L5 Primera Part
Marta Granero i Martí

Exercici 1

$$a_{ij} = \frac{1}{i+j} \quad i \quad b_i = \frac{1}{i} + \frac{1}{i+1} + \cdots + \frac{1}{i+n-1}, \quad 1 \le i, j \le n$$

a) Com canvia el nombre de condició d'aquesta matriu quan l'ordre creix? Presenta un estudi per = 3,4,…,10,…15

```
%Per n = 3
n = 3;
a3 = zeros(n,n);
for (i = 1:n)
    for (j = 1:n)
        a3(i,j) = 1/(i+j);
    end
end

b3 = zeros(n,0);
for (i = 1:n)
    b3(i) = 1/i;
end
a3;
b3;
per3 = cond(a3,1);
array2table(per3, 'VariableNames',{'Nombre de condició: n = 3'})
```

ans = table table

	Nombre de condició: n = 3
1	2.0150e+03

```
%Per n = 4
n = 4;
a4 = zeros(n,n);
for (i = 1:n)
    for (j = 1:n)
        a4(i,j) = 1/(i+j);
    end
end

b4 = zeros(n,0);
for (i = 1:n)
    b4(i) = 1/i;
end
per4 = cond(a4,1);
```

```
array2table(per4, 'VariableNames',{'Nombre de condició: n = 4'})
```

ans = table table

	Nombre de condició: n = 4
1	8.1389e+04

```
%Per n = 10
n = 10;
a10 = zeros(n,n);
for (i = 1:n)
    for (j = 1:n)
        a10(i,j) = 1/(i+j);
    end
end

b10 = zeros(n,0);
for (i = 1:n)
    b10(i) = 1/i;
end
per10 = cond(a10,1);
array2table(per10, 'VariableNames',{'Nombre de condició: n = 10'})
```

ans = table table

	Nombre de condició: n = 10
1	1.3285e+14

```
%Per n = 15
n = 15;
a15 = zeros(n,n);
for (i = 1:n)
    for (j = 1:n)
        a15(i,j) = 1/(i+j);
    end
end

b15 = zeros(n,0);
for (i = 1:n)
    b15(i) = 1/i;
end
per15 = cond(a15,1);
```

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 1.819112e-18.

```
array2table(per15, 'VariableNames',{'Nombre de condició: n = 15'})
```

ans = table table

	Nombre de condició: n = 15
1	5.7515e+17

```
%A mesura que n augmenta el nombre de condició també ho fa
```

```
valors = [per3;per4;per10;per15];
nombreCondicio = ["n = 3"; "n = 4"; "n = 10"; "n = 15"];
taula = table(nombreCondicio,valors)
```

taula = 4×2 table

	nombreCondicio	valors
1	"n = 3"	2.0150e+03
2	"n = 4"	8.1389e+04
3	"n = 10"	1.3285e+14
4	"n = 15"	5.7515e+17

b) Resoleu el sistema lineal per eliminació gaussiana sense pivotament. Presenta un estudi per = 3÷10

```
tic
x3 = ElimGaussSensePivot(a3,b3');
tempsGauss3 = toc
```

```
tempsGauss3 = 0.0180
```

```
tic
x4 = ElimGaussSensePivot(a4,b4');
tempsGauss4 = toc
```

```
tempsGauss4 = 0.0080
```

```
tic
x10 = ElimGaussSensePivot(a10,b10');
tempsGauss10 = toc
```

```
tempsGauss10 = 0.0151
```

```
tempsGauss = [tempsGauss3;tempsGauss4;tempsGauss10];

x15 = ElimGaussSensePivot(a15,b15');
```

c) Resoleu el sistema lineal fent ús de les funcions de Matlab® decompositon i linsolve. Presenta un estudi per = 3,4,…,10,…

```
tic
sol3 = linsolve(a3,b3');
tempsSolve3 = toc
```

```
tempsSolve3 = 0.0041
```

```
tic
sol4 = linsolve(a4,b4');
tempsSolve4 = toc
```

```
tempsSolve4 = 0.0036
```

```
tic
sol10 = linsolve(a10,b10');
tempsSolve10 = toc
```

```
tempsSolve10 = 0.0085
```

```
tempsSolve = [tempsSolve3;tempsSolve4;tempsSolve10];

sol15 = linsolve(a15,b15');
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
1.819112e-18.
```

d) Resoleu el sistema lineal pel mètode de Gauss-Seidel. Presenta un estudi per =3÷10

```
% Càlculs per fer n = 3
tic
D = diag(diag(a3));
d = diag(1 ./ diag(a3));

L = tril(a3,-1);
U = triu(a3,1);
dG = inv(L+D);
Bgs = -dG*U; %matriu iteració
cgs = dG*b3'; %vector

rhoGS = max(abs(eig(Bgs))); %radi espectral del mètode de Gauss-Seidel

if (rhoGS + eps >= 1) resp = "mètode de GAUSS-SEIDEL divergent";
else resp = "mètode de GAUSS-SEIDEL convergent";
end

gauss = {rhoGS, resp}';

table(gauss, 'VariableNames',{'Gauss-Seidel'})
```

ans = 2×1 table

	Gauss-Seidel
1	0.9943
2	"mètode de GAUSS-SEIDEL convergent"

```
%mètode de Gauss
clear residu criteriParada e delta
if rhoGS >=1
    error('mètode de gauss-seidel divergent')
else
```

```
    n = 1000; i = 0;
    x = zeros(size(b3'));
    tol = 0.5e-8;
    criteriParada(1) = 1;
    while(i<n && criteriParada(max(i,1)) > tol)
        i = i+1; y = Bgs*x+cgs;
        residu(i) = norm(b3'-a3*y,'inf');
        e(i) = norm(y-x,'inf');
        delta(i) = norm(y-x,'inf');
        criteriParada(i) = delta(i)/norm(y,'inf');
        x = y;
    end
    tempsGaussSeidel3 = toc
    table(residu',e',delta',criteriParada','VariableNames',{'residu','e^k','delta^k','
    table(i, residu(1), residu(end), 'VariableNames',{'# iteracions', 'Primer Residu',
    x(1);
end
```

```
tempsGaussSeidel3 = 0.2580
ans = 1000×4 table
```

	residu	e^k	delta^k	stop
1	0.2722	2	2	1
2	0.2230	0.5659	0.5659	0.2224
3	0.1834	0.4847	0.4847	0.1621
4	0.1515	0.4192	0.4192	0.1249
5	0.1259	0.3663	0.3663	0.1001
6	0.1052	0.3236	0.3236	0.0827
7	0.0885	0.2891	0.2891	0.0701
8	0.0751	0.2611	0.2611	0.0607
9	0.0643	0.2385	0.2385	0.0536
10	0.0555	0.2201	0.2201	0.0481
11	0.0485	0.2052	0.2052	0.0437
12	0.0428	0.1930	0.1930	0.0403
13	0.0382	0.1830	0.1830	0.0376
14	0.0345	0.1748	0.1748	0.0353

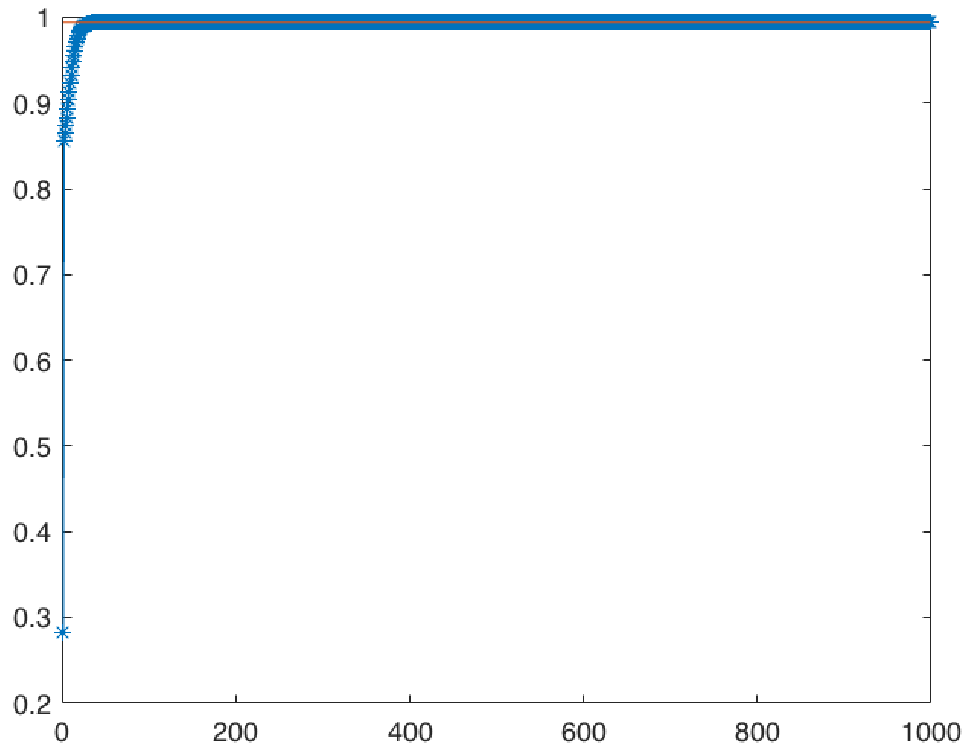
⋮

```
ans = 1×3 table
```

	# iteracions	Primer Residu	Darrer Residu
1	1000	0.2722	8.2706e-05

```
%És interessant estudiar com evoluciona: delta^k/delta^k-1
ratio = delta(2:end) ./ delta(1:end-1);
plot(ratio,'-*')
hold on
```

```
plot(rhoGS * ones(size(ratio)))
hold off
```




```
% Càlculs per fer n = 4
tic
D = diag(diag(a4));
d = diag(1 ./ diag(a4));

L = tril(a4,-1);
U = triu(a4,1);
dG = inv(L+D);
Bgs = -dG*U; %matriu iteració
cgs = dG*b4'; %vector

rhoGS = max(abs(eig(Bgs))); %radi espectral del mètode de Gauss-Seidel

if (rhoGS + eps >= 1) resp = "mètode de GAUSS-SEIDEL divergent";
else resp = "mètode de GAUSS-SEIDEL convergent";
end

gauss = {rhoGS, resp}';
%table(gauss, 'VariableNames',{'Gauss-Seidel'})

%mètode de Gauss
```

```
clear residu criteriParada e delta
if rhoGS >=1
    error('mètode de gauss-seidel divergent')
else
    n = 1000; i = 0;
    x = zeros(size(b4'));
    tol = 0.5e-8;
    criteriParada(1) = 1;
    while(i<n && criteriParada(max(i,1)) > tol)
        i = i+1; y = Bgs*x+cgs;
        residu(i) = norm(b4'-a4*y,'inf');
        e(i) = norm(y-x,'inf');
        delta(i) = norm(y-x,'inf');
        criteriParada(i) = delta(i)/norm(y,'inf');
        x = y;
    end
    tempsGaussSeidel4 = toc
    table(residu',e',delta',criteriParada','VariableNames',{'residu','e^k','delta^k','
    table(i, residu(1), residu(end), 'VariableNames',{'# iteracions', 'Primer Residu',
    x(1);
end
```

```
tempsGaussSeidel4 = 0.0426
ans = 1000×4 table
```

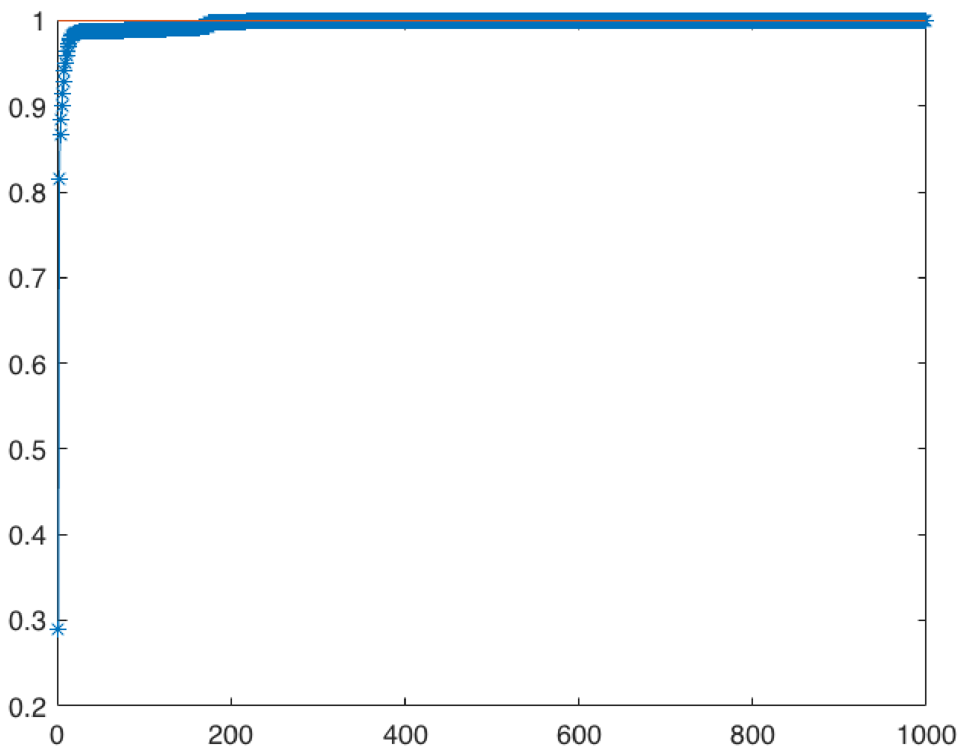
	residu	e^k	delta^k	stop
1	0.2887	2	2	1
2	0.2246	0.5775	0.5775	0.2240
3	0.1768	0.4712	0.4712	0.1557
4	0.1412	0.4083	0.4083	0.1208
5	0.1146	0.3608	0.3608	0.0985
6	0.0948	0.3248	0.3248	0.0835
7	0.0799	0.2973	0.2973	0.0729
8	0.0688	0.2762	0.2762	0.0651
9	0.0604	0.2598	0.2598	0.0593
10	0.0540	0.2469	0.2469	0.0549
11	0.0491	0.2367	0.2367	0.0514
12	0.0454	0.2283	0.2283	0.0485
13	0.0425	0.2215	0.2215	0.0462
14	0.0402	0.2157	0.2157	0.0442

⋮

```
ans = 1×3 table
```

	# iteracions	Primer Residu	Darrer Residu
1	1000	0.2887	0.0028

```
%És interessant estudiar com evoluciona: delta^k/delta^k-1
ratio = delta(2:end) ./ delta(1:end-1);
plot(ratio,'-*')
hold on
plot(rhoGS * ones(size(ratio)))
hold off
```




```
% Càlculs per fer n = 10
tic
D = diag(diag(a10));
d = diag(1 ./ diag(a10));

L = tril(a10,-1);
U = triu(a10,1);
dG = inv(L+D);
Bgs = -dG*U; %matriu iteració
cgs = dG*b10'; %vector

rhoGS = max(abs(eig(Bgs))); %radi espectral del mètode de Gauss-Seidel

if (rhoGS + eps >= 1) resp = "mètode de GAUSS-SEIDEL divergent";
else resp = "mètode de GAUSS-SEIDEL convergent";
end

gauss = {rhoGS, resp}';
```

```
%table(gauss, 'VariableNames',{'Gauss-Seidel'});

%mètode de Gauss
clear residu criteriParada e delta
if rhoGS >=1
    error('mètode de gauss-seidel divergent')
else
    n = 1000; i = 0;
    x = zeros(size(b10'));
    tol = 0.5e-8;
    criteriParada(1) = 1;
    while(i<n && criteriParada(max(i,1)) > tol)
        i = i+1; y = Bgs*x+cgs;
        residu(i) = norm(b10'-a10*y,'inf');
        e(i) = norm(y-x,'inf');
        delta(i) = norm(y-x,'inf');
        criteriParada(i) = delta(i)/norm(y,'inf');
        x = y;
    end
    tempsGaussSeidel10 = toc
    table(residu',e',delta',criteriParada','VariableNames',{'residu','e^k','delta^k','s
    table(i, residu(1), residu(end), 'VariableNames',{'# iteracions', 'Primer Residu',
    x(1);
end
```

```
tempsGaussSeidel10 = 0.0426
ans = 1000×4 table
```

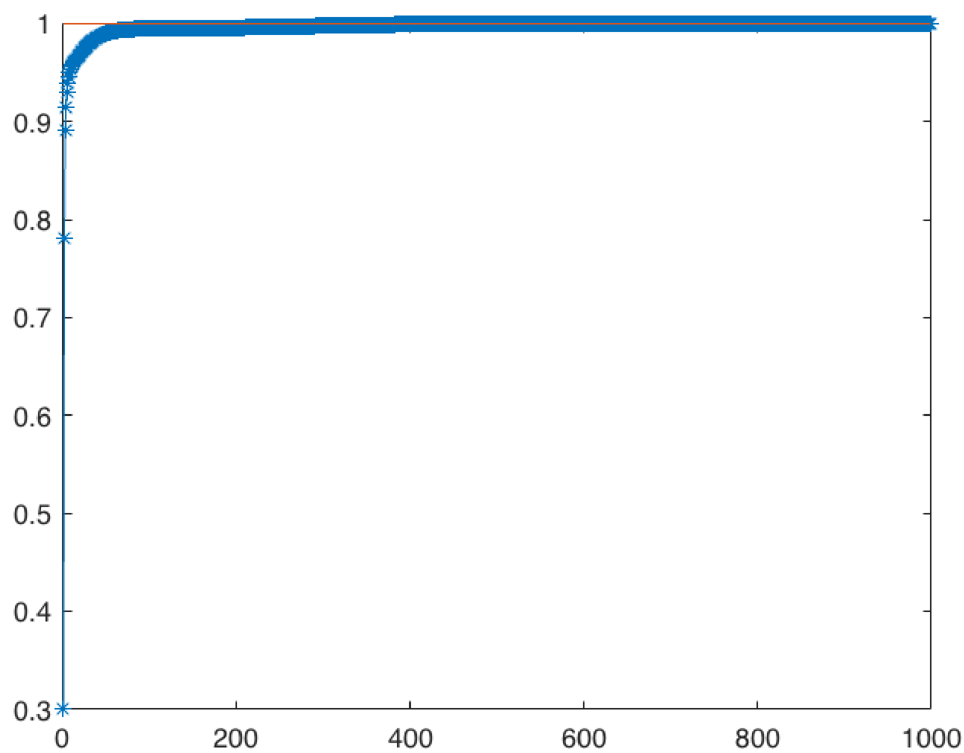
	residu	e^k	delta^k	stop
1	0.3004	2	2	1
2	0.2136	0.6008	0.6008	0.2310
3	0.1622	0.4690	0.4690	0.1549
4	0.1309	0.4182	0.4182	0.1247
5	0.1109	0.3823	0.3823	0.1058
6	0.0975	0.3553	0.3553	0.0926
7	0.0879	0.3338	0.3338	0.0828
8	0.0806	0.3157	0.3157	0.0750
9	0.0748	0.3001	0.3001	0.0687
10	0.0699	0.2861	0.2861	0.0633
11	0.0656	0.2735	0.2735	0.0587
12	0.0617	0.2619	0.2619	0.0547
13	0.0582	0.2512	0.2512	0.0511
14	0.0551	0.2414	0.2414	0.0473

⋮

```
ans = 1×3 table
```

	# iteracions	Primer Residu	Darrer Residu
1	1000	0.3004	0.0031

```
%És interessant estudiar com evoluciona: delta^k/delta^k-1
ratio = delta(2:end) ./ delta(1:end-1);
plot(ratio,'-*')
hold on
plot(rhoGS * ones(size(ratio)))
hold off
```




```
tempsGaussSeidel = [tempsGaussSeidel3;tempsGaussSeidel4;tempsGaussSeidel10];
```

```
% Càlculs per fer n = 15 //no funciona, el mètode deixa de ser convergent
% i passa a ser divergent, per tant tenim un problema quan augmentem el
% nombre de variables en el nostre sistema
```

e) Consulteu la documentació de Matlab®: Measure the Performance of Your Code, i Techniques to Improve Performance. Afegiu el còmput del temps en els tres tòpics treballats. Presenteu els resultats en taules.

```
table(tempsGauss,tempsSolve,tempsGaussSeidel, 'VariableNames',{'Temps Mètode Gauss','T
```

```
ans = 3×3 table
```

	Temps Mètode Gauss	Temps Mètode Solve	Temps Mètode Gauss-Seidel
1 n=3	0.0180	0.0041	0.2580

	Temps Mètode Gauss	Temps Mètode Solve	Temps Mètode Gauss-Seidel
2 n=4	0.0080	0.0036	0.0426
3 n=10	0.0151	0.0085	0.0426

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

Tal i com podem veure a la taula anterior, hem pogut notar que pel mètode de linsolve que implementa MATLAB el temps d'execució a mesura que n es fa gran es manté constant al llarg del temps tot i augmentar la mida de la matriu. Ja que el temps que triguen aquests algorismes implementats al programari de MATLAB és molt menor que si el implementem nosaltres, a causa del suport de Vector Processing (processament paral·lel) quan MATLAB tracta amb les matrius.

D'altra banda, també hem pogut veure que a mesura que augmentem la mida de la matiu, i.e el nombre de variables que entren en joc en el sistema d'equacions, el mètode de Gauss-Seidel, per n = 15, passa a ser divergent, per tant aquest mètode iteratiu pateix el problema de la convergència. I no és un mètode que ens interessi escollir alhora de resoldre el sistemes.

També hem pogut veure, que el mètode d'eliminació gaussiana, en mitjana aconseguix un millor temps que el mètode iteratiu de Gauss-Seidel. Ja que l'eficiència del mètode d'eliminació gaussiana depèn del nombre de càlculs implicats en la solució de l'equació lineal simultània. A mesura que augmenta el nombre de càlculs; l'eficiència del mètode d'eliminació gaussiana disminueix i viceversa a causa que l'algorisme ha trigat més temps en la substitució cap enrere.

Funció per resoldre sistemes lineals per eliminació gaussiana sense pivotament.

```
function x = ElimGaussSensePivot(A, b)
[n, n] = size(A);
[n, k] = size(b);
x = zeros(n,k);
for i = 1:n-1
    m = -A(i+1:n,i)/A(i,i);
    A(i+1:n,:) = A(i+1:n,:) + m*A(i,:);
    b(i+1:n,:) = b(i+1:n,:) + m*b(i,:);
end;
x(n,:) = b(n,:)/A(n,n);
for i = n-1:-1:1
    x(i,:) = (b(i,:) - A(i,i+1:n)*x(i+1:n,:))/A(i,i);
end
end
```