Métodos numéricos
Examen Parcial

Salim Vargas Hernández

25 de octubre del 2018

Problema 1

Se busca resolver el sistema de ecuaciones $Ax = b$ formado por la matriz `A_1000.mtx` y el vector `b_1000.vec`. La matriz A es una matriz llena, por lo que no se puede aplicar ninguna técnica de manejo de matrices ralas.

Se pide calcular la inversa de la matriz A. Esto se realizó mediante la factorización LU de A para posteriormente resolver los sistemas de ecuaciones $Ax_i = LUx_i = e_i$ donde e_i es el vector canónico compuesto por ceros y un 1 en la posición i.
Cada x_i corresponde a la columna i de la matriz inversa de A.

La factorización LU permite resolver el sistema de ecuaciones de forma más rápida y sencilla pues son matrices triangulares. Entonces, resolviendo n sistemas de ecuaciones para los n vectores canónicos, podemos formar en su totalidad A^{-1}.
Para resolver el sistema $Ax = b$ basta con multiplicar la matriz inversa A^{-1} que recién se calculó, por el vector b.

Se prefirió usar factorización LU por sobre el método de *Gauss-Jordan* dada la facilidad que presenta LU para resolver los n sistemas de ecuaciones.

El programa guarda en el archivo `A_inv_1000.mtx` la matriz inversa A^{-1} y en el archivo `sol_1000.vec` el vector solución del sistema $Ax = b$.

Problema 2

Se busca resolver el sistema de ecuaciones $Ax = b$ formado por la matriz `A_5000.mtx` y el vector `b_5000.vec`. La matriz A es una matriz rala, por lo que se puede optimizar el código para reducir la memoria y el tiempo utilizados al correr los métodos. Además A es una matriz simétrica, lo cual permite el uso de algunos métodos exclusivos para este tipo de matrices.

Se utilizaron dos métodos para la resolución del sistema, uno iterativo y uno directo.
El método iterativo fue *Gradiente Conjugado*, se escogió este método por asegurar convergencia en a lo mucho n iteraciones, siempre y cuando la matriz A sea simétrica positiva definida.
Para este método se implementaron métodos especiales para matrices ralas, siendo realmente sencillas pues sólo hizo falta adaptar la búsqueda de elementos de la diagonal y la multiplicación de matriz sparse por un vector.
El método convergió en 824 iteraciones, lo cual es mucho menor a la dimensión de la matriz que es $n = 5000$. Además el método convergió en 0.52 segundos, siendo un método realmente veloz y eficaz. Por otra parte, error fue de orden 1×10^{-12} lo cual convierte a la solución encontrada en una excelente aproximación.

El método directo fue *factorización LU*, para este método se tuvo que utilizar una matriz completa, es decir, no se conservó la estrucutra rala de la matriz, lo cual representó un gasto importante en memoria.
El método tardó mucho más tiempo que su contraparte iterativa, requiriendo de un total de 480.5 segundos para completar el cálculo del vector solución.

Si bien *LU* ofrece una solución directa al sistema de ecuaciones, el tiempo y la eficiencia en memoria hacen de *Gradiente Conjugado* una alternativa mucho más factible sin perder en exactitud.
La figura 1 muestra el resultado en consola de los dos métodos implementados para el problema.

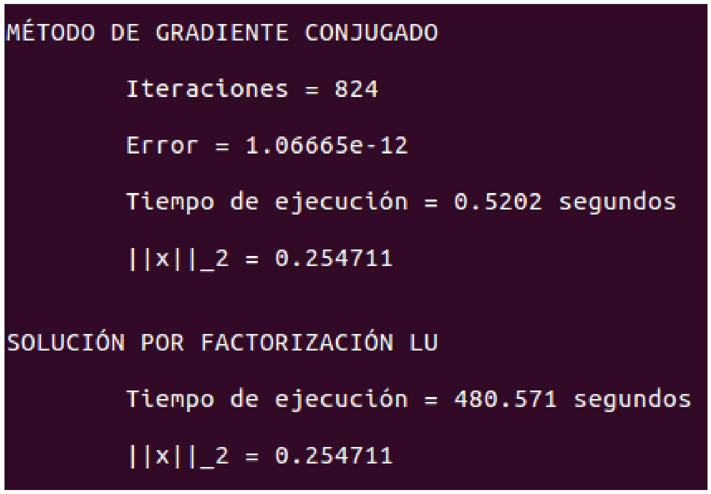

Figura 1: Resultados de métodos iterativo y directo para la matriz `A_5000.mtx`

Problema 3

Se busca encontrar los 10 eigenvalores más grandes y los 10 eigenvalores más pequeños de una matriz A pentadiagonal de dimensión $n = 2000$.
Para ello se puede implementar el método de iteraciones en el subespacio con método de la potencia y método de la potencia inversa.

Dada la forma de la matriz A, algunas operaciones como Ax donde x es un vector cualquiera, y AX donde X es una matriz cualquiera, pueden ser optimizadas para no tener que recorrer toda la matriz A, cuya información yace solamente alrededor de la diagonal.

Tanto el método de iteraciones en el subespacio basado en el método de la potencia, como el basado en el método de la potencia inversa, representan problemas muy grandes para ejecutar en la computadora, además, los eigenvalores de la matriz A están muy juntos entre si, siendo diferentes solamente en los decimales, lo cual hace que el método tarde en converger.
Los parámetros utilizados fueron los siguientes:

- Tolerancia método de iteraciones en el subespacio = ε_M
- Máximo de iteraciones del método de iteraciones en el subespacio = 10000000
- Tolerancia método de Jacobi = 1×10^{-5}
- Máximo de iteraciones del método de Jacobi = 100

El método basado en potencia inversa converge a los 10 siguientes menores eigenvalores:

24.0000906448914	24.0003636233225	24.0008246125480	24.0014841830719	24.0023589382517
24.0034756179779	24.0048769284262	24.0066357071028	24.0088932908350	24.0120118645011

El algoritmo basado en el método de la potencia no logró converger, pero la mejor aproximación fueron los siguientes valores para los 10 eigenvalores más grandes:

72.1660893542794	72.1643125647839	72.1643122340782	72.1632981638515	72.1611765488957
72.1611745121897	72.1563052678319	72.1563013907723	72.1514512739904	72.1486677214188

El método puede no haber convergido principalmente a la cercanía que existe entre los eigenvalores, así como por la dimensión del problema.
Las siguientes figuras muestran ejemplos de las salidas en consola de los métodos implementados. Se imprime el número de iteración y los valores para los 10 eigenvalores (mayores o menores).

```
30831 - 24.0003634791716145 24.00236663737422305 24.01201854442948758 24.00487950637620926 24.00148490383431721 24.00347726849926033 24.0088916
0994845639 24.00082493825320995 24.00663925697041989 24.00009118435019317
30832 - 24.00036346930912856 24.00606402739281009 24.01201817411396888 24.0048793660224824 24.00148486434651929 24.00347716999030112 24.0051938
9932640735 24.00082491560374365 24.00663905590696956 24.00009118264978625
30833 - 24.00036345945393634 24.00235979158938804 24.01201780370660899 24.00684615946339306 24.00148482488188861 24.00347707154033827 24.006930
4301794979 24.00082489296974231 24.00663885488382832 24.0000911809505908
30834 - 24.00036344960602719 24.00856717769338289 24.01201743320555337 24.0026900327547601 24.00148478544276287 24.0034769731315194 24.00487896
398873033 24.00082487035108159 24.00663865390095708 24.00009117925250024
30835 - 24.00036343976534425 24.00235981833827381 24.0120170626138858 24.00889705836348753 24.00148474602756821 24.00101114995783291 24.0048789
0457470641 24.00329057256411858 24.00663845295848375 24.00009117755584498
30836 - 24.00036342993195504 24.00235983633768555 24.01201669192741761 24.00889679142721533 24.00148470663643963 24.00250682389989976 24.004878
76310545943 24.0017947585754996 24.00663825205647584 24.00009117586036211
30837 - 24.00036342010598034 24.00235977327226422 24.01201632114660001 24.0088965202247735 24.0014846672694766 24.00305924986376027 24.00487862
167855724 24.00124216293889035 24.00663805119506478 24.00009117416615112
30838 - 24.00036341028713593 24.00235971024093118 24.0120159502723709 24.00889624902492869 24.00148462792685677 24.0012778431277809 24.00487848
029837679 24.00302347801129343 24.00663785037423281 24.00009117247324397
30839 - 24.00036389063813402 24.00235964724418025 24.0120155793049328 24.00889597782746065 24.00612595590850873 24.00082469442045152 24.0048783
3897492607 24.00347650334317962 24.00199628229388082 24.00009068061880058
30840 - 24.00036388111386287 24.00235958428147143 24.01201520824387003 24.008895706631872 24.00326157062067267 24.00082467858443636 24.00487819
771053921 24.00347644003430503 24.00486032857707031 24.00009067865596535
30841 - 24.0003638715966261 24.00235952135306405 24.01201483708905471 24.00646327319396178 24.00148439746799056 24.00082465570028489 24.0048780
5648863215 24.00590850456946512 24.006637110788553 24.00009067668958096
30842 - 24.00036386208645922 24.00235945845889418 24.01201446584075327 24.00889504577505917 24.00148444399952297 24.00082463282855372 24.004877
91530924753 24.00536393299466909 24.00474920178696436 24.00009067472473845
30843 - 24.00036385258318461 24.00235939559895115 24.01201409449888757 24.00889491118966745 24.00148440435378561 24.00082460997453992 24.004877
77417968971 24.00477327555725537 24.00533962729210202 24.00009067276130992
30844 - 24.00036384308701543 24.00235933277317102 24.01201372306340076 24.00889464112695038 24.00148436473273961 24.0008245871370498 24.0048776
3309927303 24.00382586086062275 24.00628674253876227 24.00009067079930958
30845 - 24.00190396098307133 24.00235926998154312 24.01201335153436389 24.00889437106732416 24.00148432513629615 24.0008245643152307 24.0048774
9206703114 24.00663646877562485 24.00193576818851326 24.00009066883868059
30846 - 24.00036374060957201 24.0023592072369425 24.01201297991944372 24.0088941010029302 24.00009559386449709 24.00082454149909594 24.00487735
107815368 24.00663630567414231 24.00347578361356682 24.00147935857845383
30847 - 24.00036378133173898 24.00345908412538165 24.01201260820577232 24.0088938309469242 24.00009064889804478 24.00082451870961009 24.0048772
1014367537 24.00663610611074716 24.00237579564506518 24.00148426173993244
30848 - 24.00082463427281709 24.00235900101399267 24.01201223640446614 24.00889356088836024 24.00009064689514915 24.00036363366440995 24.004877
06926104892 24.00663590658722768 24.00347571618236486 24.00148422253626279
30849 - 24.00082461254801913 24.00235893825170308 24.01201186450117575 24.00889329083506496 24.00009064489147903 24.00036362332259543 24.004876
92842629883 24.00663570710280936 24.0034756179779869 24.00148418307191633

Eigenvalores
        24.00082461254801913 24.00235893825170308 24.01201186450117575 24.00889329083506496 24.00009064489147903 24.00036362332259543 24.004876
92842629883 24.00663570710280936 24.0034756179779869 24.00148418307191633
```

Detalles de implementación

Problema 1

El programa se compila con el comando `g++ -std=c++0x main.cpp AlgebraLineal.cpp IOMatrices.cpp -o main -O2`
El programa se ejecuta con el comando `./main A_1000.mtx b_1000.vec`
Se crean los archivos con la matriz inversa y el vector solución

Problema 2

El programa se compila con el comando `g++ -std=c++0x main.cpp AlgebraLineal.cpp IOMatrices.cpp -o main -O2`
El programa se ejecuta con el comando `./main A_5000.mtx b_5000.vec`
Se imprimen los resultados de los métodos iterativo y directo.

Problema 3

El programa se compila con el comando `g++ -std=c++0x main.cpp AlgebraLineal.cpp IOMatrices.cpp -o main -O2`
El programa se ejecuta con el comando `./main`
Se imprimen en pantalla las iteraciones del método y los eigenvalores finales