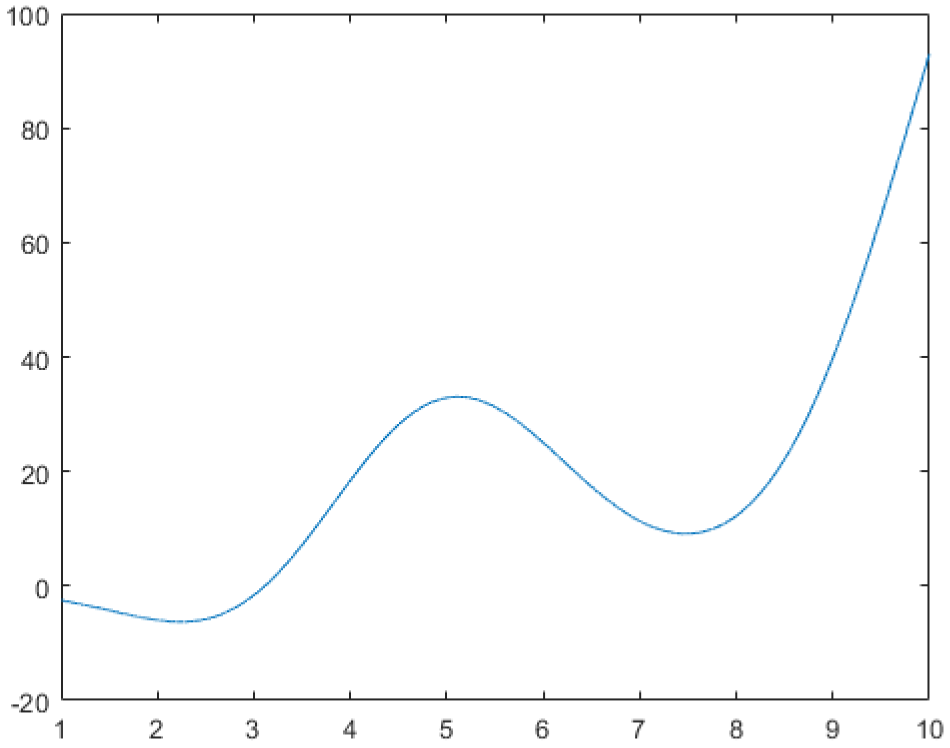
P3: Oleart y Chen

```
% Exercici 11
df1 = deriv(@f1, 1);
df2 = deriv(@f2, 1);

% Exercici 12

plot_f(@f3, 3, 1, 10);
```



Apartat b

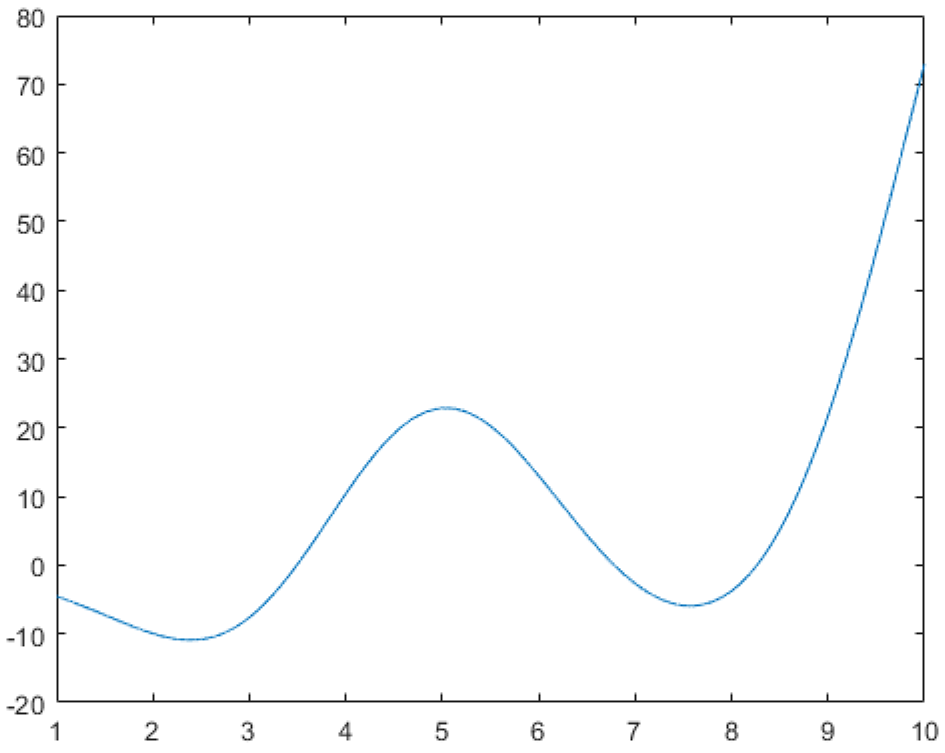
```
x1 = 4;
tol = 10^-10;
itmax = 50;
[xv, fv, iter] = newtonA(x1, tol, itmax, @fun1);
disp(['valor iteratiu: ',num2str(xv)]); fprintf('error iteratiu: %d\n', fv);
fprintf('numero de iteracions: %d\n', iter);
%la d és per especificar el format,

%{
%d is used to display x as an integer.
%f is used to display y as a floating-point number.
%s is used to display str as a string.
%}

%Funció anònima: doubleNumber = @(x) 2 * x;
% on (x) és l'input i l'output és 2*x
```

Apartat c

```
plot_f(@f3, 5, 1, 10);
```



Apartat d

```
x = (1:1/10000:10);
x1_r1 = 3; %diferents llavors des d'on començar newtonA
x1_r2 = 7;
x1_r3 = 8;
tol = 10^-10;
itmax = 50;
[xv_r1, fv2_r1, iter2_r1] = newtonA(x1_r1, tol, itmax, @fun2);
[xv_r2, fv2_r2, iter2_r2] = newtonA(x1_r2, tol, itmax, @fun2);
[xv_r3, fv2_r3, iter2_r3] = newtonA(x1_r3, tol, itmax, @fun2);
```

Apartat e

Con param = 3

```
param = 3;
x1 = 4;
tol = 10^-10;
itmax = 50;
[xv, fv, iter] = newtonB(x1, tol, itmax, @funp, param);
```

con param = 5

```
param = 5;
x = (1:1/10000:10);
x1_r1 = 3;
x1_r2 = 7;
x1_r3 = 8;
tol = 10^-10;
itmax = 50;
[xv_r1, fv2_r1, iter2_r1] = newtonB(x1_r1, tol, itmax, @funp, param);
[xv_r2, fv2_r2, iter2_r2] = newtonB(x1_r2, tol, itmax, @funp, param);
[xv_r3, fv2_r3, iter2_r3] = newtonB(x1_r3, tol, itmax, @funp, param);
```

---------- FUNCTIONS ----------

```
% Function exercici 11
function f1 = f1(x)
    f1 = x^2;
end

function f2 =  f2(x)
    f2 = x^3;
end

% Function exercici 12
function f3 = f3(x, al) % x may be a vector
    f3 = x.^2 - 4*x.*sin(x) + (2*sin(x)).^2 - al*x;
end
```

Apartat a

```
function plot_f(f, al, a, b) %plots function f in interval [a, b]
x = (a:1/1000:b);
plot(x, f(x, al));
end


function f3 = funp(x, param) % x may be a vector
    f3 = x.^2 - 4*x.*sin(x) + (2*sin(x)).^2 - param*x;
end
```

Apartat b

```
function fun1 = fun1(x) % x may be a vector
    fun1 = x.^2 - 4*x.*sin(x) + (2*sin(x)).^2 - 3*x;
end
```

```
valor iteratiu: 4      3.1802      3.1155      3.1131      3.1131      3.1131
error iteratiu: 1.839984e+01
error iteratiu: 1.069211e+00
error iteratiu: 3.716455e-02
error iteratiu: 5.348086e-05
error iteratiu: 1.527134e-10
error iteratiu: 0
numero de iteracions: 5
```

Apartat c

```
function fun1 = fun2(x) % x may be a vector
    fun1 = x.^2 - 4*x.*sin(x) + (2*sin(x)).^2 - 5*x;
end
```

Apartat e

```
%{
function deriv = deriv_param(f, x0, param)
    Ax = 10 ^ -10; % arbitrary small number, to represent infinitessimal
increase
    deriv = (f(x0 + Ax, param) - f(x0, param)) / Ax;
end
%}


% Ja hem creat newtonA i newtonB de manera que acceptin paràmetres,
% l'apartat f ja està fet
```

-------- Other Functions ---------

A baix apareixen les funcions que s'han escrit en arxius diferents i que el publish no inclou, per tal de que el profe corregeixi s'afegeixen com a comentaris.

```
%{
function [xk, fk, it] = newtonB(x1, tol, itmax, fun, param)
    %x1: punt inicial       tol: interval tolerancia
    it = 0;
    xk = [x1];
    fk = [fun(x1, param)];
    while (abs(fun(x1, param)) > tol) && (it < itmax)
        new_x = x1 - (fun(x1, param) / deriv_param(fun, x1, param));
        x1 = new_x;
        xk = [xk, x1];
        fk = [fk, fun(x1, param)];
        it = it + 1;
    end
end


function [xk, fk, it] = newtonA(x1, tol, itmax, fun)
    %x1: punt inicial       tol: interval tolerancia
    it = 0;
    xk = [x1];
    fk = [fun(x1)];
    while (abs(fun(x1)) > tol) && (it < itmax)
        new_x = x1 - (fun(x1) / deriv(fun, x1));
        x1 = new_x;
        xk = [xk, x1];
        fk = [fk, fun(x1)];
        it = it + 1;
    end
end

function deriv = deriv(f, x0)
    Ax = 10 ^ -10; % arbitrary small number, to represent infinitessimal
increase
    deriv = (f(x0 + Ax) - f(x0)) / Ax;
end
%}
```

Published with MATLAB® R2024a