Amelia Rotondo
CWID: 887925113

31

Homework: Chapter 2

5

1. Alice buys three apples, a dozen bananas, and one cantaloupe for $2.36. Bob buys a dozen apples and two cantaloupes for $5.26. Carol buys two bananas and three cantaloupes for $2.77. How much do single pieces of each fruit cost? (You might want to set: `format bank`.)

```
%{ Method for Problem 1:
% using Ax = b and solving}
A = [3, 12, 1; 12, 0, 2; 0, 2, 3];
b = [2.36; 5.26; 2.77];
x = A \ b;
% Output and Check Result
disp(['Cost of a single apple: $', num2str(x(1))]);
disp(['Cost of a single banana: $', num2str(x(2))]);
disp(['Cost of a single cantaloupe: $', num2str(x(3))]);
```

```
      Cost of a single apple: $0.29
      Cost of a single banana: $0.05
      Cost of a single cantaloupe: $0.89
```

3

5. These conditions are difficult or expensive to use as the basis for checking if a particular matrix is positive definite. In Matlab , the best way to check positive definiteness is with the `chol` function. See `help chol`

a) Which of the following families of matrices are positive definite?

i) `M = magic(n);`
```
   1) Matrix is not positive definite.
```
ii) `H = hilb(n);`
```
   1) Matrix is positive definite.
```
iii) `P = pascal(n);`
```
   1) Matrix is positive definite.
```
iv) `I = eye(n,n);`
```
   1) Matrix is positive definite.
```
v) `R = randn(n,n);`
```
   1) Matrix is not positive definite.
```
vi) `R = randn(n,n); A = S' * S;`
```
   1) Matrix is positive definite.
```
vii) `R = randn(n,n); A = T' + T;`
```
   1) Matrix is not positive definite.
```
viii) `R = randn(n,n); I = eye(n,n); D = R' + R + n*I;`
```
   1) Matrix is not positive definite.
```

b) Using these equations in different orders yields different variants of the Cholesky algorithm for computing the elements of R. What is one such algorithm?

i) The book shows one algorithm for A = RR^T in the question at-hand. However, one variant of this algorithm is A = RDR^T, which uses LDL Decomposition to find a determinant and reconstruct the matrix.

3

6. Show how to generate this matrix in Matlab with eye, ones, and triu.

Show that K_1(A) = n2^{n-1}.

```
% Method for Problem 6:
x = 3;
I = eye(x,x);
O = ones(x);
S = 2*I-O;
T = triu(S);
A = T';
```

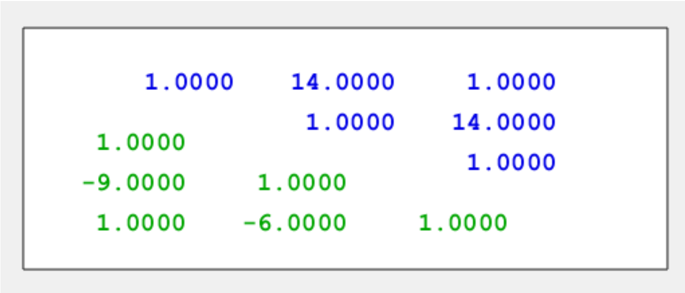
Here it is possible to calculate A^{-1} and $\kappa(A)$ explicitly

a) For what n does κ1(A) exceed 1/eps?
   - i) k1(A) exceeds 1/eps at n = 48

b) This matrix is not singular, so Ax cannot be zero unless x is zero. However, there are vectors x for which ‖Ax‖ is much smaller than ‖x‖. Find one such x.
   - i) setting `x = [0.001; 0.01; 0.1;];` yields the result [0.0010, 0.0090, 0.0890], therefore showing how Ax can be less than x.

c) Because this matrix is already upper triangular, Gaussian elimination with partial pivoting has no work to do. What are the pivots?
   - i) The pivots of a are: -1, 1, and 1.

d) Use lugui to design a pivot strategy that will produce smaller pivots than partial pivoting. (Even these pivots do not completely reveal the large condition number.)
   - i) Diagonal Pivoting produces smaller pivots than Partial Pivoting.



```
    1.0000    14.0000     1.0000
               1.0000    14.0000
 1.0000
                          1.0000
-9.0000     1.0000
 1.0000    -6.0000     1.0000
```

7. Modify lux(A) and write mydet(A):

2

```
function [L,U,p,sig, detA] = lutx(A)
%LU Triangular factorization
% [L,U,p,sig] = lutx(A) computes a unit lower triangular
% matrix L, an upper triangular matrix U, a permutation
% vector p, and a scalar sig, so that L*U = A(p,:) and
% sig = +1 or -1 if p is an even or odd permutation.
[n,n] = size(A);
p = (1:n)';
sig = 1; % Initialize sig to +1
for k = 1:n-1
% Find index of largest element below diagonal in k-th column
[r,m] = max(abs(A(k:n,k)));
m = m+k-1;
% Skip elimination if column is zero
if (A(m,k) ~= 0)
% Swap pivot row
if (m ~= k)
A([k m],:) = A([m k],:);
p([k m]) = p([m k]);
sig = -sig; % Flip the sign if a row swap occurs
end
% Compute multipliers
i = k+1:n;
A(i,k) = A(i,k)/A(k,k);
% Update the remainder of the matrix
j = k+1:n;
A(i,j) = A(i,j) - A(i,k)*A(k,j);
end
end
%Separate result
L = tril(A,-1) + eye(n,n);
U = triu(A);
% MYDET Compute the determinant of a square matrix A using LU
decomposition.
% detA = mydet(A) computes the determinant of A.
% Compute the determinant using the diagonal elements of U and the sign
sig
detA = sig * prod(diag(U));
end
```

Start with this & modify

9. A = [1, 2, 3; 4, 5, 6; 7, 8, 9;] and b = [1; 3; 5;]:

5

a) Show that the set of linear equations Ax = b has infinitely many solutions. Describe the set of possible solutions.

i) det(A) = 0, and therefore it is Linearly Dependent. The set of solutions is:
x = [(⅓ + t); (⅓ +-2t); t;]

b) Suppose Gaussian elimination is used to solve Ax = b using exact arithmetic. Because there are infinitely many solutions, it is unreasonable to expect one particular solution to be computed. What happens?

   i) det(A) = 0, and therefore it is Linearly Dependent. The set of solutions is: x = [(⅓ + t); (⅓ +-2t); t;]. Therefore, it is unreasonable for this linear system to have one particular solution.

c) Use bslashtx to solve Ax = b on an actual computer with floating-point arithmetic. What solution is obtained? Why? In what sense is it a “good” solution? In what sense is it a “bad” solution?

   i) I obtained the solution x = [4.33; -7.67; 4;]. While this is technically represented by the parametric solution shown earlier, it does not fully express the linear dependence that demands this to be represented as a parametric.

d) Explain why the built-in backslash operator x = A\b gives a different solution from x = bslashtx(A,b).

   i) While bslashtx(A,b) uses triangular decomposition and back substitution, A \ b uses whatever method produces the most efficient method based on the inputs provided. Since this matrix is linearly dependent, different linear system solutions can be reached based on the algorithms used.

4

11. The inverse of a matrix A can be defined as the matrix X whose columns x_j solve the equations $Ax_j = e_j$, where e_j is the j-th column of the identity matrix

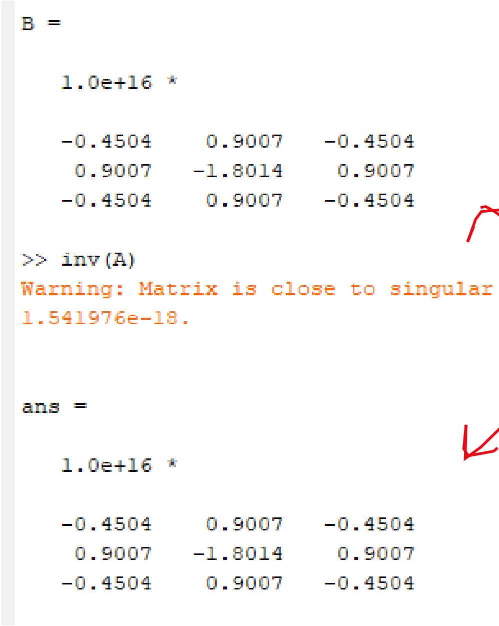
a) Starting with the function bslashtx, write a Matlab function X = myinv(A)

```
function A_inv = myinv(A)
% MYINV Compute the inverse of a matrix A using bslashtx.
% Perform LU decomposition with pivoting
[L, U, p] = lutx(A);
% Identity matrix of the same size as A
[n, ~] = size(A);
I = eye(n);
% Initialize the inverse matrix
A_inv = zeros(n);
% Solve for each column of the identity matrix
for k = 1:n
e_k = I(:, k); % Extract the k-th column of the identity matrix
x_k = bslashtx(L, e_k(p)); % Solve for x_k using bslashtx
y_k = backsubs(U, x_k); % Solve for y_k using back substitution
A_inv(:, k) = y_k; % Set the k-th column of the inverse matrix
end
```

b) Test your function by comparing the inverses it computes with the inverses obtained from the built-in inv(A) on a few test matrices.

   i) Yes, I did this.

Not clear that it works



```
B =

   1.0e+16 *

  -0.4504    0.9007   -0.4504
   0.9007   -1.8014    0.9007
  -0.4504    0.9007   -0.4504

>> inv(A)
Warning: Matrix is close to singular
1.541976e-18.


ans =

   1.0e+16 *

  -0.4504    0.9007   -0.4504
   0.9007   -1.8014    0.9007
  -0.4504    0.9007   -0.4504
```

same?

5 18. Find p, c_1 and c_2:

a) Write two parallel lines for kappa lo and hi:

   i)
```
kappalo = n.^(7/5);
kappahi = 500*n.^(7/5);
```

b) Based on this experiment, what is your guess for the exponent p? How confident are you?

   i) Based on my experiment, the constant for p is (7/5). While I believe this value can be improved, I was able to fit most of the results within these lines' boundaries.

c) The program uses ('erasemode','none'), so you cannot print the results. What would you have to change to make printing possible?

   i) To print results, you would have to change it to ('erasemode', 'normal').

19. For n = 100, solve this tridiagonal system of equations three different ways:

4

a) Use diag three times to form the coefficient matrix and then use lutx and bslashtx to solve the system.

```
% Define the main diagonal elements (2's)
main_diag = 2 * ones(n, 1);
% Define the subdiagonal elements (-1's)
sub_diag = -1 * ones(n - 1, 1);
% Create the coefficient matrix A using diag three times
A = diag(main_diag) + diag(sub_diag, -1) + diag(sub_diag, 1);
% Initialize the right-hand side vector b
b = [1; (2:n - 1)'; n];
% Perform LU decomposition on A
[L, U, P] = lutx(A);
% Solve the linear system A * x = b using bslashtx
x = bslashtx(A,b);
```

b) Use spdiags once to form a sparse representation of the coefficient matrix and then use the backslash operator to solve the system.

```
% Define the subdiagonal, main diagonal, and superdiagonal vectors
sub_diag = -1 * ones(n, 1); % Subdiagonal elements
main_diag = 2 * ones(n, 1); % Main diagonal elements
super_diag = -1 * ones(n, 1); % Superdiagonal elements
% Create the coefficient matrix A using spdiags
% Diagonals: [sub_diag, main_diag, super_diag]
% Offsets: [-1, 0, 1]
A = spdiags([sub_diag, main_diag, super_diag], [-1, 0, 1], n, n);
% Initialize the right-hand side vector b
b = [1; (2:n - 1)'; n];
% Solve the linear system A * x = b
x = A \ b;
```

what solution did you get?

c) Use tridisolve to solve the system.

i) same as above

ii)
```
x = tridisolve(sub_diag, main_diag, super_diag, b);
```

iii) I will not be writing out the solution since it is a huge matrix…

d) Use condest to estimate the condition of the coefficient matrix.

i) The estimated condition number for this matrix is: 5.1000e+03, which is rather large.