Métodos numéricos para matrices

Curso de Física Computacional

M. en C. Gustavo Contreras Mayén

Facultad de Ciencias - UNAM

19 de abril de 2018

Métodos de descomposición **LU**

Se puede demostrar que cualquier matriz cuadrada $\mathbf{A}$ se puede expresar como un producto de una matriz triangular inferior $\mathbf{L}$ y una matriz triangular superior $\mathbf{U}$:

$$\mathbf{A} = \mathbf{L}\,\mathbf{U}$$

Factorización **LU**

El proceso de calcular **L** y **U** para un determinada matriz **A**, se conoce como descomposición LU o factorización LU.

Descomposición LU

La descomposición $\mathbf{LU}$ no es única (las combinaciones de $\mathbf{L}$ y $\mathbf{U}$ para una determinada matriz A son infinitas), salvo ciertas restricciones de $\mathbf{L}$ o $\mathbf{U}$.

Estas limitaciones distinguen un tipo de descomposición de otro.

Métodos LU más utilizados

Los tres métodos de descomposición LU más utilizados son:

Nombre	Restricciones
Doolittle	$L_{ii} = 1, \quad i = 1, 2, \ldots, n$
Crout	$U_{ii} = 1, \quad i = 1, 2, \ldots, n$
Choleski	$L = U^T$

Versatilidad de los métodos

Después de descomponer a $\mathbf{A}$, se facilita resolver las ecuaciones $A\,x = b$.

En primer lugar, volvemos a escribir las ecuaciones como $L\,U\,x = b$.

Versatilidad de los métodos

Al usar la notación $U\ x = y$, las ecuaciones quedan

$$\mathbf{L}\,\mathbf{y} = \mathbf{b}$$

que se puede resolver para y por sustitución hacia delante.

Versatilidad de los métodos

Entonces

$$\mathbf{U}\,\mathbf{x} = \mathbf{y}$$

nos devolverá x con el proceso de sustitución hacia atrás.

Versatilidad de los métodos

La ventaja de la descomposición LU sobre el método de eliminación de Gauss es que una vez descompuesta, podemos resolver $Ax = b$ para muchos vectores constantes b.

Versatilidad de los métodos

El costo de cada solución adicional es relativamente pequeño, ya que el avance y las operaciones de sustitución están consumiendo mucho menos tiempo que el proceso de descomposición.

Descomposición de Doolittle

El método de descomposición de Doolittle está estrechamente relacionado con el proceso de eliminación de Gauss.

Veamos el método con un ejemplo:

Descomposición de Doolittle

Considera una matriz A de 3×3 y suponganmos que existen las matrices triangulares:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{bmatrix} \qquad U = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

tales que $\mathbf{A} = \mathbf{LU}$.

Descomposición de Doolittle

Después de realizar la multiplicación del lado derecho de $\mathbf{A} = \mathbf{LU}$, tenemos que:

$$\mathbf{A} = \begin{matrix} U_{11} & U_{12} & U_{13} \\ U_{11}L_{21} & U_{12}L_{21} + U_{22} & U_{13}L_{21} + U_{23} \\ U_{11}L_{31} & U_{12}L_{31} + U_{22}L_{32} & U_{13}L_{31} + U_{23}L_{32} + U_{33} \end{matrix}$$

Descomposición de Doolittle

Apliquemos ahora la eliminación de Gauss a la ecuación anterior.

El primer paso de la eliminación consiste en la elección de la primera fila como la fila pivote y la aplicación de las operaciones elementales:

Descomposición de Doolittle

$$\text{renglón } 2 \leftarrow \text{renglón } 2 - L_{21} \times \text{renglón } 1 \quad (\text{elimina } A_{21})$$

$$\text{renglón } 3 \leftarrow \text{renglón } 3 - L_{31} \times \text{renglón } 1 \quad (\text{elimina } A_{31})$$

El resultado es

$$\mathbf{A}' = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & U_{22}L_{32} & U_{23}L_{32} + U_{33} \end{bmatrix}$$

Descomposición de Doolittle

El siguiente paso es tomar la segunda fila como pivote y utilizar la operación:

$$\text{renglón } 3 \quad \leftarrow \quad \text{renglón } 3 - L_{32} \times \text{renglón } 2 \quad (\text{elimina } A_{32})$$

dejando al final:

$$\mathbf{A}'' = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

Descomposición de Doolittle

Del ejemplo vemos dos características importantes del proceso de descomposición de Doolittle:

1. La matriz $\mathbf{U}$ es idéntica a la matriz triangular inferior que resulta del proceso de eliminación de Gauss.

Descomposición de Doolittle

2. Los elementos fuera de la diagonal de $\mathbf{L}$ son multiplicadores de la ecuación pivote que se utilizan durante la eliminación de Gauss, es decir, L_{ij} es el multiplicador que elimina el elemento A_{ij}.

Descomposición de Doolittle

Es una práctica habitual almacenar los multiplicadores en la parte triangular inferior de la matriz de coeficientes, re-emplazando los coeficientes ya que se eliminaron (L_{ij} sustituye a A_{ij})

Descomposición de Doolittle

Los elementos diagonales de L no tienen que guardarse, ya que se entiende que cada uno de ellos es la unidad.

La forma final de la matriz de coeficientes sería una mezcla de L y U:

$$[\mathbf{L}\backslash\mathbf{U}] = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ L_{21} & U_{22} & U_{23} \\ L_{31} & L_{32} & U_{33} \end{bmatrix}$$

Fase de eliminación Doolittle

El algoritmo para la descomposición de Doolittle es idéntico al procedimiento de eliminación de Gauss, excepto por que cada multiplicador λ se almacena en la parte triangular inferior de A.

Algortimo de Doolittle

Código 1: Código para obtener las matrices LU

```
1 for k in range(0, n-1):
2     for i in range(k+1, n):
3         if a[i,k] != 0.0:
4             lam = a[i, k]/a[k, k]
5             a[i, k+1:n] = a[i, k+1:n
    ] - lam * a[k, k+1:n]
6             a[i,k] = lam
```

Fase de solución Doolittle

Considera ahora la fase de solución $Ly = b$ por sustitución hacia adelante. La forma escalar de las ecuaciones es (recuerda que $L_{ii} = 1$):

$$
\begin{aligned}
y_1 &= b_1 \\
L_{21}y_1 + y_2 &= b_2 \\
&\vdots \\
L_{k1}y_1 + L_{k2}y_2 + \ldots + L_{k,k+1}y_{k-1} + y_k &= b_k \\
&\vdots
\end{aligned}
$$

Fase de solución Doolittle

Resolviendo la k-ésima ecuación, tenemos

$$y_k = b_k - \sum_{j=1}^{k-1} L_{kj} y_j \qquad k = 2, 3, \ldots, n$$

Código 2: Fase de solución

```
1 y[0] = b[0]
2 for k in range(1, n):
3     y[k] = b[k] - dot(a[k, 0:k], y[
    0:k])
```

Ejercicio 1

Resuelve con el método de Doolittle, $A\,x = b$ donde

$$A = \begin{bmatrix} -3 & 6 & -4 \\ 9 & -8 & 24 \\ -12 & 24 & -26 \end{bmatrix} \qquad b = \begin{bmatrix} -3 \\ 65 \\ -42 \end{bmatrix}$$

Ejercicio 2. Para entregar.

Resuelve con el método de Doolittle, $A\,X = B$ donde

$$A = \begin{bmatrix} 4 & -3 & 6 \\ 8 & -3 & 10 \\ -4 & 12 & -10 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Descomposición de Choleski

La descomposición de Choleski $\mathbf{A} = \mathbf{L}\,\mathbf{L}^T$ tiene dos limitaciones:

1. Dado que $\mathbf{L}\,\mathbf{L}^T$ devuelve siempre una matriz simétrica, la descomposición de Choleski requiere que la matriz $\mathbf{A}$ sea simétrica.

Descomposición de Choleski

2. El proceso de descomposición implica tomar raíces cuadradas de ciertas combinaciones de la elementos de $\mathbf{A}$.

   Se puede demostrar que, para evitar los valores negativos de las raíces, la matriz $\mathbf{A}$ debe de ser definita positiva.

Definición de matriz definida positiva

Se dice que una matriz $\mathbf{M}$ es definida positiva, si para todo vector no nulo x, se tiene que

$$x^T \, M \, x > 0$$

cumpliendo esto, también se cumple que, todos los eigenvalores de M, son positivos (definición alterna).

Desventaja del método de Choleski

Aunque el número de multiplicaciones en todos los métodos de descomposición es el mismo, la descomposición de Choleski no es tan popular en la solución de ecuaciones simultáneas, debido a las restricciones mencionadas anteriormente.

El proceso de factorización de Choleski

Consideremos la matriz $\mathbf{A}$ de 3×3

$$\mathbf{A} = \mathbf{L}\,\mathbf{L}^{\mathbf{T}}$$

tal que

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} L_{11} & L_{21} & L_{31} \\ 0 & L_{22} & L_{32} \\ 0 & 0 & L_{33} \end{bmatrix}$$

Factorización de Choleski

Luego de resolver la multiplicación del lado derecho, resulta

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} =$$

$$= \begin{bmatrix} L_{11}^2 & L_{11}L_{21} & L_{11}L_{31} \\ L_{11}L_{21} & L_{21}^2 + L_{22}^2 & L_{21}L_{31} + L_{22}L_{32} \\ L_{11}L_{31} & L_{21}L_{31} + L_{22}L_{32} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{bmatrix}$$

Factorización de Choleski

Vemos que la matriz del lado derecho es simétrica.

$$\begin{bmatrix} L_{11}^2 & L_{11}L_{21} & L_{11}L_{31} \\ L_{11}L_{21} & L_{21}^2 + L_{22}^2 & L_{21}L_{31} + L_{22}L_{32} \\ L_{11}L_{31} & L_{21}L_{31} + L_{22}L_{32} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{bmatrix}$$

Factorización de Choleski

Al igualar las matrices $\mathbf{A}$ y $\mathbf{L}\,\mathbf{L}^T$ elemento a elemento, tendremos seis ecuaciones (revisando que hay simetría en los elementos triangulares superiores e inferiores)

Factorización de Choleski

Consideremos la parte triangular inferior de cada matriz, al igualar los elementos de la primera columna, empezando por la primera fila y hacia abajo:

Factorización de Choleski

Calculando L_{11}, L_{21} y L_{31} en ese orden:

$$A_{11} = L_{11}^2 \quad \rightarrow \quad L_{11} = \sqrt{A_{11}}$$

$$A_{21} = L_{11}L_{21} \quad \rightarrow \quad L_{21} = \frac{A_{21}}{L_{11}}$$

$$A_{31} = L_{11}L_{31} \quad \rightarrow \quad L_{31} = \frac{A_{31}}{L_{11}}$$

Factorización de Choleski

De la segunda columna, iniciamos con la segunda fila, para obtener L_{22} y L_{32}:

$$A_{22} = L_{21}^2 + L_{22}^2 \quad \rightarrow \quad L_{22} = \sqrt{A_{22} - L_{21}^2}$$

$$A_{32} = L_{21}L_{31} + L_{22}L_{32} \quad \rightarrow \quad L_{32} = \frac{A_{32} - L_{21}L_{31}}{L_{22}}$$

Factorización de Choleski

Finalmente con la tercera columna y de la tercera fila, obtenemos L_{33}:

$$A_{33} = L_{31}^2 + L_{32}^2 + L_{33}^2 \rightarrow L_{33} = \sqrt{A_{33} - L_{31}^2 - L_{32}^2}$$

Factorización de Choleski

Podemos extrapolar los resultados para una matriz de $n \times n$.

Para un elemento en la parte triangular inferior de $\mathbf{L}\mathbf{L}^T$ resulta:

$$\begin{aligned}\mathbf{L}\mathbf{L}^T_{ij} =& L_{i1}L_{j1} + L_{i2}L_{j2} + \ldots + L_{ij}L_{jj} = \\ =& \sum_{k=1}^{j} L_{ik}L_{jk} \qquad\qquad i \geq j\end{aligned}$$

Factorización de Choleski

Igualando los términos para los correspondientes elementos de $\mathbf{A}$:

$$A_{ij} = \sum_{k=1}^{j} L_{ik} K_{jk} \qquad i = j, j+1, \ldots, n, \qquad j = 1, 2, \ldots,$$

El intervalo de índices de los elementos mostrados limita a la parte triangular inferior.

Factorización de Choleski

Para la primera columna ($j = 1$), obtenemos de la ecuación anterior:

$$L_{11} = \sqrt{A_{11}} \qquad L_{i1} = \frac{A_{1j}}{L_{11}}, \quad i = 2, 3, \ldots, n$$

Continuando con las otras columnas, vemos que la incógnita en la ecuación es L_{ij} (los otros elementos de $\mathbf{L}$ que aparecen en la ecuación, ya han sido calculados).

Factorización de Choleski

Tomando el término que incluye a L_{ij} fuera de la suma de la ecuación para los elementos de $\mathbf{A}$ tenemos:

$$A_{ij} = \sum_{k=1}^{j-1} L_{ik} L_{jk} + L_{ij} L_{jj}$$

Factorización de Choleski

Si $i = j$ (un elemento de la diagonal), la solución es:

$$L_{jj} = \sqrt{A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2} \qquad j = 2, 3, \ldots, n$$

Factorización de Choleski

Para un elemento que no está en la diagonal:

$$L_{ij} = \frac{\left(A_{ij} - \sum_{k=1}^{j-1} L_{ik}\, L_{jk} \right)}{L_{jj}}$$

donde

$$j = 2, 3, \ldots, n-1, \quad i = j+1, j+2, \ldots, n$$

Implementación del código

Antes de presentar el algoritmo de descomposición de Choleski, hacemos una observación útil: A_{ij} aparece sólo en la fórmula de L_{ij}.

Implementación del código

Por lo tanto, una vez que L_{ij} se ha calculado, A_{ij} ya no se necesita.

Esto hace que sea posible escribir los elementos de $\mathbf{L}$ sobre la parte triangular inferior de $\mathbf{A}$, cuando se calculan.

Implementación del código

Los elementos de la diagonal principal de $\mathbf{A}$ permanecerán intactos.

Si se encuentra un elemento negativo en la diagonal durante la descomposición, se presenta un mensaje de error y el programa termina.

Código 3: Algoritmo para la factorización con Choleski

```
1  def choleski(a):
2      n = len(a)
3      for k in range(n):
4          try:
5              a[k,k] = sqrt(a[k,k] -
       dot(a[k,0:k],a[k,0:k]))
6          except ValueError:
7              err('La matriz no es
       definida positiva')
8          for i in range(k+1,n):
9              a[i,k] = (a[i,k] - dot(a
       [i,0:k],a[k,0:k]))/a[k,k]
10     for k in range(1,n): a[0:k,k] =
       0.0
11     return a
```

Pendientes para revisión del código

Como se hizo con la descomposición de Doolittle, se han incluido las rutinas para sustituir hacia adelante y hacia atrás para obtener la solución final del sistema.

Revisa el código correspondiente del módulo.

Ejercicio 1

Resuelve el siguiente sistema mediante la factorización de Choleski

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 3/2 \\ 3 \end{bmatrix}$$

Ejercicio 1

Al usar el algoritmo de la descomposición de Choleski, tenemos las matrices LL^T

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

$$L^T = \begin{pmatrix} 1. & 1. & 1. \\ 0. & 1. & 1. \\ 0. & 0. & 1. \end{pmatrix}$$

Ejercicio 1

Ahora bien, hay que resolver primero en sustitución hacia adelante el sistema $L * b = c$

$$c = \begin{pmatrix} 1. & 0. & 0. \\ 1. & 1. & 0. \\ 1. & 1. & 1. \end{pmatrix} \begin{pmatrix} 1 \\ 3/2 \\ 3 \end{pmatrix}$$

donde $c = [1, 0.5, 1.5]$

Ejercicio 1

Para finalizar, hacemos ahora la sustitución hacia adelante con el sistema $L^T * x = c$

$$x = \begin{pmatrix} 1. & 1. & 1. \\ 0. & 1. & 1. \\ 0. & 0. & 1. \end{pmatrix} \begin{pmatrix} 1.0 \\ 0.5 \\ 1.5 \end{pmatrix}$$

Y el resultado es: $x = [0.5, -1., 1.5]$

Ejercicio 2. Para entregar.

Prueba la función Choleski para descomponer la siguiente matriz

$$\mathbf{A} = \begin{bmatrix} 1.44 & -0.36 & 5.52 & 0.00 \\ -0.36 & 10.33 & -7.78 & 0.0 \\ 5.52 & -7.78 & 28.40 & 9.00 \\ 0.00 & 0.00 & 9.00 & 61.00 \end{bmatrix}$$

Sugerencia para la solución

Verifica previamente que los valores propios de la matriz, son todos positivos, usando la respectiva función de `python` que nos devuelve los *eigenvalores*.

Sugerencia para la solución

Verifica previamente que los valores propios de la matriz, son todos positivos, usando la respectiva función de `python` que nos devuelve los *eigenvalores*.

Posteriormente revisa si recuperamos la matriz inicial $\mathbf{A}$ al multiplicar por la matriz transpuesta, es decir $\mathbf{L} * \mathbf{L}^T$.

Coeficientes simétricos y en banda

Algunos problemas en física e ingeniería plantean la necesidad de trabajar con matrices *escasamente pobladas*, en inglés *sparse*, donde la gran mayoría de los elementos de la matriz, son cero.

Coeficientes simétricos y en banda

Si todos los elementos no nulos de la matriz se ubican sobre la diagonal principal, se dice entonces que la matriz es una *matriz en banda*.

Matriz tridiagonal

Sea la matriz

$$A = \begin{bmatrix} X & X & 0 & 0 & 0 \\ X & X & X & 0 & 0 \\ 0 & X & X & X & 0 \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{bmatrix}$$

Matriz tridiagonal

En donde X indica un elemento no nulo, dando la forma de una banda (considera que algunos de éstos elementos, aún así, pueden ser cero).

Todos los demás elementos fuera de la banda, son nulos.

Propiedades de las matrices **L** y **U**

Si una matriz en banda se descompone de la forma $\mathbf{A} = \mathbf{LU}$, donde $\mathbf{L}$ y $\mathbf{U}$ mantienen la estructura en banda de $\mathbf{A}$, por ejemplo, si descomponemos la matriz mostrada anteriormente, obtendremos:

Propiedades de las matrices **L** y **U**

$$\mathbf{L} = \begin{bmatrix} X & 0 & 0 & 0 & 0 \\ X & X & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} X & X & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{bmatrix}$$

Propiedades de las matrices **L** y **U**

La estructura en banda de una matriz de coeficientes pues aprovecharse para guardar valores y reducir el tiempo de cálculo.

Si la matriz es de coeficientes, y además es simétrica, la economía de operaciones sobre la misma, es mayor.

Matriz de coeficientes tridiagonal

Considera la solución de un sistema $\mathbf{A}x = \mathbf{b}$ mediante la descomposición de Doolittle, donde $\mathbf{A}$ es una matriz triadiagonal de $n \times n$

Matriz de coeficientes tridiagonal

$$\mathbf{A} = \begin{bmatrix} d_1 & e_1 & 0 & 0 & \dots & 0 \\ c_1 & d_2 & e_2 & 0 & \dots & 0 \\ 0 & c_2 & d_3 & e_3 & \dots & 0 \\ 0 & 0 & c_3 & d_4 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & c_{n-1} & d_n \end{bmatrix}$$

Matriz de coeficientes tridiagonal

Como la notación lo indica, podemos almacenar los elementos no nulos de $\mathbf{A}$ en los vectores

$$\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} \qquad \mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \qquad \mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n-1} \end{bmatrix}$$

Matriz de coeficientes tridiagonal

El ahorro resultante de almacenamiento puede ser significativo.

Por ejemplo, una matriz tridiagonal de 100×100, contiene 10000 elementos, que pueden almacenarse en solo $99 + 100 + 99 = 298$ entradas, lo cual representa una compresión de $33 : 1$.

Descomposición LU

Apliquemos ahora la descomposición $\mathbf{L}\,\mathbf{U}$ a la matriz de coeficientes.

Podemos reducir el renglón k eliminando c_{k-1} con la operación elemental

$$\text{renglón } k \leftarrow \text{ renglón } k - \left(\frac{c_{k-1}}{d_{k-1}}\right) \times \text{ renglón } (k-1),$$

Descomposición LU

El cambio en d_k es

$$d_k \leftarrow d_k - \left(\frac{c_{k-1}}{d_{k-1}}\right) e_{k-1}$$

donde e_k no se ve afectado.

Descomposición LU

Para finalizar la descomposición de Doolittle de la forma $[\mathbf{L}\backslash\mathbf{U}]$, guardamos el multiplicador $\lambda = c_{k-1}/d_{k-1}$, en la posición previamente ocupada por c_{k-1}:

$$c_{k-1} \leftarrow \frac{c_{k-1}}{d_{k-1}}$$

Algoritmo de descomposición

Código 4: Descomposición LU

```
1 for k in range(1, n):
2     lam = c[k-1]/d[k-1]
3     d[k] = d[k] - lam * e[k -1]
4     c[k-1] = lam
```

Fase de solución

La fase de solución viene dada por $\mathbf{L}y = \mathbf{b}$, seguida por $\mathbf{U}x = \mathbf{y}$.

Las ecuaciones $\mathbf{L}y = \mathbf{b}$ pueden interpretarse como la matriz de coeficientes aumentada.

Fase de solución

$$[\mathbf{L}\backslash\mathbf{b}] = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & b_1 \\ c_1 & 1 & 0 & 0 & \dots & 0 & b_2 \\ 0 & c_2 & 1 & 0 & \dots & 0 & b_3 \\ 0 & 0 & c_3 & 1 & \dots & 0 & b_4 \\ \vdots & \vdots & \vdots & \vdots & \dots & 0 & \vdots \\ 0 & 0 & \dots & 0 & c_{n-1} & 1 & b_n \end{bmatrix}$$

Fase de solución

Nótese que los valores originales de $\mathbf{c}$, se destruyen y se reemplazan por los multiplicadores durante la descomposición.

Fase de solución

El algoritmo de solución para y, por sustitución hacia adelante es:

Código 5: Sustitución hacia adelante

```
1 y[0] = b[0]
2 for k in range(1, n):
3     y[k] = b[k] - c[k-1] * y[k-1]
```

Fase de solución

La matriz de coeficientes aumentada, representada por $\mathbf{U}x = \mathbf{y}$ es

$$[\mathbf{U}\backslash\mathbf{y}] = \begin{bmatrix} d_1 & e_1 & 0 & \dots & 0 & 0 & y_1 \\ 0 & d_2 & e_2 & \dots & 0 & 0 & y_2 \\ 0 & 0 & d_3 & \dots & 0 & 0 & y_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & d_{n-1} & e_{n-1} & y_{n-1} \\ 0 & 0 & 0 & \dots & 0 & d_n & y_n \end{bmatrix}$$

Fase de solución

Revisemos que los valores de d se modificaron durante la fase de descomposición (pero los valores de e se mantienen).

Fase de solución

La solución para $\mathbf{x}$ se obtiene con sustitución hacia atrás, con el algoritmo

Código 6: Sustitución hacia atrás

```
1 x[n-1] = y[n-1]/d[n-1]
2 for k in range(n-2, -1, -1):
3     x[k] = (y[k] - e[k] * x[k+1]) /
    d[k]
```

Algortimo LUdescomp3

El siguiene módulo contiene las funciones **`LUdescomp3`** y **`LUsoluc3`** para la fase de descomposición y solución para una matriz tridiagonal.

Algortimo LUdescomp3

En **`LUsoluc3`**, el vector y se escribe sobre el vector constante b en la fase de sustitución hacia atrás.

De manera análoga el vector x se sobreescribe en y durante la sustitución hacia atrás. En b se almacena la solución que devuelve **`LUsoluc3`**.

Algoritmo I

Código 7: Algoritmo para la solución tridiagonal

```
1  def LUdescomp3(c, d, e):
2      n = len(d)
3      for k in range(1, n):
4          lam = c[k-1]/d[k-1]
5          d[k] = d[k] - lam * e[k-1]
6          c[k-1] = lam
7      return c, d, e
8
9  def LUsoluc3(c, d, e, b):
10     n = len(d)
11
12     for k in range(1, n):
```

Algoritmo II

```
13          b[k] = b[k] - c[k-1] * b[k-
    1]
14      b[n-1] = b[n-1]/d[n-1]
15
16      for k in range(n-2, -1, -1):
17          b[k] = (b[k] - e[k] * b[k+
    1])/d[k]
18      return b
```

Ejemplo

Usando las funciones **`LUdescomp3`** y **`LUsoluc3`**, resuelve el sistema $\mathbf{A}\,\mathbf{x} = \mathbf{b}$, donde

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 5 \\ -5 \\ 4 \\ -5 \\ 5 \end{bmatrix}$$

¿Qué necesitamos?

Se requiere almacenar los arreglos $\mathbf{c}$, $\mathbf{d}$ y $\mathbf{e}$ para utilizarlos, hay que tomar en cuenta que en particular, la matriz $\mathbf{A}$ de nuestro ejercicio es:

1. Simétrica.

¿Qué necesitamos?

Se requiere almacenar los arreglos $\mathbf{c}$, $\mathbf{d}$ y $\mathbf{e}$ para utilizarlos, hay que tomar en cuenta que en particular, la matriz $\mathbf{A}$ de nuestro ejercicio es:

1. Simétrica.
2. Tridiagonal.

¿Qué necesitamos?

Se requiere almacenar los arreglos $\mathbf{c}$, $\mathbf{d}$ y $\mathbf{e}$ para utilizarlos, hay que tomar en cuenta que en particular, la matriz $\mathbf{A}$ de nuestro ejercicio es:

1. Simétrica.
2. Tridiagonal.
3. Los elementos de $\mathbf{c}$, $\mathbf{d}$ y $\mathbf{e}$ son los mismos, lo que no quiere decir que así sean todos los problemas, pero nos facilita para éste ejercicio la manera en que podemos crearlos.

Creando los vectores columna

Del Tema 0 que presentamos al inicio del curso, se revisaron algunas funciones particulares para el manejo de arreglos, una de ellas es la función **`ones`** de la librería `numpy`.

Lo que nos devuelve la función **`ones(n)`** es un arreglo de n elementos, donde todos ellos valen 1.

Creando los vectores columna

Nuestra tarea será ahora modificar este arreglo para nuestro ejercicio.

Nota: En caso de que los elementos de los vectores $\mathbf{c}$, $\mathbf{d}$ y $\mathbf{e}$, sean diferentes, la construcción será diferente, por no decir “a mano”.

Código

Código 8: Código completo para la solución tridiagonal

```
1 d = ones(5)*2.0
2 c = ones(4)*(-1.0)
3 b = array([5.0, -5.0, 4.0, -5.0, 5.0
    ])
4 e = c.copy()
5
6 c, d, e = LUdescomp3(c,d,e)
7
8 x = LUsoluc3(c,d,e,b)
9 print ('x = ',x)
```

Código

Código 9: Código completo para la solución tridiagonal

```
1 d = ones(5)*2.0
2 c = ones(4)*(-1.0)
3 b = array([5.0, -5.0, 4.0, -5.0, 5.0
    ])
4 e = c.copy()
5
6 c, d, e = LUdescomp3(c,d,e)
7
8 x = LUsoluc3(c,d,e,b)
9 print ('x = ',x)
```

La solución es: $x = [2. \ -1. \ 1. \ -1. \ 2.]$