Scilab: Numerical Analysis

APESAM [1]

April 2, 2025

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

Contents

Here are some examples of how to use Scilab software to solve some common numerical analysis problems that are encountered in the studies of bachelor's degree in the first years of university.

All Scilab programs are provided in the form of a source file that simply needs to be loaded into Scilab and run. We do not offer presentations on these algorithms but provide a link to a more detailed presentation.

Bisection method

Click on this link for an explanation of the method

Listing 1 – Bisection method

```
// Bisection method
// Search for the root of the function f in the interval [a,b]
// The precision is given by the value of 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

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

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

Run the program with Scilab

Gaussian elimination

Click on this link for an explanation of the method

Solving a linear system $Ax = b$ with A a square matrix: the matrix must be invertible and the successive pivots must not be zero.

Listing 2 – Gauss 1

```
// Gaussian elimination method for solving a
// linear system Ax=b with A invertible square matrix
// The successive pivots must not be zero

function x=Gauss(A,b)
    n=size(b,"*"); x=b;
    //  Gauss algorithm
    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
    // feedback method
    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

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

Gauss(A,b)
```

Run the program with Scilab

Same method but with search for partial pivots: the square matrix A must be invertible.

Listing 3 – Gauss 2

```
// Gaussian elimination method with partial pivots
//   to solve a linear system Ax=b with A invertible square matrix
// A pivot is considered zero when its absolute value is less than eps
function x=Gauss2(A,b,eps)
    n=size(b,"*"); x=b;
    for k=1:n-1
          // case where the diagonal term is close to 0
         // search for a non-zero element in the column
        if abs(A(k,k))<eps then
            kk=find(abs(A(k:n,k))>eps);
            if kk==[] then
                disp("Non-invertible matrix");
                return;
            end
            // exchange of lines k and kk in A and in 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
        // Gauss algorithm
        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
     // ascent method
    if abs(A(n,n))<eps then
        disp("Non-invertible matrix");
        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

// Example
A=[0 2 3;
   4 5 6;
   7 8 10];
b=[1;1;1];
Gauss2(A,b,1D-10)
```

Run the program with Scilab

Newton's method

Click on this link for an explanation of the method
Termination after n iterations.

Listing 4 – Newton's method 1

```
// Newton's method for finding a zero of the function f
// whose derivative is the function fprim
// u0 approximates initial value of the solution
// stop after n iterations

function u=Newton1(f,fprim,u0,n)
    u=u0;
    for i=1:n
        fp=fprim(u);
        if abs(fp)<=%eps then
            error("Derivative is equal to 0")
        end
        u=u-f(u)/fp;
    end
endfunction

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

// function derived from f: y=f'(x)
function y=fprim(x)
    y=3*x^2-1;
endfunction

// 5 iterations starting from 1
Newton1(f,fprim,1,5)
```

Run the program with Scilab

Termination test on the value of the function.

Listing 5 – Newton's method 2

```
// Newton's method to find a zero of the function f
//   whose derivative is the function fprim
// u0 approximates initial value of the solution
// stop when f(u) is less than or equal to 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("Derivative is equal to 0")
        end
        u=u-f(u)/fp
    end
endfunction

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

// function derived from f: y=f'(x)
function y=fprim(x)
    y=3*x^2-1;
endfunction

// Error of 10^(-12) starting from 1
Newton2(f,fprim,1,1D-12)
```

Run the program with Scilab

Numerical integration

Click on this link for an explanation of the method
Integration by the trapezoidal method.

Listing 6 – Trapezoidal method

```
// Integration by the trapezoidal method of the function y=f(x)
//   between a and b
// subdivisions into powers of 2
// stop after n iterations
// res = the successive values of the integral during the iterations

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

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

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

Run the program with Scilab

Integration by the rectangular method.

Listing 7 – Rectangular method

```
// Integration by the method of rectangles of the function y=f(x)
//   between a and b
// subdivisions into powers of 2
// stop after n iterations
// res = the successive values of the integral during the iterations

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

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

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

Run the program with Scilab

Simpson's method.

Listing 8 – Simpson method

```
// Integration by Simpson's method of the function y=f(x) between a and b
// stop after n iterations

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

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

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

Run the program with Scilab

Romberg's method.

Listing 9 – Romberg's method

```
// Integration by the Romberg method of the function y=f(x)
// between a and b
// stop after n iterations

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

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

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

Run the program with Scilab