Álgebra Matricial

Proyecto Parcial 4

Implementación de Eliminación Gaussiana y Método de Jacobi
para la resolución de sistemas de ecuaciones lineales

Reyes Sierra María Margarita

Introducción

La resolución de sistemas de ecuaciones lineales es una necesidad que surge en una amplia variedad de aplicaciones en diversas áreas de investigación, así como en problemas de la industria. Los métodos clásicos usados para este fin son los basados en la reducción gaussiana, así como el método de Gauss-Jordan. Cuando los sistemas son muy grandes, es decir, involucran un alto número de variables, los métodos antes mencionados resultan costosos, computacionalmente hablando. Por esta razón, existen diversos métodos numéricos que trabajan a través de algoritmos iterados mediante los cuales es posible obtener soluciones aproximadas a los sistemas en cuestión.

En este proyecto se comparan las soluciones obtenidas por el método de eliminación gaussiana y el método de Jacobi, reportado en la literatura relacionada [1,2].

Metodología

Se implementó en lenguaje C el algoritmo de eliminación gaussiana [2] y el algoritmo de Jacobi [1], para resolver sistemas de ecuaciones lineales. Se probaron ambos algoritmos en distintos sistemas de ecuaciones, particularmente de 3 y 4 variables (aunque los códigos pueden usarse para sistemas de cualquier dimensión menor que 10).

El método de eliminación gaussiana fue visto en clase, por lo que no se discutirá en este documento. Ahora bien, el método de Jacobi es un método iterativo que inicia con una aproximación x_0 de la solución x del sistema $Ax = b$. El método genera, a través de las iteraciones, una sucesión de vectores $x_k,\ k = 0,1,2,\ldots$ que converge a la solución x.

El algoritmo se muestra en la imagen de la derecha. Como se puede ver, tiene dos criterios de paro posible: la diferencia mínima entre soluciones de dos iteraciones consecutivas (convergencia) o el número máximo de iteraciones. En este caso, el algoritmo implementado se detiene cuando se alcanza el número de iteraciones indicadas por el usuario.

Jacobi Iterative

To solve $A\mathbf{x} = \mathbf{b}$ given an initial approximation $\mathbf{x}^{(0)}$:

INPUT the number of equations and unknowns n; the entries a_{ij}, $1 \le i, j \le n$ of the matrix A; the entries b_i, $1 \le i \le n$ of $\mathbf{b}$; the entries XO_i, $1 \le i \le n$ of $\mathbf{XO} = \mathbf{x}^{(0)}$; tolerance TOL; maximum number of iterations N.

OUTPUT the approximate solution $x_1, \ldots, x_n$ or a message that the number of iterations was exceeded.

Step 1 Set $k = 1$.

Step 2 While $(k \le N)$ do Steps 3–6.

Step 3 For $i = 1, \ldots, n$

$$\text{set } x_i = \frac{-\sum_{\substack{j=1 \\ j \ne i}}^{n} (a_{ij} XO_j) + b_i}{a_{ii}}.$$

Step 4 If $\|\mathbf{x} - \mathbf{XO}\| < TOL$ then OUTPUT $(x_1, \ldots, x_n)$;
(*The procedure was successful.*)
STOP.

Step 5 Set $k = k + 1$.

Step 6 For $i = 1, \ldots, n$ set $XO_i = x_i$.

Step 7 OUTPUT ('Maximum number of iterations exceeded');
(*The procedure was successful.*)
STOP. ■

Como se mencionó anteriormente, el algoritmo de Jacobi requiere un punto de inicio, una "aproximación a la solución", como primer término de la sucesión. Derivado la experimentación realizada, se decidió fijar este punto de inicio, x_0, como el vector 0 para los sistemas mostrados a continuación. Esto se discutirá más adelante con más detalle.

Ambos algoritmos se probaron en diversos sistemas, de los cuales, se muestran los siguientes por razones que se discutirán en la sección siguiente:

Sistemas de 3 variables	**Sistemas de 4 variables**
Sistema 1: $10x_1 + 2x_2 + x_3 = 7$ $x_1 + 5x_2 + x_3 = -8$ $2x_1 + 3x_2 + 10x_3 = 6$ Sistema 2: $10x_1 + x_2 + 2x_3 = 3$ $4x_1 + 6_{x2} - x_3 = 9$ $-2x_1 + 3x_2 + 8x_3 = 51$ Sistema 2: $2x_1 - x_2 + x_3 = 7$ $x_1 + 2_{x2} - x_3 = 6$ $x_1 + x_2 + x_3 = 12$	Sistema 4: $10x_1 - x_2 + 2x_3 = 6$ $-x_1 + 11x_2 - x_3 + 3x_4 = 25$ $2x_1 - x_2 + 10x_3 - x_4 = -11$ $3x_2 - x_3 + 8x_4 = 15$ Sistema 5: $8x_1 - 3x_2 + x_3 - x_4 = 20$ $-x_1 + 20x_2 - 3x_3 + 8x_4 = 3$ $x_1 + x_2 + 3x_3 - x_4 = 3$ $6x_1 - 4x_2 + 11x_3 + 15x_4 = 9$ Sistema 6: $45x_1 + 13x_2 - 4x_3 + 8x_4 = -25$ $-5x_1 - 28x_2 + 4x_3 - 14x_4 = 82$ $9x_1 + 15x_2 + 63x_3 - 7x_4 = 75$ $2x_1 + 3x_2 - 8x_3 - 42x_4 = -43$

Resultados

Los resultados de los algoritmos implementados, para los sistemas mostrados anteriormente, fueron los siguientes:

Algoritmo de reducción gaussiana	Algoritmo de Jacobi
Sistema 1 La matriz es: 10.00 2.00 1.00 7.00 1.00 5.00 1.00 -8.00 2.00 3.00 10.00 6.00 Solucion: x1: 1.000000 x2: -2.000000 x3: 1.000000	Sistema 1 Introduzca el numero de iteraciones: 10 x0: 0.0000 0.0000 0.0000 x1: 0.7000 -1.6000 0.6000 x2: 0.9600 -1.8600 0.9400 x3: 0.9780 -1.9800 0.9660 x4: 0.9994 -1.9888 0.9984 x5: 0.9979 -1.9996 0.9968 x6: 1.0002 -1.9989 1.0003 x7: 0.9998 -2.0001 0.9996 x8: 1.0001 -1.9999 1.0001 x9: 1.0000 -2.0000 1.0000 x10: 1.0000 -2.0000 1.0000
Sistema 2 La matriz es: 10.00 1.00 2.00 3.00 4.00 6.00 -1.00 9.00 -2.00 3.00 8.00 51.00 Solucion: x1: -1.000000 x2: 3.000000 x3: 5.000000	Sistema 2 Introduzca el numero de iteraciones: 13 x0: 0.0000 0.0000 0.0000 x1: 0.3000 1.5000 6.3750 x2: -1.1250 2.3625 5.8875 x3: -1.1137 3.2313 5.2078 x4: -1.0647 3.1105 4.8848 x5: -0.9880 3.0239 4.9424 x6: -0.9909 2.9824 4.9940 x7: -0.9970 2.9929 5.0089 x8: -1.0011 2.9995 5.0034 x9: -1.0006 3.0013 4.9999 x10: -1.0001 3.0004 4.9994 x11: -0.9999 3.0000 4.9998 x12: -1.0000 2.9999 5.0000 x13: -1.0000 3.0000 5.0000

Sistema 3

```
La matriz es:

 2.00 -1.00 1.00 7.00
 1.00 2.00 -1.00 6.00
 1.00 1.00 1.00 12.00

Solucion: x1: 3.000000 x2: 4.000000 x3: 5.000000
```

Sistema 3

```
Introduzca el numero de iteraciones: 68
x0: 0.0000  0.0000  0.0000
x1: 3.5000  3.0000  12.0000
x2: -1.0000  7.2500  5.5000
x3: 4.3750  6.2500  5.7500
x4: 3.7500  3.6875  1.3750
x5: 4.6562  1.8125  4.5625
x6: 2.1250  2.9531  5.5312
x7: 2.2109  4.7031  6.9219
x8: 2.3906  5.3555  5.0859
x9: 3.6348  4.3477  4.2539
x10: 3.5469  3.3096  4.0176
x11: 3.1460  3.2354  5.1436
x12: 2.5459  3.9988  5.6187
x13: 2.6901  4.5364  5.4553
x14: 3.0405  4.3826  4.7736
x15: 3.3045  3.8665  4.5768
x16: 3.1448  3.6362  4.8289
x17: 2.9036  3.8421  5.2190
x18: 2.8115  4.1577  5.2543
x19: 2.9517  4.2214  5.0308
x20: 3.0953  4.0395  4.8269
x21: 3.1063  3.8658  4.8651
x22: 3.0003  3.8794  5.0279
x23: 2.9258  4.0138  5.1203
x24: 2.9468  4.0972  5.0605
x25: 3.0184  4.0569  4.9560
x26: 3.0504  3.9688  4.9248
x27: 3.0220  3.9372  4.9808
x28: 2.9782  3.9794  5.0408
x29: 2.9693  4.0313  5.0424
x30: 2.9944  4.0366  4.9994
x31: 3.0186  4.0025  4.9690
x32: 3.0168  3.9752  4.9789
x33: 2.9981  3.9811  5.0080
x34: 2.9865  4.0050  5.0208
x35: 2.9921  4.0171  5.0085
x36: 3.0043  4.0082  4.9908
x37: 3.0087  3.9932  4.9875
x38: 3.0029  3.9894  4.9980
x39: 2.9957  3.9976  5.0077
x40: 2.9949  4.0060  5.0067
x41: 2.9996  4.0059  4.9990
x42: 3.0034  3.9997  4.9944
x43: 3.0026  3.9955  4.9969
x44: 2.9993  3.9971  5.0019
x45: 2.9976  4.0013  5.0036
x46: 2.9989  4.0030  5.0011
x47: 3.0009  4.0011  4.9982
x48: 3.0015  3.9986  4.9979
x49: 3.0003  3.9982  4.9999
x50: 2.9992  3.9998  5.0014
x51: 2.9992  4.0011  5.0011
x52: 3.0000  4.0009  4.9997
x53: 3.0006  3.9998  4.9990
x54: 3.0004  3.9992  4.9996
x55: 2.9998  3.9996  5.0004
x56: 2.9996  4.0003  5.0006
x57: 2.9998  4.0005  5.0001
x58: 3.0002  4.0001  4.9997
x59: 3.0002  3.9997  4.9997
x60: 3.0000  3.9997  5.0000
x61: 2.9998  4.0000  5.0003
x62: 2.9999  4.0002  5.0002
x63: 3.0000  4.0001  4.9999
x64: 3.0001  3.9999  4.9998
x65: 3.0001  3.9999  4.9999
x66: 3.0000  3.9999  5.0001
x67: 2.9999  4.0001  5.0001
x68: 3.0000  4.0001  5.0000
```

Sistema 4

```
La matriz es:

 10.00 -1.00 2.00 0.00 6.00
 -1.00 11.00 -1.00 3.00 25.00
 2.00 -1.00 10.00 -1.00 -11.00
 0.00 3.00 -1.00 8.00 15.00

Solucion: x1: 1.000000 x2: 2.000000 x3: -1.000000 x4: 1.000000
```

Sistema 4

```
Introduzca el numero de iteraciones: 12
x0: 0.0000  0.0000  0.0000  0.0000
x1: 0.6000  2.2727  -1.1000  1.8750
x2: 1.0473  1.7159  -0.8052  0.8852
x3: 0.9326  2.0533  -1.0493  1.1309
x4: 1.0152  1.9537  -0.9681  0.9738
x5: 0.9890  2.0114  -1.0103  1.0214
x6: 1.0032  1.9922  -0.9945  0.9944
x7: 0.9981  2.0023  -1.0020  1.0036
x8: 1.0006  1.9987  -0.9990  0.9989
x9: 0.9997  2.0004  -1.0004  1.0006
x10: 1.0001  1.9998  -0.9998  0.9998
x11: 0.9999  2.0001  -1.0001  1.0001
x12: 1.0000  2.0000  -1.0000  1.0000
```

Sistema 5

```
La matriz es:

 8.00 -3.00 1.00 -1.00 20.00
 -1.00 20.00 -3.00 8.00 3.00
 1.00 1.00 3.00 -1.00 3.00
 6.00 -4.00 11.00 15.00 9.00

Solucion: x1: 2.618655 x2: 0.378694 x3: -0.092096 x4: -0.278940
```

Sistema 5

```
Introduzca el numero de iteraciones: 17
x0: 0.0000  0.0000  0.0000  0.0000
x1: 2.5000  0.1500  1.0000  0.6000
x2: 2.5062  0.1850  0.3167  -1.0933
x3: 2.3931  0.7601  -0.2615  -0.5854
x4: 2.7446  0.4646  -0.2462  0.0372
x5: 2.7097  0.2354  -0.0573  -0.1934
x6: 2.5713  0.3542  -0.0461  -0.3791
x7: 2.5912  0.4233  -0.1015  -0.3002
x8: 2.6339  0.3844  -0.1049  -0.2492
x9: 2.6261  0.3656  -0.0892  -0.2741
x10: 2.6140  0.3776  -0.0886  -0.2876
x11: 2.6167  0.3824  -0.0930  -0.2799
x12: 2.6201  0.3788  -0.0930  -0.2765
x13: 2.6191  0.3776  -0.0918  -0.2788
x14: 2.6182  0.3787  -0.0919  -0.2796
x15: 2.6185  0.3790  -0.0922  -0.2790
x16: 2.6188  0.3787  -0.0922  -0.2787
x17: 2.6187  0.3786  -0.0921  -0.2789
```

Sistema 6	Sistema 6
La matriz es: 45.00 13.00 -4.00 8.00 -25.00 -5.00 -28.00 4.00 -14.00 82.00 9.00 15.00 63.00 -7.00 75.00 2.00 3.00 -8.00 -42.00 -43.00 Solucion: x1: 0.384800 x2: -2.961872 x3: 1.892936 x4: 0.470012	Introduzca el numero de iteraciones: 10 x0: 0.0000 0.0000 0.0000 0.0000 x1: -0.5556 -2.9286 1.1905 1.0238 x2: 0.2143 -3.1712 2.0809 0.5614 x3: 0.4457 -2.9503 1.9773 0.4111 x4: 0.3994 -2.9313 1.8749 0.4577 x5: 0.3766 -2.9609 1.8822 0.4763 x6: 0.3824 -2.9651 1.8946 0.4717 x7: 0.3856 -2.9621 1.8942 0.4694 x8: 0.3851 -2.9615 1.8928 0.4698 x9: 0.3847 -2.9618 1.8928 0.4701 x10: 0.3848 -2.9619 1.8929 0.4700

Para poder reproducir los resultados mostrados anteriormente, consulte el apéndice al final de este documento.

Discusión de resultados

Como se puede observar, ambos algoritmos encuentran las soluciones a los sistemas mostrados. Sin embargo, una desventaja del algoritmo de Jacobi es el parámetro de máximo número de iteraciones (y no olvidemos que se omitió el parámetro de tolerancia para determinar convergencia). En los resultados se puede ver que, para cada sistema, el número de iteraciones requerido para alcanzar el óptimo fue diferente.

Por otra parte, hay algo muy importante de mencionar. Cuando se estuvieron probando los algoritmos, hubo muchos casos en los que el algoritmo de Jacobi no pudo encontrar la solución, es decir, donde la sucesión de vectores x_k no converge (sin importar el punto de inicio). A raíz de esto, estuve investigando y resulta que el algoritmo sólo converge cuando la matriz de coeficientes cumple cierta condición: ser *diagonalmente dominante*. La condición de ser una matriz diagonalmente dominante significa que los elementos de la diagonal son mayores (en valor absoluto) que la suma de los valores absolutos de los demás elementos del mismo renglón [3,4]. Si la matriz de coeficientes del sistema a resolver no es diagonalmente dominante, el algoritmo de Jacobi no converge, sin importar el punto de inicio. Por el contrario, si la condición se satisface, el algoritmo converge prácticamente sin importar el punto de inicio (por eso se fijó el vector 0 como punto de inicio en todos los casos, de lo contrario habría sido un parámetro más). Cabe mencionar que los sistemas que se usaron para mostrar resultados cumplen con la condición antes mencionada; esto permitió mostrar la convergencia del algoritmo de Jacobi.

Conclusiones y trabajo futuro

Con el fin de comparar dos algoritmos para resolver sistemas de ecuaciones lineales, uno exacto y uno aproximado, se implementaron los algoritmos de reducción gaussiana y Jacobi. Ambos algoritmos son capaces de encontrar la solución de los sistemas; sin embargo, a diferencia de la reducción gaussiana, el algoritmo de Jacobi sólo converge cuando la matriz de coeficientes correspondiente al sistema es diagonalmente dominante. Esta condición limita mucho la aplicación del algoritmo de Jacobi, y es curioso porque no se menciona en los libros de texto comunes. Aunado a esto, dicho algoritmo requiere de 3 parámetros, los cuales, especialmente el número máximo de iteraciones, pueden ser difícil de calibrar.

Lo anterior nos permite concluir que debemos cuidadosos al querer aplicar algún método numérico a cualquier tipo de problema que queramos resolver, pues debemos siempre investigar antes sus condiciones de convergencia, así como los parámetros requeridos.

Como actividad a futuro, se contempla la experimentación con sistemas de ecuaciones con un mayor número de variables; esto con la finalidad de llevar a cabo un análisis sobre el tiempo de ejecución, pues en los experimentos realizados la diferencia entre ambos algoritmos fue imperceptible. Probablemente, cuando el número de variables aumenta, las ventajas del algoritmo de Jacobi pueden mostrarse más claramente.

Referencias

[1] Burden, R., Faires, D. (2002), **Análisis numérico** (7a. edición), Thomson Learning.
[2] Grossman, S. I. (2008). **Álgebra Lineal** (6a. edición). McGraw-Hill Interamericana.
[3] https://hmong.es/wiki/Diagonally_dominant_matrix
[4] https://es.wikipedia.org/wiki/Matriz_de_diagonal_estrictamente_dominante

Apéndice. Guía para la reproducción de resultados.

Para poder reproducir los resultados mostrados en este reporte deben seguirse las siguientes instrucciones. Tome en cuenta que su equipo debe tener instalado un compilador de lenguaje C, si no cuenta con uno, descárguelo de https://visualstudio.microsoft.com/es/vs/features/cplusplus/.

1. gauss.c (algoritmo de reducción gaussiana)

Para compilar:

```
gcc gauss.c -o gauss -lm
```

Para ejecutar:

```
./gauss
```

Cuando se ejecute, pedirá el número de variables y posteriormente el valor de cada entrada de la matriz de coeficientes (según el sistema a resolver). El número de variables máximo permitido es 10. Ejemplo:

```
Introduzca el numero de variables:3
Introduzca a11:2
Introduzca a12:-1
Introduzca a13:1
Introduzca a14:7
Introduzca a21:1
Introduzca a22:2
Introduzca a23:-1
Introduzca a24:6
Introduzca a31:1
Introduzca a32:1
Introduzca a33:1

La matriz es:

 2.00 -1.00 1.00 7.00
 1.00 2.00 -1.00 6.00
 1.00 1.00 1.00 12.00

Solucion: x1: 3.000000 x2: 4.000000 x3: 5.000000
```

2. jacobi.c (algoritmo de Jacobi)

Para compilar:

```
gcc jacobi.c -o jacobi -lm
```

Para ejecutar:

```
./jacobi
```

Cuando se ejecute, pedirá el número de variables y posteriormente el valor de cada entrada de la matriz de coeficientes (según el sistema a resolver). Ejemplo:

```
Introduzca el numero de variables: 3
Introduzca a11:10
Introduzca a12:1
Introduzca a13:2
Introduzca a14:3
Introduzca a21:4
Introduzca a22:6
Introduzca a23:-1
Introduzca a24:9
Introduzca a31:-2
Introduzca a32:3
Introduzca a34:51

La matriz es:

 10.00 1.00 2.00 3.00
 4.00 6.00 -1.00 9.00
 -2.00 3.00 8.00 51.00
```

Posteriormente le pedirá el número máximo de iteraciones y mostrará los resultados correspondientes:

```
Introduzca el numero de iteraciones: 10
x0: 0.0000  0.0000  0.0000
x1: 0.3000  1.5000  6.3750
x2: -1.1250  2.3625  5.8875
x3: -1.1137  3.2313  5.2078
x4: -1.0647  3.1105  4.8848
x5: -0.9880  3.0239  4.9424
x6: -0.9909  2.9824  4.9940
x7: -0.9970  2.9929  5.0089
x8: -1.0011  2.9995  5.0034
x9: -1.0006  3.0013  4.9999
x10: -1.0001  3.0004  4.9994
```