Diferencias Finitas

Yarko Niño C. y Paulo Herrera R.

Departamento de Ingeniería Civil, Universidad de Chile

Semestre Primavera 2011

Curso hasta ahora . . .

Ecuación de onda (**hiperbólica**, **problema de valor inicial**)

$$\frac{\partial^2 u}{\partial t^2} = v^2 \frac{\partial^2 u}{\partial x^2}$$

Ecuación de difusión (**parabólica**, **problema de valor inicial**)

$$\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2}$$

Ecuación de Poisson (**elíptica**, **problema de valore de borde**):

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

Soluciones numéricas

Puntos a considerar para implementar soluciones numéricas:

- Métodos para calcular soluciones de Problemas de Valor Inicial pueden ser inestables $\Rightarrow$ **estabilidad** es un punto importante a considerar.

- En cambio estabilidad no es tan importante en la solución de Problemas de Valor de Borde. Sin embargo, **eficiencia** en la resolución de sistemas lineales muy grandes, si es importante.

Soluciones numéricas

Estrategia general para la solución de EDPs:

1. Evaluar la solución (variables independientes) en un número finito de puntos (malla o grilla).

2. Aproximar derivadas en término de los valores conocidos en los nodos de la grilla.

3. Reemplazar derivadas temporales por alguna de las discretizaciones estudiadas para resolver Problemas de Valor Inicial.

4. Reducir la(s) ecuación(es) diferencial(es) parcial(es) y condiciones de borde a un sistema de ecuaciones algebraicas. En el caso de EDPs no lineales, es necesario linearizarlas para conseguir un sistema lineal.

Aproximación de derivadas

La forma más simple de representar derivadas es aproximarlas por expresiones finitas basadas en los valores de las variables dependientes en los nodos de la grilla.

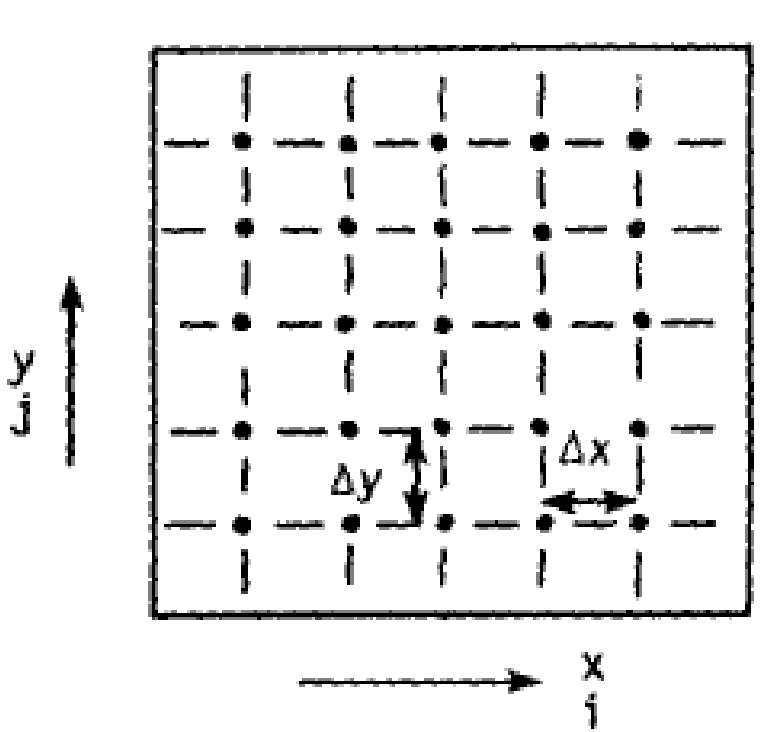
Esta técnica se denomina **Diferencias Finitas**.

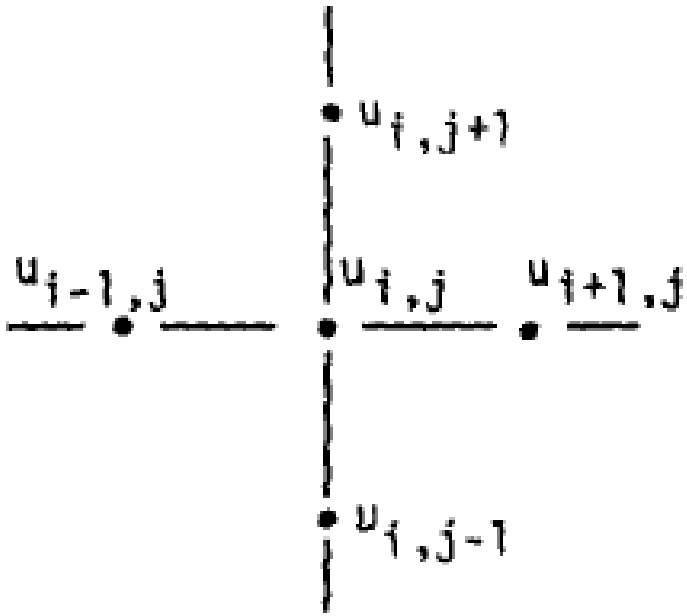
Por ejemplo, tomando una grilla bidimensional con espaciamiento $\Delta x = \Delta y = \Delta$, tenemos

$$\frac{\partial u}{\partial x} \approx \frac{u(x_i + \Delta, y_j) - u(x_i, y_j)}{\Delta}$$

Notación: $u(x_i, y_j) \equiv u_{i,j}$ y $u(x_i + \Delta, y_j) \equiv u_{i+1,j}$

Grilla numérica





Diferencias Finitas

Del tema anterior sabemos que es posible derivar diferentes aproximaciones (**de distinto orden**) para estimar derivadas.

Aplicando series de Taylor,

$$u(x_i + \Delta x, y_j) = u_(x_i, y_i) + \left.\frac{\partial u}{\partial x}\right|_{x_i,y_i} \Delta x + \left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i,y_i} \frac{(\Delta x)^2}{2!} + \dots$$

Entonces podemos encontrar una aproximación para la primera derivada,

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta x} + O(\Delta x)$$

El término $O(\Delta x)$ corresponde al **error de truncación** de la aproximación de diferencia finita de la derivada. En este caso el **error es orden Δx**.

Diferencias Finitas (cont.)

Otras aproximaciones para la primera deriva también pueden ser derivadas a partir de la serie de Taylor.

Por ejemplo,

$$u(x_i - \Delta x, y_j) = u(x_i, y_i) - \left.\frac{\partial u}{\partial x}\right|_{x_i,y_i} \Delta x + \left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i,y_i} \frac{(\Delta x)^2}{2!} + \dots$$

Entonces, la aproximación hacia atrás de la primera derivada usando diferencias finitas es,

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{\Delta x} + O(\Delta x)$$

Diferencias Finitas (cont.)

También es posible derivar aproximaciones de más alto orden.

Por ejemplo, restando las aproximaciones para $u(x_i + \Delta x, y_i)$ y $u(x_i - \Delta x, y_0)$ obtenemos,

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + O\left((\Delta x)^2\right)$$

Este aproximación es de segundo orden en Δx. Esto quiere decir que el error de truncación (ET) de la aproximación satisface,

$$|ET| \leq C|\Delta x|^2 \qquad \Delta x \to 0$$

Diferencias Finitas (cont.)

También podemos encontrar expresiones para derivadas de más alto orden.

Por ejemplo, sumando las aproximaciones para $u(x_i + \Delta x, y_i)$ y $u(x_i - \Delta x, y_i)$ obtenemos,

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + O\left((\Delta x)^2\right)$$

De esta forma se pueden derivar expresiones con errores de truncación de distinto orden para aproximar derivadas de distinto orden.

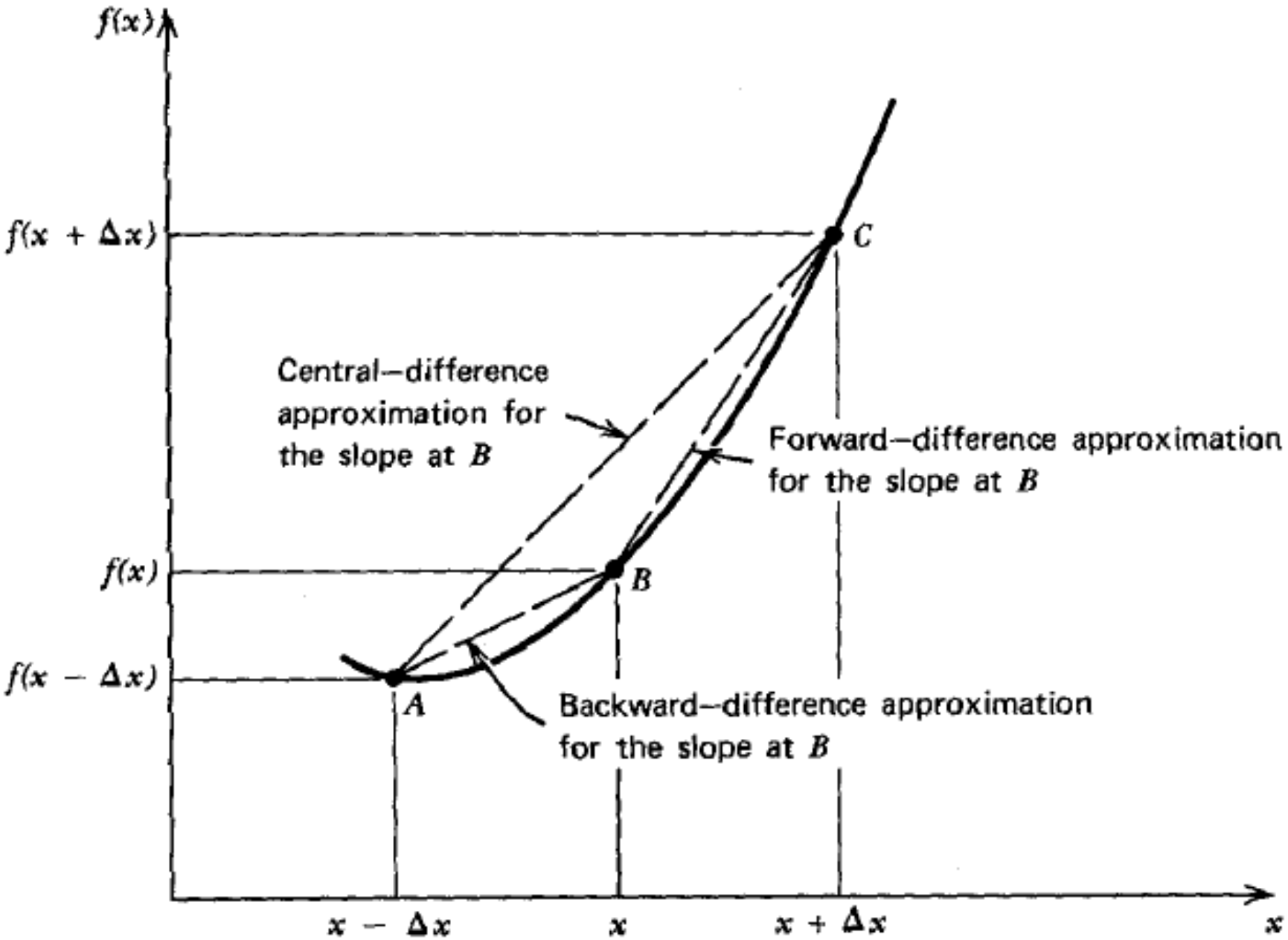
Diferencias Finitas: Aproximaciones

Derivada	Aproximación	Error
$\left.\frac{\partial u}{\partial x}\right\|_i$	$\frac{u_{i+1}-u_i}{\Delta}$	$O(\Delta)$
	$\frac{u_i-u_{i-1}}{\Delta}$	$O(\Delta)$
	$\frac{u_{i+1}-u_{i-1}}{2\Delta}$	$O\left(\Delta^2\right)$
$\left.\frac{\partial^2 u}{\partial x^2}\right\|_i$	$\frac{u_{i+1}-2u_i+u_{i-1}}{\Delta^2}$	$O(\Delta^2)$
	$\frac{-u_{i+2}+16u_{i+1}-30u_i+16ui-1-u_{i-2}}{12\Delta^2}$	$O(\Delta^4)$
$\left.\frac{\partial^3 u}{\partial x^3}\right\|_i$	$\frac{u_{i+2}-2u_{i+1}+2u_{i-1}-u_{i-2}}{2\Delta^3}$	$O(\Delta^2)$

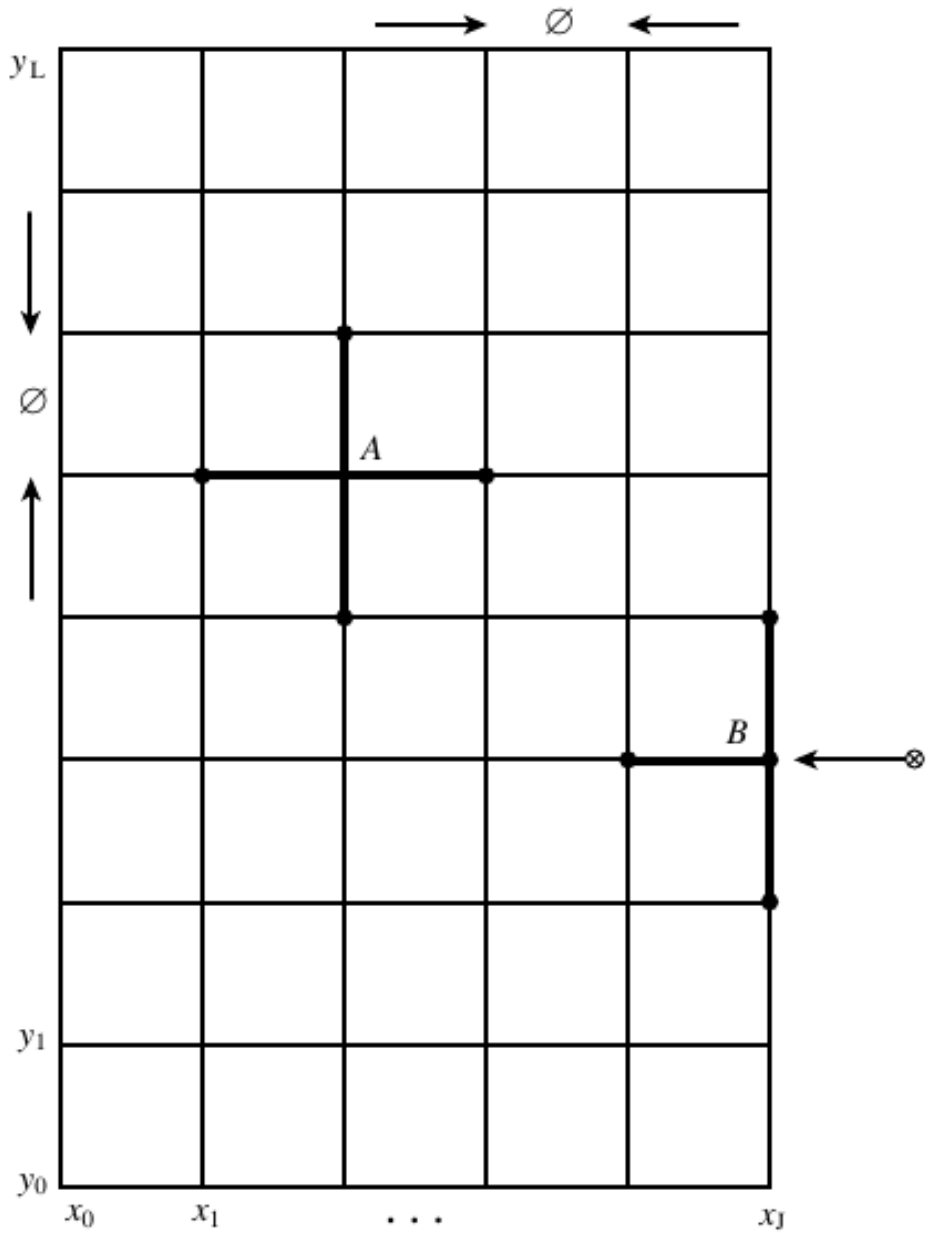
La gran mayoría de las EDPs que describen procesos que nos interesan son solo de **segundo orden**.

(Apuntes incluye otras aproximaciones.)

Diferencias Finitas: Aproximaciones (cont.)



Diferencias Finitas: *Stencil*



Diferencias Finitas: *Stencil* (cont.)

Aproximaciones de orden más alto producen soluciones más precisas al costo de utilizar más nodos de la grilla (*stencils* más amplios) $\Rightarrow$ nodos cercanos a los bordes deben ser tratados de forma especial.

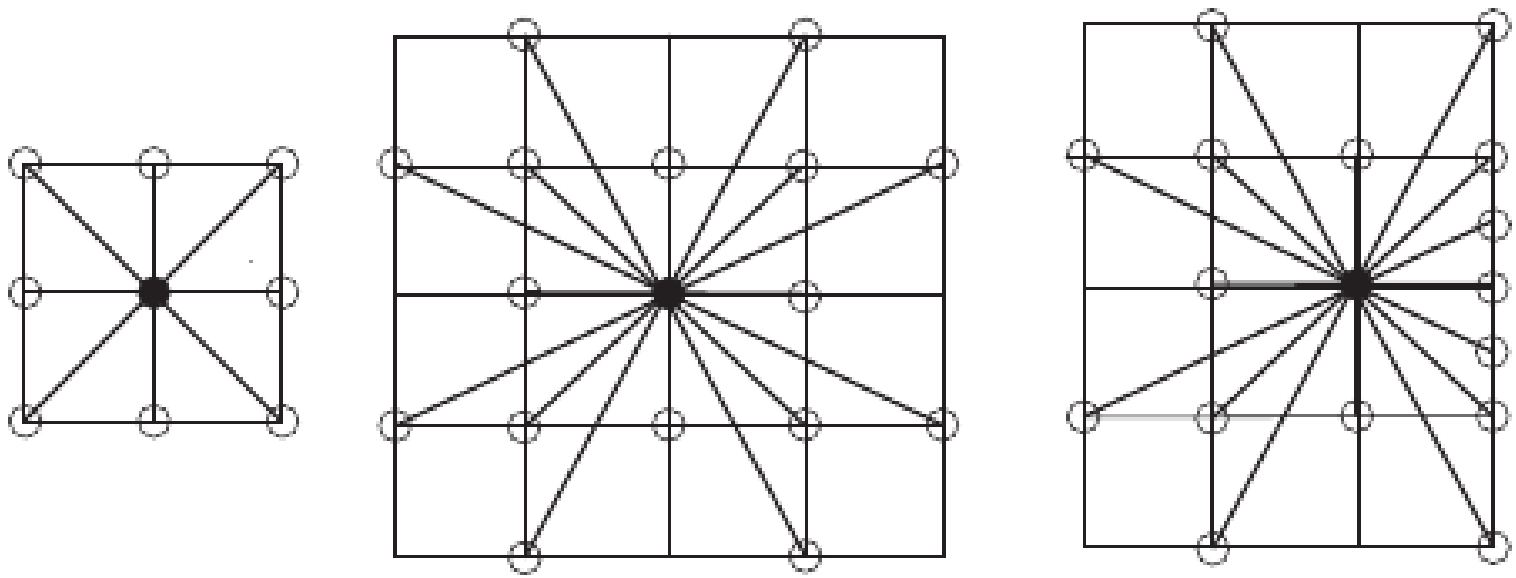


FIGURE 1. (a) Computational stencils for the 9 and 17 point schemes. (b) Computational stencil near the boundary.

Ejemplo: Flujo en un acuífero confinado 1D

Ecuación que describe la distribuciíon de carga hidráulica en un acuífero confinado es,

$$T\frac{\partial^2 h}{\partial x^2} = S\frac{\partial h}{\partial t} \qquad \text{o} \qquad D\frac{\partial^2 h}{\partial x^2} = \frac{\partial h}{\partial t}$$

Con condiciones iniciales y de borde,

$$h(x, t = 0) = H_{inicial}(x), \quad h(x = 0, t) = H_0(t), \quad h(x = L, t) = H_L(t)$$

Aplicando una aproximación central para la segunda derivada espacial obtenenemos,

$$\frac{\partial h}{\partial t} = \frac{D}{\Delta^2}\left[h_{i+1}(t) - 2h_i(t) + h_{i-1}(t)\right]$$

Ejemplo: Flujo en un acuífero confinado 1D (cont.)

Aplicando una aproximación de primer orden explícita para aproximar la derivada temporal obtenemos,

$$\frac{h_i^{n+1} - h_i^n}{\Delta t} = \frac{D}{\Delta^2}\left[h_{i+1}^n - 2h_i^n + h_{i-1}^n\right]$$

Reordenando términos,

$$h_i^{n+1} = \mathcal{D}\left[h_{i+1}^n - 2h_i^n + h_{i-1}^n\right] + h_i^n$$

donde $\mathcal{D} = D\Delta t/\Delta^2$.

Ejemplo: Flujo en un acuífero confinado 1D (cont.)

La última expresión puede ser escrita en notación matricial como,

$$\mathbf{h}^{n+1} = \mathbf{A} \cdot \mathbf{h}^n$$

donde $\mathbf{h}^n = \{h_0^n, h_1^n, \ldots, h_N^n\}$ y la matriz $\mathbf{A}$ es tridiagonal con elementos $A_{i,i} = 1 - 2\mathcal{D}$, y $A_{i+1,i} = A_{i-1,i} = \mathcal{D}$.

Qué pasa con las condiciones de borde?

$$h_0^n = H_0^n \qquad h_N^n = H_L^n$$

Esto se puede incorporar tomando $A_{0,0} = 1, \quad A_{0,1} = 0$ y $A_{N,N} = 1, \quad A_{N,N-1} = 0$.

Esta alternativa es fácil pero ineficiente cuando hay muchos nodos en los bordes.

Ejemplo: Flujo en un acuífero confinado 1D (cont.)

Otra alternativa es observar que en los bordes,

$$\begin{aligned} h_1^{n+1} &= h_1^n + \mathcal{D}\left[h_2^n - 2h_1^n + h_0^n\right] \\ &= h_1^n + \mathcal{D}\left[h_2^n - 2h_1^n\right] + \mathcal{D}h_0^n \end{aligned}$$

Lo cual puede ser reescrito como,

$$\tilde{\mathbf{h}}^{n+1} = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{h}}^n + \mathbf{b}$$

donde $\tilde{\mathbf{h}} = \{h_1, \ldots, h_{N-1}\}$ y $\mathbf{b} = \{\mathcal{D}H_0^n, 0, \ldots, 0, \mathcal{D}H_L^n\}$.

En este caso la matriz $\tilde{\mathbf{A}}$ tiene $(N-2)$x$(N-2)$ elementos en vez de los NxN elementos de $\mathbf{A}$.

Si $N = 10000$, $\tilde{\mathbf{A}}$ tiene $\approx 3,9996 \cdot 10^4$ elementos menos que $\mathbf{A}$.

Ejemplo: Flujo en un acuífero confinado 1D (cont.)

Qué pasa si usamos una aproximación implícita para la derivada temporal?

$$\frac{h_i^{n+1} - h_i^n}{\Delta t} = \frac{D}{\Delta^2}\left[h_{i+1}^{n+1} - 2h_i^{n+1} + h_{i-1}^{n+1}\right]$$

que se puede escribir como,

$$\mathbf{A}\mathbf{h}^{n+1} = \mathbf{h}^n$$

donde $A_{i,i} = 1 + 2\mathcal{D}$ y $A_{i,i-1} = A_{i,i+1} = -\mathcal{D}$.
También posible reescribir como,

$$\tilde{\mathbf{A}}\tilde{\mathbf{h}}^{n+1} = \tilde{\mathbf{h}}^n + \mathbf{b}$$

o simplemente,

$$\tilde{\mathbf{A}}\tilde{\mathbf{h}}^{n+1} = \tilde{\mathbf{b}}$$

Ejemplo: Flujo en un acuífero confinado 1D (cont.)

- Entonces, para calcular la solución en el tiempo $n+1$ debemos resolver un sistema lineal dado por una matriz *sparse*. La **matriz $\tilde{\mathbf{A}}$ es** *sparse* si tiene la **mayoría de los elementos iguales a cero**.
- Este es el tipo de **matriz que resulta al aplicar la gran mayoría de métodos numéricos para resolver EDPs**.
- **Métodos tradicionales para invertir la matriz $\tilde{\mathbf{A}}$** (ej. Eliminación Gausiana) **requieren del orden de $O(N^3)$ operaciones** y requieren almacenar todos los coeficientes de la matriz ($O(N^2)$).
- Existen varios métodos para resolver este tipo de sistemas que aprovechan la estructura de la matriz para ahorrar memoria (solo los coeficientes que son no ceros son almacenados o para acelerar el cómputo).

Sistema tridiagonal: Algoritmo de Thomas

La discretización de una EDP en 1D con un *stencil* de tres puntos, genera una matriz tridiagonal, tal que el sistema lineal se puede expresar como:

$$\begin{aligned} b_1 x_1 + c_1 x_2 &= d_1 \quad ; \quad i = 1 \\ a_i x_{i-1} + b_i x_i + c_i c_{i+1} &= d_i \quad ; \quad i = 2, \ldots, N-1 \\ a_n x_{n-1} + b_n x_n &= d_n \quad ; \quad i = N \end{aligned}$$

Existe un algoritmo para resolver este tipo de sistemas conocido como *Algoritmo de Thomas* que consiste en transformar el sistema para obtener una matriz diagonal superior y luego aplicar sustitución inversa con un costo $O(N)$.

Sistema tridiagonal: Algoritmo de Thomas (cont.)

$$
\begin{aligned}
b_1 x_1 + c_1 x_2 &= d_1 \quad ; \quad i = 1 \\
a_i x_{i-1} + b_i x_i + c_i c_{i+1} &= d_i \quad ; \quad i = 2, \ldots, N-1 \\
a_n x_{n-1} + b_n x_n &= d_n \quad ; \quad i = N
\end{aligned}
$$

$$
\hat{c}_i = \begin{cases} \frac{a_1}{b_1}, & i = 1 \\ \frac{c_i}{b_i - \hat{c}_{i-1} a_i}, & i = 2, \ldots, n-1 \end{cases}
$$

$$
\hat{d}_i = \begin{cases} \frac{d_1}{b_1}, & i = 1 \\ \frac{d_i - \hat{d}_{i-1} a_i}{b_i - \hat{c}_{i-1} a_i}, & i = 2, \ldots, n \end{cases}
$$

$$
\begin{aligned}
x_n &= \hat{d}_n \\
x_i &= \hat{d}_i - \hat{c}_i x_{i+1}, \qquad i = n-1, \ldots, 1
\end{aligned}
$$

Ejemplo: Flujo en un acuífero confinado 2D

La distribución de carga hidráulica en un acuífero confinado 2D es modelado por,

$$T\left[\frac{\partial^2 h}{\partial x^2}+\frac{\partial^2 h}{\partial y^2}\right]=S\frac{\partial h}{\partial t} \qquad \text{o} \qquad D\nabla^2 h=\frac{\partial h}{\partial t}$$

Aplicando diferencias finitas en un *stencil* de 5-puntos y una aproximación explícita para discretizar la derivada temporal,

$$\frac{h_{i,j}^{n+1}-h_{i,j}^{n}}{\Delta t}=D\left[\frac{h_{i+1,j}^{n}-2h_{i,j}^{n}+h_{i-1,j}^{n}}{(\Delta x)^2}+\frac{h_{i,j+1}^{n}-2h_{i,j}^{n}+h_{i,j-1}^{n}}{(\Delta y)^2}\right]$$

Asumiendo $\Delta x=\Delta y=\Delta$ y tomando $\mathcal{D}=D\Delta t/\Delta$,

$$h_{i,j}^{n+1}=(1-4\mathcal{D})\,h_{i,j}+\mathcal{D}\left(h_{i+1,j}^{n}+h_{i-1,j}^{n}+h_{i,j+1}^{n}+h_{i,j-1}^{n}\right)$$

Ejemplo: Flujo en un acuífero confinado 2D (cont.)

Si definimos un orden para los nodos de la grilla, tal que podemos asignar un número único a cada nodo $k = i \cdot N_x + j$; entonces podemos escribir las expresiíon anterior en notación matricial,

$$\mathbf{h}^{n+1} = \mathbf{A}\mathbf{h}^n$$

donde,

$$A_{k,k} = 1 - 4\mathcal{D},$$

$$A_{k+1,k} = A_{k-1,k} = A_{k+N_x,k} = A_{k-N_x,k} = 1$$

En este caso el tamaño de la matriz $\mathbf{A}$ es NxN con $N = N_x \cdot N_y$.

Por ejemplo, si $N_x = N_y = 1000$ obtenemos una matriz de 10^6x10^6 elementos, pero con solo $5N - 2(N_x - 1)$ coeficientes que son no cero.

Matriz para EDP elíptica en 2D

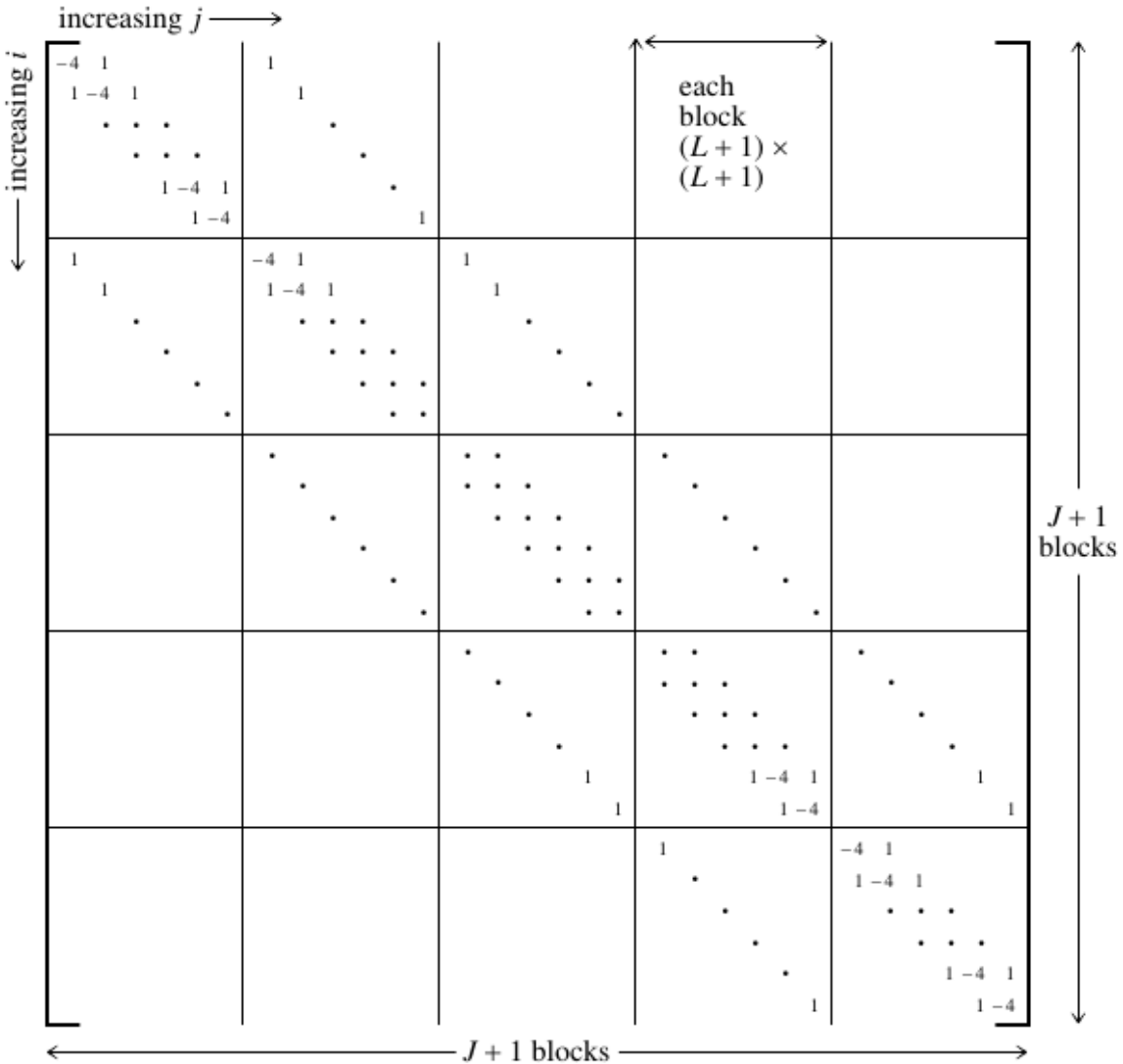


Figure 19.0.3. Matrix structure derived from a second-order elliptic equation (here equation 19.0.6). All elements not shown are zero. The matrix has diagonal blocks that are themselves tridiagonal, and sub- and super-diagonal blocks that are diagonal. This form is called "tridiagonal with fringes." A matrix this sparse would never be stored in its full form as shown here.

\# coeficients no cero,

$$\begin{aligned} & (N - 2N_x) \cdot 5 \\ + \quad & 2(N_x - 1) \cdot 4 \\ + \quad & 2 \cdot 3 \\ = \quad & 5N - 2(N_x - 1) \end{aligned}$$

Matrices *sparse*

En este caso es incluso más importante utilizar un sistema de almacenamiento eficiente de la matriz para no requerir $O(N^2)$ espacio de memoria.

También es importane utilizar algoritmos óptimos para la solución del sistema lineal. Existen dos tipos:

1. *Direct Sparse Solvers*: Invierten la matriz requeriendo un mínimo de espacio adicional para guardar resultados intermedios. Ejemplo: UMFPACK, MUMPS, PARDISO (Intel MKL), etc.

2. **Métodos iterativos**: Evitan invertir la matriz. En general, solo requieren multiplicaciones y adiciones de matrices y vectores . La principal desventaja es que la convergencia no está garantizada. En algunos casos pueden requerir mucho más tiempo que los algoritmos de resolución directa. Ejemplos: CG (conjugate gradient), BiCGStab, GMRES, etc.