

## 7. Programación

Matlab tiene capacidades muy poderosas de **programación estructurada** y **programación funcional**.

Posee comandos para realizar todas las estructuras de **secuencia** (instrucciones normales), de **selección** ( **if – else** ) y **repetitivas** (ciclos **while** y **for** ).

Recuerde que para probar estos programas basta con teclearlos como archivos **.m** , guardarlos en disco duro exactamente con el **mismo nombre** que tienen en la primera de sus líneas con negritas.

También recuerde que el directorio en donde los guarde deberá de estar necesariamente **dado de alta en el path** (rutas) de Matlab para que los pueda encontrar automáticamente (ver path en parte 1).

=====

```
function y=evalua(x)
y=exp(0.25*x)-cos(3*x)-1;
```

=====

```
function biseccion(funx)
clc;
xa=0;xp=0;ep=0;
fprintf('Método de bisección\n\n');
xa=input('Teclee el valor de xa: ');
xp=input('Teclee el valor de xp: ');
ep=input('Teclee el valor de Tolerancia: ');
```

```
clc;
fprintf('Método de bisección\n\n');
fprintf(' ite   xa       fa       xm       fm       ps\n');
fprintf('-----\n');
i=0;
fa=feval(funx,xa);
while (1)
    xm=(xa+xp)/2;
    fm=feval(funx,xm);
    i=i+1;
    fprintf('%5.0f',i);
    fprintf('%13.6f %13.6f %13.6f %13.6f',xa,fa,xm,fm);
    if (abs(fm)<=ep)
        fprintf('\n\nSe cumplió criterio vertical.\n');
        break;
    end
    if (abs(xm-xa)<=ep)
        fprintf('\n\nSe cumplió criterio horizontal.\n');
        break;
    end
    if ((fa*fm)>0)
        xa=xm;
        fa=fm;
        fprintf(' +\n');
    end
    if ((fa*fm)<0)
        xp=xm;
        fp=fm;
        fprintf(' -\n');
    end
    if ((fa*fm)==0)
        fprintf(' 0\n');
        break;
    end
end
fprintf('\nLa raíz buscada es: %13.6f\n\n',xm);
```

```
=====
function y=derivada(x)
y=0.25*exp(0.25*x)+3*sin(3*x);
=====
```

```
function newtonr(funx,derix)
clc;
xa=0;xp=0;ep=0;
fprintf('Método de Newton-Raphson\n\n');
xa=input('Teclee el valor de xa: ');
ep=input('Teclee el valor de Error Permissible (%): ');
clc;
fprintf('Método de Newton-Raphson\n\n');
fprintf(' ite    xa        fa        xp        fp\n');
fprintf('-----\n');
i=0;
while (1)
    fa=feval(funx,xa);
    da=feval(derix,xa);
    xp=xa-(fa/da);
    fp=feval(funx,xp);
    i=i+1;
    fprintf('%5.0f',i);
    fprintf('%13.6f %13.6f %13.6f %13.6f\n',xa,fa,xp,fp);
    if (abs((xp-xa)/xp)<=ep)
        fprintf('\n\nSe cumplió el Error Permissible.\n');
        break;
    end
    xa=xp;
end
fprintf('\nLa raíz buscada es: %13.6f\n\n',xp);
=====
```

```
function newtonr2(funx)
clc;
xa=0;xp=0;ep=0;dx=0;
fprintf('Método de Newton-Raphson2\n\n');
xa=input('Teclee el valor de xa: ');
ep=input('Teclee el valor de Error Permissible (%): ');
dx=input('Teclee valor para aproximación de la derivada: ');
clc;
fprintf('Método de Newton-Raphson2\n\n');
fprintf(' ite    xa        fa        xp        fp\n');
fprintf('-----\n');
i=0;
while (1)
    fa=feval(funx,xa);
    f1=feval(funx,xa+dx);
    f2=feval(funx,xa-dx);
    da=(f1-f2)/(2*dx);
    xp=xa-(fa/da);
    fp=feval(funx,xp);
    i=i+1;
    fprintf('%5.0f',i);
    fprintf('%13.6f %13.6f %13.6f %13.6f\n',xa,fa,xp,fp);
    if (abs((xp-xa)/xp)<=ep)
        fprintf('\n\nSe cumplió el Error Permissible.\n');
        break;
    end
    xa=xp;
end
fprintf('\nLa raíz buscada es: %13.6f\n\n',xp);
=====
```

```
function newtonb(funx)
clc;
xa=0;xp=0;ep=0;dx=0;
fprintf('Método de Newton-Bayle\n\n');
xa=input('Teclee el valor de xa: ');
ep=input('Teclee el valor de Error Permisible (%): ');
dx=input('Teclee valor para aproximación de la derivada: ');
clc;
fprintf('Método de Newton-Bayle\n\n');
fprintf(' ite    xa        fa        xp        fp\n');
fprintf('-----\n');
i=0;
while (1)
    fa=feval(funx,xa);
    f1=feval(funx,xa+dx);
    f2=feval(funx,xa-dx);
    da=(f1-f2)/(2*dx);
    sa=(f1+f2-2*fa)/(dx^2);
    xp=xa-(fa/(da-((sa*fa)/(2*da))));
    fp=feval(funx,xp);
    i=i+1;
    fprintf('%5.0f',i);
    fprintf('%13.6f %13.6f %13.6f %13.6f\n',xa,fa,xp,fp);
    if (abs((xp-xa)/xp)<=ep)
        fprintf('\n\nSe cumplió el Error Permisible.\n');
        break;
    end
    xa=xp;
end
fprintf('\nLa raíz buscada es: %13.6f\n\n',xp);
```

```
function y=integral(x)
y=exp(x);
```

```
function trapecial(funx)
clc;
fprintf('Método Trapecial de Integración\n\n');
xa=input('Teclee el límite inferior xa: ');
xp=input('Teclee el límite superior xp: ');
nt=input('Teclee el número de trapecios: ');
dx=(xp-xa)/nt;
x=xa:dx:xp;
gx=feval(funx,x);
clc;
fprintf('Método Trapecial de Integración\n\n');
fprintf(' i    x        gx \n');
fprintf('-----\n');
for i=1:nt+1
    fprintf('%5.0f %13.6f %13.6f\n',i,x(i),gx(i));
end
area=dx/2*(gx(1)+2*sum(gx(2:nt))+gx(nt+1));
fprintf('\nEl valor de la integral es: %13.6f\n\n',area);
```

```
function simpson(funx)
clc;
fprintf('Método de Simpson de Integración\n\n');
xa=input('Teclee el límite inferior xa: ');
xp=input('Teclee el límite superior xp: ');
nt=input('Teclee el número de áreas: ');
if mod(nt,2)~=0
    fprintf('\nERROR: El número de áreas debe ser PAR\n\n');
    break;
end
```

```

dx=(xp-xa)/nt;
x=xa:dx:xp;
gx=feval(funx,x);
clc;
fprintf('Método de Simpson de Integración\n\n');
fprintf(' i      x      gx \n');
fprintf('-----\n');
for i=1:nt+1
    fprintf('%5.0f %13.6f %13.6f\n',i,x(i),gx(i));
end
area=dx/3*(gx(1)+4*sum(gx(2:2:nt))+2*sum(gx(3:2:nt-1))+gx(nt+1));
fprintf('\nEl valor de la integral es: %13.6f\n\n',area);

```

```

=====

function y=factorial(x)
if x=1
    y=1;
else
    y=x*factorial(x-1);
end

```

¡ Fin de Curso !