Project 2: Runge-Kutta Methods

Cameron Weigel

May 6, 2016

1 Analysis

Runge-Kutta of Order Four and Runge-Kutta Fehlberg were used to approximate the initial-value problem

$$y' = (-1/25)(t-2)^3y^2; \quad y(-a) = \frac{1}{((-a-2)^4+1)}; \quad t \in [-10, 10]$$

Since the solution to this initial-value problem has a sharp increase around $t = 2$ a standard Runge-Kutta method with a fixed step size could produce a bad approximation to the actual solution, especially around $t = 2$. To remedy this, we developed an algorithm that would vary the step-size based on the difference between two Runge-Kutta order four approximations. This allowed us to better approximate the solution at the cost of a longer run-time. While the run-time was longer, having a bad approximation to the solution is of no use to us. I predicted that as we decreased the tolerance of error between y_1 and y_2, our approximation would be more accurate. As you will see, the approximations using $\epsilon = 10^{-7}$ are not as accurate as the approximations using $\epsilon = 10^{-11}$.

2 Code

This section contains the code used to approximate the soluton to the initial-value problem. Some formatting has been changed when transferring the text. Comments are in blue and code is in black.

2.1 Runge-Kutta Order 4

Applies one iteration of Runge-Kutta order four with step-size h

```
function [y] = RKS1(t,y,h)


K1 = h*externalf(t,y);
```

K2 = h*externalf(t+h/2,y+K1/2);
K3 = h*externalf(t+h/2,y+K2/2);
K4 = h*externalf(t+h,y+K3);

y = y + (K1+2*K2+2*K3+K4)/6;

end

2.2 Runge-Kutta Error Check

Subroutine to check the difference between y1 and y2 and change h and t based on the tolerance provided from the Runge-Kutta subroutine

INPUTS: t1 = interval point in [a,b]
y = Value to approximate using Runge-Kutta methods
h1 = step size
e = tolerance of error

OUTPUTS: t = changed or unchanged t based on the tolerance check
h = updated h based on the tolerance check
y = updated y based on tolerance check

function [t,h,y] = RKE1(t1,y,h1,e)

format long;

Tolerance is built from a Taylor Series difference

y1 = RKS1(t1,y,h1);
ymid = RKS1(t1,y,h1/2);
y2 = RKS1(t1+h1/2,ymid,h1/2);

while (abs(y1-y2) <15.*abs(h1)*e)

h1 = .8.*(((15.*abs(h1).*e)/(abs(y1-y2))).^(1/4)).*h1;

y1 = RKS1(t1,y,h1);
ymid = RKS1(t1,y,h1/2);

y2 = RKS1(t1+h1/2,ymid,h1/2);

end

t = t1+h1;

h1 = .8.*(((15.*abs(h1).*e)/(abs(y1-y2))).^(1/4)).*h1;

h = h1;

y = y2;

end

2.3 Runge-Kutta approximations

This subroutine loops over the interval [-10,10] and applies the subroutines RKE1.m to approximate the differental equation provided in externalf.m

INPUTS: t = start interval t = -10 for this project

y = initial condition of the IVP

h = intial step-size guess

e = tolerance used to change h in RKE

function [ts ys hs] = RKK1(t,y,h,e)

format long;

i = 1;

This while loop will only save values of t and y that are accepted by RKE1

while (t <10)

[t,h,y] = RKE1(t,y,h,e);

TS(i) = t;
YS(i) = y;
HS(i) = h;
i = i + 1;
end

ts = TS;
ys = YS;
hs = HS;
end

2.4 Runge-Kutta Fehlberg

Runge-Kutta Fehlberg Method

INPUTS: t = mesh point in the interval [a,b] y = y value associated with t h1 = step size to apply RKF

OUTPUTS: y = RKF approximation

function [y] = RKF(t,y,h1)

format long;

K1 = h1*externalf(t,y);
K2 = h1*externalf(t+h1/4,y+K1/4);
K3 = h1*externalf(t+(3*h1)/8,y+(3/32)*K1+(9/32)*K2);
K4 = h1*externalf(t+(12/13)*h1,y+(1932/2197)*K1-(7200/2197)*K2+(7296/2197)*K3);
K5 = h1*externalf(t+h1,y+(439/216)*K1-8*K2+(3880/2565)*K3-(845/4104)*K4);
K6 = h1*externalf(t+h1/2,y-(8/27)*K1+2*K2-(3544/2565)*K3+(1859/4104)*K4-(11/40)*K5);
y2 = y + ((16/135)*K1+(6656/12825)*K3+(28561/56430)*K4-(9/50)*K5+(2/55)*K6);

end

2.5 Runge-Kutta Fehlberg Error Check

Subroutine to check the difference between w1 and w2 and change h and t based on the tolerance provided from the Runge-Kutta subroutine

INPUTS: t1 = interval point in [a,b]
y = value to run Runge-Kutta approximation methods
h1 = step size
e = tolerance of error

OUTPUTS: t = changed or unchanged t based on the tolerance check
h = updated h based on the tolerance check
w = updated Runge-Kutta value

function [t,h,w] = RKE2(t1,y,h1,e)

format long;

Tolerance is built from a Taylor Series difference

w1 = RKS1(t1,y,h1);
w2 = RKF(t1,y,h1);

while (abs(w1-w2) <15.*abs(h1)*e)

h1 = .8.*(((15.*abs(h1).*e)/(abs(w1-w2))).^(1/4)).*h1;

w1 = RKS1(t1,y,h1);
w2 = RKF(t1,y,h1);

end

t = t1+h1;

h1 = .8.*(((15.*abs(h1).*e)/(abs(w1-w2))).^(1/4)).*h1;
h = h1;
w = w2;

end

2.6 Runge-Kutta Fehlberg Approximations

This subroutine loops over the interval [-10,10] and applies the subroutines RKE2.m to approximate the differental equation provided in externalf.m

INPUTS: t = start interval t = -10 for this project
y = initial condition of the IVP
h = intial step-size guess
e = tolerance used to change h in RKE

function [ts ys hs] = RKK2(t,y,h,e)

format long;

i = 1;

while (t <10)

[t,h,y] = RKE2(t,y,h,e);

TS(i) = t;
YS(i) = y;
HS(i) = h;
i = i + 1;

end

ts = TS;
ys = YS;
hs = HS;

end

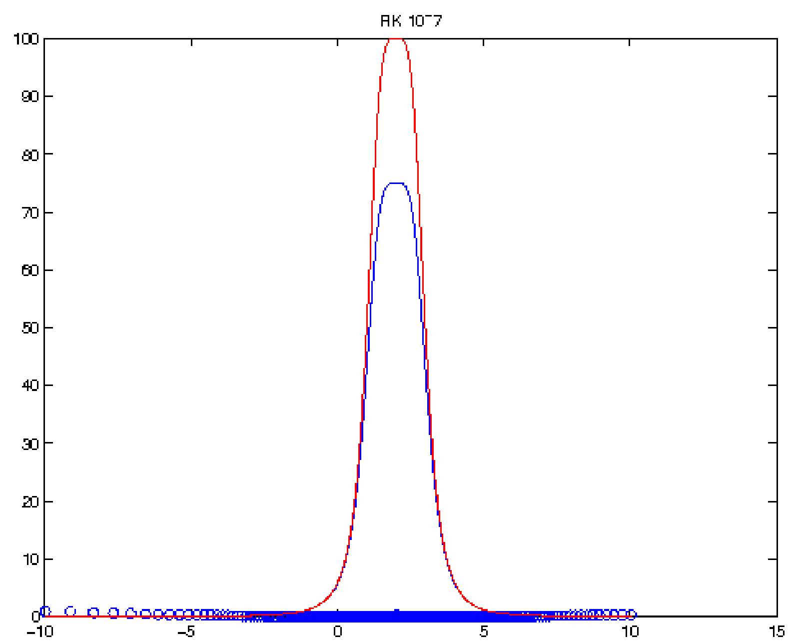
3 Results

The results were of no surprise, as ϵ shrank, the approximations to the actual solution increased. In the case of $\epsilon = 10^{-}11$ the actual solution and the approximation were indistinguishable on the graphs.

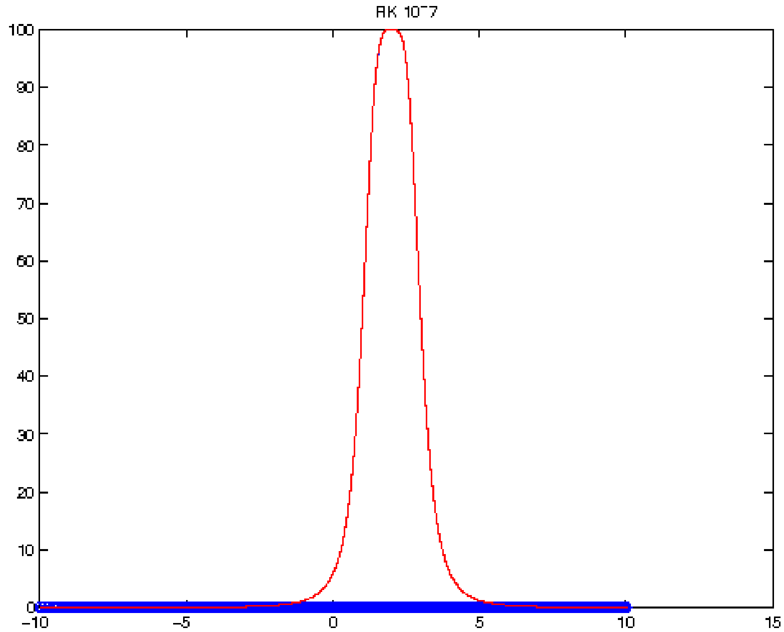
3.1 Graphs

The approximation of the solution is the blue line, the blue circles are the step-sizes and the red line is the actual solution

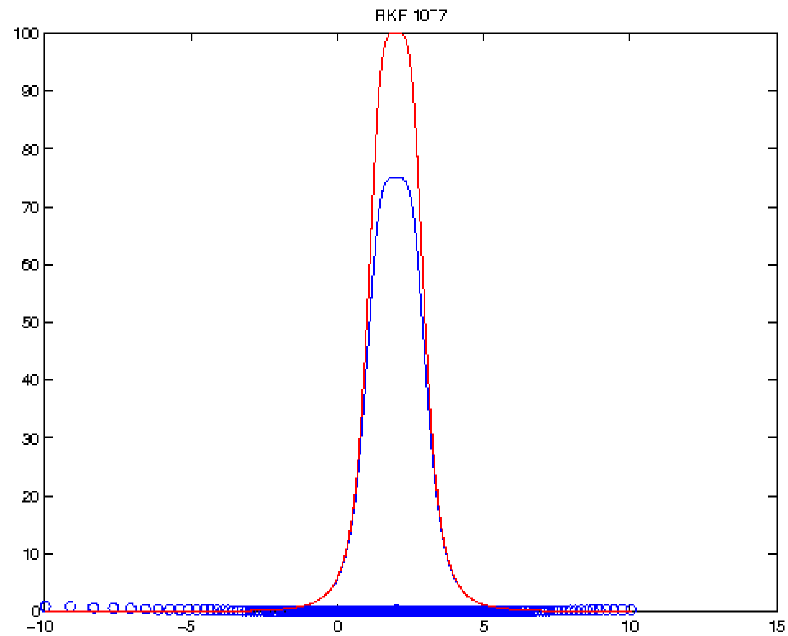
Runge-Kutta Method with $\epsilon = 10^{-7}$



Runge-Kutta Method with $\epsilon = 10^{-11}$



Runge-Kutta Fehlberg Method with $\epsilon = 10^{-7}$



Runge-Kutta Fehlberg Method with $\epsilon = 10^{-11}$