular, then for any nonzero \( y, z = X y \) is nonzero as well. So 
\[
y^T B y = z^T A z
\]
for some nonzero \( z \). But \( A \) is p.d., so this means \( z^T A z > 0 \). Thus \( y^T B y > 0 \), so \( B \) is p.d.

- 1.4.69

The key here is to show that \( C \) is symmetric, and that \( V \) determines whether it’s p.d. or positive semidefinite. \( C \) is symmetric because 
\[
C^T = (V^T V)^T = V^T (V^T)^T = V^T V = C.
\]
Now assume \( V \) is linearly independent columns. That means that if \( x \) is nonzero, \( y = V x \) has to be nonzero as well. So 
\[
x^T C x = x^T V^T V x = (V x)^T (V x) = y^T y
\]
for nonzero \( y \). But 
\[
y^T y = \sum_i y_i^2 > 0.
\]
So \( C \) is p.d.

If \( V \) is linearly dependent, then it’s possible that \( y = V x = 0 \) for a nonzero \( x \). But still, 
\[
x^T C x = y^T y.
\]
So now if \( y = 0 \), we get that \( x^T C x = 0 \). If \( y \neq 0 \), then
\[ x^T C x = y^T y > 0 \] by argument similar to linearly independent case. So whatever the case, \( x^T C x \geq 0 \).

- 1.5.9

We solve \( Rx = y \) with back sub, assuming \( R \) has band \( s \). Then in the equation

\[
x_i = \frac{1}{R_{ii}} \left( y_i - \sum_{j=i+1}^{n} R_{ij} x_j \right)
\] (1)

we can reduce the sum significantly and get

\[
x_i = \frac{1}{R_{ii}} \left( y_i - \sum_{j=i+1}^{i+s} R_{ij} x_j \right).
\] (2)

Now there are \( s \) elements in the sum only, and each has one multiplication. This means there are \( s \) multiplications inside the parentheses (along with \( s \) sums), and one division at the end. This gives \( 2s + 1 \) flops per solve of \( x_i \). Since there are \( n \) equations, we get \( n(2s + 1) \) flops, so \( O(ns) \) flops.

- 1.5.12 (print out the 6x6 matrices of \( R \) and \( A^{-1} \))

In this problem, after setting up \( A \), we get

\[
R = \begin{bmatrix}
1.4142 & -0.7071 & 0 & 0 & 0 & 0 \\
0 & 1.2247 & -0.8165 & 0 & 0 & 0 \\
0 & 0 & 1.1547 & -0.8660 & 0 & 0 \\
0 & 0 & 0 & 1.1180 & -0.8944 & 0 \\
0 & 0 & 0 & 0 & 1.0954 & -0.9129 \\
0 & 0 & 0 & 0 & 0 & 1.0801
\end{bmatrix}
\]

If we’d done \( A^{-1} \), we’d get

\[
A^{-1} = \begin{bmatrix}
0.8571 & 0.7143 & 0.5714 & 0.4286 & 0.2857 & 0.1429 \\
0.7143 & 1.4286 & 1.1429 & 0.8571 & 0.5714 & 0.2857 \\
0.5714 & 1.1429 & 1.7143 & 1.2857 & 0.8571 & 0.4286 \\
0.4286 & 0.8571 & 1.2857 & 1.7143 & 1.1429 & 0.5714 \\
0.2857 & 0.5714 & 0.8571 & 1.1429 & 1.4286 & 0.7143 \\
0.1429 & 0.2857 & 0.4286 & 0.5714 & 0.7143 & 0.8571
\end{bmatrix}
\]

This is really bad, simply considering multiplication of \( A^{-1}b \), since there are no 0 entries. This means we’re using the maximum number of flops \( 2n^2 \), independent of the fact that it was likely expensive to compute \( A^{-1} \).
• **Written Problem:** For sparse matrices (matrices with mostly zero entries, such as banded matrices), explain why trying to calculate \( A^{-1} \) in order to solve \( Ax = b \) is even more inefficient (and thus foolish) than usual. (As compared to, say, using the Cholesky decomposition.) (Hint: Look at the paragraph above Exercise 1.5.12.)

This is really bad, simply considering multiplication of \( A^{-1}b \), since there are no 0 entries. This means we’re using the maximum number of flops \( 2n^2 \), independent of the fact that it was likely expensive to compute \( A^{-1} \)

**Additional MATLAB:**

• Let \( A \) be a symmetric, positive definite, tridiagonal matrix. We store \( A \) as two vectors:
  
  – an \( n \times 1 \) vector \( v \) representing the main diagonal
  – an \( (n-1) \times 1 \) vector \( w \) representing the upper diagonal

Using basic programming (for loops, while loops, and if statements):

1. Write a function that inputs vectors \( v \) and \( w \), then uses Cholesky’s method to find the Cholesky factor, and then outputs the number of flops used. DO NOT form the full matrix \( A \) or \( R \), as this could take up too much memory. Simply form a vector \( r \) that contains the main diagonal of the Cholesky matrix \( R \), and a vector \( s \) that contains the upper diagonal. Print out or write out this function and turn it in.

2. Run your program for \( n = 10; 100; 400 \), and \( v \) the vector of all 2’s and \( w \) the vector of all −1’s. Print out or write out your results of the number of flops (not the actual resulting decomposition) and turn them in.

Code is in solutions folder. The key here is that all the internal for loops go away. There are two tricky parts. The reason there are if statements is because the for loops sometimes don’t execute when \( i = 1 \) or when \( i = n \). Another way to think about this is that the \( s \) vector (the upper diagonal) only has \( n - 1 \) elements in it, and we don’t want to index in with \( n \).

Another note is that calculating the off diagonal terms becomes super easy. This is because \( R_{ki}R_{kj} \) can never be anything other than 0 for a band 1 triangular matrix. If \( i = j + 1 \) and \( k = i - 1 \), then we have \( R_{i-1,i}R_{i-1,i+1} \). The second term is 0, so there’s nothing we need to subtract.

For \( n = 10 \), flops should be around \( n = 40 \). For \( n = 1000 \), flops should be around 4000. You don’t have to have the exact number, but the point is that it scales linearly with \( n \).

**Ungraded:**

• 1.4.22

A: no because not symmetric. B: yes. C: No because there’s a negative on the diagonal. D: No because on \( r_{22} \) you get a squareroot of 0.
• 1.5.4

Since the points lie on a $m \times m$ grid, there are a total of $m^2$ points. So the matrix we will form is size $m^2 \times m^2$. But each point only depends on itself and its four neighbors, so each row will have at most 5 entries. So the total number of nonzeros is $5m^2$. Now we consider the band. In the $10 \times 10$ case, the point above $x_{34}$ is $x_{24}$, which is because we ordered the indices from left to right, top to bottom, and in that way there were 10 points between because there are 10 points per row. So for $m$ points per row, there are $m$ points between a point $x_i$ and the point $x_{i-m}$ above it. So the band of the $m^2 \times m^2$ matrix is $m$.

• 1.4.62 (remember that $\det(A \cdot B) = \det(A) \cdot \det(B)$)

If $A$ is positive definite and symmetric, then $A = R^T R$. But $\det(A) = \det(R^T R) = \det(R^T) \cdot \det(R)$. But since $R$ is triangular, $\det(R)$ is just the product of the diagonal. And since all diagonal elements of $R$ are positive, $\det(R) > 0$. This means $\det(A) > 0$ since it’s the product of two positive terms.

• 1.5.11 (can be done with rough calculation of flops, don’t implement in code)

For this grid problem, the number of points $n = m^2$. For banded matrices with band $s$, cholesky decomposition will only have $r_{ij}$ terms for $j < i + s$. This means there are $s$ per row, or $ns$ total terms. To compute each term, we have to use the equation for the entry, which now only has $s$ terms in the sum. This means it takes $O(s)$ flops per entry, so for $ns$ entries we have $O(ns^2)$ flops. In terms of $m$, since $s = m$ from 1.5.4, this gives $O(m^2 \cdot (m)^2) = O(m^4)$ flops. If we’d done Cholesky naively, it would have taken $O(n^3) = O(m^6)$ flops. So if $m = 100$, banded cholesky has $m^6/m^4 = 100^2 = 10000$ times less flops.

Also for storage, instead of storing $n^2/2$ terms in $R$, we only need now to store $ns$ terms in $R$. In terms of $m$, banded cholesky has $O(m^3)$ storage, whereas full Cholesky has $O(m^4)$ storage.

• 1.4.23

Same as in 1.4.22