Assignment 6: Scalar Solvers

Tufts ME 149: Optimal Control For Robotics

Assigned: Feb 20 — Due: Feb 28 at 11:55pm

Introduction

In this assignment you will implement two iterative solvers in Matlab: Ridder's method for scalar root finding and the Golden Section Search for scalar optimization. Both methods start with a bracketed search interval.

Ridder's Method

Ridder's method is an algorithm that searches for the root of a smooth function, starting with an interval that brackets the root. The method works by fitting an exponential through three data points on each iteration and then computing the root of that exponential. Ridder's method is described in section 9.2 of Numerical Recipes in C.

Golden Section Search

The golden section search is method for robust optimization of a scalar function on a bounded (bracketed) interval. The algorithm keeps track of three points that are known to bracket the minimum. On each iteration it samples a new point such that the relative size of the two sub-intervals remains constant. It turns out that the relative size is given by the golden ratio, hence the name. See section 10.1 of Numerical Recipes in C for implementation details.

Deliverables

Implement Ridder's method and the golden section search using the template files that are included with the assignment, and submit both of these files to Trunk. You code will be run through an automated unit test, so make sure that to test it yourself on several functions to be sure that it works. Carefully read the inputs and output of each function.

Write-Up

In addition to the Matlab file for you root solver and your optimization, please also upload a short write-up for this assignment:
`hw_06_studentName_writeup.txt`

- Header: full name, date, assignment name and number
- List any other students that you worked with.
- List any other references that you used.
- How long did this assignment take you?
- Briefly describe how you tested each function to verify that it was implemented correctly.

Root Solving Challenge

You may choose to implement Brent's method for root finding instead of Ridder's method. Use the template that is provided, but change the name to `brentRootSolver` and update the documentation. You will receive an extra 5 points on your assignment, although the max score will still be 50. This method is described in section 9.3 of Numerical Recipes in C and it is what Matlab uses to implement the `fzero` function.

Optimization Challenge

You may choose to implement Brent's method for optimization instead of the golden section search. Use the template that is provided, but change the name to `brentOptimization` and update the documentation. You will receive an extra 10 points on your assignment, although the max score will still be 50. This method is described in section 10.2 of Numerical Recipes in C and it is what Matlab uses to implement the `fminbnd` function.

```
function [xZero, fZero, nEval, exitCode] = ...
                      riddersMethod(func, xLow, xUpp, xTol, fTol, nEvalMax)
% [xZero, fZero, nEval, exitCode] = ...
%                      riddersMethod(func, xLow, xUpp, xTol, fTol, nEvalMax)
%
% This function solves uses Ridder's method to compute the root of a
% function. The root must be bracketed:
%
%       sign(func(xLow)) ~= sign(func(xUpp))
%
% INPUTS:
%   func = a function for a SISO function: y = f(x)
%   xLow = the lower search bound
%   xUpp = the upper search bound
%   xTol = if |xLow - xUpp| < xTol then return success
%   fTol = if |fVal| < fTol then return success
%   nEvalMax = maximum number of function evaluations
%
% OUTPUTS:
%   xZero = the root of the function on the domain [xLow, xUpp]
%   fZero = func(xZero) = function value at xZero
%   nEval = number of function evaluations
%   exitCode = integer indicating the status of the solution:
%      1 --> successful convergence (either xTol or fTol)
%      0 --> maximum iteration count reached
%     -2 --> [xLow, xUpp] does not bracket a root (bad input)
%
% NOTES:
%   1) The function must be smooth and continuous on [xLow, xUpp]
%   2) sign(f(xLow)) ~= sign(f(xUpp))
%
% REFERENCE:
%       Numerical Recipes in C, 1992 edition, by
%           William H. Press,âĂŐ Saul A. Teukolsky,âĂŐ
%           William T. Vetterling, Brian P. Flannery
%       --> Chapter 9, Section 2
%

%%%% TODO:  implement this function

end
```

```
function [xMin, fMin, nEval, exitCode] = goldenSection(func, xLow, xUpp, xTol, nEvalMax)
% [xMin, fMin, nEval, exitCode] = goldenSection(func, xLow, xUpp, xTol, nEvalMax)
%
% This function solves uses a golden section search to compute the minimum
% value of a smooth and continuous function on a bracketed search interval
%
% INPUTS:
%   func = a function for a SISO function: y = f(x)
%   xLow = the lower search bound (exclusive)
%   xUpp = the upper search bound (exclusive)
%   xTol = tolerance on the search interval
%   nEvalMax = maximum number of function evaluations
%
% OUTPUTS:
%   xZero = the root of the function on the domain [xLow, xUpp]
%   fZero = func(xZero) = function value at xZero
%   nEval = number of function evaluations
%   exitCode = integer indicating the status of the solution:
%      1 --> successful convergence
%      0 --> maximum iteration count reached
%
% NOTES:
%   1) The function must be smooth and continuous on [xLow, xUpp]
%   2) [xLow, xUpp] must bracket a local minima
%
% If the minimum value is at one of the roots, then this algorithm will
% converge to within xTol of the boundary and then return success.
%
% The bracket tolerance xTol should be larger than
% sqrt(eps)*(abs(x1)+abs(x2))
% This limit is derived in Numerical Recipes and has to do with floating
% point precision and Taylor expansions near extremum of functions.
% You can set the limit smaller than this value, but the result will be
% additional function evaluations without improving accuracy.
%
% REFERENCE:
%       Numerical Recipes in C, 1992 edition, by
%           William H. Press,âĂŐ Saul A. Teukolsky,âĂŐ
%           William T. Vetterling, Brian P. Flannery
%       --> Chapter 10, Section 1
%

%%%% TODO:  implement this function

end
```