Scilab: análisis numérico

APESAM [1]

28 de marzo de 2025

1. https://www.apesam.fr/

Índice

Hay aquí algunos ejemplos de cómo Scilab puede utilizarse para resolver algunos de los problemas clásicos de análisis numérico que se encuentran en los estudios de licenciatura en los primeros años de universidad.

Todos los programas Scilab se suministran en forma de un archivo fuente que simplemente necesita ser cargado en Scilab y ejecutado. No ofrecemos conferencias sobre estos algoritmos, pero proporcionamos un enlace que amplía su presentación.

Método de bisección

Haga clic en este enlace para ver una explicación del método

Listing 1 – Método de bisección

```
// Algoritmo de dicotomía
// Trata de encontrar la raíz de la función f en el intervalo [a,b].
// La precisión viene dada por el valor de epsilon

function c=Dicho(f,a,b,epsilon)
    c=(a+b)/2;
    while (b-a)/2 >= epsilon
        c=(a+b)/2;
        if f(a)*f(c)<0 then
            b=c;
        else
            a=c;
        end
    end
endfunction

// Ejemplo
function y=f(x)
    y=x^5+5*x-2;
endfunction

Dicho(f,0,1,1D-2)
```

Ejecutar el programa con Scilab

Eliminación de Gauss-Jordan

Haga clic en este enlace para ver una explicación del método

Resolver un sistema lineal $Ax = b$ con A como matriz cuadrada: la matriz debe ser invertible y los pivotes sucesivos no deben ser cero.

Listing 2 – Gauss 1

```
// Método del pivote de Gauss para resolver un sistema
// lineal Ax=b con A una matriz cuadrada invertible
// Los pivotes sucesivos no deben ser cero

function x=Gauss(A,b)
    n=size(b,"*"); x=b;
    // algoritmo de Gauss
    for k=1:n-1
        for l=k+1:n
            p=A(l,k)/A(k,k);
            for m=k:n
                A(l,m)=A(l,m)-A(k,m)*p;
            end
            x(l)=x(l)-x(k)*p;
        end
    end
// método de retroalimentación
    x(n)=x(n)/A(n,n);
    for i=n-1:-1:1
        s=0;
        for j=i+1:n
            s=s+A(i,j)*x(j);
        end
        x(i)=(x(i)-s)/A(i,i);
    end
endfunction

// Ejemplo
// solution : [-1;1;0]
A=[1 2 3;
   4 5 6;
   7 8 10];
b=[1;1;1];

Gauss(A,b)
```

Ejecutar el programa con Scilab

Mismo método pero con búsqueda de pivotes parciales: la matriz cuadrada A debe ser invertible.

Listing 3 – Gauss 2

```
// método del pivote de Gauss con búsqueda de pivotes parciales
// para resolver un sistema lineal Ax=b con A como matriz
// cuadrada invertible. Se considera que un pivote es cero cuando
// u valor absoluto es menor que eps
function x=Gauss2(A,b,eps)
    n=size(b,"*"); x=b;
    for k=1:n-1
        // si el término diagonal es cercano a 0
        // busca un elemento distinto de cero en la columna
        if abs(A(k,k))<eps then
            kk=find(abs(A(k:n,k))>eps);
            if kk==[] then
                disp("Matriz no invertible");
                return;
            end
            // intercambia las líneas k y kk en A y b
            kk=kk(1);
            lignek=A(k,:); A(k,:)=A(kk,:); A(kk,:)=lignek;
            lignek=b(k); b(k)=b(kk); b(kk)=lignek;
        end
        // Algoritmo de Gauss
        for l=k+1:n
            p=A(l,k)/A(k,k);
            for m=k:n
                A(l,m)=A(l,m)-A(k,m)*p;
            end
            x(l)=x(l)-x(k)*p;
        end
    end
    // método de retroalimentación
    if abs(A(n,n))<eps then
        disp("Matriz no invertible");
        return;
    end
    x(n)=x(n)/A(n,n);
    for i=n-1:-1:1
        s=0;
        for j=i+1:n
            s=s+A(i,j)*x(j);
        end
        x(i)=(x(i)-s)/A(i,i);
    end
endfunction
Ejemplo
A=[0 2 3;
   4 5 6;
   7 8 10];
b=[1;1;1];
Gauss2(A,b,1D-10)
```

Ejecutar el programa con Scilab

Método de Newton

Haga clic en este enlace para ver una explicación del método
Parar después de n iteraciones.

Listing 4 – Newton 1

```
// Método de Newton para encontrar un cero de la función f
// cuya derivada es la función fprim
// u0 valor inicial aproximado de la solución
// se detiene después de n iteraciones

function u=Newton1(f,fprim,u0,n)
    u=u0;
    for i=1:n
        fp=fprim(u);
        if abs(fp)<=%eps then
            error("La derivada es nula")
        end
        u=u-f(u)/fp;
    end
endfunction

// Ejemplo
// función y=f(x)
function y=f(x)
    y=x^3-x-1;
endfunction

// función derivada de f: y=f'(x)
function y=fprim(x)
    y=3*x^2-1;
endfunction

// 5 iteraciones a partir de 1
Newton1(f,fprim,1,5)
```

Ejecutar el programa con Scilab

Detener la prueba en el valor de la función.

Listing 5 – Newton 2

```
// Método de Newton para encontrar un cero de la función f
// cuya derivada es la función fprim
// u0 valor inicial aproximado de la solución
// se detiene cuando f(u) es menor o igual que eps

function u=Newton2(f,fprim,u0,eps)
    u=u0;
    while abs(f(u))>eps then
        fp=fprim(u);
        if abs(fp)<=%eps then
            error("La derivada es nula")
        end
        u=u-f(u)/fp
    end
endfunction

// Ejemplo
// función y=f(x)
function y=f(x)
    y=x^3-x-1;
endfunction

// función derivada de f: y=f'(x)
function y=fprim(x)
    y=3*x^2-1;
endfunction

// Error de 10^(-12) a partir de 1
Newton2(f,fprim,1,1D-12)
```

Ejecutar el programa con Scilab

Integración numérica

Haga clic en este enlace para ver una explicación del método
Integración utilizando el método trapezoidal.

Listing 6 – Método trapezoidal

```
// Integración trapezoidal de la función y=f(x)
// entre a y b
// subdividir en potencias de 2
// parar después de n iteraciones
// res = los valores sucesivos de la integral a lo largo de
// las iteraciones

function res=trapeze(f,a,b,n)
    res=[(f(a)+f(b))/2];
    h=b-a;
    for k=0:n-2
        h=h/2;
        s=0;
        for i=1:2^k
            s=s+f(a+(2*i-1)*h);
        end
        res=[res,res(k+1)/2+h*s];
    end
endfunction

// Ejemplo
function y=f(x)
    y=1/x;
endfunction

trapeze(f,1,2,12)
```

Ejecutar el programa con Scilab

Integración utilizando el método de rectángulos..

Listing 7 – Método de rectángulos

```
// Integración de la función y=f(x) por el método de los rectángulos
// entre a y b
// subdividir en potencias de 2
// parar después de n iteraciones
// res = los valores sucesivos de la integral a lo largo de
// las iteraciones

function res=rectangle(f,a,b,n)
    res=[f(a)];
    h=b-a;
    for k=0:n-2
        h=h/2;
        s=0;
        for i=1:2^k
            s=s+f(a+(2*i-1)*h);
        end
        res=[res,res(k+1)/2+h*s];
    end
endfunction

// Ejemplo
function y=f(x)
    y=1/x;
endfunction

rectangle(f,1,2,12)
```

Ejecutar el programa con Scilab

Método de Simpson.

Listing 8 – Método de Simpson

```
// integración de Simpson de la función y=f(x) entre a y b
// parada después de n iteraciones

function res=Simpson(f,a,b,n)
    h=(b-a)/n;
    res=(f(a)+f(b))/2;
    k=1:n-1;
    res=res+sum(feval(a+k*h,f));
    k=0:n-1;
    res=res+2*sum(feval(a+(2*k+1)*h/2,f));
    res=res*h/3;
    res*h/3
endfunction

// Ejemplo
function y=f(x)
    y=1/x;
endfunction

Simpson(f,1,2,8)
```

Ejecutar el programa con Scilab

Método de Romberg.

Listing 9 – Método de Romberg

```
// integración de Romberg de la función y=f(x) entre a y b
// parada después de n iteraciones

function res=Romberg(f,a,b,n)
    h=b-a;
    T(1)=(f(a)+f(b))/2;
    for i=0:n-2
        h=h/2; k=1:2^i;
        T(i+2)=T(i+1)/2+h*sum(feval(a+(2*k-1)*h,f));
    end
    for i=1:n-1
        for k=1:n-i
            T(k)=(4^i*T(k+1)-T(k))/(4^i-1);
        end
    end
    res=T(1);
endfunction

// Ejemplo
function y=f(x)
    y=1/x;
endfunction

Romberg(f,1,2,10)
```

Ejecutar el programa con Scilab