# Pràctica 2 – Enunciat B
# Computació Numèrica

**David Moreno Borràs**

**2017-18 Q2**

# 1 | Àlgebra lineal numèrica: mètodes iteratius

Siguin $A(N) = (a_{ij})_{N\times N}$ i $B(N) = (b_{i1})_{N\times 1}$ la matriu i el vector d'ordre $N$ definits per

$$a_{ij} = \begin{cases} -1 & |i-j| \leq 2\,, \\ 5 & i = j\,, \\ 0 & |i-j| > 2\,, \end{cases} \qquad \text{i} \qquad b_{i1} = \begin{cases} 3 & i = 1, N\,, \\ 2 & i = 2, N-1\,, \\ 1 & i \neq 1, 2, N-1, N\,, \end{cases}$$

$$\text{per a } i = 1, \ldots, N\,, \quad j = 1, \ldots, N\,.$$

Per a tots els ordres **N** tals que $6 \leq N \leq 30$ es demana:

(a) Calculeu el determinant i el nombre de condició de les matrius $A$.

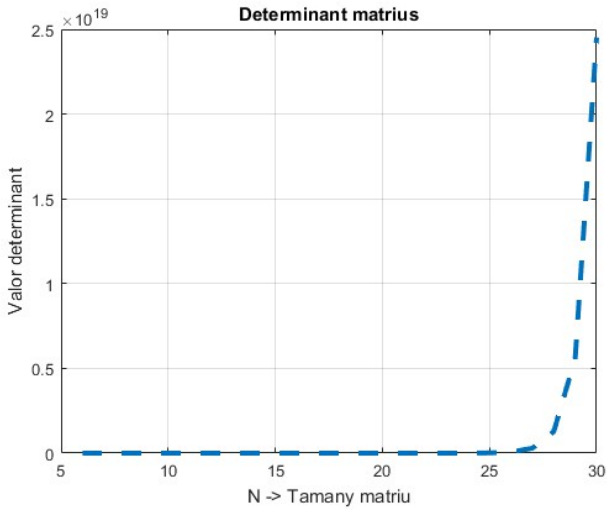
(b) Demostreu que $X = (1, 1, \ldots, 1)^t$ és solució exacte per a qualsevol $N$. (**No Matlab**)

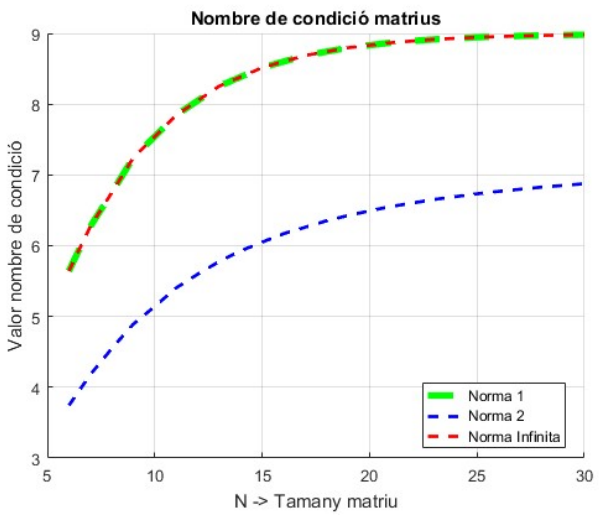
(c) Estudieu la convergència dels mètodes de Jacobi i Gauss-Seidel per a la resolució del sistema d'equacions lineals. Abans de calcular res, feu un gràfic d'evolució del radi espectral de la matriu d'iteració de cadascun dels mètodes estudiats en funció de **N**.

(d) Trobeu la solució $X$ del sistema $Ax = b$ per ambdós mètodes amb com a mínim 8 decimals correctes. Quantes iteracions calen en cada pas? Expliqueu els avantages i inconvenients dels mètodes per aquest cas concret, expliqueu les desviacions de la solució que s'obtenen.

a) Per començar s'inicialitzen la matriu A i el vector b amb la funció initMatriu.m (a l'annex) segons la definició de l'enunciat.

S'inicialitzen per valors entre 6 i 30 i el resultat dels seus determinants i nombres de condició és el de la següent taula i plots:



| N | Determinant | Nombre de condició |
|---|---|---|
| 6 | 9313 | 5,641791045 |
| 7 | 40920 | 6,270967742 |
| 8 | 179712 | 6,75 |
| 9 | 789096 | 7,242921013 |
| 10 | 3464545 | 7,541899441 |
| 11 | 15210629 | 7,844459621 |
| 12 | 66779280 | 8,049087591 |
| 13 | 293179500 | 8,254736842 |
| 14 | 1287135720 | 8,385137386 |
| 15 | 5650860869 | 8,515809873 |
| 16 | 24808738945 | 8,601425277 |
| 17 | 1,08917E+11 | 8,687115371 |
| 18 | 4,78173E+11 | 8,742216687 |
| 19 | 2,0993E+12 | 8,797338121 |
| 20 | 9,21647E+12 | 8,833131975 |
| 21 | 4,04627E+13 | 8,868931299 |
| 22 | 1,77641E+14 | 8,89205072 |
| 23 | 7,79892E+14 | 8,915171619 |
| 24 | 3,42393E+15 | 8,930146927 |
| 25 | 1,50319E+16 | 8,945122636 |
| 26 | 6,59939E+16 | 8,95480666 |
| 27 | 2,8973E+17 | 8,964490793 |
| 28 | 1,27199E+18 | 8,970758467 |
| 29 | 5,58437E+18 | 8,97702617 |
| 30 | 2,45168E+19 | 8,981080759 |

Al plot dels nombres de condició també he afegit el càlcul d'aquest utilitzant la norma 1, la 2 i la infinita per veure si hi ha alguna diferència. Com es pot apreciar entre la 1 i la infinita no hi ha diferència però la 2 és més petita.

Per guardar les dades simplement dins un for de 6 a 30 s'anaven generant les matrius i els resultats dels càlculs s'anaven guardant per després poder fer el plot. El codi complet està a l'annex.

b) Si $X = (1,1,....,1)^t$ és solució exacte per qualsevol N, això vol dir que totes les línies satisfan:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n &= b_1\,, \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n &= b_2\,, \\ \vdots & \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots a_{mn}x_n &= b_m\,. \end{aligned}$$

En efecte, per la pròpia definició de la matriu la suma de totes les files satisfan l'anterior:

Les **files 1 i n** tenen b = 3 i la suma dels elements de la fila de la matriu A sempre dona 3 ja que sempre hi ha un 5 a cada fila (diagonal de la matriu A) i només 2 posicions que compleixen |i-j| <= 2, per tant hi haurà dos -1: 5 -1 -1 = 3

En el cas de les **files 2 i n-1** hi ha una posició més que compleix |i-j| <= 2 per tant la resta és 2, com al vector b.

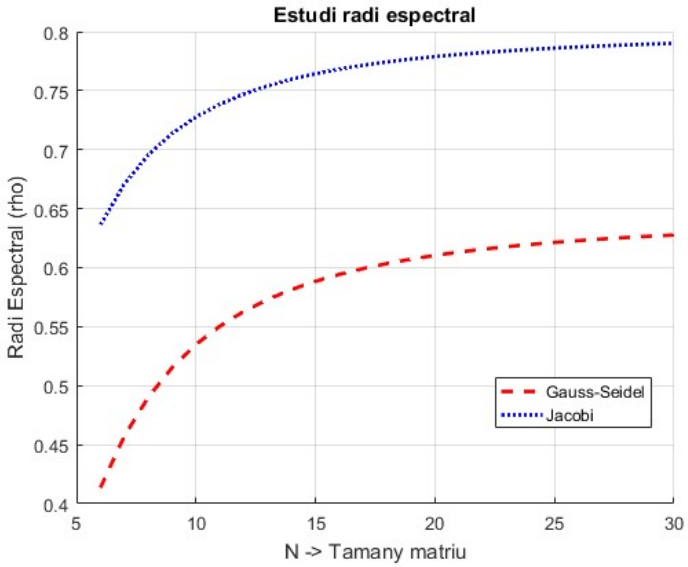
Finalment per la resta de files tenim un 5 i quatre -1s sempre, independentment de la mida de la matriu, per tant dona 1 la suma de tots els $a_i * x_i$.

c) Per estudiar la convergència hem de veure el radi espectral de la matriu B del sistema equivalent x = Bx + C. Els dos mètodes seran convergents si i només si el radi espectral és menor que 1.

En els dos casos calculem les matrius auxiliars necessàries (diagonal, lower triangular i upper triangular) i fem un for de 6 a 30 on calculem rho de B:

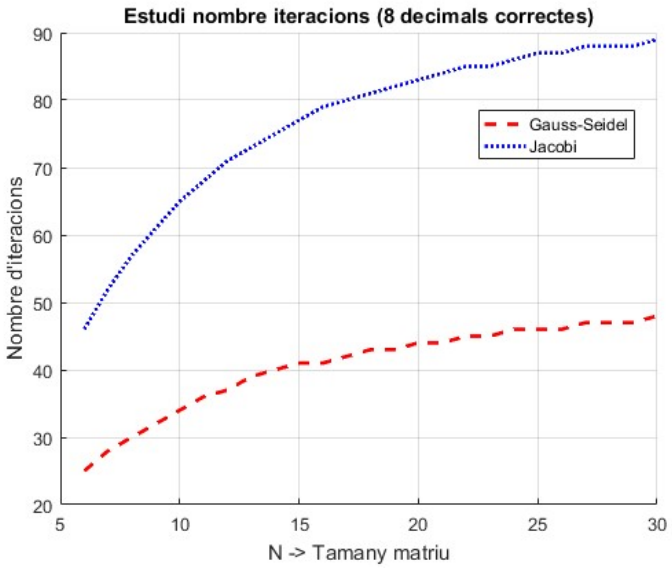
```
aux = inv(D);
Bj = -aux *(L+U);
rhoJ = max(abs(eig(Bj)));
```

Si aquest rho és menor que 1, és convergent. Guardem els resultats a una matriu i obtenim els següents gràfics on podem veure que els dos mètodes convergeixen sempre per valors d'N entre 6 i 30.



d) Per trobar la solució amb 8 decimals correctes agafem un valor de tolerància tol = 0.000000005. Fem servir la funció "itera.m" (als annexes) a la que li passem tol, el nombre màxim d'iteracions, i les matrius necessàries.

   Per cada valor d'N entre 6 i 30 es guarda a una taula el nombre d'iteracions que ha necessitat el mètode per obtenir un resultat amb 8 decimals correctes. Aquests són els resultats:




   Com podem veure, el nombre d'iteracions necessàries als dos mètodes augmenta amb la mida de la matriu i el mètode de Jacobi necessita moltes més que el de Gauss-Seidel ja que aquest té una velocitat de convergència (*-log(rho)*) major, com hem vist a l'anterior apartat.

   Al codi estan comentades les línies:

```
% filename = 'GaussSeidelIteracions.xlsx';
% xlswrite(filename,iteracionsGauss);
```

   Perquè al principi la representació de les dades l'havia fet amb excel però al final ho fet tot amb Matlab per així poder tenir els dos mètodes a un mateix plot i poder apreciar millor les diferències.

# Annex

## Codis Matlab Exercici 1: Àlgebra lineal numèrica: mètodes iteratius

## Estudi determinants i nombres de condició

```
%% Inicialització matrius
[A, b] = initMatriu(6);
detA = det(A);
taulaDet = (zeros(30-6,1));
taulaCond = (zeros(30-6,1));
taulaCond2 = (zeros(30-6,1));
taulaCondInf = (zeros(30-6,1));
i = 1;
taulaDet(i) = det(A);
taulaCond(i) = cond(A,1)
taulaCond2(i) = cond(A)
taulaCondInf(i) = cond(A,'inf')
i = i + 1;
for n=7:30
    [A, b] = initMatriu(n);
    taulaDet(i) = det(A);
    taulaCond(i) = cond(A,1);
    taulaCond2(i) = cond(A);
    taulaCondInf(i) = cond(A,'inf');
    i = i + 1;
end

%% Plot determinant

t = 6:30;
plot(t, taulaDet,'--','LineWidth',3), title('Determinant matrius')
ylabel('Valor determinant')
xlabel('N -> Tamany matriu')
grid('on')

%% Plot nombre condició
t = 6:30;
hold all,
plot(t, taulaCond,'--','Color','g','LineWidth',4,'DisplayName','Norma 1')
plot(t, taulaCond2,'--','Color','b','LineWidth',2,'DisplayName','Norma 2')
plot(t, taulaCondInf,'--','Color','r','LineWidth',2,'DisplayName','Norma
Infinita')
title('Nombre de condició matrius')
ylabel('Valor nombre de condició')
xlabel('N -> Tamany matriu')
legend('show','location','best'),
grid('on')
```

## Convergència

```
%% Estudi convergència Jacobi
totalConvJac = zeros(30-6,1);
i = 1;
for n=6:30
    [A, b] = initMatriu(n);
    D = diag(diag(A));
    L = tril(A,-1);
    U = triu(A,1);
    aux = inv(D);
    Bj = -aux *(L+U);
    rhoJ = max(abs(eig(Bj)));
    totalConvJac(i) = rhoJ;
    if(rhoJ < 1)
        %fprintf('Mètode de Jacobi Convergent\n ')
    else
        fprintf('Mètode de Jacobi divergent\n ')
    end
    i = i + 1;
end
% filename = 'JacobiEstudiConvergencia.xlsx';
% xlswrite(filename,totalConv);

%% Estudi convergència Gauss-Seidel
totalConvGauss = zeros(30-6,1);
i = 1;
for n=6:30
    [A, b] = initMatriu(n);
    D = diag(diag(A));
    L = tril(A,-1);
    U = triu(A,1);
    aux = inv(L+D);
    Bs =- aux*(U);
    rhoS = max(abs(eig(Bs)));
    totalConvGauss(i) = rhoS;
    if(rhoS < 1)
        %fprintf('Mètode de Gauss-Seidel Convergent\n ');
    else
        fprintf('Mètode de Gauss-Seidel divergent\n ');
    end
    i = i + 1;
end
% filename = 'GaussSeidelEstudiConvergencia.xlsx';
% xlswrite(filename,totalConv);

%% Plot comparació convergencies
t = 6:30;
hold all,
plot(t, totalConvGauss,'--','Color','r','LineWidth',2,'DisplayName','Gauss-
Seidel')
plot(t, totalConvJac,':','Color','b','LineWidth',2,'DisplayName','Jacobi')
title('Estudi radi espectral')
ylabel('Radi Espectral (rho)')
xlabel('N -> Tamany matriu')
legend('show','location','best'),
grid('on')
```

## Càlcul mètodes

```
%% Mètode de Jacobi
i = 1;
for n=6:30
    [A, b] = initMatriu(n);
    D = diag(diag(A));
    L = tril(A,-1);
    U = triu(A,1);
    aux = inv(D);
    Bj = -aux *(L+U);
    cj = aux * b;
    rhoJ = max(abs(eig(Bj)));
    if(rhoJ < 1)
        %fprintf('Mètode de Jacobi Convergent\n ');
        kMax = 100; tol = 0.000000005;
        [ xJ, residuJ, it ] = itera( A, b, Bj, cj, kMax, tol);
        iteracionsJacobi(i) = it;
    else
        fprintf('Mètode de Jacobi divergent\n ');
    end
    i = i + 1;
end

%% Mètode de Gauss-Seidel
i = 1;
for n=6:30
    [A, b] = initMatriu(n);
    D = diag(diag(A));
    L = tril(A,-1);
    U = triu(A,1);
    aux=inv(L+D);
    Bs=-aux*(U);
    cs = aux*b;
    rhoS = max(abs(eig(Bs)));
    if(rhoS < 1)
        %fprintf('Mètode de Gauss-Seidel Convergent\n ');
        kMax = 100; tol = 0.000000005;
        [ xS, residuS, it ] = itera( A, b, Bs, cs, kMax, tol);
        iteracionsGauss(i) = it;
    else
        fprintf('Mètode de Gauss-Seidel divergent\n ');
    end
    i = i + 1;
end

%% Plot comparació iteracions
t = 6:30;
hold all,
plot(t, iteracionsGauss,'--','Color','r','LineWidth',2,'DisplayName','Gauss-
Seidel')
plot(t,
iteracionsJacobi,':','Color','b','LineWidth',2,'DisplayName','Jacobi')
title('Estudi nombre iteracions (8 decimals correctes)')
ylabel('Nombre d''iteracions')
xlabel('N -> Tamany matriu')
legend('show','location','best'),
grid('on')
```

# 2 | Àlgebra lineal numèrica: valors propis

Fent ús del mètode de la potència calculeu el valor propi de mòdul més gran i el valor propi de mòdul més petit de la matriu $A$ definida per:

$$A = \begin{pmatrix} -3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 \end{pmatrix}.$$

(a) Cerca documentació sobre *les matrius de Wilkinson*. Escriu un breu resum del que has entès (màxim 1/2 full). Dóna les teves fonts bibliogràques.

(b) Apliqueu el mètode de la potència per trobar el valor propi de mòdul més gran de la matriu $A$, fent ús de l'aritmètica de coma flotant de `Matlab` amb $tol_{min} = 10^{-8}$.

(c) Apliqueu el mètode de la potència per trobar el valor propi de mòdul més petit de la matriu $A$, fent ús de l'aritmètica de coma flotant de `Matlab` amb $tol_{min} = 10^{-8}$.

a) Les matrius de Wilkinson (anomenades així pel matemàtic angles James H. Wilkinson) són unes matrius simètriques, tridiagonals d'ordre N amb parells de valors propis casi iguals.

Hi ha dos tipus, les negatives $W_n^-$ (com la de l'enunciat de l'exercici) i les positives $W_n^+$, que son iguals però amb tots el nombres positius.

Aquestes matrius tenen una sèrie de propietats:

- A $W_n^-$, tots els valors propis de la matriu, excepte el del mig, apareixen en parelles.
- La matriu $W_n^-$ és singular, és a dir, no té matriu inversa
- Tots els valors propis de $W_n^+$excepte un, són positius.
- Aques valors propis de $W_n^+$ apareixen en parelles, amb un valor propi positiu de $W_n^-$ al centre de cada parella.
- Les parelles de valors propis més grans de $W_n^+$ són molt properes entre elles.

Bibliografia:


- The Algebraic Eigenvalue Problem by J. H. WILKINSON
- https://blogs.mathworks.com/cleve/2013/04/15/wilkinsons-matrices-2/

b) Debut a la matriu, que com hem vist té una sèries de propietats especials, tot i que el mètode de la potència convergeix a un valor que segons el càlcul dels valors propis de Matlab és correcte. Si fem:

```
[Wm,Wp] = generate_wilkinson_matrices(3);
disp('       eig(Wm(21))')
disp(flipud(eig(Wm)))
```

com fa Moler a [blogs.mathworks.com/cleve/2013/04/15/wilkinsons-matrices-2/](blogs.mathworks.com/cleve/2013/04/15/wilkinsons-matrices-2/), el resultat és el següent:

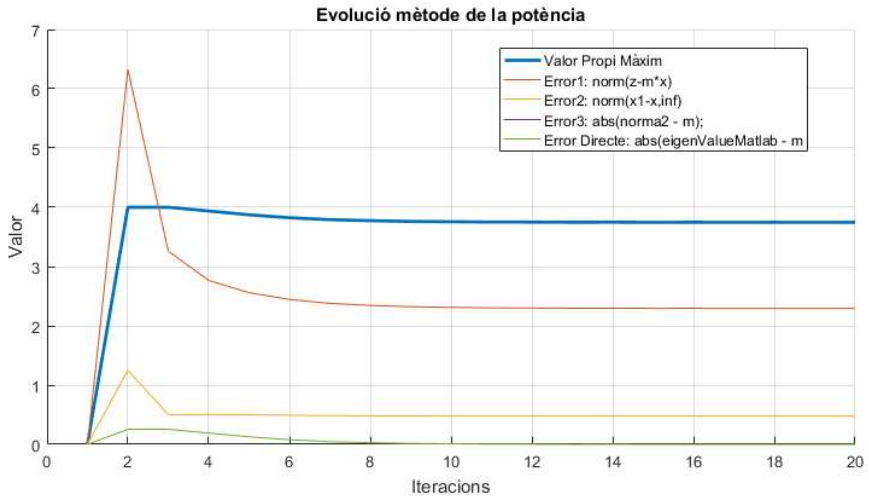
| eig(Wm(21)) |
|---|
| 3.746194162881017 |
| 2.210673621807088 |
| 1.038725869439356 |
| -0.000000000000000 |
| -1.038725869439356 |
| -2.210673621807087 |
| -3.746194162881015 |

Ara veurem que el mètode de la potència convergeix al valor propi de mòdul més gran, 3.746194162881017, però arriba on punt on no "avança" més i no podem garantir que el resultat es doni amb 8 decimals correctes.

Per començar la generem amb una funció anomenada generate_wilkinson_matrices que es troba a l'annex que genera la matriu de Wilkinson segons una entrada n.

Pel mètode de la potència, comencem amb un vector arbitrari no nul, en aquest cas, per exemple: x0 = [0;1;1;1;1;1;1]

Aquí podem veure un plot amb l'evolució del mètode i diferents formes de calcular l'error:



Com podem veure a partir de l'iteració 5 el mètode no varia el resultat (**3.747129750461244**) i les úniques formes de calcular l'error que s'aproximen a 0 són la directa (comparant-lo amb el valor que ens dona Matlab) i amb la norma2.

Com la matriu és simètrica i sabem que una matriu simètrica compleix que la seva norma 2 és igual al seu radi espectral (màxim dels valors propis de la matriu) podem fer servir això també per aproximar el valor.

L'error 2, calculat amb la norma infinit de la diferència entre iteracions tampoc dona bons resultats.

Provant amb diferents vectors inicials, el que millors resultats donava era el mencionat abans, amb el qual s'obté un valor propi màxim = 3.747129750461244 i un vector propi:

0.238344954633258
-0.174120845436878
0.090877748752864
0.065473058971411
0.307464374792267
0.745334864643855
1.000000000000000

c) Com sabem, per calcular el valor propi d'una matriu de mòdul més petit, hem de fer el mateix que amb el de mòdul més gran, però usant la matriu inversa.

   Com hem vist abans, però, La matriu $W_n^-$ és singular, és a dir, no té matriu inversa, per tant, no podem trobar-lo fent servir el mètode de la potència. El propi Matlab ens avisa si ho intentem fer:

   Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.
   RCOND = 1.734723e-18.

# Annex

## Codis Matlab Exercici 2: Àlgebra lineal numèrica: valors propis

### Funció per generar les matrius:

```
function [Wm,Wp] = generate_wilkinson_matrices(n)
   D = diag(ones(2*n,1),1);
   Wm = diag(-n:n) + D + D';
   Wp = abs(diag(-n:n)) + D + D';
end
```

### Mètode de la potència:

```
[B, pos] = generate_wilkinson_matrices(3);
eigsMatlab = flipud(eig(B));
eigenValueMatlab = max(eigsMatlab);
A = B;
x = [0;1;1;1;1;1;1];
norma2 = norm(A,2); % Com A es simetrica, la norma 2 es igual al radi
espectral, es a dir, el maxim dels valors propis de la matriu, que es el que
busquem
while (iter < 20 && errorDirecte > 0.000005)
    z = A*x;
    m2 = (z'*x)./(x'*x);
    m = norm(z,'inf');
    x1 = z/m;
    error = norm(z-m*x);
    error2 = norm(x1-x,'inf');
    error3 = abs(norma2 - m);
    errorDirecte = abs(eigenValueMatlab - m);
    iter = iter + 1;
    x = x1;
    taula1(iter,:)=[iter, m, error,error2,error3,m2,errorDirecte];
end
vap_max = m;
vep = x;

%% Display results
t = 1:iter; hold all
plot(t, taula1(:,2),'LineWidth', 2, 'DisplayName', 'Valor Propi Màxim')
plot(t, taula1(:,3),'DisplayName', 'Error1: norm(z-m*x)')
plot(t, taula1(:,4),'DisplayName', 'Error2: norm(x1-x,inf)')
plot(t, taula1(:,5),'DisplayName', 'Error3: abs(norma2 - m);')
plot(t, taula1(:,7),'DisplayName', 'Error Directe: abs(eigenValueMatlab - m')
xlabel('Iteracions')
ylabel('Valor')
legend('show','location','best'), title('Evolució mètode de la potència'),
grid('on')

%% Eigenvalues matlab
%https://blogs.mathworks.com/cleve/2013/04/15/wilkinsons-matrices-2/
format long
[Wm,Wp] = generate_wilkinson_matrices(3);
disp('       eig(Wm(21))')
disp(flipud(eig(Wm)))
```

# 3 | Integració numèrica: Àrea dins una regió tancada

Doneu una aproximació de l'àrea de la regió delimitada per la vostra ma.

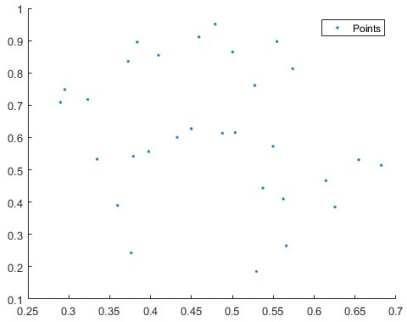
Referència: http://www.mathworks.es/moler/chapters.html

(a) Obteniu una imatge de la vostra ma. Seguiu les indicacions de l'exercici 3.4 de la pàgina 20 del capítol "Interpolation"de Cleve Moler.

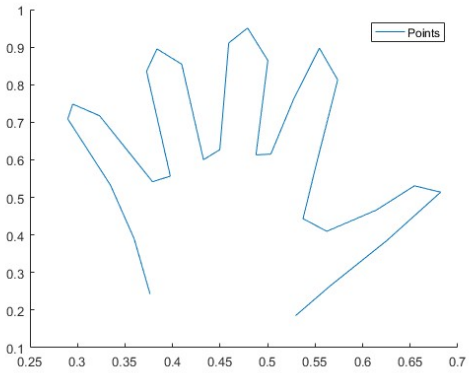
(b) Obteniu per interpolació la corba que delimita la imatge de la ma seguint els indicacions de l'exercici 3.4 i l'exercici 3.5 de les pàgines 20 – 21 – 22. Responeu les preguntes que us formulen en els dos exercicis.

(c) Què tan gran és la teva mà? Calcula l'àrea que ocupa la teva mà. Segueix les indicacions i respon les preguntes de l'exercici 6.23 de les pàgines 19 – 20 del capítol "Quadrature"de Cleve Moler.

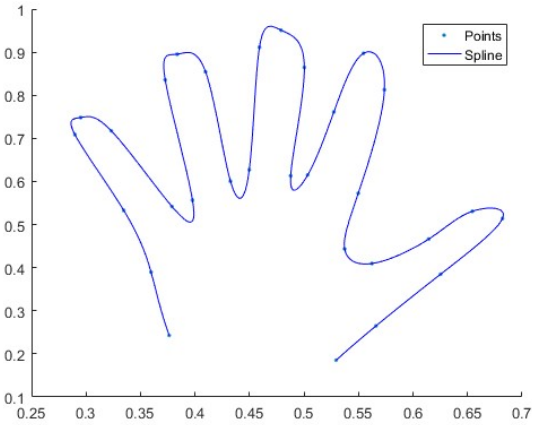
a) Seguim les instruccions de l'exercici per guardar les dades de la mà i guardem a un fitxer les variables amb save('hand.mat') per no haver de registrar la mà cada vegada:

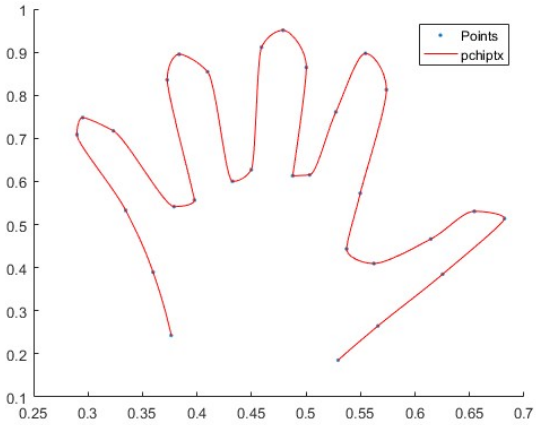



b) A aquesta imatge podem veure un plot dels valors x, y enregistrats:



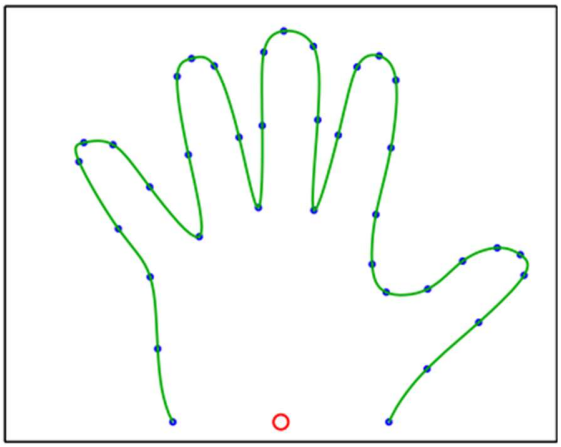
Es pot apreciar bastant la forma d'una mà una vegada pintem les línies entre els punts però s'ha d'interpolar per veure un millor dibuix.




**Interpolació amb Spline** **Interpolació amb polinomi cúbic**

Respecte la pregunta de Moler sobre amb quin mètode s'ha dibuixat la seva mà:



Veient els resultats de la meva mà, diria que ha estat amb Spline, que fa unes corbes molt més "suaus" i uns canvis menys bruscos que el polinomi cúbic

c) Per mesurar la mida de la mà, Moler proposa 3 mètodes diferents per calcular-la.

Primer de tot s'han de fer uns ajustaments. L'àrea ha de quedar tancada, així que bàsicament a la matriu amb les dades de la mà poso un últim punt idèntic al punt inicial.

I després si els punts s'han guardat en sentit "horari", el resultat queda negatiu, així que a la solució de cada un dels mètodes li aplico abs() per obtenir l'àrea.

## A) Àrea del polígon

Tenint un polígon amb n vèrtexs *(xi, yi)*, l'àrea del polígon és:

$$(x1y2 - x2y1 + x2y3 - x3y2 + \cdots + xny1 - x1yn)/2$$

Això ho podem calcular a Matlab així:

```
total = 0;
for i = 1:length(x)-1
    total = total + x(i)*y(i+1);
    total = total - x(i+1)*y(i);
end
areaMaA = abs(total/2)
```

I dona com a resultat **0.1259**

## B) Quadratura simple

A aquest mètode fem server *inpolygon*, que com el seu nom indica determina si un conjunt de punts es troba dins una regió poligonal de l'espai (en aquest cas una graella amb mida h. El resultat d'aquesta funció posa a 1 els punts que són dins la regió i a 0 els que no, per tant l'àrea ve donada per *nnz* de k, el número de punts que no són 0:

```
xmin = min(x(:));
ymin = min(y(:));
xmax = max(x(:));
ymax = max(y(:));
[u,v] = meshgrid(xmin:h:xmax,ymin:h:ymax);
k = inpolygon(u,v,x,y);
areaMaB(i) = abs(h^2*nnz(k));
```

Amb un valor d'h = 0.005 obtenim com a resultat **0.1265.**

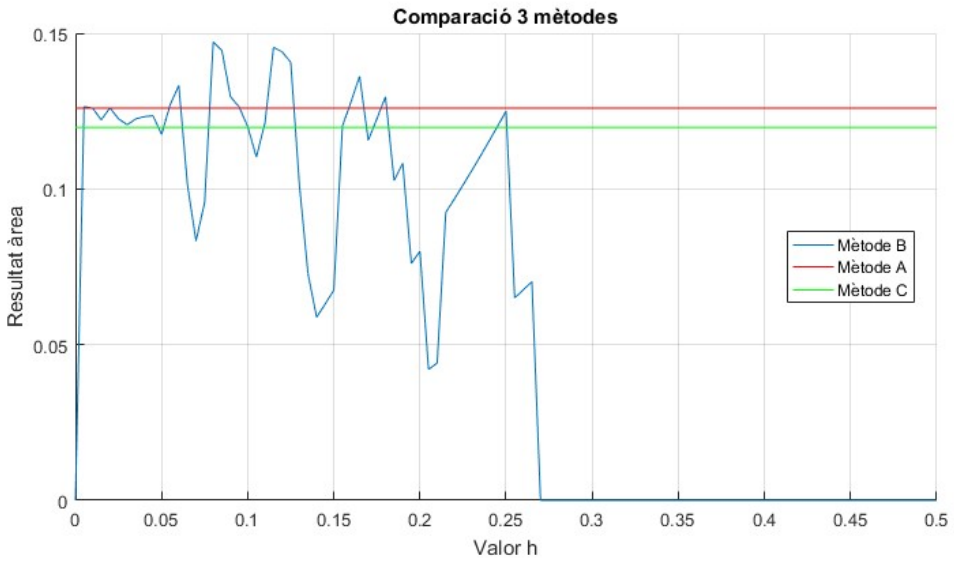
El resultat varia molt, però, segons aquesta h, per això al final de l'apartat es veu a un plot com canvia per valors entre 0 i 0.5, comparant-lo amb els altres mètodes.

## C) Quadratura adaptativa (2D)

Per aquest últim mètode fem servir la funció inpolygon de nou però amb l'ajuda de la funció *chi,* que fa el mateix però considerant si els paràmetres d'entrada u i v són escalars o arrays:

```
function k = chi(u,v,x,y)
    if all(size(u) == 1), u = u(ones(size(v))); end
    if all(size(v) == 1), v = v(ones(size(u))); end
    k = inpolygon(u,v,x,y);
end
```

Aquí podem veure el resultat dels tres mètodes:




Com podem veure A i C donen resultats semblants, però el problema que té C és que és el més costós de tots tres. El B, tot i que alguns valors s'assemblen als resultats d'A i C, presenta moltes irregularitats i varia massa segons el paràmetre h.

# Annex

## Codis Matlab Exercici 3: Integració numèrica: Àrea dins una regió tancada

```
%% Read data
figure('position',get(0,'screensize'))
axes('position',[0 0 1 1])
[x,y] = ginput;
filename = 'handClosed.mat';
save(filename);

%%
clear, clc, load('handClosed.mat');
n = length(x);
s = (1:n)';
t = (1:.05:n)';
u1 = spline(s,x,t);
v1 = spline(s,y,t);
u2 = pchip(s,x,t);
v2 = pchip(s,y,t);
clf reset
hold all,
plot(x,y,'.','DisplayName', 'Points')
plot(u1,v1,'-','Color','g','DisplayName', 'Spline')
%plot(u2,v2,'-','Color','r','DisplayName', 'pchiptx')
legend('show')


%% Calcul area ma
%% Opció A -> àrea del poligon
total = 0;
for i = 1:length(x)-1
    total = total + x(i)*y(i+1);
    total = total - x(i+1)*y(i);
end
areaMaA = abs(total/2)

% en una linea: areaMaA = (x'*y([2:n 1]) - x([2:n 1])'*y)/2

%% Opció B -> Quadratura simple
i = 1;
%for h = 0:0.005:0.5
h = 0.005;
    xmin = min(x(:));
    ymin = min(y(:));
    xmax = max(x(:));
    ymax = max(y(:));
    [u,v] = meshgrid(xmin:h:xmax,ymin:h:ymax);
    k = inpolygon(u,v,x,y);
    areaMaB(i) = abs(h^2*nnz(k))
    i = i + 1;
%end

%% Opció C -> Quadratura adaptativa (2D)
tol = 0.005;
areaMaC = abs(dblquad(@(u,v)chi(u,v,x,y),xmin,xmax,ymin,ymax,tol))
k = chi(u,v,x,y);
```

```
%% Comparació 3 mètodes
hold all
t = 0:0.005:0.5;
plot(t, areaMaB,'DisplayName','Mètode B')
xlabel('Valor h')
ylabel('Resultat àrea')
plot(t, areaMaA*ones(size(t)), 'Color', 'r','DisplayName','Mètode A')
plot(t, areaMaC*ones(size(t)), 'Color', 'g','DisplayName','Mètode C')
title('Comparació 3 mètodes')
legend('show','location','best'),
grid('on')
```

# 4 | Aproximació de dades

Les dades de la taula següent estan relacionats amb l'esperança de vida al nèixer dels ciutadants de dos països

| *any* | 1975 | 1980 | 1985 | 1990 | 1995 | 2000 | 2005 | 2010 |
|---|---|---|---|---|---|---|---|---|
| *Belice* | 67.7 | 69.6 | 71.1 | 69.6 | 68.4 | 69.0 | 69.8 | 70.3 |
| *Espanya* | 73.3 | 75.3 | 76.2 | 76.8 | 78.0 | 79.0 | 80.2 | 81.6 |

Es demana:

(a) Useu el polinomi interpolador de grau 7 per estimar l'esperança de vida el 1970, 1992, 2007 per cada pais. Compareu els valors obtinguts, amb les xifres oficials per cada pais, que són:

| *any* | 1970 | 1992 | 2007 | 2015 |
|---|---|---|---|---|
| *Belize* | 65.5 | 70.7 | 69.5 | 70.3 |
| *Espanya* | 72.0 | 77.4 | 80.9 | 83.4 |

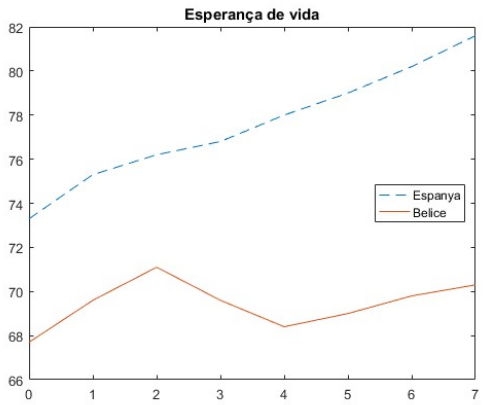
(b) Busqueu un polinomi del grau escaient per mínims quadrats. Justifiqueu l'elecció mostrant una cota de l'error d'aquest i de la resta amb els que heu provat. Compareu els valors obtinguts, amb les xifres oficials per cada pais.

(c) Extrapoleu un valor per l'any 2015 pels dos models obtinguts per cada pais. Recordant les dades oficials, i els resultats obtinguts els models estudiats són vàlids per estimar amb precissió l'esperança de vida per l'any 2015?

(d) Feu una gràfica on apareguin les dades (representats per una rodona) i les totes solucions trobades per pais.

**Comentari:** Numèricament és millor que considereu la taula inicial amb abscisses $0, 1, \cdots, 7$.

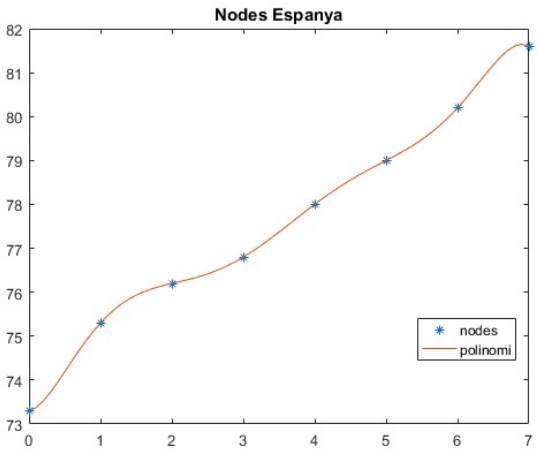
a) Per començar guardem les dades a Matlab. Aquí podem veure el plot de la progressió de l'esperança de vida dels dos països:



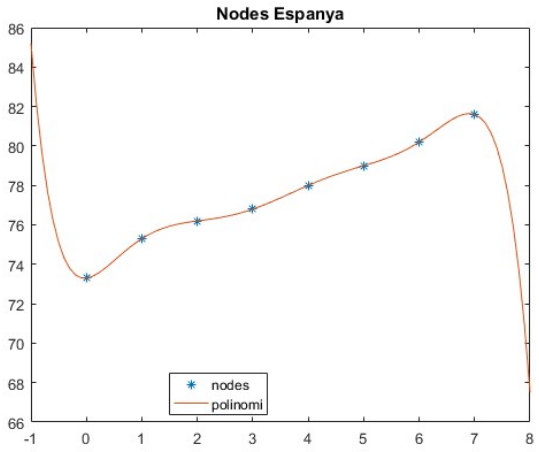
Com es comenta a l'enunciat, s'ha considerat en tot moment la taula d'abscisses 0:7 enlloc dels anys, ja que aquesta donava problemes.

Una vegada tenim les dades, procedim a buscar el polinomi interpolador.

Per obtenir el polinomi de grau 7 hem de cridar la funció polyfit. El resultat transposat són els coeficients. Els quals els fem servir per obtenir el polyval. El resultat, si fem un plot de 0 a 7, és la imatge de l'esquerra:




**Polinomi Interpolador**




**Fenomen de Runge**

Com podem observar si fem el plot amb un interval més ampli, la interpolació resultat oscil·la molt als extrems de l'interval, així que podem observar el Fenomen de Runge.

Per obtenir els valors que demana l'enunciat: 1970, 1992 i 2007 hem de veure el valor que pren el polinomi en aquests punts, però com hem comentat abans la taula d'abscisses no són els anys, és 0:7.

L'equivalència de "passos" és *1 pas = 5 anys*, per tant 1970 correspondria a -1, tenint en compte que l'inicial es 0. Com està "fora" hem d'ampliar el rang: t = -2:0.1:9 i tenint en compte que ara cada pas són 10 elements de la matriu i a Matlab es comença a indexar a 1, podem veure que v(21) = 73.3000, el primer valor, per tant, **v(11) = 85.200** equival a l'estimació per

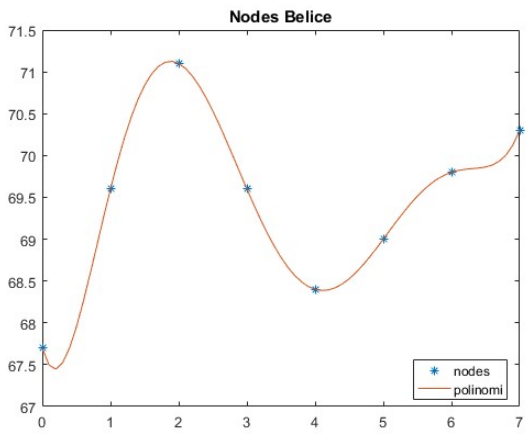
l'any 1970. De fet, per comprovar, podem fer "data cursor" al plot i veure que efectivament el valor corresponent a -1 és **85,2.**

Per l'any 2007 tenim que v(91) = 81.0, l'any 2010. I v(81) = 80.2, lany 2005, per tant, com entre 5 anys hi ha 10 indexs, cada any són 2 indexs. Per tant, si volem l'any 2007 hem de buscar v(81 + 2 + 2) = **v(85) = 80.9803**.

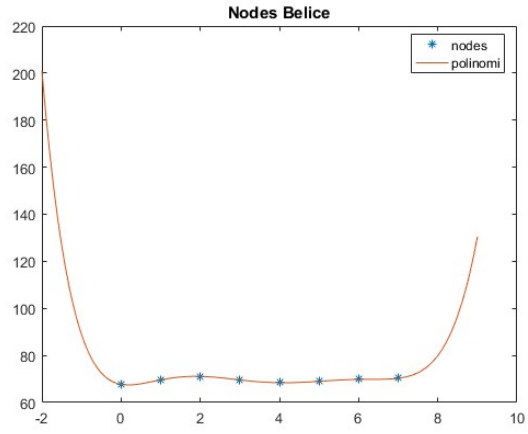
Finalment, per l'any 1992 hem d'observar quin index correspon a 1990 i sumar 4, com abans. Li correspon v(51), per tant l'estimació per 1992 és v(51 + 4) = **v(55) = 77.2448.**

Com podem observar, els valors de l'any 2007 i 1992, al estar dins l'interval, tenen una estimació força aproximada, concretament l'error absolut en el cas de l'any 2007 és **0.083** i l'any 1992 és **0.1552**.L'any 1970, al estar fora i degut al fenomen de Runge té un error absolut de **13.2000**.

Ara fem el mateix amb Belice:




**Polinomi Interpolador**




**Fenomen de Runge**

De nou podem observar el fenomen de Runge quan estudiem el polinomi fora de l'interval.

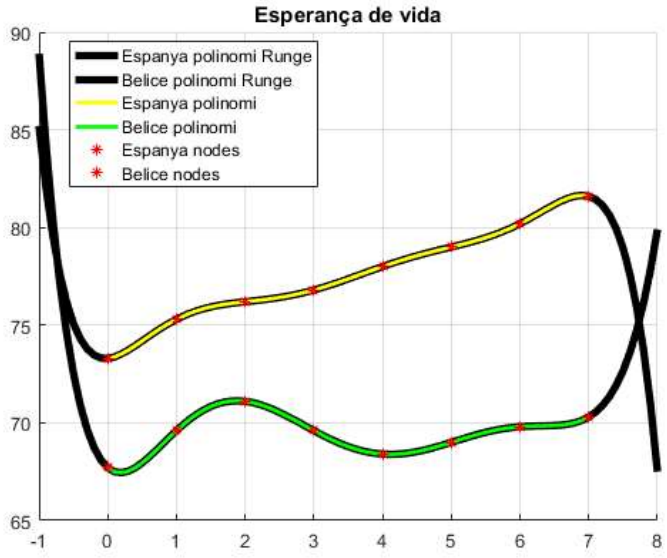
Seguint els mateixos passos que amb Espanya, obtenim que l'any 1970 té una estimació de v(11) = 88.90, l'any 2007 v(85) = 69.8506 i l'any 1992 v(55) = 68.9179

Si calculem els errors absoluts obtenim:

1970 -> 23.40
1992 -> 1.7821
2007 -> 0.3506

Així que de nou veiem que els dos que estan dins dels intervals fan estimacions bastants correctes però la de 1970 és molt diferent.

La següent imatge és un plot amb tots els elements vists:




Els codis usats es troben a l'annex.

b) Ara estudiarem polinomis de graus diferents per mínims quadrats (amb polyfit), veurem els errors del valors aproximats per determinar quin polinomi funciona millor amb aquest cas.

Amb Espanya comencem fent l'estimació com a l'apartat anterior però amb els diferents graus dins un for (codi als annexes).

Resultats dels valors estimats per cada any depenent del grau:

| Grau | 1970 | 1992 | 2007 | 2015 |
|---|---|---|---|---|
| 1 | 72,61 | 77,44 | 80,73 | 82.48 |
| 2 | 72,63 | 77,43 | 80,74 | 82.50 |
| 3 | 71,33 | 77,46 | 80,67 | 83.80 |
| 4 | 69,79 | 77,29 | 80,84 | 83.80 |
| 5 | 67,62 | 77,27 | 80,68 | 84.42 |
| 6 | 67,95 | 77,27 | 80,67 | 84.75 |
| 7 | 85,20 | 77,24 | 80,98 | 67.5 |

I aquí podem veure els errors absoluts dels anteriors valors, quan els comparem amb els oficials:

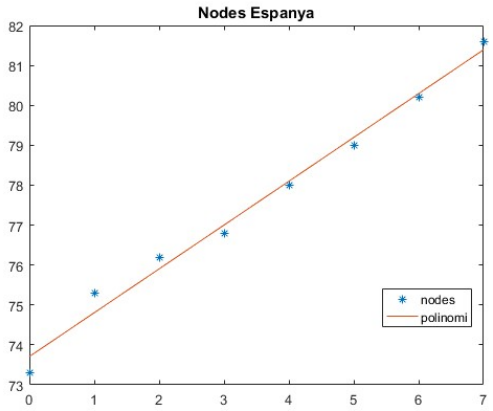
| Grau | 1970 | 1992 | 2007 | 2015 |
|---|---|---|---|---|
| 1 | 0,61 | 0,04 | 0,17 | 0.91 |
| 2 | 0,63 | 0,03 | 0,16 | 0.89 |
| 3 | 0,67 | 0,06 | 0,23 | 0.40 |
| 4 | 2,21 | 0,11 | 0,06 | 1.13 |
| 5 | 4,38 | 0,13 | 0,22 | 1.02 |
| 6 | 4,05 | 0,13 | 0,23 | 1.35 |
| 7 | 13,20 | 0,16 | 0,08 | 15.9 |

Com podem veure tots són molt semblants, però es pot veure que la diferència entre el grau 7 i la resta és que empitjora el 1970 i 2015 (com hem vist abans amb el fenomen de Runge).

Podem sumar el valor de tots els valors absoluts per cada grau per veure quin és l'error "total" per aquests anys a cada grau:

```
     1        1.7286
     2        1.7186
     3         1.364
     4        3.5206
     5        5.7474
     6        5.7657
     7        29.335
```

Sorprenentment, el polinomi de grau 1, per exemple, obté millors resultats que altres, tot i ser lineal:




En aquest cas concret sembla anar bé per casualitat de com ha anat progressant l'esperança de vida al llarg dels anys. El millor, però, és el polinomi de grau 3 en aquest cas.

Ara fem el mateix però amb Belice:

| Grau | 1970 | 1992 | 2007 | 2015 |
|---|---|---|---|---|
| 1 | 68,81 | 69,42 | 69,84 | 70.06 |
| 2 | 68,37 | 69,58 | 69,75 | 69.62 |
| 3 | 63,94 | 69,66 | 69,53 | 74.06 |
| 4 | 60,00 | 69,23 | 69,95 | 70.12 |
| 5 | 67,88 | 69,29 | 70,54 | 62.25 |
| 6 | 87,21 | 68,92 | 69,82 | 81.58 |
| 7 | 88,90 | 68,92 | 69,85 | 79.9 |

I els errors:

| Grau | 1970 | 1992 | 2007 | 2015 |
|---|---|---|---|---|
| 1 | 3,19 | 7,98 | 11,06 | 13.33 |
| 2 | 3,63 | 7,82 | 11,15 | 13.77 |
| 3 | 8,06 | 7,74 | 11,37 | 9.33 |
| 4 | 12,00 | 8,17 | 10,95 | 13.27 |
| 5 | 4,12 | 8,11 | 10,36 | 21.15 |
| 6 | 15,21 | 8,48 | 11,08 | 1.81 |
| 7 | 16,90 | 8,48 | 11,05 | 3.5 |

De nou observem com abans la diferencia al passar a grau 7 als anys 1970 i 2015.

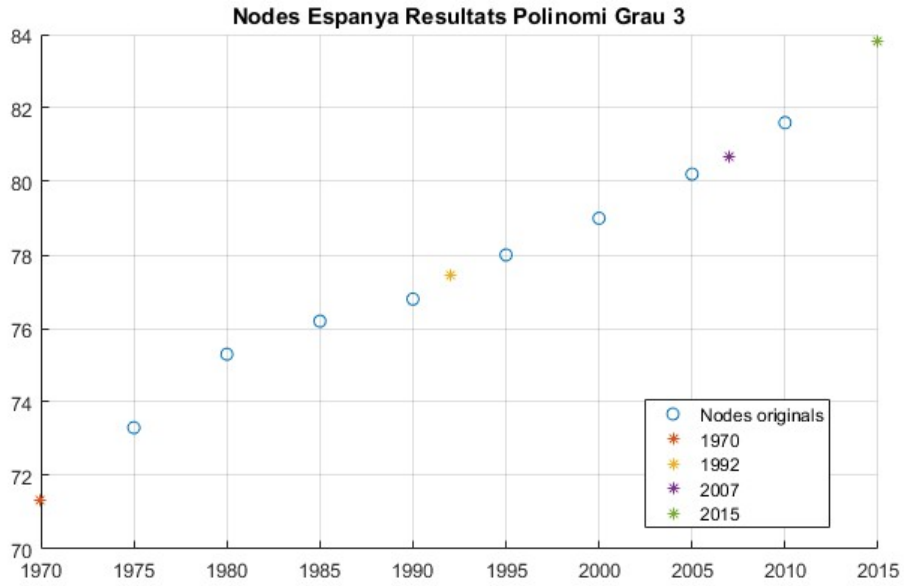
```
1        35.56
2       36.374
3        36.51
4       44.389
5       43.745
6       36.585
7       39.932
```

En aquest cas, però, els valors d'error absolut són considerablement majors. Això es deu a que com hem vist al primer gràfic per comparar les esperances de vida la de Belice, a diferencia de la d'Espanya, és molt irregular. En aquest cas el grau que sembla funcionar millor és el primer, lineal. Això podria variar molt si tinguéssim en compte altres punts, però.

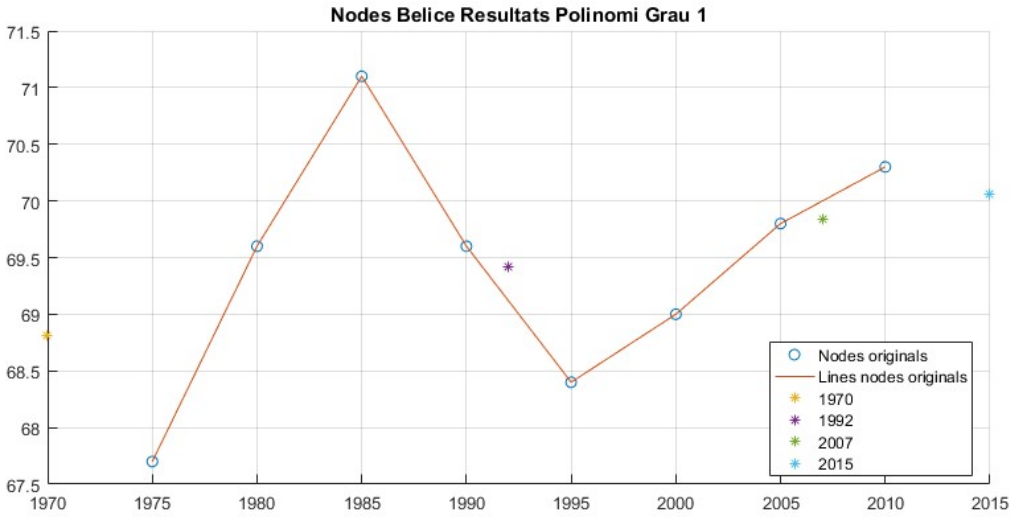
c) A l'apartat anterior ja hem vist les estimacions per l'any 2015, els "millors" models, però, no són els millors per estimar l'esperança de vida d'aquest any. En els dos casos com ja hem vist l'any 2015 "millora" bastant amb el grau 7. A la taula anterior, per exemple, de Belice, podem veure com l'error amb grau 7 es 5.50 mentre que si fem servir una aproximació lineal l'error és 14.59.

   Així que tot i que en general semblen els millors models, per aquest cas en concret són dels pitjors.

d) Ara farem una gràfica on veurem els nodes inicials i els valors estimats de cada país fent servir el millor model trobat, en el cas d'Espanya, fent servir el polinomi de grau 3:




Podem veure que les aproximacions queden bastant bé junt els punts originals. Respecte Belice, he afegit també línies entre els punts per poder veure millor el gràfic:




Aquest, fent servir el polinomi de grau 1, per alguns punts com 1992 ho fa prou bé però altres queden molt lluny degut a la irregularitat de l'evolució de l'esperança al país.

# Annex

## Codis Matlab Exercici 4: Aproximació de dades

```
%% Aproximació dades
clear, clc;

%% Dades
x = 0:7;
y1 = [73.3 75.3 76.2 76.8 78.0 79.0 80.2 81.6];
y2 = [67.7 69.6 71.1 69.6 68.4 69.0 69.8 70.3];
plot(x,y1,'--',x,y2,'-'), title('Esperança de vida')
legend('Espanya','Belice', 'location', 'best')

%% Espanya
% *Useu el polinomi interpolador de grau 7*
n = length(x);
p = polyfit(x,y1,n-1);
coefs = p';

% A = vander(x)
% coefs = A\y1'
% u = 7
% v = polyval(coefs, u)

% Gràfic polinomi
% t = 0:0.1:7; %Per grafic
t = -2:0.1:9; % Pel calcul dels valors
v = polyval(coefs,t);
plot(x,y1,'*',t,v), title('Nodes Espanya')
legend('nodes','polinomi','location','best')

%
% %2007 (2005 es 81)
v(81 + 2 + 2);
err = 80.9 - v(81 + 2 + 2);
abs(err);
%
% %1992 (1990 es 51)
v(51 + 4);
err = 77.4 - v(51 + 4);
abs(err);

%1970
% per sortir "fora" es necessari t = -2:0.1:9;
% En aquest cas 1970
 v(11);
err = 72.0 - v(11);
abs(err);

%% Belice
% *Polinomi interpolador *
n = length(x);
p = polyfit(x,y2,n-1);
coefs = p';

% Gràfic polinomi
t = 0:0.1:7;
```

```
v = polyval(coefs,t);
plot(x,y2,'*',t,v), title('Nodes Belice')
legend('nodes','polinomi','location','best')

%1970
% per sortir "fora" es necessari t = -2:0.1:9;
% En aquest cas 1970
% v(11);
% err = 65.5 - v(11);
% abs(err);
% %2007 (2005 es 81)
% v(81 + 2 + 2);
% err = 69.5 - v(81 + 2 + 2);
% abs(err);
%
% %1992 (1990 es 51)
% v(51 + 4);
% err = 70.7 - v(51 + 4);
% abs(err)

%% Resultats junts
clf,
t = 0:0.1:7;
tR = -1:0.1:8;

% Esp
n = length(x);
p1 = polyfit(x,y1,n-1);
coefs1 = p1';
v1 = polyval(coefs1,t);
v1R = polyval(coefs1,tR);

% Belice
n = length(x);
p2 = polyfit(x,y2,n-1);
coefs2 = p2';
v2 = polyval(coefs2,t);
v2R = polyval(coefs2,tR);

txt1 = 'Espanya nodes';
txt2 = 'Belice nodes';
txt3 = 'Espanya polinomi';
txt4 = 'Belice polinomi';
txt3R = 'Espanya polinomi Runge';
txt4R = 'Belice polinomi Runge';
hold all;
plot(tR,v1R,'Color','k','LineWidth',4,'DisplayName', txt3R),
plot(tR,v2R,'Color','k','LineWidth',4,'DisplayName', txt4R),
plot(t,v1,'Color','y','DisplayName',txt3,'LineWidth',2),
plot(t,v2,'Color','g','DisplayName',txt4,'LineWidth',2),
plot(x,y1,'Marker','*','Color','r','LineStyle','none','DisplayName', txt1)
plot(x,y2,'Marker','*','Color','r','LineStyle','none','DisplayName', txt2)
legend('show','location','best'), title('Esperança de vida'), grid('on')
```

```
%% Aproximació dades, apartats b-d
clear, clc;
format short g;

%% Dades
x = 0:7;
y1 = [73.3 75.3 76.2 76.8 78.0 79.0 80.2 81.6];
y2 = [67.7 69.6 71.1 69.6 68.4 69.0 69.8 70.3];
% plot(x,y1,'--',x,y2,'-'), title('Esperança de vida')
% legend('Espanya','Belice', 'location', 'best')

%% Espanya
% *Busqueu un polinomi del grau escaient per mínims quadrats*
% hold all
taula(1,:) = [0,0,0,0,0];
for i = 1:7
    p = polyfit(x,y1,i);
    coefs = p';
    t = -2:0.1:9;
    v = polyval(coefs,t);

    % 2007
    valor2007 = v(81 + 2 + 2);
    err = 80.9 - v(81 + 2 + 2);
    error2007 = abs(err);

    % 1992
    valor1992 = v(51 + 4);
    err = 77.4 - v(51 + 4);
    error1992 = abs(err);

    % 1970
    valor1970 = v(11);
    err = 72.0 - v(11);
    error1970 = abs(err);

    % 2015
    valor2015 = v(101);
    err = 83.4 - v(101);
    error2015 = abs(err);

    taula(i,:) = [i, valor1970, valor1992, valor2007, valor2015];
    taulaErr(i,:) = [i, error1970, error1992, error2007, error2015];
end
% disp('Taula resultat valors');
% disp(taulaErr);
% filename = 'taulaValors2.xlsx';
% xlswrite(filename,taula);
% filename = 'taulaErrors.xlsx';
% xlswrite(filename,taulaErr);
for j = 1:size(taulaErr)
   taulaTotal(j, :) = [j, (sum(taulaErr(j,:)) - j)];
end
disp(taulaTotal)

%% Belice
% *Busqueu un polinomi del grau escaient per mínims quadrats*
% hold all
```

```
taulaB(1,:) = [0,0,0,0,0];
for i = 1:7
    p = polyfit(x,y2,i);
    coefs = p';
    t = -2:0.1:9;
    v = polyval(coefs,t);

    % 2007
    valor2007 = v(81 + 2 + 2);
    err = 80.9 - v(81 + 2 + 2);
    error2007 = abs(err);

    % 1992
    valor1992 = v(51 + 4);
    err = 77.4 - v(51 + 4);
    error1992 = abs(err);

    % 1970
    valor1970 = v(11);
    err = 72.0 - v(11);
    error1970 = abs(err);

    % 2015
    valor2015 = v(101);
    err = 83.4 - v(101);
    error2015 = abs(err);

    taulaB(i,:) = [i, valor1970, valor1992, valor2007, valor2015];
    taulaErrB(i,:) = [i, error1970, error1992, error2007, error2015];
end
disp('Taula resultat valors');
disp(taulaErrB);
% filename = 'taulaValorsB.xlsx';
% xlswrite(filename,taulaB);
% filename = 'taulaErrorsB.xlsx';
% xlswrite(filename,taulaErrB);
for j = 1:size(taulaErrB)
   taulaTotalB(j, :) = [j, (sum(taulaErrB(j,:)) - j)];
end
disp(taulaTotalB)

%% Gràfic polinomi espanya
clear, clc;
format short g;
x = 0:7;
y1 = [73.3 75.3 76.2 76.8 78.0 79.0 80.2 81.6];
p = polyfit(x,y1,3);
coefs = p';
t = -2:0.1:9;
v = polyval(coefs,t);
valor2007 = v(81 + 2 + 2);
valor1992 = v(51 + 4);
valor1970 = v(11);
valor2015 = v(101);
x = 1975:5:2010;
hold all
plot(x,y1,'o','DisplayName','Nodes originals')
plot(1970,valor1970,'*','DisplayName','1970')
plot(1992,valor1992,'*','DisplayName','1992')
```

```
plot(2007,valor2007,'*','DisplayName','2007')
plot(2015,valor2015,'*','DisplayName','2015')
title('Nodes Espanya Resultats Polinomi Grau 3')
grid on
legend('location','best')

%% Gràfic polinomi belice
clear, clc;
format short g;
x = 0:7;
y2 = [67.7 69.6 71.1 69.6 68.4 69.0 69.8 70.3];
p = polyfit(x,y2,1);
coefs = p';
t = -2:0.1:9;
v = polyval(coefs,t);
valor2007 = v(81 + 2 + 2);
valor1992 = v(51 + 4);
valor1970 = v(11);
valor2015 = v(101);
x = 1975:5:2010;
hold all
plot(x,y2,'o','DisplayName','Nodes originals')
plot(x,y2,'DisplayName','Lines nodes originals')
plot(1970,valor1970,'*','DisplayName','1970')
plot(1992,valor1992,'*','DisplayName','1992')
plot(2007,valor2007,'*','DisplayName','2007')
plot(2015,valor2015,'*','DisplayName','2015')
title('Nodes Belice Resultats Polinomi Grau 1')
grid on
legend('location','best')
```

Tots aquests codis estan també a la carpeta **Codis**