REPORT FOR SOLVING LINEAR SYSTEM

Abstract

Report for finding error and time taken to solve linear system using different methods.

Ram Prasad Gaire
rpgaire@ucdavis.edu

Given report describes the method to find the solution to system of linear equation.
For the matrix in the problem, the lower matrix found is:

```
   1.0000         0         0
   2.0000    1.0000         0
   2.5000    1.5625    1.0000
```

And the upper matrix is

```
2.0000    7.0000    9.0000
     0   -8.0000   -7.0000
     0         0   -8.5625
```

Values of x,y found from the factored matrices is

```
x =

  -0.2920
   0.4745
   0.0292

y =

   3.0000
  -4.0000
  -0.2500
```

Values of x found for each method is given below:

Gaussian method

```
-0.2920
   0.4745
   0.0292
```

Lu methods

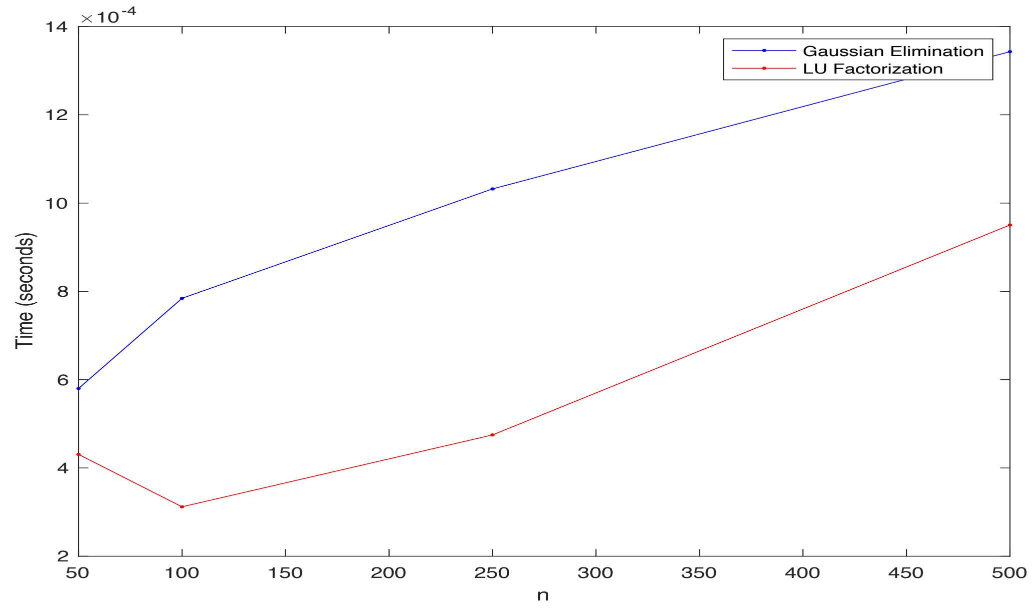
```
1.5000
  -0.6667
  -1.0556
```

Table 1: Error for $||Ax - b||_2$for different sized matrices GE, and LU factorizations is given below:

n	Error for GAUSSIAN ELEMENATION	Error for LU FACTORIZATION
50	5.813047e+01	4.705305e+01
100	1.101581e+02	1.001289e+02
250	2.577317e+02	2.493154e+02
500	5.196045e+02	9.769218e+02

From the given table we can see that, error is more in LU factorization than in gaussian elimination methods.

Given graph represent time taken to solve matrix for different size of integration.



From the one can conclude that LU factorization is more efficient than gaussian elimination methods.

```
%Code
function[x] = Gaussian_elemenation(A,b)
[m,n] =  size(A);
tic
for k = 1:n-1
    for i = k+1:m
        factor = A(i,k)/A(k,k);
        A(i,k:n) = A(i,k:n)-factor * A(k,k:n);
        b(i) = b(i) - factor * b(k);
    end
end
x =  backSubstitution(A, b);
toc
end
```

```
%code
function[x] = backSubstitution(A, b)
[m, n] = size(A);
x = zeros(n,1);
x(n) = b(n) / A(n,n);
tic
for i = n-1:-1:1
    x(i) = (b(i) - A(i,i+1:n) * x(i+1:n)) / A(i,i);
end
toc
end
```

```
%code
function [x] = ForwardSubst(A, b)
[m, n] = size(A);
x = zeros(n, 1);
x(1) = b(1) / A(1,1);
tic
for i = 2:n
sum = b(i);
for j = 1:i-1
sum = sum - A(i,j) * x(j);
end
x(i) = sum / A(i,i);
end
toc
end
```

```
%code
function [L,U] = LUfactorization(A)
[m, n] = size(A);
L = eye(m);
U = A;
tic
for k = 1:n-1
    for i = k+1:m
        factor = U(i,k) / U(k,k);
        L(i,k) = factor;
        U(i,k:n) = U(i,k:n) - factor * U(k,k:n);
    end
```

```
end
toc
end
```

%code

```
function [x,y] = LUsolve(A, b)
[L,U] = LUfactorization(A);
y = ForwardSubst(L,b);
x = backSubstitution(U,y);
end
```

% Code to call function and plot graph

```
A = [2 7 9;4 6 11;5 5 3]
b = [3;2;1]

[x] = Gaussian_elemenation(A,b)
[x] = backSubstitution(A, b)
[x] = ForwardSubst(A, b)
[L,U] = LUfactorization(A)
[x,y] = LUsolve(A, b)
[x] = Gaussian_elemenation(A,b)
n = [50,100,250,500];
tic
for i = 1:length(n)
    A = 5 ∗ eye(n(i)) + randn(n(i));
    b = randn(n(i), 1);
    [x,y] = LUsolve(A, b)
    error = norm(A.∗x − b)
    fprintf('For n = %d, the error ||Ax − b||^2 = %d\n', n(i), error)
    t = toc
end

n_values = [50,100,250,500];
times_gaussian_elimination = zeros(size(n_values));
times_LU_factorization = zeros(size(n_values));

for i = 1:length(n_values)
n = n_values(i);
A = 5 ∗ eye(n) + randn(n);
b = randn(n, 1);

tic;
x_gaussian = Gaussian_elemenation(A, b)
times_gaussian_elimination(i) = toc;

tic;
[L, U] = LUfactorization(A)
[x] = LUsolve(A, b);
times_LU_factorization(i) = toc
end
```

```
figure;
plot(n_values, times_gaussian_elimination, 'b.-')
hold on;
plot(n_values, times_LU_factorization, 'r.-')
xlabel('n');
ylabel('Time (seconds)');
legend('Gaussian Elimination', 'LU Factorization');
```