

QUICK START GUIDE

Problem-Based Optimization with Optimization Toolbox™

Use a natural syntax for defining and solving optimization problems, least squares problems, and systems of nonlinear equations.

1. Define Problem

Following the *problem-based workflow*, first create an optimization problem with `optimproblem` to hold the objective, constraints, and associated variables. Create an `eqnproblem` when solving a system of nonlinear equations.

Examples:

```
assignmentProb = optimproblem
responseProb = optimproblem
initialStateProb = eqnproblem
```

2. Define Variables

Create optimization variables with `optimvar`. Set display name and optional dimensions, bounds, and type. Index with integers or character strings.

Examples:

```
x = optimvar("x");
y = optimvar("y");

employees = ["a","b","c"];
tasks = ["t1","t2","t3"];
assign = optimvar("assign",employees,tasks,"LowerBound",0,"UpperBound",1,"Type","integer")
```

3. Define Expressions to Use in Objective, Constraints, and Equations

Directly specify an `OptimizationExpression` with *supported operations*.

Examples:

```
response = -3*(y - x.^3 - x).^2 - (x - 4/3).^2;
totalCost = sum(sum(cost.*assign));
sumByEmployee = sum(assign,2);
sumByTask = sum(assign,1);
```

Use any MATLAB® function by converting it to an optimization expression with `fcn2optimexpr`.

Examples:

```
a = 4;
xyfcn = @(x,y,a)gamma(y)*a*x.^2;
xyexpr = fcn2optimexpr(xyfcn,x,y,a);
```

4. Define Objective

Set the *sense* of the optimization. Set the *objective function* with a scalar `OptimizationExpression`.

Examples:

```
responseProb.ObjectiveSense = "maximize";
responseProb.Objective = response;

assignmentProb.ObjectiveSense = "minimize";
assignmentProb.Objective = totalCost;
```

5. Define Constraints and Equations

Combine **OptimizationExpressions** with a relational operator to specify an **OptimizationConstraint** or an **OptimizationEquality**. Assign to a problem.

Examples:

```
responseProb.Constraints.ellipse = x.^2/2 + y.^2/4 <= 1;
responseProb.Constraints.xyconstr = xyexpr >= 1;

assignmentProb.Constraints.oneTaskPerEmployee = sumByTask <= 1;
assignmentProb.Constraints.oneEmployeePerTask = sumByEmployee == 1;

initialStateProb.Equations.eq1 = x*cos(y) + y*sin(x) == 1/2;
initialStateProb.Equations.eq2 = exp(-exp(-(x + y))) == y*(1 + x^2);
```

6. Review

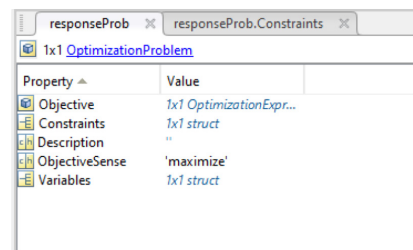
Display with **show** and **showbounds**. Write to a file with **write** and **writebounds**.

```
OptimizationProblem :
minimize :
0.81472*assign('a', 't1') + 0.90579*assign('b', 't1') + 0.1269
+ 0.91338*assign('a', 't2') + 0.63236*assign('b', 't2') + 0.00975
+ 0.2785*assign('a', 't3') + 0.54688*assign('b', 't3') + 0.95751

subject to oneTaskPerEmployee:
assign('a', 't1') + assign('b', 't1') + assign('c', 't1') <= 1
assign('a', 't2') + assign('b', 't2') + assign('c', 't2') <= 1
assign('a', 't3') + assign('b', 't3') + assign('c', 't3') <= 1

subject to oneEmployeePerTask:
assign('a', 't1') + assign('a', 't2') + assign('a', 't3') == 1
```

View with the Workspace browser.



Property	Value
Objective	1x1 OptimizationExpr...
Constraints	1x1 struct
Description	"
ObjectiveSense	'maximize'
Variables	1x1 struct

7. Solve and Analyze

Solve the problem, providing an initial point for nonlinear problems. The **solve** function returns solution values, objectives values, the reason the problem stopped, and more.

Example:

```
x0.x = 0;
x0.y = 0;
[sol,fval,exitflag] = solve(responseProb,x0)
```

```
sol = struct with fields:
    x: 0.8883
    y: 1.5563
    fval: -0.2013
    exitflag =
        OptimalSolution
```

Solve with *optimization options*.

Example:

```
o = optimoptions(assignProb,"MaxTime",10);
sol = solve(assignmentProb,"Options",o)
```

Do More

- Use **evaluate** and **infeasibility** to analyze results
- **Interpret** and **improve** results
- Convert to solver-based form with **prob2struct**

Learn more: mathworks.com/help/optim