Taller Ecuaciones Lineales

Jiménez Nelson, Velandia Joan

16 de agosto de 2019

Análisis de la sensibilidad

El objetivo es identificar el impacto que resulta en los resultados del problema original luego de determinadas variaciones en los parámetros, variables o restricciones del modelo, sin que esto pase por resolver el problema nuevamente, para ello se utiliza el inverso del número de condición.

Para la solución se uso la siguiente matriz:

```
##      [,1]      [,2]
## [1,]  3.5 4.2000000
## [2,]  0.5 0.3333333
```

```
## [1] 1.30 1.75
```

Se realizó un cambio a la variable $x_{1,1}$

el resultado del nuevo número de condición de la matriz:

```
##      [,1]      [,2]
## [1,]  3.8 4.5000000
## [2,]  0.5 0.3333333
```

fue:

```
## [1] 40.79661
```

Con un error de:

```
## [1] -2.185533
```

Al ser el número condicional mayor a 1, se dice que la matriz está mal condicionada.

Gauss-Jordan en matrices ralas

El método de Gauss-Jordan es un método directo, afectado por errores de redondeo, además, es un proceso muy lento debido al manejo que hace a la memoria RAM al almacenar y operar la matriz. Se hicieron tres pruebas al algoritmo de Gauss-jordan con matrices ralas de tamaño $100 * 100$, $500 * 500$, y $1000 * 1000$ empezando a tomar bastante tiempo a partir de está última prueba.

Descomposición LU

La descomposición por el método de LU permite resolver diferentes matrices de la forma $A\mathbf{X} = b$ con distintos vectores b, sin necesidad de eliminación gaussiana. Se requiere una matriz L con la parte triangular superior y su diagonal en valores 1, y su contrario U con la parte triangular inferior, de tal manera que en nuestro ejemplo sucedería así:

$$L = \begin{pmatrix} 1 & 0 \\ 0.5 & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} 3.5 & 4.2 \\ 0.0 & 1/3 \end{pmatrix}$$

resolvemos:

```
##           [,1]       [,2]
## [1,] 3.5000000  4.2000000
## [2,] 0.1428571 -0.2666667
```

Solucionamos:

```
## [1]  7.410714 -5.866071
```

Factorización de una matriz

El método de Jacobi, es un método iterativo simple pero lento que no permite llegar a la solucion de una matriz de la siguiente forma:

Para el sistema: $A\mathbf{x} = b$

A se reescribe como $A = D + L + U$

Reemplazando

$D + L + U\mathbf{X} = b$

$DX + LX + UX = b$

por lo tanto:

$X = D^{-1}B - D^{-1}LX - UX$

$X = D^{-1}B - D^{-1}(L + U)X$ análogo a método de punto fijo $X = G(X)$

Para nuestra matriz ejemplo y con una tolerancia de 5 iteraicones:

```
##      [,1]  [,2]   [,3] [,4]
## [1,] -8.1 -7.00  6.123 -2.0
## [2,] -1.0  4.00 -3.000 -1.0
## [3,]  0.0 -1.00 -5.000  0.6
## [4,] -1.0  0.33  6.000  0.5
```

obtenemos una solución:

```
## Error  0   4.118073
## Error  1   1.014324
## Error  2   2.830197
## Error  3   1.268594
## Error  4   2.063461
## Solución a 5 iteraciones:  -4.627352 -1.394185 -4.90428 -89.51575
```

No obstante la solución que nos ofrece R con su función solve(...) es:

```
##              [,1]
## [1,] -1.091148268
## [2,]  0.002898634
## [3,] -0.920772493
## [4,]  0.865060281
```

Esto debido a la convergencia del método que fue de:

```
## [1] 1.6026
```

Con un número de condición:

```
## [1] 24.18476
```

Diagonalización

La diagonalización de una matriz hace posible calcular cualquier función que este definida sobre el espectro de la matriz tales como: potencia de A o el e

Para la diagonalización se descompone la matriz de la siguiente forma: $A = PDP^{-1}$, donde P son los vectores propios de A y D es su diagonal de los vectores propios. Por ejemplo, para encontrar la k-ésima potencia de A, basta con descomponerla de la siguiente manera: $A^k = PD^kP^{-1}$; Realizar esta descomposición disminuye el uso de recursos de procesamiento y almacenamiento, aumentando su eficiencia.

Para la siguiente matriz A:

```
##      [,1] [,2] [,3]
## [1,]    1    2    4
## [2,]    2    1   -4
## [3,]    0    0    3
```

sus vectores P

```
##           [,1]        [,2]       [,3]
## [1,] 0.7071068  0.02616462 -0.7071068
## [2,] 0.7071068 -0.88882680  0.7071068
## [3,] 0.0000000  0.45749571  0.0000000
```

su diagonal

```
##      [,1] [,2] [,3]
## [1,]    3    0    0
## [2,]    0    3    0
## [3,]    0    0   -1
```

Y finalmente A^3

```
##      [,1] [,2] [,3]
## [1,]   13   14   28
## [2,]   14   13  -28
## [3,]    0    0   27
```