1. Resolució de sistemes triangulars

1.1 resoltrisup

He implementat aquesta funció per tal de resoldre sistemes lineals triangulars superiors de l'estil: Ax = b. Per fer-ho s'ha implementat una funció de tipus int que rep per paràmetres la dimensió de la matriu (n), la matriu, el vector de termes independents, el vector on es guardarà la solució i la màxima tolerància acceptada.

Per resoldre el sistema s'ha suposat que la matriu és triangular superior (és a dir, només utilitzarem els valors de la part superior), i s'ha resolt utilitzant substitució enrere. El sistema tindrà solució (retornarà 0) sempre i quan cap $|a_{ii}| < \varepsilon$, on ε és la tolerància donada. És inevitable que una certa quantitat d'error es produeixi a l'hora de resoldre el sistema, per això és important especificar el rang d'error permès. Si no té solució ($|a_{ii}| < \varepsilon$) retornarà 1.

1.2 main_trisup

Per tal de comprovar el correcte funcionament de la funció anterior, s'ha implementat una funció principal que cridarà a ***resoltrisup***, i escriurà la solució *x* del sistema i el residu $||Ax-b||_2$. Per fer el càlcul del residu es faran servir les funcions ***prodMatVec*** i ***prod_esc*** fetes en anteriors sessions.

La funció principal (guardada en el fitxer ***main_trisup.c***) agafarà d'un fitxer (***ex2_1.in*** i ***ex2_2.in***) totes les dades necessàries: la dimensió de la matriu A, els valors per inicialitzar la matriu, els valors del vector de termes independents i per últim la màxima tolerància acceptada per a resoldre el sistema. Posteriorment es crida a la funció ***resoltrisup*** amb els paràmetres esmentats anteriorment; si aquesta retorna 0 vol dir que ha pogut solucionar el sistema i procedim a fer el producte de A*x fent ús de la funció ***prodMatVec*** a la que passem les paràmetres següents:

- Matriu **A**
- Vector solució del sistema, **x** (prèviament inicialitzat)
- Vector on es guardarà el resultat de fer l'operació, **v** (prèviament inicialitzat)
- Dimensió de la matriu A, **n**

Actualitzem el valor del vector **v** restant-li el valor de **b** (*v* -= *b*) per guardar el resultat de fer ***Ax-b***. Posteriorment cridem a la funció ***prod_esc*** passant-li com a paràmetres la dimensió ***n***, i el vector ***v*** dos cops (per tal de calcular el producte escalar per ell mateix), per posteriorment fer l'arrel quadrada (*sqrt*) i així obtenir el valor del residu del sistema. Aquest residu en condicions ideals hauria de ser 0 (i això em mostra el programa pel primer sistema d'exemple), però podem comprovar que amb el segon sistema de l'enunciat això no és així (tot i que és un nombre de l'ordre de 10^{-15}).

1.3 resoltriinf

Exercici opcional en el què es demanava implementar una funció anàloga a l'anterior, anomenada aquest cop ***resoltriinf***. Aquesta funció resol sistemes triangulars inferiors utilitzant el mètode de substitució endavant (suposant que la matriu donada és triangular inferior, ja que només s'utilitzaran els valors de la part inferior).

Per comprovar el funcionament d'aquesta funció s'ha implementat una funció principal (guardada en el fitxer ***main_triinf.c***). Aquesta funció treballa de forma anàloga a la que resol sistemes triangulars superiors, tot i que resolent $A^Tx = b$, amb A (suposem que la matriu A s'introdueix ja trasposada) i b de l'exercici anterior.

2. El mètode de Gauss sense pivotatge

2.1 Gauss

S'ha implementat una funció de tipus *int* que és capaç de resoldre el sistema Ax = b, on $A \in \mathbb{R}^{nxn}$, on ara la matriu A és una qualsevol (és a dir, no ha de ser obligatòriament una matriu triangular superior/inferior com en l'anterior apartat). Resoldrà el sistema fent ús del mètode de Gauss sense pivotatge.

Un cop triangularitzat el sistema amb la funció, es cridarà a ***resoltrisup*** per resoldre'l. Els paràmetres de la funció de Gauss són els següents:

- Dimensió de la matriu, **n**
- Matriu **A**
- Vector de termes independents, **b**
- Tolerància acceptada, **tol**

A la sortida de la funció però tindrem la matriu A **modificada** (triangularitzada resultant d'aplicar Gauss) i el vector de termes independents b ara contindrà la **solució** del sistema. La funció ***gauss*** retornarà 0 si ha trobat la solució (cap $|a_{ii}| < \varepsilon$), 1 altrament.

2.2 main_gauss

Per comprovar el funcionament de la funció ***gauss***, he escrit una funció principal (guardada al fitxer ***main_gauss.c***) que llegeix una matriu **A**, un vector de termes independents **b**, la màxima tolerància acceptada (totes aquestes dades guardades en els fitxers: ***ex5_1.in*** i ***ex5_2.in***) i crida a ***gauss***.

En el cas que la funció retorni 0 (el sistema té solució), es procedeix al càlcul del valor del residu $\mathbf{||Ax-b||_2}$. Aquest cop però, no podem fer el càlcul del residu amb la matriu A i el vector b, ja que resultat de fer Gauss, la matriu A ara està **modificada** i el vector b conté la solució del sistema, per tant abans de cridar a gauss s'ha fet una **còpia** de la matriu A (inicial) i del vector de termes independents b (inicial) per procedir correctament a realitzar el càlcul del residu ($\mathbf{||Acopy * b – x||_2}$, on ara x és la còpia del vector b inicial).

Al resoldre el primer sistema amb una tolerància igual a 10^{-3} obtenim un residu de l'ordre de 10^{-16}, mostrant-nos que el càlcul de la resolució del sistema evidentment no dona exacte.

Si intentem resoldre el segon sistema amb una tolerància de 10^{-3} podem comprovar que no es pot resoldre el sistema ja que l'error en els càlculs és més gran que la tolerància. Per una tolerància de 10^{-6} si que podem resoldre el sistema i ens retorna un residu de l'ordre de 10^{-16}, indicant que el càlcul de la resolució del sistema no és exacte (té error acumulat).

2.3 checkLU

La funció ***checkLU***, de tipus *double*, rep per paràmetres:

- Dimensió de la matriu, **n**
- Matriu **A original**, abans d'aplicar ***gauss***
- Matriu **A modificada** (***acp***), resultat d'aplicar ***gauss***.

La matriu **L** contindrà la *part triangular inferior estricte amb 1 a la diagonal*, de la matriu modificada. La matriu L guarda els multiplicadors fets servir per triangularitzar una matriu per Gauss, per això s'ha modificat la funció ***gauss*** per tal de que en comptes de 0, guardi els multiplicadors en la part inferior estricte (això no afecta a la funció ***resoltrisup*** ja que aquesta només farà servir els valors de la part superior). La matriu **U** en canvi serà la *part triangular superior de la matriu A modificada*.

La funció ***checkLU*** retornarà $\max_{0\leq i,j\leq n} |B_{i,j}|$ si B = A – LU. He implementat la funció sense usar cap matriu ni vector auxiliar, fent la multiplicació LU directament a la matriu modificada (***acp***) i restant-li el valor de a[i][j] per guardar-ho en la matriu A original (***a***), tot això com he dit, sense crear una matriu B ni una matriu LU. La funció va actualitzant a cada iteració (si es necessari) el valor del màxim, per posteriorment retornar-lo.

La funció consta de tres bucles ***for*** (els necessaris per fer una multiplicació) amb la condició que l'últim ***for*** arribarà com a màxim fins el mínim valor de les dues anteriors variables dels anteriors ***fors*** (i, j). Si ens trobem a la diagonal, per fer el càlcul haurem d'agafar el valor que hi hagi a la diagonal de la matriu ***acp***, ja que la matriu L té 1 a la diagonal, per tant hauríem d'agafar el valor que tindria la matriu U en aquest cas. Aquesta condició sobre la diagonal l'he realitzat fent un ***if***, tot i que es podria modificar un dels ***fors*** per tal de fer més òptim el càlcul (ens evita fer la comprovació de si ens trobem a la diagonal a cada iteració).

Per comprovar el correcte funcionament de la funció (guardada en el fitxer ***main_gaussLU.c***) s'han utilitzat els sistemes de l'apartat **2.2** (guardats en els fitxers: ***ex5_1.in*** i ***ex5_2.in***). Aquesta funció principal llegeix una matriu (de la qual es guardarà una còpia per passar-la a la funció ***checkLU*** sense modificar), un vector de termes independents i la tolerància. Es crida a la funció ***gauss***, si aquesta té solució (retorna 0) es procedeix a cridar a ***checkLU*** per imprimir el màxim valor del sistema.

Pel primer sistema (guardat en el fitxer ***ex5_1.in***) el màxim que retorna és 0.

En canvi pel segon sistema (guardat en el fitxer ***ex5_2.in***) el màxim que retorna és de l'ordre de 10^{-17}, indicant-nos així que el càlcul de B = A – LU que de forma ideal hauria de donar 0, no és així, ja que arrosseguem l'error d'haver aplicat ***gauss***.

2.4 $Ax = e_i$, usant LU de A_2

Exercici opcional en què s'ha implementat una funció (en l'arxiu ***main_inverses.c***) que resol $Ax = e_i$, on e_i denota el vector i-èssim de la base canònica de R^n, usant la descomposició LU de A_2 (matriu d'exemple a l'apartat **2.2**, l'obtenim del fitxer ***ex5_2.in***). Concretament es demana resoldre els *n* sistemes lineals que donarien la inversa.

Primer he cridat a gauss per obtenir la matriu A ***modificada*** d'on inicialitzaré les matrius L i U. Posteriorment crido a la funció ***resoltriinf*** per a resoldre el sistema $Ly = e_i$ (on e_i denota el vector i-èssim de la base canònica de R^n), on y = Ux. Després crido a la funció ***resoltrisup*** per a resoldre el sistema Ux = y. Finalment s'imprimeixen les inverses per a totes les bases canòniques.

3. Mètode de Gauss – pivotatge maximal per columnes

3.1 gausspiv

Funció que resol el sistema Ax = b usant el mètode de Gauss amb pivotatge. És una modificació de la funció ***gauss***, ja que primer es pivota (si és necessari) i després es resol per Gauss.

Per tal de pivotar s'ha de tenir en compte que abans de calcular el multiplicador per fer Gauss, hem de buscar $\max_{j=k,\ldots,n} |a^{(k)}_{jk}|$. Si denotem $l \in \{k, k+1, \ldots, n\}$ la fila tal que $a^{(k)}_{lk}$ assoleix aquest màxim, llavors s'intercanvia la fila l per la fila k (també l'element corresponent del vector b) i es continua amb el mètode. Per dur a terme l'intercanvi de files s'intercanvien els apuntadors corresponents per no haver de canviar els elements un per un.

Bàsicament, si l'element de la diagonal (en valor absolut) és més petit que qualsevol terme inferior a ell, fem pivotatge (intercanviant les files corresponents i l'element corresponent del vector b).

CORRECCIÓ: En l'anterior codi a l'hora de fer pivotatge, intercanviava vàries files en el cas de trobar diferents màxims a l'hora de recórrer els valors. En la modificació es busca el màxim i es guarda la posició *k* de la fila on s'ha trobat, per només haver de fer un canvi.

Un cop fet el pivotatge (si era necessari) resolem per Gauss. Finalitzat Gauss (ja tenim la matriu A triangularitzada), cridem a la funció ***resoltrisup*** per resoldre el sistema.

3.2 main_gausspiv

En aquest apartat hem modificat la funció principal de l'apartat **2.2** per tal de que cridi la funció ***gausspiv*** en lloc de la funció ***gauss*** (és l'única diferència). Aquesta funció l'hem guardat al fitxer ***main_gausspiv.c***.

Per provar la funció hem utilitzat els mateixos sistemes que a l'apartat 2.2 (guardats en els fitxers: ***ex5_1.in*** i ***ex5_2.in***) per tal de comprar els valors dels residus en ambdós casos:

- Pel sistema guardat en el fitxer ***ex5_1.in*** el residu és de l'ordre de 10^{-16}, pràcticament insignificant, però ens indica que estem cometen un error a l'ordre de fer les operacions.
- Pel sistema guardat en el fitxer ***ex5_2.in*** tenim un residu del mateix ordre que l'anterior, la qual cosa ens diu que també estem cometen un error al fer el càlcul de la solució d'aquest sistema.

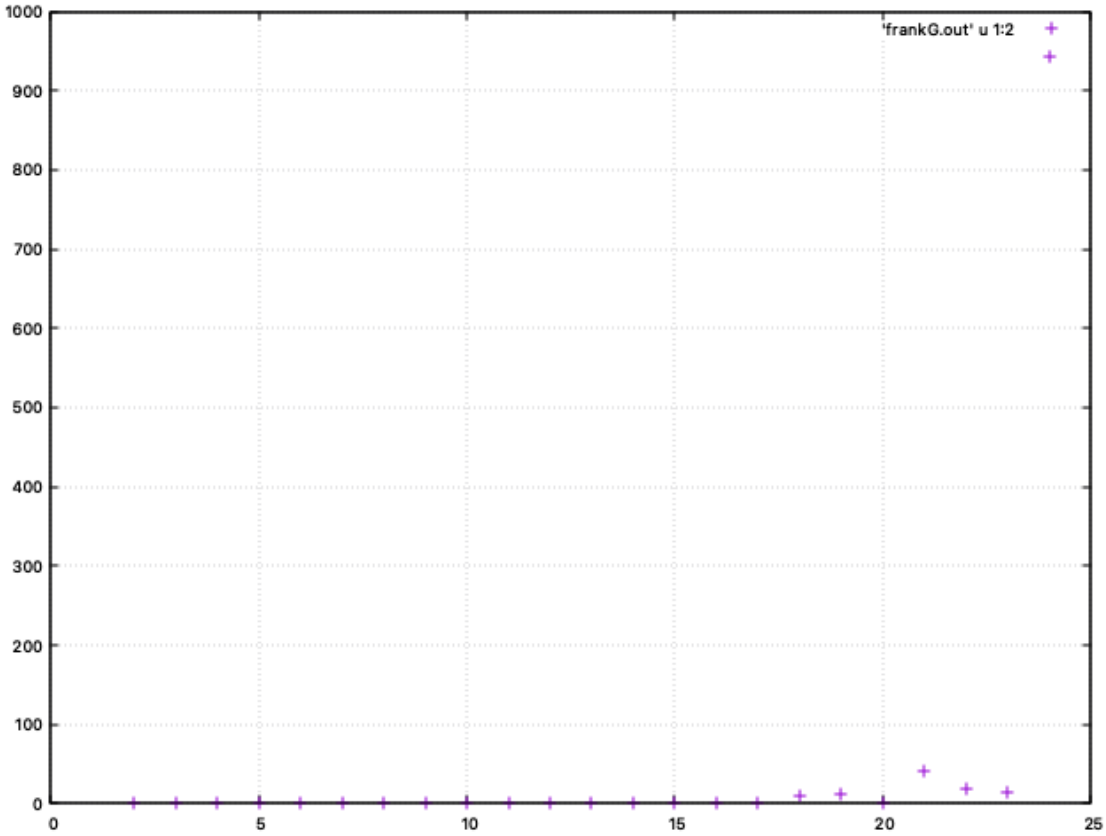
3.3 Matriu Hessenberg superior de Frank

En aquest apartat se'ns demanava primerament inicialitzar diferents matrius Hessenberg superior de Frank amb un certs coeficients (condicions indicades a l'enunciat de la pràctica) i un vector de termes independents resultat de fer $b = Ax_0$ on $x_0 = (1, \ldots, 1)^T$. Per inicialitzar els coeficients ho he fet seguint les indicacions de l'enunciat fent distincions amb ***if***, ***else if*** i ***else***. Potser es podria haver implementat

d'una millor manera per tal de no haver de fer aquestes comprovacions a cada iteració.

He fet una funció principal (guardada en el fitxer ***main_frank.c***) que resol el sistema Ax = b per n = 2, ..., 24 i escriu en un fitxer el valor de n i el error relatiu de la solució x obtinguda. La funció deixa escollir si s'utilitza ***gauss*** o ***gausspiv*** (1 per ***gauss***, 0 per ***gausspiv***) i escriurà en el fitxer ***frankG.out*** o ***frankGP.out***, respectivament.

He representat les gràfiques de $e_{rel}(x)$ en funció de n usant ***gnuplot*** i he obtingut el següent:



Aquesta gràfica representa $e_{rel}(x)$ (eix vertical) en funció de *n* (eix horitzontal) utilitzant la funció de ***gauss*** (he introduït una tolerància de 1.e-3). Podem veure com la precisió de de la solució del sistema en funció de *n* va disminuït conforme la *n* creix. La precisió es manté bastant estable fins n = 16, i per n = 24 podem apreciar com la precisió s'ha perdut totalment (és el valor més dispar).

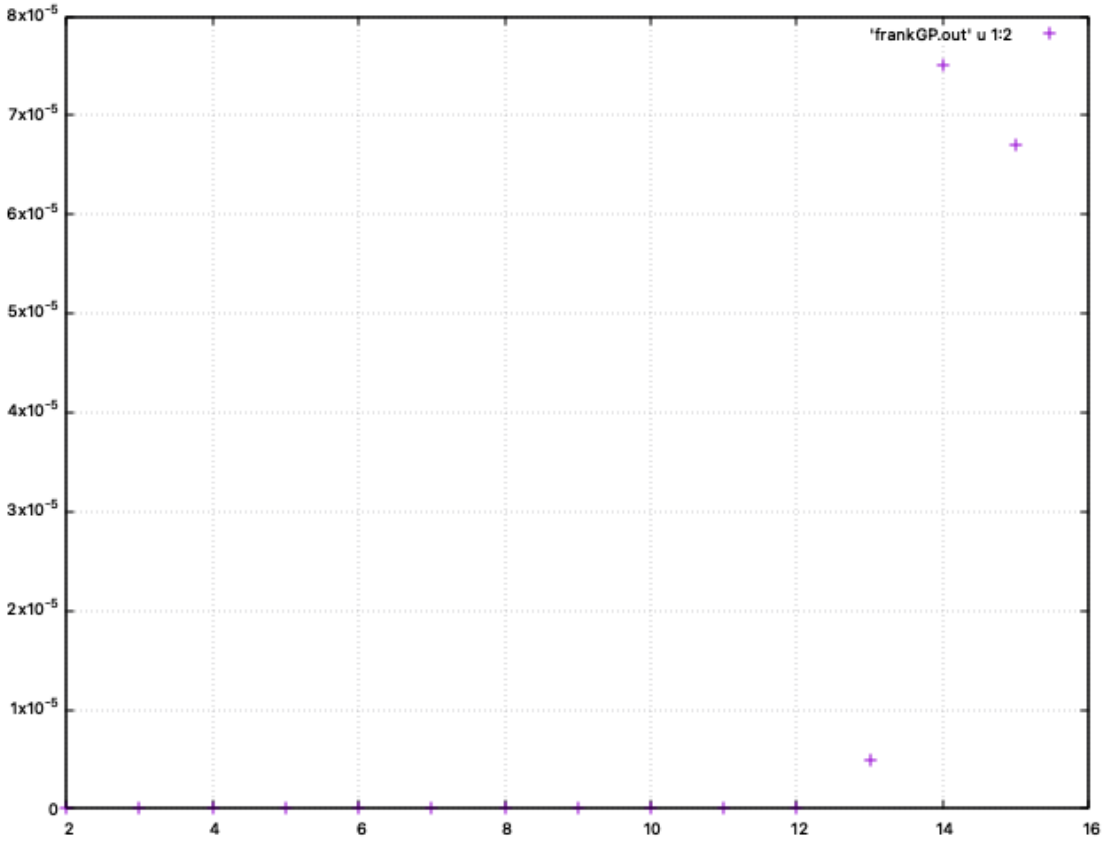
En canvi si volem resoldre els sistemes utilitzant la funció ***gausspiv***, ens trobem amb el problema de que tot i introduir les toleràncies que s'esmenten en l'enunciat (10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}) mai podem arribar a resoldre tots els sistemes per n = 2, ..., 24.

Amb una tolerància de 10^{-3} podem arribar a resoldre fins a n = 6 i obtenim tots els errors relatius = 0.000000, per tant, tot i que pugui no ser exacte, no represento la gràfica ja que no variaria.

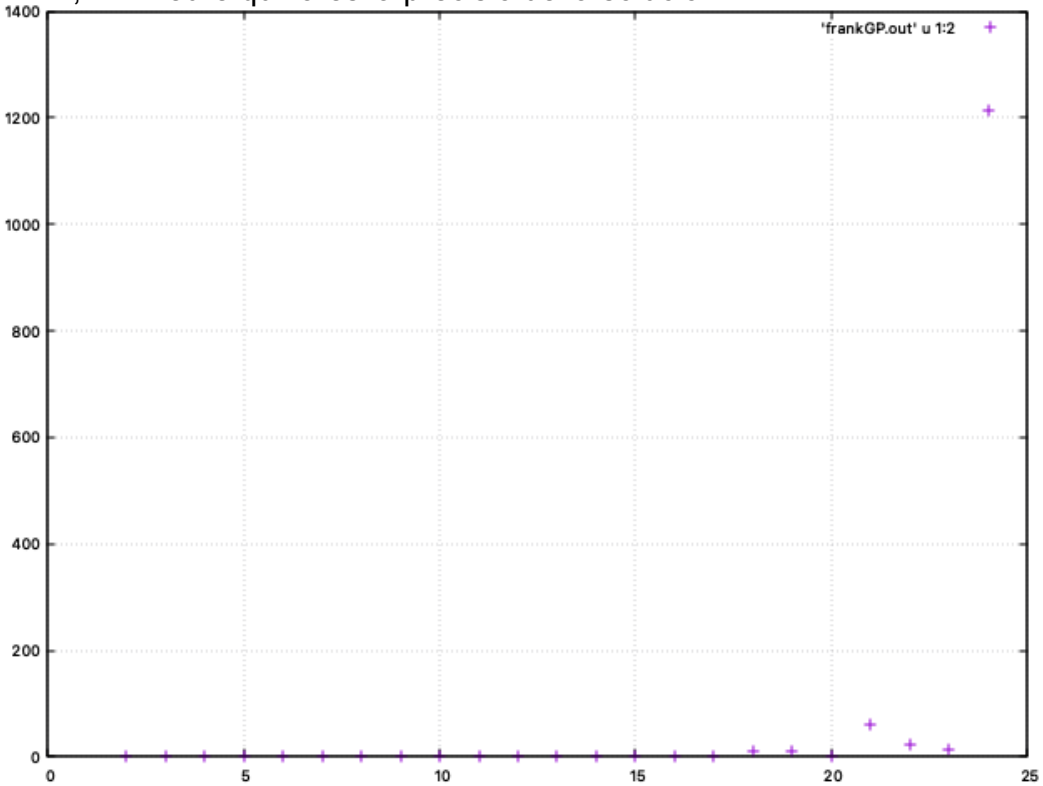
Amb una tolerància de 10^{-6} podem arribar a resoldre fins a n = 9, però ens trobem en la mateixa situació que abans, l'error relatiu calculat ens dona = 0.000000, per tant no cal representar la gràfica.

Per una tolerància de 10^{-9} podem arribar a resoldre fins a n = 12, però tornem a trobar-nos en la mateixa situació que les toleràncies anteriors.

Per una tolerància de 10^{-12} podem arribar a resoldre fins a n = 15. Ara si tenim valors diferents per poder representar una gràfica:



Es pot apreciar que la precisió augmentant la n fins a 15 no varia gaire. Per això he introduït un valor de tolerància encara més petit (10^{-18}) per a poder veure què succeeix si es resolen tots els sistemes per n = 2, ..., 24 i veure quina és la precisió de la solució:



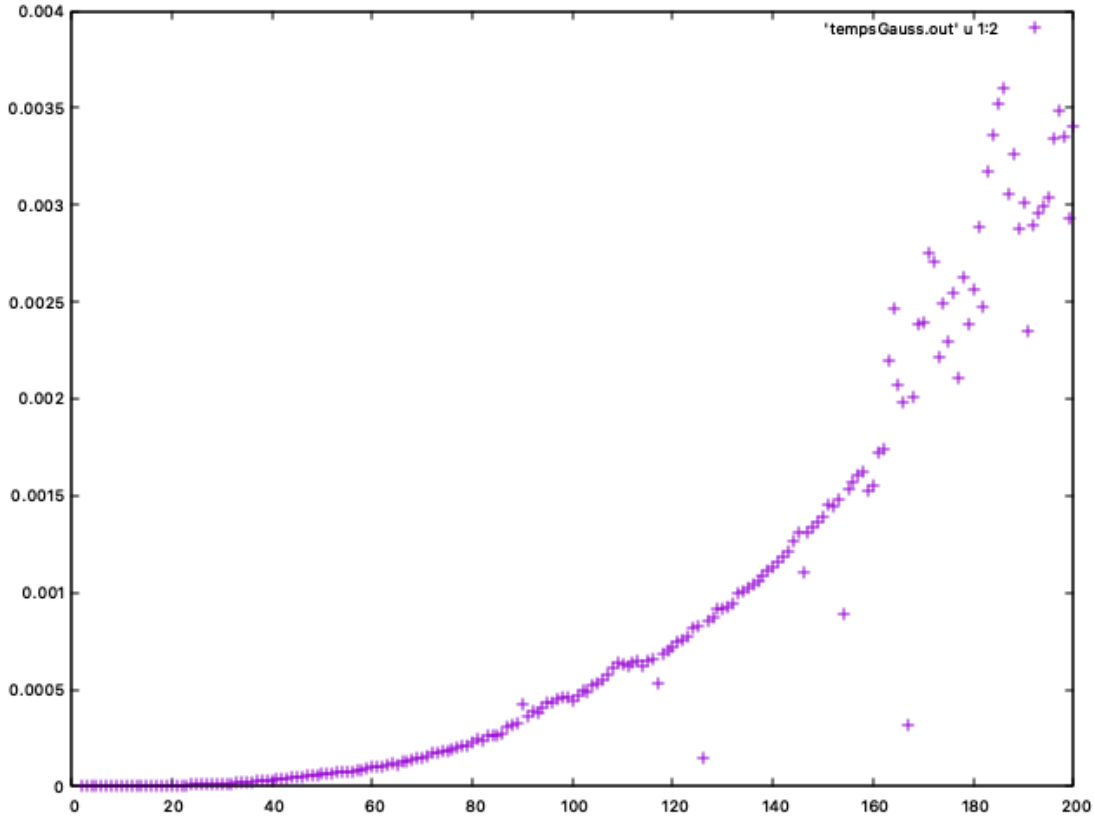
Podem apreciar que fins a n = 16 no tenim gaire error relatiu, però a partir d'aquesta n els valors dels errors són inadmissibles i podem confirmar que s'ha perdut totalment la precisió en la resolució del sistema.

3.4 Temps de resolució de sistemes lineals utilitzant Gauss

En aquest apartat he comprovat el temps de resolució de sistemes lineals usant el mètode de Gauss mitjançant una funció principal guardada en un *main* amb el següent nom: '***main_tempsGauss.c***'.

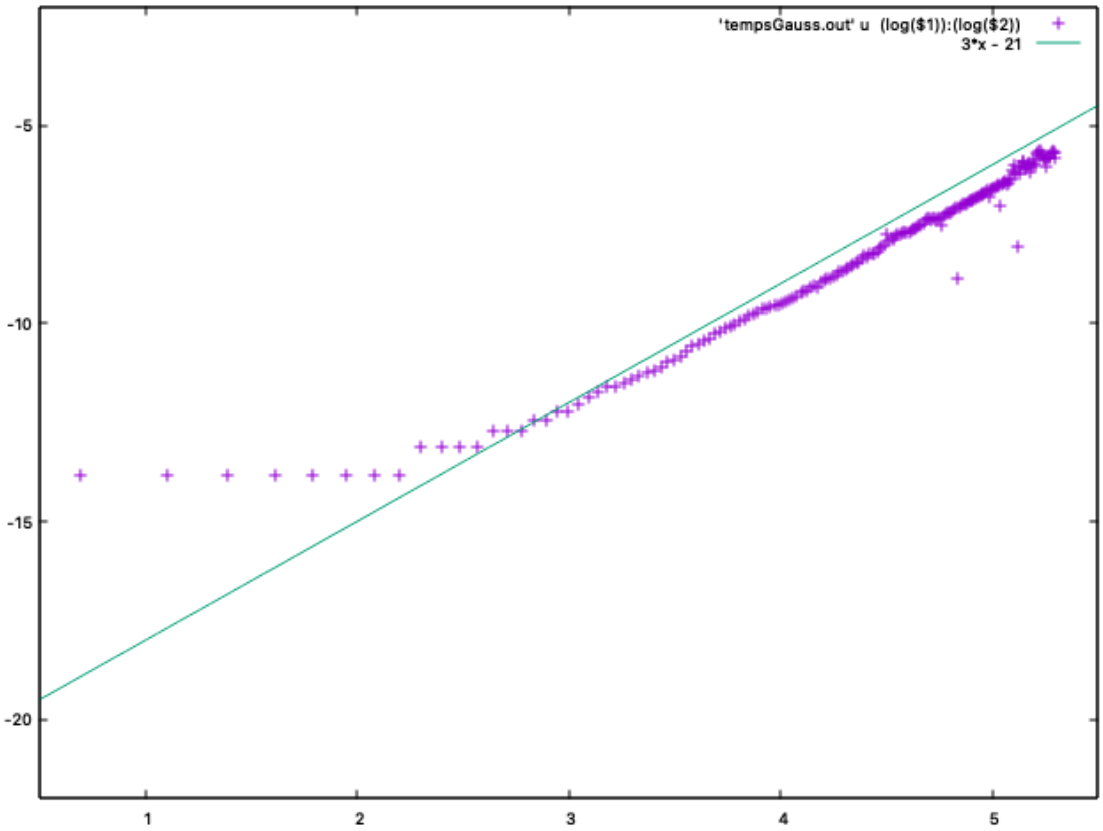
Per fer-ho, per a cada $2 \leq n \leq 200$ genero 10^3 sistemes aleatoris i mesuro el que triga el programa en resoldre el total de sistemes. Afegeixo a un fitxer anomenat '***tempsGauss.out***' el càlcul del temps de còmput per a cada n, per posteriorment fent ús de ***gnuplot*** obtenir la gràfica equivalent.

Representació de la gràfica del temps de còmput en funció de n:



Podem observar un creixement en el temps conforme creix la n, tot i així podem observar casos en què per una n bastant gran, n = 167 (per exemple), el temps d'execució és de només 0.000317 segons. Possiblement això és degut als números generats aleatòriament per tal de generar el sistema; per tant si tornéssim a fer el càlcul del temps, ens sortirien algunes disparitats diferents ja que fem ús de ***srand(time(NULL))*** per tal de tenir una *seed* diferent per generar els números aleatoris a cada execució del programa.

Podem observar que quant més gran és la dimensió dels sistemes ($2 \leq n \leq 200$) més temps triga a executar la funció de Gauss. La gràfica hauria de seguir els valors donats per $\boldsymbol{f(n) = n^3}$, és a dir, creixement cúbic (ja que gauss consta de tres bucles ***fors***). Podem comprovar que això és així fent ús del *gnuplot* per tal de representar ***f(n)*** com a: ***log(f(n)) = 3*log(n)*** i representar també a l'hora la recta ***y = 3x – 21*** (per veure-la a sobre de la funció). Obtenim el següent:



Podem dir llavors que la computació del temps és la correcta i esperada.