Módulo X: Optimización

- Programación lineal.
- Optimización de funciones no lineales.

```
optimtool
```

Ingresar al link: **Elección de solucionador**

Función de costo lineal y restricciones lineales

Forma estándar:

$$\min_x \ (f^T x) \ \text{such that} \begin{cases} \text{Ax} \ \leq \ b \\ A_{\text{eq}} x = b_{\text{eq}} \\ l_b \leq x \leq u_b \end{cases}$$

Ex. 1.

Maximizar: $Z = 4x_1 - x_2 + 2x_3$

Sujeto a:

$2x_1 + x_2 + 2x_3 \leq 6$
$x_1 - 4x_2 + 2x_3 \leq 0$
$5x_1 - 2x_2 - 2x_3 \leq 4$
$x_1, x_2, x_3 \geq 0$

Antes, cambiar el problema a la forma estándar:

Minimizar: $Z = -4x_1 + x_2 - 2x_3$

Sujeto a:

$2x_1 + x_2 + 2x_3 \leq 6$
$x_1 - 4x_2 + 2x_3 \leq 0$
$5x_1 - 2x_2 - 2x_3 \leq 4$
$x_1, x_2, x_3 \geq 0$

```
f = [-4, 1, -2];
A = [2, 1, 2; 1, -4, 2; 5, -2, -2];
b = [6; 0; 4];
% Este problema no tiene A_eq,  b_eq  ni límite superior u_b
lb = [0, 0, 0];

options = optimoptions('linprog');
options = optimoptions(options,'Display', 'off');
options = optimoptions(options,'Algorithm', 'dual-simplex');

[x,fval,exitflag,output,lambda] = ...
    linprog(f,A,b,[],[],lb,[],[],options);

disp('Solution: ')
```

```
Solution:
```

```
x
```

```
x = 3×1
    1.5556
    0.8889
    1.0000
```

```
fval
```

```
fval = -7.3333
```

Función de costo cuadrática y restricciones lineales

Forma estándar:

$$\min_x \left(\frac{1}{2} x^T \mathrm{H} x + f^T x \right) \text{ such that } \begin{cases} \mathrm{A}x \leq b \\ A_{\mathrm{eq}} x = b_{\mathrm{eq}} \\ l_b \leq x \leq u_b \end{cases}$$

Ex. 2.

Minimizar: $Z = x_1 - 2x_2 + 4x_3 + x_1^2 + 2x_2^2 + 3x_3^2 + x_1 x_3$

Sujeto a:

$$3x_1 + 4x_2 - 2x_3 \leq 10$$
$$-3x_1 + 2x_2 + x_3 \geq 2$$
$$2x_1 + 3x_2 + 4x_3 = 5$$
$$0 \leq x_1 \leq 5$$
$$1 \leq x_2 \leq 5$$
$$0 \leq x_3 \leq 5$$

Convertir a forma estándar:

Minimizar: $Z = x_1 - 2x_2 + 4x_3 + x_1^2 + 2x_2^2 + 3x_3^2 + x_1x_3$

Sujeto a:

$$3x_1 + 4x_2 - 2x_3 \leq 10$$
$$3x_1 - 2x_2 - x_3 \leq -2$$
$$2x_1 + 3x_2 + 4x_3 = 5$$
$$0 \leq x_1 \leq 5$$
$$1 \leq x_2 \leq 5$$
$$0 \leq x_3 \leq 5$$

```
f = [1, -2, 4];

H = 2.*[1, 0, 1;
    0, 2, 0;
    0, 0, 3];

A = [3, 4, -2;
    3, -2, -1;
    2, 3, 4];

b = [10; -2; 5];

A_eq = [2, 3, 4];

b_eq = 5;

lb = [0, 1, 0];
ub = [5, 5, 5];

options = optimoptions('quadprog');
options = optimoptions(options,'Display', 'off');
[x,fval,exitflag,output,lambda] = ...
```

```
quadprog( H, f, A, b, A_eq, b_eq, lb, ub, [], options);
```

```
Warning: Your Hessian is not symmetric. Resetting H=(H+H')/2.
```

```
x
```

```
x = 3×1
    0.2905
    1.4133
    0.0448
```

```
fval
```

```
fval = 1.7413
```

Funciones de costo no lineales y restricciones lineales

Ex. 3. Minimizar la función de Rosenbrock:

$f_{(x,y)} = (1-x)^2 + 100(y-x^2)^2$

Sujeto a:

$x_1 + 2x_2 \leq 1$

```
x_range = linspace( -2, 2, 50 );
y_range = linspace( -2, 2, 50 );
[X, Y] = meshgrid( x_range, y_range );
Z = 100.*( Y - X.^2 ).^2 + ( 1 - X ).^2;
```

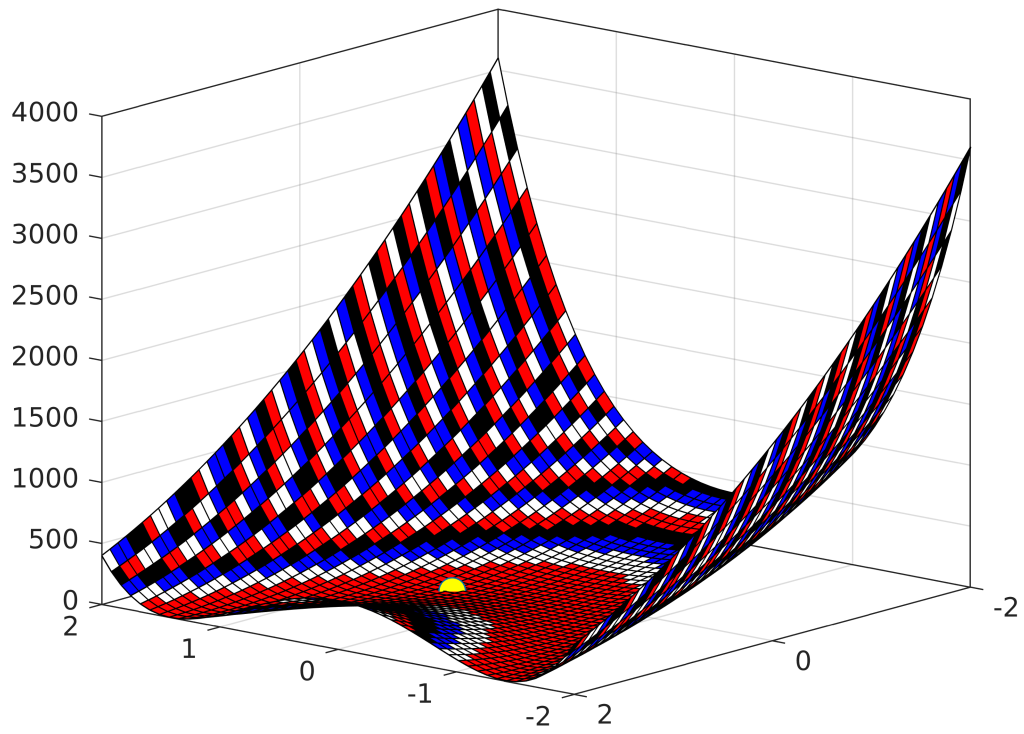
```
x0 = [-1,2];
A = [1, 2];
b = 1;
[x,fval,exitflag,output,lambda] = ...
    fmincon(@rosenbrock_function, x0, A, b)
```

```
Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the default value of the optimality tolerance,
and constraints are satisfied to within the default value of the constraint tolerance.

<stopping criteria details>
x = 1×2
    0.5022    0.2489
fval = 0.2489
exitflag = 1
```

```
output = struct with fields:
         iterations: 30
          funcCount: 103
    constrviolation: 0
           stepsize: 1.1259e-05
          algorithm: 'interior-point'
      firstorderopt: 8.0208e-08
       cgiterations: 2
            message: 'Local minimum found that satisfies the constraints.↲↲Optimization completed because the o
```

```
lambda = struct with fields:
         eqlin: [0×1 double]
      eqnonlin: [0×1 double]
       ineqlin: 0.3309
         lower: [2×1 double]
         upper: [2×1 double]
    ineqnonlin: [0×1 double]
```

```
figure(1)
clf;
surf( X, Y, Z )
colormap flag
view(-140, 16)
hold on
scatter3( x(1), x(2), fval, 100, 'MarkerFaceColor', 'y' )
box on
```




```
function f_xy = rosenbrock_function( v )
```

```
    x = v(1);
    y = v(2);
    f_xy = 100.*( y - x.^2 ).^2 + ( 1 - x ).^2;
end
```