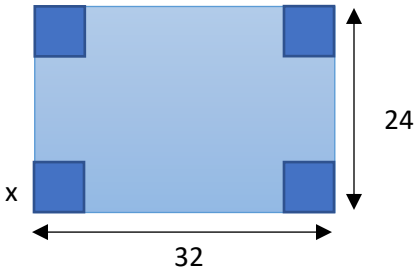
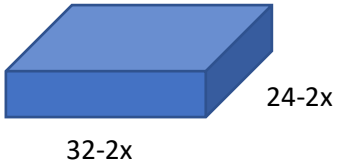
Juan León y María Garcés

Análisis numérico. Taller 1

Punto 2





El volumen de un cubo es base x altura x arista, por ende:

$$V_x = (32 - 2\text{x})(24 - 2\text{x})\text{x}$$

$$V_x = 4x^3 - 112x^2 + 768\text{x}$$

Para maximizar el volumen de la caja debemos encontrar los puntos donde la derivada vale 0. Lo que implica que:

$$V_x{}' = 12x^2 - 224\text{x} + 768$$

Con el fin de encontrar la raíz utilizamos el método de Newton.

a. ¿Cuál etapa del proceso de resolución de un problema numérico requiere más atención?
Planteamiento del problema porque es el que requiere más pericia matemática ya que es necesario establecer las relaciones entre las variables del problema y delimitar las restricciones necesarias para su ejecución. Cualquier error en esta etapa resultará en una pérdida de precisión y exactitud en la obtención del resultado.

b. ¿Qué conocimientos son necesarios para formular un modelo matemático?
- Lógica matemática pues es necesaria para la interpretación del problema a través de relaciones matemáticas.
- Capacidad de análisis requerida para la abstracción del problema.

c. En el ejemplo de la caja ¿Cuál seria la desventaja de intentar obtener experimentalmente la solución mediante prueba y error en lugar de analizar el modelo matemático?
Sería mas demorado el procedimiento ya que tocaría hacer muchas pruebas para encontrar la solución.

d. ¿Qué es más crítico: el error de truncamiento o el error de redondeo?

Depende del problema, si este requiere de información muy exacta el error de truncamiento resultaría más critico pues se perdería información que podría ser importante para el desarrollo del problema. Por otro lado, si el problema requiere de una gran cantidad de cálculos o procedimientos el error de redondeo seria mas critico pues se propagaría afectando el resultado.

e. **¿Cuál es la ventaja de instrumentar computacionalmente un método numérico?**
Aumenta la eficiencia porque la capacidad de procesamiento del computador es muy grande por lo tanto permite acelerar el proceso gastando menos recursos.

f. **¿Por qué es importante validar los resultados obtenidos?**
Para determinar la exactitud de la solución que se implemento a partir del problema.

Punto 12

Librería en R

La librería donde se encuentran los métodos de bisect y newtonRaphson es pracma.

Si la función tiene más de una raíz solo muestra la primera que encuentra.

- **bisect <- function(f, a, b, maxiter = 100, tol = NA)**

Funciona en cualquier caso si la función tiene signos opuestos en los puntos finales del intervalo.

Bisect se detiene cuando se alcanza la precisión del punto flotante, por lo que no es necesario adjuntar una tolerancia.

Esta versión está recortada para exactitud, no velocidad. Se tiene especial cuidado cuando 0.0 es una raíz de la función. El argumento 'tol' está en desuso.

Parámetros

f: Función o su nombre como una cadena.

a, b: float

Puntos finales del intervalo.

maxiter: int

Número máximo de iteraciones; predeterminado 100.

tol: float

Tolerancia absoluta.

Retorna

Una lista con los componentes raíz, f.raiz, el valor de la función en la raíz encontrada, iter, el número de iteraciones realizadas, y estim.prec de precisión estimada.

- **newtonRaphson <- function(fun, x0, dfun = NULL, . . . , maxiter = 100, tol = .Machine$double.eps^0.5)**

Conocido algoritmo de búsqueda de raíz para funciones reales, univariadas y continuas.

Parámetros

fun: función o su nombre como una cadena.

x0: float

Valor inicial para newtonRaphson ().

dfun: Una función para calcular la derivada de f. Si es NULL, una derivada numérica será calculada.

maxiter: int

Número máximo de iteraciones; predeterminado 100.

tol: float

Tolerancia absoluta.

...: argumentos adicionales que se pasarán a f.

Retorna

Una lista con los componentes raíz, f.raiz, el valor de la función en la raíz encontrada, iter, el número de iteraciones realizadas, y estim.prec de precisión estimada.

El siguiente método se encuentra en la librería FixedPoint

- **FixedPoint(Function, Inputs, Outputs = c(), Method = c("Anderson", "Simple", "Aitken", "Newton", "MPE", "RRE", "VEA", "SEA"), . . . , MaxIter = 1000)**

Párametros

Function:
Esta es la función para la cual se busca un punto fijo. Esta función debe tomar y devolver un vector de la misma dimensión.

Inputs:
Esto puede ser un vector de valores que es una suposición inicial para un punto fijo o

puede ser una matriz N x A de entradas previas para las cuales las salidas correspondientes están disponibles.

Outputs:

(Opcional) Esta es una matriz de los valores de la función para cada columna de la entrada.

Method:

Este es el método de punto fijo que se utilizará. Puede ser "Anderson", "Simple", "Aitken", "Newton", "MPE", "RRE", "VEA" o "SEA".

MaxIter:
Este es el número máximo de iteraciones que se llevarán a cabo.

...: argumentos adicionales.

Retorna

Una lista que contiene el FixedPoint, las Entradas y Salidas correspondientes, y los valores de convergencia (que se computan bajo "ConvergenceMetric"). La lista también incluirá una declaración "Finalizar" describiendo por qué ha terminado.

Librería en Python

La librería donde se encuentran los métodos es SciPy.optimize

- **Def newton (func, x0, fprime=None, args=(), tol=1.48e8, maxiter=50, fprime2=None):**

Encuentra un cero de la función `func` dado un punto de inicio cercano` x0`.

El método Newton-Raphson se usa si la derivada `fprime` de` func` se proporciona, de lo contrario, se utiliza el método secante. Si el segundo orden se proporciona el derivado `fprime2` de` func`, el método parabólico de Halley es usado.

Parámetros:

func: función

La función cuyo cero es deseado. Debe ser una función de variable única de la forma f (x, a, b, c ...), donde a, b, c ... son extra argumentos que se pueden pasar en el parámetro `args`.

x0: float

Una estimación inicial del cero que debería estar cerca del cero real.

fprime: función, opcional

La derivada de la función cuando está disponible y es conveniente. Si es NULL (predeterminado), luego se usa el método secante.

args: tupla, opcional

Argumentos adicionales que se utilizarán en la llamada a la función.

tol: float, opcional

El error permitido del valor cero.

maxiter: int, opcional

Número máximo de iteraciones

fprime2: función, opcional

La derivada de segundo orden de la función cuando está disponible y conveniente.

Retorna:

Zero: float

Ubicación estimada donde la función es cero.

- **Def bisect (f, a, b, args=(), xtol=_xtol, rtol=_rtol, maxiter=_iter, full_output=False, disp=True):**

Encuentra la raíz de una función dentro de un intervalo. La rutina de bisección básica para encontrar un cero de la función `f` entre los argumentos `a` y` b`. `f (a)` y `f (b)` no pueden tener los mismos signos.

Parámetros

f: función

La función Python devuelve un número. `f` debe ser continuo, y f (a) y f (b) deben tener signos opuestos.

a: número

Un extremo del intervalo de [a, b].

b: número

El otro extremo del intervalo de [a, b].

xtol: número, opcional

La rutina converge cuando se sabe que una raíz se encuentra dentro de `xtol` de valor de retorno. Debe ser> = 0. La rutina lo modifica para tener en cuenta la precisión relativa de dobles.

rtol: número, opcional

La rutina converge cuando se sabe que una raíz está dentro de `rtol` por el valor devuelto. Debe ser> = 0.

maxiter: número, opcional

Número maximo de iteraciones. Debe ser> = 0.

args: tupla, opcional

Contiene argumentos adicionales para la función `f`.

full_output: bool, opcional

Si `full_output` es False, se devuelve la raíz. Si `full_output` es True, el valor de retorno es `` (x, r)``, donde x es la raíz y r es un objeto `RootResults`.

disp: bool, opcional

Si es True, levanta RuntimeError si el algoritmo no convergió.

Retorna

x0: float

Cero de `f` entre` a` y `b`.

r: RootResults (presente si `` full_output = True``)

Objeto que contiene información sobre la convergencia. En particular, `` r.converged`` es True si la rutina convergió.

Punto 15

Punto b y d se encuentran en el codigo.

a. La ecuación que se obtiene al realizar la integral es la siguiente:

$$f(x) = -e^x + 5x - 1 = 0$$

c. Para obtener un g(x) se le adiciona una x ambos lados de la ecuación.

$$\boldsymbol{x = -e^x + 5x - 1 + x}$$

$$\boldsymbol{x = -e^x + 6x - 1}$$

Para encontrar el intervalo de convergencia se resolvió la siguiente desigualdad.

$$-1 < -e^x + 6x - 1 < 1$$

$$0 < -e^x + 6x < 2$$