CRANFIELD UNIVERSITY

Mateusz Golab

Forward integration of simultaneous ordinary differential equations with graphical output

School of Engineering

Software Engineering for Technical Computing

MSc THESIS

Academic Year: 2011 - 2012

Supervisor: Dr Peter Sherar, Prof Joanna Polanska

August 2012

CRANFIELD UNIVERSITY

School of Engineering

Software Engineering for Technical Computing

MSc THESIS

Academic Year 2010 - 2011

Mateusz Golab

Forward integration of simultaneous ordinary differential equations with graphical output

Supervisor: Dr Peter Sherar, Prof Joanna Polanska

August 2012

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science

This thesis is submitted in accordance with the Double Degree programme regulations. Home institution : Silesian University of Technology, Poland

© Cranfield University 2012. All rights reserved. No part of this publication may be reproduced without the written permission of the copyright owner.

ABSTRACT

Click here to enter abstract text

Keywords:

ODE, Runge-Kutta, Modified Midpoint, Predictor - Corrector, Web development, GWT, AppEngine, Datastore, Unit testing.

ACKNOWLEDGEMENTS

Click here to enter acknowledgement text

TABLE OF CONTENTS

[ABSTRACT i](#_Toc331630162)

[ACKNOWLEDGEMENTS ii](#_Toc331630163)

[LIST OF FIGURES v](#_Toc331630164)

[LIST OF TABLES vi](#_Toc331630165)

[LIST OF EQUATIONS vii](#_Toc331630166)

[LIST OF ABBREVIATIONS viii](#_Toc331630167)

[1 Introduction 1](#_Toc331630168)

[1.1 Aims and objectives 1](#_Toc331630169)

[1.2 Motivation 1](#_Toc331630170)

[2 Literature review 2](#_Toc331630171)

[2.1 ODE numerical routines 2](#_Toc331630172)

[2.1.1 Runge-Kutta methods 2](#_Toc331630173)

[2.1.2 Modified midpoint 4](#_Toc331630174)

[2.1.3 Richardson extrapolation 5](#_Toc331630175)

[2.1.4 Rosenbrock 6](#_Toc331630176)

[2.1.5 Predictor- Corrector 6](#_Toc331630177)

[2.2 Technologies 6](#_Toc331630178)

[2.2.1 AJAX approach 6](#_Toc331630179)

[2.2.2 Google Web Toolkit 6](#_Toc331630180)

[2.2.3 AppEngine 6](#_Toc331630181)

[2.2.4 Datastore 6](#_Toc331630182)

[3 Methodologies chosen 7](#_Toc331630183)

[3.1 Prototyping 7](#_Toc331630184)

[3.2 Test Driven Development 7](#_Toc331630185)

[3.3 AJAX 7](#_Toc331630186)

[3.4 Versioning 7](#_Toc331630187)

[4 Design 8](#_Toc331630188)

[4.1 Architecture 8](#_Toc331630189)

[4.2 Design patterns 8](#_Toc331630190)

[4.3 Technologies applied 8](#_Toc331630191)

[5 Testing 9](#_Toc331630192)

[5.1 Test driven development approach 9](#_Toc331630193)

[5.2 Testing methods 9](#_Toc331630194)

[5.2.1 Unit testing 9](#_Toc331630195)

[5.2.2 Integration testing 9](#_Toc331630196)

[5.2.3 System testing 9](#_Toc331630197)

[5.2.4 Cross – browser testing 9](#_Toc331630198)

[6 Implementation 10](#_Toc331630199)

[6.1 Parser 10](#_Toc331630200)

[6.2 Solver 10](#_Toc331630201)

[6.3 Graph viewer 10](#_Toc331630202)

[6.4 Datastore connector 10](#_Toc331630203)

[7 Results 11](#_Toc331630204)

[7.1 Results validation and verification 11](#_Toc331630205)

[7.2 Application’s outputs 11](#_Toc331630206)

[8 Discussion and conclusion 12](#_Toc331630207)

[8.1 Solvers correctness 12](#_Toc331630208)

[8.2 Limitations 12](#_Toc331630209)

[8.3 Problems faced 12](#_Toc331630210)

[8.4 Quality of implementation 12](#_Toc331630211)

[8.5 Conclusion 12](#_Toc331630212)

[8.6 Future work 12](#_Toc331630213)

[REFERENCES 14](#_Toc331630214)

[APPENDICES 15](#_Toc331630215)

# LIST OF FIGURES

[Figure 1 4th Order Runge-Kutta method 3](#_Toc331630216)

LIST OF TABLES

**Nie można odnaleźć pozycji dla spisu ilustracji.**

LIST OF EQUATIONS

[(2‑1) 2](#_Toc331630217)

[(2‑2) 3](#_Toc331630218)

LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| CU  GWT  ODE | Cranfield University  Google Web Toolkit  Ordinary differential equation |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

The Introduction chapter points aims and objectives of the thesis project. All requirements of the project are explained in this chapter. Moreover, main motivators to perform this specific topic are included as well.

## Aims and objectives

The main objective of this thesis project is to develop following application :

1. Parsing equations entered by the user
2. Solving series of 1st order simultaneous ordinary differential equations (linear or non-linear) for initial value problem.
3. Presenting the solution on the 2D graph.
4. Storing and loading equations along with parameters entered by the user.

## Motivation

In general, two main motivators of this project are :

1. Discovering most efficient numerical methods for solving series of ordinary differential equations.
2. Familiarise with AJAX applications development and Google AppEngine along with Datastore.

# Literature review

This chapter demonstrates knowledge about recommended practical numerical methods for solving ODEs. Presented literature review gives an insight into major types of numerical routines taking into account efficiency and accuracy. What is more, technologies supporting AJAX applications development along with App Engine are also presented in this chapter. It is important to perform detailed research about technologies essential to develop the project.

## ODE numerical routines

Describes types of practical numerical methods for solving ODEs including Runge-Kutta methods, Bulirsch-Stoer, Rosenbrock metods and predictor-corrector rmethods.

### Runge-Kutta methods

Runge-Kutta methods propagate a solution over an interval by combining the data from several Euler-style steps. Each step involves one evaluation of the right-hand of the function .

|  |  |
| --- | --- |
|  | (2‑) |

Thenusing the information obtained to match a Taylor series expansion up to higher order.

Developing higher – order methods made Runge-Kutta competitive with the other numerical methods in many cases. It is usually the fastest method when moderate accuracy is required (≤ ) and evaluation of the function is not too expensive. There are few kinds of Runge-Kutta methods : 2nd order method, (called midpoint method), 4th order method and also method with adaptive stepsize. [Numerical recipes]

#### 4th Order Runge-Kutta

The most often used Runge-Kutta method is fourth – order formula . In general it is superior to 2nd order method, however high order does not always mean high accuracy. 4th order method requires four evaluations of the function. [numerical recipes]

|  |  |
| --- | --- |
|  | (‑) |

|  |  |
| --- | --- |
|  | (2‑3) |

|  |  |
| --- | --- |
|  | (2‑4) |

|  |  |
| --- | --- |
|  | (2‑5) |

|  |  |
| --- | --- |
|  | (2‑6) |



Figure 4th Order Runge-Kutta method

During single step, the derivative is evaluated four times. Once at initial point, (1) twice at midpoints (2)(3) and once at trial endpoint (4). Final function value is calculated on the basis of these derivatives. [numerical recipes]

Each step in the sequence of steps is treated in an identical manner, so prior behaviour of the solution is not used in its propagation. Such approach is mathematically proper, since any point along the trajectory of an ODE can be an initial point. [numerical recipes]

#### Runge-Kutta with adaptive stepsize

The purpose of adaptive stepsize method is to achieve predetermined accuracy in the solution with minimum computational effort. It is possible to face very smooth interval , while performing Runge-Kutta steps. Few great strides instead of small steps should speed through such undifferentiated interval, what may result in significant gain in efficiency. The idea of adaptive stepsize method is to control the size of the step and increase it when possible maintaining required level of accuracy. It is important to estimate truncation error to control accuracy level while increasing step size. Obviously the calculation of this information will add to the computational overhead, however it is profitable investment in terms of efficiency.

### Modified midpoint

Modified midpoint method is a second order method like 2nd order Runge-Kutta, however with the advantage of requiring only one derivative evaluation per single step instead of two evaluations present in Runge-Kutta. This method generates the solution as a vector of values from a point  to a point by a sequence of substeps each of size . Where

|  |  |
| --- | --- |
|  | (2‑7) |

The total number of function evaluations required by this method is . The formulas essential to provide the solution for are as follows

|  |  |
| --- | --- |
|  | (2‑8) |

|  |  |
| --- | --- |
|  | (2‑9) |

|  |  |
| --- | --- |
|  | (2‑10) |

|  |  |
| --- | --- |
|  | (2‑11) |

The is the final approximation to whereas represents intermediate approximations calculated along in steps of .

### Richardson extrapolation

Richardson extrapolation bases on idea of extrapolating a computed value to the value that would have been obtained if the stepsize had been remarkably smaller than it actually was. The practical numerical method using this idea is called Bulirsch-Stoer method.

#### Burlisch – Stoer

The idea of Burlisch-Stoer method is to perform iterations of modified midpoint method . Each iteration uses different number of substepsfor modified midpoint method ends up with polynomial extrapolation of the given values. Burlish and Stoer originally proposed following sequence of substeps :

|  |  |
| --- | --- |
|  | (2‑12) |

However sequence discovered by Deuflhard is usually more efficient :

|  |  |
| --- | --- |
|  | (2‑13) |

In terms of number of iterations usually 8 gives satisfactory results.



Figure Richardson extrapolation used in the Burlisch-Stoer method with substep n = 2,4,6

Extrapolation of the each iteration results

### Rosenbrock

### Predictor- Corrector

#### Adams-Bashforth-Moulton

## Technologies

### AJAX approach

### Google Web Toolkit

### AppEngine

### Datastore

# Methodologies chosen

## Prototyping

## Test Driven Development

## AJAX

## Versioning

# Design

## Architecture

## Design patterns

## Technologies applied

# Testing

## Test driven development approach

## Testing methods

### Unit testing

### Integration testing

### System testing

### Cross – browser testing

# Implementation

## Parser

## Solver

## Graph viewer

## Datastore connector

# 

# Results

## Results validation and verification

## Application’s outputs

# Discussion and conclusion

## Solvers correctness

## Limitations

## Problems faced

## Quality of implementation

## Conclusion

## Future work

REFERENCES

Insert list of references here

APPENDICES

Whilst Heading 1 to Heading 6 can be used to number headings in the main body of the thesis, Heading styles 7–9 have been modified specifically for lettered appendix headings with Heading 7 having the ‘Appendix’ prefix as shown below.

Appendix Title (Use Heading 7)

Appendix Section (Use Heading 8)

Appendix Subsection (Use Heading 9)

Creating captions in Appendices

If you have chosen to include chapter numbers in your captions then follow the instructions given here to apply the same format to the captions in your appendices. This section explains how to caption the figures and tables in your Appendices, assuming that Heading 7 is numbered “Appendix A” and that the Figures and Tables are going to be labelled ‘Figure A-1’, ‘Figure A-2’, ‘Table B-1’ etc.

You will have to create new, separate labels that look like the ‘Figure’ and ‘Table’ labels you used in the main body of your thesis.

1. Select the **References** tab on the Ribbon then click on **Insert Caption**
2. Click **New Label**. Type **Figure\_Apx** then click **OK**
3. You now have two labels for figures, called **Figure** and **Figure\_Apx**  
   Repeat for table captions.
4. In the **Caption** box, type your caption text
5. Click **Numbering**. Tick **Include chapter numbering** and choose **Heading 7** from the drop-down list of styles and click **OK** twice
6. Your caption should look something like this:

**Figure\_Apx A‑1 This is the caption text for a Figure in the Appendix**

1. Delete the extraneous ‘\_Apx’ from the caption label so it reads:  
   **Figure A‑1 This is the caption text for a Figure in the Appendix**  
   **TIP:** Instead of deleting each ‘\_Apx’ individually use **Find & Replace** to modify all the labels at once.

Creating Lists of Figures and Tables for Appendices

This template already includes a List of Figures and a List of Tables, however you will have to create two new lists for the ‘Figure\_Apx’ and the ‘Table\_Apx’ labels.

1. Place the insertion point on a blank row after the existing List of Figures
2. Select the **Insert Table of Figures** command on the **References** tab of the Ribbon
3. Set the **Caption Label** box to ‘**Figure\_Apx**’ and click **OK**  
   **Note:** Word will put a single blank line between the original and new lists preventing it from appearing as one seamless list. However if you select the blank paragraph between the tables you can hide it by opening the Font dialog box from the Home tab and selecting **Hidden**.
4. Click after the List of Tables and repeat for the Caption Label ‘Table\_Apx’