Pràctica

(a) Com canvia el nombre de condició d'aquesta matriu quan l'ordre creix? Presenta un estudi per n=5÷30.

```
for tam=5:30;
    A = zeros(tam,tam);
    A(1,1) = -1;
    i = 1;
    j = 2;
    i2 = 2;
    j2 = 1;
    while i < tam;
        A(i,j) = 1;
        A(i2,j) = 4;
        A(i2,j2) = 1;
        i = i+1;
        j = j+1;
        i2 = i2+1;
        j2 = j2+1;
    end
    A(tam,tam) = -1;
    for i=1:tam;
        if i == 1;
            b = [0];
        elseif i == tam;
            b = [b;0];
        else
            b = [b;6];
        end
    end
    if tam == 5;
        R = [tam,cond(A)];
        x = [tam];
        y = [cond(A)];
    else R = [R;tam,cond(A)];
        x = [x;tam];
        y = [y;cond(A)];
    end
end
Ta=array2table(R,'VariableNames',{'tam','cond'})
```

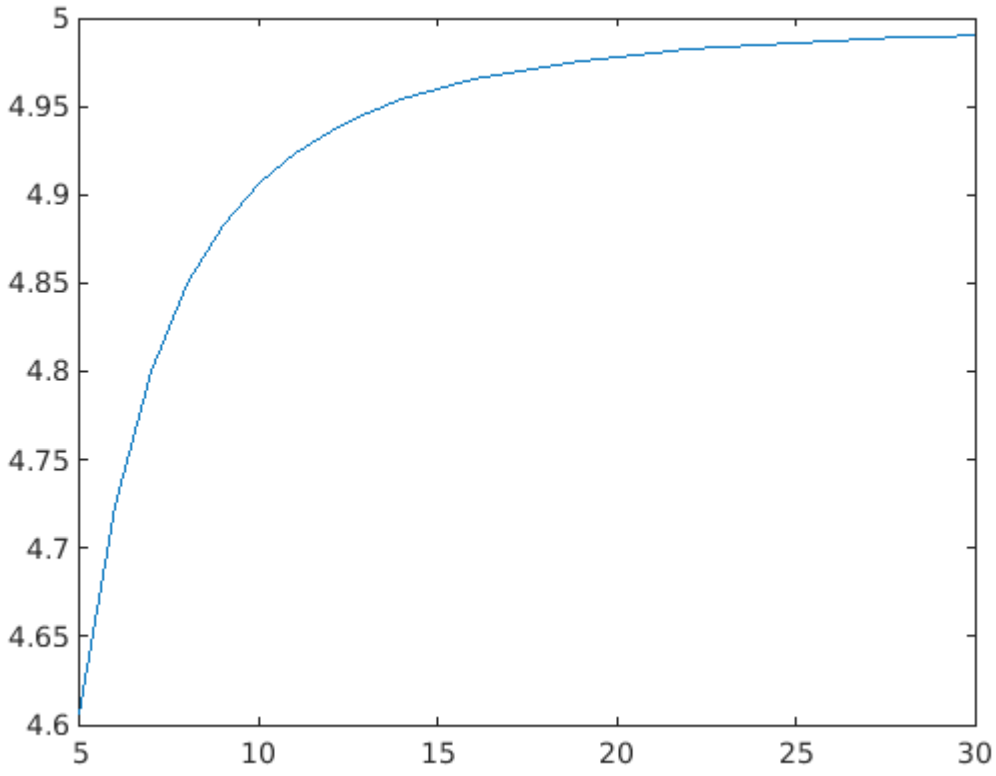
Ta = 26×2 table

	tam	cond
1	5	4.6062
2	6	4.7238
3	7	4.7998
4	8	4.8494
5	9	4.8829
6	10	4.9064

	tam	cond
7	11	4.9235
8	12	4.9363
9	13	4.9462
10	14	4.9539
11	15	4.9601
12	16	4.9651
13	17	4.9693
14	18	4.9727

⋮

```
Pa=plot(x,y)
```



A la taula **Ta** podem veure com a mesura que augmenta l'ordre de la matriu, també augmenta el nombre de condició. A la gràfica **Pa** podem veure com l'augment de la condició s'aproxima molt a una funció logarítimica i que a partir d'un cert ordre prou gran de matriu, l'**augment** del nombre de condició tendeix a disminuir.

(b) Resoleu el sistema lineal =Ax=b per eliminació gaussiana sense pivotament.

```
format long
for tam=5:30;
```

```
    A = zeros(tam,tam);
    A(1,1) = -1;
    i = 1;
    j = 2;
    i2 = 2;
    j2 = 1;
    while i < tam;
        A(i,j) = 1;
        A(i2,j) = 4;
        A(i2,j2) = 1;
        i = i+1;
        j = j+1;
        i2 = i2+1;
        j2 = j2+1;
    end
    A(tam,tam) = -1;
    for i=1:tam;
        if i == 1;
            b = [0];
        elseif i == tam;
            b = [b;0];
        else
            b = [b;6];
        end
    end
    tic
    [L,U,P] = lu(A);
    prova = norm(P*A-L*U,1);
    %la solució del sistema és en dos etapes, primer resolem Ly=Pb i després  Ux=y
    y = linsolve(L,P*b); xs = linsolve(U,y);  % linsolve més eficient
    t = toc;
    if tam == 5;
        R = [tam,t];
        TA = [t];
    else
        R = [R;tam,t];
        TA = [TA;t];
    end
end
Tb=array2table(R,'VariableNames',{'tam','t'})
```

Tb = 26×2 table

	tam	t
1	5	0.000433...
2	6	0.000204...
3	7	0.000348...
4	8	0.000133...
5	9	0.001108...
6	10	0.000053...
7	11	0.000027...

	tam	t
8	12	0.000025...
9	13	0.000027...
10	14	0.000028...
11	15	0.000031...
12	16	0.000031...
13	17	0.000053...
14	18	0.000045...
15	19	0.000046...
16	20	0.000047...
17	21	0.000051...
18	22	0.000053...
19	23	0.000081...
20	24	0.000058...
21	25	0.000067...
22	26	0.000061...
23	27	0.000062...
24	28	0.000072...
25	29	0.000080...
26	30	0.000074...

(c) Resoleu el sistema lineal = fent ús de les funcions de Matlab® decompositon i linsolve.

```
format long
for tam=5:30
    A = zeros(tam,tam);
    A(1,1) = -1;
    i = 1;
    j = 2;
    i2 = 2;
    j2 = 1;
    while i < tam;
        A(i,j) = 1;
        A(i2,j) = 4;
        A(i2,j2) = 1;
        i = i+1;
        j = j+1;
        i2 = i2+1;
        j2 = j2+1;
    end
```

```
    A(tam,tam) = -1;
    for i=1:tam;
        if i == 1;
            b = [0];
        elseif i == tam;
            b = [b;0];
        else
            b = [b;6];
        end
    end
    tic
    da = decomposition(A);
    da\b;
    td = toc;
    tic
    linsolve(A,b);
    tl = toc;
    if tam == 5;
        R = [tam,td,tl];
        TD = [td];
        TL = [tl];
    else
        R = [R;tam,td,tl];
        TD = [TD;td];
        TL = [TL;tl];
    end
end
Tc=array2table(R,'VariableNames',{'tam','td','tl'})
```

Tc = 26×3 table

	tam	td	tl
1	5	5.030000000...	8.700000000...
2	6	4.180000000...	5.900000000...
3	7	5.020000000...	6.100000000...
4	8	3.070000000...	5.400000000...
5	9	6.760000000...	3.310000000...
6	10	1.950000000...	1.900000000...
7	11	1.290000000...	1.400000000...
8	12	1.170000000...	1.400000000...
9	13	1.130000000...	1.300000000...
10	14	1.320000000...	1.500000000...
11	15	1.170000000...	1.400000000...
12	16	1.130000000...	1.500000000...
13	17	1.150000000...	2.400000000...
14	18	1.200000000...	2.200000000...

	tam	td	tl
15	19	1.160000000...	2.100000000...
16	20	1.160000000...	2.200000000...
17	21	1.160000000...	2.300000000...
18	22	1.160000000...	2.300000000...
19	23	1.170000000...	2.400000000...
20	24	1.180000000...	2.500000000...
21	25	1.520000000...	2.800000000...
22	26	1.390000000...	2.900000000...
23	27	1.730000000...	3.500000000...
24	28	1.640000000...	3.300000000...
25	29	1.450000000...	5.900000000...
26	30	2.010000000...	3.700000000...

(d) Resoleu el sistema lineal =Ax=b pel mètode de Gauss-Seidel.

```
format long
for tam=5:30
    A = zeros(tam,tam);
    A(1,1) = -1;
    i = 1;
    j = 2;
    i2 = 2;
    j2 = 1;
    while i < tam;
        A(i,j) = 1;
        A(i2,j) = 4;
        A(i2,j2) = 1;
        i = i+1;
        j = j+1;
        i2 = i2+1;
        j2 = j2+1;
    end
    A(tam,tam) = -1;
    for i=1:tam;
        if i == 1;
            b = [0];
        elseif i == tam;
            b = [b;0];
        else
            b = [b;6];
        end
    end
    tic
    D = diag(diag(A));
    L = tril(A-D);
```

```
    U = triu(A-D);
    DI = inv(D+L);
    Bgs =-DI*U;
    cgs = DI*b;
    rhogs = abs(eigs(Bgs,1));
    if rhogs < 1;
        k = 0;
        xs = zeros(size(b));
        r =norm(A*xs-b,1);
        while (k < 100 && r > 5.0e-4);
            k=k+1;
            xs = Bgs*xs+cgs;
            r =norm(A*xs-b,1);
        end
        %fprintf('métode de Gauss-Seidel convergent, en k = %d, el residu és %5.4g i el
        xs;
    %else
        %fprintf('métode de Gauss-Seidel no convergent');
    end
    t = toc;
    if tam == 5;
        R = [tam,t];
        TG = [t];
    else
        R = [R;tam,t];
        TG = [TG;t];
    end
end
Td=array2table(R,'VariableNames',{'x','t'})
```

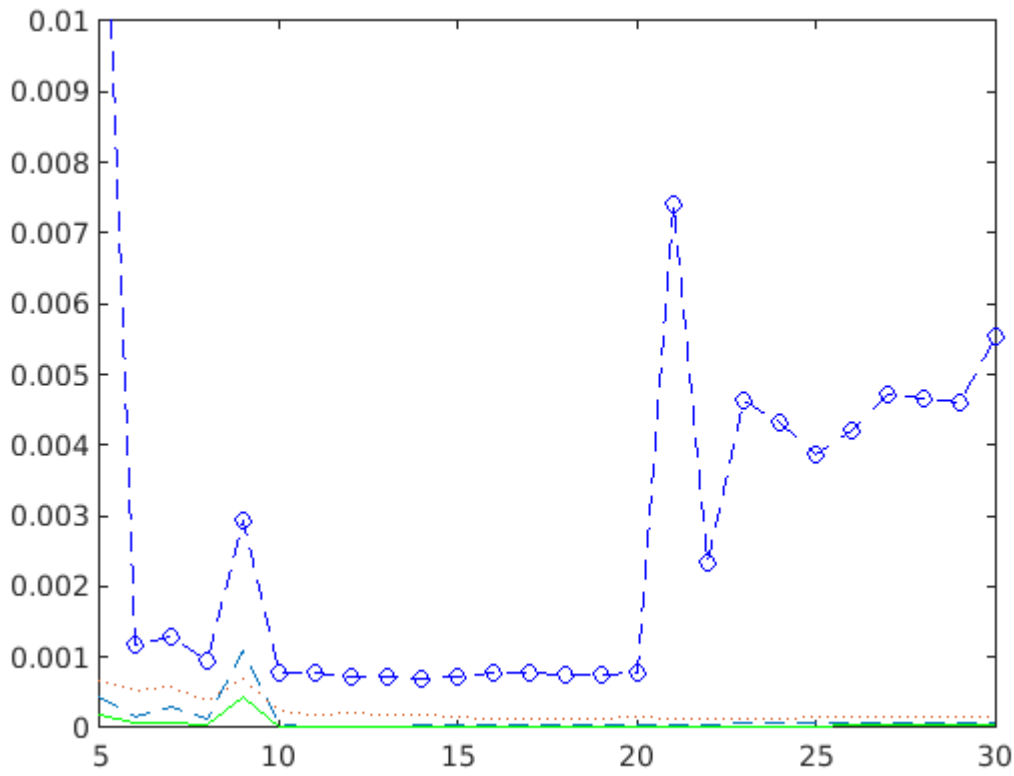
Td = 26×2 table

	x	t
1	5	0.003109...
2	6	0.000733...
3	7	0.000911...
4	8	0.000565...
5	9	0.002456...
6	10	0.000436...
7	11	0.000331...
8	12	0.000316...
9	13	0.000420...
10	14	0.000381...
11	15	0.000347...
12	16	0.000390...
13	17	0.000370...
14	18	0.000376...

	x	t
15	19	0.000378...
16	20	0.000401...
17	21	0.002944...
18	22	0.001792...
19	23	0.004636...
20	24	0.003942...
21	25	0.003696...
22	26	0.004379...
23	27	0.004482...
24	28	0.004692...
25	29	0.005109...
26	30	0.005375...

```
Pd=plot(x,TA,'--',x,TD,':',x,TL,'g',x,TG,'b--o')

xlim([5.0 30.0])
ylim([0.0000 0.010])
```



(f) Comenteu els avantatges i els inconvenients dels tres mètodes i l'evolució dels resultats quan n es fa gran.

1. Eliminació gaussiana sense pivotament (Factorització LU)

- **Inconvenients**

Si algún menor principal és zero, pot no existir la descomposició LU.

- **Avantatges**

1. Per a qualsevol matriu no singular, les files poden ser reordenades de tal manera que existeixi una descomposició LU.

2. És més ràpid i convenient per resoldre múltiples vegades les equacions per diferents b sense haver d'aplicar l'eliminació gaussiana cada vegada.

2. Decomposition i linsolve

Decomposition és típicament més ràpid i és molt útil per a resoldre problemes que necessiten solucions repetidament ja que la descomposició de la matriu només s'ha de fer una vegada.

Linsolve utilitza la factorització LU amb pivotament parcial i per els altres casos utilitza factorització QR amb pivotament per columna. La factorització QR és més costosa que la LU.

3. Gauss-Seidel

En general, si convergeix el mètode Jacobi, el mètode de Gauss-Seidel convergirà més ràpidament que el mètode Jacobi, tot i que encara és relativament lent.

De fet, aquest és el mètode més lent (es pot comprovar a la gràfica Pd) i la que dòna més errors d'aproximació en la solució.

4. Evolució dels resultats quan *n* es fa gran

La solució trobada pels diferents mètodes són bastants similars per a tots els ordres de matriu i no hi ha gaire errors menys en el cas del Gauss-Seidel.

Amb aquest métode, hi han més errors d'aproximació quan més petit és l'ordre.