UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

Doble grau en Matemàtiques i Administració i Direcció d'Empreses

Treball final de grau

APLICACIÓ DEL MÈTODE DE L'ANÀLISI MÈTRIC SOBRE L'ESPERANÇA DE VIDA

Autor: Guillem Sànchez Escoi

Directores:	Dra. Carme Florit Selma
	Dra. María Reyes Pérez Domingo
Realitzat a:	Departament de Matemàtiques i Informàtica
	Departament de Matemàtica Econòmica, Financera i Actuarial
Barcelona,	13 de juny de 2022

Abstract

Knowing the date on which you will die has been a question that has been with human thoughts since its inception. Currently, the amount of information that is collected over the course of a year allows a very accurate calculation of the life expectancy of a person who has survived the year in question.

However, the constant technological and health innovations of the society imply that these figures vary from year to year. In addition, these variations may occur for specific ages, and may not affect others in any way.

In this Bachelor's Degree Final Project we will apply an interpolation and extrapolation method using metric analysis on functions of multiple variables devised by Dr. Aleksandr Vitalievich Kryanev. Adapted to life expectancy, the method will allow to know the life expectancy of an individual with age x at the year y.

To do this, we will use the data collected since 1991 by the National Statistics Institute (Spanish: *Instituto Nacional de Estadística*, INE). This data are structured in the form of a table, and they give the value of life expectancy according to your age (x) and the year in which you are located (y).

Resum

Saber la data en la qual moriràs ha estat una pregunta que porta en els pensaments dels éssers humans des dels seus inicis. Actualment, la quantitat d'informació que és recopilada durant el transcurs d'un any permet fer un càlcul molt acurat sobre l'esperança de vida que tindrà una persona que ha sobreviscut a l'any en qüestió.

No obstant, les constants innovacions tecnològiques i sanitàries de la societat fa que any rere any aquestes xifres vagin variant. A més a més, aquestes variacions poden donar-se per a unes edats concretes, i no afectar en res a d'altres.

En aquest Treball de Fi de Grau aplicarem un mètode d'interpolació i extrapolació usant l'anàlisi mètric sobre funcions de múltiples variables ideat pel doctor Aleksandr Vitalievich Kryanev. Adaptat a l'esperança de vida, el mètode permetrà saber l'esperança de vida d'un individu d'edat x a l'any y.

Per a fer-ho, es farà ús de les dades recopilades des de l'any 1991 per *l'Institut Nacional d'Estadística (INE)*. Aquestes dades estan estructurades en forma de taula, i donen valor a l'esperança de vida segons l'edat que tinguis (x) i l'any en el que s'estigui situat (y).

Agraïments

Primer de tot m'agradaria donar les gràcies a les meves tutores Carme Florit Selma i María Reyes Domingo Pérez per la seva disposició durant tot el treball. La seva ajuda, els seus consells i el temps que hi han dedicat han estat fonamentals per a poder dur a terme aquest projecte. Els hi estic molt agraït per haver tingut l'oportunitat de treballar amb elles.

També vull agrair a tota la gent que he conegut i que ha estat al meu voltant durant aquests sis anys de carrera. Compartir aquest viatge amb vosaltres ho ha fet tot més fàcil, moltes gràcies per tots els moments viscuts.

Per últim, m'agradaria donar les gràcies a la meva família. Hi ha hagut moments difícils durant aquests anys, però gràcies a la confiança que m'han mostrat i l'empenta que m'han donat, m'han ajudat a continuar i he aconseguit superar els obstacles. Sou el més important que tinc.

2020 Mathematics Subject Classification. 11G05, 11G10, 14G10

Índex

1 Introducció

1.1 Definició de l'esperança de vida

L'esperança de vida correspon a la mitjana d'anys que encara li quedarien per viure a una persona que ha assolit una edat exacta, si en el temps que li resta de vida fos sotmesa a les condicions de mortalitat actuals.

En el cas particular de l'esperança de vida a l'edat zero, o també coneguda com l'esperança de vida en néixer, representa la durada de vida mitjana d'una generació fictícia sotmesa a les condicions de mortalitat del període.

Per a calcular l'esperança de vida dels que tenen k anys en l'any t, $EV_k(t)$, es fa la mitjana de l'edat de les persones mortes (amb k o més anys) en un any, és a dir:

$$EV_k(t) = \sum_{j=k}^{\infty} p_j(t) \cdot j - k$$

on $p_j(t)$ és el tant per u de persones mortes a l'edat de j anys en l'any t respecte a les persones mortes amb k o més anys aquell any t. [1]

També es pot calcular seguint amb aquesta variant de la fórmula , que més coneguda:

$$EV_k(t) = \frac{T_{k_t}}{l_{k_t}}$$

on T_{k_t} surt de calcular la suma del número de anys que li queden per a viure a totes les persones vives amb edat k en l'any t, i l_{k_t} aquesta xifra de persones vives d'edat k en l'any t.
Per exemple, una persona de 24 anys l'any 2020 li queda per viure encara 50 anys, però una altra d'aquesta mateixa edat en aquest precís any li quedaria 35. La suma d'aquests anys que els hi queden per a viure a aquests individus (l_{k_t}) formen T_{k_t}.

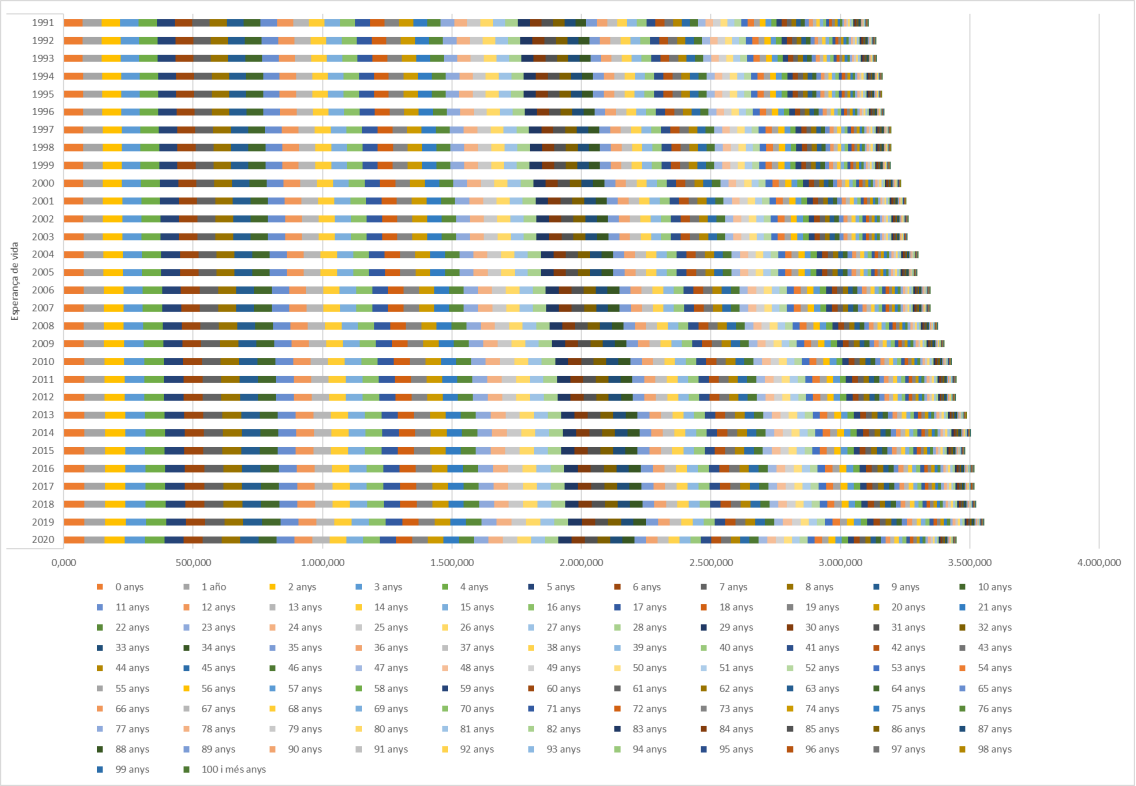
Aplicant un seguit d'operacions aritmètiques i aplicant definicions [2], es pot abreviar l'eqüació a:

$$EV_k(t) = 0,5 + \sum_{j=1}^{w-k-1} \frac{l_{(k+j)_t}}{l_{k_t}}$$

on w és el límit d'anys que pot viure una persona. És a dir, en $w+1$ no queda cap persona viva.

1.2 Presentació de les dades

Vistes ja la definició i les diferents maneres que hi ha té per a calcaluar-la, centrem-nos ara en com ha evolucionat l'esperança de vida a Espanya en aquests darrers 30 anys.



Font: *Instituto Nacional de Estadística (INE)*

Figura 1: Número d'anys acumulats si es sumen totes les esperances de vida en cada any

La Figura 1 mostra clarament com hi ha hagut un augment considerable de l'esperança en les darreres tres dècades. Això es deu a que aquesta suma engloba els petits canvis que es donen en cada franja d'edat.
Per exemple, si l'esperança de vida a l'any 2005 per a un infant de 3 anys era de 72,410 i l'any 2006 era de 73,074, i es mantinguessin totes les altres dades, la fila del 2006 seria la del 2005 desplaçada 0,664 anys. El sumatori de tots aquests canvis provoquen en la suma total variacions que poden anar des de l'any a dècades.
Dels canvis socioeconòmics, sanitaris i demogràfics que es venen donant durant el llarg del segle XX fins arribar a l'actualitat, n'hi hauria dos de claus per al profesor Manuel García González en [3]:

- **El control de la mortalitat en els nens menors de 4 anys:** És responsable, per si sol, d'un 50 % de l'increment assolit ja que les millores a edats més primerenques tenen un impacte més gran en la taxa global (es guanya molta més edat si es corregeixen les morts en nadons que en adults). Hi ha tingut a veure els avenços en l'atenció del part, així com la lluita contra les principals malalties infeccioses: meningitis, pneumònies, diarrees o la tuberculosi.

- **El control de les malalties cardiovasculars:** Els avenços enfront dels accidents cerebrovasculars (ictus) o patologia isquèmica (infarts), gràcies a les millores introduïdes al control de la hipertensió arterial, el colesterol, el tractament de la insuficiència cardíaca o la recent reducció del tabaquisme entre els homes.

Aquesta evolució es va veure truncada l'any 2020 degut a la pandèmia ocasionada pel virus de la COVID-19. L'esperança de vida va caure en 1,41 anys de mitjana respecte l'any 2019, suposant el major descens des de la Segona Guerra Mundial.

1.3 Conseqüències econòmiques de l'expansió de l'esperança de vida

Si alguna cosa va demostrar la pandèmia de la COVID-19 és que les futures millores en l'esperança de vida són incertes i difícils de predir. Una de les tasques més difícils en la creació de models del risc de longevitat és el tema del risc de longevitat sistemàtic i en alguns casos, encara que menys freqüents, de longevitat extrema. Aquests es tracten de casos en què les persones viuen molt més del que s'esperava.

La longevitat implica que, cada cop més, les persones corren el risc de sobreviure als estalvis que han acumulat durant la seva vida laboral. És per això que volen busquen assegurar aquest risc a través dels sistemes públics de seguretat social PAYGO, de plans de pensions ocupacionals, d'assegurances de vida privades i de productes de rendes vitalícies, o bé mitjançant hipoteques inverses.

Aquestes diferents formes de rendes vitalícies són, en realitat, assegurances contra el risc de longevitat de les persones. En ells s'agrupen el risc de viure més del que s'esperava, i, per tant, de necessitar més recursos a la jubilació, entre els titulars de rendes vitalícies o membres d'un pla de pensions.

Per als titulars dins del sector de les assegurances i els fons de pensions, sempre ha estat crucial tenir accés a un model fiable de mortalitat que es pogués utilitzar per calcular preus i reserves. També per a gestionar el risc, particularment en productes com ara rendes vitalícies, els pagaments de les quals estan supeditats a la supervivència.

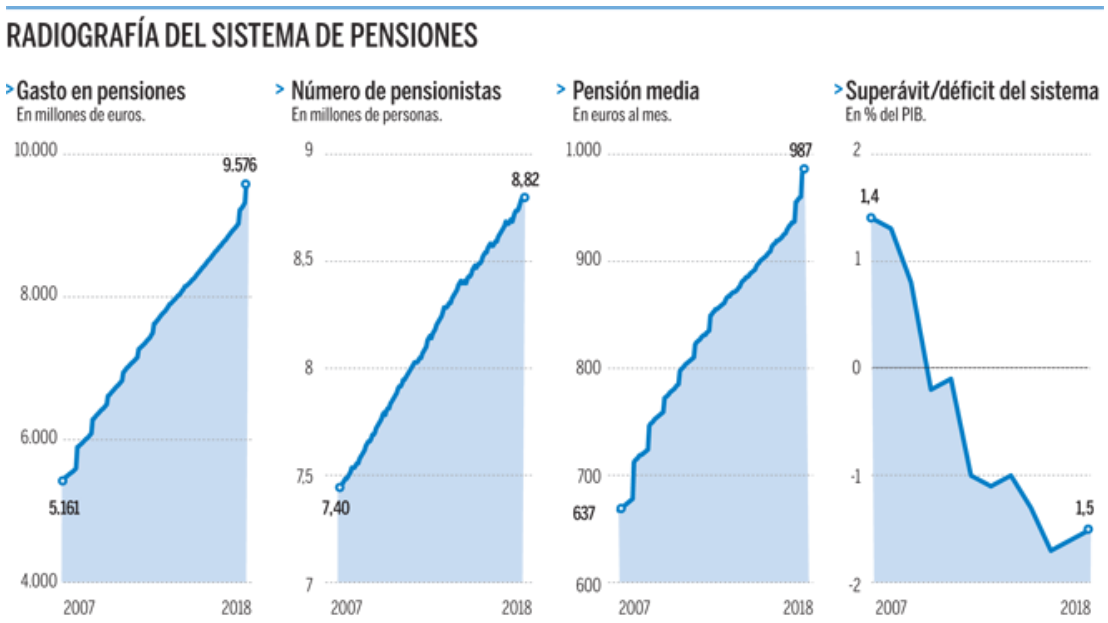
Per tant, les suposicions sobre les probabilitats de supervivència, donada l'edat real dels titulars de les rendes o jubilats, esdevenen unes de les coses més essencials a l'hora d'establir els preus d'aquests contractes o productes.

Les taules de vida que incorporen una previsió de les tendències futures de la mortalitat són l'instrument més popular utilitzat per representar la distribució subjacent de la durada de la vida futura, i l'exactitud d'aquestes taules depèn de la fiabilitat de les dades de mortalitat.

En els contractes tradicionals, els asseguradors (governs, fons de pensions, empreses d'assegurances o provïdors de rendes vitalícies) corren el risc que les projeccions de la mortalitat puguin resultar incorrectes, fent que els assegurats acabin vivint més del que s'esperava. La quantitat total d'exposició al risc de longevitat global en relació amb les pensions a les empreses del sector privat s'ha estimat en 25 bilions de dòlars.

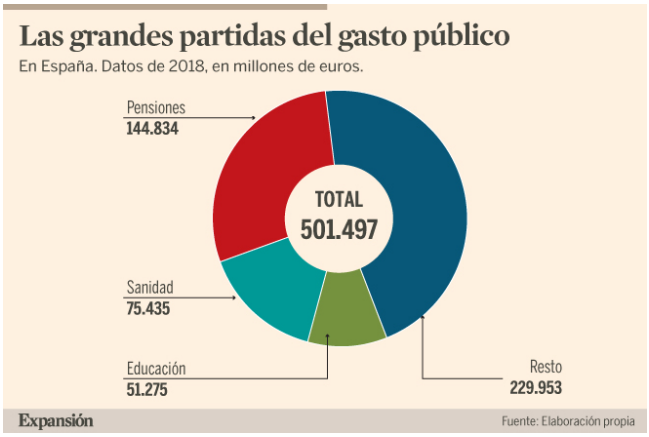
Abans, les empreses d'assegurances solien ser capaces de compensar qualsevol esdeveniment advers al risc de longevitat gràcies als rendiments d'altres inversions rendibles. No obstant, això és cada vegada més difícil.
La tendència global cap a la desregulació i la liberalització dels mercats d'assegurances ha comportat una ferotge competència i la disminució dels marges de beneficis.

Per als sistemes públics de pensions, els desafiaments de la creixent esperança de vida i del risc de longevitat agregat són tant polítics com financers. Consegüentment a aquests canvis demogràfics, socials i econòmics, diversos països de l'OCDE porten canviant els seus sistemes de pensions des dels anys noranta per assegurar la viabilitat a llarg termini. [4]



Font: *Diari Expansión* [5]

Figura 2: Evolució del sistema de pensions des del 2005 fins el 2018



Font: *Diari Expansión* [6]

Figura 3: Despesa pública espanyol durant l'any 2018

En la Figura 2 queda visualitzat el problema. El número de pensionistes no ha parat de créixer, cosa que implica que el import mensual en pensions també ho faci. La conseqüència més directa és que aquest augment cada cop creixent incideix de ple en el gast públic espanyol, ja que és una de les principals partides pressupostàries (Figura 3. Per tant, el dèficit del sistema en % sobre el PIB no ha fet sinó augmentar en aquest interval.

Les diferents maneres en què les pensions futures es veuran afectades pels canvis a l'esperança de vida i el problema de com compartir la càrrega de tals ajustaments entre els contribuents i els jubilats d'avui, i els contribuents i els jubilats futurs, suposa un tema crític.

Per tal de fer front a aquesta incertesa, i aplicant el mètode matemàtic de l'anàlisi mètric, intentarem obtenir prediccions acurades amb les dades que prèviament hem obtingut. A més a més, i tal hi com hem narrat en l'*Abstract*, ho farem de forma individualitzada segons quina sigui l'edat de l'individu.

2 Mètode de l'anàlisi mètric

2.1 Justificació matemàtica del mètode escollit

L'objectiu de tot mètode d'interpolació es trobar una funció $F(X)$ tal que:

$$Y = F(X_1, \ldots, X_m) = F(\overrightarrow{X}) \tag{2.1}$$

i aquesta funció evaluar-la en un punt $\overrightarrow{X^*}$. Aquest punt es troba dins d'un conjunt delimitat pels punts $\overrightarrow{X_k}, k = 1, \ldots, n$, que son de la forma $\overrightarrow{X_k} = (X_{k_1}, \ldots, X_{k_m})$.

La imatge d'aquests $\overrightarrow{X_k}$ en $F(\overrightarrow{X})$ són valors coneguts. Concretament, $Y_k = F(\overrightarrow{X_k}), k = 1, \ldots, n$. ([7])

Seguint l'esquema de l'anàlisis mètric, dissenyem la matriu W per una mètrica desconeguda pel punt $\overrightarrow{X^*}$ en el conjunt de punts $\overrightarrow{X_k} = (X_{k_1}, \ldots, X_{k_m}), k = 1, \ldots, n$.

$$W = \begin{pmatrix} \rho^2(\overrightarrow{X_1}, \overrightarrow{X^*})_\omega & (\overrightarrow{X_1}, \overrightarrow{X_2})_\omega & \cdots & (\overrightarrow{X_1}, \overrightarrow{X_n})_\omega \\ (\overrightarrow{X_2}, \overrightarrow{X_1})_\omega & \rho^2(\overrightarrow{X_2}, \overrightarrow{X^*})_\omega & \cdots & (\overrightarrow{X_2}, \overrightarrow{X_n})_\omega \\ \cdots & \cdots & \cdots & \cdots \\ (\overrightarrow{X_n}, \overrightarrow{X_1})_\omega & (\overrightarrow{X_n}, \overrightarrow{X_2})_\omega & \cdots & \rho^2(\overrightarrow{X_n}, \overrightarrow{X^*})_\omega \end{pmatrix} \tag{2.2}$$

on $\rho^2(\overrightarrow{X_i}, \overrightarrow{X^*})_\omega = \sum_{k=1}^m \omega_k \cdot (X_{ik} - X_k^*)^2, (\overrightarrow{X_i}, \overrightarrow{X_j})_\omega = \sum_{k=1}^m \omega_k \cdot (X_{ik} - X_k^*) \cdot (X_{j_k} - X_k^*)$ [8]
El pes ω_k determina el grau de tolerància o sensibilitat de la funció respecte els seus arguments. Els seus valors es poden obtenir de diferents maneres. En aquest treball, ho farem mitjançant les "taxes de correlació"dels punts on el valor de la funció (2.1) és conegut per a després normalitzar-les [9]:

$$\omega_k = \frac{|r_k|}{\sum_{k=1}^m |r_k|}, k = 1, \ldots, m \tag{2.3}$$

on

$$r_k = \frac{cov(Y, X_k)}{\sigma(Y) \cdot \sigma(X_k)}, k = 1, \ldots, m$$

$$cov(Y, X_k) = \frac{1}{n-1} \cdot \sum_{j=1}^n (Y_j - \overline{Y})(X_{j_k} - \overline{X_k}), k = 1, \ldots, m$$

$$\sigma^2(X_k) = \frac{1}{n-1} \cdot \sum_{j=1}^n (X_{j_k} - \overline{X_k})^2, k = 1, \ldots, m$$

$$\sigma^2(Y) = \frac{1}{n-1} \cdot \sum_{j=1}^n (Y_j - \overline{Y})^2,$$

$$\overline{Y} = \frac{1}{n} \cdot \sum_{j=1}^n Y_j, \quad \overline{X_k} = \frac{1}{n} \cdot \sum_{j=1}^n X_{j_k}, k = 1, \ldots, m$$

D'acord amb la definició de la matriu W donada en (2.2) es tracta d'una matriu simètrica i no-negativa. A més a més es pot assumir que la matriu és definida positiva.

Ara, suposem que la fórmula que ens ha de tornar el valor Y^* en el punt $\overrightarrow{X^*}$ té la forma de una combinació lineal de totes les solucions conegudes, és a dir:

$$Y^* = \sum_{j=1}^{n} z_j \cdot Y_j = (\overrightarrow{z}, Y) \tag{2.4}$$

on els pesos $z_j, j = 1, \ldots, n$ com és habitual en les fórmules d'interpolació satisfan que estan normalitzats. Això és: $\sum_{j=1}^{n} z_j = 1$.

Definim un valor numèric que mesuri la incertesa del valor restaurat Y^* en el punt $\overrightarrow{X^*}$ en base als valors coneguts en $\overrightarrow{X_1}, \ldots, \overrightarrow{X_n}$ [10] :

$$\sigma^2_{vn}(Y^*) = (W(\overrightarrow{X^*}; \overrightarrow{X_1}, \ldots, \overrightarrow{X_n}) \cdot \overrightarrow{z}, \overrightarrow{z}), \tag{2.5}$$

on $\overrightarrow{z} = (z_1, \ldots, z_n)^\intercal$

El problema que se'ns planteja es minimitzar el valor d'incertesa, tot respectant les condicions de normalització de $\overrightarrow{z}$

$$\begin{cases} (W\overrightarrow{z}, \overrightarrow{z}) - min\overrightarrow{z} \\ (\overrightarrow{z}, \overrightarrow{1}) = 1, \quad \overrightarrow{1} = (1, \ldots, 1)^\intercal \end{cases} \tag{2.6}$$

Per a resoldre el problema(2.6), farem ús del mètode de Lagrange.
Primerament, observem que per la definició de W, al tractar-se d'una matriu simètrica, els valors propis associats a la matriu seran reals ([11]), podem aplicar teorema de Schur ([12]). Fent-lo servir obtenim que W sempre serà diagonalitzable. Siguin ara $\lambda_1, \ldots, \lambda_n$ els seus respectius valors propis i $\overrightarrow{\phi_1}, \ldots, \overrightarrow{\phi_n}$ els vectors propis associats de tal manera que formen una base ortonormal. D'aquesa manera, podem reformular $\overrightarrow{z}$ i σ^2_{vn} de la següent manera:

$$\begin{aligned} \overrightarrow{z} = \sum_{j=1}^{n} c_i \cdot \overrightarrow{\phi_j} \rightarrow W\overrightarrow{z} = \sum_{j=1}^{n} c_j \cdot W\overrightarrow{\phi_j} \overset{1}{=} \sum_{j=1}^{n} c_i \cdot \lambda_j \cdot \overrightarrow{\phi_j} \\ \sigma^2_{vn} = (W\overrightarrow{z}, \overrightarrow{z}) = (\sum_{i=1}^{n} c_i \cdot \lambda_i \cdot \overrightarrow{\phi_i}, \sum_{j=1}^{n} c_j \cdot \overrightarrow{\phi_j}) \overset{2}{=} \sum_{j=1}^{n} c_j^2 \cdot \lambda_j \\ (\overrightarrow{z}, \overrightarrow{1}) = (\sum_{j=1}^{n} c_j \cdot \overrightarrow{\phi_j}, \overrightarrow{1}) = \sum_{j=1}^{n} c_j \cdot (\overrightarrow{\phi_j}, \overrightarrow{1}) \end{aligned} \tag{2.7}$$

En 1 hem utilitzat la definició de vectors i valors propis, i en 2 que $\langle \overrightarrow{\phi_i}, \overrightarrow{\phi_j} \rangle = 0$ si $i \neq j$ al ser ortogonals entre si i $\langle \overrightarrow{\phi_i}, \overrightarrow{\phi_i} \rangle = 1$ al ser una base ortonormal.

Amb això, el problema (2.6) es pot reescriure com:

$$\begin{cases} \sum_{j=1}^{n} c_j^2 \cdot \lambda_j - min\overrightarrow{c} \\ \sum_{j=1}^{n} c_j \cdot (\overrightarrow{\phi_j}, \overrightarrow{1}) = 1 \end{cases} \tag{2.8}$$

Un cop fet el canvi de notació, resolem pel mètode de Lagrange (veure els passos a [7]) i obtenim:

$$\overrightarrow{z^*} = \frac{\sum_{j=1}^{n} \frac{(\overrightarrow{\phi_j}, \overrightarrow{1})}{\lambda_j} \cdot \overrightarrow{\phi_j}}{\sum_{j=1}^{n} \frac{(\overrightarrow{\phi_j}, \overrightarrow{1})}{\lambda_j}}, \quad Y^* = \frac{\sum_{j=1}^{n} \frac{(\overrightarrow{\phi_j}, \overrightarrow{1})}{\lambda_j} \cdot (\overrightarrow{\phi_j}, \overrightarrow{Y})}{\sum_{j=1}^{n} \frac{(\overrightarrow{\phi_j}, \overrightarrow{1})}{\lambda_j}} \tag{2.9}$$

Sabem que $W\overrightarrow{\phi_1} = \lambda_1 \cdot \overrightarrow{\phi_1}, \ldots, W\overrightarrow{\phi_n} = \lambda_n \cdot \overrightarrow{\phi_n}$, o equivalentment $W(\overrightarrow{\phi_1}, \ldots, \overrightarrow{\phi_n}) = (\lambda_1\overrightarrow{\phi_1}, \ldots, \lambda_n\overrightarrow{\phi_n})$.
Sigui $V = (\overrightarrow{\phi_1}, \ldots, \overrightarrow{\phi_n})$ i $D = diag(\lambda_1, \ldots, \lambda_n)$. Veiem que es pot recompondre tot el sistema de vectors i valors propis de forma matricial com $MV = DV$. Com V és invertible al estar formada per vectors linealment independents, podem trobar que $M = VDV^{-1}$.

Alhora, aquesta descomposició permet també donar un valor per a M^{-1}, que és $M^{-1} = VD^{-1}V$. Fixant-nos de nou amb el resultat donat en (2.9), podem reestructurar $\overrightarrow{z^*}$ i Y^* com [13]:

$$\overrightarrow{z^*} = \frac{W^{-1}\overrightarrow{1}}{(W^{-1}\overrightarrow{1}, \overrightarrow{1})} \quad Y^* = \frac{(W^{-1}\overrightarrow{Y}, \overrightarrow{1})}{(W^{-1}\overrightarrow{1}, \overrightarrow{1})} \tag{2.10}$$

Ara bé, pot succeïr que la matriu W sigui singular (la matriu tingui determinant 0). Si ocorrés, s'hauria de fer un reajustament a l'equació (2.10) de manera que quedés [8]:

$$\overrightarrow{z^*} = \frac{W^{+}\overrightarrow{1}}{(W^{+}\overrightarrow{1}, \overrightarrow{1})} \quad Y^* = \frac{(W^{+}\overrightarrow{Y}, \overrightarrow{1})}{(W^{+}\overrightarrow{1}, \overrightarrow{1})} \tag{2.11}$$

on W^+ és la pseudoinversa de la matriu W. El càlcul d'aquestes matrius el deixem per a la subsubsecció 2.2.1.

Suposem que coneixem el seu càlcul, i també assumim que $(W^{+}\overrightarrow{1}, \overrightarrow{1}) > 0$. Aleshores, recuperant la fórmula del valor numèric de l'incertesa de (2.5) i substituïnt obtenim:

$$\begin{aligned}\sigma^2_{vn}(Y^*) = (W \cdot \overrightarrow{z}, \overrightarrow{z}) = \frac{(WW^{+}\overrightarrow{1}, W^{+}\overrightarrow{1})}{(W^{+}\overrightarrow{1}, \overrightarrow{1})^2} = \frac{(W^{+}WW^{+}\overrightarrow{1}, \overrightarrow{1})}{(W^{+}\overrightarrow{1}, \overrightarrow{1})^2} \overset{1}{=} \\ \overset{1}{=} \frac{(W^{+}\overrightarrow{1})}{(W^{+}\overrightarrow{1}, \overrightarrow{1})^2} = \frac{1}{(W^{+}\overrightarrow{1}, \overrightarrow{1})}\end{aligned} \tag{2.12}$$

Notem que, en 1 hem utilitzar una de les propietats de les *condicions de Penrose* de l'apartat 2.2.1.

El valor invers $I(\overrightarrow{X^*}; \overrightarrow{X_1}, \ldots, \overrightarrow{X_n}) = (W^{+}\overrightarrow{1}, \overrightarrow{1}) = \sum_{i=1}^{n}\sum_{j=1}^{n} W^{+}_{ij}$ ens dóna la informació mètrica del punt $\overrightarrow{X^*}$ respecte el conjunt de punts $\overrightarrow{X_1}, \ldots, \overrightarrow{X_n}$.. De les propietats de les matrius pseudoinverses, és conegut que, al afegir un nou punt $\overrightarrow{X_n}$ a un conjunt de punts $\overrightarrow{X_1}, \ldots, \overrightarrow{X_n}$, la informació mètrica en qualsevol punt $\overrightarrow{X^*}$ respecte al conjunt de punts $\overrightarrow{X_1}, \ldots, \overrightarrow{X_n}, \overrightarrow{X_{n+1}}$ no és menor a la informació mètrica en el punt $\overrightarrow{X^*}$ respecte el conjunt de punts $\overrightarrow{X_1}, \ldots, \overrightarrow{X_n}$.

$$I(\overrightarrow{X^*}; \overrightarrow{X_1}, \ldots, \overrightarrow{X_n}, \overrightarrow{X_{n+1}}) \geq I(\overrightarrow{X^*}; \overrightarrow{X_1}, \ldots, \overrightarrow{X_n}),$$

provocant que per la mesura mètrica de l'incertesa es compleixi ([8]):

$$\sigma^2_{vn}(\overrightarrow{X^*}; \overrightarrow{X_1}, \ldots, \overrightarrow{X_n}, \overrightarrow{X_{n+1}}) \leq \sigma^2_{vn}(\overrightarrow{X^*}; \overrightarrow{X_1}, \ldots, \overrightarrow{X_n})$$

2.2 Càlculs intermitgos del procès

2.2.1 Les matrius pseudoinverses

Els sistemes d'equacions $Ax = b$ amb $A = \mathbb{R}^{m\times n}$,$m > n$ y $b \in \mathbb{R}^m$ i $rang(A|b) \neq rang(A) \leq n$, no tenen solució, però se'ls hi pot trobar una *pseudosolució* seguint el criteri de trobar una $x \in \mathbb{R}^n$ que minimitzi la norma $\|Ax - b\|_2$. També escollir entre les *pseudosolucions* existents aquella que tingui una norma mínima.
Els procediments que s'encarreguen de buscar solucions que minimitzin les normes reben el nom de **miníms quadrats**. En forma de expressió matemàtica, si anomenem X és el conjunt de solucions d'aquests *problemes de mínims quadrats* [14]:

$$X = \{x \in \mathbb{R}^n : \|Ax - b\|_2 = min\},$$

on $\|y\|_2 := \sqrt{y_1^2 + \cdots + y_n^2}$ Els procediments més usats pr a resoltre aquests *problemes de mínims quadrats* comporten la reducció de la matriu A a alguna forma canònica mitjançant transformacions ortogonals. Les matrius pseudoinverses permeten resoldre aquests problemes, i per a definir les seves propietats i com es construeixen ens basarem en [15].

Definició 2.1. *Donada una matriu $A \in \mathbb{R}^{m\times n}$, la matriu pseudoinversa es tracta de l'única matriu $A^+ \in \mathbb{R}^{n\times m}$ que satisfà les següents quatre condicions, anomenades condicions de Penrose :*

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. $A^+A = (A^+A)^\intercal$
4. $AA^+ = (AA^+)^\intercal$

Existeixen altres propietats rellevants de la pseudoinversa, que poden ser deduïdes de les condicions anteriors i que també mantenen una analogia amb les propietats de l'inversa pel cas de matrius quadrades. Si recordem l'equació (2.11) i la definició de W de (2.10), notem que aquest és propietats ens interessen, al ser W una matriu quadrada per deinició.

Teorema 2.2. *Sigui $A \in \mathbb{R}^{m\times n}$.La pseudoinversa A^+ satisfà les següents propietats:*

1. $(A^+)^+ = A$

2. $(A^+)^\intercal = (A^\intercal)^+$

3. $(\alpha A)^+ = \alpha^{-1}A^+,\ \forall\alpha \in \mathbb{R}\ tal\ que\ \alpha \neq 0$

4. $(A^\intercal A)^+ = A^+(A^+)^\intercal$

5. Si $AA^\intercal = A^\intercal A$, *aleshores* $A^+A = AA^+$ *i* $(A^n)^+ = (A^+)^n\ \forall n \in \mathbb{Z}$

6. $rang(A) = rang(A^\intercal) = rang(A^+) = rang(A^+A) =$*traça*$(A^+A)$

Si m=n i A no és singular, es evident que A^{-1} satisfà de manera trival les quatre condicions de Penrose, o equivalentment, que $A^+ = A^{-1}$.
El càlcul d'una pseudoinversa per a una matriu $A \in \mathbb{R}^{m\times n}$ de rang complet és senzill de realitzar. Les seves fórmules varien segons el tamany de m i n:

1. $A^+ = A^\intercal(A^\intercal A)^{-1}$, si $m \leq n$
2. $A^+ = (A^\intercal A)^{-1}A^\intercal$, si $n \leq m$

No obstant, si el rang no és complet, a l'hora de fer $(A^\intercal A)^{-1}$ hi haurà problemes, ja que $A^\intercal A$ tindrà determinant = 0. S'haurá d'introduïr un nou concepte: els **valors singulars**. Però primer, un teorema.

Teorema 2.3. *Els valors propis de $A^\intercal A$ són no negatius*

demostracio: Si $\lambda \in \sigma(A^\intercal A)$ (valor espectral de $A^\intercal A$) té com a vector propi associat v_1 (que podem considerar-lo unitari $\|v_1\| = 1$), aleshores:

$$\lambda = \lambda\|v_1\|^2 = \lambda\langle v_1, v_1\rangle = \langle v_1, \lambda v_1\rangle = \langle v_1, A^\intercal A v_1\rangle = \langle Av_1, Av_1\rangle = \|Av_1\|^2 \geq 0$$

Els **valors singulars** d'una matriu arbitraria $A \in \mathbb{R}^{m\times n}$ són les r arrels quadrades (postives) dels r valors propis no negatius de $A^\intercal A$. Aquests valors singulars es denoten de forma habitual per $\sigma_i(= \sqrt{\lambda_i})(i = 1, \ldots, r)$ i s'ordenen de forma decreixent, tinguent en compte les seves multiplicitats algebraiques:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$$

Amb els valors singulars trobats, podem trobar una factorització especial per a la nostra matriu quadrada singular W de (2.2).Aquesta rep el nom de **descomposició en valors singulars (DVS)**

$$W = USV^\intercal$$

siguent U ortogonal, S diagonal i V ortogonal. Veiem quin és l'origen de cada una d'aquestes matrius i quina és la seva estructura:

- Matriu V:
$$V = (v_1, \ldots, v_n), \tag{2.13}$$
amb $v_1, \ldots, v_n$ els vectors propis de $A^\intercal A$ normalitzats i amb l'ordre donat pels valors singulars.

- Matriu U:
$$U = (u_1, \ldots, u_n), \tag{2.14}$$
amb les col·lumnes $u_1, \cdots, u_r$ calculades mitjançant:
$$u_i = \frac{1}{\sigma_i} W v_i \tag{2.15}$$
Les altres $r+1, \cdots, n$ no tenen interès calcular-les, ja que amb la multiplicació amb la matriu S, donarien files de zeros.

- Matriu S:

$$S = diag(\sigma_1, \cdots, \sigma_r, 0, \cdots, 0)$$

Un cop vista la descomposició en valors singulars (DSV) de la matriu W, la seva matriu pseudoinversa associada W^+ serà:

$$W^+ = V S^+ U^\intercal \tag{2.16}$$

on la matriu S^+ es calcula com:

$$S^+ = diag(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \cdots, \frac{1}{\sigma_r}, 0, \cdots, 0) \tag{2.17}$$

Quin problema apareix ara? Cal trobar quan valen aquests valors propis i els seus vectors propis, que ho farem fent ús del **mètode de Jacobi**.

2.2.2 Mètode de Jacobi

El mètode de Jacobi es un mètode iteratiu per a calcular valors i vectors propis d'una matriu simètrica real amb l'ajuda d'una seqüencia de de *rotacions de Jacobi* [16]:

Definició 2.4. *Una **rotació de Jacobi** és una transformació ortogonal en la qual es transforma en zeros una parella d'elements de fora la diagonal d'una matriu simètrica W. La transformació la representem de forma matricial com J(p,q).*

$$J(p,q) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \tag{2.18}$$

Es quasi idèntica a la matriu identitat excepte en quatre elements d' índexos pp, pq, qp i qq. Diem que $c = \cos\theta$ i $s = \sin\theta$.
Aleshores, si fem l'operació matricial $J(p,q)WJ(p,q)^\intercal$, obtindrem una matriu W' tal que:

$$\begin{aligned} W'_{ii} &= c^2\, W_{ii} - 2\, sc\, W_{ij} + s^2\, W_{jj} \\ W'_{jj} &= s^2\, W_{ii} + 2sc\, W_{ij} + c^2\, W_{jj} \\ W'_{ij} &= W'_{ji} = (c^2 - s^2)\, W_{ij} + sc\, (W_{ii} - W_{jj}) \\ W'_{ik} &= W'_{ki} = c\, W_{ik} - s\, W_{jk} && k \neq i, j \\ W'_{jk} &= W'_{kj} = s\, W_{ik} + c\, W_{jk} && k \neq i, j \\ W'_{kl} &= W_{kl} && k, l \neq i, j \end{aligned} \tag{2.19}$$

Al ser $J(p,q)$ ortogonal, W i W' tindran la mateixa norma de Frobenius $\|\cdot\|_F$ (l'arrel quadrada de la suma dels quadrats de tots els components de la matriu).

Ara bé, s'ha d'escollir la θ per a aconseguir que W'_{ij}=0. D'aquesta manera, W' tindrà una suma de quadrats major en la diagonal.
Igualant (2.19) a 0 i aïllant, obtenim que:

$$\tan(2\theta) = \frac{2W_{ij}}{W_{jj} - Wii}$$

Si resulta que $W_{jj} - W_{ii} = 0$, agafarem $\theta = \frac{\pi}{4}$.

Per tal d'optimitzar al màxim el procés, l'element W_{ij} de fora la diagonal a convertir a zero fem que sigui el de major valor absolut. A aquest W_{ij} se l'anomena *pivot*.
Per a trobar els vectors propis V de (2.13) , definirem V com la matriu resultant de fer:

$$V = IdJ_1J_2\cdots,$$

on $J_1, J_2, \ldots$ son les successives matrius de rotacions de Jacobi. Cada vegada que agreguem una rotació de Jacobi a la matriu V, aquesta pateix següents canvis:

$$\begin{aligned} V_{ip} &= cV_{ip} - sV_{iq} \\ V_{iq} &= sV_{ip} + cV_{iq} \\ V_{ij} &= V_{ij} \qquad\qquad j \neq p, q \end{aligned}$$

El procés iteratiu de Jacobi acaba quan $max|W_{i<j}| < \varepsilon$, és a dir, que el *pivot* és més petit que una tolerància ε.

Un cop trobats els valors i vectors propis, cal fer un cribatge entre ells. Els que són més grans que 0 o que un ε, cal posar-los al principi de la diagonal per a construir la matriu S^+ (2.17).

Si n'haguessin dos o mes d'iguals, cal posar-los junts. També s'ha de verificar que els seus vectors propis siguin ortogonals entre si, per a després normalitzar-los i formar així una base ortonormal.

Amb els vectors propis de V associats a valors propis diferents de 0, els apliquem la fórmula (2.15) i així obtenim la matriu U (2.14).
Trobades les tres matrius, fem el producte i aconseguim W^+ (2.16).

2.3 Extrapolació de la funció de l'esperança de vida en dues variables

L'esquema per a esbrinar el valor de l'esperança de vida per a un punt $\overrightarrow{X^*}=(X_1^*, X_2^*)$ es divideix en dues parts, seguint les pautes donades per [17] :

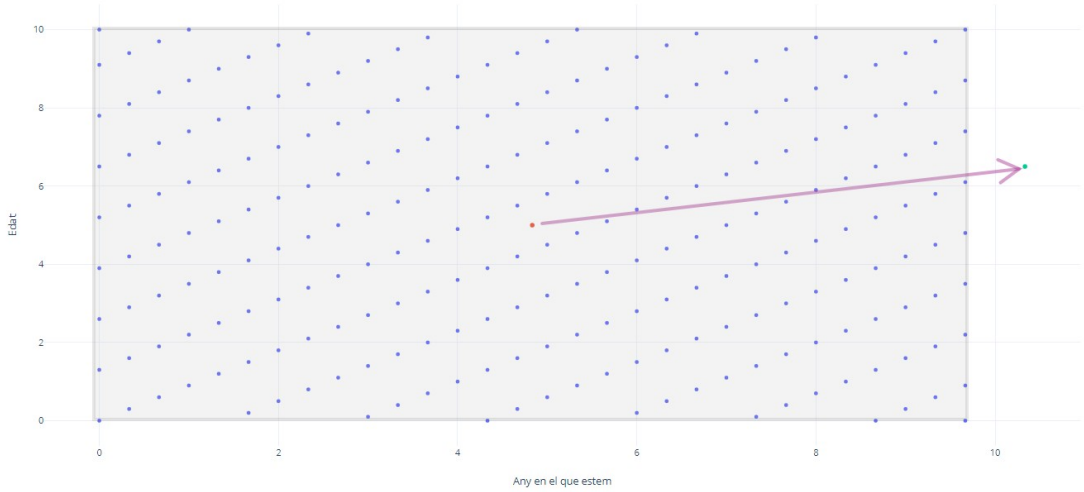


Figura 4: Gràfic del esquema del procés

2.3.1 A l'interior de l'espai de punts

En primer lloc, seleccionem un punt $\overrightarrow{X_0}=(X_{01}, X_{02})$ situat dins de l'espai de punts (representat en la Figura 4 pel rectangle) dels quals sabem el seu valor en $Y = F(\overrightarrow{X})$. A continuació, tracem un segment que uneixi $\overrightarrow{X_0}$ i $\overrightarrow{X^*}$.

$$(1-s)\cdot\overrightarrow{X_0}+s\cdot\overrightarrow{X^*},\quad 0\leq s\leq 1, \tag{2.20}$$

Alhora, aquest segment és dividit en L segments iguals amb nodes:

$$\overrightarrow{S_k}=(S_{k1},S_{k2}),\quad k=1,\ldots,L+1,\quad \overrightarrow{S_{L+1}}=\overrightarrow{X^*} \tag{2.21}$$

Aleshores, en els punts

$$\overrightarrow{S_k}=(S_{k1},S_{k2}),\quad k=1,\ldots,l,\quad l<L+1, \tag{2.22}$$

que pertanyen al segment (2.20) i estan dins la regió delimitada pels punts se'ls hi aplica el esquema d'interpolació establerts en (2.2) i (2.10), que recuperem per a l'ocasió.

$$W=\begin{pmatrix}\rho^2(\overrightarrow{X_1},\overrightarrow{S_k})_\omega & (\overrightarrow{X_1},\overrightarrow{X_2})_\omega & \cdots & (\overrightarrow{X_1},\overrightarrow{X_n})_\omega\\ (\overrightarrow{X_2},\overrightarrow{X_1})_\omega & \rho^2(\overrightarrow{X_2},\overrightarrow{S_k})_\omega & \cdots & (\overrightarrow{X_2},\overrightarrow{X_n})_\omega\\ \cdots & \cdots & \cdots & \cdots\\ (\overrightarrow{X_n},\overrightarrow{X_1})_\omega & (\overrightarrow{X_n},\overrightarrow{X_2})_\omega & \cdots & \rho^2(\overrightarrow{X_n},\overrightarrow{S_k})_\omega\end{pmatrix},Y^*=\frac{(W^{-1}\overrightarrow{Y},\overrightarrow{1})}{(W^{-1}\overrightarrow{1},\overrightarrow{1})} \tag{2.23}$$

En aquest cas, $\overrightarrow{Y}=(Y_1,\ldots,Y_n)$, valors associats als $\overrightarrow{X_1},\ldots,\overrightarrow{X_n}$ de dintre del rectangle. Si W fos singular, caldria aplicar la fórmula amb matrius pseudoinverses descrita en (2.11)

2.3.2 Fora de l'espai de punts:

Els valors $Y_1, \ldots, Y_l$ que s'han obtingut interpolant en cada punt de (2.22) s'utilitzen per a trobar el valor de $Y = F(\overrightarrow{X})$ en els punts restants $\overrightarrow{S_k} = (S_{k1}, S_{k2}), \quad k = l+1, \ldots, L+1$
Per a fer-ho, es segueix el procés iteratiu següent:

Sabem quins són els valors $Y_1, \ldots, Y_l$. Aleshores, el problema per a trobar el valor en Y_{l+1} es redueix a un problema d'interpolació de funcions de varies variables resolt mitjançant un model autoregressiu no linear:

$$\begin{aligned} Y_{m^*+1} &= F(Y_1, \ldots, Y_{m^*}) \\ Y_{m^*+2} &= F(Y_2, \ldots, Y_{m^*+1}) \\ &\vdots \\ Y_l &= F(Y_{n-m^*}, \ldots, Y_{l-1}) \end{aligned} \tag{2.24}$$

Ara,la funció $Y = F(\overrightarrow{X})$ queda definida com $Y = F(y_1, y_2, \ldots, y_{m*})$. En aquesta funció, sabem $l - m^*$ valors en els $l - m^*$ punts :

$$\overrightarrow{X_1} = (Y_1, \ldots, Y_{m^*}), \quad \overrightarrow{X_2} = (Y_2, \ldots, Y_{m^*+1}), \quad \ldots \quad, \overrightarrow{X_{l-m^*}} = (Y_{l-m^*}, \ldots, Y_{l-1})$$

El punt Y_{l+1} queda determinat com el valor interpolat en la nova funció m^*-dimensional F en el punt $\overrightarrow{X^*}$.

$$Y_{l+1} = F(\overrightarrow{X^*}) = \frac{(W^{-1}\overrightarrow{1}, \overrightarrow{Y})}{(W^{-1}\overrightarrow{1}, \overrightarrow{1})}, \quad \overrightarrow{X^*} = (Y_{l-m^*+1}, \ldots, Y_l)^\intercal, \quad \overrightarrow{Y} = (Y_{m^*+1}, \ldots, Y_n)^\intercal \tag{2.25}$$

W^{-1} és la matriu inversa d'una mètrica incerta, de dimensions $(l - m^*) \times (l - m^*)$.
El número m^* determina la dimensió de l'espai de vectors $\overrightarrow{X}$ i el seu valor es troba resolent el problema d'extrems:

$$m^* = \arg\min_{m^*} \|\overrightarrow{Y} - \overrightarrow{Y}_{for}\|, \tag{2.26}$$

Y_{for} surt d'aplicar (2.27) a cadascun dels $Y_{m^*+1}, \ldots, Y_l$ coneguts.
És a dir, per a cada $m^* \in 1, \ldots, l-1$, apareixen $l - m^*$ punts on el valor de la funció es coneguda, que són $\overrightarrow{X_1} = (Y_1, \ldots, Y_{m^*}), \quad \overrightarrow{X_2} = (Y_2, \ldots, Y_{m^*+1}), \quad \ldots \quad, \overrightarrow{X_{l-m^*}} = (Y_{l-m^*}, \ldots, Y_{l-1})$. Definim les matriu $W_i, i = 1, \ldots, l - m^*$ per a cada un d'aquests $\overrightarrow{X_i}$ (s'omplirà la W descrita a (2.2) amb $\overrightarrow{X^*} = \overrightarrow{X_i}$), i després:

$$Y_{for_i} = F(\overrightarrow{X_i}) = \frac{(W_i^{-1}\overrightarrow{1}, \overrightarrow{Y})}{(W_i^{-1}\overrightarrow{1}, \overrightarrow{1})}, \quad \overrightarrow{Y} = (Y_{m^*+1}, \ldots, Y_n)^\intercal, \quad i = 1, \ldots, l - m^* \tag{2.27}$$

Aleshores, un cop conegut el valor Y_{l+1}, l'incorporaríem als $Y_1, \ldots, Y_l$ previs i tornaríem a començar el procés des de (2.24).

3 Aplicació del mètode de l'anàlisi mètric a l'esperança de vida

3.1 Problemes de condicionament de les dades

Un cop feta la presentació del mètode i el perquè de fer-ho, i fent ús de les xifres recaptades per *l'Institut Nacional d'Estadística*, apareixen un seguit de problemes que dificulten aplicar el mètode.

- **Problemàtica amb el número de punts:**

  A la base de dades de l'INE, es disposen les esperances de vida calculades des de l'any 1991 fins el 2020, per a edats que comprenen des dels 0 anys fins als 100 i més anys. En total, tenim 3030 dades $((2020 - 1991) \times 101)$. Al crear la matriu W mencionada prèviament en (2.2), la matriu en qüestió seria de dimensions 3030×3030. Això implicaria que la matriu tingués 9180900 elements.
  Com és lògic, operar amb una matriu de tals dimensions no resulta eficient en cap cas. Per a posar un exemple, amb aquesta matriu W en el primer pas de l'extrapolació (2.23) per a calcular Y^* se l'ha d'invertir, i després fer multiplicacions amb el resultant.

  Per tal de reduïr el número d'operacions sense perdre pel camí gaire informació, baixem el número de punts on la funció es coneguda a un total de 234 elements. El número no surt de manera trivial, ja que si fem $13 \cdot (234 - 1)$ obtenim 3029, que és el total de dades que disposem menys el punt que hem tret en l'operació. La disposició de punts serà la següent

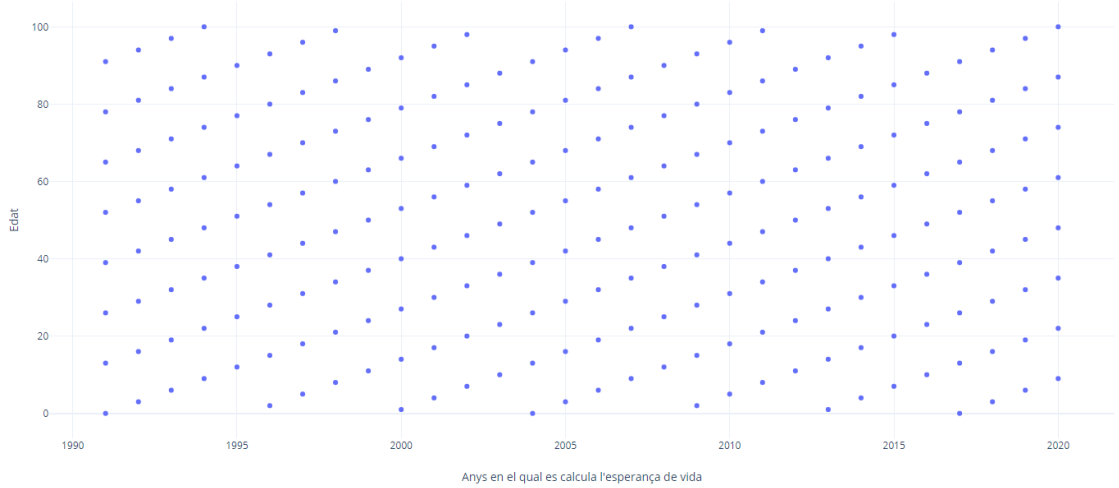


Figura 5: Gràfic de la situació dels 234 punts escollits

Per tal de que aquesta figura sigui rectangular i faciliti així delimitar quina és la regió de punts, afegim els punts 235 i 236. En conclusió, la graella de punts finals és la següent:

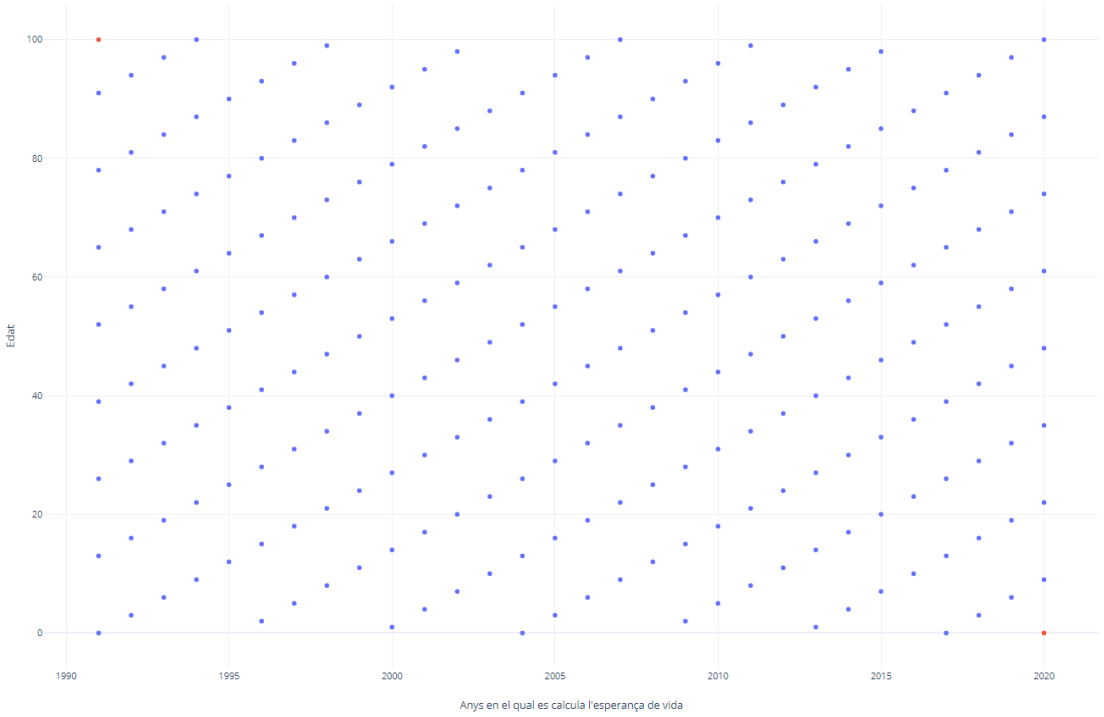


Figura 6: Gràfic de la situació dels 236 punts escollits finals

- **Problemàtica per la distància entre punts**

  En aquest punt, la figura clau torna a ser la matriu W. Ara no ens fixem amb el seu número d'elements, sinó amb els components d'aquesta. I és que si recuperem la fórmula de les components definida en (2.2):

$$\rho^2(\overrightarrow{X_i}, \overrightarrow{X^*})_\omega = \sum_{k=1}^{m} \omega_k \cdot (X_{ik} - X_k^*)^2$$
$$(\overrightarrow{X_i}, \overrightarrow{X_j})_\omega = \sum_{k=1}^{m} \omega_k \cdot (X_{ik} - X_k^*) \cdot (X_{jk} - X_k^*)$$

  El problema es produeix quan certes components són números molt elevats. Posem per cas el valor $(\overrightarrow{X_r}, \overrightarrow{X_s})_\omega$, amb $\overrightarrow{X_r} = (1991, 0)$ i $\overrightarrow{X_s} = (2020, 100)$. El resultat serà 10000. El fet de tenir números tant elevats pot provocar que al fer les aproximacions per matrius pseudoinverses, l'error d'aproximació es multipliqui per un número gran i sigui elevat.

  Per tal de posar-hi remei,farem una translació i una reducció de l'espai on estan situats els punts $\overrightarrow{X_1}, \ldots, \overrightarrow{X_{236}}$.

$$[1991, 2022] \times [0, 100] \overset{translació}{\longrightarrow} [0, 29] \times [0, 100] \overset{reducció}{\longrightarrow} [0, \frac{29}{3}] \times [0, 10]$$

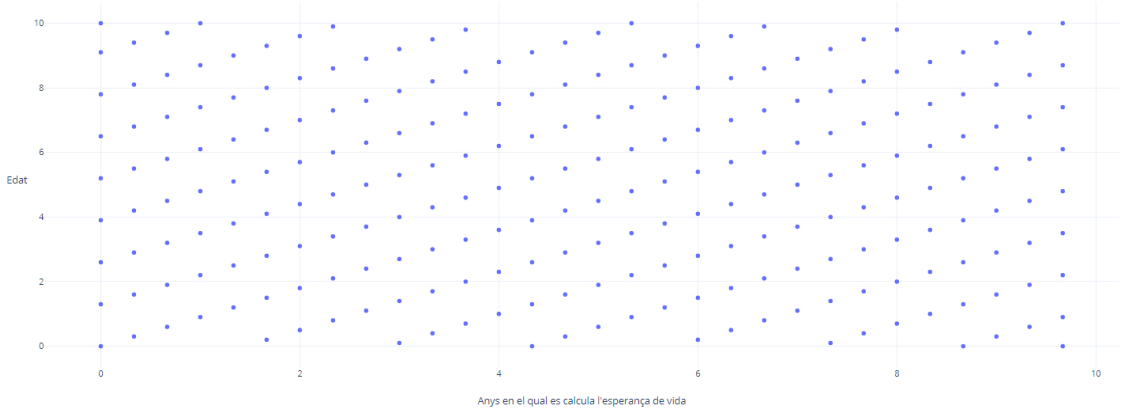


Figura 7: Gràfic de la nova disposició dels punts després de la translació i reducció

3.2 Problemes relacionats amb el mètode

- **Problemes amb la fórmula de les ω**

  Recuperem novament la fórmula de les ω en (2.3),que es calculen per a omplir la matriu W.

$$\omega_k = \frac{|r_k|}{\sum_{k=1}^{m}|r_k|}, k = 1, \ldots, m$$
$$r_k = \frac{cov(Y, X_k)}{\sigma(Y) \cdot \sigma(X_k)}, k = 1, \ldots, m$$
$$cov(Y, X_k) = \frac{1}{n-1} \cdot \sum_{j=1}^{n}(Y_j - \overline{Y})(X_{jk} - \overline{X_k}), k = 1, \ldots, m$$
$$\sigma^2(X_k) = \frac{1}{n-1} \cdot \sum_{j=1}^{n}(X_{jk} - \overline{X_k})^2, k = 1, \ldots, m$$

  El problema de la fórmula arriba si $\sigma^2(X_k)$ o bé $\sigma^2(Y)$ són iguals a zero, que pot succeïr si la mitjana dels valors és igual a tots els valors $j = 1, \ldots, n$.
  En el cas que $n = 1$, que és el cas que ens ocupa, és trivial que $\sigma^2(X_k) = 0$ i $\sigma^2(Y) = 0$.

  Per tant, per a evitar dividir entre zero, quan estem trobant els valors extrapolats fora de l'espai de punts en la subsubsecció 2.3.2 , evitem fer el cas on $m^* = r - 1$, amb r és el número de valors coneguts en la segona fase.

- **Problemes en la restauració dels valors coneguts**

  Per tal d'aplicar el mètode a una funció $Y = F(\overrightarrow{X})$, prèviament s'ha de conéixer el seu valor per a n punts $\overrightarrow{X_1}, \ldots, \overrightarrow{X_n}$. Ara bé, pot passar que al aplicar el procés definit en (2.10) o en (2.11) en aquests n punts, el valor retornat difereixi bastant amb el valor conegut d'avantmà. A aquest valor retornat li direm Y_k^*

  Això succeeix perquè al aplicar el mètode per a cada punt apareix un error d'aproximació. És a dir:

$$Y_k^* = Y_{det_k} + \varepsilon_k, \quad k = 1, \ldots, n, \tag{3.1}$$

$Y_{det} = (Y_{1_{det}}, \ldots, Y_{n_{det}})^\intercal$ és el vector de components deterministes de la estimació del valor en els punts $X_k = (X_{k_1}, \ldots, Xk_m)^\intercal$, $k = 1, \ldots, n$. $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)^\intercal$ és el vector de components caòtiques.

Aleshores, seguint les indicacions donades en [9], per a cada punt $\overrightarrow{X^*}$ estaríem buscant un valor Y^* de manera que es compleixen les equacions (2.4)-(2.6). Afegint-li la nova definició que li donem al valor retornat en (3.1), el sistema (2.6) es tranforma en:

$$\begin{cases} (W\overrightarrow{z}, \overrightarrow{z}) + \alpha \cdot (K_Y \overrightarrow{z}, \overrightarrow{z}) - min\overrightarrow{z} \\ (\overrightarrow{z}, \overrightarrow{1}) = 1, \quad \overrightarrow{1} = (1, \ldots, 1)^\intercal \end{cases} \tag{3.2}$$

La variable $\alpha \geq 0$ rep el el nom de *paràmetre d'allisament*, K_Y és la matriu de covariàncies del vector de components aleatòries $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)^\intercal$, i W és la matriu d'incertesa calculada respetet el punt $\overrightarrow{X^*}$

Resolent novament pel mètode de Lagrange, la solució del problema (3.2) vindrà donada per la igualtat:

$$Y_\alpha = \frac{(W + \alpha \cdot K_Y)^{-1}\overrightarrow{1}, Y)}{(W + \alpha \cdot K_Y)^{-1}\overrightarrow{1}, \overrightarrow{1})} \tag{3.3}$$

El que ens interessa és trobar un valor per al paràmetre α que permeti que els valors resultants d'aplicar (2.10) o bé (2.11) en punts $\overrightarrow{X_1}, \ldots, \overrightarrow{X_n}$ siguin els més propers als valors $Y_1, \ldots, Y_n$.
Suposem que tot $\varepsilon_k \sim N(0, \sigma^2)$. Generarem n components aleatòries, que s'afegiran als valors coneguts i exactes Y_k, per a rebre els valors Y_{error_k}, amb $k = 1, \ldots, n$.
El valor òptim per a α, que li direm α^*, és aquell que compleix que:

$$\alpha^* = \arg\min_\alpha |\frac{1}{n}\sum_{k=1}^{n}(Y_{error_k} - Y_{\alpha_k})^2 - \sigma^2|, \tag{3.4}$$

Cal tenir en compte que per a tots els valors de α s'ha de calcular quan val Y_{α_k}, amb $k = 1, \ldots, 236$. Si no es restringeixen ambdós paràmetres, el número d'operacions a realitzar pot ser considerablement gran. És per això que prendrem un $n_0 = 51$, i α serà de l'estil:

$$\alpha_t = t \cdot 0.2, \quad t = 0, 1, \ldots, 650$$

Per tant, $\alpha \in [0, 130]$. En total, s'evaluarà la funció descrita en (3.3) 33201 vegades $(651 \cdot 51)$.

4 Resultats obtinguts

Subsanats tots els problemes mencionats en els dos apartats anteriors, comencem a operar amb les xifres. Les dades entren en el programa en ordre creixent respecte l'edat de l'individu. És a dir, el primer punt en ser incorporat (tenint en compte les prèvies transformacions descrites en la subsecció 3.1) a la xarxa és el punt $[0, \frac{1991-1991}{10}]$, el següent serà el punt $[0, \frac{2004-1991}{10}]$, etc. D'aquesta manera, s'aconsegueix que el valor Y_k, $k = 1, \ldots, n$ entre un punt i altre no sigui molt elevat. Per tant, els últims punts en ser incorporats són $[\frac{100}{10}, \frac{2007-1991}{3}]$ i $[\frac{100}{10}, \frac{2020-1991}{10}]$.

Per tal de veure quin és el ajustament dels valors coneguts Y_k amb els recuperats, fem ús del concepte estadístic conegut com *error relatiu*. En el nostre cas, diferenciarem en dos errors relatius segons l'aproximació que escollim [9]:

1. **Error relatiu inicial** (Δ_k): Surt de comparar els Y_k amb els valors Y_{error_k}, que són aquells als quals els hi hem afegit un error ε_k. Apareixeran en el cas de que volguem calcular el valor α^* Per tant:

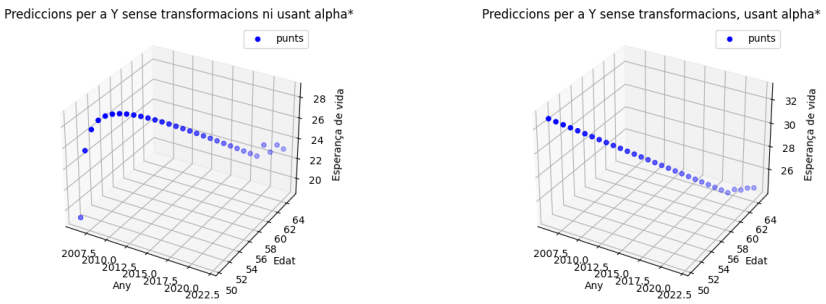
$$\Delta_k = |\frac{Y_k - Y_{error_k}}{Y_k}|, \quad k = 1, \ldots, n \tag{4.1}$$

2. **Error relatiu de recuperació** (δ_k): Surt de comparar els Y_k amb els valors Y_k^* de (3.1) en el cas de no fer servir el criteri d'α^*, o bé amb els $Y_{\alpha_k^*}$ de (3.4) en el cas que sí. Aleshores quedarà de la forma:

$$\delta_k = |\frac{Y_k - Y_k^*}{Y_k}|, \quad o \ \ bé \ \ \delta_k = |\frac{Y_k - Y_{\alpha_k^*}}{Y_k}| \tag{4.2}$$

La funció de l'esperança de vida rebrà a partir d'ara el nom d'$Y = E(X_1, X_2)$, on X_1 és la variable que indica l'any en el qual estem situats, i X_2 l'edat de l'individu del qual volem saber la seva esperança de vida. Per a poder estudiar els resultats seguint el esquema presentat en la subsecció 2.3, suposem que volem calcular l'esperança de vida d'un home de 65 anys a l'any 2022. Per tant, $\overrightarrow{X^*} = (\frac{2022-1991}{3}, \frac{65}{10})$. La L que donarem al programa és $L = 30$.

Els primers resultats surten de fer el procés sense cap transformació. En un cas s'usa el recurs de trobar α^* i en l'altre no s'utilitza.



(a) Predicció sense utilitzar α^* (b) Predicció usant α^*

Figura 8: Càlcul de l'esperança de vida sense fer cap transformació a Y

Queda clar que les maneres de arribar a la predicció final són bén diferents. Mentres que al aplicar α^*, els punts interpolats segueixen una estructura en forma de recta, quan no és té en compte α^* els punts formen una figura en forma de paràbol·la.
A més, el valor que és retornat per al primer punt queda molt lluny del seu valor. El punt en qüestió és el $[2006.032258, 50.483871]$ i val 19.117395. Revisant les taules de l'INE, els valors que envolten el punt són:

$$\begin{array}{lcllcl} [2006{,}50] & = & 29.981015 & [2006{,}51] & = & 29.105624 \\ [2007{,}50] & = & 29.970069 & [2007{,}51] & = & 29.089090 \end{array}$$

Per a veure-ho amb més deteniment, fixem-nos amb els errors relatius inicials per a cada cas.

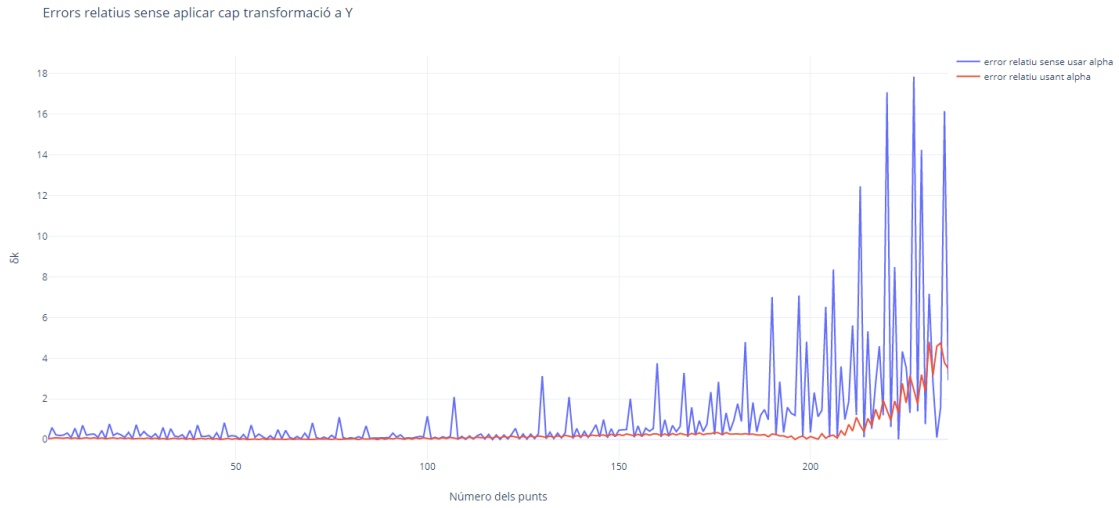


Figura 9: Gràfic dels errors relatius sense aplicar cap transformació a Y

Figura 10: Gràfic dels punts originals i restaurats en la variable Y

És palpable com al arribar als darrers números en ser evaluats (els més petits), aquests errors relatius es disparen. Sense usar el paràmetre de suavització α^*, els errors relatius es disparen fins quasi un 1800%. Amb l'entrada d'aquest, l'error relatiu és com a màxim d'un 400%.

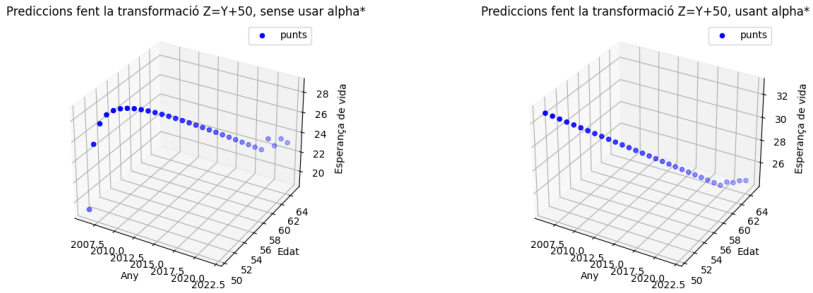
Per a comparar errors relatius entre ells, usem un altre instrument estadístic que és l' *error mitjà de recuperació* (δ)[9]:

$$\delta = \frac{1}{n} \cdot \sum_{k=1}^{n} \delta_k \tag{4.3}$$

El error mitjà de recuperació d'un és $\delta_{Y,sense\ usar\ \alpha^*} = 1.1410$, mentres que per $\delta_{Y,usant\ \alpha^*} = 0.3443$. Hom pot semblar que una bona solució per a evitar aquests errors relatius tant grans pot ser fer que totes les dades siguin grans. És per a això que a $Y = E(X_1, X_2)$ li apliquem una transformació que genera la variable Z, que vé donada per:

$$Z = Y + 50$$

Un cop feta la transformació en el programa, obtenim els següents resultats:



(a) Predicció sense utilitzar α^* (b) Predicció usant α^*

Figura 11: Càlcul de l'esperança de vida sense fent la transformació $Z = Y + 50$

La tendència dels punts interpolats segueix sent exactament la mateixa que el cas previ. No és d'extranyar, ja que els valors són exactament els mateixos.
Si bé els valors interpolats, anem a veure què passa amb els errors relatius:

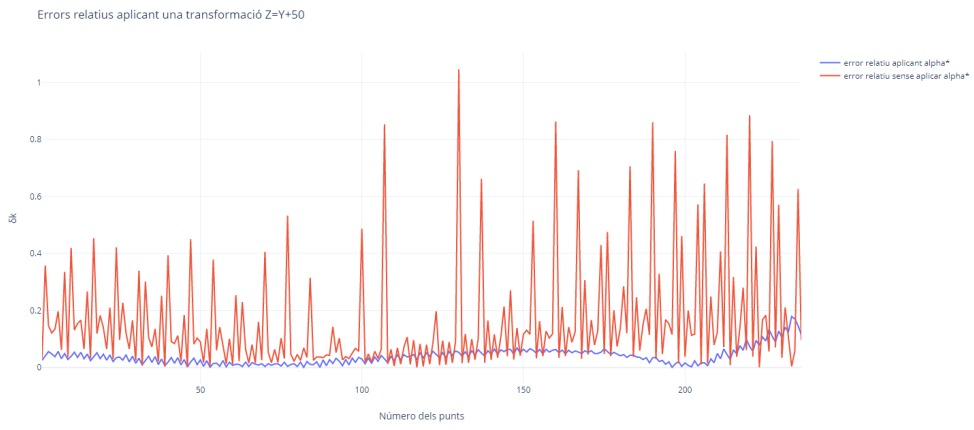


Figura 12: Gràfic dels errors relatius aplicant la transformació $Z = Y + 50$

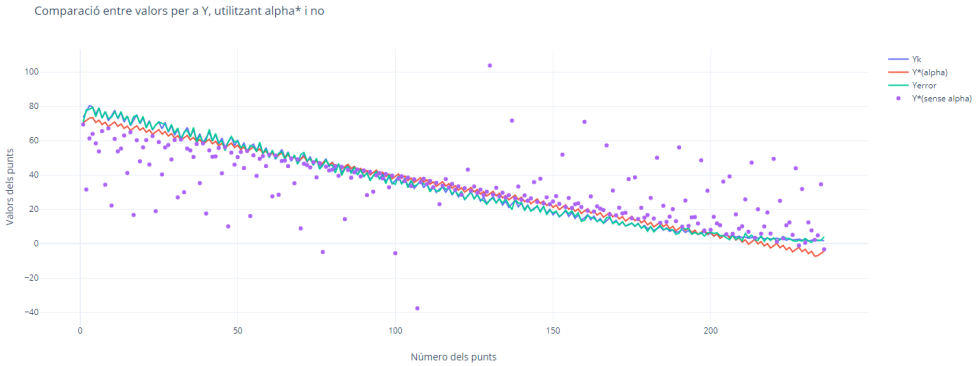
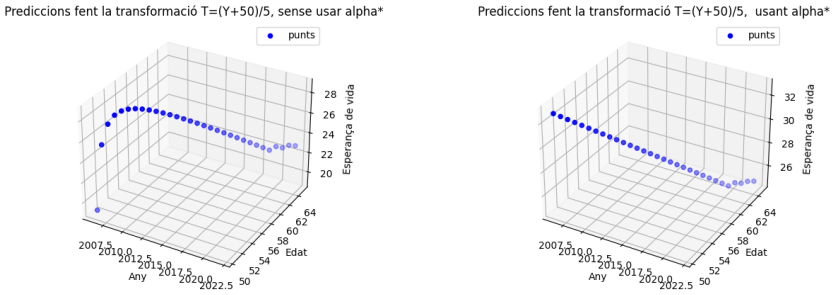


Figura 13: Gràfic dels punts originals i restaurats en la variable $Z = Y + 50$

L'error relatiu baixa ja que, recuperant la fórmula de δ_k descrita en (4.2), el numerador no variaria en excés. No obstant, el denominador sí que ho faria amb 50 unitats. Hi hauria "*més error a repartir*".
En conseqüència, també baixen els errors mitjans de recuperació. Recalculant-los obtenim que $\delta_{Z,sense\ usar\ \alpha^*} = 0.1593020$ i $\delta_{Z,usant\ \alpha^*} = 0.0393$ Per tant, la solució no passa en augmentar els valors de Y i prou. Per tant, provem amb una nova estratègia. Augmentem els valors, per a seguidament fer que estiguin més units. Matemàticament parlant, el resultat seria una transformació T com la següent:

$$T = \frac{Y + 50}{5}$$

Amb la transformació obtenim els valors interpolats següents:



(a) Predicció sense utilitzar α^* (b) Predicció usant α^*

Figura 14: Càlcul de l'esperança de vida sense fent la transformació $T = \frac{T+50}{5}$

En la Figura 14 , comparades amb les Figures 8 i 11, observem com en la darrera part, la de la part de l'extrapolació, els valors no segueixen la mateixa tendència.
En aquest cas, sí que hauríem conseguit un canvi real en els resultats.

Pel que fa als errors relatius i els punts retornats, obtenim:

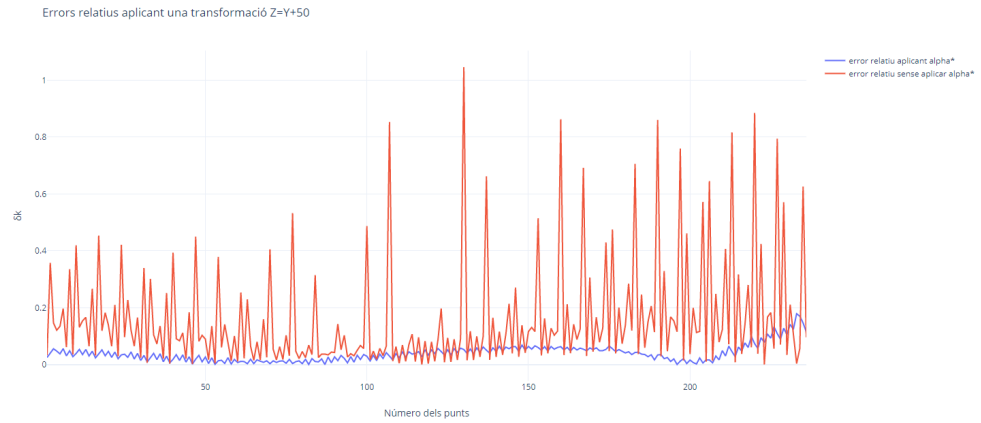


Figura 15: Gràfic dels errors relatius aplicant la transformació $T = \frac{Y+50}{5}$

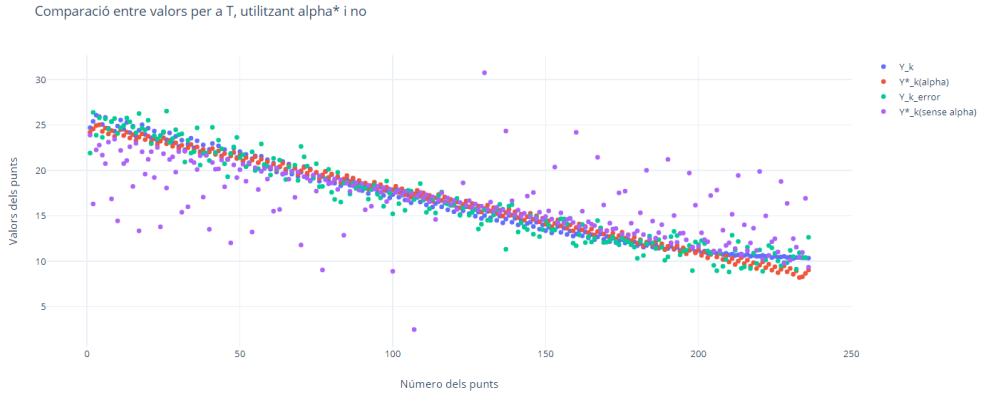


Figura 16: Gràfic dels punts originals i restaurats en la variable $T = \frac{Y+50}{5}$

Notem que, exceptuant diferències mínimes, els valors obtinguts fent la transformació Z són els mateixos que els obtinguts per a la transformació Y. Amb això, els errors de recuperació queden $\delta_{T,sense\ usar\ \alpha^*} = 0.1593020$ i $\delta_{Z,usant\ \alpha^*} = 0.0386$.

Si unim totes les representacions dels punts interpolats en una sola gràfica i prenem com a referència la superfície inicial, adquirim:

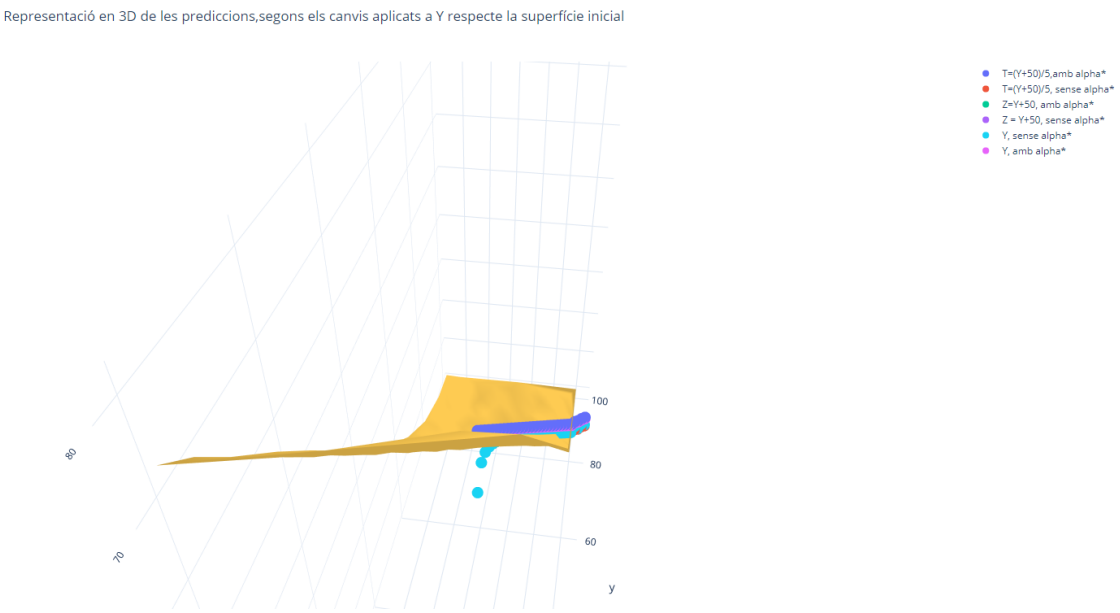


Figura 17: Comparació dels 6 mètodes d'obtenció de les prediccions

Observem de nou que les diferències són mínimes depenent de si s'utilitza el paràmetre de suavització α^* o no.

A continuació, volem comparar quant valen les α^* obtingudes depenent de la variable amb la que operem (Y, Z o T), i les respectives normes associades a cadascuna (vist a (3.4)).

Comparació de α^* i les normes mínimes associades		
Variable	α^*	Norma mínima
Y	44.6	0.00719675
$Z = Y + 50$	44.6	0.00719675
$T = \frac{Y+50}{5}$	5.6	0.01003319

Taula 1: Comparació de α^* i les normes mínimes associades

Com era d'esperar, del fet d'obtenir els mateixos punts interpolats i extrapolats es podia intuïr que les α^* tindrien el mateix valor. També es pot observar que en cap cas s'assoleix el valor màxim que α pot prendre, que és 130.

En últim lloc, fem una comparació entre les Δ_k i les δ_k per a cada variable. La gràcia d'aplicar el paràmetre α^* és que les δ_k siguin menors que les Δ_k en global, amb $k = 1, \ldots, n_0$. Això voldria dir que s'ha millorat el error relatiu inicial que es dóna si als valors exactes Y_k els hi afegim uns errors $\varepsilon_k \sim N(0, \sigma^2)$, $k = 1, \ldots, n$.

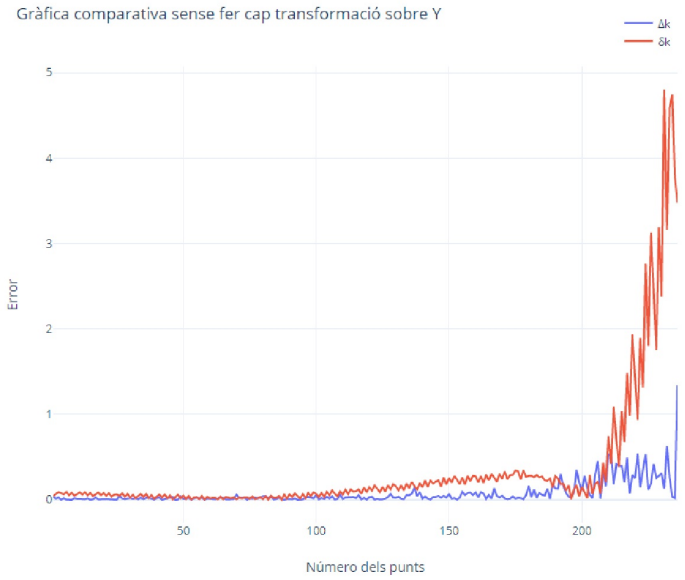


Figura 18: Gràfica de comparació dels errors relatius inicials amb els de recuperació en Y

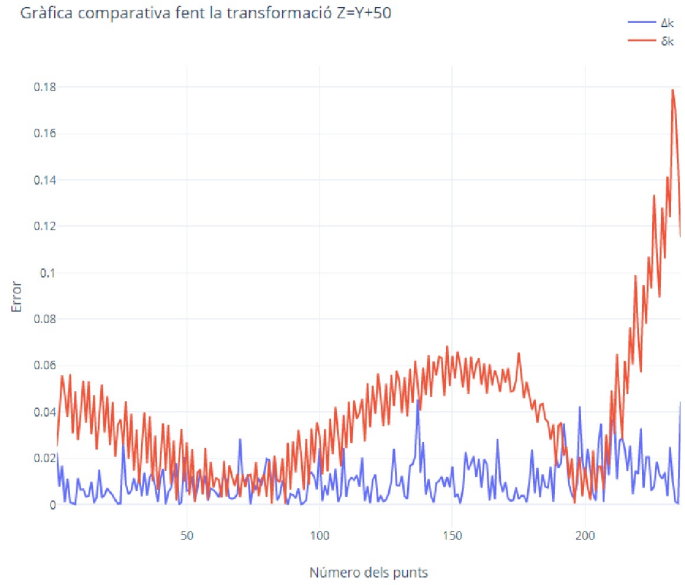


Figura 19: Gràfica de comparació dels errors relatius inicials amb els de recuperació en $Z = Y + 50$

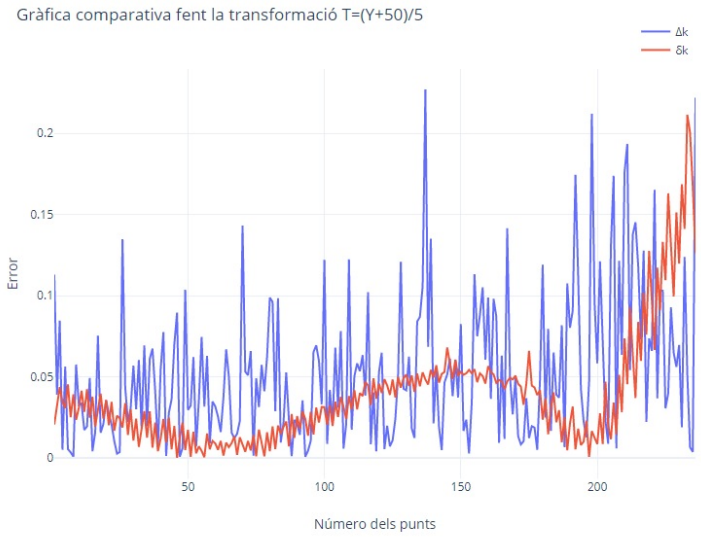


Figura 20: Gràfica de comparació dels errors relatius inicials amb els de recuperació en $T = \frac{Y+50}{5}$

L'única variable que compliria aquest requisit és la variable $T = \frac{Y+50}{5}$. Per reafirmar-ho numèricament, i tal com es menciona en [9], introduïm ara *l'error mitjà inicial*, que segueix en part l'esquema de l'error mitjà de recuperació vist a (4.3):

$$\Delta = \frac{1}{n} \cdot \sum_{k=1}^{n} \Delta_k \tag{4.4}$$

Seguidament, construïm una taula comparativa entre les Δ i δ depenent de la variable amb la que estiguem treballant:

Comparació entre Δ i δ segons la variable amb la que es treballa		
Variable	δ	Δ
Y	1.1410	0.0757
$Z = Y + 50$	0.0393	0.0106
$T = \frac{Y+50}{5}$	0.0386	0.0532

Taula 2: Comparació entre Δ i δ

Els resultats són clars. Per a que funcioni bé el paràmetre de suavització, la imatge de la funció interessa que estiguin en un interval que no hi hagi molta distància entre el valor màxim i el mínim. En el cas de no fer-ho, caldria generar aquests errors de reajustament ε_k amb una variança més gran. Això provocaria que els resultats obtinguts posteriorment a la suavització no siguin tan acurats respecte als valors coneguts inicials.

A mode de conclusió, fem la darrera taula que resumeixi totes les variants estudiades a les quals els hi apliquem el mètode de l'anàlisi mètric, i les seves dades associades.

Comparació entre els mètodes estudiats						
Variables		δ	Δ	α^*	Norma mínima	$F(\overrightarrow{X^*})$
Y	usant α^*	0.34430	0.07568	44.6	0.0071967	24.528417
	no usant α^*	1.14105				23.142421
Z	usant α^*	0.03935	0.01063	44.6	0.0071967	24.528417
	no usant α^*	0.15930				23.142421
T	usant α^*	0.0386	0.0532	5.6	0.010033	24.788574
	no usant α^*	0.15930				22.88199

Taula 3: Comparació entre tots els mètodes estudiats

Tal i com poder observar en la Taula 3, la variable amb la qual extrapolaríem els valors per a conèixer l'esperança de vida d'una persona seria T. En el cas de no utilitzar α^*, és la que dóna un valor bastant acurat amb les xifres prèviament conegudes en [18].

Pel que fa al valor extrapolat fent ús d' α^*, és la extrapolació que ens dona un valor més gran de les sis. Tot i això, el fet que $\delta < \Delta$ i que ens dongui la δ més petita de totes, són indicadors de què és un valor coherent estadísticament parlant.

El fet que la norma mínima en Y i Z sigui menor que en T és irrellevant, ja que al final les variables només serien acurades pels últims darrers 51 punts, no en els altres 185 restants.

5 Explicació del programa

En el programa que trobareu a l'apèndix, trobareu el codi en lleguatge C usat per a obtenir els resultats finals. Com anotar comentaris al marge del codi pot quedar barroer, dedicarem aquest petit apartat a parlar detalladament sobre el funcionament del programa i la seva estructura.

- `main`:

  En les primeres línies, es declaren totes les funcions i les llibres que s'utilitzaran més endavant. També definim l'estructura `struct valor`, que s'utilitzarà per a guardar les diferents components que conformen una funció multivariable. És a dir, si la funció és de l'estil:

  $$Y_i = F(X_{i_0}, X_{i_1}, \ldots, X_{i_n}),$$

  a la variable `y` s'hi guardarà el valor Y_i, i en el vector `x[3100]` el punt que té com imatge Y_i.

  Declarem ara totes les variables que s'utilitzaran en el `main`, tenint en compte el número de punts i valors per paràmetres que prèviament s'han fixat a 3.1 i 3.2 (n_0, σ, n i X_0). Obrim els fitxers i comencem a llegir dades, que guardarem en el vector de tipus `valor` que anomenarem `matriu`.

  Com volem entrar els valors de les esperances de forma ordenada, els dos punts que afegim a la Figura 6 s'han de posar en l'ordre que toca. Per tant, el bucle de lectura quedarà fragmentat en 3 parts. L'ordre en que s'ingressen les dades és edat a edat, i de 13 anys en 13 anys.

  A `matriu[i].x[0]` guardem l'any en el que estem calculant l'esperança de vida, dividint-lo entre 3.
  A `matriu[i].x[1]` guardarem de l'edat de l'individu al qual se li calcula l'esperança de vida, dividint-la entre 10. Ho fem per allò explicat a 3.2.

  Seguidament, creem el vector Y_{error} mencionat a (4.1) i guardem els valors al vector `Y`. A continuació, entrem en el bucle per a trobar, en el cas de voler-ho, un valor α^* que minimitzi l'equació donada en (3.4).

  Provem quina $\alpha \in [0, 130]$ fa que, pels $n_0 = 51$ darrers punts, que són els de menor valor, ajusta millor la funció d'interpolació. Si estem en el cas de $\alpha = 0$ (inici) o bé la norma mínima prèvia és més que la norma calculada, guardem el valor de la `norma` en la variable `NormaMin`. El mateix fem amb la variable `alphaestrella`, que ara valdrà `alpha`.
  Acabem el bucle, i amb l'`alphaestrella` trobada, recuperem els 256 valors coneguts coneguts prèviament.

  S'inicia en aquest moment la part d'extrapolació de la funció de l'esperança de vida (apartat). Comencem primerament demanant quin és el punt del qual es vol saber quin és el seu valor ($\overrightarrow{X^*}$), el qual es condiciona adequadament per a poder operar amb ell. Aquest el guardarem en el vector `X`.

  Seguidament, demanem la dimensió de L, per a crear un punter de dimensió $L+1$ de tipus `valor` que anomenarem `S`. Aquest servirà per a guardar les coordenades dels punts $\overrightarrow{S_k} = (S_{k_1}, S_{k_2}), \quad k = 1, \ldots, L+1, \quad \overrightarrow{S_0} = \overrightarrow{X_0}, \quad \overrightarrow{S_{L+1}} = \overrightarrow{X^*}$.

Els punts $\overrightarrow{S_k} = (S_{k_1}, S_{k_2})$ estan construïts de la forma següent:

$$S_{(k+1)_1} = S_{k_1} + \Delta_{X_1}, \quad k = 1, \ldots, L+1$$
$$S_{(k+1)_2} = S_{k_2} + \Delta_{X_2}, \quad k = 1, \ldots, L+1$$

Els valors Δ_{X_1} i Δ_{X_2}, que en el programa els hi diem `incx` i `incy` respoectivament, surten de:

$$\Delta_{X_1} = \frac{S_{(L+1)_1} - S_{0_1}}{L+1}$$
$$\Delta_{X_2} = \frac{S_{(L+1)_2} - S_{0_2}}{L+1}$$

Ara si, entrem en la primera etapa del procés d'extrpolació, representada en el programa pel `while` de la línia 198. Mentres s'estigui dintre de la regió de punts, es faran els càlculs estipulats en 2.3.1.

Un cop evaluats tots els $k = 1, \ldots, l$ que estan dins de l'espai de punts, comença la segona etapa del procés. S'han de trobar extrapolant les esperances de vida per als punts $\overrightarrow{S_k} = (S_{k_1}, S_{k_2})$, amb $k = l+1, \ldots, L+1$.

Per començar, creem un punter de tipus `struct valor`, al que anomenarem `matriu2`, i que serà de dimensió k. Aleshores anem provant per a cada $j = 1, \ldots, l-2$ els processos descrits a 2.3.2 per a trobar el valor m^* òptim. Per tant, per a cada j, les variables que participen en el procés iteratiu aniran variant de dimensions de la forma següent:

- `matriu2`: Tindrà dimensió $l - j$, i els vectors `x` que incorpora cada component dimensió j. S'omplirà tal i com diu l'esquema de 2.24.
- ω: Com el vector `x` té dimensió j, ella també.
- Y_{for}: Tindrà dimensió $l - j$.

Un cop establerts els punts i inicialitzades les variables de la llista prèvia, omplim les matrius W i $W^\intercal$ i procedim a fer Jacobi, exceptuant el cas on $j = l - 2$. En aquest cas, la funció `jacobi` donarà error, degut als que tots els elements excepte un seran igual a 0 i no funcionarà bé. És per això que fem el càlcul de la pseudoinversa de la matriu $W \cdot W^\intercal$ invertint el valor que és diferent a 0.

Ara, fem els càlculs corresponents de 2.27 que guardarem a `Yfor`, i resolem el problema d'extrems estipulat a 2.26. La variable `norma` guardarà la diferència entre Y (situats en membre `y` de `matriu2`) i Y_{for}. Si estem en el cas de $normaMin = 0$ (inici) o bé la norma mínima prèvia és més que la norma calculada, guardem el valor de la `norma` en la variable `NormaMin`. El mateix fem amb la variable `m` (a 2.26 és la variable m^*, que ara valdrà `j`.

Avaluats totes les $j = 1, \ldots, l-2$, fem el càlcul per saber quant valdrà la funció en $\overrightarrow{S_{l+1}}$ utilitzant la `m` trobada i tenint en compte l'equació vista a 2.25. Ara, aquest valor Y_{l+1} l'afegirem al vector `matriu` dels valors calculats en $\overrightarrow{S_1}, \ldots, \overrightarrow{S_l}$ i repetirem aquest procés per a $l+2, \ldots, L+1$. Al augmentar el número, també variaran les dimensions de la llista prèvia.

Un cop trobats tots els valors per als $\overrightarrow{S_k}$, $k = 1, \ldots, L+1$, els imprimim en el fitxer `Sortida.txt` (7).

- `omplirW`

  Utilitzem la definició de (2.2) per a crear la matriu W. Com es tracta d'una matriu simètrica, només cal fer càlculs per la meitat de la matriu.A l'altre meitat de la matriu, s'igualen els elements amb el seu element simètric. La funció, al acabar, albergarà els canvis realitzats a cada component de W.

- `ompliromega`

  Seguint les directrius marcades per les equacions (2.3) i següents, fem el càlcul del vector ω, de dimensió m. Creem el vector `mitjana` amb dimensió $m+1$. A `mitjana[0]` guardarem la mitjana de y, mentres que a `mitjana[1]`,...,`mitjana[n]` els valors de les mitjanes de X_1,...,X_n.

  Més tard, es calcula un a un els factors del vector `r` mitjançant les covariàncies i variàncies respectives, per acabar aconseguint el vector de ω. Aquest últim és que el que retornarà la funció un cop finalitzada.

- `jacobi`

  A 2.2.2 s'explica amb escreix el mètode de Jacobi. La matriu b albergarà a la seva diagonal principal els valors propis en la seva diagonal. La matriu o guardarà els respectius vectors propis associats a cada un dels valors propis que hi ha a b, mantenint el ordre. És a dir, el valor propi λ_i situat en la posició $b[i][i]$ té com a valor propi associat els valors situats en la col·lumna i de la matriu o.

  Un cop trobades b i o i ordenats els seus elements, trobem la matriu u. Ara, ja tenim les matrius necessàries per a trobar la pseudoinversa de la matriu W. La pseudoinversa es guarda en la seva variable homònima `pseudoinversa`, que és el que es retorna un cop acaba la funció.

  En el cas de que es volgués repasar que el càlcul de la pseudoinversa és correcte, es pot afegir aquesta part de codi abans d'acabar la funció i comprovar que els valors de o són els mateixos que els de W.
  Per fer la comprovació es fa ús de la primera propietat de la definició 2.1.

```
printf("\n\nComprovacio que la pseudoinversa es bona\n");
for (l=0; l<n;l++){
    for(j=0;j<n;j++){
        u[l][j]=0;
        for(k=0;k<n;k++){
            u[l][j]+=W[l][k]*pseudoinversa[k][j];
        }
    }
}
for (l=0; l<n;l++){
    for(j=0;j<n;j++){
        o[l][j]=0;
        for(k=0;k<n;k++){
            o[l][j]+=u[l][k]*W[k][j];
        }
    }
}
for (k = 0; k < n; k = k + 1) {
    for (j = 0; j < n; j = j + 1) {
```

```
            printf("%lf ", o[k][j]);
            }
        printf("\n");
    }
```

- `productetransposada` La funció torna el producte d'matriu amb la seva matriu transposada d'una matriu simètrica, que facilita molt el càlcul. La matriu resultant de l'operació és la matriu `result`, que després es retorna quan finalitza el procés iteratiu.

- `solucio`

  Depenent de si estem en un cas on la matriu W s'hagi pogut invertir o no, seguirem l'equació (2.10) o la equació (2.11) respectivament. La solució que es torna està dins de la variable `sol`.

- `inv`

  En aquesta funció fem ús d'una matriu ampliada de dimensions $n \times 2n$, a la qual li direm `matrix`. Un cop declarada, fem ús del mètode de reducció gaussiana per a trobar la matriu inversa d'una matriu. A continuació, i per a fer-ho més visual, posem un exemple del seu funcionament:

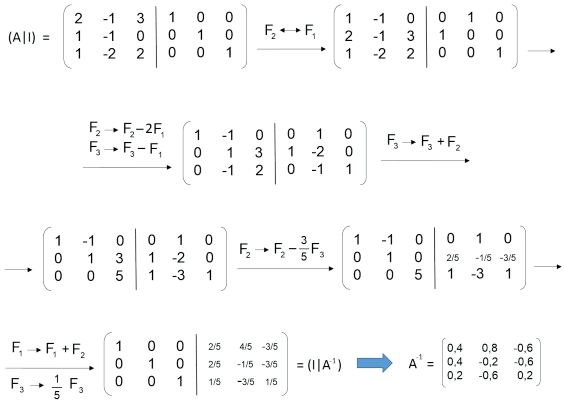


Figura 21: Exemple del càlcul d'una matriu inversa mitjançant eliminació gaussiana amb pivotatge

  No obstant, al treballar amb moltes matrius que no tenen inversa, cal posar condicionals `if` per a comprovar que no es faci un càlcul erroni, com per exemple dividir entre 0. Per això, si el valor màxim d'una col·lumna és més petita que una tolerància, retornem directament la matriu la qual volíem calcular la seva inversa.

  La matriu inversa es retornarà al `main` amb la variable `result`.

- `intercambiarFiles`

  Fent ús de punters, es canvien la posició de files de una matriu. Aquesta funció s'utilitza en la funció `inv` anterior, quant el valor màxim d'una col·lumna no es troba en el *pivot*.

6 Conclusions

Com s'ha pogut comprovar al llarg del treball, per tal de què el mètode funcioni correctament és necessari fer una sèrie de condicionaments a totes les dades que disposem. Un cop fet, l'anàlisi mètric es presenta com una molt bona eina per a tot tipus de problemes de funcions on s'hagi de interpolar o extrapolar funcions en diverses variables.

Els resultats obtinguts poden variar, com podem veure a 3, segons com les transformacions que pugui patir la funció original. En aquesta secció també juga un paper molt important el factor de suavització α^*, ja que també proporcionarà resultats diferents.

No obstant, en el camp de l'estadística també tenim altres mètodes per a poder trobar vincles entre variables, com ara el model de regressió lineal múltiple ([19], [20]). La seva aplicació pot resultar més fàcil d'aplicar i és la més extesa per a resoldre aquest tipus de problemes. D'aquesta manera, queda oberta al lector d'aquest treball aquesta futura línea de recerca. Comparar els resultats obtinguts amb l'anàlisi mètric amb altres tipus de models estadístics.

7 Codi en Llenguatge C

```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#define CERO 1e-8
struct valor {
  double y;
  double x[3100];
};
double **jacobi ( double **, double**, int);
void omplirW(double **, struct valor *, double *, double *, int, int);
void ompliromega(double *, struct valor *, int, int);
double solucio(double **, struct valor *, int);
double **productetransposada(double **,int);
void intercambiarFilas( double **, int , int , int  );
double **inv (double **, int );

int main() {
    int i, j, k, m, h,s,r,n0=51, n1, n2, L, numpunts = 236;
    double **W,alphaestrella,alpha, *X, *omega, X0[2], incx,incy, d, var
    =1.44,norma, normaMin, *Yfor, **Wt, *Y, *Ydet;
    struct valor *matriu, *S, *matriu2;
    FILE *fout, *fin, *foutaux,*foutreals, *fouterror,*foutrestaurats;
    X0[0] = (double)14.5/3;
    X0[1] = 5.;
    fin = fopen("entrada2.txt", "r");
    fout = fopen("Valors predits.txt", "w");
    foutaux = fopen("Alpha estrella i norma m nima", "w");
    foutreals = fopen("Valors Reals", "w");
    fouterror = fopen("Valors amb error", "w");
    foutrestaurats = fopen("Valors restaurats", "w");
    if (fout == NULL || foutaux == NULL || fin == NULL) {
        printf("Error al crear fitxer Euler o Practica2.txt\n");
        exit(1);
    }
    fscanf(fin, "%d", &n1);
    fscanf(fin, "%d", &n2);
    X = (double *)calloc(numpunts, sizeof(double));
    Y = (double *)calloc(numpunts, sizeof(double));
    Ydet = (double *)calloc(numpunts, sizeof(double));
    matriu = (struct valor *)malloc(numpunts * sizeof(struct valor));
    matriu2 = (struct valor *)malloc(numpunts * sizeof(struct valor));
    omega = (double *)calloc(2, sizeof(double));
    W = (double **)malloc(numpunts * sizeof(double *));
    for (i = 0; i <3; i++) {
        W[i] = (double *)malloc(numpunts * sizeof(double));
        fscanf(fin, "%lf", &matriu[i].y);
        matriu[i].y=(matriu[i].y);
        matriu[i].x[0] = (i*13) % n1;
        matriu[i].x[1] = (i*13) / (n1);
        matriu[i].x[0] = matriu[i].x[0] / 3;
        matriu[i].x[1] =  matriu[i].x[1] / 10 ;
        for(j=1;j<13;j++){
            fscanf(fin, "%lf", &X[i]);
        }
    }
    matriu[3].x[0]=9.6666666666666666;
    matriu[3].x[1]=0;
    matriu[3].y=(79.586509);
```

```
58      for (i = 4; i <numpunts-4; i++) {
59          W[i] = (double *)malloc(numpunts * sizeof(double));
60          fscanf(fin, "%lf", &matriu[i].y);
61          matriu[i].y=(matriu[i].y);
62          matriu[i].x[0] = ((i-1)*13) % n1;
63          matriu[i].x[1] = ((i-1)*13) / (n1);
64          matriu[i].x[0] = matriu[i].x[0] / 3;
65          matriu[i].x[1] =  matriu[i].x[1] / 10 ;
66          for(j=1;j<13;j++){
67              fscanf(fin, "%lf", &X[i]);
68          }
69      }
70      matriu[numpunts-4].x[0]=0;
71      matriu[numpunts-4].x[1]=10.;
72      matriu[numpunts-4].y=(2.029861);
73      for (i = numpunts-3; i < numpunts; i++) {
74          W[i] = (double *)malloc(numpunts * sizeof(double));
75          fscanf(fin, "%lf", &matriu[i].y);
76          matriu[i].y=(matriu[i].y);
77          matriu[i].x[0] = ((i-2)*13) % n1;
78          matriu[i].x[1] = ((i-2)*13) / (n1);
79          matriu[i].x[0] = matriu[i].x[0] / 3;
80          matriu[i].x[1] =  matriu[i].x[1] / 10 ;
81          for(j=1;j<13;j++){
82              fscanf(fin, "%lf", &X[i]);
83          }
84      }
85      ompliromega(omega, matriu, numpunts, 2);
86      fclose(fin);
87      fin = fopen("random errors.txt", "r");
88      for(i=0;i<numpunts;i++){
89          fprintf(foutreals,"[%lf, %lf, %lf]\n",matriu[i].x[0],matriu[i].x[1],
        matriu[i].y);
90          fscanf(fin,"%lf",&Y[i]);
91          Y[i]=((Y[i]+(matriu[i].y)));
92          fprintf(fouterror,"[%lf, %lf, %lf]\n",matriu[i].x[0],matriu[i].x[1],
        Y[i]);
93      }
94      for(i=0;i<=650;i++){
95          norma = 0;
96          alpha=i*0.2;
97          for(j=0;j<n0;j++){
98              W = (double **)calloc(n0 , sizeof(double *));
99              Wt=(double **)malloc(n0*sizeof(double*));
100             for (k = 0; k < n0; k++) {
101                 W[k] = (double *)calloc(n0 , sizeof(double));
102             }
103             omplirW(W,matriu,matriu[numpunts-j-1].x,omega,n0,2);
104             for(k=0;k<n0;k++){
105                 W[k][k]=W[k][k]+alpha*var;
106                 Wt[k]=(double *)malloc(n0*sizeof(double));
107                 for (j=0;j<n0;j++){
108                     Wt[k][j]=W[k][j];
109                 }
110             }
111             Wt=inv(Wt,n0);
112             if(fabs(W[0][0]-Wt[0][0])<1e-6){
113                 Wt=productetransposada(W, n0);
114                 Wt=jacobi(W,Wt,n0);
115             }
116             Ydet[j] = solucio(Wt, matriu, n0);
```

```
117             norma+=(Y[j]-Ydet[j])*(Y[j]-Ydet[j]);
118             for (k = 0; k <n0; k++) {
119                 free (W[k]);
120                 free (Wt[k]);
121             }
122             free(W);
123             free (Wt);
124         }
125         norma=(double) (norma)/n0;
126         norma=fabs(norma-var);
127         if(i==0 ||norma<normaMin){
128             alphaestrella=alpha;
129             normaMin=norma;
130         }
131     }
132     fprintf(foutaux,"\nalphaestrella=%lf\n norma min %lf",alphaestrella,
    normaMin);
133     for(i=0;i<numpunts;i++){
134         W = (double **)calloc(numpunts , sizeof(double *));
135         for (k = 0; k < numpunts; k++) {
136             W[k] = (double *)calloc(numpunts , sizeof(double));
137         }
138         omplirW(W,matriu,matriu[i].x,omega,numpunts,2);
139         Wt=(double **)calloc(numpunts,sizeof(double*));
140         for(k=0;k<numpunts;k++){
141             Wt[k]=(double *)calloc(numpunts,sizeof(double));
142         }
143         for(k=0;k<numpunts;k++){
144             W[k][k]=W[k][k];
145             for(j=0;j<numpunts;j++){
146                 Wt[k][j]=W[k][j];
147             }
148         }
149         Wt=inv(Wt,numpunts);
150         if(fabs(W[0][0]-Wt[0][0])<1e-8){
151             Wt=productetransposada(W, numpunts);
152             Wt=jacobi(W,Wt,numpunts);
153         }
154         matriu2[i].y = solucio(Wt, matriu, numpunts);
155         fprintf(foutrestaurats,"[%lf, %lf, %lf]\n",matriu[i].x[0],matriu[i].
    x[1],matriu2[i].y);
156         for (k=0;k<numpunts;k++){
157             free (W[k]);
158             free (Wt[k]);
159         }
160         free(W);
161         free (Wt);
162     }
163     printf("\nDonem el punt a extrapolar: (Any en vigor, edat) ");
164     scanf("%lf %lf", &X[0], &X[1]);
165     X[0] = (double)(X[0] - 1991)/3;
166     X[1] = (double)X[1]/10;
167     printf("\nTamany de L? ");
168     scanf("%d", &L);
169     incx = (double)(X[0] - X0[0]) / (L+1);
170     incy = (double)(X[1] - X0[1]) / (L+1);
171     S = (struct valor *)malloc((L + 2) * sizeof(struct valor));
172     S[0].x[0] = X0[0];
173     S[0].x[1] = X0[1];
174     for (i = 0; i <= L; i++) {
175         S[i + 1].x[0] = S[i].x[0] + incx;
```

```
176         S[i + 1].x[1] = S[i].x[1] + incy;
177     }
178     i = 1;
179     omplliromega(omega, matriu, numpunts, 2);
180     while (S[i].x[0]*3 < (n1-1)  && S[i].x[0] > 0 && S[i].x[1]*10 < (n2-1)
    && S[i].x[1] > 0 && i <= L+1) {
181         W = (double **)calloc(numpunts , sizeof(double *));
182         for (k = 0; k < numpunts; k++) {
183             W[k] = (double *)calloc(numpunts , sizeof(double));
184         }
185         omplirW(W,matriu,S[i].x,omega,numpunts,2);
186         Wt=(double **)calloc(numpunts,sizeof(double*));
187         for(k=0;k<numpunts;k++){
188             Wt[k]=(double *)calloc(numpunts,sizeof(double));
189             W[k][k]=W[k][k];
190             for(j=0;j<numpunts;j++){
191                 Wt[k][j]=W[k][j];
192             }
193         }
194         Wt=inv(Wt,numpunts);
195         if(fabs(W[0][0]-Wt[0][0])<1e-8){
196             Wt=productetransposada(W, numpunts);
197             Wt=jacobi(W,Wt,numpunts);
198         }
199         S[i].y = solucio(Wt, matriu, numpunts);
200         i++;
201         for (k = 0; k < numpunts; k++) {
202             free (W[k]);
203             free (Wt[k]);
204         }
205         free(W);
206         free (Wt);
207     }
208     i--;
209     free(omega);
210     for (; i <=L; i++) {
211         normaMin = 0;
212         m = 1;
213         matriu2 = (struct valor *)malloc((i) * sizeof(struct valor));
214         for (j = 1; j < i-1; j++) {
215             norma = 0;
216             for (k = 0; k < i - j; k++) {
217                 matriu2[k].y = S[k + j + 1].y;
218                 for (h = 0; h < j; h++) {
219                     matriu2[k].x[h] = S[k + h + 1].y;
220                 }
221             }
222             omega = (double *)calloc(j, sizeof(double));
223             omplliromega(omega, matriu2, i - j, j);
224             Yfor = (double *)calloc(i - j, sizeof(double));
225             for (k = 0; k < i-j ; k++) {
226                 W = (double **)calloc(i-j , sizeof(double *));
227                 Wt=(double **)calloc(i-j,sizeof(double*));
228                 for (r = 0; r < i-j; r++) {
229                     W[r] = (double *)calloc(i-j , sizeof(double));
230                     Wt[r]=(double *)calloc(i-j,sizeof(double));
231                 }
232                 omplirW(W, matriu2, matriu2[k].x, omega, i - j, j);
233                 if(i-j!=2){
234                     Wt=productetransposada(W, i-j);
235                     Wt=jacobi(W,Wt,i-j);
```

```
236                 }else{
237                     for(r=0;r<2;r++){
238                         if(fabs(W[r][r])>1e-8){
239                             Wt[r][r]=(double)1/W[r][r];
240                         }
241                     }
242                 }
243                 Yfor[k] = solucio(Wt, matriu2, i - j);
244                 Yfor[k] = (Yfor[k] - matriu2[k].y) * (Yfor[k] - matriu2[k].y
    );
245                 norma = norma + sqrt(Yfor[k]);
246                 for (r=0;r<i-j;r++){
247                     free (W[r]);
248                     free (Wt[r]);
249                 }
250                 free (W);
251                 free(Wt);
252             }
253             if (norma < normaMin || normaMin == 0) {
254                 normaMin = norma;
255                 m = j;
256             }
257             free(Yfor);
258             free(omega);
259         }
260         omega = (double *)calloc(m, sizeof(double));
261         X = (double *)calloc( m, sizeof(double));
262         for (k = 0; k < i - m; k++) {
263             matriu2[k].y = S[k + m + 1].y;
264             for (h = 0; h < m; h++) {
265                 matriu2[k].x[h] = S[k + h + 1].y;
266             }
267         }
268         for(k=0;k<m;k++){
269             X[k] = S[k+1+i-m].y;
270         }
271         W = (double **)calloc(i-m , sizeof(double *));
272         Wt = (double **)calloc(i-m , sizeof(double *));
273         for (r = 0; r < i-m; r++) {
274             W[r] = (double *)calloc(i-m , sizeof(double));
275             Wt[r] = (double *)calloc(i-m , sizeof(double));
276         }
277         ompliromega(omega, matriu2, i - m, m);
278         omplirW(W, matriu2, X, omega, i - m, m);
279         Wt=productetransposada(W, i-m);
280         Wt=jacobi(W,Wt,i-m);
281         S[i+1].y = solucio(W, matriu2, i - m);
282         for (r = 0; r < i-m; r++) {
283                 free (W[r]);
284                 free (Wt[r]);
285         }
286         free(W);
287         free(Wt);
288     }
289     for (i = 0; i <= L+1; i++) {
290         fprintf(fout, " [%lf , %lf , %lf] , \n", S[i].x[0]*3+1991,S[i].x
    [1]*10, (S[i].y));
291     }
292     fclose(foutaux);
293     fclose(fout);
294     fclose(fin);
```

```
}
void omplirW(double **W, struct valor *M, double *X, double *omega, int N,
    int m) {
    int i, j, k;
    for (i = 0; i < N; i++) {
        for (j = 0; j < i; j++) {
            W[i][j] = W[j][i];
        }
        for (j = i; j < N; j++) {
            W[i][j] = 0;
            for (k = 0; k < m; k++) {
                W[i][j] += omega[k] * (M[i].x[k] - X[k]) * (M[j].x[k] - X[k
    ]);
            }
        }
    }
}
void omplir omega(double *omega, struct valor *M, int N, int m) {
    int i, j, k;
    double cov, varx, *r, *mitjana, vary = 0, total = 0;
    r = (double *)calloc((m), sizeof(double));
    mitjana = (double *)calloc((m + 1), sizeof(double));
    for (i = 0; i < N; i++) {
        mitjana[0] += (M[i].y);
    }
    mitjana[0]=(double)(mitjana[0])/N;
    for (j = 0; j < m; j++) {
        for (i = 0; i < N; i++) {
            mitjana[j + 1] +=  (M[i].x[j]);
        }
        mitjana [j+1]=(double)(mitjana[j+1])/(N);
    }
    for (i = 0; i < N; i++) {
        vary +=(M[i].y - mitjana[0]) * (M[i].y - mitjana[0]);
    }
    vary=(double)(vary) / (N - 1);
    vary=sqrt(vary);
    for (j = 0; j < m; j++) {
        cov = 0;
        varx = 0;
        for (k = 0; k < N; k++) {
            varx += (M[k].x[j] - mitjana[j + 1]) *(M[k].x[j] - mitjana[j +
    1]);
            cov += (M[k].y - mitjana[0]) *(M[k].x[j] - mitjana[j + 1]);
        }
        varx = (double)(varx)/(N-1);
        cov = (double)(cov)/(N-1);
        varx = sqrt (varx);
        r[j] = cov / (vary * varx);
        r[j] = fabs(r[j]);
        total += r[j];
    }
    for (j = 0; j < m; j++) {
        omega[j] = r[j] / total;
    }
    free(r);
    free(mitjana);
}
double solucio(double **W, struct valor *M, int N) {
    int i, j;
    double sol = 0, aux , norma = 0;
```

```
353     for (i = 0; i < N; i++) {
354         aux=0;
355         for (j = 0; j < N; j++) {
356             norma += W[i][j];
357             sol+=W[i][j]*M[j].y;
358         }
359     }
360     sol = sol / norma;
361     return sol;
362 }
363 double **jacobi(double **W,double **Wt, int n){
364         int i, j, p,k,q,maxIter=100000, iter=0, COUNT=0;
365         double **pseudoinversa,**auxiliar,**b,**o,**u,tolerancia=1.e-9, max
    ,aux,s,c,*vector,d,y,CERO=1e-14;
366         vector=(double *)calloc(n,sizeof(double));
367         pseudoinversa = (double **)calloc (n,sizeof(double*));
368         auxiliar = (double **)calloc (n,sizeof(double*));
369         b=(double**)malloc(n*sizeof(double*));
370         o=(double**)malloc(n*sizeof(double*));
371         u=(double**)calloc(n,sizeof(double*));
372         for(k=0;k<n;k++){
373             b[k]=(double*)malloc((k+1)*sizeof(double));
374             o[k]=(double*)malloc(n*sizeof(double));
375             u[k]=(double*)calloc(n,sizeof(double));
376             pseudoinversa[k] = (double *)calloc (n,sizeof(double ));
377             auxiliar[k] = (double *)calloc (n,sizeof(double));
378                 if (b[k] == NULL || o[k] == NULL ){
379                     exit(1);
380                 }
381                 for(j=0; j<=k; j++){
382                     b[k][j]=Wt[k][j];
383                     o[k][j]=0.;
384                 }
385                 for(j=k+1;j<n;j++){
386                     o[k][j]=0.;
387                 }
388                 o[k][k]=1.;
389         }
390 
391         do{
392             p=0;
393             q=1;
394             max=fabs(b[q][p]);
395             for (i=2;i<n;i++){
396                 for(j=0;j<i;j++){
397                     if(max<fabs(b[i][j])){
398                         max=fabs(b[i][j]);
399                         q=i;
400                         p=j;
401                     }
402                 }
403             }
404             y=b[p][p]-b[q][q];
405             if(fabs(y)<CERO){
406                 c=sin(acos(0)/2);
407                 s=c*fabs(b[q][p])/b[q][p];
408             }else{
409                 d=sqrt(4*b[q][p]*b[q][p]+y*y);
410                 c=sqrt((d+fabs(y))/(2*d));
411                 s=(y*b[q][p])/(fabs(y*b[q][p]))*sqrt((d-fabs(y))/(2*d));
412             }
```

```
413             for(i=0;i<p;i++){
414                 y=b[p][i]*c+b[q][i]*s;
415                 b[q][i]=(-1)*b[p][i]*s+b[q][i]*c;
416                 b[p][i]=y;
417             }
418             for(i=p+1;i<q;i++){
419                 y=b[i][p]*c+b[q][i]*s;
420                 b[q][i]=(-1)*b[i][p]*s+b[q][i]*c;
421                 b[i][p]=y;
422             }
423             for(i=q+1;i<n;i++){
424                 y=b[i][p]*c+b[i][q]*s;
425                 b[i][q]=(-1)*b[i][p]*s+b[i][q]*c;
426                 b[i][p]=y;
427             }
428             b[p][p]=b[p][p]+b[q][p]*(s/c);
429             b[q][q]=b[q][q]+b[q][p]*(-1)*(s/c);
430             b[q][p]=0.;
431             for (i=0;i<n;i++){
432                 y=o[i][p]*c+o[i][q]*s;
433                 o[i][q]=(-1)*o[i][p]*s+o[i][q]*c;
434                 o[i][p]=y;
435             }
436             iter++;
437             if(n<4){
438               max=0.;
439             }
440         }while((iter<maxIter && max>tolerancia));
441         if(maxIter<=iter){
442             printf("S'ha superat el numero maxim d'iteracions");
443         }else {
444             for(j=0;j<n;j++){
445                 if(fabs(b[j][j])>1e-7){
446                     b[j][j]=sqrt(b[j][j]);
447                     for(k=COUNT-1;k>=0;k--){
448                         if(fabs(b[k][k]-b[j][j])<tolerancia){
449                             for(q=0;q<n;q++){
450                                 vector[q]=o[q][j];
451                             }
452                             s=b[j][j];
453                             for (q=k+1;q<j;q++){
454                                 for(p=0;p<n;p++){
455                                     o[p][q+1]=o[p][q];
456                                 }
457                                 b[q+1][q+1]=b[q][q];
458                             }
459                             for(p=0;p<n;p++){
460                                 o[p][k+1]=vector[p];
461                             }
462                             b[k+1][k+1]=s;
463                             k=-1;
464                         }
465                     }
466                     if(k!=0 || COUNT==0){
467                         for(q=0;q<n;q++){
468                             aux=o[q][COUNT];
469                             o[q][COUNT]=o[q][j];
470                             o[q][j]=aux;
471                         }
472                         aux=b[COUNT][COUNT];
473                         b[COUNT][COUNT]=b[j][j];
```

```
                        b[j][j]=aux;
                    }
                    COUNT++;
                }
            }
            for(i=0;i<COUNT;i++){
                s=0;
                for(k=0;k<n;k++){
                    s+=o[k][i]*o[k][i];
                }
                s=sqrt(s);
                for(k=0;k<n;k++){
                    o[k][i]=o[k][i]/s;
                }
                for(k=0;k<n;k++){
                    for(j=0;j<n;j++){
                        u[i][k]+=(W[k][j]*o[j][i]);
                    }
                    auxiliar[k][i]=u[i][k];
                    u[i][k]=u[i][k]/b[i][i];
                }
            }
            for(i=0;i<n;i++){
                for (j=0; j<n; j++){
                    pseudoinversa[i][j]=0;
                    for(k=0;k<COUNT;k++){
                        pseudoinversa[i][j]+=auxiliar[i][k]*o[j][k];
                    }
                }
            }
            for(i=0;i<COUNT;i++){
                for(j=0;j<n;j++){
                    o[j][i]=o[j][i]/b[i][i];
                }
            }
            for(i=0;i<n;i++){
              free (auxiliar[i]);
                for(j=0;j<n;j++){
                    pseudoinversa[i][j]=0;
                    for(k=0;k<COUNT;k++){
                        pseudoinversa[i][j]+=o[i][k]*u[k][j];
                    }
                }
            }
            free(auxiliar);
            for (k = 0; k < n; k++) {
                free(u[k]);
                free(b[k]);
                free(o[k]);
            }
            free(u);
            free (b);
            free (o) ;
        }
    return pseudoinversa;
}
double **productetransposada(double **W,int n){
    int i,j,k;
    double **result;
    result=(double **)calloc(n,sizeof(double*));
    for (i=0;i<n;i++){
```

```
535         result[i]=(double *)calloc(n,sizeof(double));
536         for (j=0;j<n;j++){
537             for(k=0;k<n;k++){
538                 result[i][j]+=W[i][k]*W[k][j];
539             }
540         }
541     }
542     return result;
543 }
544 double **inv (double **W, int n){
545     double **matrix, ratio=0,a,aux, **result;
546     int i, j, k, canvifiles=1;;
547     int max;
548     double factor, temp,pivot,det=1;
549     matrix=(double **)malloc(n*sizeof(double *));
550     result = (double **)calloc(n,sizeof (double *));
551     for(i = 0; i < n; i++){
552         matrix[i]=(double *)malloc(2*n*sizeof(double));
553         result[i]=(double *)calloc(n,sizeof(double));
554         for(j = 0; j < n; j++){
555             matrix[i][j]=W[i][j];
556         }
557         for(j = n; j < 2*n; j++){
558             if(i==(j-n))
559                 matrix[i][j] = 1.0;
560             else
561                 matrix[i][j] = 0.0;
562         }
563     }
564     for(i = 0; i < n; i++){
565         max = i;
566         for( int j = i + 1; j < n; j++ ){
567             if( fabs( matrix[j][i] ) > fabs( matrix[max][i] ) )
568                 max = j;
569         }
570         if(i!=max){
571             canvifiles*=(-1);
572             intercambiarFilas( matrix, 2*n, i, max );
573         }
574         for(j =  i+1; j < n; j++){
575               if(fabs(matrix[i][i])>1e-7){
576                   ratio = matrix[j][i]/matrix[i][i];
577                   for(k = i; k < 2*n; k++){
578                       matrix[j][k] -= ratio * matrix[i][k];
579                   }
580               }
581         }
582     }
583     for(i = 0; i < n; i++){
584         a = matrix[i][i];
585         if(fabs(a)<1e-13){
586           return W;
587         }
588         det = det * a;
589         for(j = 0; j < 2*n; j++){
590             matrix[i][j] /= a;
591         }
592     }
593     det=det*canvifiles;
594     for(i=n-1;i>0;i--){
595         for(j=i-1;j>=0;j--){
```

```
596             pivot=matrix[j][i];
597             for(k=i;k<2*n;k++){
598                 matrix[j][k]=matrix[j][k]-matrix[i][k]*pivot;
599             }
600         }
601     }
602     for(i = 0; i < n; i++){
603         for(j = n; j < 2*n; j++){
604             result[i][j-n]=matrix[i][j];
605         }
606     }
607     for (i=0 ; i<n ;i++) free (matrix[i]);
608     free(matrix);
609     return result;
610 }
611 void intercambiarFilas( double **M, int col, int k, int max )
612 {
613     double temp[col];
614     for( int i = k; i < col; i++ ) {
615         temp[i] = M[k][i];
616         M[k][i] = M[max][i];
617         M[max][i] = temp[i];
618     }
619 }
```

Valors predits

```
[2006.032258 , 50.483871 , 19.117395] ,
[2006.564516 , 50.967742 , 25.549422] ,
[2007.096774 , 51.451613 , 27.452098] ,
[2007.629032 , 51.935484 , 28.222434] ,
[2008.161290 , 52.419355 , 28.539837] ,
[2008.693548 , 52.903226 , 28.630770] ,
[2009.225806 , 53.387097 , 28.592294] ,
[2009.758065 , 53.870968 , 28.472937] ,
[2010.290323 , 54.354839 , 28.299659] ,
[2010.822581 , 54.838710 , 28.088636] ,
[2011.354839 , 55.322581 , 27.850163] ,
[2011.887097 , 55.806452 , 27.591102] ,
[2012.419355 , 56.290323 , 27.316203] ,
[2012.951613 , 56.774194 , 27.028862] ,
[2013.483871 , 57.258065 , 26.731566] ,
[2014.016129 , 57.741935 , 26.426182] ,
[2014.548387 , 58.225806 , 26.114137] ,
[2015.080645 , 58.709677 , 25.796541] ,
[2015.612903 , 59.193548 , 25.474272] ,
[2016.145161 , 59.677419 , 25.148029] ,
[2016.677419 , 60.161290 , 24.818380] ,
[2017.209677 , 60.645161 , 24.485790] ,
[2017.741935 , 61.129032 , 24.150643] ,
[2018.274194 , 61.612903 , 23.813258] ,
[2018.806452 , 62.096774 , 23.473904] ,
[2019.338710 , 62.580645 , 23.132807] ,
[2019.870968 , 63.064516 , 22.790163] ,
[2020.403226 , 63.548387 , 23.796553] ,
[2020.935484 , 64.032258 , 23.006744] ,
[2021.467742 , 64.516129 , 23.624386] ,
[2022.000000 , 65.000000 , 23.142421] ,
```

`aplhaestrella`=44.600000, `norma min`=0.007192

Valors resultants d'evaluar els punts $\overrightarrow{S_k}$ (2.21) a la funció de l'esperança de vida. Per a comprendreu-ho millor, fem un exemple.

Exemple: A la fila 2, l'esperança de vida en l'any 2006.5645$\approx$ *7 de juny del 2006* per a una persona amb 50.967742 anys$\approx$ *50 anys, 11 mesos i 15 dies* de vida és de 25 anys i mig aproximadament.

Referències

[1] *Esperanza de Vida*. 2022. URL: https://es.wikipedia.org/wiki/Esperanza_de_vida.

[2] Julio Pérez Díaz. *FÓRMULA PARA EL CÁLCULO DE LA ESPERANZA DE VIDA*. 2010. URL: https://apuntesdedemografia.com/curso-de-demografia/temario/tema-4-analisis-de-la-mortalidad/formula-para-el-calculo-de-la-esperanza-de-vida/.

[3] Juan Manuel García González. "¿ Por qué vivimos más? Descomposición por causa de la esperanza de vida española de 1980 a 2009". A: *Revista Española de Investigaciones Sociológicas (REIS)* 148.1 (2014), pàg. 39 - 59.

[4] Jorge Miguel Bravo i Javier Díaz-Giménez. "¿ La longevidad es un riesgo asegurable? Cubriendo lo Incubrible". A: *Vocales del Foro de Expertos del Instituto BBVA de Pensiones. Madrid. Informe* (2014), pàg. 04 - 06.

[5] Pablo Cerezal. "La verdad sobre el futuro de las pensiones". A: *Expansión* (2019).

[6] Iñaki Garay. "El Estado de Bienestar, en el alambre". A: *Expansión* (2020).

[7] AV Kryanev i DK Udumyan. "Metric analysis, properties and applications as a tool for interpolation". A: *International Journal of Mathematical Analysis* 8.45 (2014), pàg. 2221 - 2228.

[8] Aleksandr Vital'evich Kryanev, GV Lukin i David Kadjikovich Udumyan. "Metric analysis and applications". A: *Numerical Methods and Programming (Vychislitel'nye Metody i Programmirovanie)* 10 (2009), pàg. 408 - 414.

[9] AV Kryanev, VV Ivanov, LA Sevastyanov i DK Udumyan. "Reconstruction of Multivariable Functions Under Uncertainty by Means of the Scheme of Metric Analysis". A: *International Conference on Stochastic Methods*. Springer. 2020, pàg. 269 - 279.

[10] AV Kryanev, DK Udumyan, GV Lukin i VV Ivanov. "Metric analysis approach for interpolation and forecasting of time processes". A: *Applied Mathematical Sciences* 8.22 (2014), pàg. 1053 - 1060.

[11] Richard E. Quandt. *Some Basic Matrix Theorems*. 2004. URL: http://www.quandt.com/papers/basicmatrixtheorems.pdf.

[12] David H Wagner. *PROOF OF SCHUR'S THEOREM*. 2011. URL: https://www.math.uh.edu/~wagner/2331/Schurs_theorem.pdf.

[13] Alexander Kryanev, Gleb Lukin i David Udumyan. "Metric analysis as a tool for interpolating multivariate functions in the case of an information lack". A: *International Conference on Distributed Computer and Communication Networks*. Springer. 2016, pàg. 553 - 564.

[14] Oskar R Cahueñas. "Precondicionamiento por aproximaciones a la matriz pseudoinversa para el problema de mínimos cuadrados lineales". A: ().

[15] Jesús Sala, Aurora Torrente i Eduardo JS Villaseñor. *Tema 14. Pseudoinversa y descomposición en valores singulares*. 2015.

[16] Dmitri Fedorov. *1 Eigenvalues and eigenvectors*. 2019.

[17] AV Kryanev, VV Ivanov, LA Sevastianov i DK Udumyan. "A review of metric analysis applications to the problems of interpolating, filtering and predicting the values of onevariable and multivariable functions". A: *International Conference on Distributed Computer and Communication Networks*. Springer. 2018, pàg. 457 - 468.

[18] *Tablas de mortalidad por año, Sexo, edad y funciones.* URL: https://www.ine.es/jaxiT3/Datos.htm?t=27153.

[19] José Manuel Rojo. *Regresión lineal múltiple.* 2007. URL: http://humanidades.cchs.csic.es/cchs/web_UAE/tutoriales/PDF/Regresion_lineal_multiple_3.pdf.

[20] *Regresión Lineal Múltiple: El modelo, estimación de los parámetros, contrastes.* 2011. URL: http://eio.usc.es/eipc1/BASE/BASEMASTER/FORMULARIOS-PHP-DPTO/MATERIALES/Mat_50140128_RegresionMultiple.pdf.