UMEÅ UNIVERSITET
Department of Computer Science

December 8, 2021

5dv005ht21

Assignment 2

Namn Mathias Hallberg
Gustaf Söderlund

Mail c19mhg@student.umu.se
et14gsd@student.umu.se

Version 1

Course
Scientific Computing
Handledare
Carl Christian Kjelgaard Mikkelsen Fredrik Petri

Contents

1 Introduction

In this paper the implementation of the Assingment 2 is presented. The paper is divided in the different sections to give the reader an easier read.

In the first section the zero theorems assignment is presented together with the matlab code.

In the second section the Approximation of derivatives assignment is presented together with proof of Taylor's theorem, the matlab code and a experimental evidence.

In the third section Hermite's piece-wise approximation assignment is presented together with a proof, matlab code and a experimental evidence.

In the last section the event location for ordinary differential equations assignment is presented

2 The zero theorems

In this assigment a script was developed to illustrate Rolle's Theorem and the mean Value theorem. This illustation is represented in Figure 1.

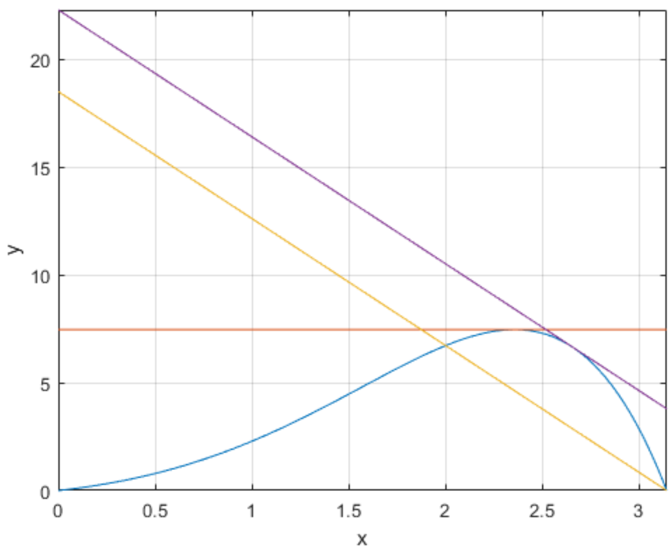

Figure 1: The figure illustrates the result from the MyZeroTheorem.m

2.1 MyZeroTheorem.m

```
% Illustration of Rolle's Theorem and the mean value theorem differentiation
% PROGRAMMING by Gustaf S derlund (et14gsd@cs.umu.se)
%                Mathias Hallberg (c19mhg@cs.umu.se)
%                Gustaf S derlund (et14gsd@cs.umu.se)
%  2021-12-03 Created
clc;clear;
% Define a nice function
f=@(x)exp(x).*sin(x);

% Define the derivative fp (fprime) of f
fp=@(x)exp(x).*(sin(x)+cos(x));

% Interval
a=0; b=pi;

% Number of subintervals
n=100;

% Sample points for plotting
s=linspace(a,b,n+1);

% Plot the graph
h=figure; plot(s,f(s));

% Hold the graph
hold on;
% Turn on grid
grid on;

% Axis tight
axis tight

% ////////////////////////////////////////////////////////////////////////
%  Illustration of Runge's theorem
% ////////////////////////////////////////////////////////////////////////


% Initial search bracket
x0=a;
x1=b;

% The function values corresponding to the initial search bracket
fp0=fp(a);
fp1=fp(b);

% Tolerances and maxit for bisection.
eps=10^-10; delta=10^-10; maxit=101;
```

```
% Run the bisection algorithm to find the zero c of fp
c=bisection(fp, x0, x1, fp0, fp1, delta, eps, maxit,
    false);

% Define the tangent at this point; this a constant
    function.
w=@(x)ones(size(x))*f(c);

% Plot the tangent
plot(s, w(s));

%
    //////////////////////////////////////////////////////////////////////////
%  Illustration of the mean value theorem
%
    //////////////////////////////////////////////////////////////////////////

% Define points for corde
x0=2; x1=pi;

% Compute corresponding function values
f0=f(x0);
f1=f(x1);

% Define the linear function which connects (x0,f0) with
    (x1,f1)
h=f(x0);
b=-((f1-f0)/(x1-x0)*2)+h;
p=@(s) ((f1-f0)/(x1-x0)).*s+b;

% Plot the straight line between (x0,f0) with (x1,f1)
plot(s,p(s))

% Compute the slope of the corde
yp=(f1-f0)/(x1-x0);

% Define an auxiliary function which is zero when fp
    equals yp
g=@(s) yp - fp(s);

% Run the bisection algorithm to find a zero c of g
c=bisection(g, x0, x1, g(x0), g(x1), delta, eps, maxit,
    false);

% Define the line which is tangent to the graph of f at
    the point (c,f(c))
h=f(c)
b=-(yp*c)+h;
q=@(s)yp.*s+b;

% Plot the tangent line
plot(s,q(s));
```

```

% Labels
xlabel('x'); ylabel('y');

% Print the figure to a file
print('MyZeroTheorems','-depsc2');
```

3 Approximation of derivatives

In this assigment the script MyDerivs.m was developed of the given code from Carl Christian Kjelgaard Mikkelsen. First of a few functions from the Taylor's theorem needed to be proved. The proof of the theorem is presented in the subsection *Proof Taylor's Theorem*.

3.1 Proof Taylor's Theorem

Let $f : I \to \mathbb{R}$ be a function which is infinitely often differetiable. Let $h > 0$. By Taylor's theorem we have:

$$f(x+h) = f(x)+f'(x)h+\frac{1}{2}f''(x)h^2+\dots,\frac{1}{p!}f^{(p)}(x)h^p+O(h^{p+1}), \quad h \to 0, \quad h > 0.$$

3.1.1 First proof

The Equation(1) will be proved using the method from the chapter 8.2 shown in [1].

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2),\ h \to 0,\ h > 0 \tag{1}$$

Assume that $f \in C^3$, then by using Taylors theorem:

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + O(h^3) \quad h \to 0, \quad h > 0.$$

Replace h with -h

$$-f(x-h) = -f(x) + f'(x)h - \frac{1}{2}f''(x)h^2 + O(h^3) \quad h \to 0, \quad h > 0.$$

Put the two functions together

$$f(x+h) - f(x-h) = 2f'(x)h + O(h^3) \quad h \to 0, \quad h > 0.$$

Extract the 2 and h from the right side. $O(h^3)$ turns into $O(h^2)$

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + O(h^2) \quad h \to 0, \quad h > 0.$$

The proof is now complete.

3.1.2 Second proof

Show that

$$f'(x) = \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h} + O(h^2), \ h \to 0, \ h > 0. \qquad (2)$$

Assume that $f \in C^3$, then by using Taylors theorem:

$$4f(x+h) = 4f(x) + 4f'(x)h + 2f''(x)h^2 + O(h^3), \quad h \to 0, \quad h > 0.$$

Not taking h into accountability

$$-3f(x) = -3f(x) - 3f'(x)0 - 3f''(x)0^2 + O(h^3) \quad h \to 0, \quad h > 0.$$

replacing h with 2h

$$-f(x+2h) = -f(x) - 2f'(x)h - 2f''(x)h^2 + O(h^3) \quad h \to 0, \quad h > 0.$$

Put the functions together

$$-3f(x)+4f(x+h)-f(x+2h) = 4f(x)-3f(x)-f(x)+4f'(x)h-2f'(x)h+2f''(x)h^2+$$
$$-2f''(x)h^2 + O(h^3) \quad h \to 0, \quad h > 0.$$

Simplify

$$-3f(x)+4f(x+h)-f(x+2h) = 0f(x)+2f'(x)h+0f''(x)+O(h^3) = 2f'(x)h+O(h^3)$$
$$h \to 0, \quad h > 0.$$

Which implies the following

$$\frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h} = f'(x) + O(h^2), \quad h \to 0, \quad h > 0.$$

The proof is now complete.

3.1.3 Third Proof

Show that

$$f'(x) = \frac{f(x-2h) - 4f(x+h) + f(x)}{2h} + O(h^2), \ h \to 0, \ h > 0. \qquad (3)$$

Assume that $f \in C^3$ then by using Taylors theorem:

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + O(h^3), \quad h \to 0, \quad h > 0.$$

replace h with -2h

$$f(x-2h) = f(x) - 2f'(x)h + 2f''(x)h^2 + O(h^3), \quad h \to 0, \quad h > 0.$$

Multiply with the constant -4

$$-4f(x+h) = -4f(x) + 4f'(x)h - 2f''(x)h^2 + O(h^3), \quad h \to 0, \quad h > 0.$$

Take no consideration of h and multiply by 3.

$$3f(x) = 3f(x) + 3f'(x)0 + \frac{3}{2}f''(x)0^2 + O(h^3), \quad h \to 0, \quad h > 0.$$

Put the functions together

$$f(x-2h) - 4f(x+h) + 3f(x) = f(x) - 4f(x) + 3f(x) - 2'f(x)h + 4f'(x)h+$$
$$+2f''(x)h^2 - 2f''(x)h^2 + O(h^3), \quad h \to 0, \quad h > 0.$$

Which when simplified looks like

$$f(x-2h) - 4f(x+h) + 3f(x) = 2f'(x)h + O(h^3), \quad h \to 0, \quad h > 0.$$

Which implies the follwing

$$\frac{f(x-2h) - 4f(x+h) + 3f(x) = 2f'(x)h}{2h} = f'(x) + O(h^2), \quad h \to 0, \quad h > 0.$$

The proof is now complete.

3.2 MyDerivs.m

```
function fp=MyDerivs(y,h)

% MyDerivs  Computes approximations of derivatives
%
% CALL SEQUENCE: fp=MyDerivs(y, h)
%
% INPUT:
%   y     a one dimensional array of function values, y = f(x)
%   h     the spacing between the sample points x
%
% OUTPUT
%   fp    a one dimension array such that fp(i) approximates f'(x(i))
%
% ALGORITHM: Space central and asymmetric finite difference as needed
%
% MINIMAL WORKING EXAMPLE: MyDerivsMWE

% PROGRAMMING by Carl Christian Kjelgaard Mikkelsen (spock@cs.umu.se)
%                Mathias Hallberg (c19mhg@cs.umu.se)
%                Gustaf S derlund (et14gsd@cs.umu.se)
%   2018-11-26 Extracted from a working code
%   2021-12-6  Finished the working code

% Extract the number of points
m=numel(y);

```

```
% The exercise is pointless unless there are at least 3
    points
if m<3
    return;
end

% Allocate space for derivatives
fp=zeros(size(y));

% Do asymmetric approximation of the derivative at the
    left endpoint
fp(1)=(-3*(y(1))+4*(y(2))-y(3))/(2*h);
% Do space central approximation of all derivatives at
    the internal points
% Do a for-loop *before* you attempt to do this as an
    array operation
for i=2:m-1
    fp(i)=(y(i+1)-y(i-1))/(2*h);
end

% Do asymmetric approximation of the derivatives at the
    right endpoint
fp(m)=(y(m-2)-4*y(m-1)+3*y(m))/(2*h);
```

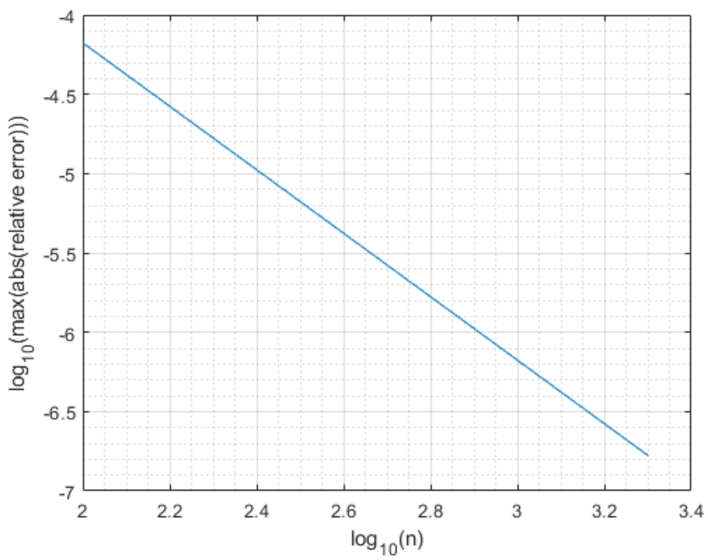

Figure 2: The figure represents the output of the program MyderivsMWE1. In this illustration it's shown that the error of the approximation of the first derivative is decaying when the size between the sample points is decreasing.

3.3 Experimental evidence

As we can see according to the points in figure 3 that for every step through step x-axis it moves two steps in the y-axis, hence, the support for the statement that the error committed $O(h^2)$ is concluded.

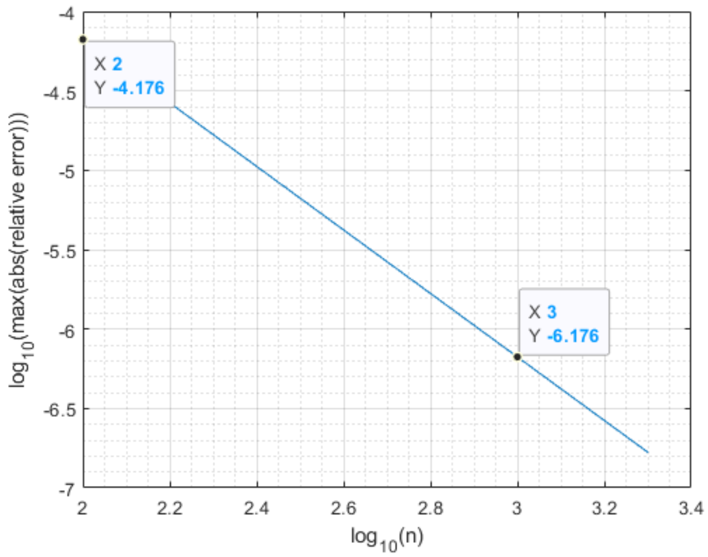

Figure 3: The figure represents the output of the program MyDerivsMWE1. In this illustration two points are set to show the value of x and y at the two different coordinates.

4 Hermite's piece-wise approximation

$$p_0(t) = (1 + 2t)(1 - t)^2, \; p_1(t) = t^2(3 - 2t)$$

Show that

$$p_0(0) = (1 + 2 * 0) * (1 - 0)^2 = 1,$$
$$p_0(1) = (1 + 2 * 1) * (1 - 1)^2 = 0,$$
$$p_1(0) = (1 + 2 * 0) * (1 - 0)^2 = 0,$$
$$p_1(1) = (1 + 2 * 1) * (1 - 1)^2 = 1,$$

$$p_0'(t) = 6 * t^2 - 6 * t, \; p_1'(t) = 6 * t - 6 * t^2$$

Show that

$$p_0'(0) = 6 * 0^2 - 6 * t = 0$$
$$p_0'(1) = 6 * 1^2 - 6 * 1 = 0$$
$$p_1'(0) = 6 * 0 - 6 * 0^2 = 0$$
$$p_1'(1) = 6 * 1 - 6 * 1^2 = 0$$

$$q_0(t) = t(1-t)^2,\ q_1(t) = t^2(t-1)$$

Show that

$$q_0(0) = 0 * (1-0)^2 = 0 * 0 = 0$$
$$q_0(1) = 1 * (1-1)^2 = 1 * 0 = 0$$
$$q_1(0) = 0 * (1-0)^2 = 0$$
$$q_1(1) = 1 * (1-1)^2 = 0$$

$$q_0' = (1-t)^2 - 2(1-t) * t,\ q_1' = 3t^2 - 2 * t$$

Show that

$$q_0'(0) = (1-0)^2 - 2(1-0) * 0 = 1 - 0 = 1$$
$$q_0'(1) = (1-1)^2 - 2(1-1) * 1 = 0$$
$$q_1'(0) = 3 * 0^2 - 2 * 0 = 0$$
$$q_1'(1) = 3 * 1^2 - 2 * 1 = 1$$

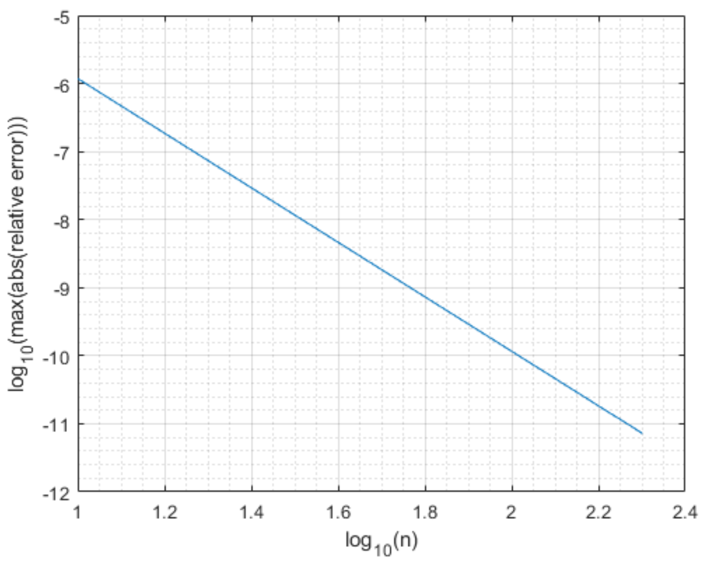

Figure 4:

4.1 Show that Hermite's approximation satisfies

Prerequisites

- Let $\phi : [a, b] \to \mathbb{R}$ denote the linear function which maps a into 0 and b into 1
- So that $\phi(x) = \frac{x-a}{b-a}$
- Hermite's approximation of $f : [a, b] \to \mathbb{R}$ is the polynomial $p : [a, b] \to \mathbb{R}$ given by
  $p(x) = f(a)p_0(\phi(x)) + f(b)p_1(\phi(x))f'(a)(b-a)q_0(\phi(x)) + f'(b)(b-a)q_1(\phi(x))$

Show that Hermite's approximation satisfies

$$p(a) = f(a)$$

$$p(a) = f(a)p_0(0) + f(b)p_1(0) + f'(a)(b-a)q_0(0) + f'(b)(b-a)q_1(0)$$

$$p(a) = f(a) * 1 + f'(a)(b-a) * 0 + f'(b)(b-a) * 0$$

Simplified

$$p(a) = f(a)$$

Show that Hermite's approximation satisfies

$$p(b) = f(b)$$

$$p(b) = f(a)p_0(1) + f(b)p_1(1) + f'(a)(b-a)q_0(1) + f'(b)(b-a)q_1(1)$$

$$p(b) = f(a) * 0 + f(b) * 1 + f'(a)(b-a) * 0 + f'(b)(b-a) * 0$$

Simplifed

$$p(b) = f(b)$$

Show that Hermite's approximation satisfies

$$p'(a) = f'(a)$$

p'(a) = f'(a)p$_0$(0) $+ f'(b)p_1(0) + f''(a)(b-a)q_0(0) + f''(b)(b-a)q_1(0)$

$$p'(a) = f'(a) * 1 + f'(b) * 0 + f''(a)(b-a) * 0 + f''(b)(b-a) * 0$$

Simplified

$$p'(a) = f'(a)$$

Show that Hermite's approximation satisfies

$$p'(b) = f'(b)$$

$$p'(b) = f'(a)p_0(1) + f'(b)p_1(1) + f''(a)(b-a)q_0(1) + f''(b)(b-a)q_1(1)$$

$$p'(b) = f'(a) * 0 + f'(b) * 1 + f''(a)(b-a) * 0 + f''(b)(b-a) * 0$$

Simplified

$$p'(b) = f'(b)$$

4.2 MyPieceWiseHermite.m

```
function z=MyPiecewiseHermite(s,f,fp,t)

% A2F4 Evaluate Hermite's piecewise approximation

% INPUT:
%   s    a linear array of m points where f and f' are known
%   f    the function values, y = f(s)
%   fp   the derivatives, yp = f'(s)
%   t    a linear array of sample points where z=p(t) is sought
%
% OUTPUT:
%   z    the values of Hermite's piecewise approximation, z = p(t)
%
% PROGRAMMING by Carl Christian Kjelgaard Mikkelsen (spock@cs.umu.se)
%                 Mathias Hallberg (c19mhg@cs.umu.se)
%                 Gustaf S derlund (et14gsd@cs.umu.se)
%   2018-11-25 Initial programming and testing
%   2021-12-07 Finished the program

% Determine the number of points
m=numel(t);

% Define the polynomial p0
p0=@(t) (1+2.*t).*(1-t).^2;
% Define the polynomial p1
p1=@(t) t.^2.*(3-2.*t);
% Define the polynomial q0
q0=@(t) t.*(1-t).^2;
% Define the polynomial q1
q1=@(t) t.^2.*(t-1);

% Determine the number of sample points where we know f and f'
n=numel(s);

% Set size for z
% Loop over all points of t
for i=1:m
    % Isolate the ith value of t into a variable tau
    tau=t(i);
    % Find the interval s(j), s(j+1) which contains tau
    j=find(s(1:n-1)<=tau,1,'last');
    % Isolate the endpoints of the interval which contains tau into a, b
    a=s(j); b=s(j+1);
    % Map tau into a point x in [0,1] using the linear transformation
    % which maps a into 0 and b into 1
```

```
    x=(tau-a)./(b-a);
    % Compute Hermite's approximation of f(tau)
        corresponding to the
    % sub-interval [a,b]
    z(i)=f(j)*p0(x)+f(j+1)*p1(x)+fp(j)*(b-a)*q0(x)+fp(j
        +1)*(b-a)*q1(x);
end
```

4.3 Experimental evidence

As we can see according to the points in figure 5 that for every step through step x-axis it moves four steps in the y-axis, hence, the support for the statement that the error $f(x) - p(x)$ decays as $O(h^4)$ is concluded.

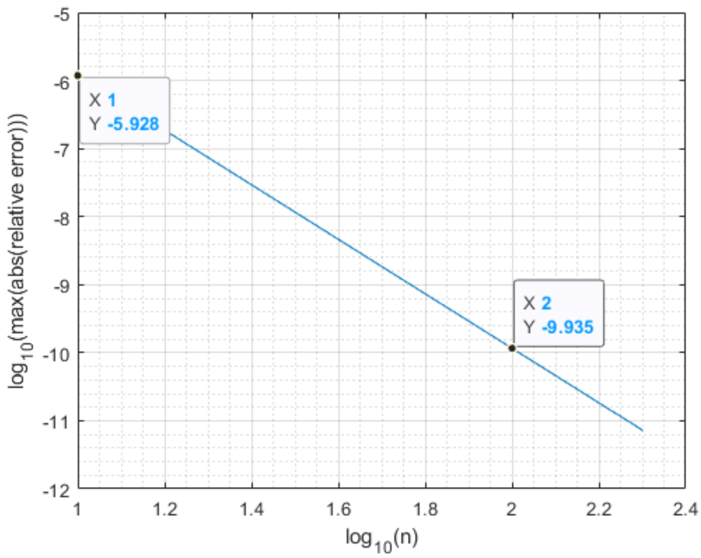

Figure 5: The figure represents the output of the program MyPieceWiseHermiteMWE1. In this illustration two points are set to show the value of x and y at the two different coordinates.

5 Event location for ordinary differential equations

5.1 MyEvent.m

```
% MyEvent.m, which simulates a impact of a projactile on a hill.
%
% PROGRAMMING by Carl Christian Kjelgaard Mikkelsen (spock@cs.umu.se)
%                Mathias Hallberg (c19mhg@cs.umu.se)
%                Gustaf S derlund (et14gsd@cs.umu.se)
%   2021-12-6  Finished the working code
clear;clc;
% Load shells models
load shells.mat

% Specify shell and enviroment
param=struct('mass',10,'cali',0.088,'drag',@(x)mcg7(x),'atmo',@(x)atmosisa(x),'grav',@(x)9.82,'wind',@(t,x)[0,0]);

% Set the muzzle velocity and the elevation of the gun
v0=780; theta=60*pi/180;

% Select the method which will be used to integrate the trajectory
method='rk2';

% Select the basic time step size and the maximum number of time steps
dt=0.1; maxstep=2000;

% Compute the range of the shell
% Where t is our point of comparison
[r, ~, t, tra]=range_rkx(param,v0,theta,method,dt,maxstep);
% Calculate hermites approximation of t->x(t) and t->y(t)
% Maximum number of iterations
a=0;b=numel(tra(1,:));
maxit=b;

% Allocate space
n=zeros(maxit,1); mre=zeros(maxit,1);
t=linspace(t(1,1),t(1,end),100*maxit+1);

% Number of sample points
n(maxit)=10*maxit;

% Sample points
s=linspace(a,b,n(maxit)+1);

x=@(t)MyPiecewiseHermite(s,tra(1,:),tra(3,:),t);
y=@(t)MyPiecewiseHermite(s,tra(2,:),tra(4,:),t);
```

```
% Below follows a long sequence of commands which demonstrates how to get
% a very nice plot of the trajectory automatically
hold on;
% Obtain the coordinates of the corners of the screen
screen=get(groot,'Screensize');

% Isolate the width and height of the screen measured in pixels
sw=screen(3); sh=screen(4);

% Obtain a handle to a new figure
hFig=gcf;

% Set the position of the desired window
set(hFig,'Position',[0 sh/4 sw/2 sh/2]);

% Plot the trajectory of the shell and the hill
hillSpace=linspace(0,16000, 2000);
hillValues =a2f6(hillSpace);
hill=plot(hillSpace,hillValues);
shell=plot(x(t),y(t));

func=@(t) y(t)-a2f6(x(t));
% Run bisection 100000 times to get a approximation as good as possible
root=bisection(func,t(round(numel(t)/2)),t(end),func(t(2)),func(t(end)),0,0,100000,0);

% Plot impact point.
impact=plot(x(root), y(root), 'k*');

%Legends for lines
legend([shell,hill,impact],'Shell','Hill','Impact');
ylabel('height (meters)'); xlabel('distance (meters)');
% Turn of the major grid lines and set the axis
grid ON; axis([0 16000 -2000 10500]); grid MINOR;
```

5.2 Result from MyEvent.m

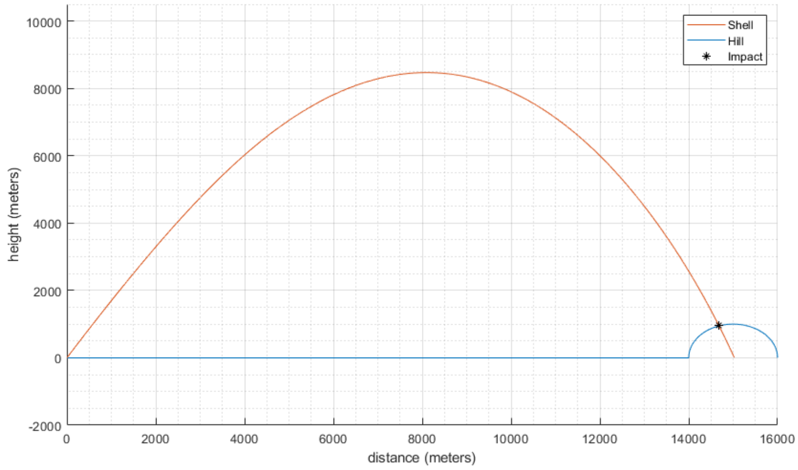

Figure 6: Simulation of an artillery being shot at a hill and calculating the root for impact.

References

[1] CARL CHRISTIAN KJELGAARD MIKKELSEN. *An Introduction to Scientific Computing*. Department of Computing Science, Umeå University.