Factorization:

```
%Ab = [1 1 0 3 4; 2 1 -1 1 1; 3 -1 -1 2 -3; -1 2 3 -1 4];  % unique solution
%Ab = [0 1 -1 -8; 1 1 1 8; 0 -1 -3 -12];
%Ab = [1 1 1 4; 2 2 1 6; 1 1 2 6];
%Ab = [2.11 -4.21 0.921 2.01; 4.01 10.2 -1.12 -3.09; 1.09 0.987 0.832 4.21];

A = [4 12 -16; 12 37 -43; -16 -43 98];
%A = [25 15 -5; 15 18 0; -5 0 11];
%A = [0 0 -1 1; 1 1 -1 2; -1 -1 2 0; 1 2 0 2];

for n=3%2.^[1:8]


[n, ~] = size(A);
Ab = [A, eye(n)];

Abo = Ab;
[n, m] = size(Ab);

Ms = cell(1,n-1);
Ls = cell(1,n-1);
U = A;
L = eye(n);

P = eye(n);

%s = max(abs(Ab(:,1:n)),[],2);
%if any(s==0)
%  error('Infinite or no solutions');
%end

opE = 0;

%Gaussian elimination
for k=1:n
  i = find(Ab(k:n,k)~=0,1);  % partial pivoting when one pivot is zero
  %[~, i] = max(abs(Ab(k:n,k)));  % partial pivoting when one pivot is "small"
  %[~, i] = max(abs(Ab(k:n,k))./s(k:n));
  p = k+i-1;
  if Ab(p,k) ~= 0
%    tmp = s(k);
%    s(k) = s(p);
%    s(p) = tmp;
    tmp = Ab(k,:);
    Ab(k,:) = Ab(p,:);
    Ab(p,:) = tmp;
    tmp = P(k,:);
    P(k,:) = P(p,:);
    P(p,:) = tmp;
  else
    error('No unique solution. Infinite or no solution.');
  end
  M = eye(n);
```

```
  Mi = eye(n);
  Ab(k,:) = Ab(k,:)/sqrt(Ab(k,k));
  Mi(k,k) = Ab(k,k);
  M(k,k) = 1/Ab(k,k);
  for i=(k+1):n
    m = Ab(i,k)/Ab(k,k);
    Ab(i,:) = Ab(i,:) - m*Ab(k,:);
    opE = opE + size(Ab,2);
    M(i,k) = -m/Ab(k,k);
    Mi(i,k) = m;
  end
%  Ab(k,k:end) = Ab(k,k:end)/sqrt(Ab(k,k));

  Ms{k} = M;
  Ls{k} = Mi;

  U = M*U;
  L = L*Mi;
end

%backward substitution
if Ab(n,n) == 0
 error('No solution');
end
m = size(Ab,2)-n;
x = zeros(m,n);
for j=1:m
for i=n:-1:1
  x(j,i) = (Ab(i,n+j) - sum(Ab(i,i+1:n).*x(j,i+1:n)))/Ab(i,i);
end
end

opE

%figure(1);
%hold on;
%loglog(n,opE,'ob');
%loglog(n,n^3,'+k');
%loglog(n,n^2,'sg');
%loglog(n,n,'^r');
%hold off;

end

return
A = Abo(1:n,1:n);
b = Abo(:,end);
A*x'

A*x' - b


function [Ub, p] = gauss_elim_srpp(Ab)
[n, ~] = size(Ab);
```

```
P = eye(n);
s = max(abs(Ab(:,1:n)),[],2);
if any(s==0)
    error('Infinite or no solutions');
end


%Gaussian elimination
for k=1:n
    %i = find(Ab(k:n,k)~=0,1);  % partial pivoting when one pivot is zero
    %[~, i] = max(abs(Ab(k:n,k)));  % partial pivoting when one pivot is "small"
    [~, p] = max(abs(Ab(k:n,k))./s(k:n));
    p = p + k - 1;
   % disp(p)
    %disp(k)
    if p ~= k
        tmp = s(k);
        s(k) = s(p);
        s(p) = tmp;
        tmp = Ab(k,:);
        Ab(k,:) = Ab(p,:);
        Ab(p,:) = tmp;
        tmp = P(k,:);
        P(k,:) = P(p,:);
        P(p,:) = tmp;
    end
    if Ab(k,k) == 0
        error("No solution");
    end
    Mi = eye(n);
    for i=(k+1):n
        m = Ab(i,k)/Ab(k,k);
        Ab(i,:) = Ab(i,:) - m*Ab(k,:);
    end
    %disp(Ab)
end
Ub = Ab;
p = P;
end

function [x] = backward_sub(Ub)
    [n, m] = size(Ub);
    if Ub(n,n) == 0
        error('No solution');
    end
    m = size(Ub,2)-n;
    x = zeros(m,n);
    for j=1:m
        for i=n:-1:1
          x(j,i) = (Ub(i,n+j) - sum(Ub(i,i+1:n).*x(j,i+1:n)))/Ub(i,i);
        end
    end
end
```

Problem 1:

```
A = [1 -2 3 0; 1 -2 3 1; 1 -2 2 -2; 2 1 3 -1];
[Ub, P] = gauss_elim_srpp(A);
A = P * A;
%run GEandLU.m with A as above:
U =

    2.0000    1.0000    3.0000   -1.0000
         0   -2.5000    0.5000   -1.5000
         0         0    1.0000    3.0000
         0         0         0   -1.0000

L =

    1.0000         0         0         0
    0.5000    1.0000         0         0
    0.5000    1.0000    1.0000         0
    0.5000    1.0000    1.0000    1.0000

P =

     0     0     0     1
     0     0     1     0
     0     1     0     0
     1     0     0     0
```

Problem 2:

```
>> A = [1 -1 1; 7 5 -1; 2 1 1];
>> B = [5 2 8 4; 8 2 0 1; 7 2 9 1];
>> [Ub, P] = gauss_elim_srpp([A B])

Ub =

    1.0000   -1.0000    1.0000    5.0000    2.0000    8.0000    4.0000
         0   12.0000   -8.0000  -27.0000  -12.0000  -56.0000  -27.0000
         0         0    1.0000    3.7500    1.0000    7.0000   -0.2500


P =

     1     0     0
     0     1     0
     0     0     1
>> x = backward_sub(Ub);
>> x

x =

    1.5000    0.2500    3.7500
    0.6667   -0.3333    1.0000
    1.0000         0    7.0000
    1.8333   -2.4167   -0.2500
```

Each row in this matrix is a set of solutions. Transpose the matrix to get each column of x.

```
>> A\B

ans =

    1.5000    0.6667    1.0000    1.8333
    0.2500   -0.3333    0.0000   -2.4167
    3.7500    1.0000    7.0000   -0.2500
```

Problem 3:
The matrix in 5c is singular, so it fails. Trying with problem 6d:

```
>> A = [2 0 1 2; 1 1 0 2; 2 -1 3 1; 3 -1 4 3];
>> [Ub, P] = gauss_elim_srpp([A eye(size(A))])

Ub =

    2.0000         0    1.0000    2.0000    1.0000         0         0         0
         0    1.0000   -0.5000    1.0000   -0.5000    1.0000         0         0
         0         0    1.5000         0   -1.5000    1.0000    1.0000         0
         0         0         0    1.0000         0   -0.3333   -1.3333    1.0000


P =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1

>> backward_sub(Ub)

ans =

    1.0000   -1.0000   -1.0000         0
   -0.0000    1.6667    0.6667   -0.3333
    1.0000    1.6667    0.6667   -1.3333
   -1.0000   -1.0000         0    1.0000

>> ans'

ans =

    1.0000   -0.0000    1.0000   -1.0000
   -1.0000    1.6667    1.6667   -1.0000
   -1.0000    0.6667    0.6667         0
         0   -0.3333   -1.3333    1.0000
```

This final ans is the inverse of the matrix.