Programación Numérica

Dr. Alfonso Alba Cadena
fac@fc.uaslp.mx

Facultad de Ciencias
UASLP

Introducción a las notas del curso

- Estas notas están diseñadas para ser una guía en un curso básico de métodos numéricos. La metodología sugerida es exponer cada uno de los métodos en clase, resaltando sus ventajas y desventajas, implementar algunos de los métodos y dejar otros como ejercicios, y utilizar los métodos para resolver problemas prácticos.

Objetivos Generales

- Entender los métodos numéricos utilizados para la solución de ecuaciones, sistemas de ecuaciones, interpolación, regresión lineal, diferenciación e integración.

- Ser capaces de implementar estos métodos en un lenguaje de uso común como C/C++, o de uso específico como Matlab u Octave.

- Desarrollar una librería de métodos numéricos para su uso en futuros cursos.

Contenido

Bibliografía sugerida

- **ANALISIS NUMERICO.**
  Richard L. Burden, J. Douglas Faires. Thompson Editores.

- **METODOS NUMERICOS PARA INGENIEROS.**
  Steven C. Chapra, Raymond P. Canale. Mc Graw Hill.

- **COMO PROGRAMAR C++.**
  Deitel y Deitel. Prentice Hall.

Unidad I

Introducción a Octave

GNU Octave...

- Es un lenguaje de alto nivel orientado al cómputo numérico
- Trabaja nativamente con vectores y matrices
- Es altamente compatible con Matlab
- Puede extenderse mediante funciones escritas en C/C++
- Es de distribución gratuita

Octave puede descargarse de `http://www.octave.org`

Operaciones con matrices y vectores

- En **Octave** se pueden definir matrices escribiendo sus elementos entre corchetes.

- La coma separa los elementos en columnas, y el punto y coma los separa en renglones.

Ejemplo: `m = [1, 2, 3; 4, 5, 6]` asigna a la variable `m` la matriz

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}.$$

Acceso a los elementos de una matriz

Dada una matriz `m` se puede tener acceso a cualquier elemento, renglón, columna, o sub-matriz de `m`:

- `m(i,j)` hace referencia a un elemento.
- `m(i,:)` hace referencia al `i`-ésimo renglón.
- `m(:,j)` hace referencia a la `j`-ésima columna.
- `m(i1:i2, j1:j2)` hace referencia a una sub-matriz.

Aritmética de matrices

- Suma y resta: `a + b`, `a - b`

- Producto matricial: `a * b`

- Producto elemento por elemento: `a .* b`

- Transpuesta conjugada: `a'`

Matrices especiales

Las siguientes funciones de **Octave** devuelven matrices de utilidad general.

- Identidad: `eye(n, m)`
- Unos: `ones(n, m)`
- Ceros: `zeros(n, m)`
- Ruido uniforme: `rand(n, m)`
- Ruido normal: `randn(n, m)`
- Vector de n valores equiespaciados: `linspace(base, limit, n)`
- Rango de índices: `a:b` (devuelve el vector `[a, a+1, ..., b]`)

Ejercicios

Defina las siguientes matrices en **Octave** :

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 1 \\ 2 & 0 \\ 1 & -2 \end{bmatrix}, \quad C = 3 \cdot I_{3\times 3}.$$

Usando las matrices anteriores, calcule las siguientes expresiones:

a) $A \times B$

b) $B \times A - C$

c) $A + \lambda N$, donde $\lambda = 0.1$ y N es ruido uniforme

Funciones

Una función de **Octave** toma cero o mas parámetros (escritos entre paréntesis), realiza algún procedimiento, y puede o no devolver algún resultado.

Ejemplos de funciones:

```
> cos(1)
ans = 0.54030
> ones(1, 5)
ans =
  1  1  1  1  1
> floor(mean([1, 2, 3, 4, 5]))
ans = 3
```

Definición de funciones

Uno puede escribir sus propias funciones de **Octave** usando la siguiente sintaxis:

`function` *resultado* = *nombre* (*parametros*)
    cuerpo de la función
`end`

Ejemplo:

```
> function y = cuad(x)
> y = x * x;
> end
> cuad(5)
ans = 25
```

Librerías de funciones

Para poder utilizar una función en el futuro, sin tener que escribirla nuevamente, es necesario guardarla en un archivo de texto con el mismo nombre que la función y extensión `.m`

```
# Archivo:       cuad.m
# Descripcion:   cuad(x) devuelve el cuadrado de x

function y = cuad(x)
    y = x * x;
end
```

Buenas prácticas de programación

- Escribir funciones de librería para realizar tareas comunes.

- Dividir un problema grande en sub-problemas mas pequeños cuya implementación sea sencilla.

- Agregar comentarios a las funciones de librería que describan la tarea que realizan y los parámetros que toman.

- Limitar las funciones a las tareas que deben realizar de manera que sean claras y reutilizables.

Ejercicios:

1. Escribir una función `fact(x)` que calcule y devuelva el factorial de `x`. Para esto, investigar cómo realizar ciclos (for, while, etc.) en **Octave** .

2. Escribir una función que tome como entrada dos vectores y devuelva el producto punto. La función debe verificar primero que los vectores tengan la misma longitud. Usar las estructuras `if`, `for`, y la función `length()`.

3. Utilizando la función anterior, escriba una función que calcule la norma de un vector x (dada por $\sqrt{x \cdot x}$).

Unidad II

Solución de ecuaciones no-lineales

Descripción del problema

- Se tiene una función $y = f(x)$, y se desea encontrar las raíces de $f(x)$, es decir, aquellos valores de x para los cuales $f(x) = 0$.

- Alternativamente, si se desea resolver la ecuación $f(x) = a$, podemos definir $g(x) = f(x) - a$ y encontrar las raíces de $g(x)$.

Detalles sobre la implementación

- Al implementar un método para encontrar las raíces de una función $f(x)$, por lo general deseamos que el método pueda trabajar con *cualquier* función $f(x)$ que nosotros proporcionemos.

- Deseamos entonces poder escribir una función de **Octave** que encuentre la raíz de *otra* función de **Octave** que nosotros le proporcionemos como parámetro.

- Una manera de lograr esto en **Octave** es utilizando la función `feval`, la cual sirve para evaluar la función que se le pasa como parámetro. Ejemplo:

```
> feval("sqrt", 4)
ans =  2
```

- Los parámetros que recibe `feval` son el nombre de la función a evaluar, escrito entre comillas, y los parámetros de la función a evaluar.

Ejemplo del uso de `feval`

Considere la siguiente función:

```
function plotf(f, xi, xf)
  x = linspace(xi, xf, 200);
  plot(x, feval(f, x));
end
```

Ejemplo de aplicación:

```
> plotf("cos", 0, 4*pi);
> plotf("exp", -1, 1);
```

Método de bisección

- Supongamos que tenemos una función $f(x)$ y conocemos dos puntos $x = a$ y $x = b$ en los cuales f tiene signo distinto; es decir $f(a) \cdot f(b) < 0$.

- Entonces, si f es contínua, podemos asegurar que existe una raíz de f en el intervalo (a, b).

- Podemos entonces reducir este intervalo, por ejemplo, partiéndolo por la mitad, y luego ver en qué mitad se presenta el cambio de signo

- De esta forma, podemos reducir iterativamente el intervalo donde se ubica la raíz hasta encontrarla.

Algoritmo básico de bisección

Dada la función $f(x)$ y el intervalo (a, b)

1. Verificar que $f(a) \cdot f(b) < 0$. De lo contrario, no podemos asegurar que existe una raíz en el intervalo dado y el algoritmo debe terminar.

2. Iterar hasta que x_r converja

   (a) Obtener una aproximación de la raíz x_r como $x_r = \frac{a+b}{2}$.

   (b) Si $f(a) \cdot f(x_r) < 0$, entonces la raíz está en la mitad izquierda del intervalo. Por lo tanto, hacer $b = x_r$.

   (c) En caso contrario, entonces la raíz está en la mitad derecha del intervalo. Por lo tanto, hacer $a = x_r$.

Convergencia de x_r

- La convergencia de x_r se puede evaluar calculando el error porcentual de la raíz estimada x_r con respecto al verdadero valor x_{real} de la raíz:

$$\text{error} = \left| \frac{x_{\text{real}} - x_r}{x_{\text{real}}} \right| .$$

- No conocemos el valor de x_{real}, pero podemos usar en su lugar la mejor estimación que se tiene hasta el momento y compararla con la estimación obtenida en la iteración anterior.

- De esta forma, el criterio de error queda como

$$\text{error} = \left| \frac{x_r^{\text{nuevo}} - x_r^{\text{ant}}}{x_r^{\text{nuevo}}} \right| .$$

- Se llega a la convergencia cuando el error es menor que un umbral establecido (e.g. 0.00001).

Implementación del método de bisección

```
function xr = biseccion(f, a, b, umbral = 0.000001)
  if feval(f, a) * feval(f, b) >= 0
    error("No hay cambio de signo entre los extremos del intervalo.");
  end

  error = 1;
  xr = a;
  while error > umbral
    xa = xr;
    xr = (a + b) / 2;
    error = abs((xr - xa) / xr);
    if feval(f, a) * feval(f, xr) < 0
      b = xr;
    else
      a = xr;
    end
  end
end
```

Ejemplos

Ejemplo 1: Encontrar la raíz de $f(x) = x + \log x$.

```
> function y = f(x); y = x + log(x); end;
> r = biseccion("f", 0.1, 1)
r = 0.56714
> f(r)
ans = 2.8227e-007
```

Ejemplo 2: Resolver la ecuación $x^3 + 3 = 2x$.

```
> function y = p(x); y = x.^3 - 2*x + 3; end;
> x = linspace(-3, 3, 100); plot(x, p(x));
> biseccion("p", -3, -1)
ans = -1.8933
```

Método de la falsa posición

- Considerar una función $f(x)$ con una raíz $x_0 \in [a, b]$. Si $f(a)$ es más cercana a cero que $f(b)$, entonces es muy probable que x_0 esté más cerca de a que de b.

- Esto sugiere una manera de acelerar la convergencia del método de bisección: tomar como siguiente estimación el punto donde se intersecta el eje x con la recta que pasa por los puntos $(a, f(a))$ y $(b, f(b))$.

- Este punto está dado por

$$x_r = b - \frac{f(b)(a-b)}{f(a)-f(b)}.$$

Algoritmo de la falsa posición

Dada la función $f(x)$ y el intervalo (a, b)

1. Verificar que $f(a) \cdot f(b) < 0$. De lo contrario, no podemos asegurar que existe una raíz en el intervalo dado y el algoritmo debe terminar.

2. Iterar hasta que x_r converja

   (a) Obtener una aproximación de la raíz x_r como

   $$x_r = b - \frac{f(b)(a-b)}{f(a) - f(b)}.$$

   (b) Si $f(a) \cdot f(x_r) < 0$, entonces la raíz está en la mitad izquierda del intervalo. Por lo tanto, hacer $b = x_r$.

   (c) En caso contrario, entonces la raíz está en la mitad derecha del intervalo. Por lo tanto, hacer $a = x_r$.

Ejercicios

1. Escriba una función que implemente el método de la falsa posición. Pruebe la función con los ejemplos vistos anteriormente. **Sugerencia:** copie la función de bisección y modifíque las partes que sean necesarias.

2. Modifique temporalmente las funciones de bisección y falsa posición de manera que devuelvan un vector $[x_r, n]$, donde x_r es la raíz encontrada y n es el número de iteraciones realizadas. Con estas nuevas funciones, indique cuál de los dos métodos converge más rápidamente para los ejemplos vistos anteriormente.

Método de iteración de punto fijo

- Un *punto fijo* de una función $g(x)$ es un valor p tal que $g(p) = p$.

- Dado un punto inicial p_0, podemos generar la secuencia $p_0, p_1, p_2, \ldots$ haciendo $p_n = g(p_{n-1})$ para $n = 1, 2, \ldots$.

- Si la secuencia converge a un punto p, entonces p es punto fijo de g, ya que

$$p = \lim_{n \to \infty} p_n = \lim_{n \to \infty} g(p_{n-1}) = g\left(\lim_{n \to \infty} p_{n-1}\right) = g(p).$$

Existencia y unicidad del punto fijo

- **Teorema:** Si $g(x)$ es contínua en $[a, b]$ y $g(x) \in [a, b]$ para toda $x \in [a, b]$, entonces $g(x)$ tiene un punto fijo en $[a, b]$.

- Además, si $g(x)$ es derivable en (a, b) y $|g'(x)| < 1$ para todo $x \in (a, b)$, entonces el punto fijo es único.

- Si se cumplen las dos condiciones anteriores, entonces para cualquier número $p_0 \in [a, b]$, la sucesión definida por

$$p_n = g(p_{n-1}),\ \ n \geq 1$$

converge en el único punto fijo p de $g(x)$ en $[a, b]$.

Solución de ecuaciones por iteración de punto fijo

- El problema de encontrar la raíz de la función $f(x)$ puede transformarse en el problema de encontrar el punto fijo de una función $g(x)$, la cual puede definirse de varias formas, por ejemplo:

$$g(x) = x + 5f(x), \quad g(x) = x - f(x), \quad g(x) = x + [f(x)]^2 .$$

- Es necesario encontrar una $g(x)$ que cumpla con las condiciones anteriores; en particular, que $|g'(x)| < 1$ en $[a, b]$.

Algoritmo de iteración de punto fijo

Dada la función $f(x)$, una aproximación inicial p_0, un umbral de error e_u y un límite de iteraciones n_{max}:

1. Encontrar una función $g(x)$ que cumpla con las condiciones de existencia y unicidad del punto fijo.

2. Inicializar $n = 0$, $e = e_u$, y $p = p_0$.

3. Iterar mientras $n < n_{\text{max}}$ y $e \geq e_u$

   (a) $p_{\text{ant}} = p$

   (b) $p = g(p)$

   (c) $n = n + 1$

   (d) $e = \left| \frac{p - p_{\text{ant}}}{p} \right|$

Ejemplo

- Se desea encontrar la raíz de $f(x) = e^x - x^2$ en el intervalo $[-2, 2]$.

- Definimos $g(x) = x - \frac{1}{2}f(x)$.

- Iniciamos con $p_0 = 0$ e iteramos 10 veces $p_n = g(p_{n-1})$ para encontrar el punto fijo de g (y la raíz de f).

```
> function y = f(x); y = exp(x) - x .^ 2; end
> function y = g(x); y = x - f(x) / 2; end
> p = zeros(1,10);
> for n = 1:10; p(n) = g(p(n-1)); end
> p(10)
ans = -0.70347
> g(p(10))
ans = -0.70347
> f(p(10))
ans = 0
```

Método de Newton-Raphson

- **Idea principal:** Dada una función $f(x)$, uno comienza con una estimación inicial x_0 cercana a la raíz de f, y luego uno aproxima la función por su tangente en el punto x_0. El cruce de la tangente con el eje X nos dará una nueva aproximación, típicamente mejor que x_0.

- La tangente a la función $f(x)$ se encuentra a partir de la derivada $f'(x)$, por lo que es necesario que f sea derivable.

- El algoritmo puede diverger si se llega a puntos donde la derivada es cercana o igual a cero.

Cálculo de la nueva estimación

- Dada la función $f(x)$ y una estimación inicial de la raíz x_a, obtenemos la nueva estimación x_r como el punto donde la recta tangente a f en x_a cruza el eje de las X.

- La pendiente de la tangente está dada por la derivada $f'(x_a)$.

- Considerando el triángulo rectángulo que se obtiene al intersectar la tangente, el eje X, y la recta vertical $x = x_a$, podemos ver también que

$$f'(x_a) = \frac{f(x_a)}{x_a - x_r}.$$

- Entonces, la nueva estimación puede obtenerse como

$$x_r = x_a - \frac{f(x_a)}{f'(x_a)}, \;\; f'(x_a) \neq 0.$$

Algoritmo de Newton-Raphson

Dada la función $f(x)$ y su derivada $f'(x)$, una aproximación inicial x_a, un umbral de error e_u y un límite de iteraciones n_{max},

1. Inicializar $n = 0$ y $e = e_u$.

2. Iterar mientras $n < n_{\text{max}}$ y $e \geq e_u$

   (a) $x_r = x_a - \frac{f(x_a)}{f'(x_a)}$

   (b) $e = \left|\frac{x_r - x_a}{x_r}\right|$

   (c) $x_a = x_r$

   (d) $n = n + 1$

Implementación en Octave

```
function xr = newrap(f, df, xa, umbral, nmax)
  n = 0;
  e = umbral;
  while (n < nmax) && (e >= umbral)
    xr = xa - feval(f, xa) / feval(df, xa);
    e = abs((xr - xa) / xr);
    xa = xr;
    n++;
  end
end
```

Raíces de polinomios

- Un *polinomio* de grado n es una función de la forma

$$p(x) = a_1 x^n + a_2 x^{n-1} + a_3 x^{n-2} + \ldots + a_n x + a_{n+1}.$$

- El polinomio está completamente determinado por los coeficientes $a_1, a_2, \ldots, a_{n+1}$, por lo que podemos representar a $p(x)$ por medio del vector

$$[a_1, a_2, \ldots, a_{n+1}].$$

  Esta es la manera en que Octave y Matlab representan polinomios. Notar que el tamaño del vector es igual al grado del polinomio mas uno.

Teorema Fundamental del Algebra

- Todo polinomio $p(x)$ de grado n tiene exactamente n raíces $x_1, \ldots, x_n$ (incluyendo multiplicidades) y puede factorizarse como

$$p(x) = (x - x_1)(x - x_2) \ldots (x - x_n).$$

- Si los coeficientes de $p(x)$ son todos reales, entonces sus raíces serán reales, o pares de complejos conjugados.

- Si $p(x)$ tiene uno o más coeficientes complejos, entonces al menos una de sus raíces será compleja.

Operaciones con polinomios

- Utilizando la representación vectorial, es muy sencillo implementar algunas operaciones básicas con polinomios, por ejemplo:

  1. Evaluación de un polinomio en un punto
  2. Suma de polinomios
  3. Producto de un polinomio por una constante
  4. Derivada de un polinomio
  5. Producto de dos polinomios
  6. División de un polinomio entre otro de la forma $(x - c)$, donde c es una constante.

Evaluación de un polinomio

- Sea $a = [a_1, a_2, \ldots, a_{n+1}]$ el vector de coeficientes de un polinomio $p(x)$. Para evaluar el polinomio en un valor dado de x, podemos simplemente aplicar la definición:

$$p(x) = a_1 x^n + a_2 x^{n-1} + \ldots + a_n x + a_{n+1}.$$

- Notar que el resultado es básicamente una sumatoria, por lo que el algoritmo de evaluación puede ser el siguiente (dados a y x):

  1. Hacer $s = 0$ y $N =$ length(a).

  2. Para k desde 1 hasta N, hacer $s = s + a_k\ x^{N-k}$.

Evaluación de un polinomio

- Una manera más eficiente de evaluar un polinomio se obtiene al escribirlo en forma anidada:

$$\begin{aligned} p(x) &= a_1x^n + a_2x^{n-1} + \ldots + a_nx + a_{n+1} \\ &= ((((a_1)x + a_2)x + a_3) \ldots)\, x + a_{n+1}. \end{aligned}$$

- De forma que el algoritmo de evaluación queda como:

  1. Hacer $s = a_1$ y $N = \text{length}(a)$.

  2. Para k desde 2 hasta N, hacer $s = sx + a_k$.

Suma de polinomios

- La suma de polinomios se realiza término a término, lo que equivale a una simple suma de coeficientes.

- El grado del polinomio resultante es igual al mayor de los grados de los sumandos.

- Sean a y b los vectores de coeficientes de dos polinomios. Si los polinomios son del mismo grado, entonces los coeficientes de la suma están dados simplemente por el vector $a + b$.

- Si los polinomios no son del mismo grado, entonces será necesario insertar ceros a la izquierda de alguno de los vectores para que tengan el mismo tamaño.

Algoritmo para la suma de polinomios

Dados los vectores de coeficientes a y b,

1. Sea $n = \max(\text{length}(a), \text{length}(b))$ el grado del polinomio resultante.

2. Insertar $n - \text{length}(a)$ ceros a la izquierda de a.

3. Insertar $n - \text{length}(b)$ ceros a la izquierda de b.

4. Devolver el vector de coeficientes $a + b$.

Producto de un polinomio por una constante

- Sea $p(x)$ un polinomio con coeficientes $a = [a_1, a_2, \ldots, a_{n+1}]$.

- El producto $kp(x)$ donde k es una constante es otro polinomio con coeficientes
$$ka_1, ka_2, \ldots, ka_{n+1}.$$

- Por lo tanto, podemos representar esta operación simplemente como el producto ka.

Derivada de un polinomio

- Dado el polinomio $p(x) = a_1 x^n + a_2 x^{n-1} + \ldots + a_n x + a_{n+1}$, su derivada $p'(x)$ es otro polinomio que está dado por:

$$p'(x) = n a_1 x^{n-1} + (n-1) a_2 x^{n-2} + \ldots + a_n.$$

- Esto sugiere el siguiente algoritmo para encontrar la derivada de un polinomio:

Dados el vector de coeficientes a de un polinomio,

1. Hacer $n =$ length(a) $- 1$.
2. Para i desde 1 hasta n, hacer $b_i = (n + 1 - i) * a_i$.
3. Devolver el vector b de coeficientes de la derivada.

Producto de dos polinomios

- Dados dos polinomios

$$p(x) = a_1 x^n + a_2 x^{n-1} + \ldots + a_n x + a_{n+1},$$

$$q(x) = b_1 x^m + b_2 x^{m-1} + \ldots + b_m x + b_{m+1},$$

  el producto pq es el polinomio de grado $n+m$ que se obtiene multiplicando cada término de p con cada término de q y sumando todos los términos que resulten:

$$pq(x) = \sum_{i=1}^{n} \sum_{j=1}^{m} a_i b_j x^{n+m+2-i-j}.$$

- Cada coeficiente c_k del polinomio pq se obtiene como la suma de coeficientes $a_i b_j$ de aquellos términos cuyo grado es $n+m+1-k$; es decir, aquellos términos para los cuales $i+j-1=k$.

- Entonces, podemos calcular los coeficientes c_k del producto como

$$c_k = \sum_{i,j \ : \ i+j-1=k} a_i b_j.$$

Algoritmo para el producto de dos polinomios

- Dados los vectores de coeficientes a y b de dos polinomios,

  1. Hacer $n = \text{length}(a)$ y $m = \text{length}(b)$.
  2. Inicializar el vector c de longitud $n + m - 1$ con ceros.
  3. Para i desde 1 hasta n, hacer lo siguiente:
     (a) Para j desde 1 hasta m, hacer lo siguiente:
        i. Hacer $k = \text{i} + \text{j - 1}$.
        ii. Hacer $c_k = c_k + a_i b_j$.
  4. Devolver el vector c de coeficientes del producto.

División de un polinomio entre $(x - c)$

- Sea $p(x) = a_1 x^n + a_2 x^{n-1} + \ldots + a_n x + a_{n+1}$. Si $(x - c)$ es un factor de $p(x)$, entonces el cociente $p(x)/(x - c)$ da como resultado otro polinomio $q(x)$.

- Podemos encontrar $q(x)$ utilizando la división sintética, con la cual se llega a que

$$
\begin{aligned}
q(x) \quad = \quad & a_1 x^{n-1} + \\
& (a_1 c + a_2) x^{n-2} + \\
& ((a_1 c + a_2) c + a_3) x^{n-3} + \\
& \qquad \qquad \vdots \\
& (((a_1 c + a_2) c + a_3) \ldots) c + a_n
\end{aligned}
$$

Algoritmo de división

Dado el vector de coeficientes a de un polinomio, y una constante c, la división del polinomio entre $(x - c)$ se obtiene como sigue:

1. Hacer $n = \text{length}(a) - 1$.

2. Hacer $b_1 = a_1$.

3. Para i desde 2 hasta n

   (a) Hacer $b_i = b_{i-1}c + a_i$

4. Devolver b como el vector de coeficientes del cociente.

Raíces de polinomios

Con las herramientas desarrolladas hasta el momento, es posible encontrar todas las raíces reales* de un polinomio $p(x)$ utilizando el siguiente procedimiento:

1. Calcular la derivada $p'(x)$ de $p(x)$.

2. Utilizar el método de Newton-Raphson con un punto inicial elegido al azar para encontrar una raíz r de $p(x)$. Si el método diverge, volver a intentar con otro punto inicial.

3. Reemplazar $p(x)$ con el resultado de dividir $p(x)/(x-r)$ y volver al paso 1.

* Para encontrar todas las raíces de un polinomio (reales y complejas), ver el método de Müller.

Unidad III

Sistemas de ecuaciones lineales

Introducción

- Una **matriz** es un arreglo rectangular de números. Algunos ejemplos de matrices son:

$$\begin{pmatrix} 2 & 4 & 1 \\ 8 & -1 & 5 \\ 1 & 3 & -2 \end{pmatrix} \qquad \begin{pmatrix} 3 & 8 \\ -2 & 3 \\ 6 & 2 \end{pmatrix} \qquad \begin{pmatrix} 5 & -1 & 3 \\ 2 & 1 & 1 \end{pmatrix}$$

- El tamaño de una matriz se especifica como número de renglones por número de columnas.

- Una matriz es **cuadrada** si tiene el mismo número de renglones que de columnas.

Vectores y escalares

- Un **vector** es una matriz que consta solamente de un renglón o una columna. Ejemplos de vectores:

$$\begin{pmatrix} 2 & 4 & 1 & 8 \end{pmatrix} \qquad \begin{pmatrix} 3 \\ -2 \\ 0 \\ 4 \end{pmatrix}$$

Vector renglón   Vector columna

- Un **escalar** es una matriz de 1×1; es decir, un solo número. Ejemplos de escalares son:

$$8, \quad 3.1416, \quad 2 - 3i$$

Notación

- Las matrices se denotan mediante letras mayúsculas:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

- Los vectores se representan con letras minúsculas:

$$a = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} \qquad b = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

- Los escalares se representan también con letras minúsculas:

$$x = 4.5, \qquad y = 3 + 4i$$

Notación

- En caso de ambigüedad entre vectores y escalares, puede agregarse una flecha a las variables que representan vectores, por ejemplo: $\vec{i}, \vec{j}$.

- Algunas veces es útil referirse a una matriz por medio de sus elementos:

$$(a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}.$$

(Notar que el primer subíndice denota el renglón y el segundo la columna)

Combinación lineal

- Sean $v_1, v_2, \ldots, v_n \in V$ vectores de igual longitud. Cualquier vector de la forma

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_3 v_3,$$

donde $\alpha_1, \alpha_2, \ldots, \alpha_3$ son escalares y no son todos cero, se llama **combinación lineal** de $v_1, v_2, \ldots, v_n$.

Dependencia e independencia lineal

- Sean $v_1, v_2, \ldots, v_n$ vectores de igual longitud. Se dice que son **linealmente dependientes** si existen escalares $c_1, c_2, \ldots, c_n \in \mathbb{R}$ tales que al menos uno de ellos es distinto de cero y

$$c_1v_1 + c_2v_2 + \ldots + c_nv_n = 0.$$

- Los vectores $v_1, v_2, \ldots, v_n$ son linealmente dependientes si cualquiera de ellos puede escribirse como una combinación lineal de los otros.

- Si los vectores no son linealmente dependientes, se dice entonces que son **linealmente independientes**.

Operaciones algebráicas

- Suma y resta de matrices:

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \pm \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & & \vdots \\ b_{m1} & \cdots & b_{mn} \end{pmatrix} = \begin{pmatrix} a_{11} \pm b_{11} & \cdots & a_{1n} \pm b_{1n} \\ \vdots & & \vdots \\ a_{m1} \pm b_{m1} & \cdots & a_{mn} \pm b_{mn} \end{pmatrix}$$

- Multiplicación por un escalar:

$$c \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} = \begin{pmatrix} ca_{11} & \cdots & ca_{1n} \\ \vdots & & \vdots \\ ca_{m1} & \cdots & ca_{mn} \end{pmatrix}$$

Producto de matrices

- Producto de un vector renglón y un vector columna:

$$\begin{pmatrix} a_1 & a_2 & \cdots & a_n \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = a_1b_1 + a_2b_2 + \ldots + a_nb_n$$

- El producto de dos matrices $A = (a_{ij})$ y $B = (b_{ij})$ es una matriz $C = (c_{ij})$ dada por

$$\begin{aligned} c_{ij} &= \text{(renglón } i \text{ de } A) \cdot \text{(columna } j \text{ de } B) \\ &= a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{in}b_{nj} \end{aligned}$$

*Notar que el número de columnas de A debe ser igual al número de renglones de B

Propiedades conmutativas, asociativas y distributivas

$$A + 0 = 0 + A = A \qquad \text{(identidad aditiva)}$$

$$0A = A0 = 0$$

$$A + B = B + A \qquad \text{(conmutatividad aditiva)}$$

$$(A + B) + C = A + (B + C) \qquad \text{(asociatividad aditiva)}$$

$$\alpha(A + B) = \alpha A + \alpha B \qquad \text{(distributividad escalar)}$$

$$(\alpha + \beta)A = \alpha A + \beta A \qquad \text{(distributividad escalar)}$$

$$(AB)C = A(BC) \qquad \text{(asociatividad multiplicativa)}$$

$$A(B + C) = AB + AC \qquad \text{(distributividad)}$$

$$(A + B)C = AC + BC \qquad \text{(distributividad)}$$

Matrices transpuesta y conjugada

- La **transpuesta** de una matriz A es otra matriz A^t que se obtiene al intercambiar los renglones por las columnas de A. Ejemplo:

$$A = \begin{pmatrix} 3 & 8 \\ -2 & 3 \\ 6 & 2 \end{pmatrix} \qquad \Rightarrow \qquad A^t = \begin{pmatrix} 3 & -2 & 6 \\ 8 & 3 & 2 \end{pmatrix}$$

- La transpuesta de un vector renglón es un vector columna, y viceversa.

- La **conjugada** $\bar{A}$ de una matriz A con elementos complejos, es la matriz que se obtiene tomando el complejo conjugado de cada elemento de A. Ejemplo:

$$A = \begin{pmatrix} 3+2i & 8-i \\ -2 & 1+3i \\ -6+4i & -5i \end{pmatrix} \qquad \Rightarrow \qquad \bar{A} = \begin{pmatrix} 3-2i & 8+i \\ -2 & 1-3i \\ -6-4i & 5i \end{pmatrix}$$

Matriz asociada

- Considerar el siguiente sistema de ecuaciones lineales:

$$\begin{aligned} 2x_1 + 4x_2 + 6x_3 &= 4 \\ x_1 - 3x_2 + x_3 &= 6 \\ 2x_2 - 7x_3 &= 16 \end{aligned}$$

- La **matriz de coeficientes** representa los coeficientes de las variables del sistema. Para el ejemplo anterior es:

$$\begin{pmatrix} 2 & 4 & 6 \\ 1 & -3 & 1 \\ 0 & 2 & -7 \end{pmatrix}.$$

- La **matriz asociada** al sistema de ecuaciones se obtiene agregando el lado derecho del sistema (términos independientes) a la matriz de coeficientes:

$$\left(\begin{array}{ccc|c} 2 & 4 & 6 & 4 \\ 1 & -3 & 1 & 6 \\ 0 & 2 & -7 & 16 \end{array}\right).$$

Determinantes, menores y cofactores

- El determinante de una matriz $|A| = \det A$ de una matriz $A = (a_{ij})$ de 2×2 se define como:

$$\det A = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}.$$

- El determinante de una matriz $A = (a_{ij})$ de 3×3 está dado por:

$$\det A = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

Determinantes, menores y cofactores

- Sea $A = (a_{ij})$ una matriz de $n \times n$. La matriz M_{ij} que se obtiene al eliminar el i-ésimo renglón y la j-ésima columna de A se conoce como la **menor** ij de A.

- El **cofactor** ij de A, denotado por A_{ij}, es un escalar dado por
$$A_{ij} = (-1)^{i+j}|M_{ij}|.$$

- El determinante de A está dado por
$$\det A = |A| = a_{11}A_{11} + a_{12}A_{12} + \ldots + a_{1n}A_{1n}.$$

Rango de una matriz

- El **rango** $\rho(A)$ de una matriz A es el número de columnas o renglones que son linealmente independientes.

- $\rho(A) = 0$ si y solo si $A = 0$.

- Si A es una matriz de $m \times n$, entonces $\rho(A) \leq \min(m, n)$.

- Un sistema de n ecuaciones tiene al menos una solución si y solo si su matriz de coeficientes y su matriz asociada tienen el mismo rango. Si el rango es igual n, entonces la solución es única.

Traza de una matriz

- La **traza** $\text{tr}(A)$ de una matriz $A = (a_{ij})$ de $n \times n$ es la suma de todos los elementos de la diagonal:

$$\text{tr}(A) = a_{11} + a_{22} + \ldots + a_{nn}.$$

- Propiedades:

  - $\text{tr}(A) = \text{tr}(A^t)$
  - $\text{tr}(AB) = \text{tr}(BA)$
  - $\text{tr}(A + B) = \text{tr}(A) + \text{tr}(B)$
  - $\text{tr}(cA) = c\,\text{tr}(A)$

Inversa de una matriz

- La **matriz identidad** I es una matriz cuadrada que tiene unos en la diagonal y ceros fuera de ella. Ejemplos:

$$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

- Propiedad: $IA = AI = A$ para toda matriz A.

- Sean A y B dos matrices de $n \times n$ tales que $AB = BA = I$. Entonces a B se le llama la **inversa** de A y se denota por A^{-1}, de manera que $AA^{-1} = A^{-1}A = I$.

- Si A tiene inversa, se dice que A es **invertible**.

Propiedades de las inversas

- Si A es invertible, entonces su inversa es única.

- $(AB)^{-1} = B^{-1}A^{-1}$.

- Si A es invertible, entonces $\det A \neq 0$ y $\det A^{-1} = 1/\det A$.

- Considerar el sistema de ecuaciones dado por $A\vec{x} = \vec{b}$, donde $\vec{x} = (x_1, \ldots, x_n)^t$ y $\vec{b} = (b_1, \ldots, b_n)^t$. Si A es invertible, entonces el sistema tiene una solución única dada por

$$x = A^{-1}b.$$

Teorema

Sea A una matriz de $n \times n$. Entonces, las siguientes afirmaciones son equivalentes:

- A es invertible.

- El sistema $Ax = b$ tiene solución única para cada n-vector b.

- $\det A \neq 0$.

- $\rho(A) = n$.

Sistemas de ecuaciones lineales

Un sistema de m ecuaciones con n incógnitas se representa por medio de su matriz asociada:

$$\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n & = & b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n & = & b_2 \\ \vdots & & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n & = & b_m \end{array}$$

$$\Downarrow$$

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array} \right)$$

Forma reducida de una matriz

- El **pivote** de un renglón es el primer número (de izquierda a derecha) distinto de cero en ese renglón.

- Una matriz se encuentra en **forma reducida** si cumple con lo siguiente:

  1. Los renglones cuyos elementos son todos cero aparecen en la parte inferior de la matriz.

  2. El pivote (si lo hay) en cualquier renglón es 1.

  3. El pivote de cualquier renglón está a la derecha del pivote del renglón anterior.

  4. Cualquier columna que contiene el pivote de un renglón tiene ceros en el resto de sus elementos.

- Si solo se cumplen las tres primeras condiciones, entonces se dice que la matriz está en forma **escalonada**.

Operaciones elementales

- Las operaciones elementales con renglones son:
  1. Multiplicar un renglón por un número distinto de cero: $R_i = cR_i$.
  2. Sumar un múltiplo de un renglón a otro: $R_j = R_j + cR_i$.
  3. Intercambiar dos renglones: $R_i \leftrightarrow R_j$.

- Si la matriz A se puede transformar en la matriz B mediante operaciones elementales, entonces se dice que A y B son **equivalentes por renglones**.

- Cualquier matriz se puede reducir a forma escalonada o reducida mediante operaciones elementales.

Reducción de matrices (método de Gauss)

Transformación de una matriz $A = (a_{i,j})$ de $m \times n$ a la forma escalonada:

1. Comenzar con el renglón $r = 1$ y la columna $c = 1$.
2. Buscar el elemento $a_{k,c}$ con mayor valor absoluto en la columna c.
3. Si $a_{k,c} = 0$, incrementar c y regresar al paso anterior.
4. Intercambiar los renglones r y k.
5. Multiplicar el renglón r por $1/a_{r,c}$.
6. Para cada renglón j debajo del renglón r (es decir, $j = r+1, r+2, \ldots, m$), sumar al renglón j el renglón r multiplicado por $-a_{j,c}$.
7. Incrementar r y c, y regresar al paso 2.

Reducción de matrices (método de Gauss-Jordan)

Transformación de una matriz $A = (a_{i,j})$ de $m \times n$ a la forma reducida:

1. Utilizar el método de Gauss para transformar la matriz a la forma escalonada.

2. Sea r el último renglón distinto de cero, y sea $a_{r,c}$ su pivote.

3. Reducir los elementos arriba del pivote sumándole a cada renglón $j = 1, \ldots, r-1$ el renglón r multiplicado por $-a_{j,c}$.

4. Decrementar r y regresar al paso anterior.

Solución de sistemas de ecuaciones lineales

- Resolver en el pizarrón el siguiente sistema:

$$\begin{aligned} 2x_1 + 4x_2 + 6x_3 &= 18 \\ 4x_1 + 5x_2 + 6x_3 &= 24 \\ 3x_1 + x_2 - 2x_3 &= 4 \end{aligned}$$

- Reducir la matriz primero a forma escalonada (método de Gauss) y luego a forma reducida (método de Gauss-Jordan).

Inversa de una matriz cuadrada

- Para encontrar la inversa de una matriz cuadrada A:

  1. Se escribe la matriz aumentada $(A|I)$.

  2. Se utilizan operaciones elementales para llevar la matriz A a su forma reducida.

  3. Se decide si A es invertible:

     (a) Si la forma reducida de A es la identidad, entonces la matriz que se tiene en el lado derecho es A^{-1}.

     (b) Si la reducción de A conduce a un renglón de ceros en el lado izquierdo, entonces A no es invertible.

Matrices elementales

- Una matriz E de $n \times n$ se llama **matriz elemental** si puede obtenerse a partir de aplicar una sola operación elemental a la matriz identidad.

- Una operación elemental en una matriz A se puede escribir como el producto EA, donde E es la matriz elemental correspondiente.

- Toda matriz elemental es invertible, y su inversa es una matriz del mismo tipo.

Matrices elementales

- La transformación de una matriz A a la forma escalonada o reducida se puede expresar como el producto

$$E_m E_{m-1} \ldots E_2 E_1 A,$$

donde E_i es la matriz elemental correspondiente a la i-ésima operación aplicada.

- Una matriz cuadrada es invertible si y sólo si es el producto de matrices elementales.

Factorización LU

- La forma escalonada de una matriz es una matriz triangular superior.

- Sea A una matriz cuadrada de $n \times n$. Si A se puede reducir por renglones a una matriz triangular superior U sin hacer permutaciones de renglones. Entonces existe una matriz triangular inferior L invertible con unos en la diagonal tal que $A = LU$. Si U tiene n pivotes (es decir, si A es invertible), entonces esta factorización es única.

Factorización LU

- Sea A una matriz de $n \times n$ y sean $E_1, E_2, \ldots, E_m$ las matrices elementales correspondientes a las operaciones requeridas (ninguna de ellas una permutación) para transformar A en una matriz triangular superior U; es decir,

$$U = E_m E_{m-1} \ldots E_2 E_1 A.$$

Entonces

$$A = \underbrace{E_1^{-1} E_2^{-1} \ldots E_m^{-1}}_{L} U$$

Factorización PA = LU

- En general, para cualquier matriz invertible de $n \times n$ existe una matriz de permutación P tal que $PA = LU$, donde L es triangular inferior con unos en la diagonal, y U es triangular superior.

- Notar que toda matriz de permutación es su propia inversa, y que si P es una matriz de permutación.

- Entonces, P se puede construír como el producto $P_k P_{k-1} \ldots P_2 P_1$ de todas las permutaciones que se realizan durante la transformación de A en U.

Factorización LU

- Encuentre la factorización LU de la matriz de coeficientes A del siguiente sistema:

$$\begin{aligned} 2x_1 + 4x_2 + 6x_3 &= 18 \\ 4x_1 + 5x_2 + 6x_3 &= 24 \\ 3x_1 + x_2 - 2x_3 &= 4 \end{aligned}$$

- Notar que $LUx = b$, por lo tanto, si se define $y = Ux$, entonces $Ly = b$. Entonces, para resolver el sistema:

  1. Se resuelve $Ly = b$ para y mediante sustitución hacia adelante.

  2. Se resuelve $Ux = y$ para x mediante sustitución hacia atrás.

Factorización LU y determinantes

- Si A y B son matrices de $n \times n$, entonces det $AB =$ det Adet B.

- Si una matriz cuadrada A tiene factorización LU tal que $A = LU$, donde L tiene unos en la diagonal y $U = (u_{ij})$, entonces

$$\text{det } A = \text{det } U = \prod_{i=1}^{n} u_{ii}.$$

- Si P es una matriz de permutación (es decir, el producto de matrices elementales de permutación), entonces det $P = \pm 1$.

- Si A tiene factorización $PA = LU$, entonces det $A = \pm$det U.

Unidad IV

Interpolación

Introducción

- En muchas aplicaciones se requiere estimar los valores intermedios de una función desconocida entre puntos conocidos.

- Este problema puede definirse como sigue: dada una serie de puntos en el plano cartesiano (x_1, y_1), (x_2, y_2), . . . , (x_n, y_n), encontrar una función contínua $y = f(x)$ que pase por todos ellos; es decir, que cumpla $y_k = f(x_k)$ para $k = 1, 2, \ldots, n$.

- En algunos casos puede requerirse que la función $f(x)$ cumpla con ciertas restricciones, por ejemplo, que no sea demasiado costosa computacionalmente, o que su derivada sea contínua.

Interpolación lineal

- El caso mas sencillo es aquel en el que solamente tenemos dos puntos: (x_1, y_1) y (x_2, y_2), sin embargo, hay una infinidad de funciones que pasan por estos dos puntos.

- Si elegimos $f(x)$ como la función mas simple que pasa por los dos puntos, entonces $f(x)$ debe ser una recta, la cual está dada por:

$$f(x) = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1).$$

- La interpolación lineal es útil si la distancia entre x_1 y x_2 es relativamente pequeña, en cuyo caso la función real (desconocida) puede aproximarse con una recta en ese intervalo.

Interpolación cuadrática

- Una mejor estimación de la función desconocida puede obtenerse mediante un polinomio de segundo grado, en lugar de una recta.

- Para esto, se requieren tres puntos (x_1, y_1), (x_2, y_2), y (x_3, y_3). Podemos entonces escribir la función $f(x)$ como:

$$f(x) = b_0 + b_1(x - x_1) + b_2(x - x_1)(x - x_2).$$

- Evaluando en $x = x_1$ obtenemos que $b_0 = y_1$.

- Evaluando en $x = x_2$ llegamos a que

$$b_1 = \frac{y_2 - y_1}{x_2 - x_1}.$$

- Finalmente, sustituyendo b_0 y b_1 en la ecuación del polinomio, encontramos que

$$b_2 = \frac{\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_1}.$$

Polinomio de interpolación de Newton

- Lo anterior puede generalizarse para construír un polinomio $p(x)$ de grado n que pase por $n+1$ puntos (x_k, y_k), $k = 1, 2, \ldots, n+1$.

- Este polinomio tiene la forma:

$$p(x) = b_0 + b_1(x - x_1) + b_2(x - x_1)(x - x_2) + \ldots + b_n(x - x_1)(x - x_2)\ldots(x - x_n),$$

  o bien,

$$p(x) = \sum_{j=0}^{n} b_j \prod_{i=1}^{j} (x - x_i).$$

- Los coeficientes b_j se encuentran utilizando el método de *diferencias divididas*.

Diferencias divididas

- Dados n puntos $(x_k, y_k = f(x_k))$, $k = 1, \ldots, n$, las diferencias divididas se definen como

$$\begin{aligned} f[x_q] &= f(x_q) = y_q, \\ f[x_q, \ldots, x_{q+r}] &= \frac{f[x_{q+1},\ldots,x_{q+r}] - f[x_q,\ldots,x_{q+r-1}]}{x_{q+r} - x_q}. \end{aligned}$$

- Los coeficientes b_j del polinomio de Newton están dados por

$$b_j = f[x_1, \ldots, x_{j+1}].$$

Ejemplo

Consideremos nuevamente la interpolación cuadrática de tres puntos (x_1, y_1), (x_2, y_2), (x_3, y_3). El polinomio de interpolación de Newton queda entonces como

$$p(x) = b_0 + b_1(x - x_1) + b_2(x - x_1)(x - x_2),$$

donde

$$\begin{aligned}
b_0 &= f[x_1] = y_1, \\
b_1 &= f[x_1, x_2] = \tfrac{f[x_2]-f[x_1]}{x_2-x_1} = \tfrac{y_2-y_1}{x_2-x_1}, \\
b_2 &= f[x_1, x_2, x_3] \\
&= \frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1} \\
&= \frac{\frac{f[x_3]-f[x_2]}{x_3-x_2} - \frac{f[x_2]-f[x_1]}{x_2-x_1}}{x_3-x_1} = \frac{\frac{y_3-y_2}{x_3-x_2} - \frac{y_2-y_1}{x_2-x_1}}{x_3-x_1}.
\end{aligned}$$

Implementación

- Diferencias divididas:

```
function d = difdiv(x, y, q, r)
  if r <= 0
    d = y(q);
  else
    d = (difdiv(x, y, q+1, r-1) - difdiv(x, y, q, r-1)) / (x(q+r) - x(q));
  end
end
```

- Polinomio de Newton:

```
function p = newtonpoly(x, y)
  p = 0; q = 1;
  n = length(x) - 1;
  for j = 0:n
    p = sumapoli(p, difdiv(x, y, 1, j) * q);
    q = multpol(q, [1, -x(j+1)]);
  end
end
```

Polinomio de interpolación de Lagrange

- Un método alternativo para obtener el polinomio de grado $n-1$ que pasa por n puntos dados está dado por la formulación de Lagrange:

  $$p(x) = \sum_{j=1}^{n} L_j(x) y_j,$$

  donde

  $$L_j(x) = \prod_{i=1, i\neq j}^{n} \frac{x - x_i}{x_j - x_i}.$$

- Notar que

  $$L_j(x_k) = \begin{cases} 1 & \text{si } k = j, \\ 0 & \text{si } k \neq j, \end{cases}$$

  por lo tanto,

  $$p(x_k) = y_k$$

  para $k = 1, \ldots, n+1$.

Implementación

- Polinomios base de Lagrange:

```
function L = lpoli(x, j)
  L = 1;
  for i = 1:length(x)
    if i != j
      L = multpol(L, [1, -x(i)] / (x(j) - x(i)));
    end
  end
end
```

- Polinomio de interpolación de Lagrange:

```
function p = lagrange(x, y)
  p = 0;
  for j = 1:length(x)
    p = sumapoli(p, lpoli(x, j) * y(j));
  end
end
```

Interpolación con splines: Motivación

- Los polinomios de interpolación de Newton y de Lagrange tienen algunas desventajas:

  1. Conforme aumenta el número de puntos, aumenta también el grado del polinomio.

  2. Un polinomio de alto grado tiende a mostrar oscilaciones alrededor de los puntos donde hay un cambio brusco.

- Una solución consiste en construír la función de interpolación por segmentos, cada uno de los cuales es un polinomio de grado pequeño que une dos puntos consecutivos.

Interpolación con splines

- Un *spline* de grado m es una función $S(x)$ definida por segmentos, donde cada segmento es un polinomio de grado $\leq m$, que cumple con la condicion de que todas las derivadas $S^{(k)}(x)$ con 0 $\leq< k < m$ son contínuas.

- Dados los puntos $\{(x_1, y_1), (x_2, y_2), \ldots, (x_{n+1}, y_{n+1})\}$, un spline se define como

$$S(x) = \begin{cases} p_1(x), & x_1 \leq x < x_2 \\ p_2(x), & x_2 \leq x < x_3 \\ \vdots & \vdots \\ p_n(x), & x_n \leq x < x_{n+1} \end{cases}$$

  con las siguientes condiciones:

  1. $p_j(x_j) = y_j$,
  2. $p_j(x_{j+1}) = y_{j+1}$,
  3. $p^{(k)}_{j+1}(x_j) = p^{(k)}_j(x_j), \quad 1 \leq k < m, \ 1 \leq j \leq n.$

Ejemplo: Splines lineales

- El caso mas sencillo es el spline de grado 1, donde los polinomios que forman el spline son rectas. Por lo tanto,

$$S(x) = \begin{cases} a_1(x - x_1) + b_1, & x_1 \leq x < x_2 \\ a_2(x - x_2) + b_2, & x_2 \leq x < x_3 \\ \quad \vdots & \quad \vdots \\ a_n(x - x_n) + b_n, & x_n \leq x < x_{n+1} \end{cases}$$

con

$$a_j = \frac{y_{j+1} - y_j}{x_{j+1} - x_j}, \quad b_j = y_j.$$

Splines cuadráticos

- Los splines lineales son muy sencillos de calcular, pero no son derivables en los nodos, lo cual puede ser un problema en ciertas aplicaciones. Esto se soluciona incrementando el grado del spline.

- Un spline cuadrático (de grado 2) tiene la siguiente forma:

$$S(x) = \begin{cases} a_1x^2 + b_1x + c_1, & x_1 \leq x < x_2 \\ \vdots & \vdots \\ a_nx^2 + b_nx + c_n, & x_n \leq x < x_{n+1} \end{cases}$$

  y su derivada es

$$S'(x) = \begin{cases} 2a_1x + b_1, & x_1 \leq x < x_2 \\ \vdots & \vdots \\ 2a_nx + b_n, & x_n \leq x < x_{n+1} \end{cases}$$

Coeficientes del spline cuadrático

- Dados $n+1$ puntos, se tienen n segmentos, y $3n$ coeficientes en el spline cuadrático. Para calcular estos coeficientes, necesitamos entonces $3n$ condiciones (ecuaciones).

- Las condiciones son:

1. $p_j(x_j) = y_j, \quad j = 1, \ldots, n$:

$$a_j x_j^2 + b_j x_j + c_j = y_j, \quad j = 1, \ldots, n$$

2. $p_j(x_{j+1}) = y_{j+1}, \;\; j = 1, \ldots, n$:

$$a_j x_{j+1}^2 + b_j x_{j+1} + c_j = y_{j+1}, \;\; j = 1, \ldots, n$$

3. $p'_j(x_{j+1}) = p'_{j+1}(x_{j+1}), \quad j = 1, \ldots, n-1$:

$$2a_j x_{j+1} + b_j = 2a_{j+1} x_{j+1} + b_{j+1}, \quad j = 1, \ldots, n-1$$

4. Condición inicial: $p''_1(x_1) = 0$, para un spline natural:

$$a_1 = 0$$

Cálculo de los coeficientes

$$\begin{pmatrix}
x_1^2 & x_1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
0 & 0 & 0 & x_2^2 & x_2 & 1 & \dots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \dots & x_n^2 & x_n & 1 \\
x_2^2 & x_2 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
0 & 0 & 0 & x_3^2 & x_3 & 1 & \dots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \dots & x_{n+1}^2 & x_{n+1} & 1 \\
2x_2 & 1 & 0 & -2x_2 & -1 & 0 & \dots & 0 & 0 & 0 \\
0 & 0 & 0 & 2x_3 & 1 & 0 & \dots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \dots & -2x_{n+1} & -1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix} a_1 \\ b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ \vdots \\ a_n \\ b_n \\ c_n \end{pmatrix}
=
\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_2 \\ y_3 \\ \vdots \\ y_{n+1} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

Implementación de splines cuadráticas

```
function q = qsplinecoef(x, y)
  n = length(x) - 1;
  a = zeros(3 * n);
  b = zeros(3 * n, 1);
  c = 1;
  for j = 1:n
    a(j, c:(c+2)) = [x(j)^2, x(j), 1];         # condicion 1
    b(j) = y(j);

    a(j + n, c:(c+2)) = [x(j+1)^2, x(j+1), 1]; # condicion 2
    b(j + n) = y(j + 1);

    if j < n                                   # condicion 3
      a(j + 2 * n, c:(c+4)) = [2 * x(j+1), 1, 0, -2 * x(j+1), -1];
    end

    c = c + 3;
  end

  a(3 * n, 1) = 1;                             # condicion 4

  q = reshape(a \ b, 3, n)';
end
```

Implementación de splines cuadráticas

```
function yy = qsplineeval(q, x, xx)
  k = zeros(size(xx));

  for i = 1:(length(x) - 1)
    k = k + (xx >= x(i));
  end

  yy = q(k,1)' .* xx.^2 + q(k,2)' .* xx + q(k,3)';
end
```

Splines cúbicos

- Los splines cúbicos se forman con polinomios de la forma

$$p_k(x) = a_k x^3 + b_k x^2 + c_k x + d.$$

- Ahora se tienen $4n$ coeficientes, donde n es el número de segmentos. Por lo tanto, se requieren $4n$ ecuaciones para encontrarlos.

- Los coeficientes de los polinomios se encuentran a partir de las siguientes condiciones:

  1. $p_k(x_k) = y_k$, $k = 1, \ldots, n$
  2. $p_k(x_{k+1}) = y_{k+1}$, $k = 1, \ldots, n$
  3. $p'_k(x_{k+1}) = p'_{k+1}(x_{k+1})$, $k = 1, \ldots, n-1$
  4. $p''_k(x_{k+1}) = p''_{k+1}(x_{k+1})$, $k = 1, \ldots, n-1$.
  5. Dos condiciones iniciales: para un spline natural son

$$p''_1(x_1) = 0 \quad \text{y} \quad p''_n(x_{n+1}) = 0.$$

Coeficientes de los splines cúbicos

- Los coeficientes se encuentran resolviendo el siguiente sistema de ecuaciones:

1. **Condición 1:**
$$x_k^3 a_k + x_k^2 b_k + x_k c_k + d_k = y_k,\ k = 1, \ldots, n,$$
2. **Condición 2:**
$$x_{k+1}^3 a_k + x_{k+1}^2 b_k + x_{k+1} c_k + d_k = y_{k+1},\ k = 1, \ldots, n,$$
3. **Condición 3:**
$$3x_{k+1}^2 a_k + 2x_{k+1} b_k + c_k - 3x_{k+1}^2 a_{k+1} - 2x_{k+1} b_{k+1} - c_{k+1} = 0,\ k = 1, \ldots, n-1,$$
4. **Condición 4:**
$$6x_{k+1} a_k + 2b_k - 6x_{k+1} a_{k+1} - 2b_{k+1} = 0,\ k = 1, \ldots, n-1,$$
5. **Condición inicial 1:**
$$6x_1 a_1 + 2b_1 = 0,$$
6. **Condición inicial 2:**
$$6x_{n+1} a_n + 2b_n = 0.$$

Ejercicios:

1. Considere los siguientes puntos en el plano cartesiano:

x	-2	-0.5	0	1	2.5
y	1	2	-1.5	-1	4

   Interpole los puntos y encuentre el valor de la función de interpolación en $x = -1$ y $x = 2$, usando los siguientes métodos:

   (a) Mediante un polinomio de cuarto grado.

   (b) Mediante un spline lineal.

   (c) Mediante un spline cuadrático natural

2. Escriba una función de Octave que, dados los vectores x y y de igual longitud, calcule y devuelva una matriz Q con los coeficientes del spline cúbico natural que pasa por todos los puntos (x_i, y_i).

3. Escriba una función de Octave que tome como parámetros: la matriz de coeficientes Q de un spline cúbico, el vector de abcisas x que definen los segmentos del spline, y un escalar z. La función debe devolver el valor del spline en z.

Splines de base (B-splines)

- Los *splines de base* o B-splines se obtienen como la combinación lineal de funciones base $B_k(x)$ de soporte finito (es decir, $B_k(x) = 0$ fuera de un cierto intervalo finito $[a_k, b_k]$).

- Formalmente, dados los *nudos* $x_0, x_1, \ldots, x_n$, un B-spline es una función $S(x)$ definida como

$$S(x) = \sum_{k=0}^{n} p_k B_{k,d}(x),$$

donde p_k son los *puntos de control*, $x_0 \leq x \leq x_n$, $2 \leq d \leq n+1$, $d-1$ es el grado de la curva, y los splines base $B_{k,d}(x)$ están dados por las fórmulas recursivas de Cox-deBoor:

$$\begin{aligned} B_{k,1}(x) &= \begin{cases} 1 & \text{si } x_k \leq x \leq x_{k+1}, \\ 0 & \text{para otros valores de } x, \end{cases} \\ B_{k,d}(x) &= \frac{x - x_k}{x_{k+d-1} - x_k} B_{k,d-1}(x) + \frac{x_{k+d} - x}{x_{k+d} - x_{k+1}} B_{k+1,d-1}(x). \end{aligned}$$

B-Splines uniformes

- Cuando los nudos $x_0, \ldots, x_n$ están equiespaciados, se dice que el spline es *uniforme*.

- Considerar un spline uniforme. Si Δx es la distancia entre dos nudos consecutivos, entonces

$$x_k = x_0 + k\Delta x.$$

- En un B-spline uniforme, las funciones base son simplemente copias desplazadas una de otra, de manera que

$$B_{k,d}(x) = B_{k+1,d}(x + \Delta x) = B_{k+r,d}(x + r\Delta x).$$

B-splines cuadráticos uniformes

- Considerar un B-spline uniforme con $d = 3$, de manera que su grado es $d - 1 = 2$. Suponer también que los nudos están en $x_k = k$; es decir, $x_0 = 0$ y $\Delta x = 1$. Entonces, el B-spline está dado por

$$S(x) = \sum_{k=0}^{n} p_k B_{k,3}(x) = \sum_{k=0}^{n} p_k B_{0,3}(x - k).$$

- Es fácil ver que

$$B_{0,3}(x) = \begin{cases} \frac{1}{2}x^2 & \text{si } 0 \leq x < 1, \\ -x^2 + 3x - \frac{3}{2} & \text{si } 1 \leq x < 2, \\ \frac{1}{2}(3 - x)^2 & \text{si } 2 \leq x \leq 3, \\ 0 & \text{para otros valores de } x. \end{cases}$$

- Todo B-spline cuadrático uniforme se puede generalizar al caso anterior mediante la transformación

$$x_k \leftrightarrow \frac{x_k - x_0}{\Delta x}.$$

Implementación en Octave

```
function y = B03(x)
  y = zeros(size(x));
  xx = x .* x;
  y = y + ((x >= 0) .* (x < 1)) .* (0.5 .* xx);
  y = y + ((x >= 1) .* (x < 2)) .* (-xx + 3 * x - 1.5);
  y = y + ((x >= 2) .* (x <= 3)) .* (4.5 - 3 * x + 0.5 * xx);
end

function y = qbspline(p, x)
  y = zeros(size(x));
  for k = 1:length(p)
    y = y + p(k) * B03(x - k + 2.5);  #  x <- x + d/2
  end
end
```

Unidad V

Regresión Lineal por Mínimos Cuadrados

Introducción

- Supongamos que tenemos n muestras (x_k, y_k), $k = 1, \ldots, n$, de una función desconocida $f(x)$. Sin embargo, las muestras fueron registradas con ruido o errores, de manera que

$$y_k = f(x_k) + e_k,$$

donde $\{e_1, e_2, \ldots, e_n\}$ es una secuencia de números aleatorios.

- El problema reside en estimar o aproximar la función $f(x)$. Podemos hacer esto ajustando un polinomio o un spline a los datos, sin embargo, el ruido presente en los datos ocasionará que la función de interpolación oscile mucho entre los puntos.

Regresión lineal

- Otra posibilidad es suponer que $f(x)$ es una función muy sencilla, por ejemplo, una línea recta, de manera que

$$y_k = mx_k + b + e_k,$$

donde m y b son, respectivamente, la pendiente y el cruce por el eje Y de la recta.

- Los *errores* o *residuos* están dados por

$$e_k = y_k - mx_k - b.$$

- Para encontrar la recta que mejor se ajusta a los datos es necesario encontrar los valores de m y b que minimizan una función de los errores.

Funciones de error

- Una función de error intenta medir qué tan bien se ajusta la función propuesta a los datos. Algunos ejemplos de funciones de error son los siguientes:

Función de error	Desventaja
$E = \sum_{i=1}^{n} e_i$	Los e_i pueden ser positivos o negativos, por lo que puede haber "cancelaciones" que afecten a la estimación del error total.
$E = \sum_{i=1}^{n} \lvert e_i \rvert$	No es derivable en todos los puntos.
$E = \sum_{i=1}^{n} e_i^2$	

Ajuste por mínimos cuadrados

- Los coeficientes de la recta se obtienen minimizando la función de error con respecto a m y b. Para ello, es necesario calcular las derivadas parciales del error con respecto a los coeficientes:

$$\begin{aligned}\frac{\partial E}{\partial m} &= -2\sum(y_i - mx_i - b)x_i,\\ \frac{\partial E}{\partial b} &= -2\sum(y_i - mx_i - b).\end{aligned}$$

- Y luego igualar las derivadas a cero para formar un sistema de dos ecuaciones en términos de m y b:

$$\begin{aligned}\left(\sum x_i\right)m + nb &= \sum y_i,\\ \left(\sum x_i^2\right)m + \left(\sum x_i\right)b &= \sum x_i y_i.\end{aligned}$$

Ajuste por mínimos cuadrados

- Resolviendo el sistema, llegamos a que

$$
\begin{aligned}
m &= \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - \left(\sum x_i\right)^2}, \\
b &= \bar{y} - m\bar{x},
\end{aligned}
$$

donde $\bar{x}$ y $\bar{y}$ son, respectivamente, los promedios de las x_i y y_i, es decir

$$
\bar{x} = \frac{1}{n} \sum x_i, \qquad \bar{y} = \frac{1}{n} \sum y_i.
$$

- Si $x = (x_1, x_2, \ldots, x_n)$ y $y = (y_1, y_2, \ldots, y_n)$, entonces podemos reescribir las fórmulas anteriores como

$$
\begin{aligned}
m &= \frac{n(x \cdot y) - n^2 \bar{x}\bar{y}}{n(x \cdot x) - n^2 \bar{x}^2} = \frac{x \cdot y - n\bar{x}\bar{y}}{x \cdot x - n\bar{x}^2}, \\
b &= \bar{y} - m\bar{x}.
\end{aligned}
$$

Implementación en Octave

```
function [m,b] = reglin(x, y, p = false)
  n = length(x);
  px = mean(x);
  py = mean(y);
  m = (dot(x,y) - n * px * py) / (dot(x,x) - n * px * px);
  b = py - m * px;

  if p
    plot(x, y, "*k");
    a = axis();
    hold on;
    plot([a(1), a(2)], [m * a(1) + b, m * a(2) + b], "b");
    hold off;
  end
end
```

Ejemplos

1. Considerar los siguientes puntos:

x	1	1.5	2	2.5	3	3.5	4
y	1.09	5.08	3.54	6.53	6.60	9.12	10.95

   Ajustar un modelo lineal de la forma $y = mx + b$ a los puntos anteriores. Graficar los puntos y la línea ajustada.

2. Ajustar un modelo lineal a los siguientes puntos:

x	1	2	3	4	5	6	7	8	9	10
y	-0.3	2.4	5.6	7.0	13	17	24	32	41	52

3. Graficar los residuos para los dos ejercicios anteriores. Indicar si se observa alguna tendencia obvia en los residuos.

Modelos no lineales

- Considerar los siguientes datos

x	9.2	5.4	3.7	1.7	9.4	6.5	3.5	4.7	2.5	10.6
y	30.1	9.1	7.2	4.7	33.8	13.5	7.4	7.5	5.7	47.3

- Al graficar los datos, se observa claramente que no siguen un modelo lineal. De hecho, los datos provienen de un modelo exponencial de la forma
$$y = \alpha e^{\beta x}.$$

- Aplicando el logaritmo (natural) a ambos lados de la ecuación anterior, llegamos a que
$$\log y = \beta x + \log \alpha.$$

- Entonces, podemos aplicar una transformación a los datos, que en ese caso consiste en reemplazar y por $\log y$, de manera que los datos transformados sigan una tendencia lineal, y podamos encontrar los coeficientes $\log \alpha$ y β mediante regresión lineal.

Modelos no lineales

- Algunos ejemplos de modelos no lineales que pueden resolverse mediante regresión lineal son:

Modelo no lineal	**Linealización**
$y = \alpha e^{\beta x}$	$\log y = \beta x + \log \alpha$
$y = \alpha x^{\beta}$	$\log y = \beta \log x + \log \alpha$
$y = a\frac{x}{x+\beta}$	$\frac{1}{y} = \frac{1}{\alpha} + \frac{\beta}{\alpha}\left(\frac{1}{x}\right)$

Residuos

- Es posible evaluar qué tan bien se ajusta un modelo a un conjunto de datos simplemente observando el comportamiento de los residuos e_i.

- Suponemos que e_i es una secuencia de números aleatorios (formalmente, se considera que provienen de una distribución normal). Por lo tanto, si los residuos muestran un comportamiento aleatorio, significa que el modelo se ajusta bien a los datos.

- Si, por el contrario, los residuos muestran alguna tendencia, entonces el modelo no encaja bien con los datos y debe ser cambiado.

Ejercicios

1. Considere los datos de la siguiente tabla

x	3.0	2.7	2.0	5.2	3.8	3.5	1.0	3.2	1.7	1.3
y	0.07	0.24	0.33	0.10	0.01	0.10	1.92	0.09	0.47	0.96

   Estos datos provienen de un modelo de la forma

$$y = \frac{1}{\alpha x^{\beta}}.$$

   (a) Aplique una transformación al modelo para linealizarlo.

   (b) Encuentre los parámetros α y β del modelo mediante regresión lineal.

   (c) Grafique los puntos y la curva correspondiente al modelo encontrado.

2. Encuentre el modelo que mejor se ajusta a los siguientes datos:

x	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
y	1.1	1.5	2.5	3.7	4.7	6.0	7.6	9.9	12.8	17.3

   Utilice como criterios de comparación: (1) el comportamiento de los residuos e_i, y (2) la función de error total $E = \sum_i e_i^2$.

Unidad VI

Integración y diferenciación numéricas

Motivación

- En muchas aplicaciones de software, es necesario calcular el valor de la derivada, o de la integral definida de una función $f(x)$ (contínua).

- Si $f(x)$ es conocida y tiene una forma sencilla, entonces uno puede calcular de manera analítica la derivada o la integral, e implementar directamente las fórmulas resultantes.

- Sin embargo, si $f(x)$ es complicada, o sólo se conocerá en tiempo de ejecución, entonces es necesario recurrir a técnicas para estimar el valor de la derivada o la integral en los puntos o intervalos dados.

Integración numérica por rectángulos

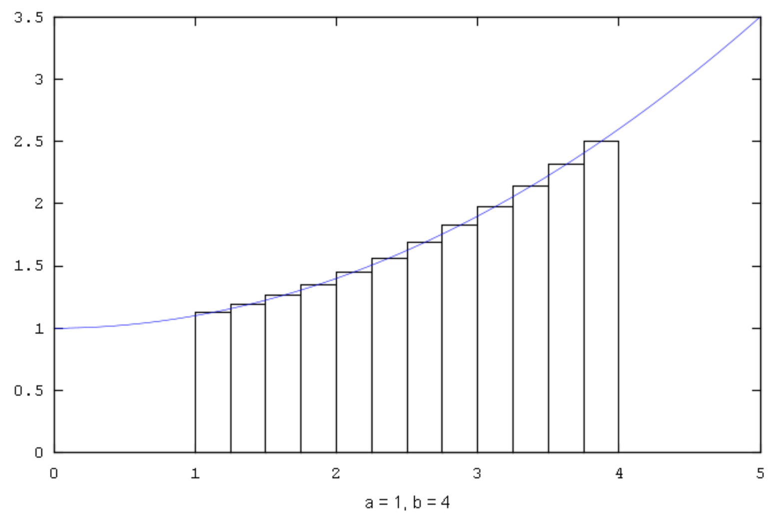
- La integral definida de una función contínua $f(x)$ en el intervalo $[a, b]$ es el área bajo la curva de esa función en ese intervalo.

- Una manera de estimar el área bajo la curva, consiste en dividirla en rectángulos con un ancho constante. La altura de cada rectángulo está dada por el valor de la función en el punto medio de la base del rectángulo.

- Entonces, la integral puede aproximarse por la suma de las áreas de todos los rectángulos:

$$\int_a^b f(x)dx \approx h \sum_{k=1}^{n} f\left(a + kh - \frac{h}{2}\right),$$

donde n es el número de rectángulos en los que se subdivide el área, y $h = (b - a)/n$ es la base de cada rectángulo.



Implementación en Octave

- Para implementar la integración numérica en Octave, recurrimos nuevamente a la función `feval()`, debido a que no conocemos de antemano la función $f(x)$.

- La integración por rectángulos es muy fácil de implementar en Octave, aprovechando las operaciones vectoriales:

```
function s = intrect(f, a, b, n = 100)
  h = (b - a) / n;
  s = h * sum(feval(f, a + h * (1:n) - h / 2));
end
```

- Ejemplo: si $f(x) = \frac{1}{10}x^2 + 1$, entonces:

```
> function y = f(x); y = 0.1 * x.^2 + 1; end
> intrect("f", 1, 4);
ans =  5.1000
```

Regla del Trapecio

- Supongamos que se divide el intervalo $[a, b]$ en n segmentos de longitud $h = (b - a)/n$. Es posible aproximar la función $f(x)$ en cada segmento con un polinomio de grado 1 (una recta), de manera que el área bajo la curva se divide en trapecios, en lugar de rectángulos.

- Es fácil ver que el área de cada trapecio está dada por:

$$A_k = \frac{h}{2}\left(f(x_k) + f(x_k + h)\right),$$

donde x_k es el extremo izquierdo del k-ésimo segmento; es decir, $x_k = a + (k - 1)h$.

- De esta manera, la integral puede estimarse como

$$\int_a^b f(x)dx \approx \frac{h}{2}\sum_{k=1}^{n}\left(f(a + (k-1)h) + f(a + kh)\right),$$

o bien,

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[f(a) + f(b) + 2\sum_{k=1}^{n-1} f(a + kh)\right].$$

Regla de Simpson

- Una mejor aproximación a la integral puede obtenerse ajustando un polinomio cuadrático por cada tres puntos (es decir, para cada pareja de segmentos adyacentes).

- Consideremos dos segmentos adyacentes centrados en $x = 0$. Entonces, lo que se desea es ajustar un polinomio cuadrático de la forma

$$p(x) = ax^2 + bx + c$$

a los siguientes puntos:

$$(-h, f(-h)), \ (0, f(0)), \ (h, f(h)).$$

- El área bajo $p(x)$ está dada por la integral

$$\int_{-h}^{h} (ax^2 + bx + c)dx = \frac{2h^3 a}{3} + 2hc.$$

Regla de Simpson

- Los coeficientes a y c del polinomio se obtienen fácilmente resolviendo el sistema de ecuaciones que se obtiene al sustituír los puntos en la ecuación del polinomio, quedando como

$$\begin{aligned} a &= \frac{f(-h) - 2f(0) + f(h)}{2h^2}, \\ c &= f(0). \end{aligned}$$

- Sustituyendo a y c en la integral anterior, llegamos a que

$$\int_{-h}^{h} p(x)dx = \frac{h}{3}\left[f(-h) + 4f(0) + f(h)\right].$$

Regla de Simpson

- Consideremos ahora la función $f(x)$ en el intervalo $[a,b]$. Podemos dividir el intervalo en un número par de segmentos n de longitud $h=(b-a)/n$.

- El área bajo la cuadrática correspondiente a los primeros dos segmentos está dada por

$$\frac{h}{3}\left[f(a)+4f(a+h)+f(a+2h)\right].$$

- Para los siguientes dos segmentos, el área es

$$\frac{h}{3}\left[f(a+2h)+4f(a+3h)+f(a+4h)\right].$$

- Entonces, es fácil ver que el área total se puede aproximar por

$$\int_a^b f(x)dx \approx \frac{h}{3}\sum_{i=0}^{n/2-1}\left[f(a+2ih)+4f\left(a+(2i+1)h\right)+f\left(a+(2i+2)h\right)\right].$$

Implementación en Octave

```
function s = intsimp(f, a, b, n = 100)
  h = (b - a) / n;
  i2 = 2 * (0:(n/2 - 1));
  x = a + i2 * h;
  s = sum(feval(f, x) + 4 * feval(f, x + h) + feval(f, x + 2 * h));
  s = s * h / 3;
end
```

Ejercicios

1. Escriba una función de Octave que utilice la regla del trapecio para estimar la integral de una función $f(x)$ dada, en el intervalo $[a, b]$ dividido en n segmentos.

2. Encuentre la integral de la función $f(x) = 5xe^{x^2/4} + 8$ en el intervalo $[-1, 2]$ utilizando los siguientes métodos:

   a) De manera analítica (papel y calculadora).

   b) En Octave mediante la función `intrect()` con $n = 100$.

   c) En Octave mediante la función `intsimp()` con $n = 100$.

3. Utilice Octave para calcule la siguiente integral usando los tres métodos (rectángulos, trapecio, y regla de Simpson):

$$\int_0^{2\pi} \frac{x^{\sin x}}{x+1} dx.$$

   El valor aproximado de la integral es 0.18974. Para cada uno de los tres métodos, indique aproximadamente cuántos segmentos se requieren para obtener la aproximación anterior.

Diferenciación numérica

- **Método ingenuo:** Recordemos la definición de la derivada de una función contínua $f(x)$:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

  Esta definición sugiere una manera de aproximar el valor de la derivada numéricamente: para un h positivo suficientemente pequeño, tenemos que

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

- Notar que la fórmula anterior da como resultado la pendiente de la recta que pasa por los puntos

$$(x, f(x)) \text{ y } (x+h, f(x+h)).$$

Diferencias hacia adelante y hacia atrás

- La fórmula anterior se conoce como la *primera diferencia hacia adelante*, ya que se toma el valor de la función en x y en otro punto adelante de x (es decir, $x + h$).

- También es posible estimar la derivada utilizando la *primera diferencia hacia atrás*:
$$f'(x) \approx \frac{f(x) - f(x-h)}{h}.$$

- Tomando el promedio de las diferencias hacia adelante y hacia atrás, podemos estimar la derivada por medio de la *diferencia centrada*:
$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}.$$

- Las diferencias centradas proporcionan una mejor aproximación que las diferencias hacia adelante o hacia atrás.

Derivadas de orden superior

- Supongamos ahora que estamos interesados en estimar la segunda derivada de $f(x)$. Podemos estimarla mediante diferencias hacia adelante de la siguiente manera:

$$f''(x) \approx \frac{f'(x+h) - f(x)}{h},$$

donde $f'(x+h)$ y $f'(x)$ pueden también estimarse mediante diferencias hacia adelante, con lo que se obtiene

$$f''(x) \approx \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2}.$$

- La misma idea puede aplicarse para obtener derivadas de orden superior.

Aproximación por medio de series de Taylor

- Otra manera de obtener la fórmula de diferencias hacia adelante es por medio de la serie de Taylor:

$$f(x+h) = f(x) + \frac{f'(x)}{1!}h + \frac{f''(x)}{2!}h^2 + \ldots.$$

- Despejando el término que corresponde a la primera derivada, se obtiene lo siguiente:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h),$$

donde $O(h^2)$ representa un término de error de orden h. Esto significa que la magnitud del error está acotada por un múltiplo de h.

- También es posible obtener las diferencias hacia atrás desarrollando la serie de Taylor alrededor de $x - h$.

Diferenciación numérica con alta precisión

- Podemos obtener una mejor aproximación de la primera derivada al agregar más términos de la serie de Taylor, por ejemplo:

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(x)}{2}h + O(h^2),$$

donde $f''(x)$ puede estimarse como se vió anteriormente, de manera que obtenemos

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f(x+2h) - 2f(x+h) + f(x)}{2h} + O(h^2).$$

- La fórmula anterior puede reducirse a

$$f'(x) = \frac{-f(x+2h) + 4f(x+h) - 3f(x)}{2h} + O(h^2).$$

Notar que ahora el error es de orden $O(h^2)$, lo cual representa una mejora.

- Utilizando la misma idea, uno puede mejorar aún mas la precisión considerando más términos de la serie de Taylor.

Ejercicios

1. Desarrolle las siguientes fórmulas de manera analítica:

   (a) Aproximación de la segunda derivada $f''(x)$ de una función $f(x)$ utilizando diferencias hacia atrás.

   (b) Aproximación de la primera derivada con error de orden $O(h^2)$ utilizando diferencias hacia atrás.