Table of Contents

```
function [L,U,P] = luFactor(A)

%BY: COLLIN ELMER, 2 APR 2019
%
%Performs LU Decomposition with partial pivoting on some square(n x n)
 matrix
%   INPUTS:
%       A - an n x n matrix. Error triggered if matrix is not square.
%   OUTPUTS:
%       L - Pivoted lower triangular matrix such that [L]*[U]=[P]*[A]
%       U - Pivoted upper triangular matrix such that [L]*[U]=[P]*[A]
%       P - Pivot matrix (modified identity matrix)
%
%   ***IMPORTANT***
%   Subtractive cancellation may occur in some instances, causing
%   [L]*[U]=[P]*[A] to no longer be true. Another possible result is
 some
%   error in the order of P and order of L column coefficients. This
 is due
%   to the lack of a comparison threshold between coefficients.
```

Check number of inputs

```
if nargin > 1
    error('Only one square matrix can be evaluated.');
end
```

DEFINE VARS

```
r=0;
c=0;
L=eye(size(A));
U=A;
P=eye(size(A));
```

Matrix dimensions error check (WORKS WITH 1ST COLUMN OF 0'S)

```
[r,c]=size(A);

if r~=c
    error('Incorrect matrix dimesions');
end
```

define pivot vertical matrix of counting coefficients (also storing a

refrence matrix of the same values)

```
piv=zeros(r);
pivRef=zeros(r);
for b=1:r
    piv(:,b)=[1:r];
    pivRef(:,b)=[1:r];
end
```

Calculate stuff *ANY SIZE***

```
% Counts the row iteration being evaluated
for a = 1:r-1
```

PIVOTING U & P, TRACKING PIVOTS IN "piv"

```
    % set Y,I using max(abs) of U from the current row iteration to
 the end
    [Y,I]=max(abs(U(a:r,:)));

    % because we excluded the finished row(s), check to see if the
    % position+a-1 equals the a value (row we're on)
    if I(a)~=1
        % switch the rows of U & P matrices that need to be switched
        U([a I(a)+a-1],:) = U([I(a)+a-1 a],:);
        P([a I(a)+a-1],:) = P([I(a)+a-1 a],:);

        % ignore the first pivot because L(1,1) is always 1, but
        % otherwise assign pivot change to piv matrix
        if a ~= 1
            piv([a I(a)+a-1],a) = piv([I(a)+a-1 a],a);
        end
    end
```

LU DECOMPOSITION

```
    for j = a:r-1
```

```
        % find forward elimination coeff, store in L matrix only if
 the
        % first spot is not already 0
        if U(a,a) ~= 0
            L(j+1,a)=(U(j+1,a)/U(a,a));

            % multiply coeff by equation 1 and subtract altered
 equation 1 from equation 2
            U(j+1,:)=U(j+1,:)-L(j+1,a)*U(a,:);
        end
    end

end
```

L PIVOTING

```
%column iteration
    for n=1:r-2

        % if piv(n+1,n+1) does not equal what its supposed to equal,
        % iterate n
        if piv(n+1,n+1)~=pivRef(n+1,n+1)

            %row iteration
            for m=n:r

                % m cant equal n because L(m,m) is always 1.
                if m~=n
                    % switch the L values in the right spots of every
                    % applicable column behind the one being evaluated
 ONLY
                    % ONCE, THEN BREAK
                    L([piv(m,m) m],1:n)=L([m piv(m,m)],1:n);
                    break
                end
            end
        end
    end

end
```

```
L =
    1.0000         0         0
   -0.2500    1.0000         0
    0.3750    0.2391    1.0000
U =
   -8.0000    1.0000   -2.0000
         0   -5.7500   -1.5000
         0         0    8.1087
P =
     0     0     1
     1     0     0
     0     1     0
```

Published with MATLAB® R2018a