CISC271
Fall 2006
Homework for week 4
in preparation for quiz 2
Solutions

This homework will give you some practice with reviewing some basic concepts of linear algebra and Gaussian elimination.

1. Recktenwald Chapter 7. questions 2, 3 and 7.

   Note: These are review questions. Solutions will not be posted for the questions from Chapter 7.

8-16 **Solution:**

Here is an m-file that I wrote that does frontward substitution to solve Lx = b when L is lower triangular . Note that I check to make sure that the matrix is in fact lower triangular.

```
function  x = lsolveDR(L,b)
% lsolve  Lx = b where L is lower triangular
%
%
% Synopsis:  x = lsolve(L,b)
%
%
% Input:     L,b  = coefficient matrix and right hand side vector
%
%
% Output:    x = solution vector, if solution exists

[m,n] = size(L);
```

```
if m~=n,  error('A matrix needs to be square');  end
% Check to see if L is lower triangular

if sum(sum(L ~= tril(L)))> 0 , error('Matrix not lower triangular'); end



x = zeros(n,1);          %  preallocate memory for and initialize x
x(1) = b(1)/L(1,1);
for i=2:n
  x(i) = (b(i) - L(i,1:i-1)*x(1:i-1))/L(i,i);
end
```

Here is a sample of using lsolveDR

```
EDU>> A = rand(3)

A =
    0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214

EDU>> [L,U] = lu(A)

L =
   1.0000         0         0
    0.2433    1.0000         0
    0.6387    0.5843    1.0000


U =
     0.9501    0.4860    0.4565
         0          0.7731   -0.0925
         0          0               0.5839

EDU>> b = rand(3,1)

b =

    0.9218
    0.7382
    0.1763

EDU>> lsolveDR(A,b)
??? Error using ==> lsolve
```

```
Matrix not lower triangular

EDU>> lsolveDR(L,b)

ans =

    0.9218
    0.5140
   -0.7128
```

8-23 Solution: This question asks to find the equation of a plane that passes through 3 points.

Use a 3 by 3 array XY

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}$$

And letting the vector $z = [z_1 z_2 z_3]^T$ and $a = [a_1 a_2 a_3]^T$, we have the system $XYa = z$, and we need to solve for a.

Here is my m-file

```
function a = PlaneDR(p1,p2,p3)

% Given three points with coordinates (x,y,z) return coefficients a
% such that a1 xi + a2 yi + a3 = zi

XY = eye(3);
XY(:,1) = [p1(1) p2(1) p3(1)]
XY(:,2) = [p1(2) p2(2) p3(2)];
XY(:,3) = [ 1 1 1]'
z = [p1(3) p2(3) p3(3)]'
a = XY \ z
```

I plugged in the given values and obtained $a = [-1, -1, 1]^T$

For the given points the $z - coordinatevalue$ is 0.5 for the first 3 and 0 for the last one.

8-28 A paint company is trying to recycle unpopular paint colours by mixing them together to create better colours. All paints are composed of four pigments (A,B,C,D). The current paints are composed of pigments (in percent) according to the following equations. are $p_1 = 80A + 16C + 4D$, $p_2 = 80B + 20C$, $p_3 = 30A + 10B + 60C$, $p_4 = 10A + 10B + 72C + 8D$. The better colour say p_5 is composed as follows: $p_5 = 40A + 27B + 31C + 2D$. How much of the paints $p_1 \ldots p_4$ should be used to make one gallon of p_5.

Solution:

We can formulate this as a system of linear equations. Set

```
A =

    80     0    30    10
     0    80    10    10
    16    20    60    72
     4     0     0     8

b =

    40
    27
    31
     2
```

and solve the system $Ax = b$. Using Matlab I obtained the solution $x = (0.4, 0.3, 0.25, 0.05)^T$. Thus the percentage of each paint used to form p_5 is $p_5 = 0.4p_1 + 0.3p_2 + 0.25p_3 + 0.05p_4$

3 Alice buys 3 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 3 cantaloupes for \$2.77. How much do single pieces of fruit cost?

Solution: To determine how much fruit costs, I set things up so that:

```
A =

     3    12     1
    12     0     2
     0     2     3

 b =

    2.3600
    5.2600
    2.7700
```

Now solving $Ax = b$, I obtained the values apples are 0.29 bananas are 0.05 and cantaloupes are 0.89.

4 The matrix factorization

LU = P A

can be used to compute the determinant of A. We have det(L)det(U) = det(P)det(A) Because L is triangular with ones on the diagonal, det(L) = 1. Because U is triangular, det(U) = $u_{11}u_{22}\ldots u_{nn}$. Because P is a permutation, det(P) = +1 if the number of interchanges is even and -1 if it is odd. So det(A) = $\pm u_{11}u_{22}\ldots u_{nn}$ Modify the luPiv function (from Recktenwald) so that it returns four outputs: Here's how I modified luPiv

```
function [L,U,pv,sig] = luPivSigDR(A,ptol)
% luPiv  LU factorization with partial pivoting
%
% Synopsis:  [L,U,pv] = luPivSigDR(A)
%            [L,U,pv] = luPivSigDR(A,ptol)
%
% Input:     A    = coefficient matrix
%            ptol = (optional) tolerance for detection of zero pivot
%                   Default:  ptol = 50*eps
%
% Output:    L,U = lower triangular matrix, L, and upper triangular
%                  matrix, U, such that A(pv,:) = L*U
%            pv  = index vector that records row exchanges used to select
%                  good pivots.  The row permutations performed during
%                  elimination can be applied to the right hand side vector
%                  with b(pv).  The L and U returned by luPiv are the
%                  factors of permuted matrix A(pv,:), which is equivalent
%                  to P*A where P is the permutation matrix created
%                  by the two statements P = eye(size(A));  P = P(pv,:).
%            sig =  +1 or -1 if pv is an even or odd permutation

if nargin<3, ptol = 50*eps;  end     %  Default tolerance for zero pivot
[m,n] = size(A);
if m~=n,  error('A matrix needs to be square');  end
pv = (1:n)';
sig = 1;    %DR zero swaps is even parity

for i = 1:n-1                          %  loop over pivot row
  [pivot,p] = max(abs(A(i:n,i)));    %  value and index of largest pivot
  ip = p + i - 1;                    %  p is index in subvector i:n
  if ip~=i                           %  ip is true row index of desired pivot
     A([i ip],:) = A([ip i],:);      %  swap the rows
     pv([i ip]) = pv([ip i]);        %  record pivot order
     sig = sig * -1;                  %DR  keep track of sig parity
  end
  pivot = A(i,i);
  if abs(pivot)<ptol, error('zero pivot encountered after row exchange'); end
  for k = i+1:n                            % row k is eliminated next
    A(k,i) = A(k,i)/pivot;        % compute and store multiplier
    A(k,i+1:n) = A(k,i+1:n) - A(k,i)*A(i,i+1:n); % row ops to eliminate A(k,i)
  end
end

L = eye(size(A)) + tril(A,-1);  %  extract L and U
U = triu(A);
```

Here is my determinant m-file called detDR

```
function Det = detDR(A)
%detDR   returns determinant of square matrix A using algorithm
%  on page  86 Q 2.7 of Moler
%   Det = detDR(A)

[n,m] = size(A);
if n ~= m error('matrix is not square'); end

[L,U,p,sig] = luPivSigDR(A);

Det = sig*prod(diag(U));
```