Método de Diferencias Finitas para ecuaciones elípticas

Julio A. Medina
Universidad de San Carlos
Escuela de Ciencias Físicas y Matemáticas
Maestría en Física
julioantonio.medina@gmail.com

1. Ecuaciones diferenciales parciales elípticas

La ecuación diferencial parcial elíptica a considerar es la ecuación de Poisson

$$\nabla^2 u(x,y) \equiv \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y) \tag{1}$$

en $R = \{(x,y) \mid a < x < b, c < y < d\}$, con $u(x,y) = g(x,y) \in S$, donde S denota al contorno de R. Si f y g son continuas en su dominio entonces hay una única solución a la ecuación.

1.1. Seleccionando un retículo

El método a utilizar es una adaptación bidimensional del método de diferencias finitas para problemas con fronteras lineales como se discute en [1]. El primer paso es escoger enteros n y m para definir el tamaño de los pasos(*steps*) $h = (b-a)/n$ y $k = (d-c)/m$ particionando de está manera el intervalo $[a,b]$ en n partes iguales de ancho h y el intervalo $[c,d]$ en m partes iguales con ancho k, formando un retículo o cuadricula como se puede ver en la figura

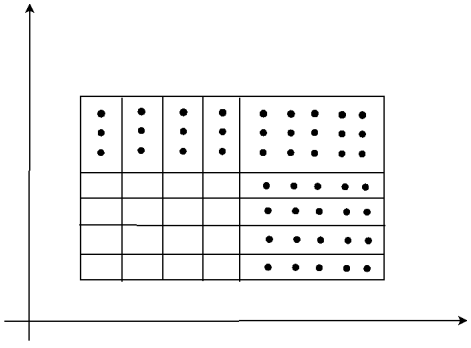

Figura 1: Cuadrícula de $n \times m$

Este retículo se construye formalmente al dibujar lineas verticales y horizontales sobre el dentro del rectángulo R en los puntos con coordenadas (x_i, y_j), donde

$$x_i = a+ih, \text{ para cada } i = 0,1,2,\ldots,n \text{ y } y = a+jk, \text{ para cada } j = 0,1,2,\ldots,m \tag{2}$$

Las rectas correspondientes a $x = x_i$ y $y = y_i$ son las lineas que forman la cuadricula y sus intersecciones son los puntos del retículo. Para cada punto interior del retículo (x_i, y_j), para $i = 1, 2, \ldots, n-1$ y $j = 1, 2, \ldots, m-1$, se puede utilizar una serie de Taylor en la variable x alrededor del punto x_i para generar una fórmula de diferencia centrada

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_i) = \frac{u(x_{i+1}, y_j) - 2u(x_i, y_i) + u(x_{i-1}, y_j)}{h^2} - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) \quad (3)$$

donde $\xi_i \in (x_{i-1}, x_{i+1})$. De igual manera se puede encontrar la serie de Taylor en la variable y alrededor del punto y_j para hallar la diferencia centrada

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_i) = \frac{u(x_i, y_{j+1}) - 2u(x_i, y_i) + u(x_i, y_{j-1})}{k^2} - \frac{k^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, \eta_j) \quad (4)$$

donde $\eta_i \in (y_{i-1}, y_{i+1})$, sustituyendo estas fórmulas en las ecuación 1 permite expresar la ecuación de Poisson en los puntos (x_i, y_j) como

$$\begin{aligned} &\frac{u(x_{i+1}, y_j) - 2u(x_i, y_i) + u(x_{i-1}, y_j)}{h^2} + \frac{u(x_i, y_{j+1}) - 2u(x_i, y_i) + u(x_i, y_{j-1})}{k^2} \\ &= f(x_i, y_j) + \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) + \frac{k^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, \eta_j) \end{aligned} \quad (5)$$

para cada $i = 1, 2, \ldots, n-1$ y $j = 1, 2, \ldots, m-1$. Las condiciones de contorno son

$$u(x_0, y_j) = g(x_0, y_j) \text{ y } u(x_n, y_j) = g(x_n, y_j), \text{ para cada } j = 0, 1, \ldots, m;$$

$$u(x_i, y_0) = g(x_i, y_0) \text{ y } u(x_i, y_m) = g(x_i, y_m), \text{ para cada } i = 1, 2, \ldots, n-1;$$

1.2. Método de Diferencias Finitas

En forma de ecuación de diferencias el Método de diferencias finitas es

$$2\left[\left(\frac{h}{k}\right)^2 + 1\right] w_{ij} - (w_{i+1,j} + w_{i-1,j}) - \left(\frac{h}{k}\right)^2 (w_{i,j+1} + w_{i,j-1}) = -h^2 f(x_i, y_j) \quad (6)$$

para cada $i = 1, 2, \ldots, n-1$ y $j = 1, 2, \ldots, m-1$, donde

$$\begin{aligned} &w_{0j} = g(x_0, y_j) \text{ y } w_{nj} = g(x_n, y_j), \text{ para cada } j = 0, 1, \ldots, m; \\ &w_{i0} = g(x_i, y_0) \text{ y } w_{im} = g(x_i, y_m), \text{ para cada } i = 1, 2, \ldots, n-1 \end{aligned} \quad (7)$$

donde w_{ij} aproxima $u(x_i, y_j)$. Este método tiene un error de truncación del orden $O(h^2 + k^2)$. La ecuación 6 involucra aproximaciones para $u(x, y)$ en los puntos

$$(x_{i-1}, y_j), \ (x_i, y_j), \ (x_i, y_{j-1}), \text{ y } (x_i, y_{j+1}).$$

Utilizando la información proveída por las condiciones de contorno 7 cuando el sistema definido por 6 lo permite, i.e. en los puntos adyacentes al contorno definido por la cuadrícula construida previamente(1.1), se obtiene un sistema de ecuaciones con una matriz de incógnitas de dimensiones $(n-1)(m-1) \times (n-1)(m-1)$

donde las incógnitas son las aproximaciones w_{ij} para $u(x_i, y_j)$ en los puntos interiores del retículo(1.1).

El sistema lineal para las incógnitas w_{ij} se puede representar más eficientemente para realizar operaciones matriciales si se realiza una re-etiquetación de de los puntos interiores del retículo, una recomendación para realizar este esquema de re-etiquetación puede hallarse en [2] de la siguiente manera

$$P_l = (x_i, y_j), \text{ y } w_l = w_{ij} \tag{8}$$

donde

$$l(i, j) = i + (m - 1 - j)(n - 1) \tag{9}$$

para cada $i = 1, 2, \ldots, n-1$ y $j = 1, 2, \ldots, m-1$, con este esquema se etiquetan a los puntos interiores del retículo de izquierda a derecha y de arriba hacia abajo y con esto se asegura que el sistema lineal resultante para hallar w_{ij} es una matriz con bandas de ancho máximo $2n - 1$, una matriz tridiagonal por bloques. Esta técnica fue utilizada para implementar este algoritmo en Python, ver algoritmos 1 y 2.

2. Algoritmo Método de Diferencias Finitas para ecuación de Poisson

Para hallar un algoritmo que genere y resuelva el método de diferencias finitas para las ecuaciones parciales elípticas descrito en la sección anterior se usa el esquema de re-etiquetado expuesto anteriormente, con esto se asegura que se obtiene un sistema tri-diagonal por bloques. Para estos sistemas tri-diagonales por bloques es conveniente aplicar una generalización del algoritmo de Factorización de Crout [2] para matrices tri-diagonales por bloques.
Otro algoritmo iterativo que se puede utilizar es el método de relajamiento SOR. Con esto se puede delinear una estrategia para construir un algoritmo para el método de diferencias finitas:

- Construir el sistema asociado a las aproximaciones w_{ij} para $u(x_i, y_j)$, utilizando el esquema de etiquetado expuesto anteriormente.
- Aplicar un método de resolución de sistemas lineales: Generalización de factorización de Crout, método iterativo SOR.
- Resolver para w_{ij}

2.1. Algoritmo diferencias finitas(sistema lineal)

2.1.1. Función de re-etiquetado l

Se implementa la función l para re-etiquetar el sistema de ecuaciones lineal

Algorithm 1 Función L para re-etiquetar las variables del sistema lineal

```
function L(i, j, n, m)
    return (i + 1) + (m − 1 − (j + 1)) × (n − 1) − 1
end function
```

2.1.2. Sistema lineal asociado

Con está función se crea el sistema asociado al método de diferencias finitas

Algorithm 2 Finite Difference Linear System

function FINITE_DIFFERENCE_LINEAR_SYSTEM(a, b, c, d, n, m, f, g)
  $h \leftarrow (b-a)/n$
  $k \leftarrow (d-c)/m$
  $x \leftarrow$ numpy.linspace$(a, b, n+1)$
  $y \leftarrow$ numpy.linspace$(c, d, m+1)$
  $W_{ij} \leftarrow$ zeros $(((n-1)\times(m-1)), ((n-1)\times(m-1)))$
  $w \leftarrow$ zeros $((n-1)\times(m-1))$
  for $i \leftarrow 0$ to $n-2$ **do**
    for $j \leftarrow 0$ to $m-2$ **do**
      $W_{ij}[L(i,j,n,m), L(i,j,n,m)] \leftarrow 2((h/k)^2+1)$
      if $i \neq 0$ and $i \neq n-2$ **then**
        $W_{ij}[L(i,j,n,m), L(i+1,j,n,m)] \leftarrow -1$
        $W_{ij}[L(i,j,n,m), L(i-1,j,n,m)] \leftarrow -1$
      else
        if $i = 0$ **then**
          $W_{ij}[L(i,j,n,m), L(i+1,j,n,m)] \leftarrow -1$
          $w[l(i,j,n,m)] \leftarrow g(x[i], y[j+1], a, b, c, d)$
        end if
        if $i = n-2$ **then**
          $W_{ij}[L(i,j,n,m), L(i-1,j,n,m)] \leftarrow -1$
          $w[L(i,j,n,m)] \leftarrow g(x[i+2], y[j+1], a, b, c, d)$
        end if
      end if
      if $j \neq 0$ and $j \neq m-2$ **then**
        $W_{ij}[L(i,j,n,m), L(i,j+1,n,m)] \leftarrow -(h/k)^2$
        $W_{ij}[L(i,j,n,m), L(i,j-1,n,m)] \leftarrow -(h/k)^2$
      else
        if $j = 0$ **then**
          $W_{ij}[L(i,j,n,m), L(i,j+1,n,m)] \leftarrow -(h/k)^2$
          $aux \leftarrow (h/k)^2 * g(x[i+1], y[j], a, b, c, d)$
          $w[L(i,j,n,m)] \leftarrow w[L(i,j,n,m)] + aux$
        end if
        if $j = m-2$ **then**
          $W_{ij}[L(i,j,n,m), L(i,j-1,n,m)] \leftarrow -(h/k)^2$
          $aux \leftarrow (h/k)^2 * g(x[i+1], y[j+2], a, b, c, d)$
          $w[L(i,j,n,m)] \leftarrow w[L(i,j,n,m)] + aux$
        end if
      end if
      $w[L(i,j,n,m)] \leftarrow -h^2 f(x[i+1], y[j+1]) + w[L(i,j,n,m)]$
    end for
  end for
  return W_{ij}, w
end function

2.2. Algoritmo de Generalización Factorización de Crout para matrices tridiagonales por bloques

Esta es la implementación de la generalización del algoritmo de factorización de Crout para matrices tri-diagonales por bloques, el algoritmo de factorización de Crout sólo es valido para matrices tri-diagonales y no para matrices tri-diagonales por bloques.

Algorithm 3 Crout Generalization Algorithm for Tridiagonal Block Matrices

Require: $A \in \mathbb{R}^{N \times N}$, $K \in \mathbb{R}^{N}$, $n \in \mathbb{N}$
Ensure: $sol \in \mathbb{R}^{N}$

function CROUT_GENERALIZATION(A, K, n)
$N \leftarrow \lfloor \frac{\text{shape}(A)[1]}{n} \rfloor$
$W \leftarrow []$
$B_1 \leftarrow A_{1:n,1:n}$
$C_1 \leftarrow A_{1:n,n:n+n}$
$W.append(\text{inv}(B_1) \cdot C_1)$
for $i \leftarrow 2$ to $N-1$ **do**
$B_i \leftarrow A_{(i-1)n:in,(i-1)n:in}$
$A_i \leftarrow A_{(i-1)n:in,(i-2)n:(i-1)n}$
$C_i \leftarrow A_{(i-1)n:in,in+1:n+(i-1)n}$
$W.append(\text{inv}(B_i - A_i \cdot W_{i-2}) \cdot C_i)$
end for
$B_N \leftarrow A_{1:n,1:n}$
$K_1 \leftarrow K_{1:n}$
$G_1 \leftarrow \text{inv}(B_1) \cdot K_1$
$G \leftarrow []$
$G.append(G_1)$
for $i \leftarrow 2$ to N **do**
$K_i \leftarrow K_{(i-1)n:in}$
$B_i \leftarrow A_{(i-1)n:in,(i-1)n:in}$
$A_i \leftarrow A_{(i-1)n:in,(i-2)n:(i-1)n}$
$G.append(\text{inv}(B_i - A_i \cdot W_{i-2}) \cdot (K_i - A_i \cdot G_{i-2}))$
end for
$Z \leftarrow G[:]$
for $i \leftarrow N-2$ to 0 step -1 **do**
$Z_i \leftarrow G_i - W_i \cdot Z_{i+1}$
end for
$sol \leftarrow \text{concatenate}(Z)$
return sol
end function

2.3. Método iterativo SOR para resolver sistemas lineales

Algorithm 4 Iterative SOR method for solving linear systems

Require: A es una $n \times n$ matriz, b es un vector de incógnitas de dimensión n, $x^{(0)}$ es una solución propuesta inicial, ω es el parámetro de relajamiento, ϵ es la tolerancia, and max_{iter} es el número máximo de iteraciones.
Ensure: x es una solución aproximada de $Ax = b$.

```
function SOR(A, b, ω, ϵ, máx_iter, x^(0))
    k ← 1
    xo ← x^(0)
    tolerance ← True
    while k < max_iter and tolerance do
        for i = 1 to n do
            s ← Σ_{j=1}^{i-1} a_ij x_j + Σ_{j=i+1}^{n} a_ij xo_j
            x_i ← (1 − ω) xo_i + (ω / a_ii)(b_i − s)
        end for
        if ||x − xo|| < ϵ then
            tolerance ← False
        end if
        k ← k + 1
        xo = x
    end while
    if k = max_iter then
        print "Se ha alcanzado el número máximo de iteraciones"
    end if
    return x
end function
```

2.4. Diferencia finitas usando generalización de factorización Crout

Algorithm 5 Algoritmo diferencias finitas usando generalización factorización de Crout

Dado un problema con condiciones de frontera de la forma 1, definir

$$a, b, c, d, n, m, f(x, y), g(x, y)$$

usando las funciones definidas en los algoritmos 2, 3

$A, b \leftarrow$ finite_difference_linear_system(a, b, c, d, n, m, f, g)
solution $\leftarrow$ Crout_generalization(A,b,n-1)

este algoritmo se ha implementado en Python, usando la librería Numpy. El repositorio se encuentra en `https://github.com/Julio-Medina/Finite_Difference_Method/tree/main/code`.

2.5. Diferencia finitas usando método de relajamiento SOR

Algorithm 6 Algoritmo diferencias finitas usando generalización factorización de Crout

Dado un problema con condiciones de frontera de la forma 1, definir

$$a, b, c, d, n, m, f(x, y), g(x, y), \omega, \epsilon, x^{(0)}$$

usando las funciones definidas en los algoritmos 2, 4

$A, b \leftarrow$ finite_difference_linear_system(a, b, c, d, n, m, f, g)
$solution \leftarrow \text{SOR}(A, b, \omega, \epsilon, \text{máx}_{iter}, x^{(0)})$

2.6. Ejemplos

Para ilustrar el funcionamiento del algoritmo no hay mejor forma que resolver algunos problemas con condiciones en la frontera.

2.6.1. Ejemplo 1

Resolver el siguiente problema

$$\frac{\partial^2 u}{\partial x^2}(x, y) + \frac{\partial^2 u}{\partial y^2}(x, y) = 0 \tag{10}$$

para cada (x, y) en el conjunto $R = \{(x, y) | \, 0 < x < 0{,}5, \, 0 < y < 0{,}5\}$. Las condiciones de contorno son

$$u(0, y) = 0, \; u(x, 0) = 0, \; u(x, 0{,}5) = 200x, \; u(0{,}5, y) = 200y,$$

Si $n = m = 4$, la ecuación de diferencias finitas 6 da como resultado

$$4w_{i,j} - w_{i+1,j} - w_{i-1,j} - w_{i,j-1} - w_{i,j+1} = 0 \tag{11}$$

para cada $i = 1, 2, 3$ y $j = 1, 2, 3$.
Por ejemplo cuando $i = 1, j = 1$ se obtiene

$$4w_{1,1} - w_{2,1} - w_{0,1} - w_{1,0} - w_{1,2} = 0 \tag{12}$$

pero $w_{1,0} = w_{0,1} = 0$, usando la función de re-etiquetado 9 se obtiene

$$l(1, 1) = 7, \; l(2, 1) = 8, \; l(1, 2) = 4$$

sustituyendo en 12 y usando las condiciones de contorno se obtiene la ecuación

$$4w_7 - w_8 - w_4 = 0$$

este proceso se repite para los indices restantes, dando como resultado 9 ecuaciones lineales para las variables $w_1, w_2, w_3, w_4, w_5, w_7, w_8, w_9$. La matriz de co-

eficientes del sistema lineal obtenido al iterar a través de todos los indices es

$$A = \begin{bmatrix} 4{,}0 & -1{,}0 & 0{,}0 & -1{,}0 & 0{,}0 & 0{,}0 & 0{,}0 & 0{,}0 & 0{,}0 \\ -1{,}0 & 4{,}0 & -1{,}0 & 0{,}0 & -1{,}0 & 0{,}0 & 0{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0 & -1{,}0 & 4{,}0 & 0{,}0 & 0{,}0 & -1{,}0 & 0{,}0 & 0{,}0 & 0{,}0 \\ -1{,}0 & 0{,}0 & 0{,}0 & 4{,}0 & -1{,}0 & 0{,}0 & -1{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0 & -1{,}0 & 0{,}0 & -1{,}0 & 4{,}0 & -1{,}0 & 0{,}0 & -1{,}0 & 0{,}0 \\ 0{,}0 & 0{,}0 & -1{,}0 & 0{,}0 & -1{,}0 & 4{,}0 & 0{,}0 & 0{,}0 & -1{,}0 \\ 0{,}0 & 0{,}0 & 0{,}0 & -1{,}0 & 0{,}0 & 0{,}0 & 4{,}0 & -1{,}0 & 0{,}0 \\ 0{,}0 & 0{,}0 & 0{,}0 & 0{,}0 & -1{,}0 & 0{,}0 & -1{,}0 & 4{,}0 & -1{,}0 \\ 0{,}0 & 0{,}0 & 0{,}0 & 0{,}0 & 0{,}0 & -1{,}0 & 0{,}0 & -1{,}0 & 4{,}0 \end{bmatrix} \tag{13}$$

y el vector constante es

$$b = \begin{bmatrix} 25{,}0 \\ 50{,}0 \\ 150{,}0 \\ 0{,}0 \\ 0{,}0 \\ 50{,}0 \\ 0{,}0 \\ 0{,}0 \\ 25{,}0 \end{bmatrix}$$

Es importante notar que la matriz resultante 13 es una matriz tri-diagonal por bloques con cada bloque de dimensión 3×3. Con esto se puede resolver el sistema

$$Aw = b \tag{14}$$

donde se obtiene

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \end{bmatrix} = \begin{bmatrix} 18{,}75 \\ 37{,}5 \\ 56{,}25 \\ 12{,}5 \\ 25 \\ 37{,}5 \\ 6{,}25 \\ 12{,}5 \\ 18{,}75 \end{bmatrix} \tag{15}$$

2.6.2. Ejemplo 2

Resolver el siguiente problema

$$\frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = 4 \tag{16}$$

para cada (x, y) en el conjunto $R = \{(x,y) | \, 0 < x < 1, \, 0 < y < 2\}$. Las condiciones de contorno son

$$u(1,y) = \ln(y^2+1), \, u(x,0) = 2\ln x, \, u(x,1) = \ln(x^2+1) \, u(2,y) = \ln(y^2+4)$$

Si $h = k = \frac{1}{3}$, entonces $n = 3, \; m = 3$, la ecuación de diferencias finitas 6 da como resultado

$$4w_{i,j} - w_{i+1,j} - w_{i-1,j} - w_{i,j-1} - w_{i,j+1} = 4 \tag{17}$$

La matriz de coeficientes asociada A es

$$\begin{bmatrix} 4{,}0 & -1{,}0 & -1{,}0 & 0{,}0 \\ -1{,}0 & 4{,}0 & 0{,}0 & -1{,}0 \\ -1{,}0 & 0{,}0 & 4{,}0 & -1{,}0 \\ 0{,}0 & -1{,}0 & -1{,}0 & 4{,}0 \end{bmatrix} \tag{18}$$

y el vector constante es

$$b = \begin{bmatrix} 1{,}165 \\ 2{,}774 \\ -0{,}444 \\ 1{,}165 \end{bmatrix}$$

Es importante notar que la matriz resultante 18 es una matriz tri-diagonal por bloques con cada bloque de dimensión 2×2. Con esto se puede resolver el sistema

$$Aw = b \tag{19}$$

donde se obtiene

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 0{,}5825 \\ 0{,}9849 \\ 0{,}1801 \\ 0{,}5825 \end{bmatrix} \tag{20}$$

2.6.3. Ejemplo 3

Resolver el siguiente problema

$$\frac{\partial^2 u}{\partial x^2}(x, y) + \frac{\partial^2 u}{\partial y^2}(x, y) = xe^y \tag{21}$$

para cada (x, y) en el conjunto $R = \{(x, y) | \, 0 < x < 2, \, 0 < y < 1\}$. Las condiciones de contorno son

$$u(0, y) = 0, \; u(x, 0) = x, \; u(x, 1) = ex, \; u(2, y) = 2e^y$$

Con $n = 6$, $m = 5$, comparar las aproximaciones con la solución analítica $u(x, y) = xe^y$.
Usando las funciones creadas en Python se obtiene la siguiente tabla. Esta tabla puede hallarse en el repositorio de GitHub mencionado anteriormente con el nombre `error_table.csv`.

	i	j	x_i	y_j	w_{ij}	$u(x_i, y_j)$	$\lvert u(x_i, y_j) - w_{ij} \rvert$
0	1	1	0.333333	0.2	0.407265	0.407134	0.000130
1	1	2	0.333333	0.4	0.497483	0.497275	0.000208
2	1	3	0.333333	0.6	0.607596	0.607373	0.000223
3	1	4	0.333333	0.8	0.742007	0.741847	0.000160
4	2	1	0.666667	0.2	0.814524	0.814269	0.000255
5	2	2	0.666667	0.4	0.994958	0.994550	0.000408
6	2	3	0.666667	0.6	1.215183	1.214746	0.000437
7	2	4	0.666667	0.8	1.484009	1.483694	0.000315
8	3	1	1.000000	0.2	1.221766	1.221403	0.000363
9	3	2	1.000000	0.4	1.492405	1.491825	0.000580
10	3	3	1.000000	0.6	1.822743	1.822119	0.000624
11	3	4	1.000000	0.8	2.225993	2.225541	0.000452
12	4	1	1.333333	0.2	1.628964	1.628537	0.000427
13	4	2	1.333333	0.4	1.989778	1.989100	0.000679
14	4	3	1.333333	0.6	2.430227	2.429492	0.000735
15	4	4	1.333333	0.8	2.967928	2.967388	0.000540
16	5	1	1.666667	0.2	2.036042	2.035671	0.000371
17	5	2	1.666667	0.4	2.486958	2.486374	0.000584
18	5	3	1.666667	0.6	3.037506	3.036865	0.000641
19	5	4	1.666667	0.8	3.709724	3.709235	0.000489

Referencias

[1] Richard L. Burden, J. Douglas Faires *Numerical Analysis*, (Ninth Edition). Brooks/Cole, Cengage Learning. 978-0-538-73351-9

[2] Richard S. Varga. *Matrix Iterative Analysis*. Second Edition. Springer. DOI 10.1007/978-3-642-05156-2