**1.1.1**. The inverses of and (with in the top corner) are and . Guess and multiply by . Find a simple formula for the entries of on and below the diagonal (), and then on and above the diagonal ()

**Sol**. Guess that

and test it by

**1.1.2**. Compute in three steps, using and in equation triu(ones(3)).

1. Check that , where has ’s on the main diagonal and ’s along the diagonal above. Its transpose is lower triangular.

2. Check that when has ’s on and above the main diagonal.

3. Invert to find . Inverses come in reverse order.

**Sol.** 1.

2.

**3**.

**1.1.3**. The difference matrix in MATLAB is eye(5) – diag(ones(4,1),1). Construct the sum matrix from triu(ones(5)). (This keeps the upper triangular part of the 5 by 5 all-ones matrix.) Multiply U \* S to verify that

**Sol**. MATLAB codes :

>> U=eye(5) - diag(ones(4,1),1)

U =

1 -1 0 0 0

0 1 -1 0 0

0 0 1 -1 0

0 0 0 1 -1

0 0 0 0 1

>> S=triu(ones(5))

S =

1 1 1 1 1

0 1 1 1 1

0 0 1 1 1

0 0 0 1 1

0 0 0 0 1

>> U\*S

ans =

1 0 0 0 0

0 1 0 0 0

0 0 1 0 0

0 0 0 1 0

0 0 0 0 1

**1.1.4**. For every , is upper triangular with ones on and above the diagonal. For check that produces the matrix predicted in Problem 1. Why is certain to be a symmetric matrix ?

**Sol**.

is certain to be a symmetric because

**1.1.5**. The inverses of and (please also invert ) have fractions : and . First guess the determinant of . Then compute and and – any software is allowed.

**Sol**.

**1.1.6**. (Challenge problem) Find a formula for the , entry of below the diagonal (). Those entries grow linearly along every row and up every column. (Section 1.4 will come back to these important inverses.) Problem 7 below is develped in the Worked Example of Section 1.4

**Sol**.

**1.1.7**. A column times a row is a rank-one matrix . All columns are multiples of , and all rows are multiples of has rank 1 : . Write in this special form . Predict a similar formula for .

**Sol**.

Prediction.

**1.1.8**. (a) Based on Problem 7, predict the , entry of below the diagonal. (b) Subtract this from your answer to Problem 1 (the formula for when ). This gives the not-so-simple formula for .

**Sol.** (a)

(b)

**1.1.9**. Following Example 1.1 A with instead of , show that is perpendicular to each column of . Solve with the singular matrix by u=pinv(C)\*f . Try u=C\e and C\f, before and after adding a fifth equation 0=0.

**Sol**.

>> C=toeplitz([2 -1 0 0]);

>> C(1,4)=-1;

>> C(4,1)=-1;

>> f=[1 -1 1 -1]';

>> u=pinv(C)\*f

u =

0.2500

-0.2500

0.2500

-0.2500

>> u=C\f

u =

-1.0000

-1.5000

-1.0000

-1.5000

>> u=[zeros(1,4);C]\[0;f]

Warning: Rank deficient, rank=3, tol= 1.216188e-15.

u =

0.5000

0

0.5000

-0.0000

>> u=C\e

u =

1.0e+16 \*

1.8014

1.8014

1.8014

1.8014

>> C\*u

ans =

-8

8

-4

0

>> u=[zeros(1,4);C] \ [0;e]

Warning: Rank deficient, rank=3, tol= 1.216188e-15.

u =

1.0e-15 \*

-0.1730

0

-0.1089

-0.3140

>> C\*u

ans =

1.0e-15 \*

-0.0320

0.2820

0.0961

-0.3461

**1.1.10**. The ‘hanging matrix’ in Worked Example 1.1 B changes the last entry of to . Find the inverse matrix from . Find the inverse also from (Check upper times lower triangular !) and .

**Sol.**

**1.1.11**. Suppose is any upper triangular matrix and is the reverse identity matrix in 1.1 B. Then is a ‘southeast matrix’. What geographies are and ? By experiment, a southeast matrix times a northwest matrix is \_\_\_\_\_ .

**Sol.**  which is a ‘southeast matrix’.

which is a ‘northwest matrix’.

which is a ‘southwest matrix’.

**1.1.12**. Carry out elimination on the 4 by 4 circulant matrix to reach an upper triangular (or try [L,U]=lu(C) in MATLAB). Two points to notice: The last entry of is \_\_\_ because is singular. The last column of has new nonzeros. Explain why this "fill-in" happens.

**Sol**. MATLAB codes :

>> C=toeplitz([2 -1 0 0]); C(1,4)=-1; C(4,1)=-1;[L,U]=lu(C)

L =

1.0000 0 0 0

-0.5000 1.0000 0 0

0 -0.6667 1.0000 0

-0.5000 -0.3333 -1.0000 1.0000

U =

2.0000 -1.0000 0 -1.0000

0 1.5000 -1.0000 -0.5000

0 0 1.3333 -1.3333

0 0 0 0.0000

The last entry of is because is singular, thus the last column of has new nonzeros.

**1.1.13**. By hand, can you factor the circulant (with three nonzero diagonals, allowing wraparound) into circulants times (with two nonzero diagonals, allowing wraparound so not truly triangular)?

**Sol**. Row transform :

where

**1.1.14**. Gradually reduce the diagonal 2, 2, 2 in the matrix until you reach a singular matrix . This happens when the diagonal entries reach \_\_\_ . Check the determinant as you go, and find a nonzero vector that solves .

**Sol**. when the diagonal entries reach’s, reachs .

Check.

For ,

**1.1.15**. How many individual multiplications to create and and ?

**Sol**. There are times of multiplications in , times of multiplications in , and times of multiplications in .

**1.1.16**. You can multiply by rows (the usual way) or by columns (more important) . Do this multiplication both ways:

By rows : By columns :

**Sol.** By rows : By columns :

**1.1.17**. The product is a linear combination of the columns of . The equations have a solution vector exactly when is a \_\_\_ of the columns. Give an example in which is not in the column space of . There is no solution to , because is not a combination of the columns of .

**Sol**. The equations have a solution vector exactly when is a combination of the columns.

has no solution because is not a combination of the columns of

**1.1.18**. Compute by multiplying the matrix times each column of : . Thus, A \* B(:,j)=C(:,j)

**Sol**.

**1.1.19**. You can also compute by multiplying each row of times : .

A solution to is also a solution to . Why?

From , how do we know

**Sol**.

Thus,

**1.1.20**. The four ways to find give numbers, columns, rows, and matrices:

1 (rows of A) times (columns of B) C(i,j)=A(i,:) \* B(:,j)

2 A times (columns of B) C( : ,j)=A \* B(:,j)

3 (rows of A) times B C(i,:)=A(i,:) \* B

4 (columns of A) times (rows of B) for k=1:n, C=C+A(:,k) \* B(k,:); end

Finish these 8 multiplications for columns times rows. How many for n by n?

**Sol**.

**1.1.21**. Which one of these equations is true for all by matrices and ?

, , ,

**Sol**. , false. , false. , false. , false.

**1.1.22**. Use n=1000; e=ones(n, 1); K=spdiags([-e, 2\*e, -e] , -1:1, n, n) ; to enter as a sparse matrix. Solve the sparse equation by u=K\e. Plot the solution by plot(u) .

**Sol**. MATLAB codes : >> n=1000; e=ones(n, 1); K=spdiags([-e, 2\*e, -e] , -1:1, n, n); u=K\e; plot(u)



**1.1.23**. Create 4-component vectors , , and enter A=spdiags([u, v, w] , - 1 : 1 , 4, 4) . Which components of and are left out from the -1 and 1 diagonals of ?

**Sol**. MATLAB codes :

>> u=[11;12;13;14];v=[21;22;23;24];w=[31;32;33;34]; full(spdiags([u,v,w],-1:1,4,4))

ans =

21 32 0 0

11 22 33 0

0 12 23 34

0 0 13 24

**The forth entries of both**  and are left out from the -1 and 1 diagonals of ?

**1.1.24**. Build the sparse identity matrix I=sparse( i, j, s, 100, 100) by creating vectors i, j, s of positions i, j with nonzero entries s. (You could use a for loop.) In this case speye(100) is quicker. Notice that sparse(eye(10000)) would be a disaster, since there isn't room to store eye(lOOOO) before making it sparse.

**Sol**. **MATLAB codes :**

>> I=sparse(100,100);

>> for i=1:100, I=I + sparse(i,i,1,100,100); end

**1.1.25**. The only solution to or is , so and are invertible. For proof, suppose is the largest component of . If , this forces . Then the next equations force every . At the end, when the boundary is reached, only gives zero if . Why does this "diagonally dominant" argument fail for and ?

**Sol**. Because the boundary condition in and the boundary condition in give no more information to .

**1.1.26**. For which vectors is toeplitz() a circulant matrix (cyclic diagonals) ?

**Sol**. toeplitz([2 -1 zeros(1,n) -1]) where n can be assigned to any natural number.

**1.1.27\***. (Important) Show that the 3 by 3 matrix comes from where is a "difference matrix".

Which column of would you remove to produce with ?

Which row would you remove next to produce with ?

The difference matrices , , have 0, 1, 2 boundary conditions. So do the "second differences" , , and .

**Sol**.

where is the first three columns of

where is the first two rows of