```
format long
```

Problema 12. Analiza Cramer si Gepp

flops sunt calculati in functii, in dreptul fiecarei operatii se incrementeaza flops cu numarul corespunzatoar.

```
n = 2;
A = randi(10, n, n);
b = A * ones(n, 1);
[x, flops] = cramer_method(A, b, n);
x, flops
```

```
x = 2×1
     1
     1
flops =
    11
```

Solutie buna, flops Cramer = 11

```
[x, flops] = Gepp(A, b);
x, flops
```

```
x = 2×1
     1
     1
flops =
     9
```

Solutie buna, flops Gepp = 9

```
n = 3;
A = randi(10, n, n);
b = A * ones(n, 1);
[x, flops] = cramer_method(A, b, n);
x, flops
```

```
x = 3×1
     1
     1
     1
flops =
    59
```

Solutie buna, flops Crammer = 59

```
[x, flops] = Gepp(A, b);
x, flops
```

```
x = 3×1
```

```
   1.000000000000002
   0.999999999999999
   0.999999999999999
flops =
    28
```

Solutie buna, flops Gepp = 28

Problema 13. Descompunere LUP pentru matrice Hessenberg.

```
n = 20;
A = randi(10, n);
H=triu(A,-1)
```

```
H = 20×20
     6     7    10     3     6     2     9     9     1    10     6     1     2 ···
     9     8     9     7    10     9     3     6     8     7     8     3     3
     0     5     2     4     1     1     5     8    10     3     8     2     4
     0     0     9     6     9    10     5     1     8    10     4     7     4
     0     0     0     7    10     4     6     9     3     8     3     6     4
     0     0     0     0     1     3    10     8     4     9     3     6     8
     0     0     0     0     0     6     8     1    10     8     3     7    10
     0     0     0     0     0     0     7     5     1     9     1     5     7
     0     0     0     0     0     0     0    10     9    10     5     3     9
     0     0     0     0     0     0     0     0     6     2     7    10     5
     ⋮
```

```
b=H*ones(n, 1);
```

```
[L, U, P] = lup(H);
Y = forwardsubst(L, P * b);
sol = backsubst(U, Y)
```

```
sol = 20×1
   0.999999999999994
   1.000000000000020
   0.999999999999983
   0.999999999999970
   1.000000000000032
   0.999999999999981
   1.000000000000026
   0.999999999999997
   1.000000000000001
   0.999999999999994
     ⋮
```

```
norm(b - H * sol) / norm(b) % Eroarea relativa
```

```
ans =
     1.299494414686878e-16
```

Complexitatea lui LUP este O(n^3), dar acest algoritm se poate imbunatati din moment ce o matrice Hessenberg este aproape o matrice triunghiulata superior. Imbunatatirile vin la gasirea pivotului (trebuie ales dintre 2 elemente) si la calcularea complementului Schur (trebuie doar pentru 1 element).

```
% lup_hessenberg are complexitate O(n^2)
[L, U, P] = lup_hessenberg(H);
Y = forwardsubst(L, P * b);
sol = backsubst(U, Y)
```

```
sol = 20×1
   0.999999999999994
   1.000000000000020
   0.999999999999983
   0.999999999999970
   1.000000000000032
   0.999999999999981
   1.000000000000026
   0.999999999999997
   1.000000000000001
   0.999999999999994
      ⋮
```

```
norm(b - H * sol) / norm(b) % Eroarea relativa
```

```
ans =
     1.299494414686878e-16
```

Problema 14. Ax=b supradeterminat

```
m = 10; n = 2;
A = randi(10, m, n);
b = A * ones(n, 1);
```

```
% Rezolvare A*x = b folosind operatorul '\'
x1 = A \ b
```

```
x1 = 2×1
   1.000000000000000
   1.000000000000000
```

```
% Rezolvarea sistemului A' * A * x = A' * b folosind Cholesky
ATA = A' * A;
ATb = A' * b;
R = Cholesky(ATA);
y = (R.') \ ATb;
x2 = R \ y
```

```
x2 = 2×1
   1.000000000000001
   0.999999999999999
```

```
err1 = norm(x1 - ones(n, 1))
```

```
err1 =
     4.002966042486721e-16
```

```
err2 = norm(x2 - ones(n, 1))
```

```
err2 =
     1.110223024625157e-15
```

```
cond_A = cond(A)
```

```
cond_A =
   3.463722820029735
```

```
cond_ATA = cond(ATA)
```

```
cond_ATA =
  11.997375773994733
```

Conditionarea lui A' * A este mai mare decat conditionarea lui A, deci prin calculul lui A' si A' * A se introduc erori. Asadar in rezolvarea sistemului eroarea este mai mare.