Non-convex Function Minimization in MATLAB

This MATLAB function **OMBOC** (Optimization Method Based on Optimal Control), implements two methods for finding the unconstrained minimum of a non-convex function. These methods are based on optimal control theory and are designed to efficiently converge to a solution even in complex optimization landscapes.

Methods

`OMBOC` is unique in its integration of optimization with control theory, providing a robust approach to solving challenging optimization problems.

`OMBOC` provides two methods for optimization:

1. **Method 1 (Algorithm in [1])**
   - **Characteristics**:
     - Relatively simple parameter selection.
     - Find different local minimum points.
     - Higher computational time due to the need for solving FBDEs.
   - **Reference**: Xu et al. (2023)
2. **Method 2 (Algorithm in [2])**
   - **Characteristics**:
     - Find different local minimum points.
     - Higher precision.
     - Reduced computational time compared to Method 1.
   - **Reference**: Zhang et al. (2023)

Comparison with Other Methods

- **Gradient Descent**: `OMBOC` offers faster convergence than traditional gradient descent methods.
- **Newton's Method**: `OMBOC` is more stable and versatile, especially in cases where the Hessian matrix is singular or indefinite.

Usage

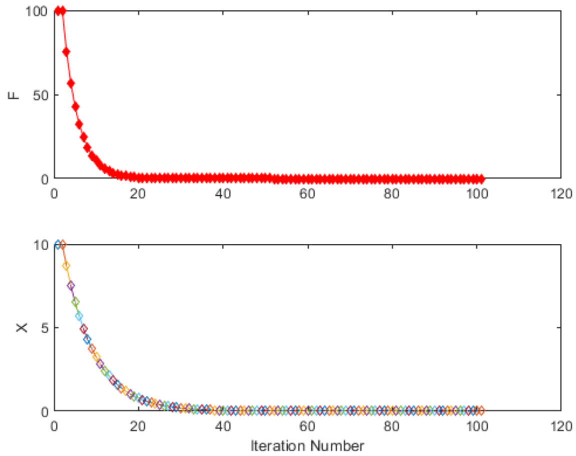
- Before using the MATLAB function, you should download the **CasADi** toolkit and add it to your MATLAB path.
- The different local minimum points can be found by adjusting **R** in some cases.
- Choose the appropriate control weight matrix **R** and the step size **Lambda** to best suit your needs for convergence speed and computational efficiency.
- To use the MATLAB function, select the desired method based on your requirements for calculation cost and efficiency:
  - To implement **Method 1**, select the appropriate option in the function.
  - To implement **Method 2**, select the corresponding option in the function.

Example Usage

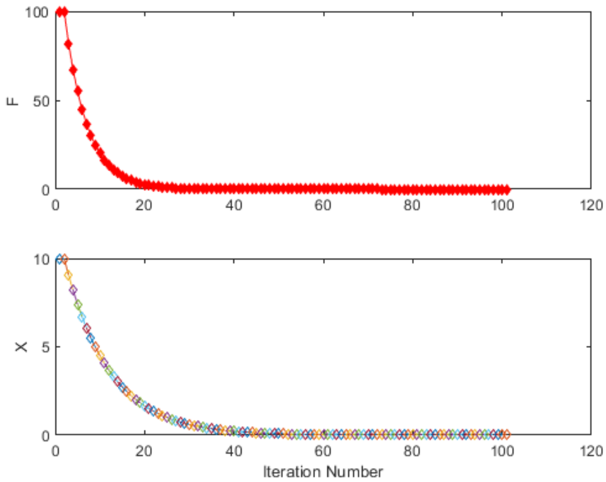
```matlab
% Define the objective function
myfun = @(x) x - 4*x^2 + 0.2*x^3 + 2*x^4;

% Initial value
x0 = 0.5;

% Call OMBOC with Method 1
[x, fval] = OMBOC(@(x)myfun(x), x0, eye(1), 'Method1', 100, 0.1, [], 1e-3,
'on');

% Display the result
disp(['Optimal X: ', num2str(x)]);
disp(['Function Value: ', num2str(fval)]);
```

Demo1（Test different method）

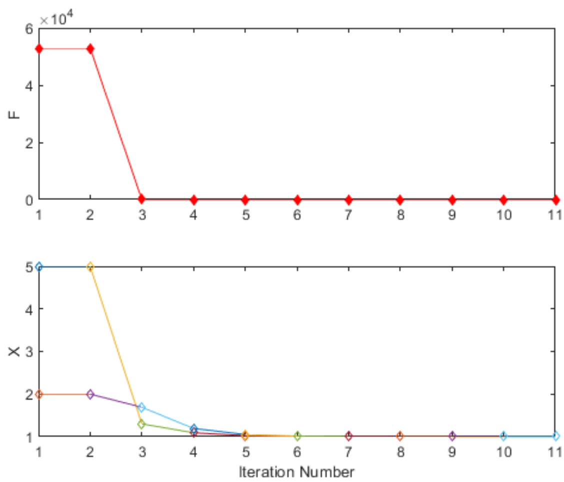
Method1



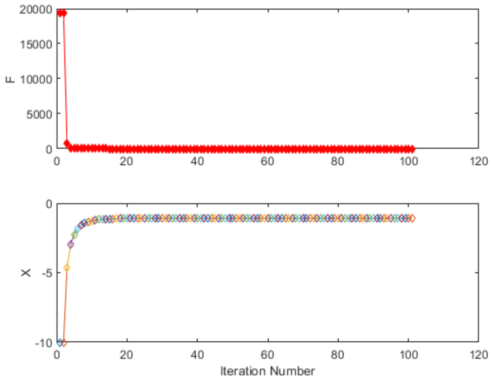
Method2

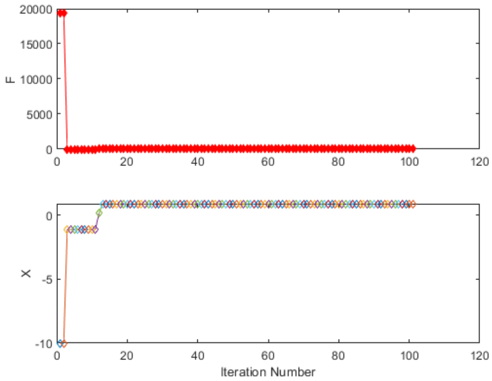


Demo2（Standard test function）



Demo3（Find different local minimum point）





Instruction for using OMBOC

OMBOC finds an unconstrained minimum of a non-convex function of several variables.
OMBOC attempts to solve problems of the form:

min F(X)

X = OBOMBOC (FUN,X0,R,Method,MaxIter,Lambda,U,TolX,ShowGraph,varargin) starts at X0 and finds a minimum X to the function FUN. FUN a accepts input X and returns a scalar function value F evaluated at X. X0 may be a scalar, vector, or matrix. R is the positive definite control weight matrix.

Method is a flag to select the optimization methods listed below.

'Method1' Algorithm in [1] .(default)

'Method2' Algorithm in (9)-(10) of [2].

If you want to decrease the computation time (especially in high-dimensional cases), it is recommended this method.

Lambda is an iteration step size in Method1 or Method2.

MaxIter is the maximum number of iterations allowed. U is the guess of the initial control sequence. TolX is the termination tolerance on x (1e-3 for default). ShowGraph is a {'on'}|{'off'} flag to show the iteration process or not.

[X,FVAL] = OMBOC(FUN,X0,...) returns the value of the objective function FUN at the solution X.

[X,FVAL,EXITFLAG] = OMBOC(FUN,X0,...) returns an EXITFLAG that describes the exit condition of OMBOC. Possible values of EXITFLAG and the corresponding exit conditions are listed below.

All algorithms:

1 First order optimality conditions satisfied to the specified tolerance.

0 Maximum number of function evaluations or iterations reached.

-1 Preserved.

[X,FVAL,EXITFLAG,GRAD] = OMBOC(FUN,X0,...) returns the value of the gradient of FUN at the solution X.

Here, myfun is the objective function that you want to minimize:

myfun = @(x)x-4*x^2+0.2*x^3 +2*x^4;

Finally, pass these anonymous functions to OMBOC:

x = OMBOC(@(x)myfun(x),x0,...,TolX,)

See details in Demos.

OMBOC uses the Huanshui Zhang and Hongxia Wang optimization method based on optimal control (described in [1] and [2]), and is coded in MATLAB 2008a and tested in subsequent versions of MATLAB.

References:

[1] Yeming Xu, Ziyuan Guo, Hongxia Wang, Huanshui Zhang, "Optimization Method Based on Optimal Control," 2023, https://arxiv.org/abs/2309.05280.

[2] Huanshui Zhang, Hongxia Wang, "Optimization Methods Rooted in Optimal Control," 2023, https://arxiv.org/abs/2312.01334.

See also FMINCON, FMINUNC, FMINBND, FMINSEARCH, @, FUNCTION_HANDLE.

$Coder: Yeming Xu&Chuanzhi Lv&Kai Peng$

$Tester: Yeming Xu$

$Reviewer: Hongxia Wang$