



MASTERARBEIT

BENCHMARKS FOR MECHATRONIC MODELS

Ausgeführt am Institut für Analysis and Scientific Computing,

Mathematical Modelling and Simulation

Technische Universität Wien

unter der Anleitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker

durch

Gemma Ferdinand Kaunang

Brigittenauer Lände 224/6331

1200 Vienna

Austria

Wien, 2 July 2008

***An educated person is one who has learned that information
almost always turns out to be at best incomplete and
very often false, misleading, fictitious, mendacious
- just dead wrong.***

RUSSEL BAKER

My Gratitude to:

Father, Mother, brothers and my girlfriend

Abstract

Before a new product is launched to the market, a company have to test the product and make sure that the product is ready and presentable, especially if the product is a problem solving system, such as control system, automated system or self-learning system. Testing is a highly cost consuming yet unavoidable activity. Therefore to reduce cost of production, company will use methods like simulation to test their product.

There are so many simulation softwares in the market which offers different abilities and advantages. The various choices has makes it even more difficult for end-users (company) to choose which one is more suitable and useful for the company. On this thesis Three “comparison problem” based on electrotechnic will be compared each other by using four simulations software (Matlab/Simulink, Dymola, Mosilab and SimulationX), with different approaches to model of the system.

The method used for this research is a literature study to have a deeper understanding about the behaviour and algorithm of the code from 4 different simulation softwares, the design model of three comparison problems and simulates these models to find the most suitable plot result.

After a thorough research of these three comparison problems, conclusion can be made as follow:

- Matlab is the only simulation software which able to simulate all calculation of matrix.
- Stateflow, stategraph and statechart which can only model the equation with switching state, the harder the equation is, the longer time required to simulate the equation.
- Dymola has the most variation type of modelling that needed in this thesis, the fastest time simulation is by dymola electrical model to simulate task d in comparison 3 = 0,015s,
- SimulationX took the longest time in simulating task diode C in comparison 20 = 1307,6718s, Type of designer block in simulationX is very useful feature for expert user in defining their code in new element type
- Below are the simulation timing ranking from fastest to slowest type of modelling:
 - a. Textual mode
 - b. Electrical model
 - c. Hybrid model
 - d. Stateflow/stategraph/statechart model

Prologue

Praise to the amorously and merciful Lord, because of His strength, love and affection, this master thesis, titled “Benchmarks for Mechatronic Models”, can be completed.

In this thesis, writer have tried to compare the simulation result between three comparison problems by using 4 different simulation softwares, which are Matlab/Simulink, Dymola, Mosilab and SimulationX with different approach methods in designing the model of these three comparison problems.

As there are many people who have generously contributed their time, knowledge and support during the process of this thesis, I would like to express my gratitude to:

1. Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breiteneker, as counsellor of this thesis, for providing dymola and mosilab software, his help and support during this research from beginning to the end.
2. Univ.Ass. Dipl.-Ing. Mag.rer.soc.oec. Aman Atri and Nicole Nagele, for their support helping me with mosilab installation in linux operating system, also problem during compiling problem in mosilab.
3. Dipl.-Ing. Günther Zauner, for his help in giving me clues, tips and suggestions about my problem in modelica language, mosilab and matlab.
4. Andreas Körner, for his help in providing me with the explanation on comparison 20, electrical model - basics.
5. Jacek Kierzenka and someone with nickname “helper” from matlab central newsreader, for their help, giving me clue, tips and suggestion about my problem in matlab.
6. Antje Heimann, Torsten Blochwitz, Matthias Illschner, Ulrich Behnert and Gerd Kurzbach from ITI Team for providing simulationX software, license file and their patient and help in answering my endless questions about simulationX.

7. Father, mother and my two brothers at home in Jakarta, Indonesia, for their continuous support and prayer for me, I am forever grateful.
8. My beloved girlfriend in Singapore, for her help, support, prayer and for boosting my spirit during this research.
9. To all my friend in Vienna, Austria for their help and support.
10. To anyone who has contributed their effort, time and support for me during this research, my deepest appreciation.

Eventhough I realize that there are many area in this thesis, which are still open for further study, I sincerely hope that this research can be used as a contribution for future academic reference.

Thank you for your time and attention.

Vienna, 2 July 2008

Gemma Ferdinand Kaunang

CONTENT

PROLOGUE	i
CONTENT	iii
FIGURE LIST	viii
TABLE LIST	xiii
1 INTRODUCTION	
1.1 Background	1
1.2 Problem's Formulation	2
1.3 Concluded Area	2
1.4 Objective.....	3
1.5 Research Plan	3
1.5.1 Study Literature	3
1.5.2 Design and Solutions	3
1.6 Writing Methods.....	3
2. THEORY	
2.1 Matlab	5
2.1.1 The Matlab System.....	5
2.1.2 ODE Solver	7
2.1.2.1 ODE File	7
2.1.2.2 Odeset.....	8
2.1.2.3 Initial Value ODE Problem Solvers.....	9
2.2 Simulink.....	9
2.2.1 States.....	9
2.2.1.1 Continuous States	11
2.2.1.2 Discrete States.....	12
2.2.1.4 Modeling Hybrid Systems	13
2.3 Stateflow	13
2.3.1 Finite State Machine Concepts.....	14
2.3.2 Statflow Notations.....	15
2.3.2.1 Nongraphical Objects	15
2.3.2.2 Graphical Objects	16

2.4 Dymola	16
2.4.1 Architecture of Dymola	17
2.4.2 Basic Operations of Dymola	18
2.5 Mosilab	19
2.5.1 Mosilab Architecture	19
2.5.2 Mosilab Configurations	19
2.5.3 The Mosilab Development Environment	20
2.6 SimulationX	21
2.6.1 Library Bar	23
2.6.2 Model View	23
3. Comparison 3: Generalized Class-E Amplifier	
3.1 Definition	24
3.2 Design and Solutions	26
3.2.1 Matlab	26
3.2.2 Simulink	29
3.2.2.1 Hybrid Model	29
3.2.2.2 Stateflow	32
3.2.3 Dymola	35
3.2.3.1 Hybrid Model	35
3.2.3.2 Stategraph Model	38
3.2.3.3 Electrical Model	39
3.2.3.4 Modelica Text Mode	40
3.2.4 Mosilab	42
3.2.4.1 Modelica Text Mode	42
3.2.4.2 StateChart	43
3.2.5 SimulationX	44
3.2.5.1 Hybrid Model	44
3.2.5.2 Electrical Model	47
4. Comparison 5: Two State Model	
4.1 Definition	49
4.2 Design and Solutions	50

4.2.1 Matlab	51
4.2.2 Simulink	54
4.2.2.1 Hybrid Model	54
4.2.2.2 Stateflow	58
4.2.3 Dymola	62
4.2.3.1 Hybrid Model	62
4.2.3.2 Stategraph Model	65
4.2.3.3 Modelica Text Mode	67
4.2.4 Mosilab	69
4.2.4.1 Modelica Text Mode	69
4.2.4.2 StateChart	71
4.2.5 SimulationX	73
5. Comparison 20: Electrical model - Basics	
5.1 Definition	77
5.2 Tasks	78
5.2.1 Steady States	78
5.2.2 Classical Simulation	78
5.2.3 Different Diode Models	79
5.2.4 Influence of Simulation Algorithms	80
5.3 Design and Solutions	80
5.3.1 Matlab	80
5.3.1.1 Steady States	80
5.3.1.2 Classical Simulation	82
5.3.1.3 Different Diode Models	84
5.3.1.4 Influence of Simulation Algorithms	85
5.3.2 Simulink	87
5.3.2.1 Hybrid Model	87
5.3.2.1 Steady States	87
5.3.2.2 Classical Simulation	90
5.3.2.3 Different Diode Models	93
5.3.2.4 Influence of Simulation Algorithms	94

5.3.2.2 Stateflow	94
5.3.2.2.1 Classical Simulation	95
5.3.3 Dymola	96
5.3.3.1 Hybrid Model	96
5.3.3.1.1 Steady States	96
5.3.3.1.2 Classical Simulation	98
5.3.3.1.3 Different Diode Models	101
5.3.3.1.4 Influence of Simulation Algorithms	102
5.3.3.2 Stategraph Model	104
5.3.3.2.1 Steady States (State C)	104
5.3.3.2.2 Classical Simulation	105
5.3.3.2.3 Different Diode Models	107
5.3.3.3 Electrical Model	108
5.3.3.3.1 Steady States	108
5.3.3.3.2 Classical Simulation	109
5.3.3.3.3 Different Diode Models	110
5.3.3.3.4 Influence of Simulation Algorithms	112
5.3.3.4 Modelica Text Mode	113
5.3.3.4.1 Steady States	113
5.3.3.4.2 Classical Simulation	114
5.3.3.4.3 Different Diode Models	114
5.3.3.4.4 Influence of Simulation Algorithms	115
5.3.4 Mosilab	116
5.3.4.1 Modelica Text Mode	116
5.3.4.1.1 Steady States	116
5.3.4.1.2 Classical Simulation	118
5.3.4.1.3 Different Diode Models	119
5.3.4.1.4 Influence of Simulation Algorithms	121
5.3.4.2 StateChart	121
5.3.4.2.1 Steady States	122
5.3.4.2.2 Classical Simulation	123

5.3.4.2.3 Different Diode Models.....	124
5.3.5 SimulationX.....	124
5.3.5.1 Hybrid Model.....	124
5.3.5.1.1 Steady States.....	124
5.3.5.1.2 Classical Simulation.....	127
5.3.5.1.3 Different Diode Models.....	128
5.3.5.1.4 Influence of Simulation Algorithms	131
5.3.5.2 Electrical Model	132
5.3.5.2.1 Steady States.....	132
5.3.5.2.2 Classical Simulation.....	133
5.3.5.2.3 Different Diode Models.....	134
5.3.5.2.4 Influence of Simulation Algorithms	135
6. Comparison	
6.1 Table of Result	137
6.1.1 Comparison 3	137
6.1.2 Comparison 5	138
6.1.3 Comparison 20.....	140
6.2 Advantage and Disadvantage.....	143
6.2.1 Matlab.....	143
6.2.2 Simulink	144
6.2.3 Dymola	144
6.2.4 Mosilab	145
6.2.5 SimulationX.....	146
7. Conclusion and Suggestion	
7.1 Conclusion	148
7.2 Suggestion.....	150
REFERENCE	152
APPENDIX	
Source Code	154

FIGURE LIST

Figure 2.1 Screenshot of Matlab	6
Figure 2.2 Screenshot of Simulink	10
Figure 2.3 Block have States.....	11
Figure 2.4 Screenshot of Stateflow	14
Figure 2.5 Graphical objects in Stateflow.....	15
Figure 2.6 Architecture of Dymola	17
Figure 2.7 Screenshot of Dymola	17
Figure 2.8 Simulation Mode of Dymola	18
Figure 2.9 Data Flow within Stateflow.....	19
Figure 2.10 Screenshot of Prototypical Implementation of the MOSILAB- IDE	21
Figure 2.11 Screenshot of SimulationX	22
Figure 2.12 Components in SimulationX	23
Figure 3.1 Class-E Amplifier	24
Figure 3.2 Function of time $R(t)$	25
Figure 3.3 Block Diagram of Comparison 3.....	26
Figure 3.4 The result for variable current switch resistor $IR(t)$ and output voltage $V_L(\text{matlab})$	28
Figure 3.5 The phase plane curves $dx_3/dt=V_L$ as a function of $x_3 =$ IL_3 for TRF (matlab)	29
Figure 3.6 Model of the system using Simulink	30
Figure 3.7 The result for variable current switch resistor $IR(t)$ and output voltage V_L (simulink).....	31
Figure 3.8 The phase plane curves $dx_3/dt=V_L$ as a function of $x_3 =$ IL_3 for TRF (simulink).....	31
Figure 3.9 Stateflow of the system.....	33
Figure 3.10 Model of the system using Simulink (stateflow).....	33
Figure 3.11 The result for variable current switch resistor $IR(t)$ and output voltage V_L (simulink stateflow).....	34
Figure 3.12 Trapezoid Source	35

Figure 3.13 The Model of the system using Dymola.....	35
Figure 3.14 The result for variable current switch resistor IR(t) and output voltage VL (dymola)	36
Figure 3.15 The phase plane curves $dx_3/dt=VL_3$ as a function of $x_3 =$ IL3 for TRF (dymola)	37
Figure 3.16 Trigerred Trapezoid.....	38
Figure 3.17 Part Controller of the System (stategraph)	38
Figure 3.18 The Model of the system using Dymola stategraph mode	38
Figure 3.19 Electrical Model for Comparison 3.....	40
Figure 3.20 The result for variable current switch resistor IR(t) and output voltage VL (mosilab)	42
Figure 3.21 The model of the system (simulationX).....	45
Figure 3.22 The result for variable current switch resistor IR(t) and output voltage VL (simulationX)	46
Figure 3.23 The phase plane curves $dx_3/dt=VL_3$ as a function of $x_3 =$ IL3 for TRF (simulationX)	46
Figure 3.24. The model of the system (simulationX electrical)	48
Figure 4.1 Block Diagram Comparison 5.....	50
Figure 4.2 Plot y1(matlab)	52
Figure 4.3 Plot y1 (task d) (matlab)	53
Figure 4.4 The model of the system (simulink).....	55
Figure 4.5 Plot y1(simulink)	56
Figure 4.6 Plot y1 (task d) (simulink).....	57
Figure 4.7 Stateflow block of the system (comparison 5)	58
Figure 4.8 The model of the system (simulink stateflow).....	59
Figure 4.9 Plot y1(simulink stateflow)	60
Figure 4.10 Plot y1 (task d) (simulink stateflow).....	61
Figure 4.11 The model of the system (dymola)	62
Figure 4.12 Plot y1(dymola)	63
Figure 4.13 Plot y1 (task d) (dymola)	64

Figure 4.14 Part Controller and Switching State of the System (dymola stategraph).....	66
Figure 4.15 The model of the system (dymola stategraph)	66
Figure 4.16 Plot y1(mosilab)	69
Figure 4.17 Plot y1 (task d) (mosilab).....	70
Figure 4.18 The model of the system (simulationX).....	73
Figure 4.19 Plot y1(simulationX)	74
Figure 4.20 Plot y1 (task d) (simulationX).....	76
Figure 5.1 Electrical circuit comparison 20	77
Figure 5.2 Steady States	78
Figure 5.3 Time Dependent S1	79
Figure 5.4 Plot x1 and x2 steady states (matlab)	82
Figure 5.5 Plot x1 and x2 classical simulation (matlab)	83
Figure 5.6 Plot x1 and x2 different diode models (matlab)	85
Figure 5.7 Plot x1 and x2 influence of simulation algorithms (matlab)	86
Figure 5.8 Model of the system steady states (simulink)	88
Figure 5.9 Plot x1 and x2 steady states (simulink).....	89
Figure 5.10 Model of the system classical simulation (simulink)	90
Figure 5.11 Plot x1 and x2 classical simulation (simulink).....	90
Figure 5.12 Model of the system different diode models (simulink)	91
Figure 5.13 Plot x1 and x2 different diode models (simulink)	92
Figure 5.14 Model of the system influence of simulation algorithms (simulink)	93
Figure 5.15 Plot x1 and x2 influence of simulation algorithms (simulink) ..	94
Figure 5.16 Model of the system classical simulation (simulink stateflow)95	
Figure 5.17 Plot x1 and x2 classical simulation (simulink stateflow).....	96
Figure 5.18 Model of the system steady states (dymola).....	97
Figure 5.19 Plot x1 and x2 steady states (dymola)	98
Figure 5.20 Model of the system classical simulation (dymola).....	99
Figure 5.21 Plot x1 and x2 classical simulation (dymola)	99
Figure 5.22 Model of the system different diode models (dymola)	100

Figure 5.23 Plot x1 and x2 different diode models (dymola).....	101
Figure 5.24 Model of the system influence of simulation algorithms (dymola).....	102
Figure 5.25 Plot x1 and x2 influence of simulation algorithms (dymola) ..	103
Figure 5.26 Model of the system steady states (dymola stategraph).....	104
Figure 5.27 Plot x1 and x2 steady states (dymola stategraph).....	105
Figure 5.28 Model of the system classical simulation (dymola stategraph)	106
Figure 5.29 Model of the system different diode models (dymola stategraph)	106
Figure 5.30 Plot x1 and x2 different diode models (dymola stategraph) ..	108
Figure 5.31 Model of the system steady states (dymola electrical)	109
Figure 5.32 Model of the system classical simulation (dymola electrical)	110
Figure 5.33 Model of the system different diode models (dymola electrical)	111
Figure 5.34 Plot x1 and x2 different diode models (dymola electrical)	111
Figure 5.35 Model of the system influence of simulation algorithms (dymola electrical).....	112
Figure 5.36 Plot x1 and x2 influence of simulation algorithms (dymola electrical)	112
Figure 5.37 Plot x1 and x2 steady states (mosilab).....	117
Figure 5.38 Plot x1 and x2 classical simulation (mosilab).....	118
Figure 5.39 Plot x1 and x2 different diode models (mosilab)	120
Figure 5.40 Plot x1 and x2 influence of simulation algorithms (mosilab) ..	121
Figure 5.41 Plot x1 and x2 steady states (mosilab statechart)	122
Figure 5.42 Plot x1 and x2 different diode models (mosilab statechart) ..	123
Figure 5.43 Model of the system steady states (simulationX)	125
Figure 5.44 Plot x1 and x2 steady states (simulationX).....	126
Figure 5.45 Model of the system classical simulation (simulationX).....	128
Figure 5.46 Plot x1 and x2 classical simulation (simulationX).....	128
Figure 5.47 Model of the system different diode models (simulationX) ...	129

Figure 5.48 Plot x1 and x2 different diode models (simulationX)	130
Figure 5.49 Model of the system influence of simulation algorithms (simulationX)	131
Figure 5.50 Plot x1 and x2 influence of simulation algorithms (simulationX)	132
Figure 5.51 Model of the system steady states (simulationX electrical) ...	133
Figure 5.52 Model of the system classical simulation (simulationX electrical)	134
Figure 5.53 Plot x1 and x2 classical simulation (simulationX electrical)...	134
Figure 5.54 Model of the system different diode models (simulationX electrical)	135
Figure 5.55 Plot x1 and x2 different diode models (simulationX electrical)	135
Figure 5.56 Model of the system influence of simulation algorithms (simulationX electrical)	136
Figure 5.57 Plot x1 and x2 influence of simulation algorithms (simulationX electrical)	136

TABLE LIST

Table 2-1 List of Solver	9
Table 3-1 Eigenvalues of $R(t)$ (matlab)	27
Table 3-2 Eigenvalues of $R(t)$ (dymola)	36
Table 3-3 Eigenvalues of $R(t)$ (simulationX)	45
Table 4-1 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (matlab)	53
Table 4-2 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (simulink).....	56
Table 4-3 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (simulink stateflow).....	60
Table 4-4 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (dymola)	64
Table 4-5 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (mosilab)	70
Table 4-6 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (simulationX)	74
Table 6-1 The Result of the simulation software completing the task for comparison 3	137
Table 6-2 The list of the time needed to simulate the task for comparison 3	138
Table 6-3 The Result of the simulation software completing the task for comparison 5	139
Table 6-4 The list of the time needed to simulate the task for comparison 5	139
Table 6-5 The Result of the simulation software completing the task for comparison 20	140
Table 6-6 The list of the time needed to simulate the task for comparison 20.....	142

1. INTRODUCTION

1.1 Background

Before a new product is launched to the market, a company have to test the product and make sure that the product is ready and presentable, especially if the product is a problem solving system, such as control system, automated system or self-learning system. Testing is a highly cost consuming yet unavoidable activity. Therefore to reduce cost of production, company will use methods like simulation to test their product.

Simulation is an imitation of some real thing, state of affairs, or process[1]. The act of simulating something generally entails representing certain key characteristics or behaviours of a selected physical or abstract system. Simulation is used in many contexts, including the modeling of natural systems or human systems in order to gain insight into their functioning. Other contexts include simulation of technology for performance optimization, safety engineering, testing, training and education. Simulation proves also cost effective, reducing the cost of the production. One of simulation that common used is computer simulation.

A computer simulation, a computer model or a computational model is a computer program, or network of computers, that attempts to simulate an abstract model of a particular system[2]. Computer simulations have become a useful part of mathematical modelling of many natural systems in physics (computational physics), chemistry and biology, human systems in economics, psychology, and social science and in the process of engineering new technology, to gain insight into the operation of those systems, or to observe their behavior.

There are so many simulation softwares in the market which offers different abilities and advantages. The various choices has makes it even more difficult for end-users (company) to choose which one is more suitable and useful for the company. On this thesis Three “comparison problem” based on electrotechnic will be compared each other by using four simulations

software (Matlab/Simulink, Dymola, Mosilab and SimulationX), with different approaches to model of the system.

1.2 Problem's Formulation

Based on the problem at hand, this thesis will focus on problems as follow:

- Designing and building model using 4 simulation softwares (Matlab/Simulink, Dymola, Mosilab, SimulationX) with different approaches for 3 “comparisons problem” based on electrotechnic method.
- Provide comparison, analysis and conclusion from the above 4 simulation softwares in search for a better solutions based on the results.

1.3 Concluded Area

Concluded area for designing and building the model of this thesis are as follow:

- Only 3 comparison problems are being used for this thesis and they are all based on electrotechnic with non-linear problem. They are comparison 3, comparison 5 and comparison 20.
- Only 4 Simulation Softwares are being used: Matlab/Simulink, Dymola, Mosilab and SimulationX.
- Only hybrid model, textual mode, electrical model and statechart approach are being used for problem solving.
- System Platform is Windows XP for Matlab/Simulink, Dymola and SimulationX software and Linux UBUNTU for Mosilab.
- The result of the research will be tested on PC Intel Pentium D, 2 x 2,66 GHz and Dell notebook Latitude D630 Intel Centrino Duo.
- Matlab/Simulink version 7.4 R2007a, Dymola version 6.0b, Mosilab version 3.1, Simulation X version 2.0 are being used.
- All the feature of Matlab/Simulink, Dymola, Mosilab and Simulation will not be discussed in details, only the features used in this thesis.

1.4 Objective

The objectives of this thesis are:

- To have better understanding on the characteristics, weaknesses and strengths in each of the 4 simulation softwares analyzed in this thesis, within the concluded area.
- As a future reference for academical purpose.

1.5 Research Plan

1.5.1 Study Literature

Literature studies will be done to have better understanding on how to use, explore and practice 4 simulation softwares used in the thesis, and to understand the approaching methods to find the solution of the problem.

1.5.2 Design and Solutions

Step by step design of the model with different approach methods and simulation softwares, simulate the model to get the solutions. Compare all solutions based on the same approach with different simulation software.

1.6 Writing Method

The writing method and the abstract of each chapter are:

- CHAPTER I INTRODUCTION

Explanation about the background and formulation of the problem, concluded area, objective and research plan which will be implemented.

- CHAPTER II THEORY

Brief discussion about the simulation software, software language, functions and methods that being used to solve the problem.

- CHAPTER III Comparison 3: Generalized Class-E Amplifier

Detail discussion about the comparison 3 problem, model design planning and model solution.

- CHAPTER IV Comparison 5: Two State Model

Detail discussion about the comparison 5 problem, model design planning and model solution.

- CHAPTER V Comparison 20: Electrical Model – Basics

Detail discussion about the comparison 5 problem, model design planning and model solution.

- Chapter VI Comparison

Consist of comparison table and discuss advantage and disadvantage for each simulation software

- CHAPTER VII CONCLUSION AND SUGGESTION

Final conclusion and suggestion to improve model development.

2. THEORY

2.1 MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation[3]. Typical uses includes:

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. MATLAB has evolved over the years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis. Screenshot of Matlab is shown in figure 2.1.

2.1.1 The MATLAB System

The MATLAB system consists of five main parts:

1. Development Environment.

This is a set of tools and facilities that will help you to use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and command Window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path.

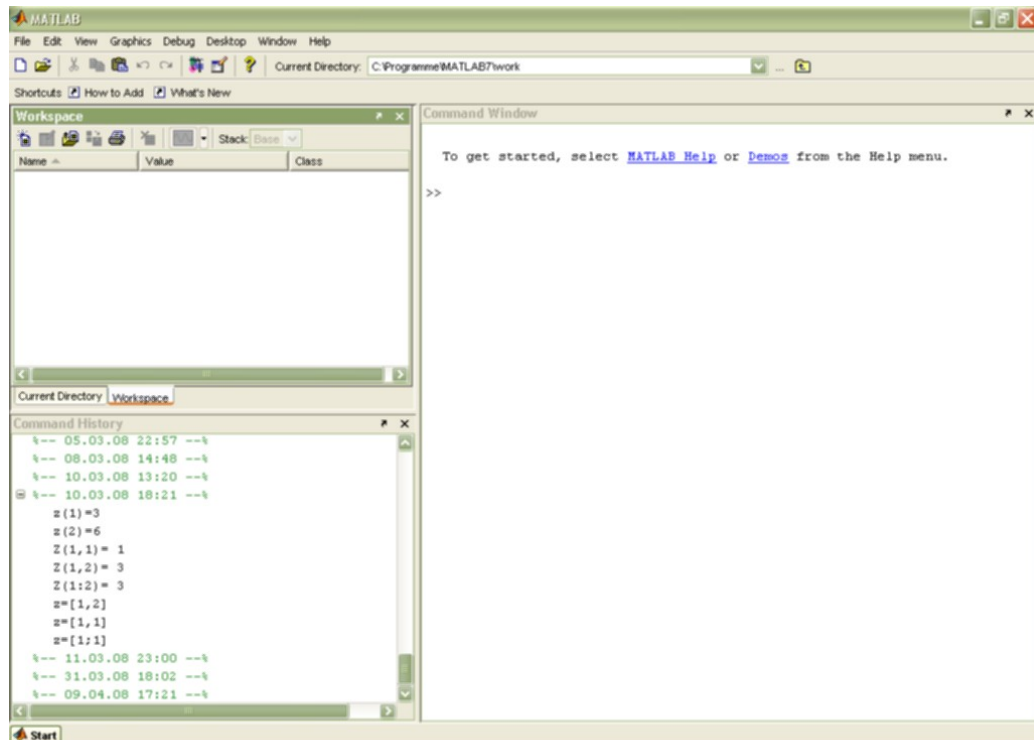


Figure 2.1 Screenshot of Matlab

2. The MATLAB Mathematical Function Library.

This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions such as matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

3. The MATLAB Language.

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create a quick

and dirty throwaway programs, and "programming in the large" to create large and complex application programs.

4. Graphics.

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

5. The MATLAB Application Program Interface (API).

This is a library that allows you to write C and Fortran programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files

2.1.2 ODE SOLVER

2.1.2.1 ODE File

Odefile is not a command or function. It is a help entry that describes how to create an M-file defining the system of equations to be solved. This definition is the first step in using any of the MATLAB ODE solvers. We can use the odefile M-file to define a system of differential equations in one of these forms

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}) \quad \text{or} \quad \mathbf{M}(t, \mathbf{y})\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$$

where: t is a scalar independent variable, typically representing time.

\mathbf{y} is a vector of dependent variables.

\mathbf{f} is a function of t and \mathbf{y} returning a column vector the same length as \mathbf{y} .

$\mathbf{M}(t, \mathbf{y})$ is a time-and-state-dependent mass matrix.

The ODE file must accept the arguments `t` and `y`, although it does not have to use them. By default, the ODE file must return a column vector the same length as `y`.

2.1.2.2 Odeset

Create or alter options structure for input to ordinary differential equation (ODE) solvers Syntax:

```
options = odeset('name1',value1,'name2',value2,...)
options = odeset(oldopts,'name1',value1,...)
options = odeset(oldopts,newopts)
```

`odeset`

Description:

The `odeset` function lets you adjust the integration parameters of the ODE solvers. `options = odeset('name1',value1,'name2',value2,...)` creates an integrator options structure in which the named properties have the specified values. Any unspecified properties have default values. It is sufficient to type only the leading characters that uniquely identify a property name. Case is ignored for property names.

`options = odeset (oldopts,'name1',value1,...)` alters an existing options structure `oldopts`.

`options = odeset (oldopts,newopts)` alters an existing options structure `oldopts` by combining it with a new options structure `newopts`. Any new options not equal to the empty matrix overwrite corresponding options in `oldopts`.

`Odeset` with no input arguments displays all property names as well as their possible and default values.

2.1.2.3 Initial Value ODE Problem Solvers

These are the initial value problem solvers. The table 2.1 lists the kind of problem you can solve with each solver, and the method each solver uses.

Table 2.1 List of Solver

Solver	Solves These Kinds of Problems	Method
ode45	Nonstiff differential equations	Runge-Kutta
ode23	Nonstiff differential equations	Runge-Kutta
ode113	Nonstiff differential equations	Adams
ode15s	Stiff differential equations and DAEs	NDFs (BDFs)
ode23s	Stiff differential equations	Rosenbrock
ode23t	Moderately stiff differential equations and DAEs	Trapezoidal rule
ode23tb	Stiff differential equations	TR-BDF2
ode15i	Fully implicit differential equations	BDFs

2.2 SIMULINK

Simulink is a software package that enables us to model, simulate, and analyze systems whose outputs change over time. Such systems are often referred to as dynamic systems. Simulink can be used to explore the behavior of a wide range of real-world dynamic systems, including electrical circuits, shock absorbers, braking systems, and many other electrical, mechanical, and thermodynamic systems[4].

Simulating a dynamic system is a two-step process with Simulink. First, a user creates a block diagram, using the Simulink model editor, which graphically depicts time-dependent mathematical relationships among the system's inputs, states, and outputs. The user then commands Simulink to simulate the system represented by the model from a specified start time to a specified stop time. The screenshot of Simulink is shown in figure 2.2.

2.2.1 States

Typically the current values of some system, and hence model, outputs are functions of the previous values of temporal variables. Such variables are called states. Computing a model's outputs from a block diagram hence

entails saving the value of states at the current time step for use in computing the outputs at a subsequent time step. Simulink performs this task during simulation for models that define states.

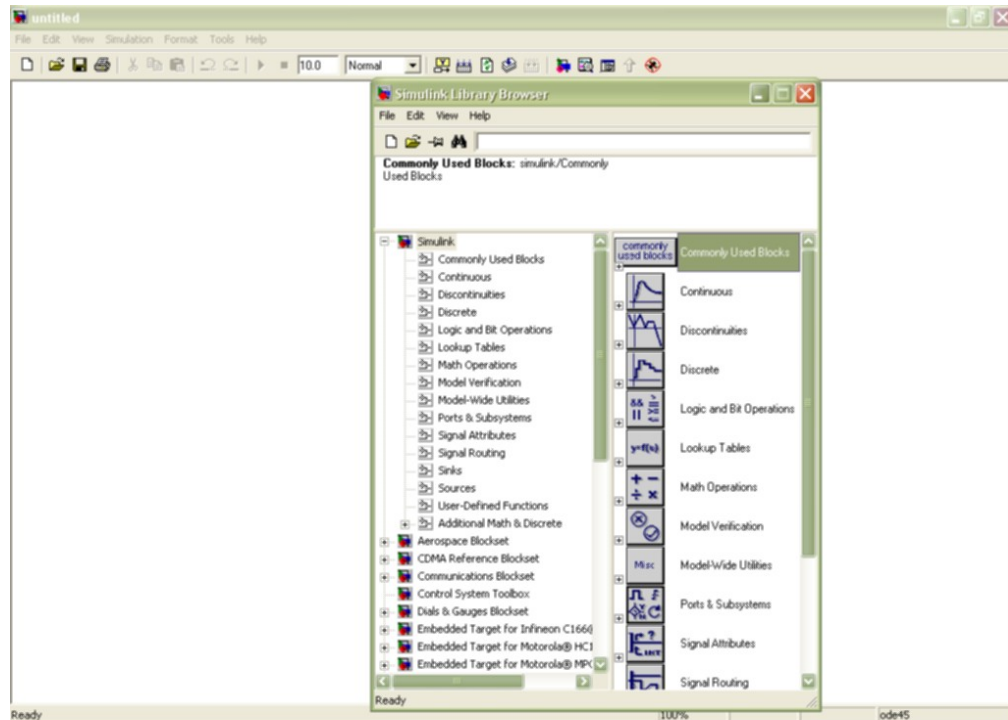


Figure 2.2 Screenshot of Simulink

Two types of states that can occur in a Simulink model: discrete and continuous states. A continuous state changes continuously. Examples of continuous states are the position and speed of a car. A discrete state is an approximation of a continuous state where the state is updated (recomputed) using finite (periodic or aperiodic) intervals. An example of a discrete state would be the position of a car shown on a digital odometer where it is updated every second as opposed to continuous state. In the limit, as the discrete state time interval approaches zero, a discrete state becomes equivalent to a continuous state.

Blocks implicitly define a model's states. In particular, a block that needs some or all of its previous outputs to compute its current outputs implicitly

defines a set of states that need to be saved between time steps. Such a block is said to have states is shown in figure 2.3.

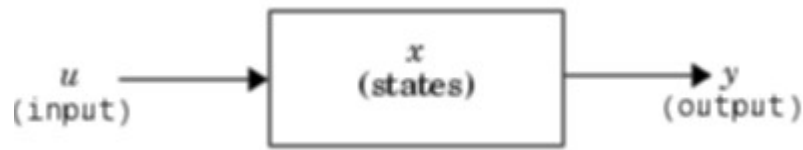


Figure 2.3 Block have states

2.2.1.1 Continuous States

Computing a continuous state entails knowing its rate of change, or derivative. Since the rate of change of a continuous state typically itself changes continuously (i.e., is itself a state), computing the value of a continuous state at the current time step entails integration of its derivative from the start of a simulation. Thus modeling a continuous state entails representing the operation of integration and the process of computing the state's derivative at each point in time. Simulink block diagrams use Integrator blocks to indicate integration and a chain of operator blocks connected to the integrator block to represent the method for computing the state's derivative. The chain of block's connected to the Integrator's is the graphical counterpart to an ordinary differential equation (ODE).

In general, excluding simple dynamic systems, analytical methods do not exist for integrating the states of real-world dynamic systems represented by ordinary differential equations. Integrating the states requires the use of numerical methods called ODE solvers. These various methods trade computational accuracy for computational workload. Simulink comes with computerized implementations of the most common ODE integration methods and allows a user to determine which it uses to integrate states represented by Integrator blocks when simulating a system.

Computing the value of a continuous state at the current time step entails integrating its values from the start of the simulation. The accuracy of numerical integration in turn depends on the size of the intervals between time steps. In general, the smaller the time step, the more accurate the

simulation. Some ODE solvers, called variable time step solvers, can automatically vary the size of the time step, based on the rate of change of the state, to achieve a specified level of accuracy over the course of a simulation. Simulink allows the user to specify the size of the time step in the case of fixed-step solvers or allow the solver to determine the step size in the case of variable-step solvers. To minimize the computation workload, the variable-step solver chooses the largest step size consistent with achieving an overall level of precision specified by the user for the most rapidly changing model state. This ensures that all model states are computed to the accuracy specified by the user.

2.2.1.2 Discrete States

Computing a discrete state requires knowing the relationship between the current time and its value at the time at which it previously changed value. Simulink refers to this relationship as the state's update function. A discrete state depends not only on its value at the previous time step but also on the values of a model's inputs. Modeling a discrete state thus entails modeling the state's dependency on the systems' inputs at the previous time step. Simulink block diagrams use specific types of blocks, called discrete blocks, to specify update functions and chains of blocks connected to the inputs of the block's to model the state's dependency on system inputs.

As with continuous states, discrete states set a constraint on the simulation time step size. Specifically a step size must be chosen that ensure that all the sample times of the model's states are hit. Simulink assigns this task to a component of the Simulink system called a discrete solver. Simulink provides two discrete solvers: a fixed-step discrete solver and a variable-step discrete solver. The fixed-step discrete solver determines a fixed step size that hits all the sample times of all the model's discrete states, regardless of whether the states actually change value at the sample time hits. By contrast, the variable-step discrete solver varies the step size to ensure that sample time hits occur only at times when the states change value.

2.2.1.3 Modeling Hybrid Systems

A hybrid system is a system that has both discrete and continuous states. Strictly speaking a hybrid model is identified as having continuous and discrete sample times from which it follows that the model will have continuous and discrete states. Solving a model of such a system entails choosing a step size that satisfies both the precision constraint on the continuous state integration and the sample time hit constraint on the discrete states.

Simulink meets this requirement by passing the next sample time hit as determined by the discrete solver as an additional constraint on the continuous solver. The continuous solver must choose a step size that advances the simulation up to but not beyond the time of the next sample time hit. The continuous solver can take a time step short of the next sample time hit to meet its accuracy constraint but it cannot take a step beyond the next sample time hit even if its accuracy constraint allows it to.

2.3. Stateflow

Stateflow is a graphical design and development tool that works with Simulink. Stateflow is a suitable environment for modeling logic used to control and supervise a physical plant modeled in Simulink.

Stateflow integrates with its Simulink environment to model, simulate, and analyze your system. Stateflow lets you design and develop deterministic, supervisory control systems in a graphical environment. It visually models and simulates complex reactive control to provide clear, concise descriptions of complex system behavior using finite state machine theory, flow diagram notations, and state-transition diagrams all in the same diagram. Stateflow brings system specification and design closer together. It is easy to create designs, consider various scenarios, and iterate until the Stateflow diagram models the desired behavior. The screenshot of stateflow is shown in figure 2.4.

2.3.1 Finite State Machine Concepts

Stateflow is an example of a finite state machine. A finite state machine is a representation of an event-driven (reactive) system. In an event-driven system, the system makes a transition from one state (mode) to another prescribed state, provided that the condition defining the change is true. For example, you can use a state machine to represent a car's automatic transmission. The transmission has a number of operating states: park, reverse, neutral, drive, and low. As the driver shifts from one position to another the system makes a transition from one state to another, for example, from park to reverse.

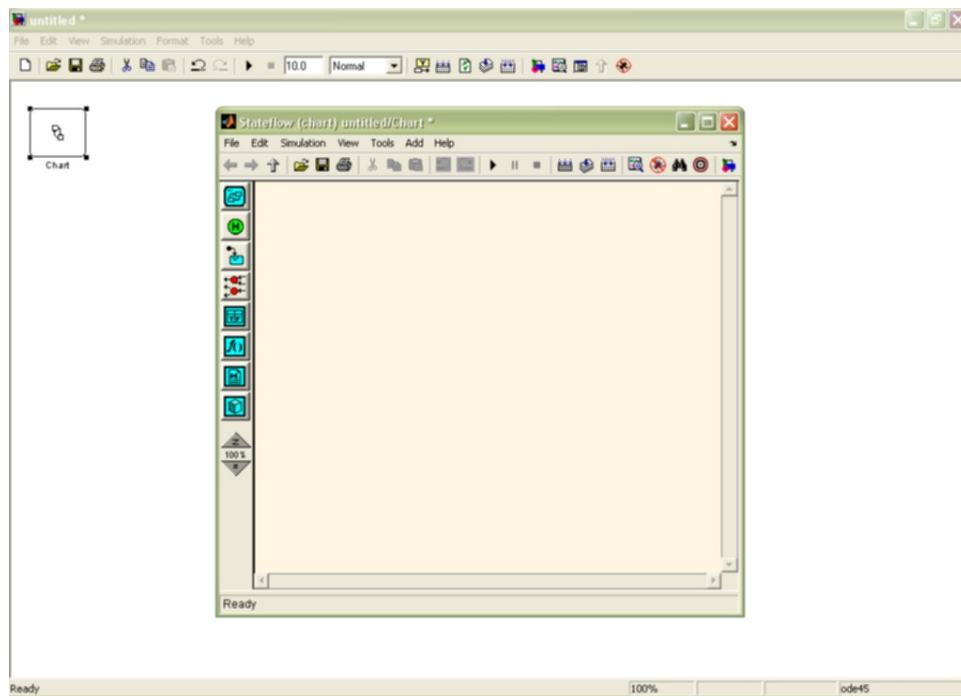


Figure 2.4 Screenshot of Stateflow

Traditionally, designers used truth tables to represent relationships among the inputs, outputs, and states of a finite state machine. The resulting table describes the logic necessary to control the behavior of the system under study. Another approach to designing event-driven systems is to model the behavior of the system by describing it in terms of transitions among states. The state that is active is determined based on the occurrence of events

under certain conditions. State-transition diagrams and bubble diagrams are graphical representations based on this approach.










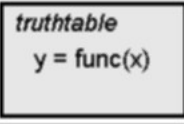

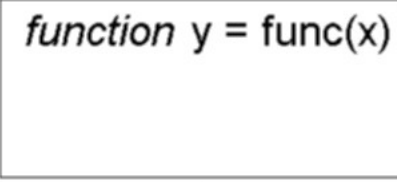

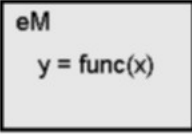



Name	Notation	Toolbar Icon
State		
Transition		NA
History Junction		
Default Transition		
Connective Junction		
Truth Table Function		
Graphical Function		
Embedded MATLAB Function		
Box		

Figure 2.5 Graphical objects in Stateflow

2.3.2 Stateflow Notations

2.3.2.1 Nongraphical Objects

Stateflow defines event, data, and target objects that do not have graphical representations in the Stateflow diagram editor.

1. Event Objects

An event is a Stateflow object that can trigger a whole Stateflow chart or individual actions in a chart. Because Stateflow charts execute by reacting to events

2. Data Objects

A Stateflow chart stores and retrieves data that it uses to control its execution. Stateflow data resides in its own workspace.

3. Target Objects

A target is a program that executes a Stateflow model or a Simulink model containing a Stateflow machine.

2.3.2.2 Graphical Object

The name of each graphical object in Stateflow is shown in figure 2.5

2.4 Dymola

Dymola - Dynamic Modeling Laboratory - is suitable for modeling of various kinds of physical systems[5]. It supports hierarchical model composition, libraries of truly reusable components, connectors and composite acasual connections. Model libraries are available in many engineering domains.

Dymola uses a new modeling methodology based on object orientation and equations[6]. The usual need for manual conversion of equations to a block diagram is removed by the use of automatic formula manipulation. Other highlights of Dymola are:

- Handling of large, complex multi-engineering models.
- Faster modeling by graphical model composition.
- Faster simulation - symbolic pre-processing.
- Open for user defined model components.
- Open interface to other programs.
- 3D Animation.
- Real-time simulation.

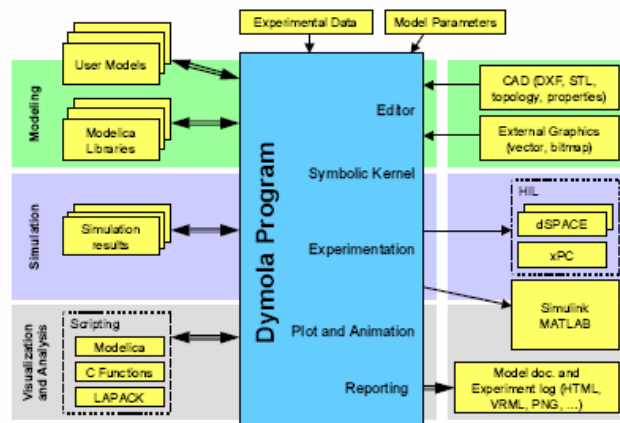


Figure 2.6 Architecture of Dymola

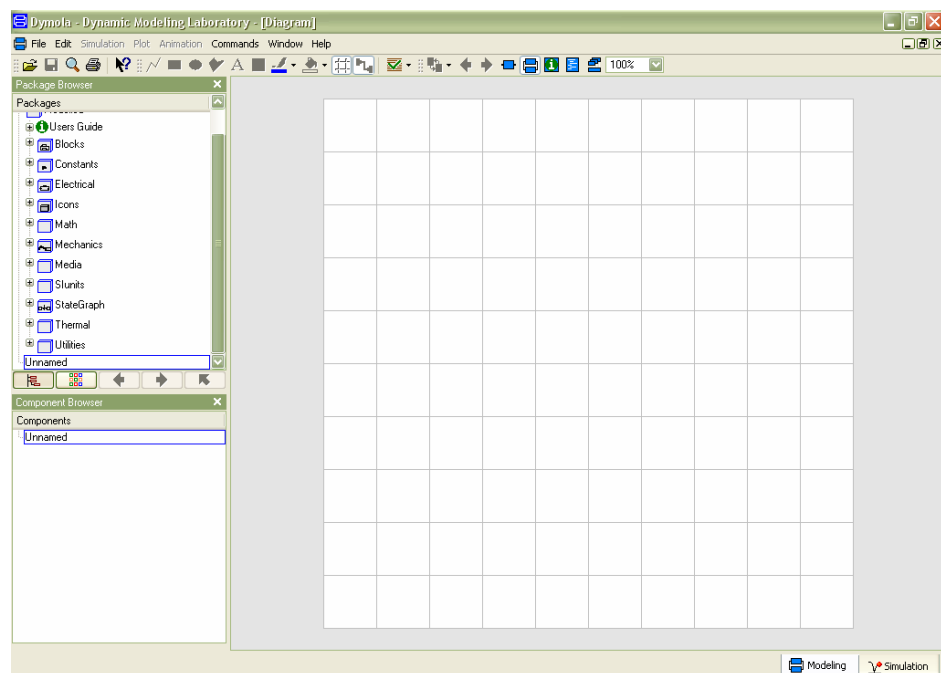


Figure 2.7 Screenshot of Dymola

2.4.1 Architecture of Dymola

The architecture of the Dymola program is shown in figure 2.6. Dymola has a powerful graphic editor for composing models. Dymola is based on the use of Modelica models stored on files. Dymola can also import other data and graphics files. Dymola contains a symbolic translator for Modelica equations

generating C-code for simulation. The C-code can be exported to Simulink and hardware-in-the-loop platforms.

Dymola has powerful experimentation, plotting and animation features. Scripts can be used to manage experiments and to perform calculations. Automatic documentation generator is provided.

2.4.2 Basic Operations of Dymola

Dymola has two kinds of windows: Main window and Library window. The Main window operates in one of two modes: Modeling and Simulation.

The Modeling mode of the Main window is used to compose models and model components. The Simulation mode is used to make experiment on the model, plot results and animate the behavior. The Simulation mode also has a scripting subwindow for automation of experimentation and performing calculations. The screenshot of dymola is shown in figure 2.7 and simulation mode of dymola is shown in figure 2.8.

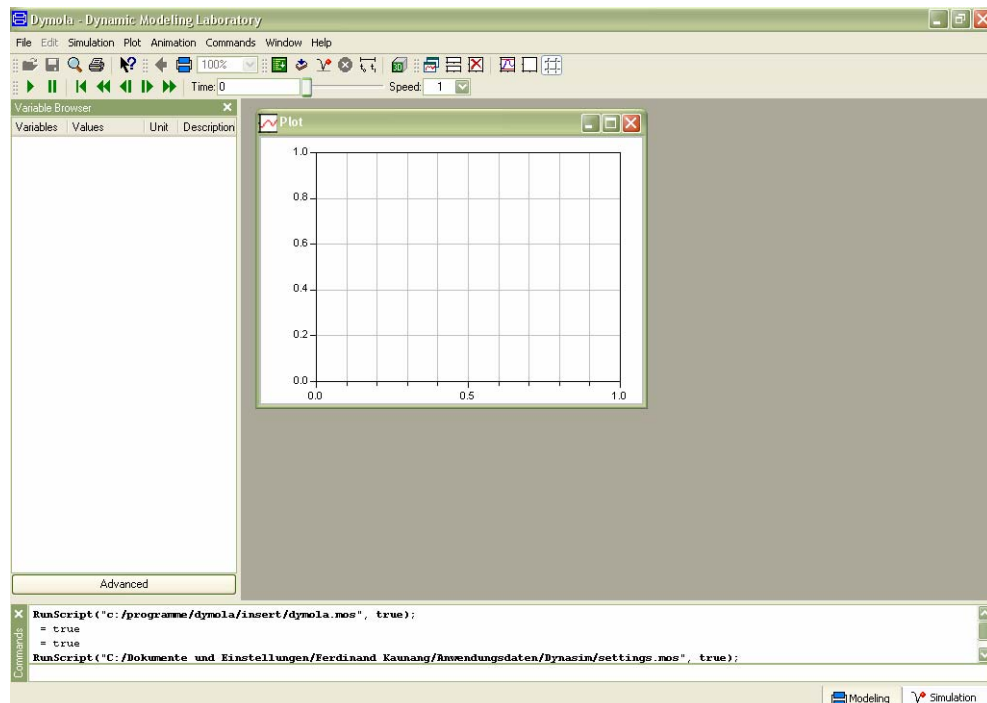


Figure 2.8 Simulation Mode of Dymola

2.5 Mosilab

2.5.1 MOSILAB Architecture

The GENSIM simulation tool MOSILAB (**M**odeling and **S**imulation **L**aboratory) includes the simulation kernel (consisting of a model compiler, a runtime system and a numerical solver framework) and an IDE (Interactive Development Environment)[7], the interface to the user of the simulation system. It supports him both in the modelling process with the help of graphical UML and text editors and during the simulation experiment.

Figure 2.9 shows the data flow within the MOSILABtools: Beside experiment definitions, the models also developed within the IDE are stored as MOSILA model classes. Together with the MOSILA standard library, these MOSILA models are compiled to C++ classes by the MOSILA compiler. Using the GNU gcc/g++ compiler, the executable simulator is built from these C++ representations and the simulator kernel classes.

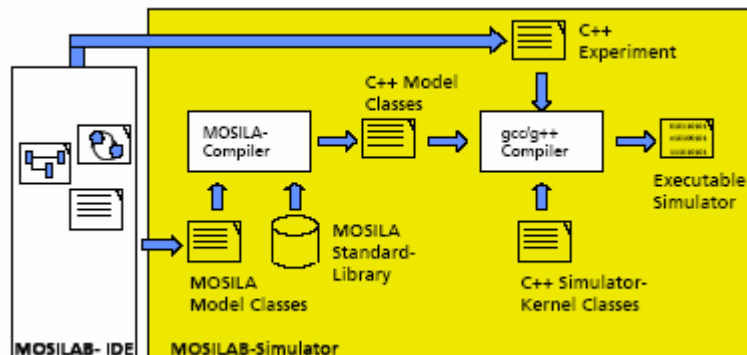


Figure 2.9 Data Flow within Mosilab

2.5.2 MOSILAB Configurations

MOSILAB can be configured in to act in three modes:

a) The generated simulator is represented by a single, monolithic C/C++ application. This option has the smallest memory footprint and only few dependencies on the underlying platform, so it is most useful e.g. for embedded applications. However, the functionality of w.r.t. dynamic parameterization at runtime is limited.

b) The simulator is represented by a shared object file which can be dynamically linked to a main program which controls the simulation. MOSILAB uses the Python language and interpreter (<http://www.python.org>) as its central mechanism for experiment control. The simulator is loaded as an “extension” into the interpreter, and “experiment scripts”, written in Python, access the simulator API via a Python-level interface.

c) The simulator acts as a service. In this mode, the simulator is linked with appropriate libraries to publish its API via standard TCP/IP-based protocols such as SOAP in a web or grid services framework (e.g. the upcoming release 4 of the Globus Toolkit). In this mode, the simulator can easily be controlled in protocol-based, platformindependent manner, and it is easy to deploy multiple (and potentially large numbers of) “simulator service instances” in a coordinated way in a heterogeneous network or Grid, for instance to solve an optimization problem. Python-based experiment control support is available in this mode as well a (Python) client library is used to talk to the simulator’s API over the network in this case. The simulator maintains a run-time representation of the model object hierarchy³⁵ (as defined in the source and evolving according to the structural variability of the model). This run-time model can be inquired via introspection features of the simulator API, so (using the synchronisation features offered by this API, too) experiment scripts are able to follow the structural changes over the entire course of a simulation run. This way, if special reactions to model structure changes are needed, which cannot be formulated in the model itself due to their complexity, such reactions can easily be implemented in the experiment script.

2.5.3 The MOSILAB Development Environment

The MOSILAB Development Environment (MOSILAB-IDE) supports the user during the modelling process and the simulation experiment. In the modelling mode the user can choose between three graphical UMLH-editors

(class diagrams, collaboration diagrams and statecharts) and a text editor. While the graphical views give the user an intuitive overview about the structure and the logic of a complex model, the text editor offers the user features like syntax highlighting for implementing the MOSILA/Modelica models.

In the experiment mode of the MOSILAB-IDE the user can define the root model for the simulation experiment, can parameterize model variables and can choose and configure a suitable numerical solver. Furthermore he can define a subset of model variables, which should be observed during the simulation experiment. The observed variables are the basis for different types of post-processing. Figure 2.10 shows a screenshot of the prototypical²⁶ implementation of the MOSILAB-IDE.

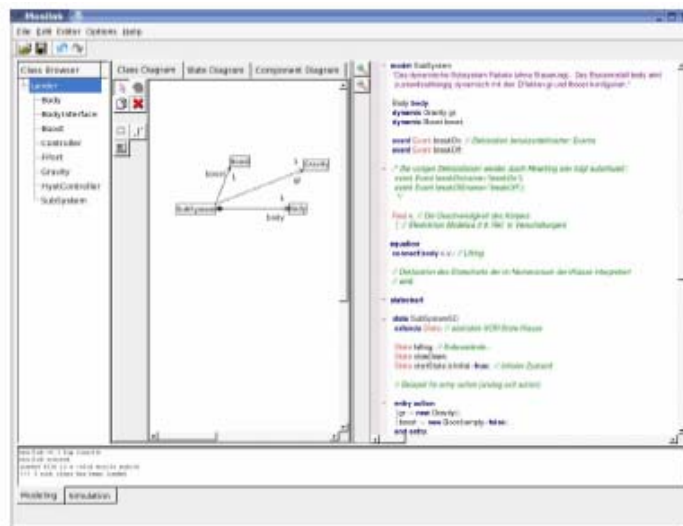


Figure 2.10 Screenshot of Prototypical Implementation of the MOSILAB-IDE

2.6 SimulationX

SimulationX is a standard software based on modelica language, for valuation of the interaction of all components of technical systems. It is the universal CAE tool for modeling, simulation and analyzing of physical effects – with ready-to-use model libraries for 1D mechanics, 3D multibody systems, power transmission, hydraulics, pneumatics, thermodynamics, electrics,

electrical drives, magnetics as well as controls – postprocessing included[8]. Figure 2.11 shows SimulationX screenshot.

SimulationX supports the use of the most convenient way of modeling in each of the engineering domains - signal blocks in the control domain, circuit diagrams in the electronic, magnetic and fluid domains, functional sketches in 1D mechanics, and 3D geometrical structures with visualization and animation in 3D mechanics.

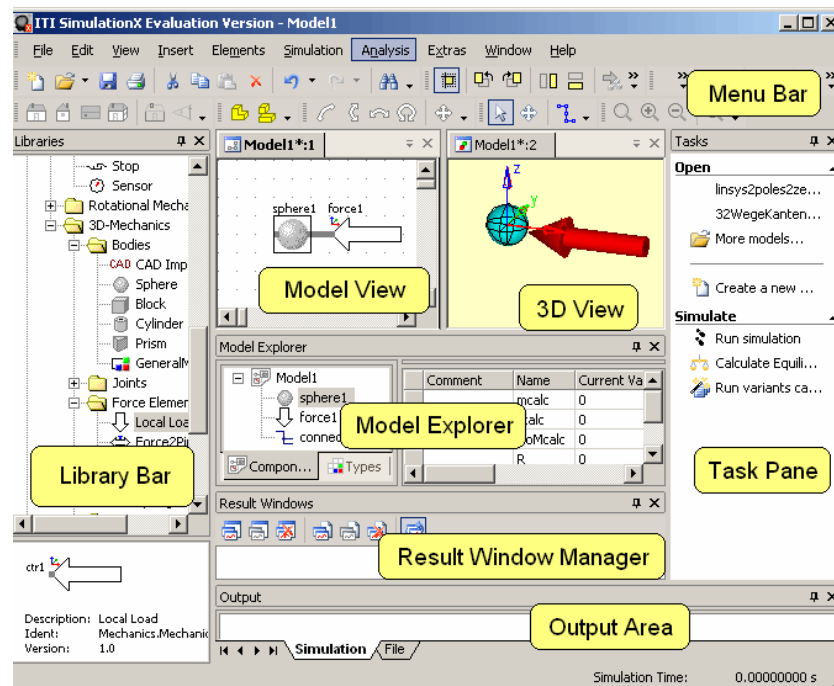


Figure 2.11 Screenshot of SimulationX.

The all-domain ITI modeling philosophy opens up new application fields for real system simulation and analysis including:

- Linear and Rotary Mechanics
- Multibodies
- Hydraulics
- Pneumatics
- Controls
- Electronics
- Magnetics
- Power Transmission
- Electromechanical
- Thermics
- Thermal Fluid
- Thermodynamics

2.6.1. Library Bar

The library bar offers access to the installed element types. For clarity, the element types are subdivided into libraries (groups). In the tree view, element types and libraries are shown according to their hierarchy. The library view offers access to the element types over symbols that are administered in folders.

2.6.2 Model View

The model view serves for the graphical representation of the structure and the modification of the simulation model. Elements and connections are the components of a simulation model. Elements have connectors that can link together via a connection. Connections can be branched arbitrarily; i.e. they can link more than two connectors.

There are different types of connectors, such as mechanical (linear and rotary), hydraulic, and electrical connectors, as well as signal inputs and outputs. Only connectors of the same type can be connected to each other. Each connector possesses an unambiguous name with respect to the corresponding element. These names can be made visible via "Pin Labels" (menu "View"). Figure 2.12 shows components in SimulationX.

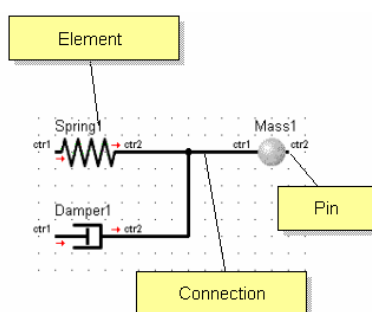


Figure 2.12 Components in SimulationX.

3. Comparison 3: Generalized Class-E Amplifier

3.1 Definition

The basic class-E power amplifier was introduced by N.O. Sokal and A.D. Sokal in their classic paper from 1975[9]. It is a switching-mode amplifier that operates with zero voltage and zero slope across the switch at switch turn-off. The actual numerical example is taken from J.C. Mandojana, K.J. Herman and R.E. Zulinski[10]. They use the following equivalent circuit of a generalized class-E amplifier as a test example for a procedure to evaluate steady state boundary conditions by means of MATLAB. Figure 3.1 shows Class E Amplifier

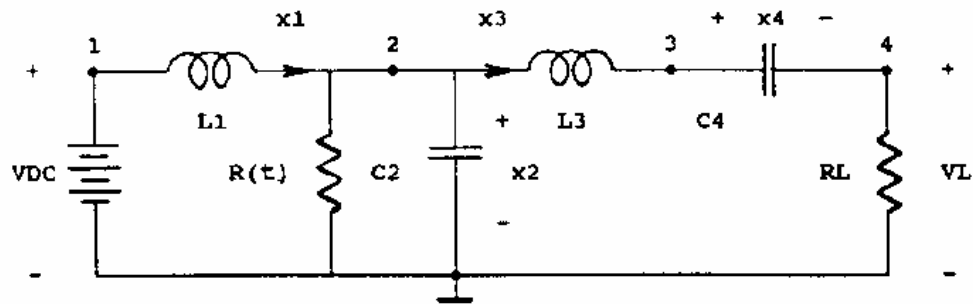


Figure 3.1 Class-E Amplifier

The component values are:

$VDC = 5$ volt, $L1 = 79.9E-6$ henry, $C2 = 17.9E-9$ farad, $L3 = 232.0E-6$ henry, $C4 = 9.66E-9$ farad and $RL = 52.4$ ohm.

The time dependent resistor $R(t)$ models the active device acting as a switch with an ON-resistance of 0.05 ohm and an OFF-resistance of $5.0E+6$ ohm. An extreme ON-resistance of value zero ohm will of course result in a pathological system i.e. the old story of what happens when an ideal capacitor with a certain charge is suddenly short circuited. Furthermore the DC voltage source will be short circuited through the ideal coil $L1$. Figure 3.2 shows function of time $R(t)$.

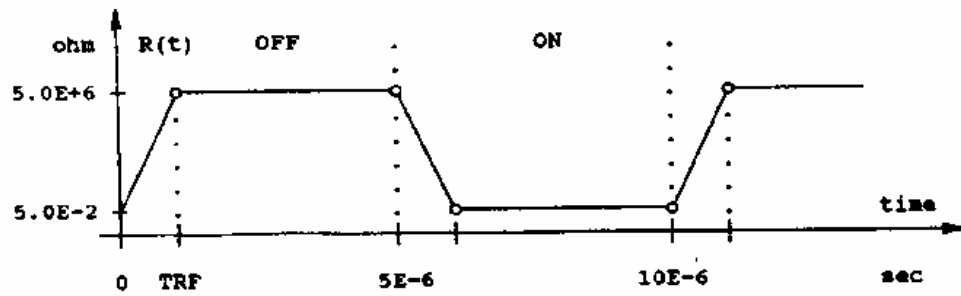


Figure 3.2 Function of time $R(t)$

The duty ratio is 50%. The period is $10E-6$ seconds (frequency 100 kHz). The rise/fall time is $TRF = 1E-15$ seconds.

The equations describing the circuit may be the state-equations where inductor currents and capacitor voltages are chosen as system variables. By using the Kirchhoff voltage and current laws we get the following differential equations:

$$L1 \cdot dx1/dt = -x2 + VDC$$

$$C2 \cdot dx2/dt = +x1 - x2/R(t) - x3$$

$$L3 \cdot dx3/dt = +x2 - RL \cdot x3 - x4$$

$$C4 \cdot dx4/dt = +x3$$

where the variables are as follows: $x1 = IL1$ (the current of $L1$), $x2 = VC2$ (the voltage of $C2$), $x3 = IL3$ (the current of $L3$) and $x4 = VC4$ (the voltage of $C4$). Note that normally the setup of state equations demands a topological analysis of the circuit excluding some inductor currents and capacitor voltages as candidates for system variables (e.g if there is a loop of N capacitors then only $N-1$ of these may be given an arbitrary initial charge).

The following tasks should be performed:

- a. Calculation of the eigenvalues of the system in the ON-period: $R(t)=0.05$ ohm and in the OFF-period: $R(t)=5E+6$ ohm.
- b. Simulation of the system over the time interval $[0, 100E-6]$ sec with the zero-solution as initial state. Time curves of the state variables, the current in the switch resistor $IR(t) = x2/R(t)$ and the output voltage $VL = x3 \cdot RL$ are wanted.

- c. A parameter variation study over the time interval $[0, 9E-6]$ sec with initial solution equal to the final solution at $100E-6$ sec from task (b). The rise/fall time TRF should be varied through the values: $1E-15$, $1E-11$, $1E-9$, $1E-7$ sec. The phase plane curves of $dx_3/dt = VL_3$ as a function of $x_3 = IL_3$ i.e the voltage difference V_2-V_3 as a function of the current IL_3 are wanted. Time curves of the current in the switch resistor $IR(t) = x_2/R(t)$ and the output voltage $VL = x_3*RL$ are wanted.

3.2 Design and Solutions

Block Diagram

The design of this model consist of only 2 block diagram: Time dependent Resistor and Differential Equation. Figure 3.3 shows block diagram for comparison 3.

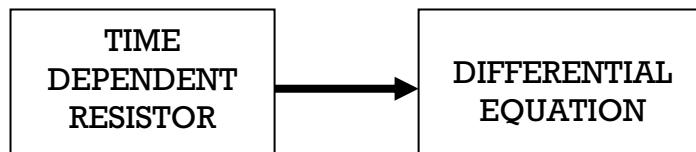


Figure 3.3 Block Diagram for Comparison 3

3.2.1 Matlab

Design of Model

For design model, using differential equation in state space form $dx/dt = A*x$, $x \in R^3$, $A \in R^{3 \times 3}$, with time dependent resistor matrix $A = A(R(t))$. The resistivity $R(t)$ and the differential equation are built by matlab function-type m-files. The time-dependent resistivity $R(t)$ was implemented as a continuous function. In the following these m-file models (A.m, R.m, deq.m)

```

function Aout=A(t)                                     %System Matrix
global VDC L1 C2 L3 C4 RL TRF
Aout=[0 -1/L1 0 0; 1/C2 -1/(C2*R(t)) -1/C2 0;
0 1/L3 -RL/L3 -1/L3; 0 0 1/C4 0];
end
  
```

```

function R_out = R(t)                                   %Time dependent Resistor
global TRF
TRF=1e-15;
k=((5e+6)-(5e-2))/TRF;
  
```

```

t_red=mod(t, (10e-6));
if(0<=t_red)&(t_red<TRF)
    R_out=(5e-2)+k*t_red;
elseif(TRF<=t_red)&(t_red<(5e-6))
    R_out=5e+6;
elseif((5e-6)<=t_red)&(t_red<((5e-6)+TRF))
    R_out=(5e+6)-k*(t_red-(5e-6));
elseif((5e-6)+TRF<=t_red)&(t_red<(10e-6))
    R_out=5e-2;
else
    R_out=-5;
end
function dx=deq(t,x)                                     %Differential equation
global VDC L1 C2 L3 C4 RL TRF
b=[VDC/L1; 0; 0; 0];
dx=(A(t)*x)+b;
end

```

Solutions

Task a Calculation of eigenvalues

Using Matlab built function eig() to determined the eigenvalue's matrix of the system when R(t) is on = 0.05 ohm and when R(t) is off = 5e+6 ohm. The result of eigenvalues is shown in Table 3-1.

```

global VDC L1 C2 L3 C4 RL TRF
TRF= 1e-15; L1= 79.9e-6; VDC= 5; C2= 17.9e-9;
L3= 232e-6; C4= 9.66e-9; RL= 52.4;
ROff= eig(A(TRF))
ROn= eig(A(0))

```

It took 0,101523s to simulate the task a

Table 3-1 Eigenvalues of R(t) (matlab)

Eigenvalues R(t) OFF	Eigenvalues R(t) ON
-54708 +1.0408e+6i	-1.1173e+9
-54708 -1.0408e+6i	-625.78
-58228 +5.3275e+5i	-1.1304e+5 +6.5835e+5i
-58228 -5.3275e+5i	-1.1304e+5 -6.5835e+5i

Task b Simulation of the system

To simulate the system, using matlab built in function ode23s (odesolver) with the solver form:

```
[tsol,xsol]=ode23s('deq',[0 100e-6],[0;0;0;0]);
```

Where ode23s = ode solver matlab built in function

deq = differential equation of the model in form of m-files

[0 100e.6] = simulation time interval

[0;0;0;0] = Initial condition of the model

tsol = time solutions of the model

xsol = variable value solutions of the model

The result for variable current switch resistor $IR(t)$ and output voltage VL is shown in figure 3.4. It took 4,344246s to simulate the task b.

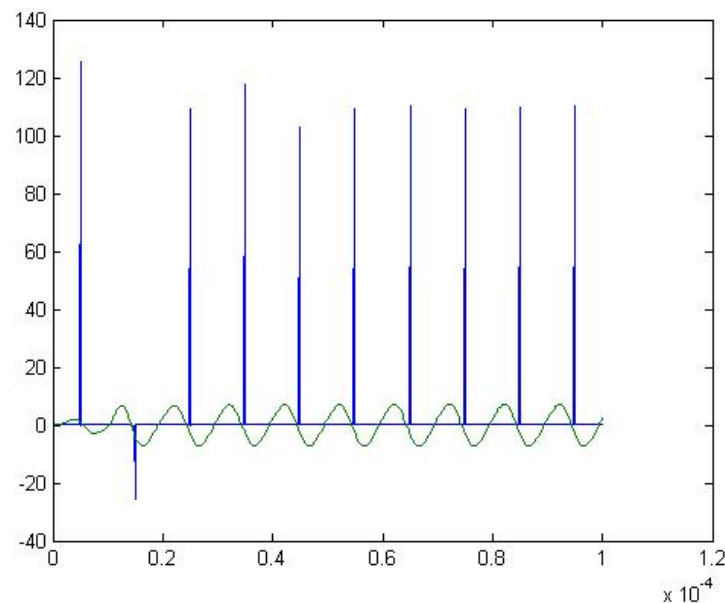


Figure 3.4 The result for variable current switch resistor $IR(t)$ and output voltage VL (matlab)

Task c Parameter Variation Study

The parameter of TRF is varied between $1e-15s$, $1e-11s$, $1e-9s$, $1e-7s$. Initial state for the task c is equal to the final solution given by task b. The time interval is $0 \dots 9e-6s$. Also using the matlab built in function ode23s in the solver form:

```
[tsol,xsol]=ode23s('deq',[0 9e-6],[0.26144;0.010869;0.044044;-14.475]);
x1=xsol(:,1);
x2=xsol(:,2);
x3=xsol(:,3);
x4=xsol(:,4);
VL=x3*RL;
D3=(1/L3)*((x2-VL)-x4);
```

The phase plane curves $dx_3/dt=VL_3$ as a function of $x_3 = IL_3$ is shown in figure 3.5. It took 1.075 s to simulate task c. The four simulations will be executed separately

For the complete calculation and simulation, using Matlab/Simulink version 7.4 R2007a on PC Intel Pentium D, 2 x 2,66 GHz.

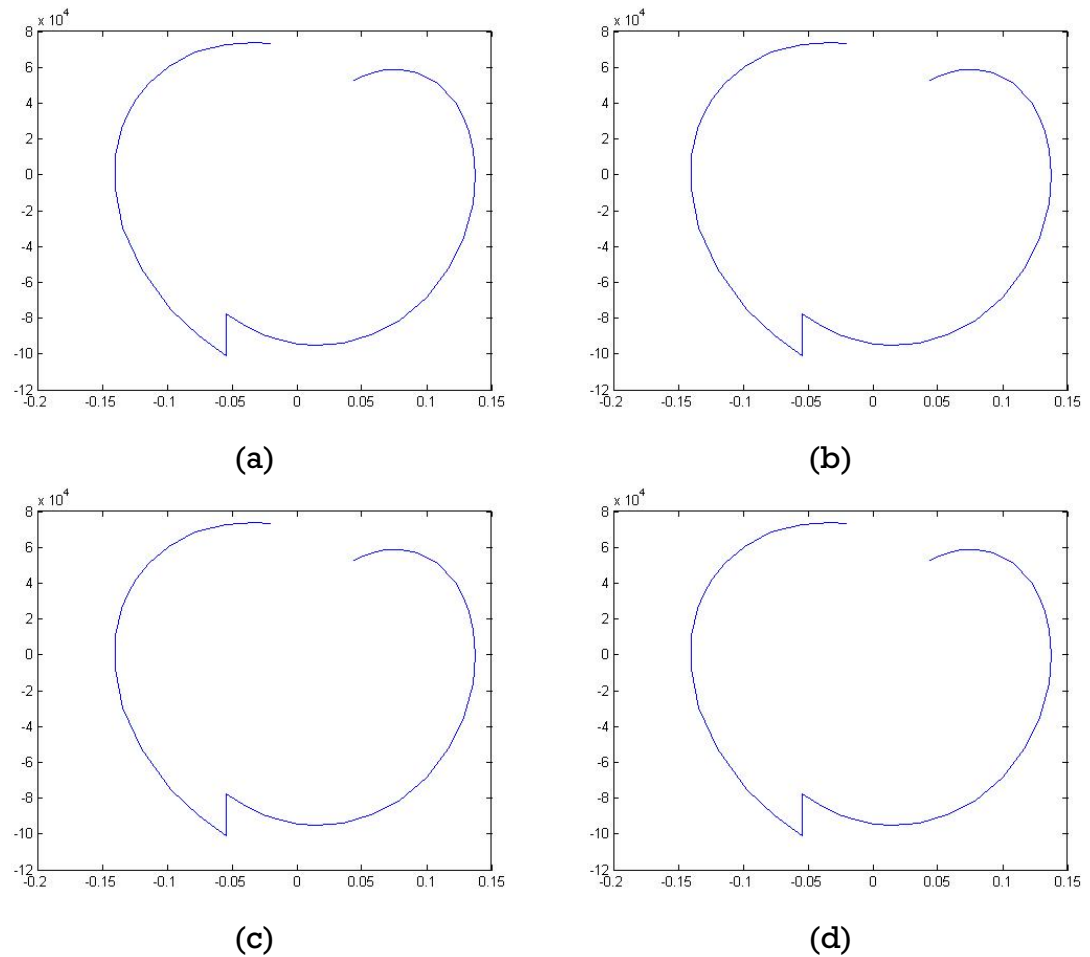


Figure 3.5 The phase plane curves $dx_3/dt=VL_3$ as a function of $x_3 = IL_3$ for TRF(matlab):

(a). 1e-15s (b). 1e-11s (c) 1e-9s (d) 1e-7s

3.2.2 Simulink

3.2.2.1 Hybrid Model

Design of Model

The time dependent resistor is using block clock as a input and built by block embedded Matlab function $R(t)$:

```

function R_out = R(t)
persistent TRF
TRF=1e-15;
k=((5e+6)-(5e-2))/TRF;
t_red=mod(t, (10e-6));
if(0<=t_red)&&(t_red<TRF)
    R_out=(5e-2)+k*t_red;
elseif(TRF<=t_red)&&(t_red<(5e-6))
    R_out=5e+6;
elseif((5e-6)<=t_red)&&(t_red<((5e-6)+TRF))
    R_out=(5e+6)-k*(t_red-(5e-6));
elseif((5e-6)+TRF<=t_red)&&(t_red<(10e-6))
    R_out=5e-2;
else
    R_out=-5;
end

```

The differential equation part is built by block integrator, add/subtract, gain, user defined block and Mux. Figure 3.6 shows the model of the system using Simulink.

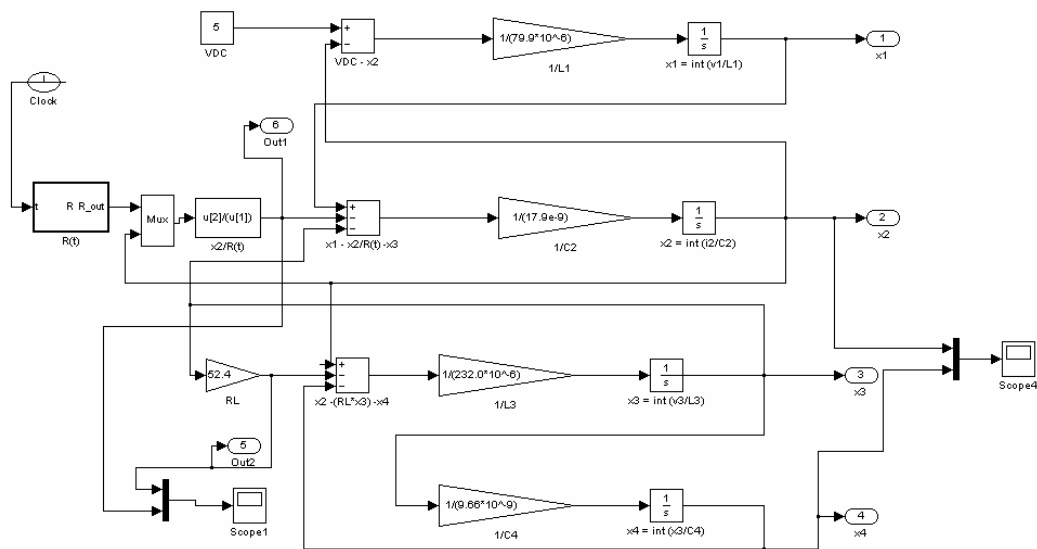


Figure 3.6 Model of the system using Simulink

Solutions

Task a Calculation of eigenvalues

Because there is no special function in simulink to calculate the eigenvalue. Therefore this task can't be done by simulink.

Task b Simulation of the system

To simulate the system, using variable step as a time step, 0 ...100e-6s as simulation time interval and ode23 (Bogacki-Shampine) as the solver. Under

the initial state zero, the result for variable current switch resistor $IR(t)$ and output voltage VL is shown in figure 3.7. It took 4s to simulate the task b.

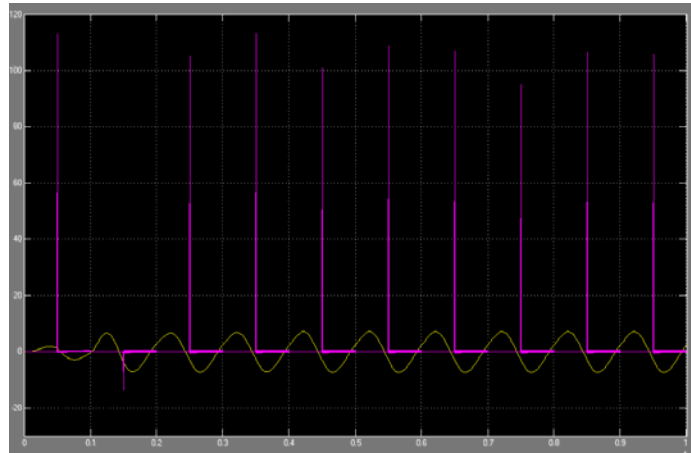
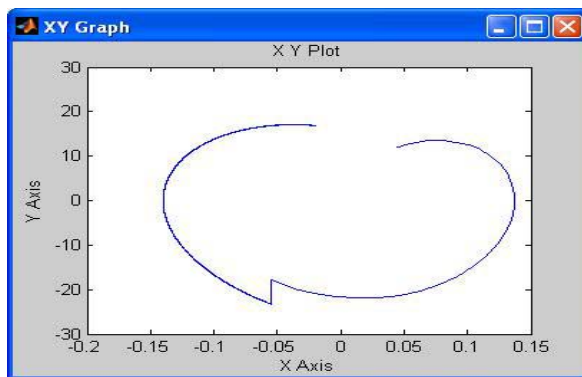
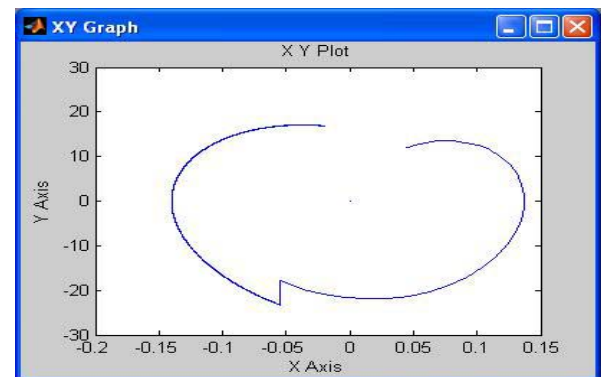


Figure 3.7 The result for variable current switch resistor $IR(t)$ and output voltage $VL(\text{simulink})$

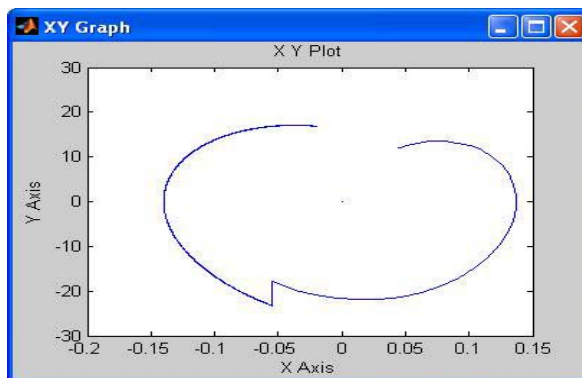
Task c Parameter Variation Study



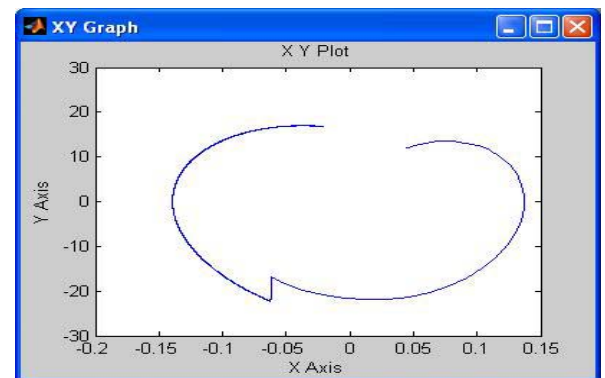
(a)



(b)



(c)



(d)

Figure 3.8 The phase plane curves $dx_3/dt = VL_3$ as a function of $x_3 = IL_3$ for TRF (simulink):

(a). $1e-15s$ (b). $1e-11s$ (c) $1e-9s$ (d) $1e-7s$

The parameter of TRF is varied $1e-15s$, $1e-11s$, $1e-9s$, $1e-7s$. Initial state for the task c is equal to the final solution is given by task b. The time interval is $0 \dots 9e-6s$. As the result, the phase plane curves $dx_3/dt=VL_3$ as a function of $x_3 = IL_3$ is shown in figure 3.8. It took 9,5s to simulate task c. The four simulations will be executed separately.

For the complete calculation and simulation, using Matlab/Simulink version 7.4 R2007a on PC Intel Pentium D, 2 x 2,66 GHz.

3.2.2.2 Stateflow

Design of Model

The model has 3 parts: controller, time dependent resistor and differential equation. The time dependent resistor has also 2 parts for state off and on, using block clock as an input and built by block embedded Matlab function Roff(t) and Ron(t):

```
function R_out = Roff(t)
persistent TRF
TRF=1e-15;
k=((5e+6)-(5e-2))/TRF;
t_red=mod(t, (10e-6));
if(0<=t_red)&&(t_red<TRF)
    R_out=(5e-2)+k*t_red;
elseif(TRF<=t_red)&&(t_red<(5e-6))
    R_out=5e+6;
else
    R_out=5e-2;
end

function R_out = Ron(t)
persistent TRF
TRF=1e-15;
k=((5e+6)-(5e-2))/TRF;
t_red=mod(t, (10e-6));
if((5e-6)<=t_red)&&(t_red<((5e-6)+TRF))
    R_out=(5e+6)-k*(t_red-(5e-6));
elseif((5e-6)+TRF<=t_red)&&(t_red<(10e-6))
    R_out=5e-2;
else
    R_out=5e+6;
End
```

Part controller using stateflow mode to control time dependent resistor block, when is state off and when is state on with output signal SGN. Figure

3.9 shows Stateflow mode of the system. The differential equation part is built by block integrator, add/subtract, gain, user defined block, switch and Mux. Figure 3.10 shows the model of the system using Stateflow mode.

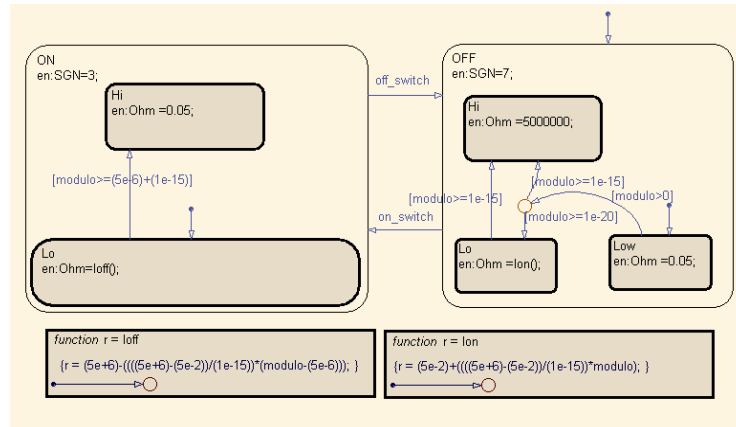
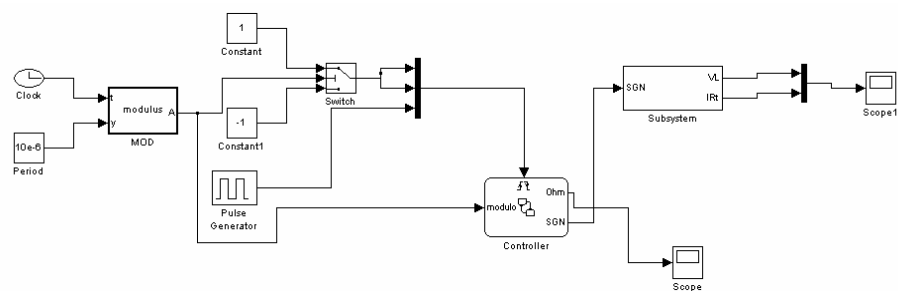
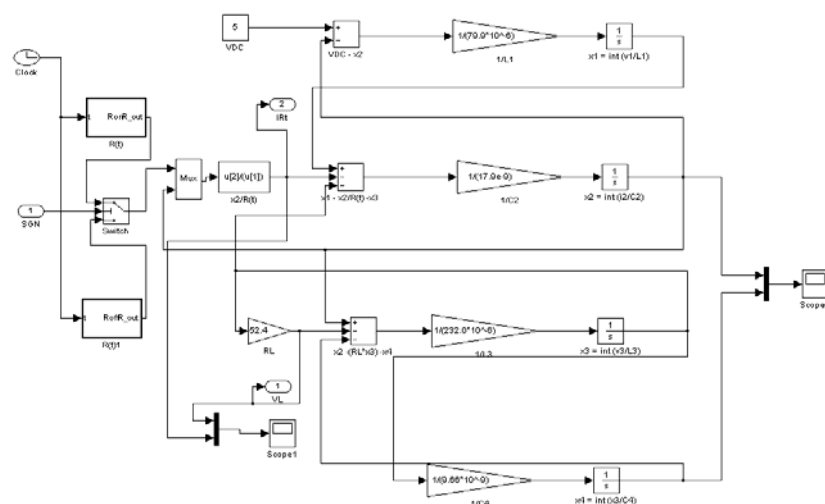


Figure 3.9 Stateflow mode of the system



(a)



(b)

Figure 3.10 Model of the system using Simulink (stateflow)

(a) Main Model (b) Subsystem

Solutions

Task b Simulation of the system

To simulate the system, using variable step as a time step, $0 \dots 100e-6s$ as simulation time interval and ode23s (Stiff/Mod. Rosenbrock) as the solver. Under the initial state zero, the result for variable current switch resistor $IR(t)$ and output voltage VL is shown in figure 3.11. It took 3s to simulate the task b.

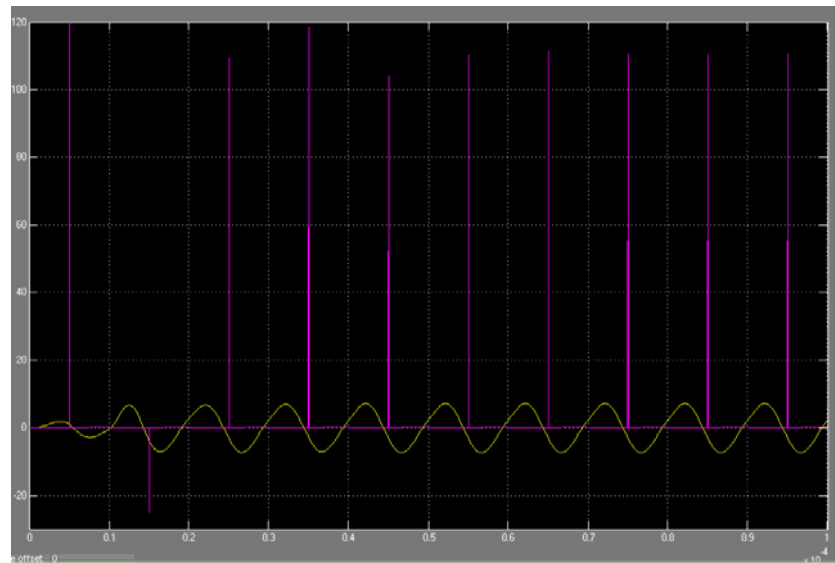


Figure 3.11 The result for variable current switch resistor $IR(t)$ and output voltage VL (simulink stateflow)

Task c Parameter Variation Study

The parameter of TRF is varied between $1e-15s$, $1e-11s$, $1e-9s$, $1e-7s$. Initial state for the task c is equal from the final solution is given by task b. The time interval is $0 \dots 9e-6s$. As the result, the phase plane curves $dx_3/dt=VL_3$ as a function of $x_3 = IL_3$ is shown in figure 3.8. It took 2s to simulate task c. The four simulations will be executed separately.

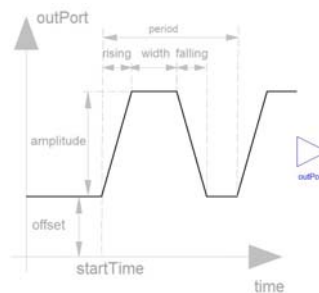
For the complete calculation and simulation, using Matlab/Simulink version 7.4 R2007a on PC Intel Pentium D, 2 x 2,66 GHz.

3.2.3. Dymola

3.2.3.1 Hybrid Model

Design of Model

The time dependent resistor is built by block Trapezoid source and shown in figure 3.12. The differential equation part is built by block integrator, add/subtract, gain, division and constant. Figure 3.13 shows the model of the system using Dymola.



- Amplitude = $5e+6$	- Offset = $5e-2$
- Rising = $1e-15$	- Falling = $1e-15$
- Width = $5e-6$	- Period = $10e-6$

Figure 3.12 Trapezoid Source

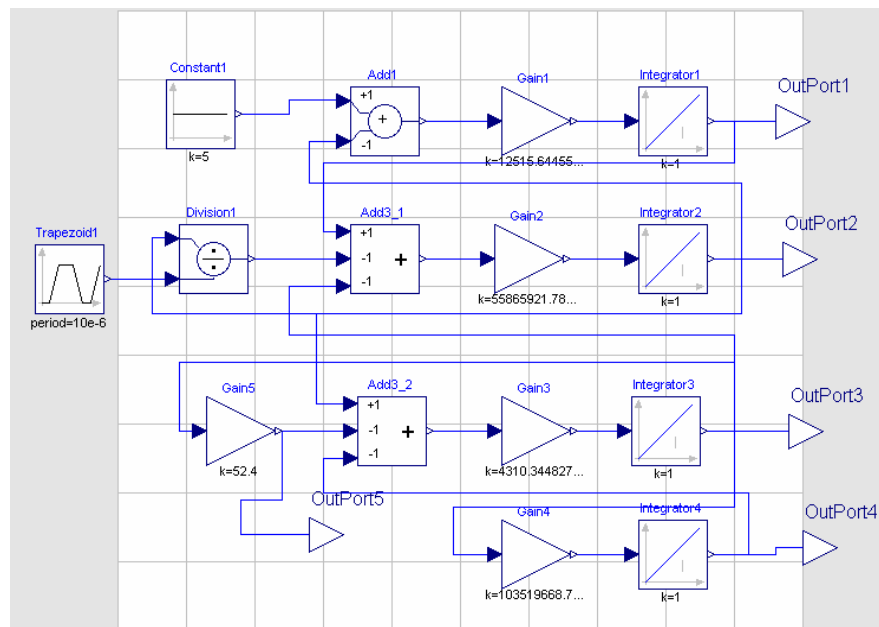


Figure 3.13 The Model of the system using Dymola

Table 3-2 Eigenvalues of R(t) (dymola)

Eigenvalues R(t) OFF	Eigenvalues R(t) ON
-54697.1634644282 +1040801.10427323i	-1117317558.64085
-54697.1634644282 -1040801.10427323i	-625.800550896034
-58228.2500355719 +532753.338849395i	-113038.779297539 +658348.699798618i
-58228.2500355719 -532753.338849395i	-113038.779297539 -658348.699798618i

Solutions

Task a Calculation of eigenvalues

To calculate the eigenvalues of the system when R(t) is on = 0,05 ohm and when R(t) is off = 5e+6 ohm by using the function “eigenValues” (included in Modelica standard library 2.2). It took 0,5s to execute task a. The result of eigenvalues is shown in Table 3-2.

Task b Simulation of the system

To simulate the system, using Dassl as an algorithm integration, 1000 as number of intervals, 0...100e-6s as simulation time interval and 1e-4 as a tolerance integration. Under the initial state zero, the result for variable current switch resistor IR(t) and output voltage VL is given by figure 3.14. It took 0,047s to simulate the task b.

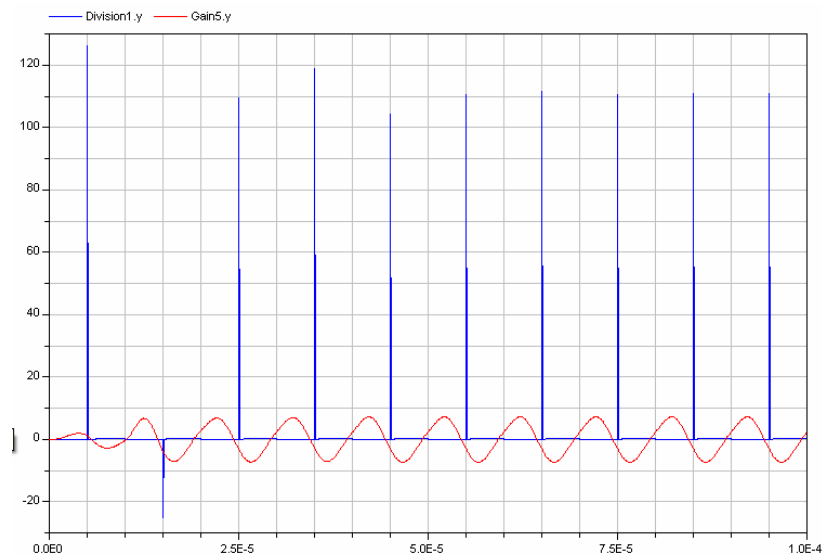
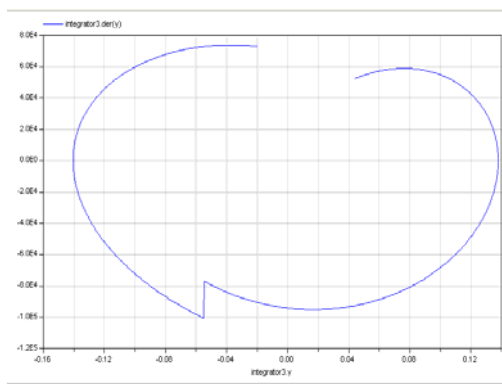


Figure 3.14 The result for variable current switch resistor IR(t) and output voltage VL(dymola)

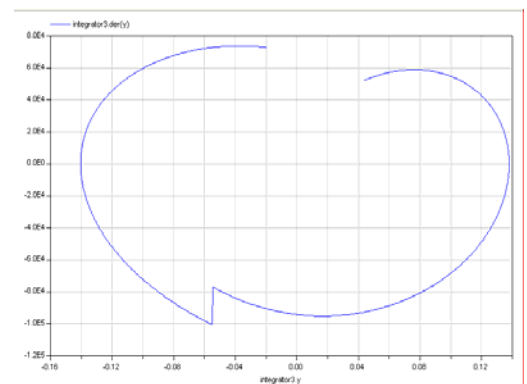
Task c Parameter Variation Study

The parameter of TRF is varied between $1e-15s$, $1e-11s$, $1e-9s$, $1e-7s$. Initial state for the task c is equal from the final solution is given by task b. The time interval is $0 \dots 9e-6s$. As result, The phase plane curves $dx_3/dt=VL_3$ as a function of $x_3 = IL_3$ is given by figure 3.15. It took $0,025s$ to simulate task c. The four simulation executed separately.

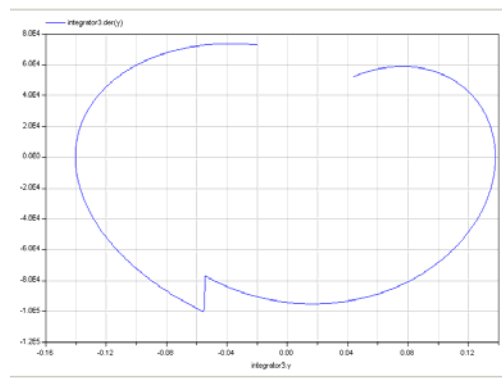
For the complete calculation and simulation, using Dymola version 6.0b on PC Intel Pentium D, 2 x 2,66 GHz.



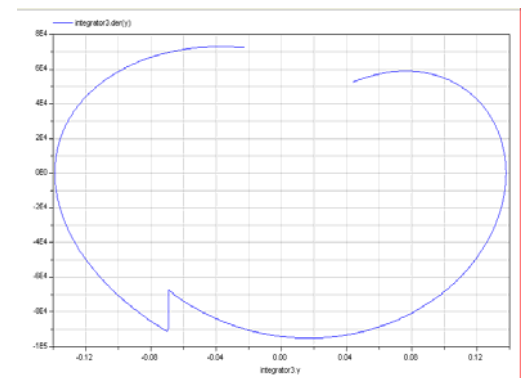
(a)



(b)



(c)



(d)

Figure 3.15 The phase plane curves $dx_3/dt=VL_3$ as a function of $x_3 = IL_3$

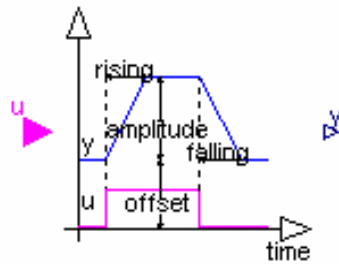
(dymola)

(a). $1e-15s$ (b). $1e-11s$ (c) $1e-9s$ (d) $1e-7s$

3.2.3.2 Stategraph Model

Design of Model

The model has 3 parts: Controller, time dependent resistor and differential equation. The time dependent resistor is built by block triggered trapezoid and shown by figure 3.16.



- | | |
|----------------------|---------------------|
| - Amplitude = $5e+6$ | - Offset = $5e-2$ |
| - Rising = $1e-15$ | - Falling = $1e-15$ |

Figure 3.16 Triggers Trapezoid

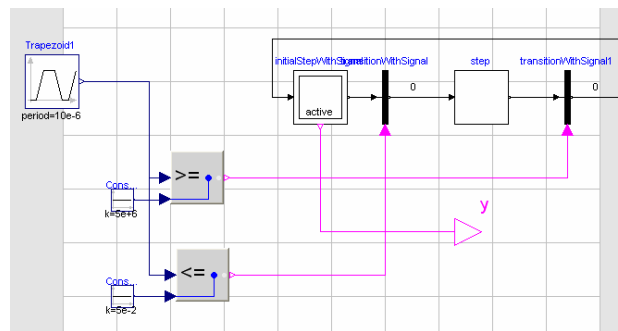


Figure 3.17 Part Controller of the system (stategraph)

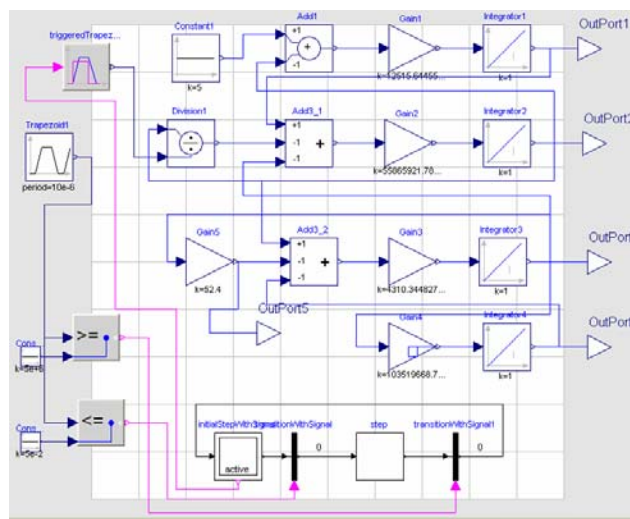


Figure 3.18 The Model of the system using Dymola stategraph mode

Part controller using stategraph mode to control time dependent resistor block, when state is off and when state is on. Controller is built by trapezoid source, greater equal, less equal, initial step, step and transition block. Figure 3.17 shows part controller of the system. The differential equation part is built by block integrator, add/subtract, gain, division and constant. Figure 3.18 shows the model of the system using Dymola stategraph mode

Solutions

Task b Simulation of the system

To simulate the system, using Dassl as an algorithm integration, 1000 as number of intervals, $0 \dots 100 \times 10^{-6}$ s as simulation time interval and 1×10^{-4} as a tolerance integration. Under the initial state zero, the result for variable current switch resistor $I_R(t)$ and output voltage V_L is given by figure 3.14. It took 0,063s to simulate the task b

Task c Parameter Variation Study

The parameter of TRF is varied between 1×10^{-15} s, 1×10^{-11} s, 1×10^{-9} s, 1×10^{-7} s. Initial state for the task c is equal from the final solution is given by task b. The time interval is $0 \dots 9 \times 10^{-6}$ s. As result, The phase plane curves $dx_3/dt = V_L$ as a function of $x_3 = I_L$ is shown in figure 3.15. It took 0,047s to simulate task c. The four simulation executed separately.

For all the calculation and simulation, using Dymola version 6.0b on PC Intel Pentium D, 2 x 2,66 GHz.

3.2.3.4. Electrical Model

Design of Model

The model built based on electrical model as given by figure 3.1. Time dependent resistor block is built by variable resistor and trapezoid source as input of variable resistor. Electrical model for comparison 3 is given by figure 3.19.

Solutions

Task b Simulation of the system

To simulate the system, using Dassl as an algorithm integration, 1000 as number of intervals, $0 \dots 100 \times 10^{-6}$ s as simulation time interval and 1×10^{-4} as a tolerance integration. Under the initial state zero, the result for variable current switch resistor $IR(t)$ and output voltage VL is given by figure 3.14. It took 0,047s to simulate the task b

Task c Parameter Variation Study

The parameter of TRF is varied between 1×10^{-15} s, 1×10^{-11} s, 1×10^{-9} s, 1×10^{-7} s. Initial state for the task c is equal from the final solution is given by task b. The time interval is $0 \dots 9 \times 10^{-6}$ s. As result, The phase plane curves $dx_3/dt = VL_3$ as a function of $x_3 = IL_3$ is shown in figure 3.15. It took 0,015s to simulate task c. The four simulation executed separately.

For all the calculation and simulation, using Dymola version 6.0b on PC Intel Pentium D, 2 x 2,66 GHz.

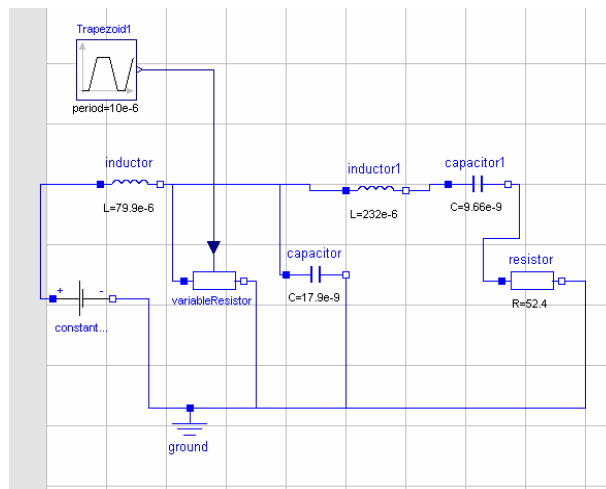


Figure 3.19 Electrical Model for Comparison 3.

3.2.3.4. Modelica Text Mode

Design of Model

For design of the model, using the exact differential equation with modelica function $\text{der}(x)$ as dx/dt in the equation. For time dependent resistor using algorithm as below:


```

equation
t_red = mod(time, 10E-6);
k=((5e+6)-(5e-2))/TRF;
algorithm
if
  (0<=t_red) and (t_red<TRF) then
    Rt:=(5e-2) + k*t_red;
elseif
  (TRF<=t_red) and (t_red<(5e-6)) then
    Rt:=5e+6;
elseif
  ((5e-6)<=t_red) and (t_red<((5e-6)+TRF)) then
    Rt:=(5e+6) - k*(t_red - (5e-6));
elseif
  ((5e-6)+TRF<=t_red) and (t_red<(10e-6)) then
    Rt:=5e-2;
else
  Rt:=-5;
end if;

```

Solutions

Task b Simulation of the system

To simulate the system, using Dassl as an algorithm integration, 1000 as number of intervals, 0...100e-6s as simulation time interval and 1e-4 as a tolerance integration. Under the initial state zero, the result for variable current switch resistor $IR(t)$ and output voltage VL is given by figure 3.14. It took 0,047s to simulate the task b

Task c Parameter Variation Study

The parameter of TRF is varied between 1e-15s, 1e-11s, 1e-9s, 1e-7s. Initial state for the task c is equal from the final solution is given by task b. The time interval is 0...9e-6s. As result, The phase plane curves $dx_3/dt=VL_3$ as a function of $x_3 = IL_3$ is shown in figure 3.15. It took 0,031s to simulate task c. The four simulation executed separately.

For all the calculation and simulation, using Dymola version 6.0b on PC Intel Pentium D, 2 x 2,66 GHz.

3.2.4 Mosilab

3.2.4.1. Modelica Text Mode

Design of Model

For design of the model, using the exact differential equation with modelica function $\text{der}(x)$ as dx/dt in the equation. For time dependent resistor using algorithm as below:

```

equation
t_red = mod(time, 10E-6);
k=((5e+6)-(5e-2))/TRF;
algorithm
if
  (0<=t_red) and (t_red<TRF) then
    Rt:=(5e-2) + k*t_red;
  elseif
    (TRF<=t_red) and (t_red<(5e-6)) then
    Rt:=5e+6;
  elseif
    ((5e-6)<=t_red) and (t_red<((5e-6)+TRF)) then
    Rt:=(5e+6) - k*(t_red - (5e-6));
  elseif
    ((5e-6)+TRF<=t_red) and (t_red<(10e-6)) then
    Rt:=5e-2;
  else
    Rt:=-5;
  end if;

```

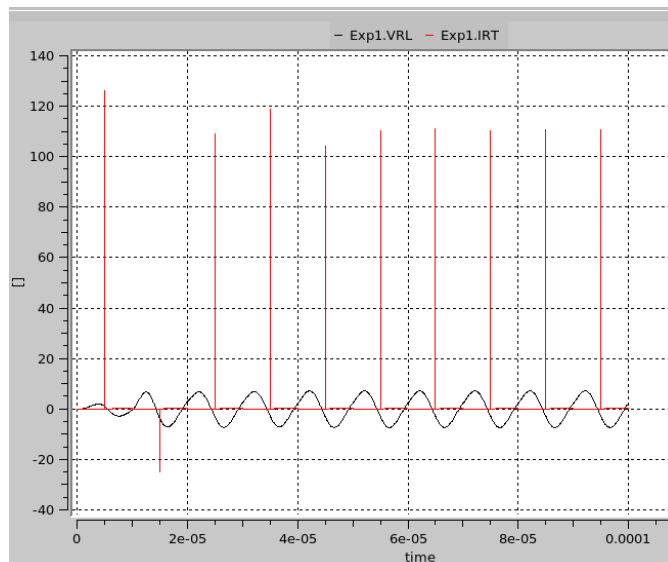


Figure 3.20 The result for variable current switch resistor $I_R(t)$ and output voltage $V_L(\text{mosilab})$

Solutions

Task a Calculation of eigenvalues

To calculate the eigenvalues of the system when $R(t)$ is on = 0,05 ohm and when $R(t)$ is off = $5e+6$ ohm can't be done by mosilab, because Mosilab didn't have "eigenValues" function in their modelica library.

Task b Simulation of the system

To simulate the system, using Dassl as an algorithm integration, $1e-10$ as min stepsize, $1e-7$ as max stepsize and $0...100e-6$ s as simulation time interval. Under the initial state zero, the result for variable current switch resistor $I_R(t)$ and output voltage V_L is given by figure 3.20. It took 1,3s to simulate the task b

Task c Parameter Variation Study

This task can't be done by mosilab because mosilab didn't have plotArray function in their core system.

For all the calculation and simulation, using Mosilab version 3.1 on Notebook Dell Latitude D630 Intel Centrino Duo.

3.2.4.2 StateChart

Design of Model

For design of the model, using the exact differential equation with modelica function $\text{der}(x)$ as dx/dt in the equation. For time dependent resistor using algorithm the same as previous. The code for statechart is written below:

```
equation
s1 = if Rt >= 5e+6 then true else false;
s2 = if Rt <= 5e-2 then true else false;
statechart
state C3MosilabStateSC extends State;
  annotation(extent=[-104,104; 44,-43]);
  State State1 annotation(extent=[-90,63; -77,59]);
  State State2 annotation(extent=[-51,62; -38,58]);
  State Initial (isInitial=true) annotation(extent=[-82,74; -80,72]);
  transition Initial->State1 action
    Rs:=5e+6;
  end transition annotation(points=[-82,72; -82,63]);
  transition State1->State2 event s2 action
    Rs:= 5e-2;
```

```

end transition annotation(points=[-77,59; -51,59]);
transition State2->State1 event s1 action
Rs:= 5e+6;
end transition annotation(points=[-51,60; -77,60]);
end C3MosilabStateSC;

```

Solutions

Task b Simulation of the system

To simulate the system, using Dassl as an algorithm integration, 1e-12 as min stepsize, 1e-9 as max stepsize and 0...100e-6s as simulation time interval. Under the initial state zero, the result for variable current switch resistor $I_R(t)$ and output voltage V_L is given by figure 3.20. It took 85s to simulate the task b

For all the calculation and simulation, using Mosilab version 3.1 on Notebook Dell Latitude D630 Intel Centrino Duo.

3.2.5 SimulationX

3.2.5.1 Hybrid Model

Design of Model

The differential equation part is built by block integrator, add/subtract, gain, function (as division) and signal generator(as constant).The time dependent resistor is built by type designer block using modelica code. The model of the system was shown by figure 3.21. The code for time dependent resistor was written below:

```

algorithm
if
  (0<=t_red) and (t_red<TRF) then
    Rt:=(5e-2) + k*t_red;
  elseif
    (TRF<=t_red) and (t_red<(5e-6)) then
    Rt:=5e+6;
  elseif
    ((5e-6)<=t_red) and (t_red<((5e-6)+TRF)) then
    Rt:=(5e+6) - k*(t_red - (5e-6));
  elseif
    ((5e-6)+TRF<=t_red) and (t_red<(10e-6)) then
    Rt:=5e-2;
  else
    Rt:=-5;
  end if;

```

equation

$$t_red = \text{mod}(\text{time}, 10E-6);$$

$$k = ((5e+6) - (5e-2)) / \text{TRF};$$

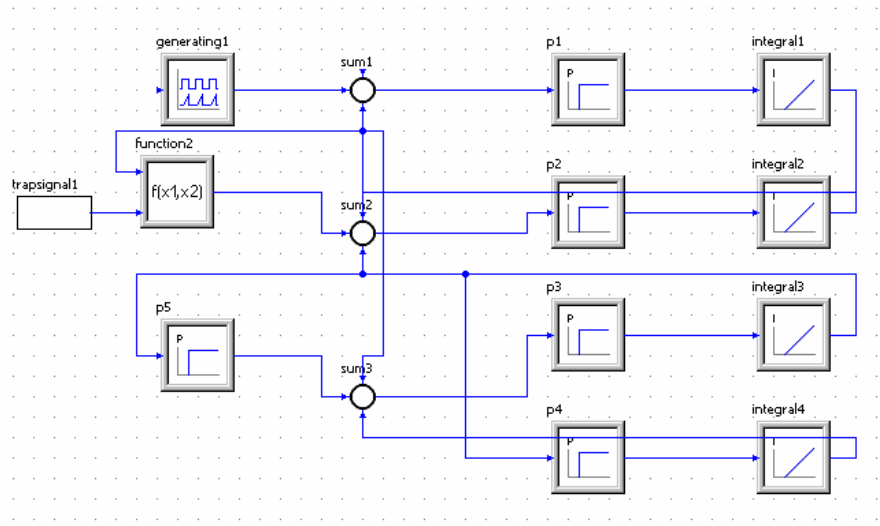


Figure 3.21. The model of the system (simulationX)

Solutions

Task a Calculation of eigenvalues

To calculate the eigenvalues of the system when $R(t)$ is on = 0,05 ohm and when $R(t)$ is off = $5e+6$ ohm by simulate the whole system first and then go to tab analysis(natural frequencies and mode shapes). In there simulationX automatically calculate the eigenvalue. It took 0,0723s to execute task a. The result of eigenvalues is in Table 3-3.

Table 3-3 Eigenvalues of $R(t)$ (simulationX)

Eigenvalues $R(t)$ OFF	Eigenvalues $R(t)$ ON
-54708+1,0408E+5i	-1,11731E+9
-54708-1,0408E+5i	-625,78
-58228+5,3275E+5i	-1,1304E+5 +6,5835E+5i
-58228+5,3275E+5i	-1,1304E+5 -6,5835E+5i

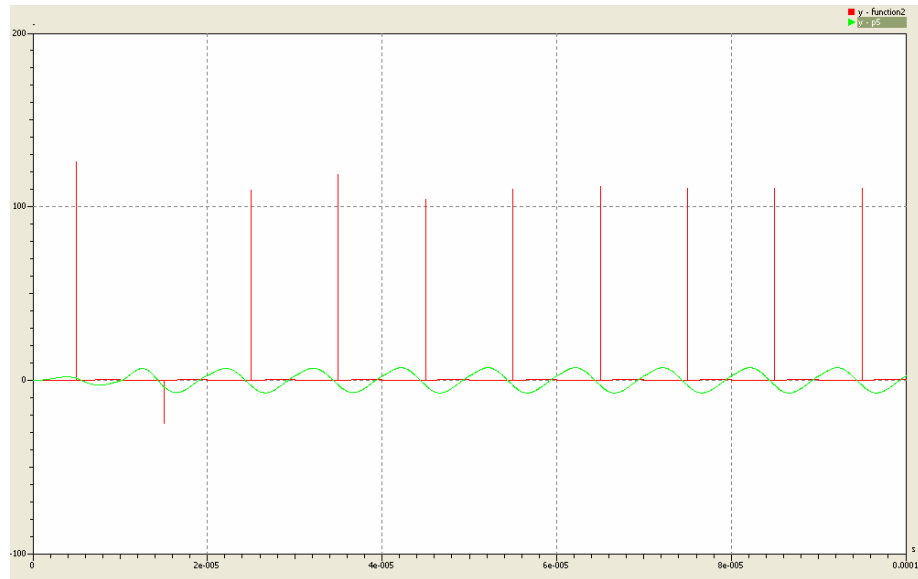
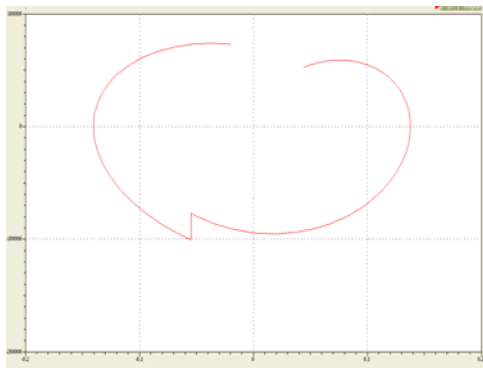
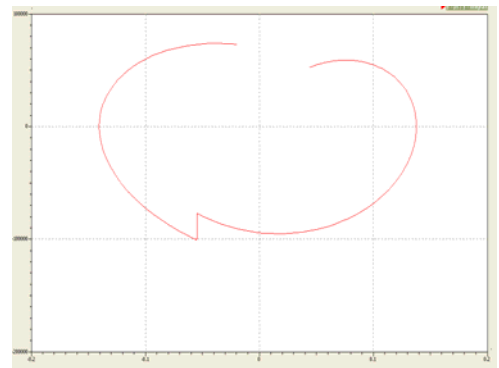


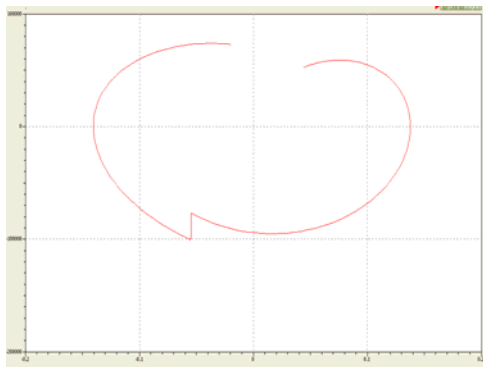
Figure 3.22 The result for variable current switch resistor $IR(t)$ and output voltage $VL(\text{simulationX})$



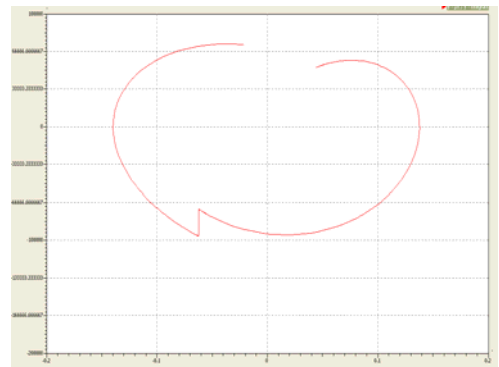
(a)



(b)



(c)



(d)

Figure 3.23 The phase plane curves $dx_3/dt = VL_3$ as a function of $x_3 = IL_3$

(a). $1e-15s$ (b). $1e-11s$ (c) $1e-9s$ (d) $1e-7s$

Task b Simulation of the system

To simulate the system, using BDF-Method as solver, $1e-18$ as min step size, $1e-15$ as min output step size, $1e-8$ as absolute tolerance, $0 \dots 100e-6s$ as simulation time interval and $1e-8$ as relative tolerance. Under the initial state zero, the result for variable current switch resistor $IR(t)$ and output voltage VL is given by figure 3.22. It took 1,2528s to simulate the task b.

Task c Parameter Variation Study

The parameter of TRF is varied $1e-15s$, $1e-11s$, $1e-9s$, $1e-7s$. Initial state for the task c is equal from the final solution is given by task b. The time interval is $0 \dots 9e-6s$. As result, The phase plane curves $dx_3/dt=VL_3$ as a function of $x_3 = IL_3$ is given by figure 3.23. It took 0,0858s to simulate task c. The four simulation executed separately.

For all the calculation and simulation, using SimulationX version 2.0 on PC Intel Pentium D, 2 x 2,66 GHz.

3.2.5.2 Electrical Model

Design of Model

The design of model based on figure 3.1, using resistor, inductor, capacitor and constant voltage as VDC. The time dependent resistor was built by type designer using modelica code. The model of the system was shown by figure 3.24. The code for time dependent resistor was written below:

```
equation
  v=pin1.v-pin2.v;
  v=R*i;
  pin1.i=i;
  pin2.i=-i;
  tred = mod(time, 10E-6);
  k=((5e+6)-(5e-2))/TRF;
  if (0<=tred) and (tred<TRF) then
    R=(5e-2) + k*tred;
  elseif (TRF<=tred) and (tred<(5e-6)) then
    R=5e+6;
  elseif ((5e-6)<=tred) and (tred<((5e-6)+TRF)) then
    R=(5e+6) - k*(tred - (5e-6));
  elseif ((5e-6)+TRF<=tred) and (tred<(10e-6)) then
    R=5e-2;
  else
```

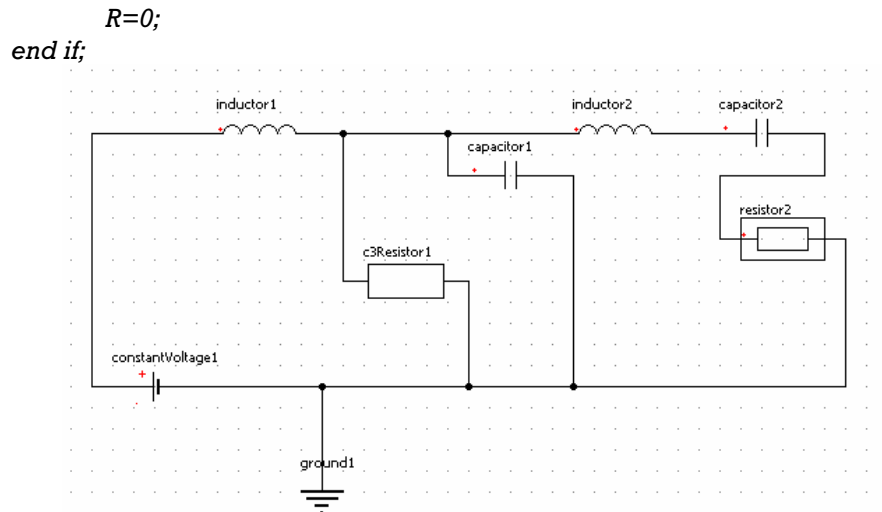


Figure 3.24. The model of the system (simulationX electrical)

Solutions

Task a Calculation of eigenvalues

To calculate the eigenvalues of the system when $R(t)$ is on = 0,05 ohm and when $R(t)$ is off = $5e+6$ ohm by simulate the whole system first and then go to tab analysis(natural frequencies and mode shapes). In there simulationX automatically calculate the eigenvalue. It took 0,0133s to execute task a. The result of eigenvalues is in Table 3-3.

Task b Simulation of the system

To simulate the system, using BDF-Method as solver, $1e-18$ as min step size, $1e-15$ as min output step size, $1e-8$ as absolute tolerance, $0...100e-6s$ as simulation time interval and $1e-8$ as relative tolerance. Under the initial state zero, the result for variable current switch resistor $IR(t)$ and output voltage VL is given by figure 3.22. It took 0,8461s to simulate the task b.

Task c Parameter Variation Study:

The parameter of TRF is varied between $1e-15s$, $1e-11s$, $1e-9s$, $1e-7s$. Initial state for the task c is equal from the final solution is given by task b. The time interval is $0...9e-6s$. As result, The phase plane curves $dx_3/dt=VL_3$ as a function of $x_3 = IL_3$ is given by figure 3.23. It took 0,11s to simulate task c. The four simulations executed separately.

For all the calculation and simulation, using SimulationX version 2.0 on PC Intel Pentium D, 2 x 2,66 GHz.

4. Comparison 5: Two State Model

4.1 Definition

In many engineering problems simulation models turn up to be discontinuous. That is, the solution itself is continuous, but either the first or higher order derivatives have jumps. Discontinuities may occur either at specific time points or when certain conditions are satisfied.

When a discontinuity has been passed, not only the model may be changed, but also the function that determines the location of the discontinuity. Consequently, if this discontinuity is not correctly modelled and determined, respectively, the results may go wrong qualitatively[11].

This example tests the ability of the simulator to handle discontinuities of the forementioned type in a satisfactory way. The problem is as follows

$$dy_1/dt = c_1 * (y_2 + c_2 - y_1)$$

$$dy_2/dt = c_3 * (c_4 - y_2)$$

This ODE system is essentially a simple linear stiff problem with exponential decays as analytical solution. One of these is a very rapid transient, and the stationary solution of the slow decay varies from the two states of the model. This actually "drives" the model (and the discontinuity).

Parameters c_1 and c_3 remain unchanged during simulation: $c_1 = 2.7E+6$, $c_3 = 3.5651205$.

The model operates in two states:

- The model is in state 1 when c_2 is 0.4 and c_4 is 5.5 (also the initial state). The initial values are $y_1(0) = 4.2$ and $y_2(0) = 0.3$. The model remains in state 1 as long as $y_1 < 5.8$. The choice of c_2 and c_4 ensures that y_1 will grow past 5.8.
- When the model switches to state 2, parameters c_2 and c_4 change to $c_2 = -0.3$ and $c_4 = 2.73$. The model remains in state 2 as long as $y_1 > 2.5$. When passing this instance the model switches back to state 1; the choice of c_2 and c_4 ensures that this will happen.

The time interval is 0 to 5.

The tasks to be performed are:

- a. Plot y_1 as function of time.
- b. Printout the time for every located discontinuity and the final value $y_1(5.0)$.
- c. Repeat question b) for the true relative accuracy varying between 10^{-6} , 10^{-10} , 10^{-14} . Analytical solution values can be found, so for comparison we state that the last discontinuity occurs at time 4.999999646 and the $y_1(5.0)$ value should be approximately 5.369. If the last discontinuity is not located, the previous ones are not found with adequate accuracy. The value of $y_1(5.0)$ also reflects the accuracy of the locations of the discontinuities and any value between 5.8 and 5.1 can be expected.
- d. Change the state 2 parameter values of c_2 to -1.25, c_4 to 4.33 and the condition to $y_1 > 4.1$ and rerun a) and b) with a true relative accuracy of 10^{-11} .

4.2. Design and Solutions

Block Diagram

The design of this model consists of only 3 block diagrams: Switching State, Controller and Differential Equation. Figure 4.1 shows block diagram for comparison 5.

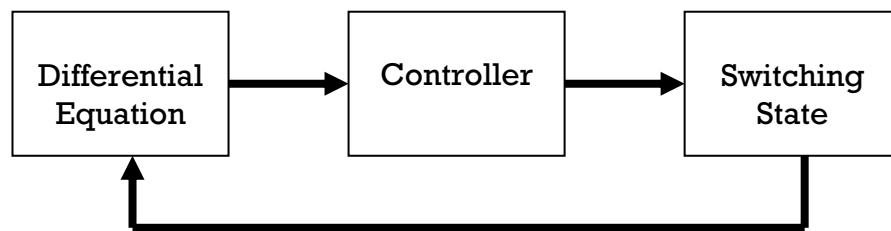


Figure 4.1 Block Diagram Comparison 5

4.2.1 Matlab

Design of Model

The model design in matlab algorithm ode15s is used to solve the system numerically and to calculate the times of the discontinuities by calling the solver's state event finder. Each time a switching event encountered, the integration of ODE solver stopped and the values of $c2$ and $c4$ were changed. Then the solver has to be restarted again at time of discontinuity. The state will switch back and forth until the time interval is reached. The differential equation code and events function are as follow:

```
function dydt = F(t, y, C)
dydt(1,1) = C(1) * (y(2) + C(2) - y(1));
dydt(2,1) = C(3) * (C(4) - y(2));
function [value, isterminal, direction] = events(t,y)
global p d
value = y(1) - [p;0];
isterminal = [1;1];
direction = [d;1];
```

Task a. Plot y1

To simulate the system, using matlab built in function ode15s (odesolver) with the solver form:

```
[t,y,te,ye,ie] = ode15s(@deq,[tstart tfinal],y0,options);
```

Where ode15s = ode solver matlab built in function

deq = differential equation of the model in form of function

[tstart tfinal] = simulation time interval

y0 = Initial condition of the model

options = an options structure that can pass as an argument to any of the ODE solvers

t = time solutions of the model

y = variable value solutions of the model

te = time event

ye = variable value event

ie = ith iteration event

The code is:

```
tstart = 0; tfinal = 5; y0 = [4.2 0.3]; C = [2.7E+6 0.4 3.5651205 5.5];
```

```

p=5.8; d=1;
options = odeset('reltol',1e-14,'Events',@events);
tout = tstart; yout = y0; teout = []; yeout = []; ieout = [];

while tout(length(tout))<5
% Call ODE Solver
FUN = @(t,y)F(t,y,C);
[t,y,te,ye,ie] = ode15s(FUN,[tstart tfinal],y0,options);
nt = length(t);
if y(nt)>=5.8
    p=2.5; d=-1; C = [2.7E+6 -0.3 3.5651205 2.73]; end
if y(nt)<=2.5
    p=5.8; d=1; C = [2.7E+6 0.4 3.5651205 5.5]; end
tout = [tout; t(2:nt)]; yout = [yout; y(2:nt,:)];
teout = [teout; te]; yeout = [yeout; ye];
ieout = [ieout; ie];
% Set the new initial conditions
y0=[y(nt,1) y(nt,2)];
tstart=t(nt);
options = odeset(options);
end

```

Plot y1 is given by figure 4.2. It took 0,890272s to simulate the task a

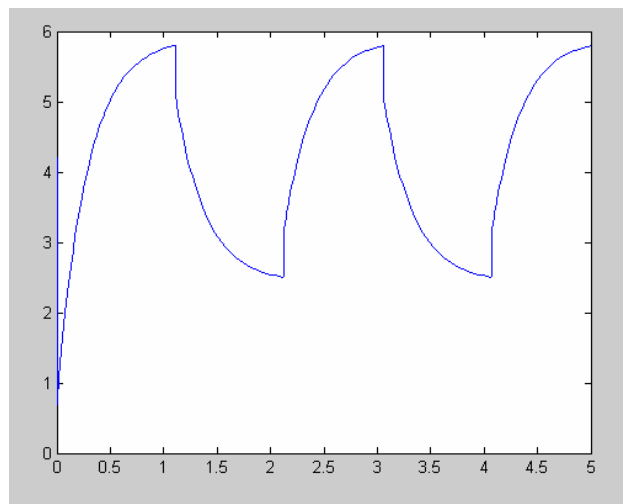


Figure 4.2 Plot y1(matlab)

Task b Time Discontinuity and Final Value of $y_1(5.0)$

The time discontinuity and the final value are:

$$t_0 = 1,1083 \quad t_1 = 2,1297$$

$$t_2 = 3,0542 \quad t_3 = 4,0756$$

$$y_1(5,0) = 5,8$$

Table 4-1 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (matlab)

Relative Tolerance	10^{-6}	10^{-10}	10^{-14}
t_0	1,1083	1,1083	1,1083
t_1	2,1297	2,1297	2,1297
t_2	3,0542	3,0542	3,0542
t_3	4,0756	4,0756	4,0756
$y_1(5,0)$	5,8000	5,8000	5,8000

Task c Time Discontinuity and Final Value of $y_1(5.0)$ with Different Relative Tolerance

The parameter of relative tolerance are varied between 10^{-6} , 10^{-10} and 10^{-14} while still using variable 0 ...5s as simulation time interval and ode15s as the solver. When relative tolerance 10^{-14} was used there is a warning message from matlab and matlab will then automatically set the relative tolerance to 2.22045e-014. Table 4-1 shows the result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance.

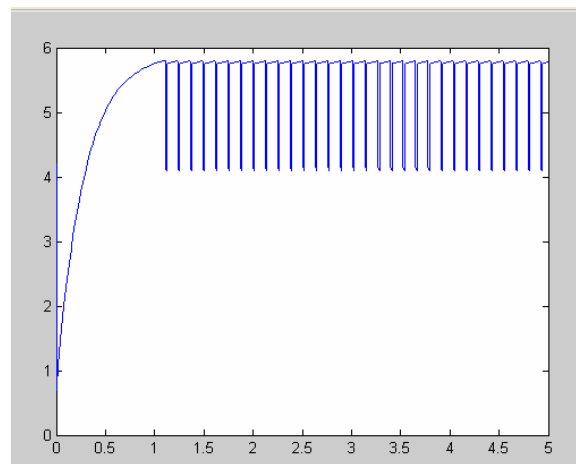


Figure 4.3 plot y_1 (task d) (matlab)

Task d Frequent Events

Changing the state 2 parameter values and switching condition will results in a high frequent event of discontinuity for y_1 with relative tolerance 1e-11. It took 10,789155s to simulate task d. Figure 4.3 shows plot y_1 (task d)

The time discontinuity and the final value are:

$t_0 = 1,1083$	$t_{13} = 1,8847$	$t_{26} = 2,7614$	$t_{39} = 3,5377$	$t_{52} = 4,4144$
$t_1 = 1,1217$	$t_{14} = 1,9984$	$t_{27} = 2,7748$	$t_{40} = 3,6515$	$t_{53} = 4,4278$
$t_2 = 1,2355$	$t_{15} = 2,0118$	$t_{28} = 2,8885$	$t_{41} = 3,6649$	$t_{54} = 4,5416$
$t_3 = 1,2489$	$t_{16} = 2,1256$	$t_{29} = 2,9019$	$t_{42} = 3,7786$	$t_{55} = 4,5550$
$t_4 = 1,3626$	$t_{17} = 2,1390$	$t_{30} = 3,0157$	$t_{43} = 3,7920$	$t_{56} = 4,6687$
$t_5 = 1,3760$	$t_{18} = 2,2527$	$t_{31} = 3,0291$	$t_{44} = 3,9058$	$t_{57} = 4,6822$
$t_6 = 1,4898$	$t_{19} = 2,2662$	$t_{32} = 3,1428$	$t_{45} = 3,9192$	$t_{58} = 4,7959$
$t_7 = 1,5032$	$t_{20} = 2,3799$	$t_{33} = 3,1563$	$t_{46} = 4,0329$	$t_{59} = 4,8093$
$t_8 = 1,6169$	$t_{21} = 2,3933$	$t_{34} = 3,2700$	$t_{47} = 4,0464$	$t_{60} = 4,9230$
$t_9 = 1,6304$	$t_{22} = 2,5070$	$t_{35} = 3,2834$	$t_{48} = 4,1601$	$t_{61} = 4,9365$
$t_{10} = 1,7441$	$t_{23} = 2,5205$	$t_{36} = 3,3972$	$t_{49} = 4,1735$	
$t_{11} = 1,7575$	$t_{24} = 2,6342$	$t_{37} = 3,4106$	$t_{50} = 4,2873$	
$t_{12} = 1,8713$	$t_{25} = 2,6476$	$t_{38} = 3,5243$	$t_{51} = 4,3007$	
$Y_1(5,0) = 5,7804$				

The whole calculation and simulation was done by using Matlab/Simulink version 7.4 R2007a on PC Intel Pentium D, 2 x 2,66 GHz.

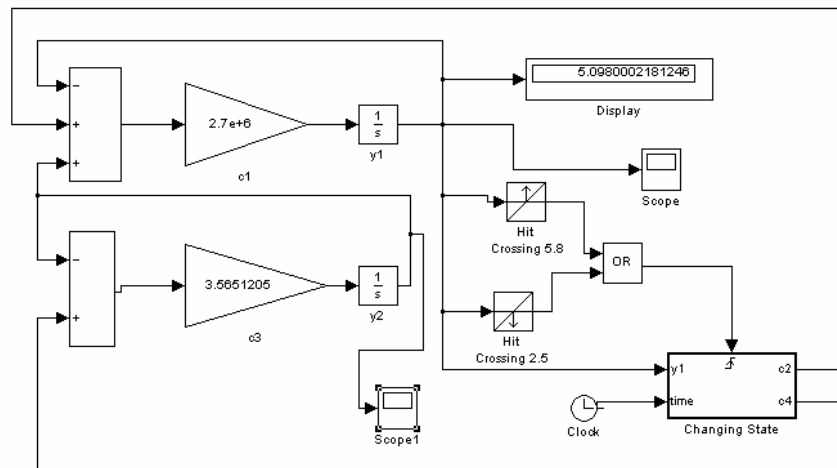
4.2.2 Simulink

4.2.2.1 Hybrid Model

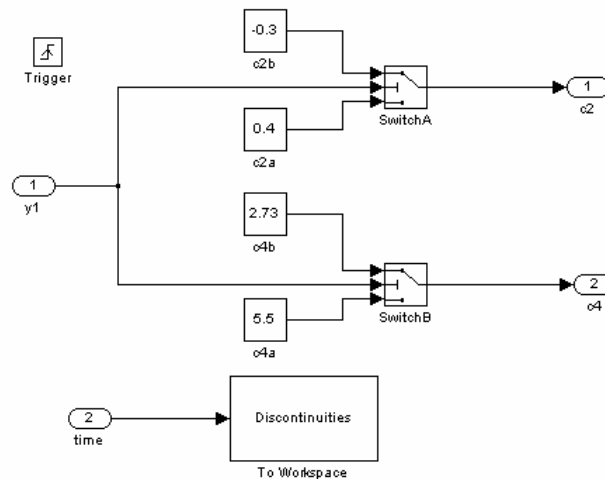
Design of Model

The model has 3 parts: Controller, Switching State and Differential equation. The task of controller is to control the signal that was sent to switching state to change the value c_2 and c_4 ; depending on which state is active. Part controller was built by using 2 hit crossing block and OR gate block. Part switching state was built by using clock and triggered subsystem block. 2 hit crossing blocks will be used for detection when value y_1 rises above 5,8 or falls below 2,5 otherwise the output is FALSE. In case of output TRUE, triggered subsystem is executed. The subsystem changes the value of c_2 and c_4 by using switches, which is different depending on the value of y_1 .

The Differential equation was built by Integrator, Gain and add/subtract block. Figure 4.4 shows the model of the system.



(a)



(b)

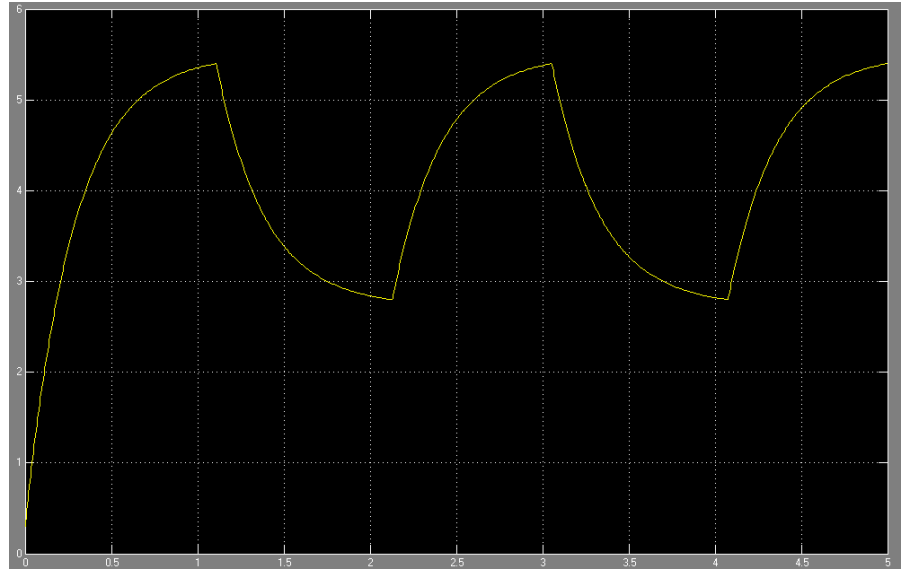
Figure 4.4 The model of the system (simulink)

(a) Main Model (b) Triggered subsystem

Solutions

Task a. Plot y_1

To simulate the system using variable step as a time step, 0 ...5s as simulation time interval, relative tolerance of 10^{-11} and ode23s (Stiff/Mod. Rosenbrock) as the solver. Under the initial state 4,2 for integrator y_1 and 0,3 for integrator y_2 , the result for value y_1 is given in figure 4.5. It took 0,5s to simulate the task a.

Figure 4.5 Plot $y_1(\text{simulink})$ **Task b Time Discontinuity and Final Value of $y_1(5.0)$**

The time discontinuity and the final value are:

$$t_0 = 2,1204\text{e-}007 \quad t_1 = 1,1083 \quad t_2 = 2,1296$$

$$t_3 = 3,054 \quad t_4 = 4,0754 \quad t_5 = 4,9998$$

$$y_1(5,0) = 5,0980002181246$$

Table 4-2 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (simulink)

Relative Tolerance	10^{-6}	10^{-10}	10^{-14}
t_0	2,1204e-007	2,1204e-007	2,1204e-007
t_1	1,1082	1,1083	1,1083
t_2	2,1294	2,1296	2,1296
t_3	3,0538	3,054	3,054
t_4	4,075	4,0754	4,0754
t_5	4,9994	4,9998	4,9998
$y_1(5,0)$	5,0940204247788	5,0979970081424	5,0980106702

Task c Time Discontinuity and Final Value of $y_1(5.0)$ with Different Relative Tolerance

The parameter of relative tolerance is varied between 10^{-6} , 10^{-10} and 10^{-14} while still using variable step as a time step, 0 ...5s as simulation time interval and ode23s (Stiff/Mod. Rosenbrock) as the solver. When relative

tolerance 10^{-14} was used and simulink was automatically set the relative tolerance to $2.842170943040401e-014$ there is a warning message from matlab. Table 4-2 shows the result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance.

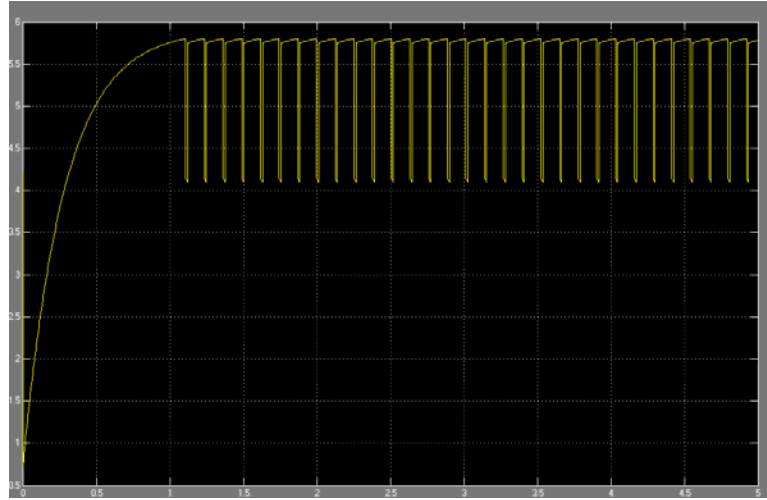


Figure 4-6 plot y_1 (task d)(simulink)

Task d Frequent Events

Changing the state 2 parameter values and switching condition will results in high frequent event of discontinuity for y_1 . It took 3s to simulate task d. Figure 4.6 shows plot y_1 (task d)

The time discontinuity and the final value are:

$t_0 = 9,6205e-009$	$t_{13} = 1,8712$	$t_{26} = 2,6475$	$t_{39} = 3,5241$	$t_{52} = 4,3004$
$t_1 = 1,1083$	$t_{14} = 1,8846$	$t_{27} = 2,7612$	$t_{40} = 3,5375$	$t_{53} = 4,4141$
$t_2 = 1,1217$	$t_{15} = 1,9983$	$t_{28} = 2,7746$	$t_{41} = 3,6513$	$t_{54} = 4,4276$
$t_3 = 1,2354$	$t_{16} = 2,0117$	$t_{29} = 2,8884$	$t_{42} = 3,6647$	$t_{55} = 4,5413$
$t_4 = 1,2488$	$t_{17} = 2,1255$	$t_{30} = 2,9018$	$t_{43} = 3,7784$	$t_{56} = 4,5547$
$t_5 = 1,3626$	$t_{18} = 2,1389$	$t_{31} = 3,0155$	$t_{44} = 3,7918$	$t_{57} = 4,6684$
$t_6 = 1,376$	$t_{19} = 2,2526$	$t_{32} = 3,0289$	$t_{45} = 3,9055$	$t_{58} = 4,6819$
$t_7 = 1,4897$	$t_{20} = 2,266$	$t_{33} = 3,1427$	$t_{46} = 3,919$	$t_{59} = 4,7956$
$t_8 = 1,5031$	$t_{21} = 2,3798$	$t_{34} = 3,1561$	$t_{47} = 4,0327$	$t_{60} = 4,809$
$t_9 = 1,6169$	$t_{22} = 2,3932$	$t_{35} = 3,2698$	$t_{48} = 4,0461$	$t_{61} = 4,9227$
$t_{10} = 1,6303$	$t_{23} = 2,5069$	$t_{36} = 3,2832$	$t_{49} = 4,1598$	$t_{62} = 4,9362$
$t_{11} = 1,744$	$t_{24} = 2,5203$	$t_{37} = 3,397$	$t_{50} = 4,1733$	
$t_{12} = 1,7574$	$t_{25} = 2,6341$	$t_{38} = 3,4104$	$t_{51} = 4,287$	
$Y_1(5,0) = 5,0980002181246$				

4.2.2.2 Stateflow

Design of Model

The model has 3 parts: Controller, Switching State and Differential equation. The task of controller is to control the signal that was sent to the switching state to change the value c_2 and c_4 , depending on which state is active. Part controller was built by using 2 hit crossing blocks and OR gate block. Part switching state was built by clock and triggered subsystem block. 2 hit crossing blocks are used for detection when value y_1 rises above 5,8 or falls below 2,5 otherwise the output is FALSE. In case of output TRUE, triggered subsystem will be executed. The subsystem changes the value of SGN by using switches, which is differed depending on the value of y_1 . Value SGN is used as a switch event of stateflow block. Stateflow block changes the value of c_2 and c_4 , which is differed depending on value SGN. The differential equation was built by Integrator, Gain and add/substract block. Figure 4.7 shows stateflow block of the system. Figure 4.8 shows the model of the system.

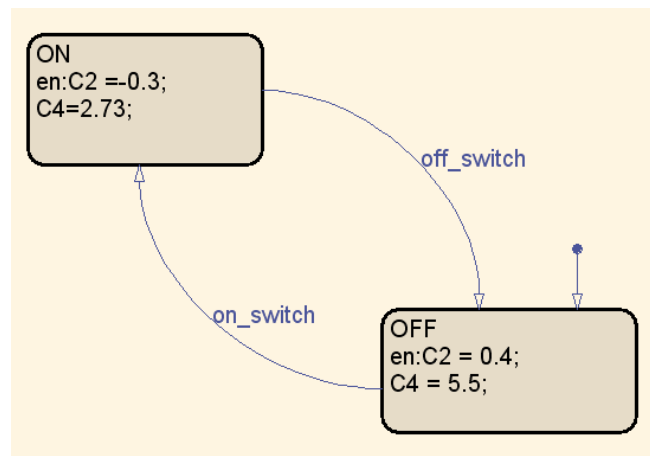
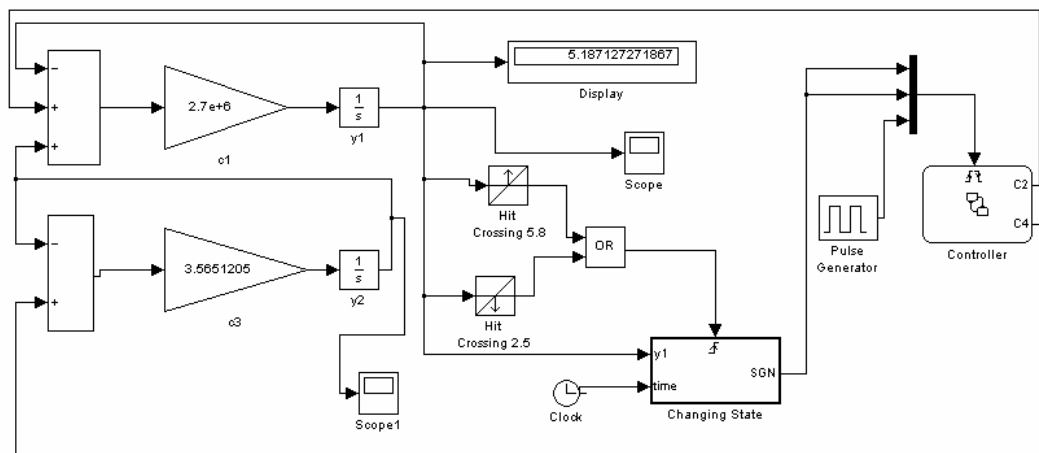
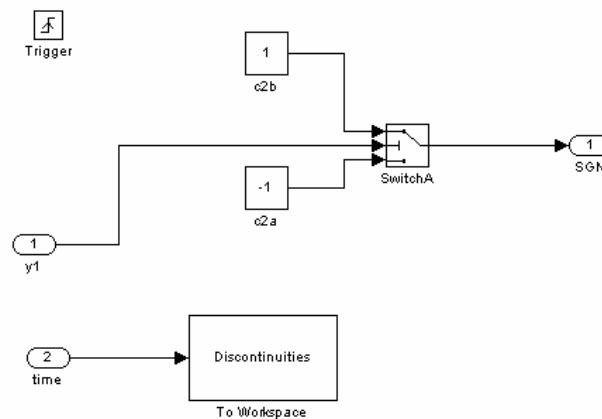


Figure 4.7 Stateflow block of the system (comparison 5)



(a)



(b)

Figure 4.8 The model of the system (simulink stateflow)

(a) Main Model (b) Triggered subsystem

Solutions

Task a. Plot y_1

To simulate the system, using variable step as a time step, 0 ...5s as simulation time interval, relative tolerance of 10^{-11} and ode23s (Stiff/Mod. Rosenbrock) as the solver. Under the initial state 4,2 for integrator y_1 and 0,3 for integrator y_2 , the result for value y_1 is shown in figure 4.9. It took 9s to simulate the task a.

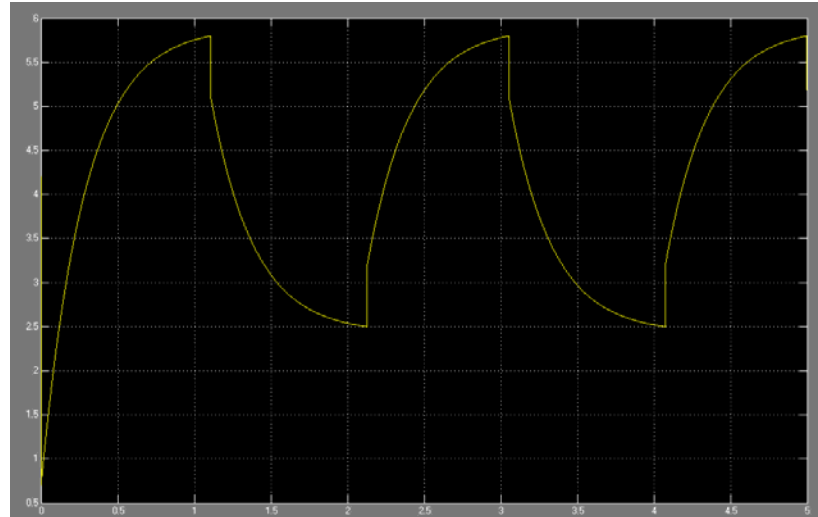


Figure 4.9 Plot y_1 (simulink stateflow)

Task b Time Discontinuity and Final Value of $y_1(5.0)$

The time discontinuity and the final value are:

$$\begin{aligned}
 t_0 &= 2,1204\text{e-}007 & t_1 &= 1,1083 & t_2 &= 2,1297 \\
 t_3 &= 3,0542 & t_4 &= 4,0755 & t_5 &= 5 \\
 y_1(5,0) &= 5,187127271867
 \end{aligned}$$

Table 4-3 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (simulink stateflow)

Relative Tolerance	10^{-6}	10^{-10}	10^{-14}
t_0	2,1204e-007	2,1204e-007	2,1204e-007
t_1	1,1083	1,1083	1,1083
t_2	2,1297	2,1297	2,1297
t_3	3,0542	3,0542	3,0542
t_4	4,0755	4,0755	4,0755
t_5	5	5	5
$y_1(5,0)$	5,1871018782202	5,1870844463401	5,1871273654863

Task c Time Discontinuity and Final Value of $y_1(5.0)$ with Different Relative Tolerance

The parameter of relative tolerance is varied between 10^{-6} , 10^{-10} and 10^{-14} , while still using variable step as a time step, 0 ...5s as simulation time interval and ode23s (Stiff/Mod. Rosenbrock) as the solver. When relative

tolerance 10^{-14} was used and simulink was automatically set the relative tolerance to 2.842170943040401e-014. Table 4-3 shows the result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance.

Task d Frequent Events

Changing the state 2 parameter values and switching condition will result in a high frequent event of discontinuity for y_1 . It took 14s to simulate task d. Figure 4.10 shows plot y_1 (task d)

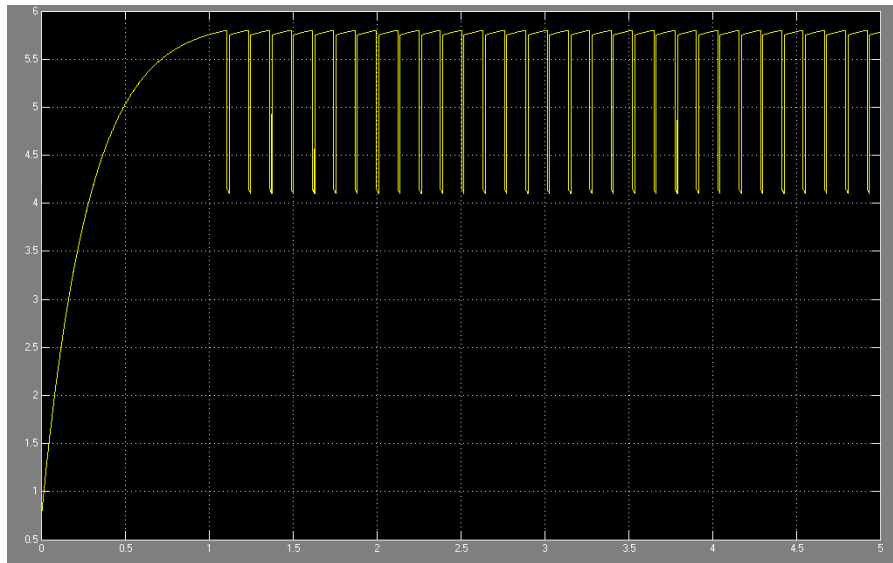


Figure 4.10 plot y_1 (task d) (simulink stateflow)

The time discontinuity and the final value are:

$t_0 = 9,6205e-009$	$t_{13} = 1,8713$	$t_{26} = 2,6476$	$t_{39} = 3,5243$	$t_{52} = 4,3007$
$t_1 = 1,1083$	$t_{14} = 1,8847$	$t_{27} = 2,7614$	$t_{40} = 3,5377$	$t_{53} = 4,4144$
$t_2 = 1,1217$	$t_{15} = 1,9984$	$t_{28} = 2,7748$	$t_{41} = 3,6515$	$t_{54} = 4,4278$
$t_3 = 1,2355$	$t_{16} = 2,0118$	$t_{29} = 2,8885$	$t_{42} = 3,6649$	$t_{55} = 4,5416$
$t_4 = 1,2489$	$t_{17} = 2,1256$	$t_{30} = 2,9019$	$t_{43} = 3,7786$	$t_{56} = 4,555$
$t_5 = 1,3626$	$t_{18} = 2,139$	$t_{31} = 3,0157$	$t_{44} = 3,792$	$t_{57} = 4,6687$
$t_6 = 1,376$	$t_{19} = 2,2527$	$t_{32} = 3,0291$	$t_{45} = 3,9058$	$t_{58} = 4,6821$
$t_7 = 1,4898$	$t_{20} = 2,2661$	$t_{33} = 3,1428$	$t_{46} = 3,9192$	$t_{59} = 4,7959$
$t_8 = 1,5032$	$t_{21} = 2,3799$	$t_{34} = 3,1563$	$t_{47} = 4,0329$	$t_{60} = 4,8093$
$t_9 = 1,6169$	$t_{22} = 2,3933$	$t_{35} = 3,27$	$t_{48} = 4,0464$	$t_{61} = 4,923$
$t_{10} = 1,6304$	$t_{23} = 2,507$	$t_{36} = 3,2834$	$t_{49} = 4,1601$	$t_{62} = 4,9365$
$t_{11} = 1,7441$	$t_{24} = 2,5205$	$t_{37} = 3,3971$	$t_{50} = 4,1735$	
$t_{12} = 1,7575$	$t_{25} = 2,6342$	$t_{38} = 3,4106$	$t_{51} = 4,2873$	
$Y_1(5,0) = 5,7804027877939$				

For all the calculation and simulation, using Matlab/Simulink version 7.4 R2007a on PC Intel Pentium D, 2 x 2,66 GHz.

4.2.3. Dymola

4.2.3.1 Hybrid Model

Design of Model

The model has 3 parts: Controller, Switching State and Differential equation. The task of controller is to control the signal that was sent to the switching state to change the value c_2 and c_4 depending on which state is active. Part controller was built by greater equal threshold block, less equal threshold block, OR gate block and triggered sampler block. Switching State was built by constant, switch and greater equal block. Greater equal threshold block and less equal threshold block are used to detect whether value y_1 rises above 5,8 or falls below 2,5, for otherwise the output is FALSE. In case of output TRUE, triggered sampler is activated, changing the value of c_2 and c_4 by using switches that is different depending on value of y_1 .

The Differential equation is built by Integrator, Gain and add/subtract block. Figure 4.11 shows the model of the system.

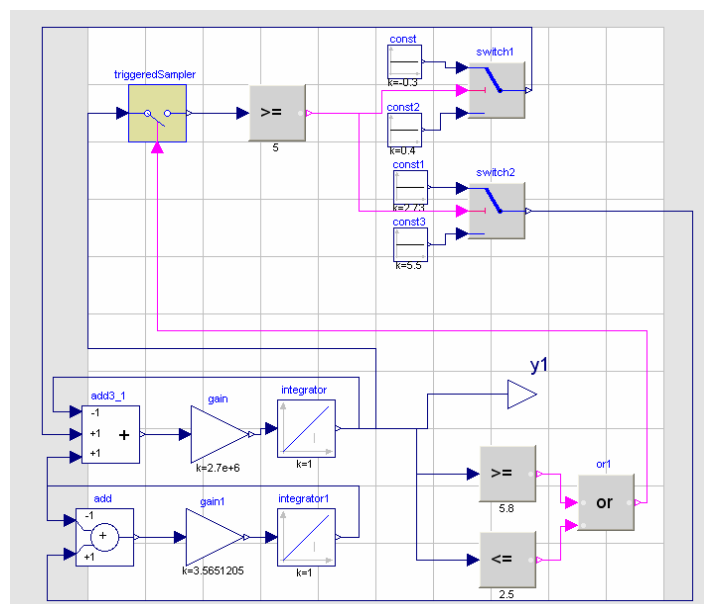


Figure 4.11 The model of the system (dymola)

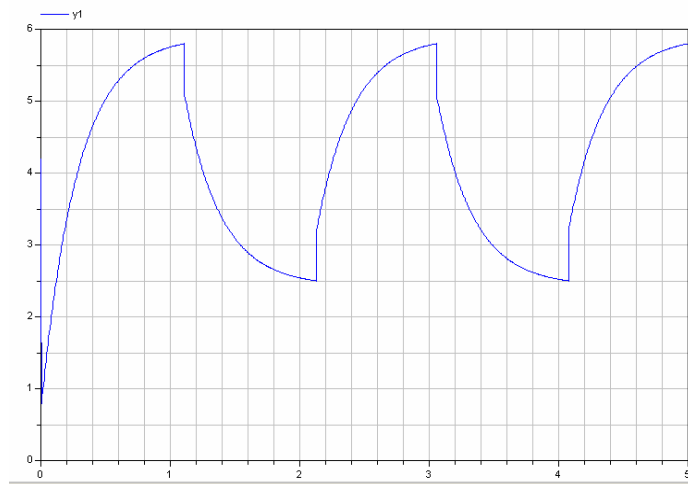


Figure 4.12 Plot y_1 (dymola)

Solutions

Task a. Plot y_1

To simulate the system, using 1000 as number of intervals, 0 ...5s as simulation time interval, relative tolerance of 10^{-11} and Dassl as the solver. Under the initial state 4,2 for integrator y_1 and 0,3 for integrator y_2 , the result for value y_1 shown in figure 4.12. It took 0,047s to simulate the task a.

Task b Time Discontinuity and Final Value of $y_1(5.0)$

The time discontinuity and the final value are:

$$t_0 = 2,46288e-007 \quad t_1 = 1,10831 \quad t_2 = 2,12968$$

$$t_3 = 3,05415 \quad t_4 = 4,07553 \quad t_5 = 5$$

$$y_1(5,0) = 5,37114$$

Task c Time Discontinuity and Final Value of $y_1(5.0)$ with Different Relative Tolerance

The parameter of relative tolerance is varied between 10^{-6} , 10^{-10} and 10^{-14} , while still using 1000 as number of intervals, 0 ...5s as simulation time interval and Dassl as the solver. There is an error message from dymola when relative tolerance 10^{-14} was used because it is unable to do the task, therefore 10^{-12} will be used as the new relative tolerance instead. Table 4-4

shows the result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance.

Table 4-4 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (dymola)

Relative Tolerance	10^{-6}	10^{-10}	10^{-12}
t_0	2,46288e-007	2,46288e-007	2,46288e-007
t_1	1,10831	1,10831	1,10831
t_2	2,12969	2,12968	2,12968
t_3	3,05416	3,05415	3,05415
t_4	4,07555	4,07553	4,07553
t_5	5	5	5
$y_1(5,0)$	5,79999	5,38522	5,36955

Task d Frequent Events

Changing the state 2 parameter values and switching condition will result in a high frequent event of discontinuity for y_1 . It took 0,204s to simulate task d. Figure 4.13 shows plot y_1 (task d)

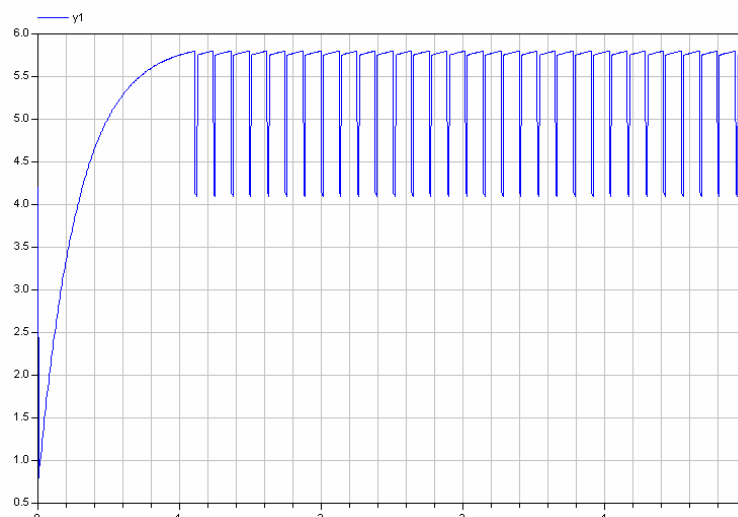


Figure 4.13 plot y_1 (task d) (dymola)

The time discontinuity and the final value are:

$t_0 = 1,07361e-008$	$t_{13} = 1,87125$	$t_{26} = 2,64762$	$t_{39} = 3,5243$	$t_{52} = 4,30068$
$t_1 = 1,10831$	$t_{14} = 1,88468$	$t_{27} = 2,76136$	$t_{40} = 3,53773$	$t_{53} = 4,41441$
$t_2 = 1,12173$	$t_{15} = 1,99841$	$t_{28} = 2,77478$	$t_{41} = 3,65146$	$t_{54} = 4,42783$
$t_3 = 1,23546$	$t_{16} = 2,01184$	$t_{29} = 2,88851$	$t_{42} = 3,66489$	$t_{55} = 4,54157$
$t_4 = 1,24889$	$t_{17} = 2,12557$	$t_{30} = 2,90194$	$t_{43} = 3,77862$	$t_{56} = 4,55499$
$t_5 = 1,36262$	$t_{18} = 2,13899$	$t_{31} = 3,01567$	$t_{44} = 3,79204$	$t_{57} = 4,66872$
$t_6 = 1,37605$	$t_{19} = 2,25273$	$t_{32} = 3,0291$	$t_{45} = 3,90578$	$t_{58} = 4,68215$
$t_7 = 1,48978$	$t_{20} = 2,26615$	$t_{33} = 3,14283$	$t_{46} = 3,9192$	$t_{59} = 4,79588$
$t_8 = 1,5032$	$t_{21} = 2,37988$	$t_{34} = 3,15625$	$t_{47} = 4,03294$	$t_{60} = 4,80931$
$t_9 = 1,61694$	$t_{22} = 2,39331$	$t_{35} = 3,26999$	$t_{48} = 4,04636$	$t_{61} = 4,92304$
$t_{10} = 1,63036$	$t_{23} = 2,50704$	$t_{36} = 3,28341$	$t_{49} = 4,16009$	$t_{62} = 4,93646$
$t_{11} = 1,7441$	$t_{24} = 2,52047$	$t_{37} = 3,39715$	$t_{50} = 4,17352$	
$t_{12} = 1,75752$	$t_{25} = 2,6342$	$t_{38} = 3,41057$	$t_{51} = 4,28725$	
$Y_1(5,0) = 5,7804$				

For all the calculation and simulation, using Dymola version 6.0b on PC Intel Pentium D, 2 x 2,66 GHz.

4.2.3.2 Stategraph Model

Design of Model

The model has 3 parts: Controller, Switching State and Differential equation. The task of controller is to send a signal to the switching state to change the value c_2 and c_4 depending on which state is active. Part controller was built by greater equal threshold block and less equal threshold block. Switching State was built by constant, switch and and stategraph block such as initial step, step and transition block. Greater equal threshold block and less equal threshold block are used to detect whether value y_1 rises above 5,8 or falls below 2,5, for otherwise the output is FALSE. In case of output TRUE, transition 1 block or transition 2 block will be activated depending on which threshold block send a TRUE output. Then the step block controls the value of c_2 and c_4 by using different switch depending on whether step block active or not. Figure 4.14 shows part controller and switching state of the system.

The Differential equation was built by Integrator, Gain and add/subtract block. Figure 4.15 shows the model of the system.

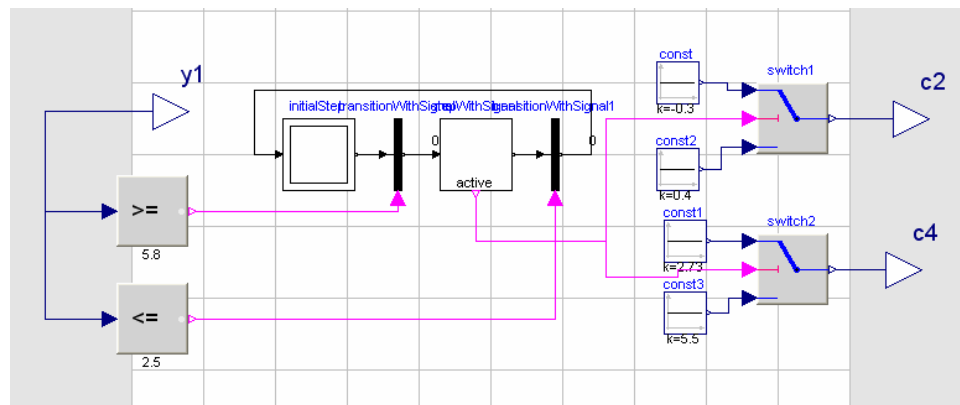


Figure 4.14 Part Controller and Switching State of the System (dymola stategraph)

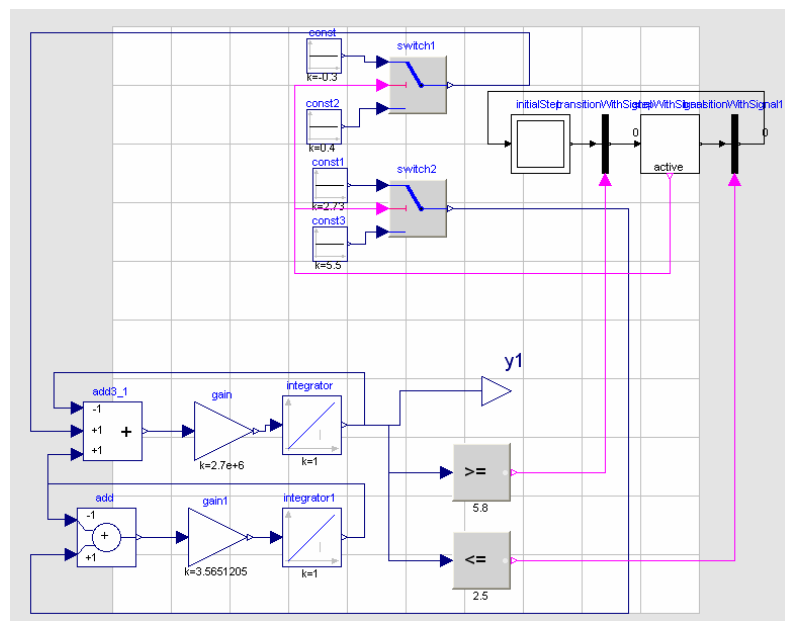


Figure 4.15 The model of the system (dymola stategraph)

Solutions

Task a. Plot y_1

To simulate the system, using 1000 as number of intervals, 0 ...5s as simulation time interval, relative tolerance of 10^{-11} and Dassl as the solver. Under the initial state 4,2 for integrator y_1 and 0,3 for integrator y_2 , the result for value y_1 is shown in figure 4.12. It took 0,062s to simulate the task a.

Task b Time Discontinuity and Final Value of $y_1(5.0)$

The time discontinuity and the final value are the same as dymola hybrid model:

Task c Time Discontinuity and Final Value of $y_1(5.0)$ with Different Relative Tolerance

The parameter of relative tolerance is varied between 10^{-6} , 10^{-10} and 10^{-14} while still 1000 as number of intervals, 0 ...5s as simulation time interval and Dassl as the solver. There is an error message from dymola when relative tolerance 10^{-14} was used because it is unable to do the task, therefore 10^{-12} will be used as the new relative tolerance instead. The result will be the same as the hybrid model shown by Table 4-4.

Task d Frequent Events

Changing the state 2 parameter values and switching condition will result in a high frequent event of discontinuity for y_1 . It took 0,25s to simulate task d. Figure 4.13 shows plot y_1 (task d)

The time discontinuity and the final value are the same as dymola hybrid model.

For all the calculation and simulation, using Dymola version 6.0b on PC Intel Pentium D, 2 x 2,66 GHz.

4.2.3.3. Modelica Text Mode

Design of Model

For design of the model, using the exact differential equation with modelica function $\text{der}(y)$ as dy/dt in the equation. For switching state using algorithm as below:

```
algorithm
  when (y1>=5.8) then
    c2:=-0.3;
    c4:=2.73;
  end when;
  when (y1<=2.5) then
    c2:=0.4;
```

```

c4:=5.5;
end when;

```

Solutions

Task a. Plot y_1

To simulate the system, using 1000 as number of intervals, 0 ...5s as simulation time interval, relative tolerance of 10^{-11} and Dassl as the solver. Under the initial state 4,2 for integrator y_1 and 0,3 for integrator y_2 , the result for value y_1 is shown in figure 4.12. It took 0,047s to simulate the task a.

Task b Time Discontinuity and Final Value of $y_1(5.0)$

The time discontinuity and the final value are the same as dymola hybrid model:

Task c Time Discontinuity and Final Value of $y_1(5.0)$ with Different Relative Tolerance

The parameter of relative tolerance is varied 10^{-6} , 10^{-10} and 10^{-14} . Still 1000 as number of intervals, 0 ...5s as simulation time interval and Dassl as the solver. There is an error message from dymola when relative tolerance 10^{-14} was used because it is unable to do the task, therefore 10^{-12} will be used as the new relative tolerance instead. The result will be the same as the hybrid model shown by Table 4-4.

Task d Frequent Events

Changing the state 2 parameter values and switching condition will result in a high frequent event of discontinuity for y_1 . It took 0,187s to simulate task d. Figure 4.13 shows plot y_1 (task d)

The time discontinuity and the final value are the same as dymola hybrid model.

For all the calculation and simulation, using Dymola version 6.0b on PC Intel Pentium D, 2 x 2,66 GHz.

4.2.4 Mosilab

4.2.4.1. Modelica Text Mode

Design of Model

For design of the model, using the exact differential equation with modelica function `der(y)` as dy/dt in the equation. For switching state using algorithm as below:

```
algorithm
  when (y1 >= 5.8) then
    c2 := -0.3;
    c4 := 2.73;
  end when;
  when (y1 <= 2.5) then
    c2 := 0.4;
    c4 := 5.5;
  end when;
```

Solutions

Task a. Plot y_1

To simulate the system, using $1e-6$ as min stepsize. 0,08 as max stepsize, relative tolerance of 10^{-6} and Dassl as the solver. Under the initial state 4,2 for integrator y_1 and 0,3 for integrator y_2 , the result for value y_1 is shown in figure 4.16. It took 0,1s to simulate the task a.

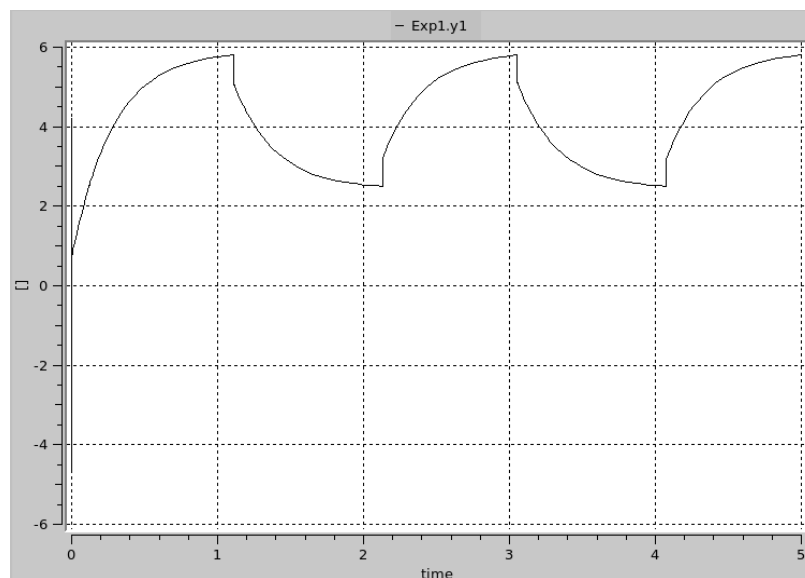


Figure 4.16 Plot y_1 (mosilab)

Task b Time Discontinuity and Final Value of $y_1(5.0)$

The time discontinuity and the final value are:

$$t_0 = 1,1088 \quad t_1 = 2,1397$$

$$t_2 = 3,0588 \quad t_3 = 4,0760$$

$$y_1(5,0) = 5,7988$$

Task c Time Discontinuity and Final Value of $y_1(5.0)$ with Different Relative Tolerance

The parameter of relative tolerance is varied 10^{-6} , 10^{-10} and 10^{-14} . Still using $1e-6$ as min stepsize. 0,08 as max stepsize and Dassl as the solver. Table 4-5 shows the result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance.

Table 4-5 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (mosilab)

Relative Tolerance	10^{-6}	10^{-10}	10^{-14}
t_0	1,1088	1,1083	1,1090
t_1	2,1397	2,1394	2,1299
t_2	3,0588	3,0584	3,0592
t_3	4,0760	4,0757	4,0764
$y_1(5,0)$	5,7988	5,7985	5,7997

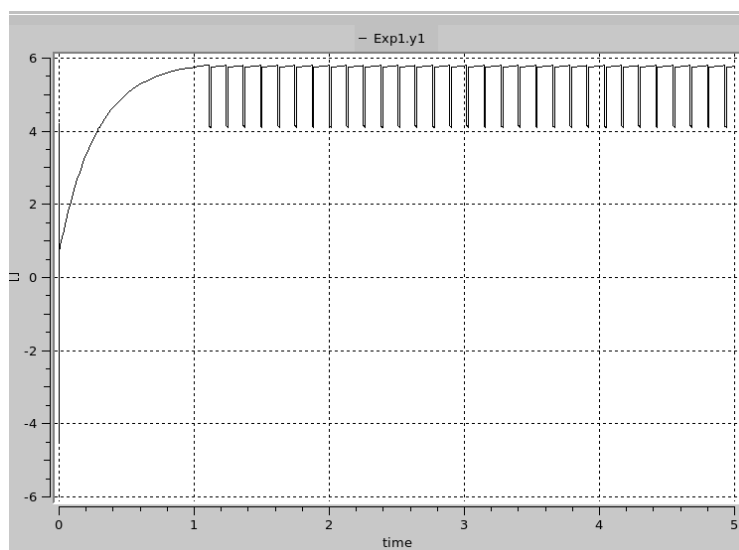


Figure 4.17 plot y_1 (task d) (mosilab)

Task d Frequent Events

Changing the state 2 parameter values and switching condition will result in a high frequent event of discontinuity for y_1 with relative tolerance $1e-11$. It took 1,3s to simulate task d. Figure 4.17 shows plot y_1 (task d)

The time discontinuity and the final value are:

$t_0 = 1,1088$	$t_{13} = 1,8852$	$t_{26} = 2,7618$	$t_{39} = 3,5382$	$t_{52} = 4,4150$
$t_1 = 1,1220$	$t_{14} = 1,9987$	$t_{27} = 2,7755$	$t_{40} = 3,6520$	$t_{53} = 4,4283$
$t_2 = 1,2357$	$t_{15} = 2,0121$	$t_{28} = 2,8890$	$t_{41} = 3,6655$	$t_{54} = 4,5419$
$t_3 = 1,2493$	$t_{16} = 2,1259$	$t_{29} = 2,9023$	$t_{42} = 3,7791$	$t_{55} = 4,5556$
$t_4 = 1,3629$	$t_{17} = 2,1397$	$t_{30} = 3,0161$	$t_{43} = 3,7925$	$t_{56} = 4,6694$
$t_5 = 1,3765$	$t_{18} = 2,2534$	$t_{31} = 3,0297$	$t_{44} = 3,9062$	$t_{57} = 4,6827$
$t_6 = 1,4899$	$t_{19} = 2,2669$	$t_{32} = 3,1433$	$t_{45} = 3,9197$	$t_{58} = 4,7963$
$t_7 = 1,5037$	$t_{20} = 2,3804$	$t_{33} = 3,1569$	$t_{46} = 4,0335$	$t_{59} = 4,8098$
$t_8 = 1,6174$	$t_{21} = 2,3939$	$t_{34} = 3,2705$	$t_{47} = 4,0469$	$t_{60} = 4,9235$
$t_9 = 1,6307$	$t_{22} = 2,5077$	$t_{35} = 3,2838$	$t_{48} = 4,1608$	$t_{61} = 4,9370$
$t_{10} = 1,7446$	$t_{23} = 2,5209$	$t_{36} = 3,3977$	$t_{49} = 4,1739$	
$t_{11} = 1,7578$	$t_{24} = 2,6349$	$t_{37} = 3,4111$	$t_{50} = 4,2878$	
$t_{12} = 1,8716$	$t_{25} = 2,6482$	$t_{38} = 3,5249$	$t_{51} = 4,3013$	

$Y_1(5,0) = 5,7827$

For all the calculation and simulation, using Mosilab version 3.1 on Notebook Dell Latitude D630 Intel Centrino Duo.

4.2.4.2 StateChart

Design of Model

For design of the model, using the exact differential equation with modelica function $\text{der}(y)$ as dy/dt in the equation. For switching state using statechart algorithm as below:

```

equation
s2 = if y1 >= 5.8 then true else false;
s1 = if y1 <= 2.5 then true else false;
statechart
state C5MosilabStateSC extends State;
  annotation(extent=[-104,105; 45,-43]);
  State State1 annotation(extent=[-90,63; -81,59]);
  State State2 annotation(extent=[-58,62; -45,58]);
  State Initial (isInitial=true) annotation(extent=[-82,74; -80,72]);

```

```

transition Initial->State1
end transition annotation(points=[-82,72; -82,63]);
transition State1->State2 event s2 action
c2:= -0.3; c4:= 2.73;
end transition annotation(points=[-81,59; -77,60; -58,60]);
transition State2->State1 event s1 action
c2:= 0.4; c4:= 5.5;
end transition annotation(points=[-58,59; -77,59; -81,59]);
end C5MosilabStateSC;

```

Solutions

Task a. Plot y_1

To simulate the system, using $1e-6$ as min stepsize. 0,08 as max stepsize, relative tolerance of 10^{-6} and Dassl as the solver. Under the initial state 4,2 for integrator y_1 and 0,3 for integrator y_2 , the result for value y_1 is shown in figure 4.16. It took 0,3s to simulate the task a.

Task b Time Discontinuity and Final Value of $y_1(5.0)$

The time discontinuity and the final value are the same as previous in the text mode

Task c Time Discontinuity and Final Value of $y_1(5.0)$ with Different Relative Tolerance

The parameter of relative tolerance is varied between 10^{-6} , 10^{-10} and 10^{-14} while still using $1e-6$ as min stepsize. 0,08 as max stepsize and Dassl as the solver. Table 4-4 shows the result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance.

Task d Frequent Events

Changing the state 2 parameter values and switching condition will result in a high frequent event of discontinuity for y_1 with relative tolerance $1e-11$. It took 2,3s to simulate task d. Figure 4.17 shows plot y_1 (task d)

The time discontinuity and the final value are the same as previous in the text mode

For all the calculation and simulation, using Mosilab version 3.1 on Notebook Dell Latitude D630 Intel Centrino Duo.

4.2.5 SimulationX

Hybrid Model

Design of Model

The model has 3 parts: Controller, Switching State and Differential equation. The task of controller is to control the signal that sent to switching state to change the value c_2 and c_4 that depend on which state active is. Part controller was built by single pass switch block, relational pass switch block, constant, add block and controlled event sample and hold block. Switching State was built by constant and single change switch. Single pass switch block and relational pass switch block are being used to detect whether value y_1 rises above 5,8 or falls below 2,5, for otherwise the output is ZERO. In case of output value of 5,8 or 2,5, controlled event sample and hold block will pass and hold this value, changing the value of c_2 and c_4 by using single change switches which is differ-depending on value of y_1 .

The Differential equation is built by Integrator, Gain and add/substract block. Figure 4.18 shows the model of the system.

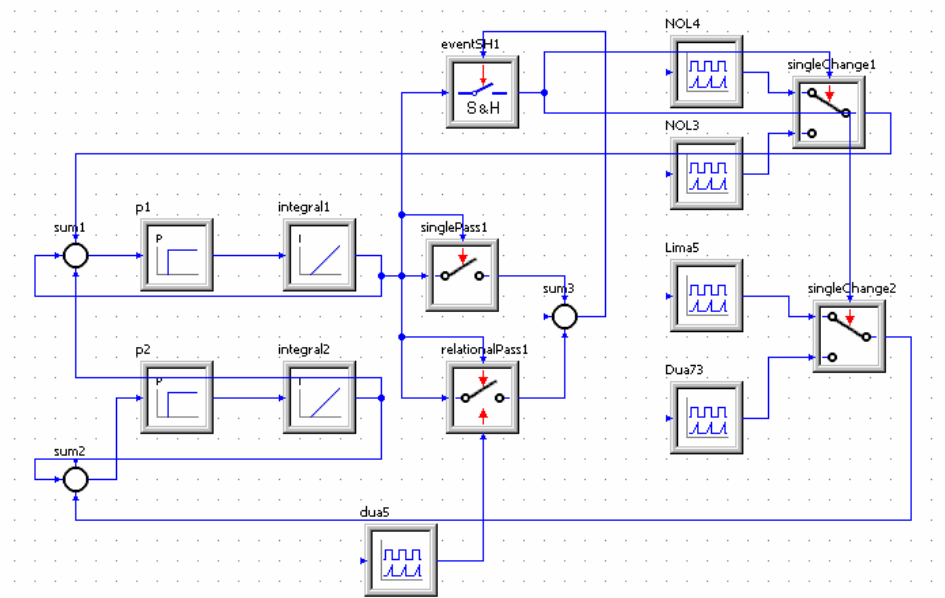


Figure 4.18 The model of the system (simulationX)

Solutions

Task a. Plot y_1

To simulate the system, using $1e-12$ as min stepsize, $1e-3$ as absolute tolerance, $0 \dots 5s$ as simulation time interval, relative tolerance of 10^{-6} and Dassl as the solver. Under the initial state 4,2 for integrator y_1 and 0,3 for integrator y_2 , the result for value y_1 shown in figure 4.19. It took 0,0582s to simulate the task a.

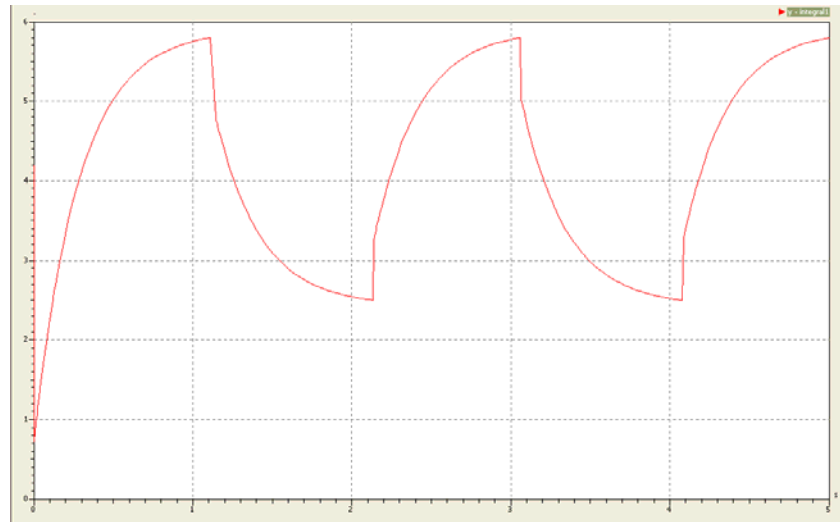


Figure 4.19 Plot y_1 (simulationX)

Task b Time Discontinuity and Final Value of $y_1(5.0)$

The time discontinuity and the final value are:

$$t_0 = 1,108 \quad t_1 = 2,130$$

$$t_2 = 3,058 \quad t_3 = 4,079$$

$$y_1(5,0) = 5,798$$

Table 4-6 The result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance (simulationX)

Relative Tolerance	10^{-6}	10^{-10}	10^{-14}
t_0	1,108	1,105	1,105
t_1	2,130	2,130	2,130
t_2	3,058	3,055	3,055
t_3	4,079	4,077	4,077
$y_1(5,0)$	5,798	5,799	5,798

Task c Time Discontinuity and Final Value of $y_1(5.0)$ with Different Relative Tolerance

The parameter of relative tolerance is varied between 10^{-6} , 10^{-10} and 10^{-14} while still using $1e-12$ as min stepsize. $1e-3$ as absolute tolerance, 0 ...5s as simulation time interval and Dassl as the solver. Table 4-6 shows the result of time discontinuity and final value $y_1(5.0)$ with vary relative tolerance.

Task d Frequent Events

Changing the state 2 parameter values and switching condition will result in a high frequent event of discontinuity for y_1 . It took 0,434s to simulate task d. Figure 4.20 shows plot y_1 (task d)

The time discontinuity and the final value are:

$t_0 = 1,107$	$t_{13} = 1,897$	$t_{26} = 2,793$	$t_{39} = 3,583$	$t_{52} = 4,475$
$t_1 = 1,122$	$t_{14} = 2,015$	$t_{27} = 2,803$	$t_{40} = 3,699$	$t_{53} = 4,486$
$t_2 = 1,239$	$t_{15} = 2,029$	$t_{28} = 2,922$	$t_{41} = 3,709$	$t_{54} = 4,605$
$t_3 = 1,250$	$t_{16} = 2,145$	$t_{29} = 2,935$	$t_{42} = 3,828$	$t_{55} = 4,615$
$t_4 = 1,368$	$t_{17} = 2,159$	$t_{30} = 3,051$	$t_{43} = 3,842$	$t_{56} = 4,734$
$t_5 = 1,382$	$t_{18} = 2,275$	$t_{31} = 3,062$	$t_{44} = 3,957$	$t_{57} = 4,745$
$t_6 = 1,498$	$t_{19} = 2,284$	$t_{32} = 3,181$	$t_{45} = 3,968$	$t_{58} = 4,863$
$t_7 = 1,511$	$t_{20} = 2,404$	$t_{33} = 3,194$	$t_{46} = 4,087$	$t_{59} = 4,874$
$t_8 = 1,627$	$t_{21} = 2,415$	$t_{34} = 3,310$	$t_{47} = 4,100$	$t_{60} = 4,993$
$t_9 = 1,638$	$t_{22} = 2,534$	$t_{35} = 3,321$	$t_{48} = 4,216$	
$t_{10} = 1,756$	$t_{23} = 2,547$	$t_{36} = 3,440$	$t_{49} = 4,227$	
$t_{11} = 1,770$	$t_{24} = 2,663$	$t_{37} = 3,453$	$t_{50} = 4,346$	
$t_{12} = 1,886$	$t_{25} = 2,677$	$t_{38} = 3,569$	$t_{51} = 4,359$	
$Y_1(5,0) = 4,123$				

For all the calculation and simulation, using SimulationX version 2.0 on PC Intel Pentium D, 2 x 2,66 GHz.

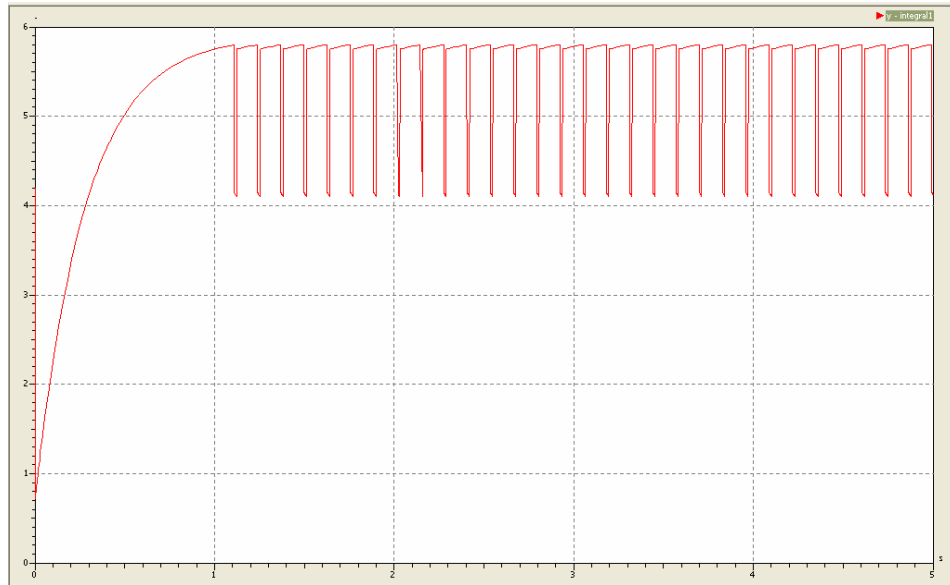


Figure 4.20 plot y_1 (task d) (simulationX)

5. Comparison 20: Electrical Model - Basics

5.1 Definition

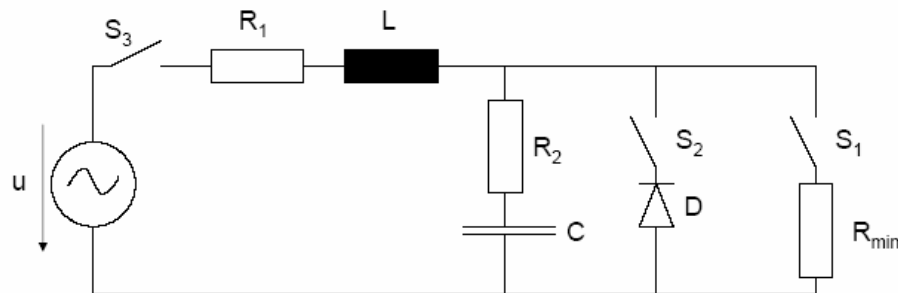


Figure 5.1 Electrical circuit comparison 20

Electrical circuit comparison 20 is given by figure 5.1. The component values are:

- $U = 2 \cdot 10 \text{ kV} \cdot \sin(2 \cdot 50 \text{ Hz} \cdot t +)$ volt
- $L = 3,18 \text{ E-3}$ henry
- $C = 11,1 \text{ E-9}$ farad
- $R_1 = 0,1$ ohm
- $R_2 = 5$ ohm
- $R_{\min} = 1 \text{ E-4}$ ohm

The equations describing the circuit may be the state-equations where inductor currents and capacitor voltages are chosen as system variables. By using the Kirchhoff voltage and current laws we get the following differential equations:

$$L \cdot dx_1/dt = U - x_1 \cdot R_1 - VD$$

$$C \cdot dx_2/dt = + x_1 - ID - VD/R_{\min}$$

$$VD = R_2 \cdot C \cdot dx_2/dt + x_2$$

$$ID = ids \cdot (e^{VD/VT} - 1)$$

Where: $x_1 = I_L$ (the current of L)

$x_2 = V_C$ (the voltage of C)

VD = voltage of diode

ID = current of diode

ids = saturation current of diode

VT = thermal voltage of diode

5.2 Tasks

5.2.1 Steady States

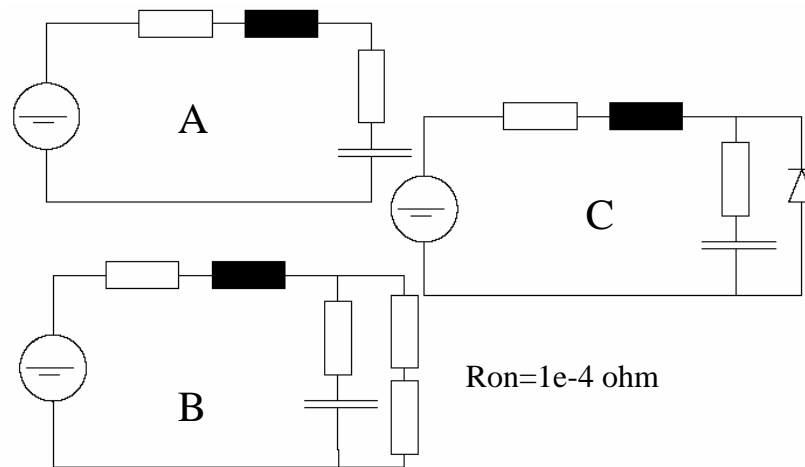


Figure 5.2 Steady States

Steady states is given by figure 5.2. Simulate the system for each state.

Equation:

-State A: $L \frac{dx_1}{dt} = U - x_1(R_1 + R_2) - x_2$

$$C \frac{dx_2}{dt} = x_1$$

-State B: $L \frac{dx_1}{dt} = U - x_1 R_1 - (R_2 C \frac{dx_2}{dt}) - x_2$

$$C(R_2 + R_{on} + R_{min}) \frac{dx_2}{dt} = x_1(R_{on} + R_{min}) - x_2$$

-State C: $V_D < 0$: $L \frac{dx_1}{dt} = -U - x_1 R_1 - V_D$

$$C \frac{dx_2}{dt} = x_1 - (1e-5 V_D)$$

$V_D \geq 0$ $L \frac{dx_1}{dt} = -U - x_1 R_1 - (1e-5 V_D)$

$$C \frac{dx_2}{dt} = x_1 - V_D [12]$$

5.2.2 Classical Simulation

The task are, plot x_1 and x_2 when switch S1 is time dependent switch is given by figure 5.3.

Equation: S1 open: $L \frac{dx_1}{dt} = U - x_1(R_1 + R_2) - x_2$

$$C \frac{dx_2}{dt} = x_1$$

S1 close $L \frac{dx_1}{dt} = U - x_1 R_1 - (R_2 C \frac{dx_2}{dt}) - x_2$

$$C(R_2 + R_{on} + R_{min}) \frac{dx_2}{dt} = x_1(R_{on} + R_{min}) - x_2$$

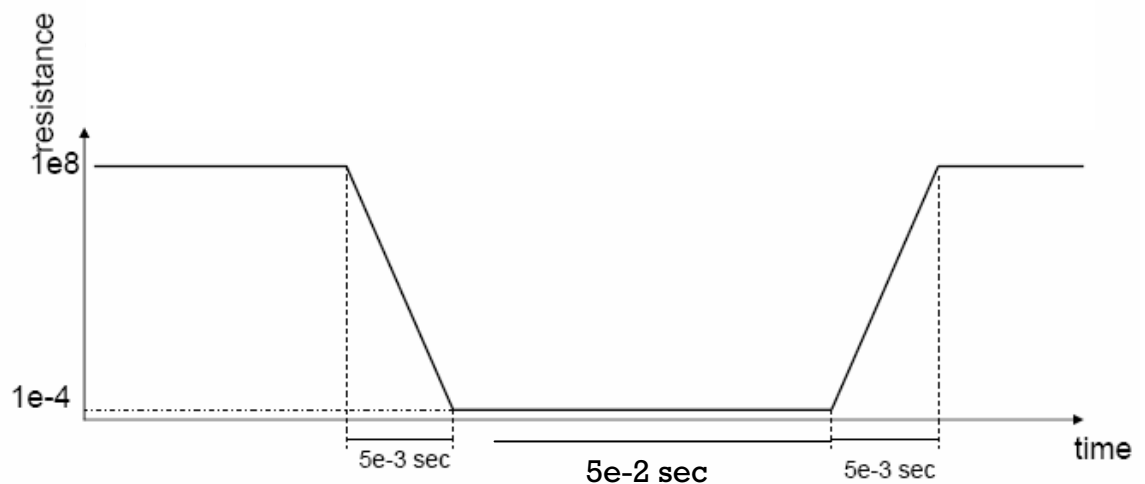


Figure 5.3 Time dependent S1

5.2.3. Different Diode models

The tasks are simulate the system when diode is:

- ideal model with simplification (diode A)

$$\text{Equation: } L \cdot dx1/dt = -U - x1 \cdot R1 - VD$$

$$C \cdot dx2/dt = x1 - ID$$

$$VD = R2 \cdot C \cdot dx2/dt + x2$$

$$ID = ids \cdot (e^{VD/VT} - 1) \quad [13]$$

- diode modelled as exponential function (diode B)

Equation:

$$L \cdot dx1/dt = -U - x1 \cdot R1 - VD$$

$$VD/VT \leq \text{maxexp} \quad C \cdot dx2/dt = x1 - ids \cdot (e^{VD/VT} - 1) + VD/R$$

$$VD/VT > \text{maxexp} \quad C \cdot dx2/dt = x1 - ids \cdot (e^{\text{maxexp} \cdot (1 + (VD/VT) - \text{maxexp})} - 1) + VD/R$$

$$VD = R2 \cdot C \cdot dx2/dt + x2$$

where R = diode resistance

maxexp = maximum exponent for linear continuation.

- realistic with data from a set of characteristic curves (temperature diode=diode C)

$$L \cdot dx1/dt = -U - x1 \cdot R1 - VD$$

$$VD/VT \leq \text{maxexp} \quad C \cdot dx2/dt = x1 - ids \cdot (e^{VD/VT} - 1) + VD/R$$

$$VD/VT > \maxexp C * dx2/dt = x1 - ids * (e^{\maxexp * (1 + (VD/VT) - \maxexp)} - 1) + VD/R$$

$$VD = R2 * C * dx2/dt + x2$$

$$VT = (k * T) / q$$

where k = Boltzmann's constant

T = absolute temperature (°K)

q = magnitude of charge on an electron

5.2.4 Influence of Simulation Algorithms

The tasks are:

- simulate the system when all switch on.

Equation: $L * dx1/dt = -U - x1 * R1 - VD$

$$C * dx2/dt = x1 - ID - VD / (Ron + Rmin)$$

$$VD = R2 * C * dx2/dt + x2$$

$$ID = ids * (e^{VD/VT} - 1)$$

- calculation of the condition of the mass matrices for each case.

5.3. Design and Solutions

5.3.1 Matlab

5.3.1.1 Steady States

Design of Model

For design of the model using matlab algorithm ode23s to solve the system numerically for state A and state B, ode15i for state C. Ode15i was used, because of its speciality that it can solve the matrix in implicit form, where ode23s can only solve the matrix in explicit form. For switching differential equation in state C, solver's state event finder was needed. The code of differential equation and events function for state C were written below:

```
function dxdt=deqx(t,x) %State A
global L C R1 R2
A=[-(R1+R2)/L -1/L; 1/C 0];
b=[U(t)/L; 0];
dxdt=(A*x)+b;
end
```



```

function dxdt= deqxB(t,x)           %State B
global L C R1 R2 Rmin Ron
A=[-(R1*R2)+((R1+R2)*(Ron+Rmin)))/(L*(R2+Ron+Rmin)) -
(Ron+Rmin)/(L*(R2+Ron+Rmin));
(Ron+Rmin)/(C*(R2+Ron+Rmin)) -1/(C*(R2+Ron+Rmin));
b=[U(t)/L; 0];
dxdt=(A*x)+b;
end

function dxdt = f1(t,x,xp)           %State C
global L C R1 R2 R
% VD<0
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-(((R2*C*xp(2))+x(2))/L)
(x(1)/C)-(((R2*C*xp(2))+x(2))*R)/C)]

function dxdt = f2(t,x,xp)
global L C R1 R2 R
% VD>=0
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-(((R2*C*xp(2))+x(2))*R)/L)
(x(1)/C)-(((R2*C*xp(2))+x(2))+R)/C)]

function [value,isterminal,direction] = events(t,x,xp)
global R2 C d
value = ((R2*C*xp(2))+ x(2)) - [0;0];
isterminal = [1;1];
direction = [0;d];

```

Simulation

To simulate the system, using matlab built in function ode23s (odesolver) for state A and state B, ode15i for state C with the solver form:

```

[tsol,xsol]=ode23s('deq',[tstart tfinal],x0);
[t,x,te,xe,ie] = ode15i(@deq,[tstart tfinal],x0, xp0, options);
Where xp0 = initial value for dx/dt

```

The result for plot x1 and x2 state A, state B and state C is shown in figure 5.4. With time interval 0 ... 0,2s, it took 29,635323s to simulate state A, 1,194620s to simulate state B and 1,437591s to simulate state C.

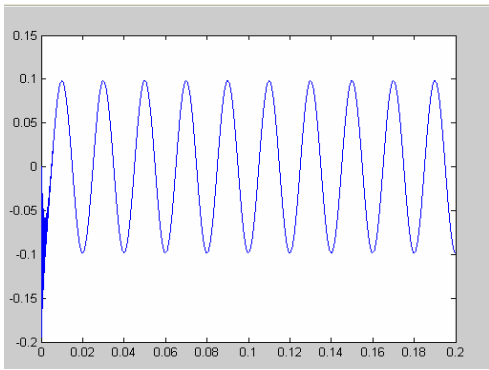


Figure 5.4 Plot x_1 and x_2 steady states (matlab)

5.3.1.2 Classical Simulation

Design of Model

For design of the model using matlab algorithm ode23s to solve the system numerically. Time dependent switch was built by matlab function-type m-files. The code of time dependent switch, differential equation and events function were written below:

function $T_{out} = T(t)$ %time dependent switch
persistent TRF

```

TRF=5e-3;
k=((1e+8)-(1e-4))/TRF;
t_red=mod(t, (1e-1));
if(0<=t_red)&&(t_red<TRF)
    T_out=(1e-4)+k*t_red;
elseif(TRF<=t_red)&&(t_red<(5e-2))
    T_out=1e+8;
elseif((5e-2)<=t_red)&&(t_red<((5e-2)+TRF))
    T_out=(1e+8)-k*(t_red-(5e-2));
elseif((5e-2)+TRF<=t_red)&&(t_red<(1e-1))
    T_out=1e-4;
else
    T_out=-5;
end
function dxdt1= deqtaskb1(t,x)                %S1 open
global L C R1 R2
dxdt1(1,1) = (-x(1)*(R1+R2)/L)-(x(2)/L)+(U(t)/L);
dxdt1(2,1) = (x(1)/C);
function dxdt2= deqtaskb2(t,x)                %S1 close
global L C R1 R2 Rmin Ron
dxdt2(1,1) = (-x(1)*((R1*R2)+((R1+R2)*(Ron+Rmin)))/(L*(R2+Ron+Rmin)))-
(x(2)*(Ron+Rmin)/(L*(R2+Ron+Rmin)))+(U(t)/L);
dxdt2(2,1) = (x(1)*(Ron+Rmin)/(C*(R2+Ron+Rmin)))-(x(2)/(C*(R2+Ron+Rmin)));
function [value,isterminal,direction] = events(t,x)
global p
value = T(t)- p + [0;0];
isterminal = [1;1];
direction = [0;0];

```

Simulation

To simulate the system, using matlab built in function ode23s with the solver form:

```
[t,x,te,xe,ie] = ode23s(@deq,[tstart tfinal],x0, options);
```

The result for plot x1 and x2 is shown in figure 5.5. With time interval 0 ... 0,3s, it took 50.511835s to simulate this task.

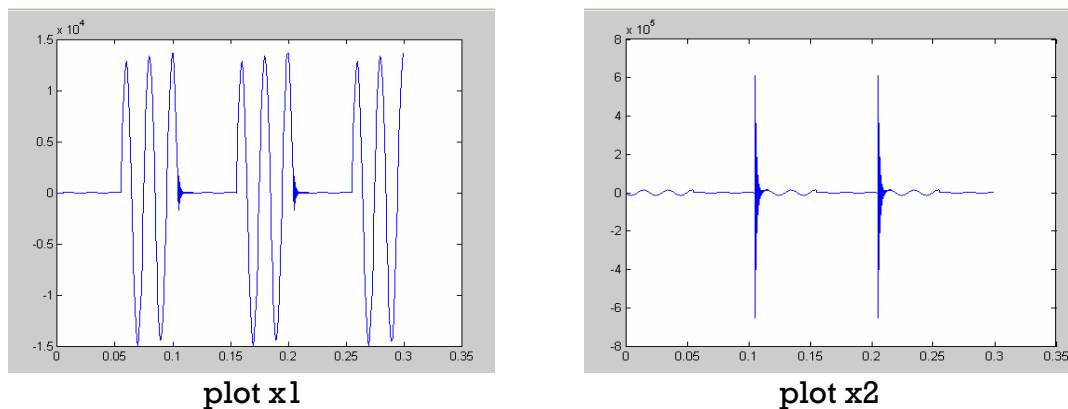


Figure 5.5 Plot x1 and x2 classical simulation (matlab)

5.3.1.3 Different Diode Models

Design of Model

For design of the model using matlab algorithm ode15i to solve the system numerically in implicit form. The function for diode B and diode C are similar, the different is in thermal voltage VT, where in diode B, VT is a variable and in diode C, VT is a function. The code of differential equation, events function and thermal voltage VT were written below:

```
function dxdt = f(t,x,xp) %diode A
global L C R1 R2 Rmin Ron ids VT R
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-((R2*C*xp(2))/L)-(x(2)/L)
        (x(1)/C)-((ids*(exp(((R2*C*xp(2))-x(2))/VT)-1))/C)];

function dxdt = f1(t,x,xp) %diode B & C, where in B VT=0.04
global L C R1 R2 Rmin Ron ids VT R in C VT=VT(t)
%VD/VT maxexp
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-((R2*C*xp(2))/L)-(x(2)/L)
        (x(1)/C)-((ids*(exp(((R2*C*xp(2))-x(2))/VT)-1))/C)+(((R2*C*xp(2))-x(2))/(R*C))];

function dxdt = f2(t,x,xp)
global L C R1 R2 Rmin Ron ids VT maxexp R
%VD/VT>maxexp
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-((R2*C*xp(2))/L)-(x(2)/L)
        (x(1)/C)-((ids*(exp(maxexp*(1+(((R2*C*xp(2)) + x(2)/VT))-maxexp))-1))/C)-
        (((R2*C*xp(2))-x(2))/(R*C))];

function [value,isterminal,direction] = events(t,x,xp)
global R2 C VT maxexp
value = (((R2*C*xp(2))+ x(2))/VT) - maxexp + [0;0];
isterminal = [1;1];
direction = [0;0];

function VTout = VT(t) %Thermal Voltage function for diode C
VTout = ((30 * sin(2*pi*100*t))+310)*8.61734681e-5;
End
```

Simulation

To simulate the system, using matlab built in function ode15i with the solver form:

```
[tsol,xsol]=ode15i(@deq,[tstart tfinal],x0, xp0); %for diode A
[t,x,te,xe,ie] = ode15i(@deq,[tstart tfinal],x0, xp0, options); %for diode B & C
```

The result for plot x1 and x2 diode A, diode B and diode C is shown in figure 5.6. With time interval 0 ... 0,2s, it took 0,861165s to simulate diode A, 1,016539s to simulate diode B and 0,958469s to simulate diode C.

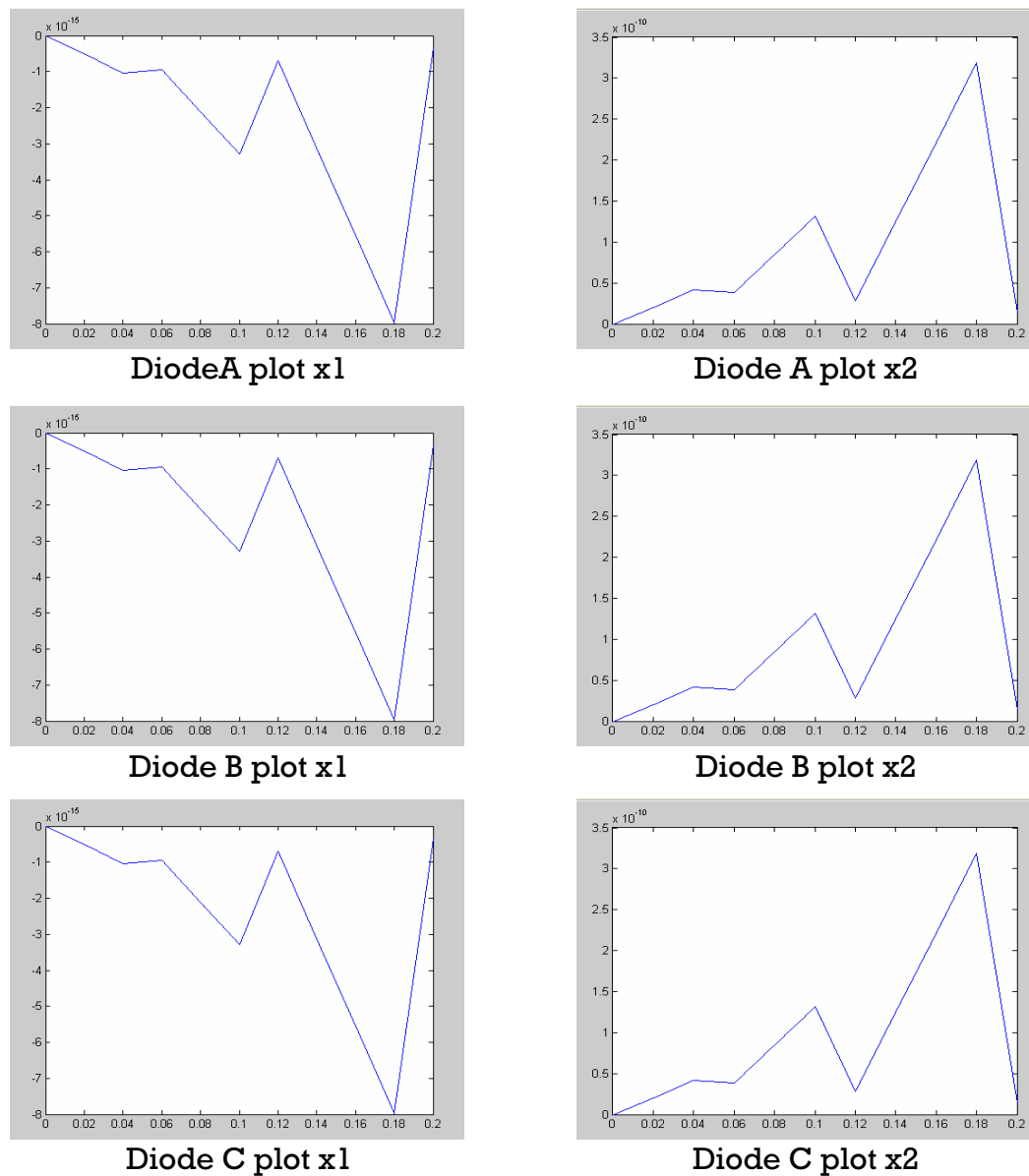


Figure 5.6 Plot x1 and x2 different diode models (matlab)

5.3.1.4 Influence of Simulation Algorithms

Design of Model

For design of the model using matlab algorithm ode15i to solve the system numerically in implicit form. The code of differential equation were written below:

```
function dxdt = f(t,x,xp)
global L C R1 R2 Rmin Ron ids VT
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-((R2*C*xp(2))/L)-(x(2)/L)
```

$$\frac{(x(1)/C)-((ids*(exp(((R2*C*xp(2))-x(2))/VT)-1))/C)-((R2*C*xp(2))/(C*(Ron+Rmin)))-}{(x(2)/(C*(Ron+Rmin)))}$$

Simulation

To simulate the system, using matlab built in function `ode15i` with the solver form:

```
[tsol,xsol]=ode15i(@deq,[tstart tfinal],x0, xp0);
```

The result for plot `x1` and `x2` is shown in figure 5.7. With time interval 0 ... 0,2s, it took 0,859261s to simulate this task.

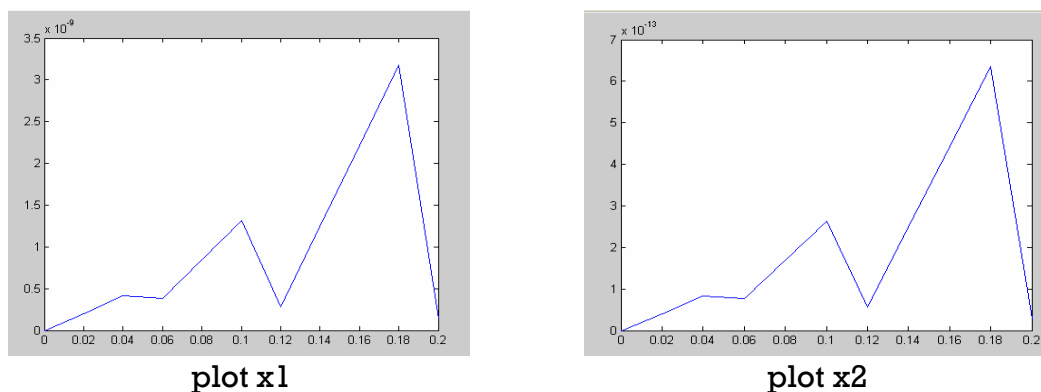


Figure 5.7 Plot `x1` and `x2` influence of simulation algorithms (matlab)
88

Using matlab built in function `cond()` for calculation of condition of massmatrices. This task can only be executed if the system matrix is in explicit form, therefore only the condition of system matrix from state A, state B and task b(classical simulation) that can be calculated. It took 0,189968s to calculate the condition. The results are:

- State A: condition = 143896,5027149321, 1-norm and infinite-norm
condition = 143891,40289569343, 2-norm
- State B: condition = 287254,3405559489, 1-norm and infinite-norm
condition = 287196,9122644086, 2-norm
- Task B: S1 open: condition = 143896,5027149321, 1-norm and infinite-norm
condition = 143891,40289569343, 2-norm

- S1 close: condition = 287254,3405559489, 1-norm and infinite-norm

condition = 287196,9122644086, 2-norm

For whole calculation and simulation, using Matlab/Simulink version 7.4 R2007a on PC Intel Pentium D, 2 x 2,66 GHz.

5.3.2 Simulink

5.3.2.1 Hybrid Model

5.3.2.1.1 Steady States

Design of Model

For design of the model using only gain, add/subtract and integrator block for differential equation and sine source block for the sinus voltage. Switch block was used with threshold 0 and condition control signal \geq threshold for switching differential equation in state C. The model for state A, state B and state C was shown in figure 5.8.

Simulation

To simulate the system, using solver ode23 (Bogacki-Shampine) for state A and ode23s (stiff/Mod.Rosenbrock) for state B and state C, $1e-6$ for relative tolerance and 0 ... 0,2 as simulation interval. Plot x1 and x2 for each state is shown by figure 5.9. it took 1s to simulate state A, 0,5s to simulate state B and 19,5s to simulate state C.

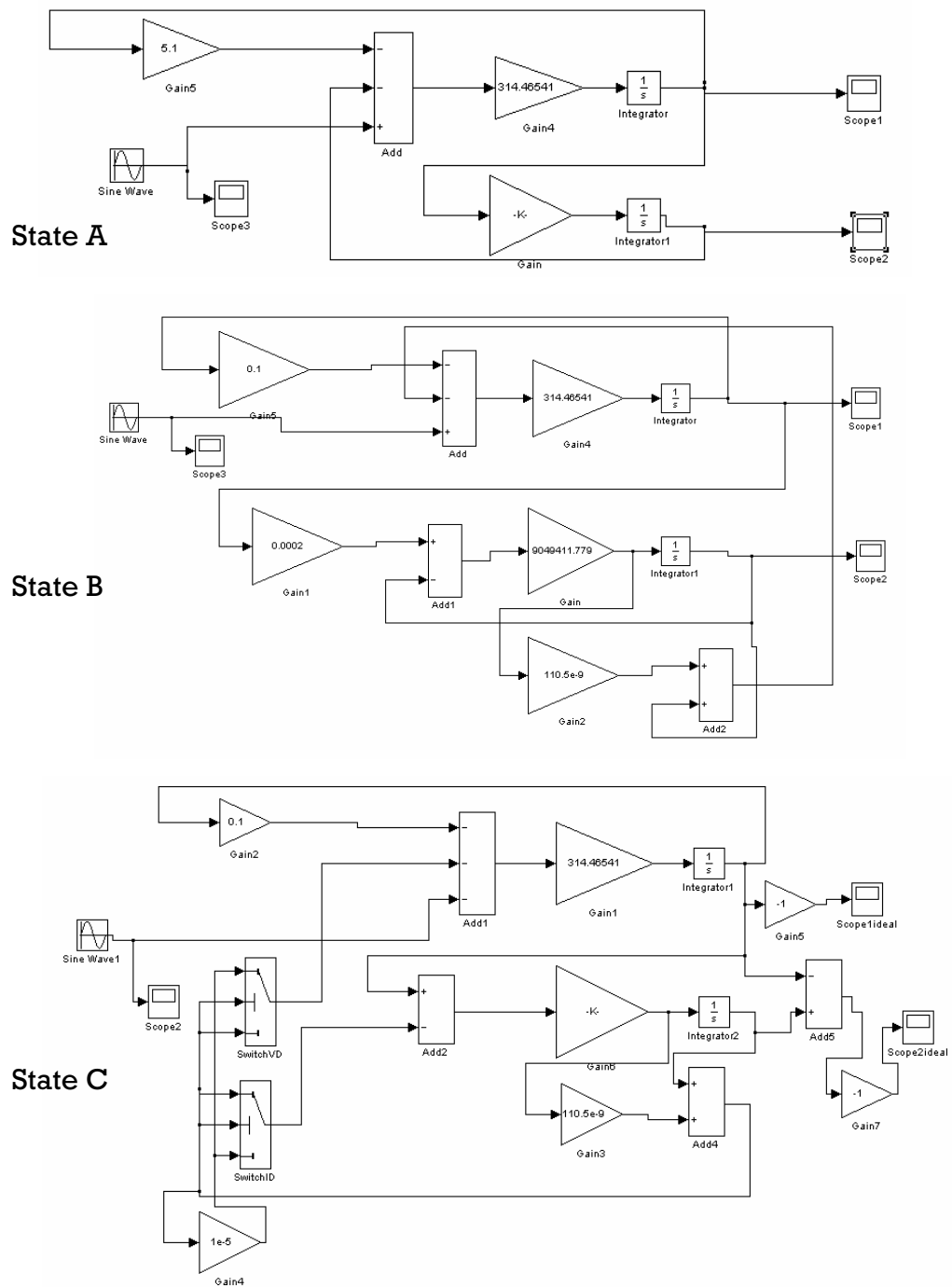
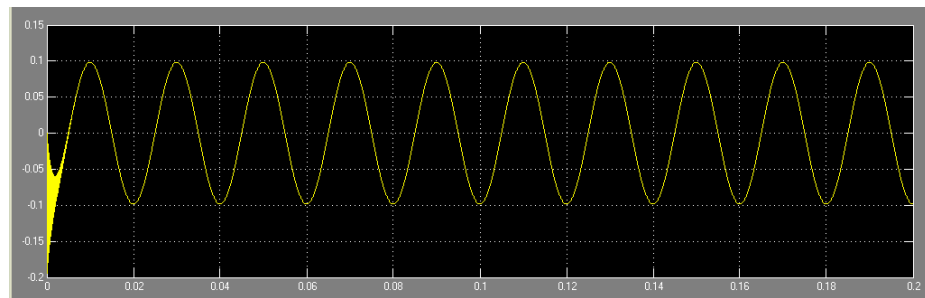
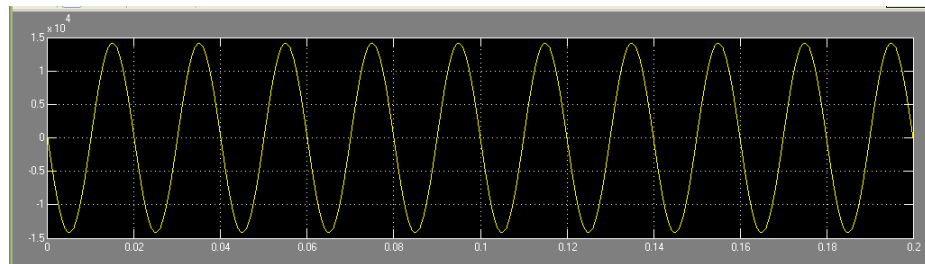
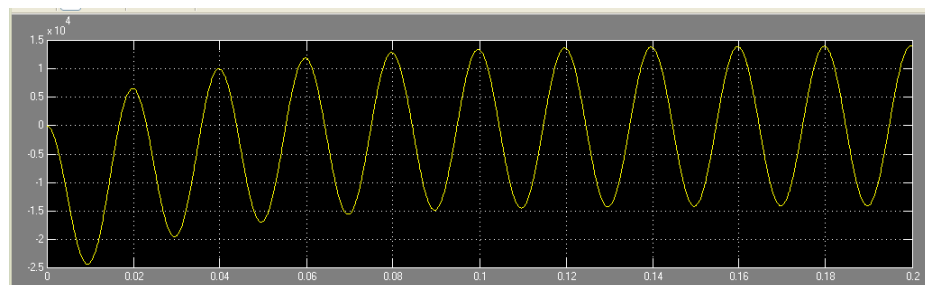


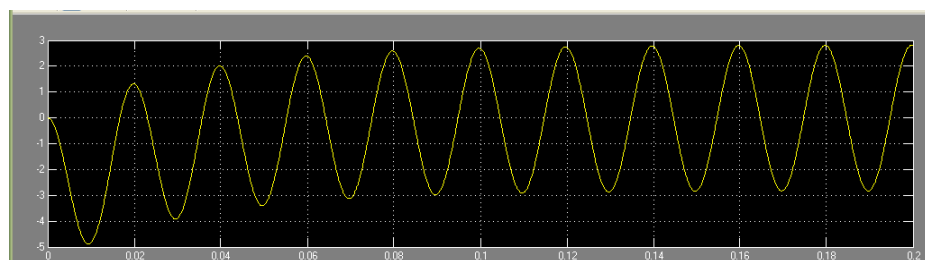
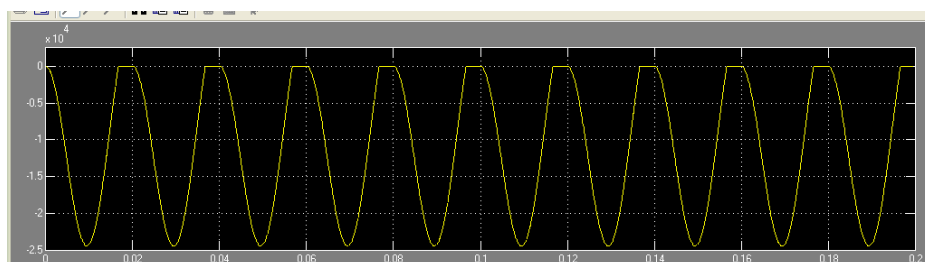
Figure 5.8 Model of the system steady states (simulink)

 x_1

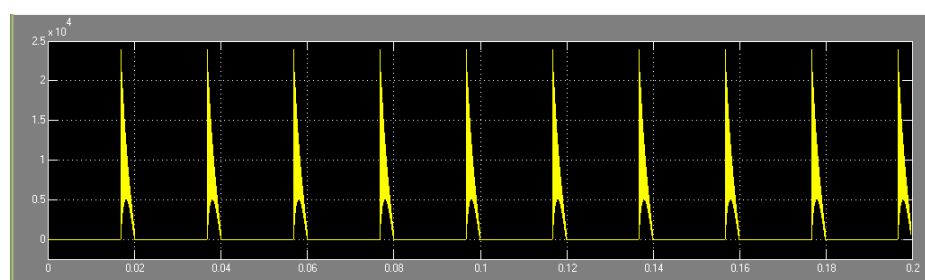
State A

 x_2  x_1

State B

 x_2  x_1

State C

 x_2 Figure 5.9 Plot x_1 and x_2 steady states (simulink)

5.3.2.1.2 Classical Simulation

Design of Model

For design of the model using only gain, add/subtract and integrator block for differential equation and sine source block for the sinus voltage. Time dependent switch was built by embedded matlab m code and switch block with threshold $1e-4$ and condition control signal $>$ threshold for switching differential equation in this model. The model for the system was shown in figure 5.10.

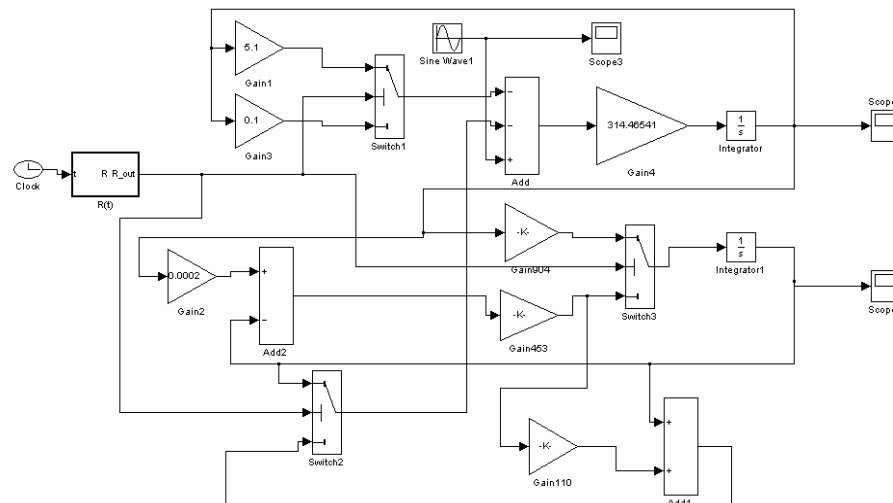


Figure 5.10 Model of the system classical simulation (simulink)

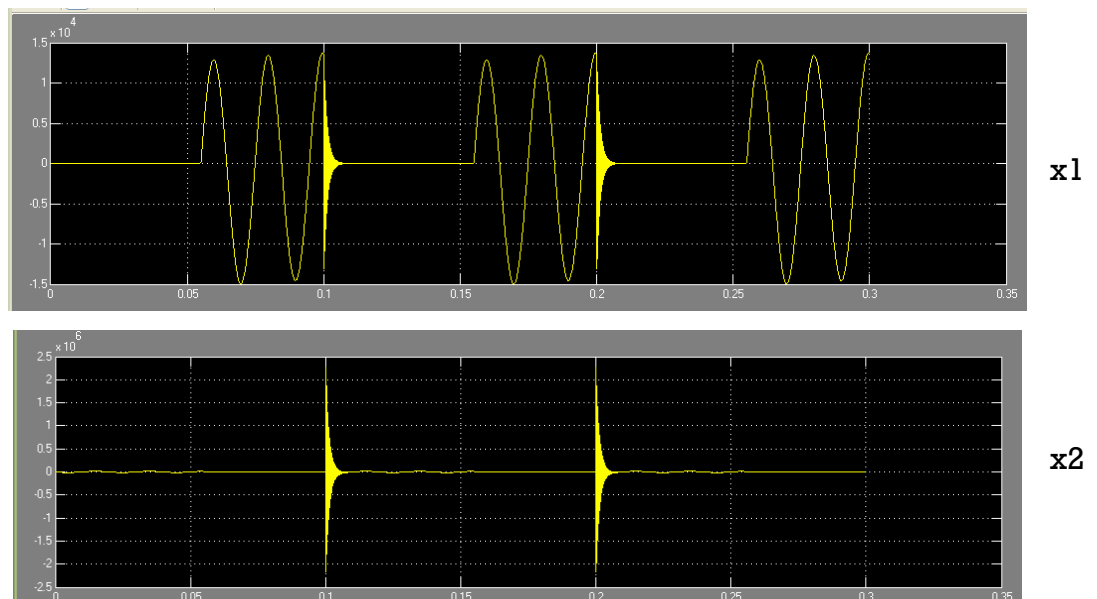
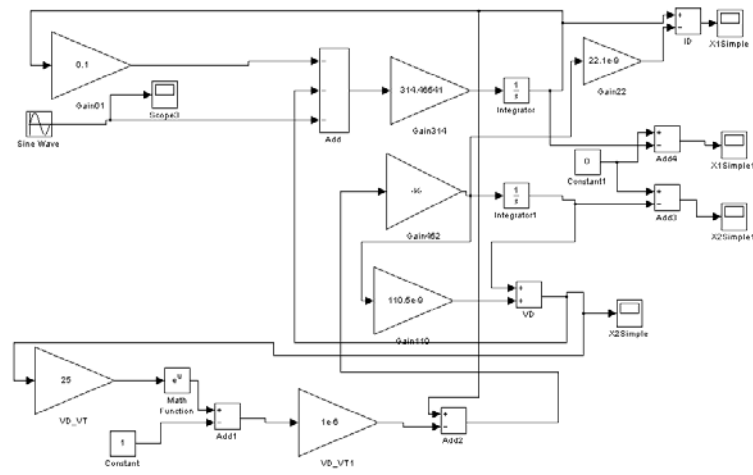


Figure 5.11 Plot x1 and x2 classical simulation (simulink)

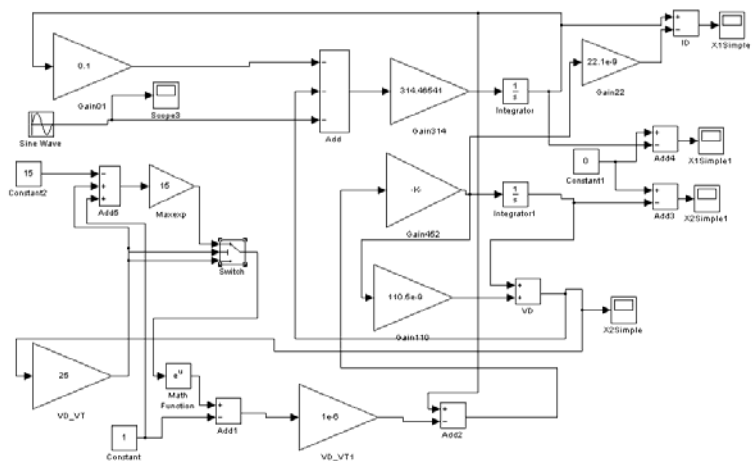
Simulation

To simulate the system, using solver ode23s (stiff/Mod.Rosenbrock), 1e-6 for relative tolerance and 0 ... 0,3 as simulation interval. Plot x1 and x2 for this task is shown by figure 5.11. it took 5,5s to simulate this task.

Diode A



Diode B



Diode C

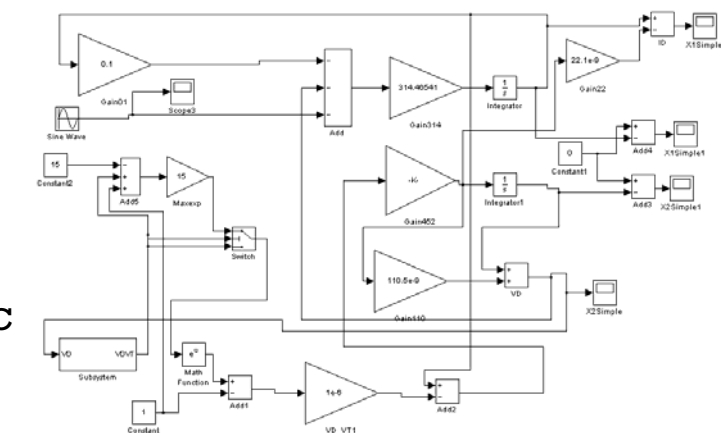


Figure 5.12 The model of the system different diode modes (simulink)

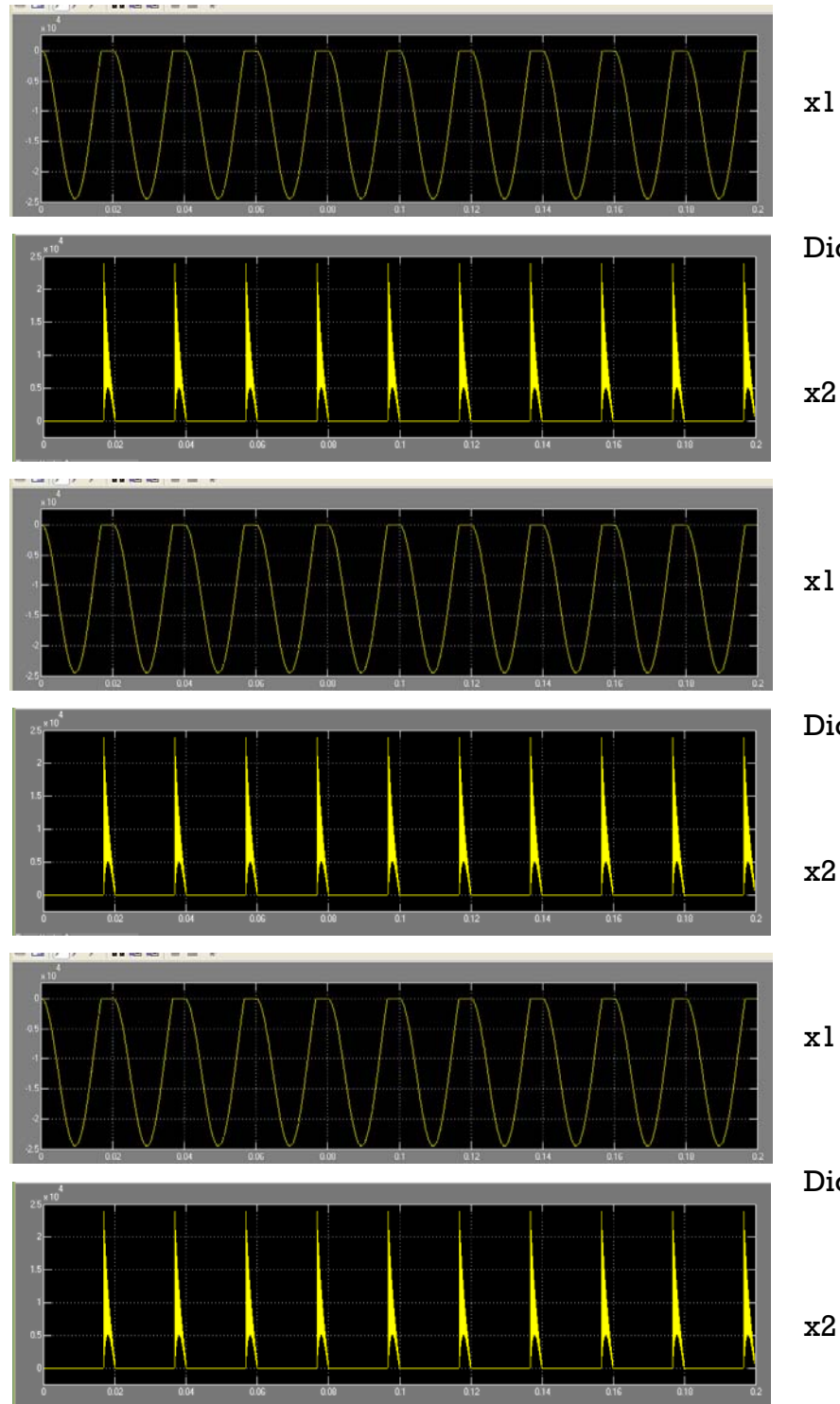


Figure 5.13 Plot x_1 and x_2 different diode models (simulink)

5.3.2.1.3 Different Diode Models

Design of Model

For design of the model for diode A using only gain, add/subtract and integrator block for differential equation, sine source block for the sinus voltage and math function block for exponential function. For diode B, the design is similar with diode A but with addition switch block (threshold 15, condition: control signal > threshold) for switching differential equation in diode B. For diode C, the design based on diode B, with changing 1 gain block (V_D/V_T) with 1 subsystem block to define the function of V_T ($V_D/V_T(t)$). The model for diode A, diode B and diode C was shown in figure 5.12.

Simulation

To simulate the system, using solver ode45 (Dormand-Prince), $1e-6$ for relative tolerance and 0 ... 0,2 as simulation interval. Plot x_1 and x_2 for this task is shown by figure 5.13. it took 153s to simulate diode A, .183s to simulate diode B and 213s to simulate diode C.

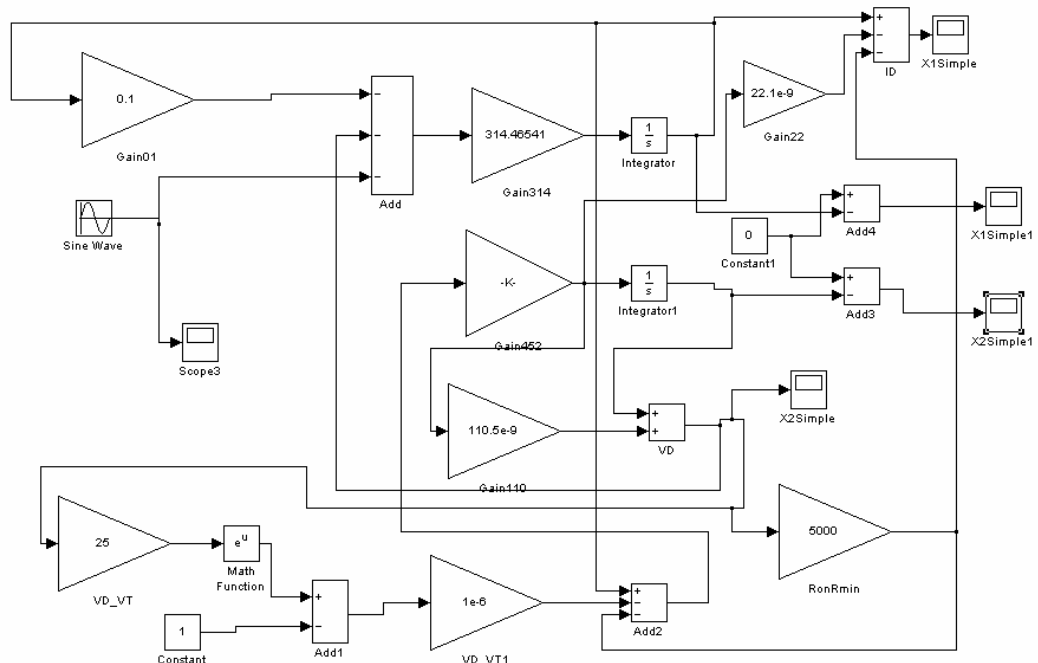


Figure 5.14 Model of the system influence of simulation algorithms (simulink)

5.3.2.1.4 Influence of Simulation Algorithms

Design of Model

For design of the model for this task using only gain, add/subtract and integrator block for differential equation, sine source block for the sinus voltage and math function block for exponential function. The model of the system was shown in figure 5.14.

Simulation

To simulate the system, using solver ode45 (Dorman-Prince), $1e-5$ for relative tolerance and 0 ... 0,2 as simulation interval. Plot x1 and x2 for this task is shown by figure 5.15. it took 255s to simulate this task.

For calculation of condition of massmatrices, can't be done by simulink, because simulink didn't have block function `cond()` in their library, therefore no calculation of condition for simulink.

For whole calculation and simulation, using Matlab/Simulink version 7.4 R2007a on PC Intel Pentium D, 2 x 2,66 GHz.

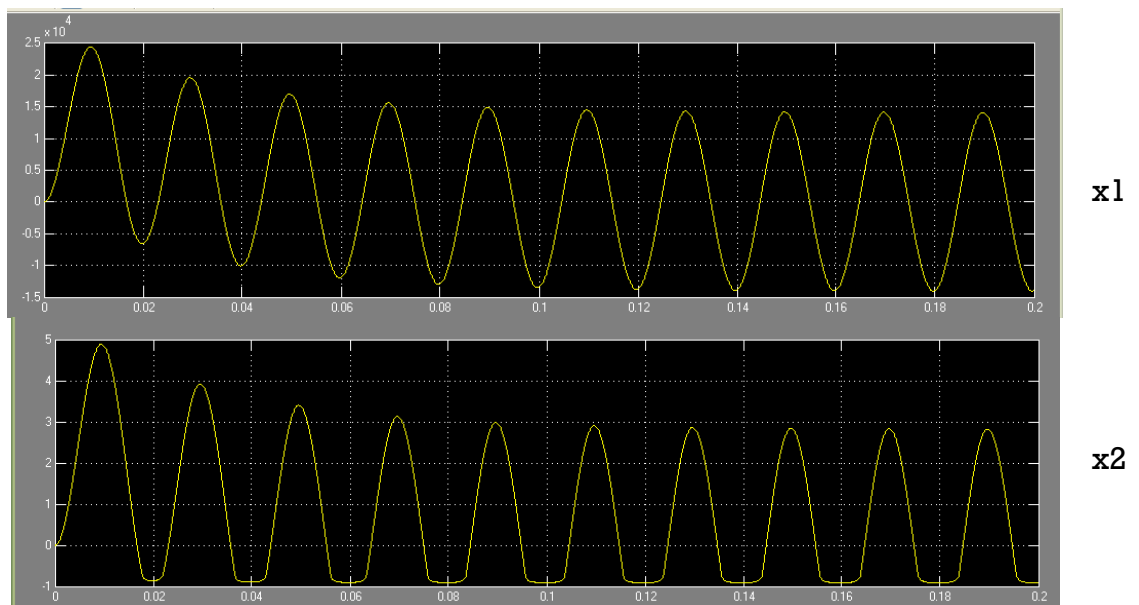


Figure 5.15 Plot x1 and x2 influence of simulation algorithms (simulink)

5.3.2.2 Stateflow

Looking from all the equation comparison 20 above, only state C, task B (classical simulation), diode B and diode C that can be modelled in stateflow mode, because they have switching differential equation in their equation.

condition control signal $>$ threshold) for switching differential equation in this model. The model for the system was shown in figure 5.16.

Simulation

To simulate the system, using solver ode45 (Dormand-Prince), $1e-5$ for relative tolerance and 0 ... 0,2 as simulation interval. Plot x_1 and x_2 for this task is shown by figure 5.17. it took 1,5s to simulate this task.

For whole calculation and simulation, using Matlab/Simulink version 7.4 R2007a on PC Intel Pentium D, 2 x 2,66 GHz.

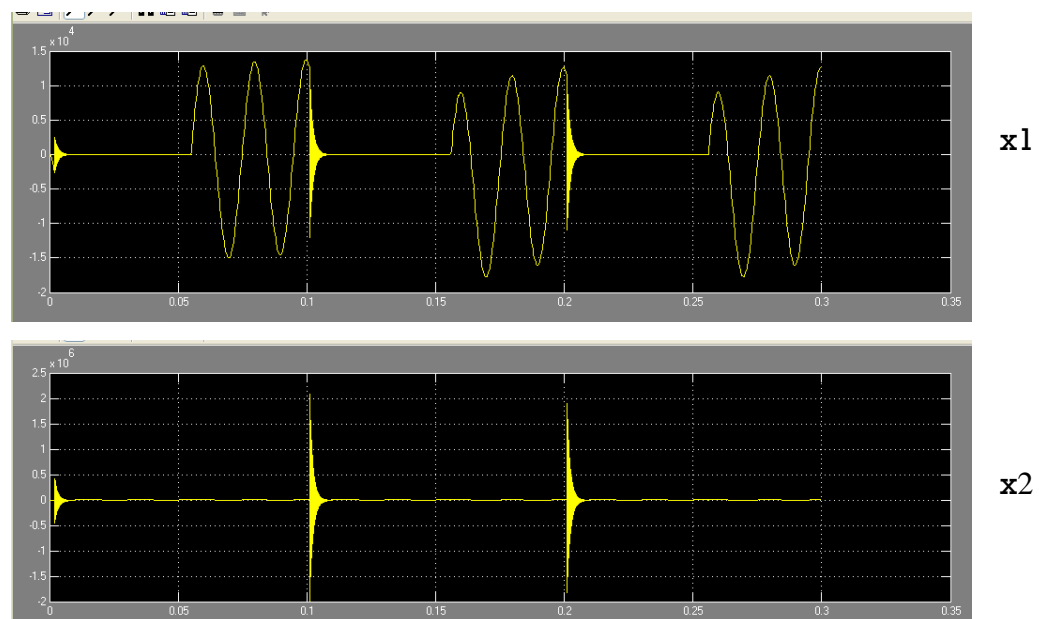


Figure 5.17 Plot x_1 and x_2 classical simulation (stateflow)

5.3.3 Dymola

5.3.3.1 Hybrid Model

5.3.3.1.1 Steady States

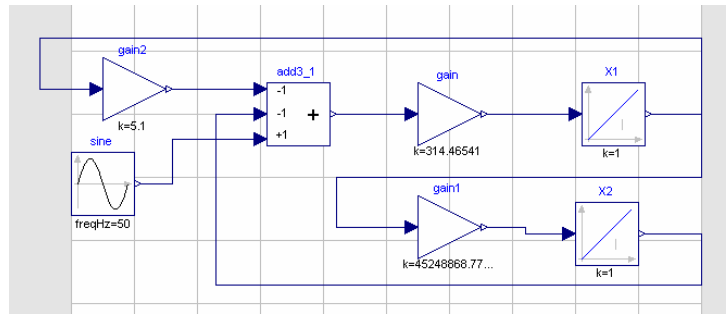
Design of Model

For design of the model using only gain, add/subtract and integrator block for differential equation and sine source block for the sinus voltage. Switch block was controlled by less (<0) block for switching differential equation in state C. The model for state A, state B and state C was shown in figure 5.18.

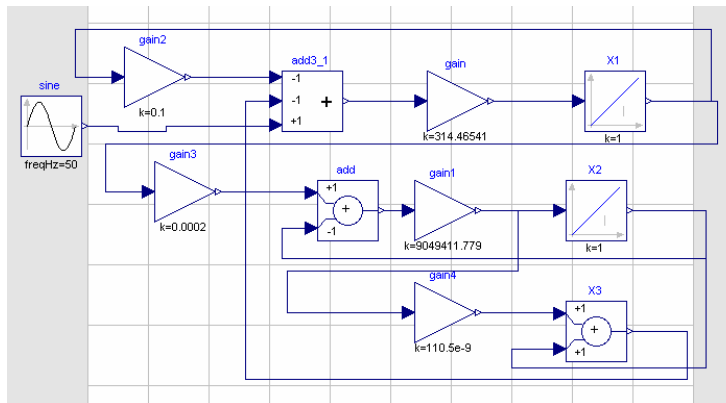
Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval, $1e-4$ for relative tolerance and 0 ... 0,2 as simulation interval. Plot x_1 and x_2 for each state is shown by figure 5.19. it took 0,046s to simulate state A, 0,031s to simulate state B and 0,234s to simulate state C.

State A



State B



State C

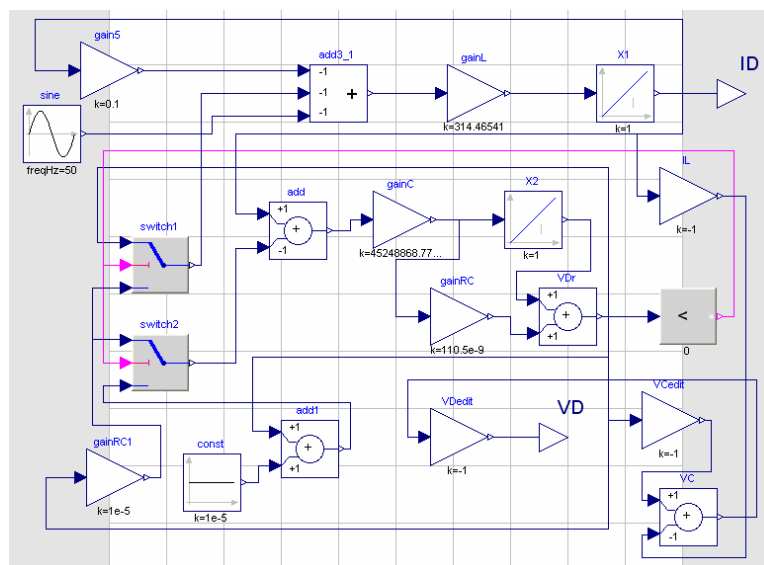


Figure 5.18 Model of the system steady states (dymola)

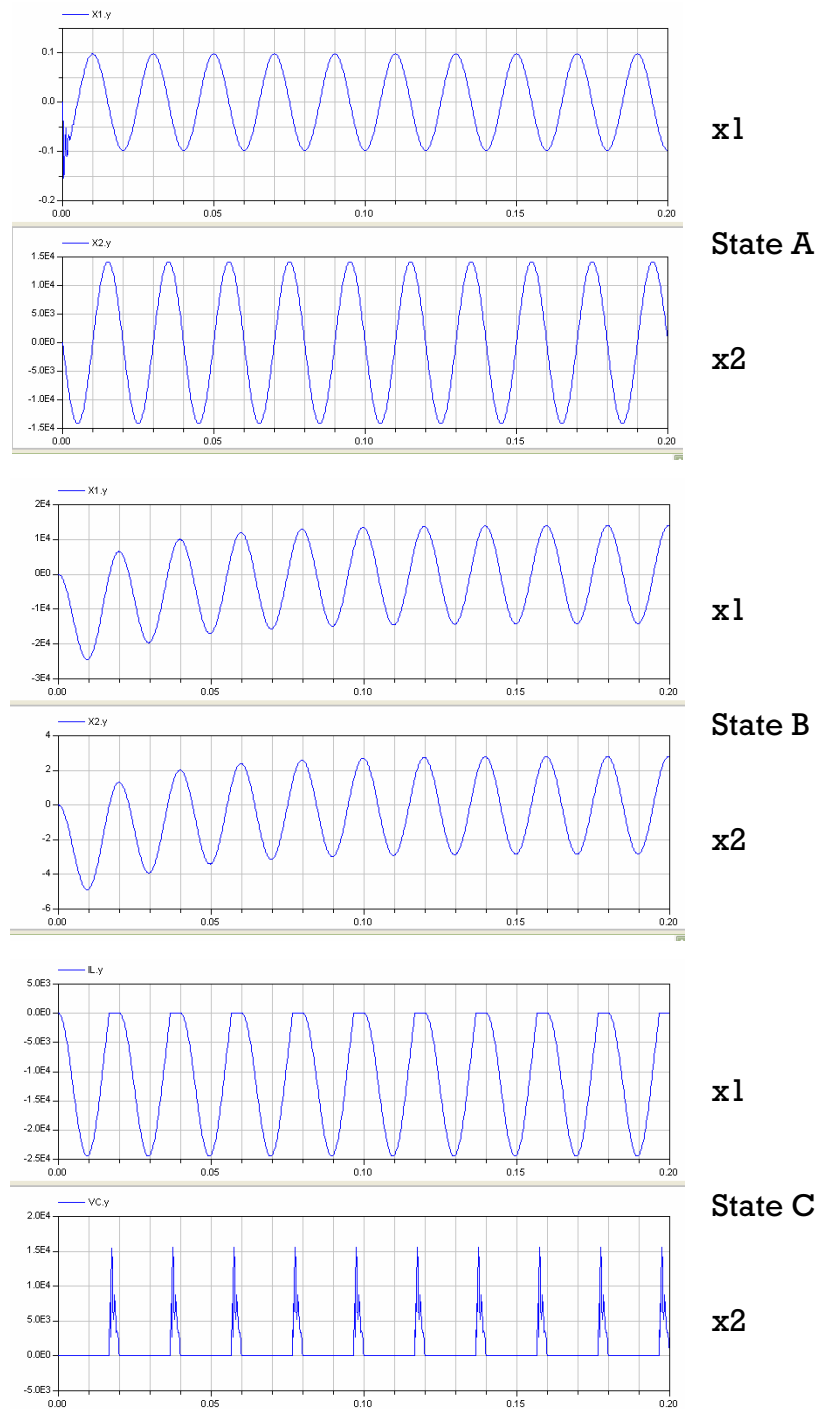


Figure 5.19 Plot x_1 and x_2 steady states (dymola)

5.3.2.1.2 Classical Simulation

Design of Model

For design of the model using only gain, add/subtract and integrator block for differential equation and sine source block for the sinus voltage. Time

dependent switch was built by switch block, less equal block and trapezoid source with parameter listed below:

- ```
- Amplitude = 1e+8 - Offset = 1e-4
- Rising = 5e-3 - Falling = 5e-3
- Width = 5e-2 - Period = 1e-1
```

Switch block was controlled by less equal( 0) block for switching differential equation in this model. The model for the system was shown in figure 5.20.

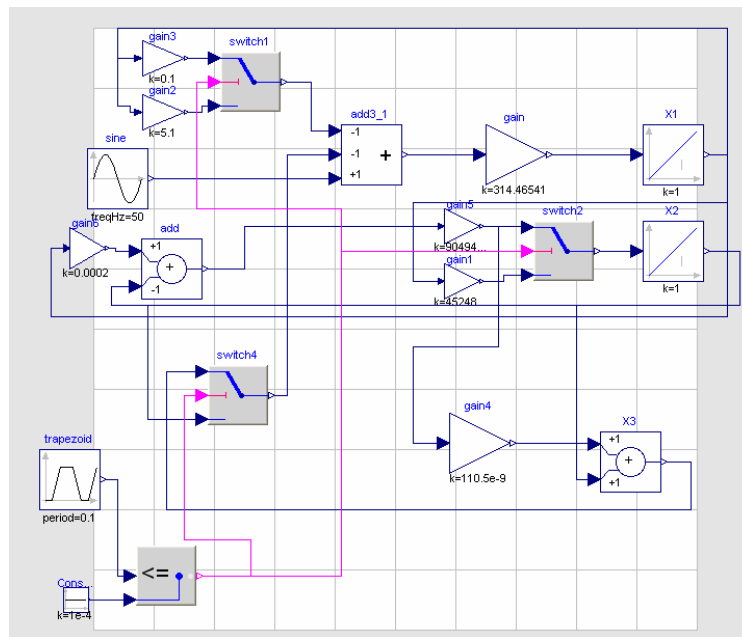


Figure 5.20 Model of the system classical simulation (dymola)

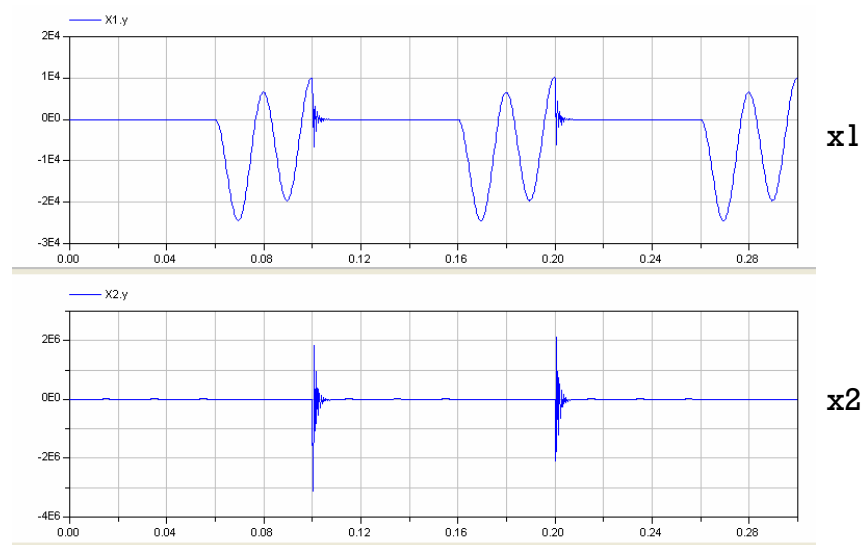
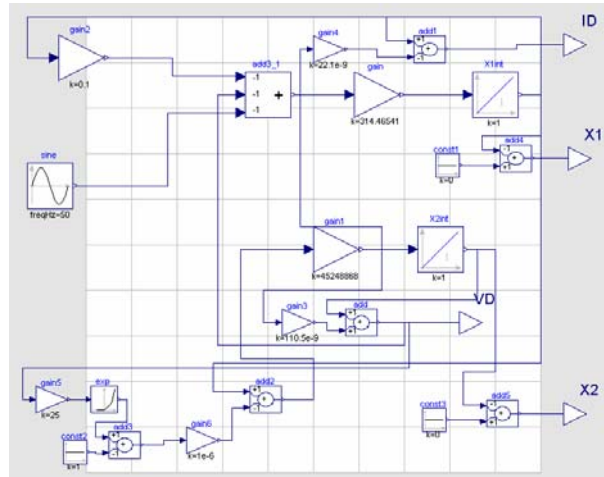


Figure 5.21 Plot x1 and x2 classical simulation (dymola)

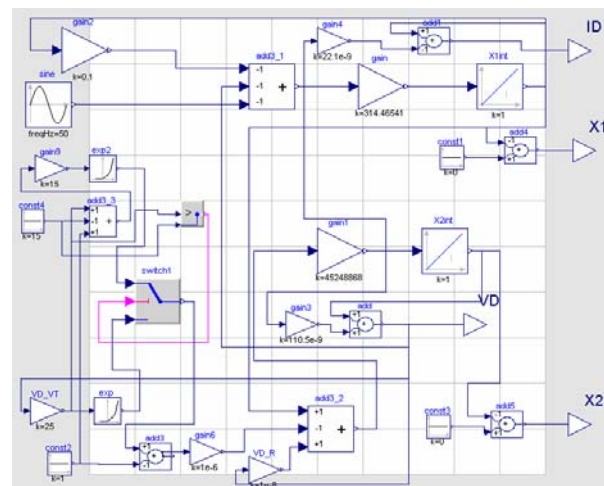
## Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval,  $1e-4$  for relative tolerance and 0 ... 0,3 as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.21. It took 0,203s to simulate this task.

Diode A



Diode B



Diode C

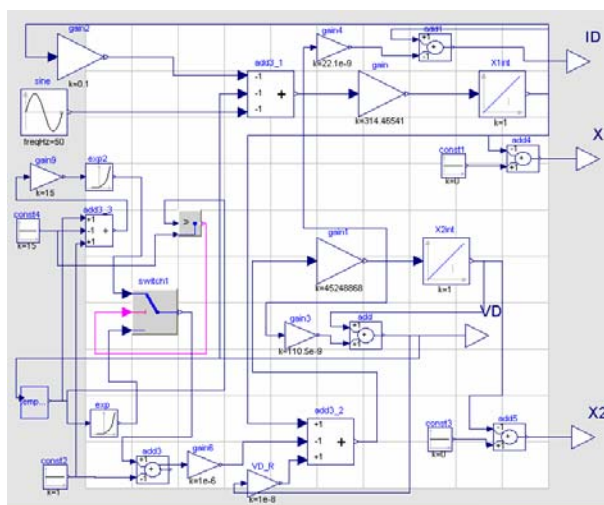


Figure 5.22 The model of the system different diode models (dymola)

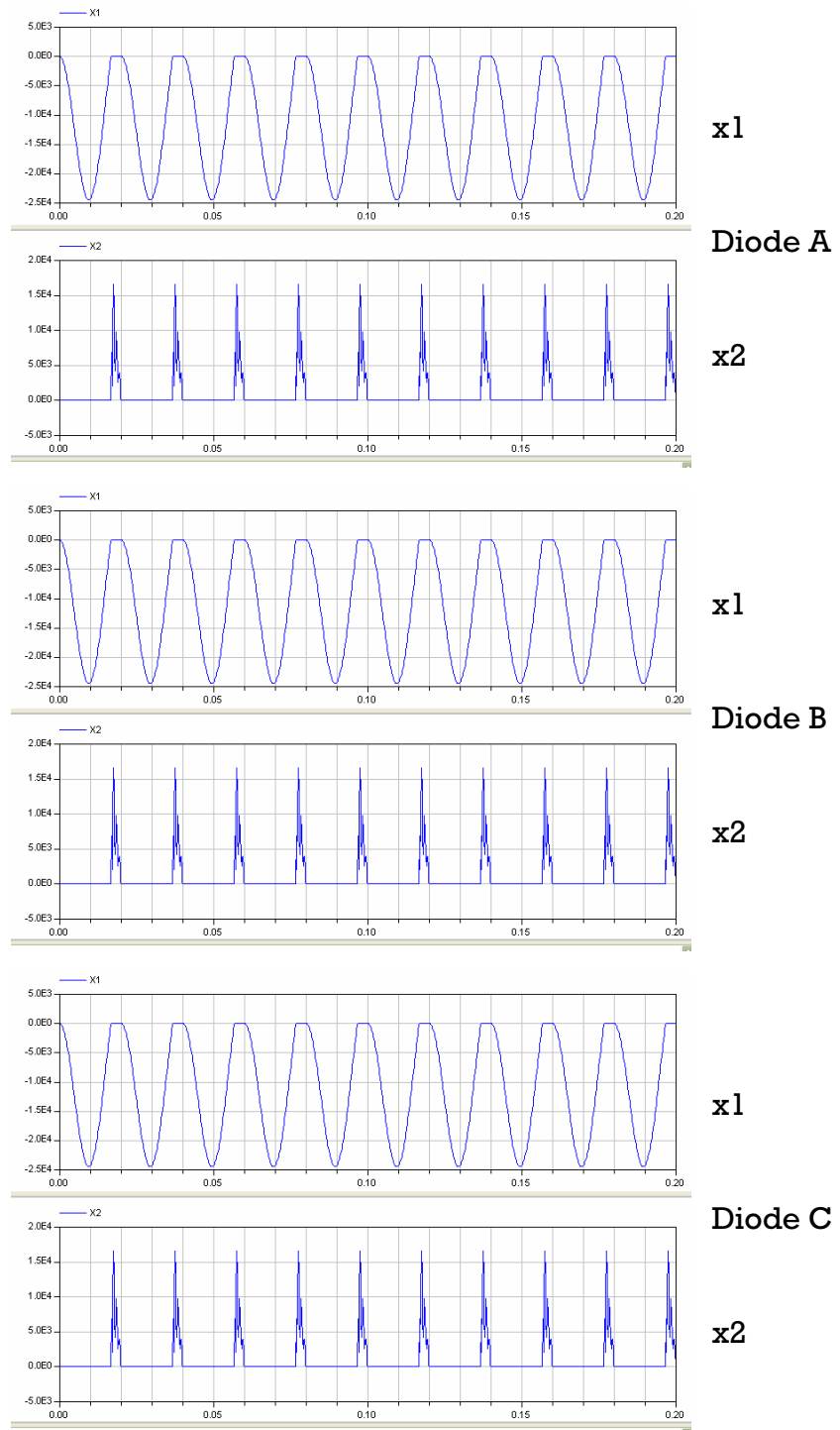


Figure 5.23 Plot  $x_1$  and  $x_2$  different diode models (dymola)

### 5.3.2.1.3 Different Diode Models

#### Design of Model

For design of the model for diode A using only gain, add/subtract and integrator block for differential equation, sine source block for the sinus

voltage and exponent block for exponential function. For diode B, the design is similar with diode A but with addition switch block and greater block(sending Boolean signal to switch block) for switching differential equation in diode B. For diode C, the design based on diode B, with changing 1 gain block ( $V_D/V_T$ ) with 1 subsystem block to define the function of  $V_T$  ( $V_D/V_T(t)$ ). The model for diode A, diode B and diode C was shown in figure 5.22.

### Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval,  $1e-8$  for relative tolerance and 0 ... 0,2 as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.23. it took 2,66s to simulate diode A, .5,42s to simulate diode B and 6,14s to simulate diode C.

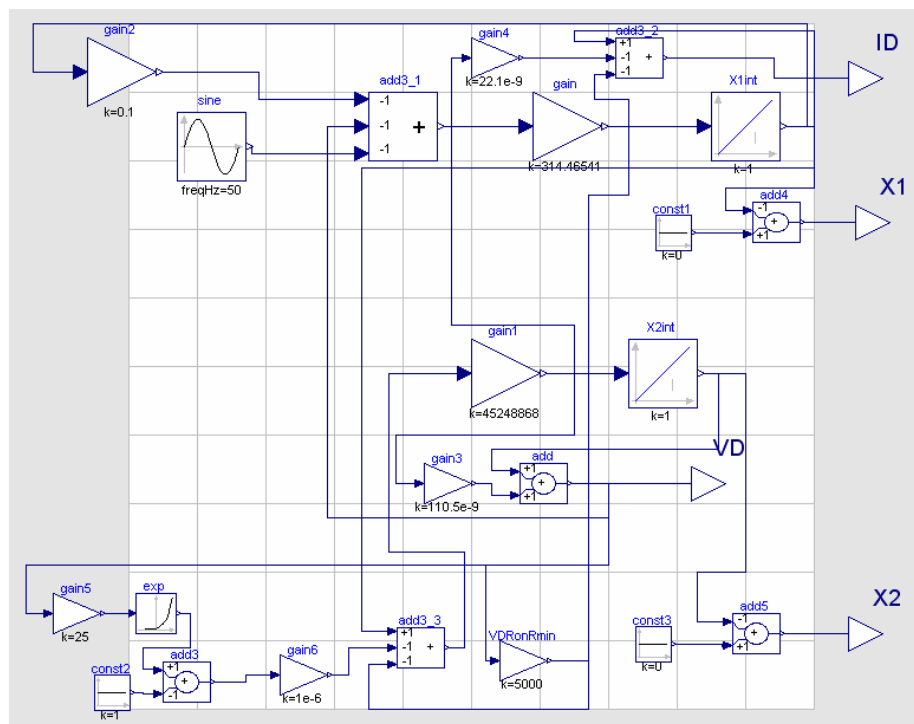


Figure 5.24 Model of the system influence of simulation algorithms (dymola)

#### 5.3.2.1.4 Influence of Simulation Algorithms

##### Design of Model

For design of the model for this task using only gain, add/substract and integrator block for differential equation , sine source block for the sinus

voltage and exponent block for exponential function. The model of the system was shown in figure 5.24.

### Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval,  $1e-8$  for relative tolerance and 0 ... 0,2 as simulation interval. Plot x1 and x2 for this task is shown by figure 5.25. it took 0,141s to simulate this task.

For calculation of condition of massmatrices, can't be done directly by dymola, because dymola didn't have block function `cond()`, but dymola do have norm and inverse function in their library, therefore the calculation of condition based on equation below.

$$\text{Condition} = \text{norm}(A,p) * \text{norm}(\text{inv}(A),p) [14]$$

Where  $A$ = system matrix (massmatrices)

$p$  = norm condition number (1, 2 or infinite)

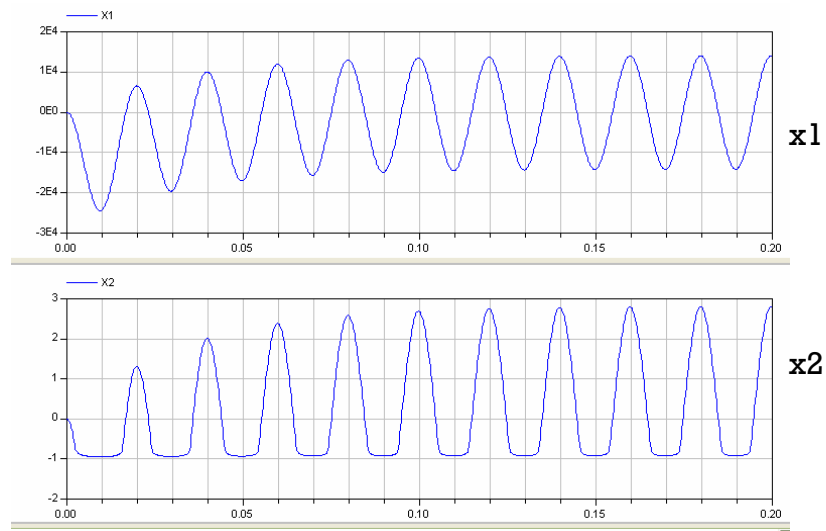


Figure 5.25 Plot x1 and x2 influence of simulation algorithms (dymola)

As written before, only equation in explicit form that the condition of massmatrices can be calculated, therefore only state A, state B and task B (classical simulation) that qualified for this task.

The result: the calculation of condition of massmatrices can't be done by dymola, because dymola can't make inverse matrix for system matrix state A, state B and task B (too stiff).

For whole calculation and simulation, using Dymola version 6.0b on PC Intel Pentium D, 2 x 2,66 GHz.

### 5.3.3.2 Stategraph Model

As written before in stateflow, only state C, task B (classical simulation), diode B and diode C that can be modelled in stategraph mode. This time, stategraph didn't have any restriction like stateflow in simulink.

#### 5.3.3.2.1 Steady States

##### Design of Model

For design of the model using only gain, add/substract and integrator block for differential equation and sine source block for the sinus voltage. Switch block was controlled by stategraph block for switching differential equation in state C. Stategraph get input signal from less ( $<0$ ) block and greater equal block ( $\geq 0$ ) to define which state is active is. The model for state C was shown in figure 5.26.

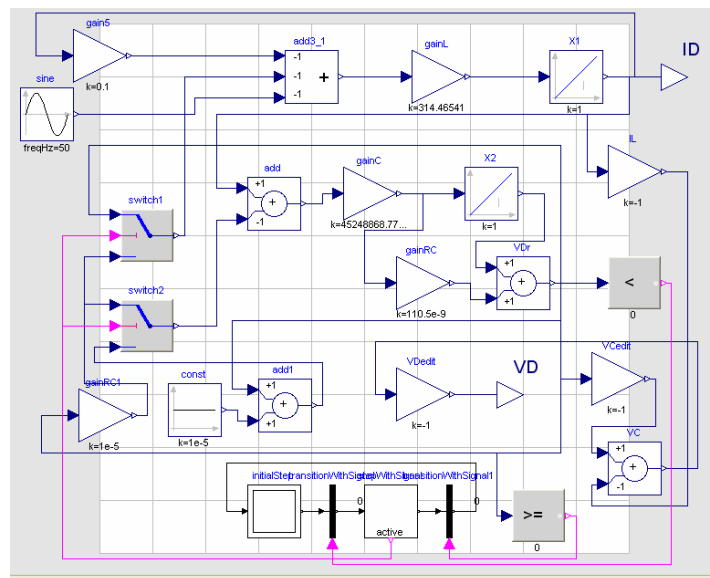


Figure 5.26 Model of the system steady states (stategraph)



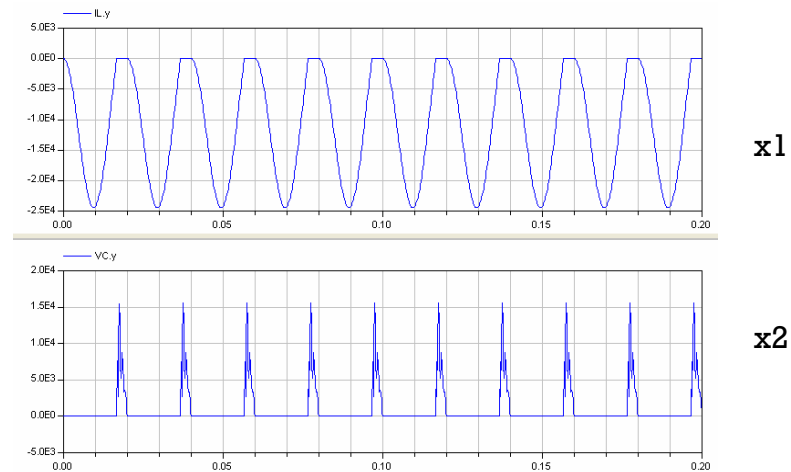


Figure 5.27 Plot x1 and x2 steady states (dymola stategraph)

### Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval,  $1e-4$  for relative tolerance and 0 ... 0,2 as simulation interval. Plot x1 and x2 is shown by figure 5.27. It took 0,234s to simulate state C.

### 5.3.2.2.2 Classical Simulation

#### Design of Model

For design of the model using only gain, add/substract and integrator block for differential equation and sine source block for the sinus voltage. Time dependent switch was built by switch block, less equal block and trapezoid source with parameter listed below:

- Amplitude =  $1e+8$
- Offset =  $1e-4$
- Rising =  $5e-3$
- Falling =  $5e-3$
- Width =  $5e-2$
- Period =  $1e-1$

Switch block was controlled by stategraph block for switching differential equation in state C. Stategraph get input signal from less equal ( $1e-4$ ) block and greater block ( $>1e-4$ ) to define which state is active is. The model for the system was shown in figure 5.28.

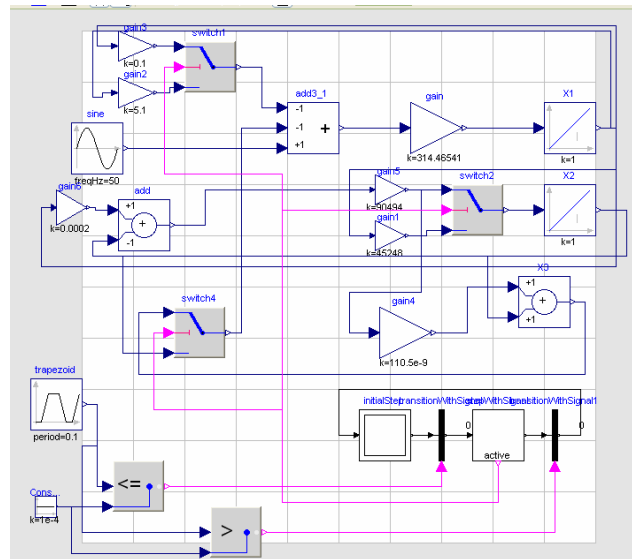
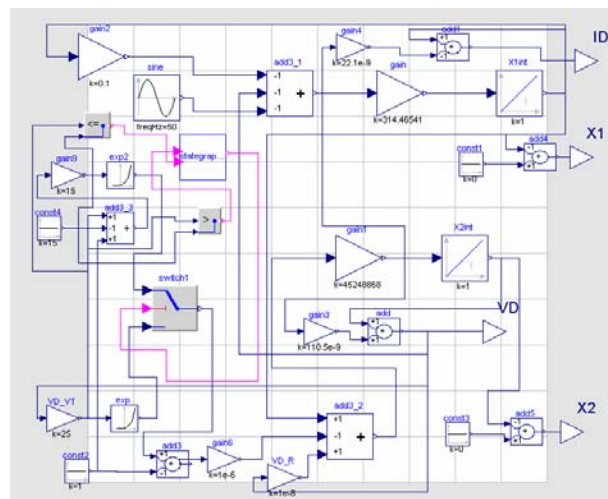


Figure 5.28 Model of the system classical simulation (stategraph)

Diode B



Diode C

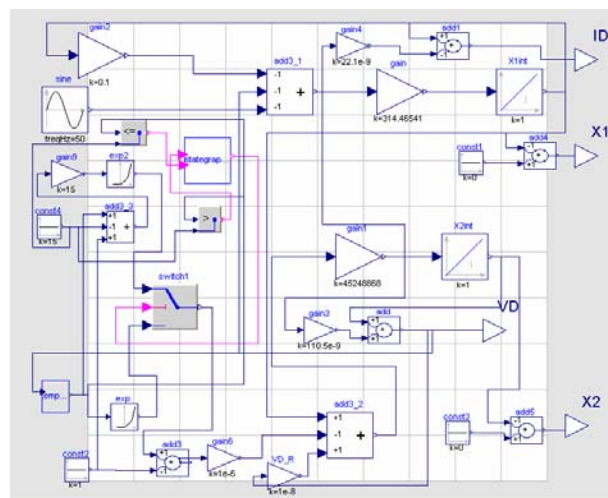


Figure 5.29 The model of the system different diode modes (stategraph)

## Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval,  $1e-4$  for relative tolerance and 0 ... 0,3 as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.21. It took 0,235s to simulate this task.

### 5.3.2.2.3 Different Diode Models

#### Design of Model

For design of the model for diode B using only gain, add/substract and integrator block for differential equation , sine source block for the sinus voltage and exponent block for exponential function. Switch block was controlled by stategraph block for switching differential equation in diode B. Stategraph get input signal from less equal ( $\leq 15$ ) block and greater equal block ( $\geq 15$ ) to define which state is active is. For diode C, the design based on diode B, with changing 1 gain block ( $VD/VT$ ) with 1 subsystem block to define the function of  $VT$  ( $VD/VT(t)$ ). The model for diode B and diode C was shown in figure 5.29.

## Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval,  $1e-8$  for relative tolerance and 0 ... 0,2 as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.30. it took.5,08s to simulate diode B and 6,11s to simulate diode C.

For whole calculation and simulation, using Dymola version 6.0b on PC Intel Pentium D, 2 x 2,66 GHz.

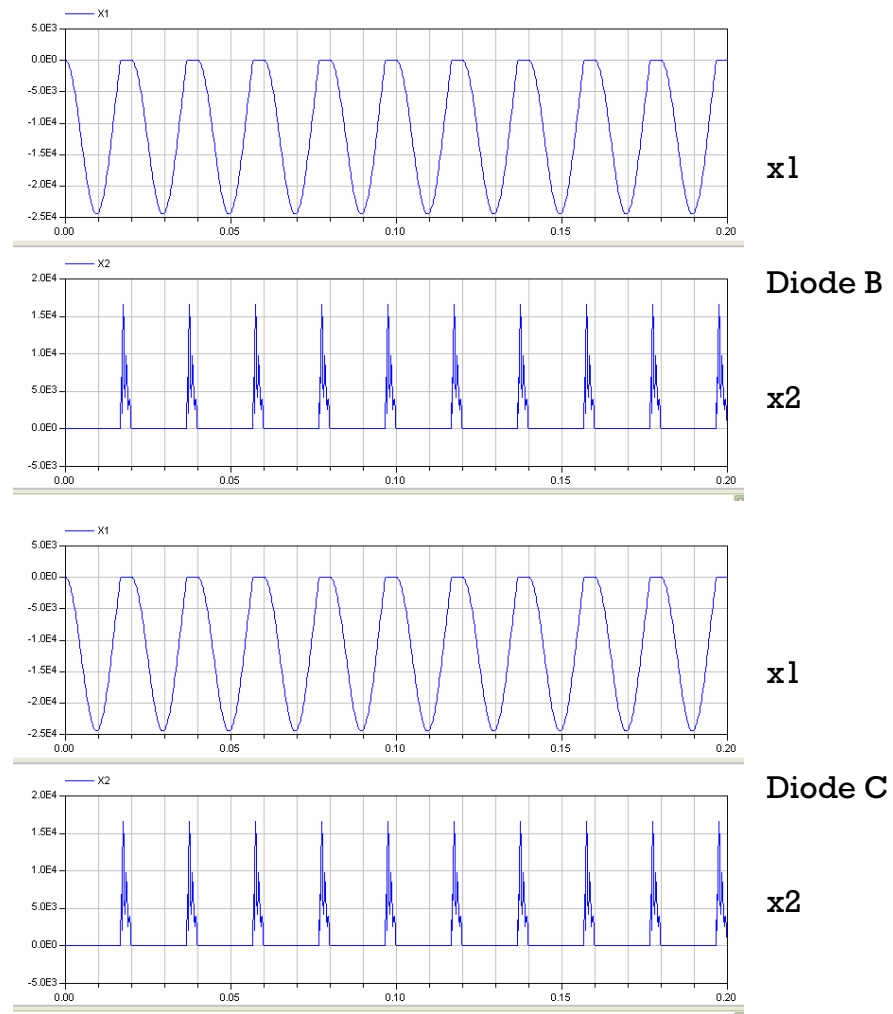


Figure 5.30 Plot x1 and x2 different diode models (stategraph)

### 5.3.3.3 Electrical Model

#### 5.3.3.3.1 Steady States

##### Design of Model

For design of the model based on figure 5.2, using basic electric resistor, capacitor, inductor and sine voltage source for state A and state B. Using diode ideal for state C. The model for state A, state B and state C was shown in figure 5.31.

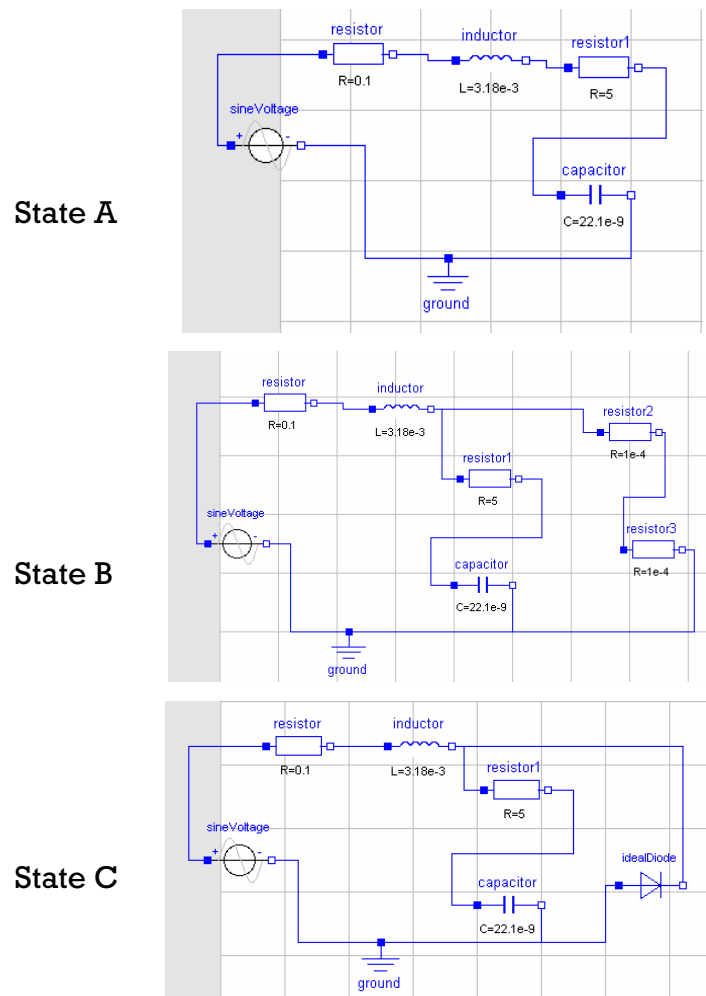


Figure 5.31 Model of the system steady states (dymola electrical)

### Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval,  $1e-4$  for relative tolerance and  $0 \dots 0,2$  as simulation interval. Plot  $x_1$  and  $x_2$  for each state is shown by figure 5.19. it took 0,031s to simulate state A, 0,031s to simulate state B and 0,188s to simulate state C.

#### 5.3.2.3.2 Classical Simulation

##### Design of Model

For design of the model using basic electric resistor, capacitor, inductor and sine voltage source Time dependent switch was built by ideal closing switch block, less equal block and trapezoid source with parameter listed below:

- Amplitude =  $1e+8$
- Rising =  $5e-3$
- Width =  $5e-2$
- Offset =  $1e-4$
- Falling =  $5e-3$
- Period =  $1e-1$

Ideal closing switch block was controlled by less equal(0) block for switching differential equation in this model. The model for the system was shown in figure 5.32.

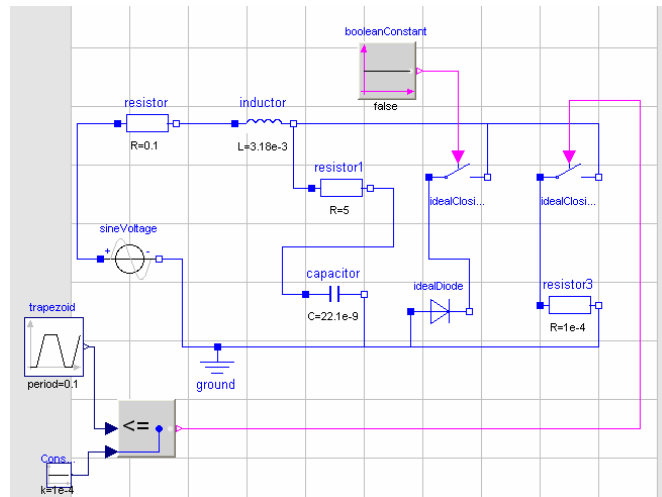


Figure 5.32 Model of the system classical simulation (dymola electrical)

## Simulation

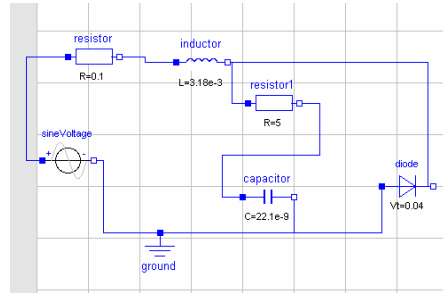
To simulate the system, using solver DASSL, 1000 as numbers of interval,  $1e-4$  for relative tolerance and 0 ... 0,3 as simulation interval. Plot x1 and x2 for this task is shown by figure 5.21. It took 0,703s to simulate this task.

### 5.3.2.3.3 Different Diode Models

#### Design of Model

For design of the model, based on state C by changing the type of diode. For diode A using ideal diode, for diode B using semiconductor diode and for diode C, using temperature diode. Sinus function and prescribed temperature as input for heating diode. The model for diode A is the same as state C, therefore the task for diode A won't be needed again. The model for diode B and diode C was shown in figure 5.33.

Diode B



Diode C

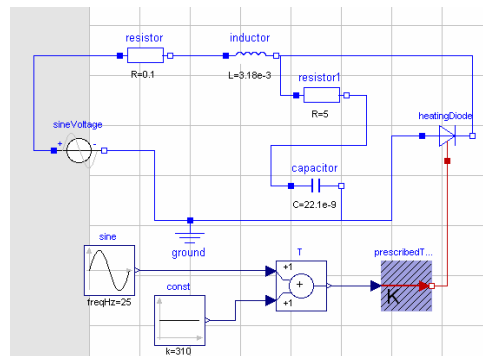


Figure 5.33 The model of the system different diode modes (dymola electrical)

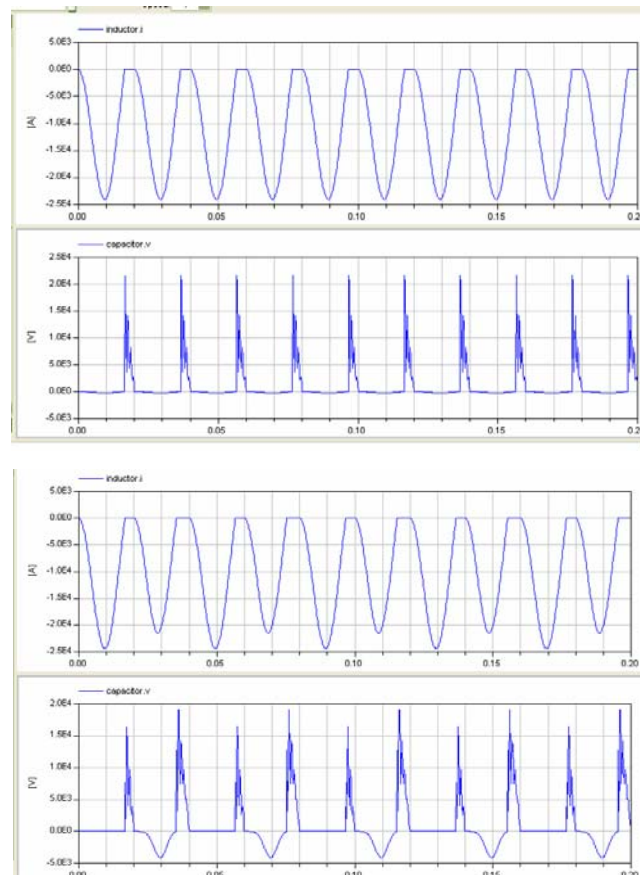


Figure 5.34 Plot x1 and x2 different diode models (dymola electrical)

## Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval,  $1e-8$  for relative tolerance and 0 ... 0,2 as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.34. it took 4,08s to simulate diode B and 4,83s to simulate diode C.

### 5.3.2.1.4 Influence of Simulation Algorithms

#### Design of Model

For design of the model based on figure 5.1 with all the switch closed and using ideal diode. The model of the system was shown in figure 5.35.

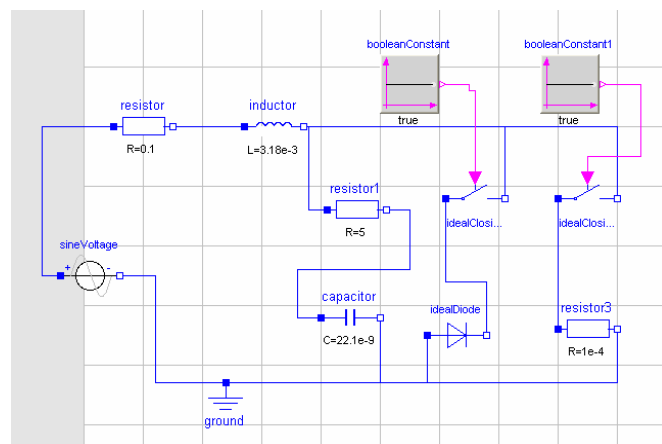


Figure 5.35 Model of the system influence of simulation algorithms (dymola electrical

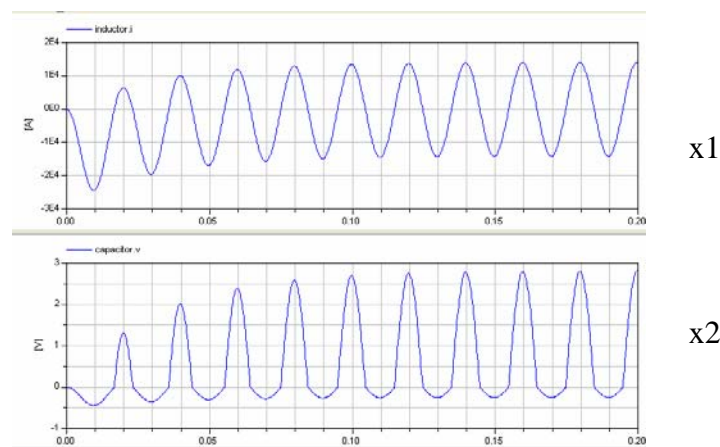


Figure 5.36 Plot  $x_1$  and  $x_2$  influence of simulation algorithms (dymola electrical



## Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval, 1e-8 for relative tolerance and 0 ... 0,2 as simulation interval. Plot x1 and x2 for this task is shown by figure 5.36. it took 0,094s to simulate this task.

For whole calculation and simulation, using Dymola version 6.0b on PC Intel Pentium D, 2 x 2,66 GHz.

### 5.3.3.4 Modelica Text Mode

#### 5.3.3.4.1 Steady States

##### Design of Model

For design of the model, using the exact differential equation with modelica function der(x) as  $dx/dt$  in the equation. The equation for state A, state B and state C was written below:

```
equation //State A
L*der(x1)=-x1*(R1+R2) - x2 + U;
C*der(x2)= x1;

equation //State B
L*der(x1)=-x1*R1 -R2*C*der(x2) - x2 + U;
C*(R2+Ron+Rmin)*der(x2)= x1*(Ron+Rmin) - x2;

equation //State C
VDr= (R2*C*der(x2)) + x2;
if VDr< 0 then
L*der(x1)=-x1*R1 - VDr - U;
C*der(x2)= x1 - VDr*1e-5;
else
L*der(x1)=-x1*R1 - VDr*1e-5 - U;
C*der(x2)= x1 - VDr;
end if;
```

## Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval, 1e-4 for relative tolerance and 0 ... 0,2 as simulation interval. Plot x1 and x2 for each state is shown by figure 5.19. it took 0,031s to simulate state A, 0,031s to simulate state B and 0,203s to simulate state C.

### 5.3.2.4.2 Classical Simulation

#### Design of Model

For design of the model, using the exact differential equation with modelica function `der(x)` as  $dx/dt$  in the equation. Time dependent switch using algorithm below:

```

equation
 t_red = mod(time, 1E-1);
 k=((1e+8)-(1e-4))/TRF;
algorithm
 if
 (0<=t_red) and (t_red<TRF) then
 Trap:=(1e-4) + k*t_red;
 elseif
 (TRF<=t_red) and (t_red<(5e-2)) then
 Trap:=1e+8;
 elseif
 ((5e-2)<=t_red) and (t_red<((5e-2)+TRF)) then
 Trap:=(1e+8) - k*(t_red - (5e-2));
 elseif
 ((5e-2)+TRF<=t_red) and (t_red<(1e-1)) then
 Trap:=1e-4;
 else
 Trap:=-5;
 end if;

```

#### Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval,  $1e-4$  for relative tolerance and 0 ... 0,3 as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.21. It took 0,219s to simulate this task.

### 5.3.2.4.3 Different Diode Models

#### Design of Model

For design of the model, using the exact differential equation with modelica function `der(x)` as  $dx/dt$  in the equation. The equation for diode A, diode B and diode C was written below:

```

equation //Diode A
 L*der(x1)= -x1*R1 -R2*C*der(x2) - x2 - U;
 C*der(x2)= x1 - ids*(exp(((R2*C*der(x2))+x2)/VT)-1);

equation //Diode B
 L*der(x1)= -x1*R1 -R2*C*der(x2) - x2 - U;
 VD= (R2*C*der(x2)) + x2;
 CTR = VD/VT;

```

```

if CTR > maxexp then
C*der(x2)= x1 - ids*(exp(maxexp*(1+CTR-maxexp))-1) + (VD/R);
else
C*der(x2)= x1 - ids*(exp(CTR)-1) + (VD/R);
end if;

equation //Diode C
L*der(x1)= -x1*R1 -R2*C*der(x2) - x2 - U;
VD= (R2*C*der(x2)) + x2;
VT= ((30 * sin(2*3.14159*100*time))+310)*8.61734681e-5;
CTR = VD/VT;
if CTR > maxexp then
C*der(x2)= x1 - ids*(exp(maxexp*(1+CTR-maxexp))-1) + (VD/R);
else
C*der(x2)= x1 - ids*(exp(CTR)-1) + (VD/R);
end if;

```

## Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval, 1e-8 for relative tolerance and 0 ... 0,2 as simulation interval. Plot x1 and x2 for this task is shown by figure 5.23. it took 2,5s to simulate diode A, .3,77s to simulate diode B and 4,39s to simulate diode C.

### 5.3.2.4.4 Influence of Simulation Algorithms

#### Design of Model

For design of the model, using the exact differential equation with modelica function der(x) as dx/dt in the equation. The equation for this task was written below:

```

equation
L*der(x1)= -x1*R1 -VD - x2 - U;
C*der(x2)= x1 - ids*(exp(VD/VT)-1) - (VD/(Ron+Rmin));

```

## Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval, 1e-8 for relative tolerance and 0 ... 0,2 as simulation interval. Plot x1 and x2 for this task is shown by figure 5.25. it took 0,125s to simulate this task.

For whole calculation and simulation, using Dymola version 6.0b on PC Intel Pentium D, 2 x 2,66 GHz.

### 5.3.4 Mosilab

#### 5.3.4.1 Modelica Text Mode

##### 5.3.4.1.1 Steady States

#### Design of Model

For design of the model, using the exact differential equation with modelica function  $\text{der}(x)$  as  $dx/dt$  in the equation. The equation for state A, state B and state C was written below:

```

equation //State A
L*der(x1)= -x1*(R1+R2) - x2 + U;
C*der(x2)= x1;

equation //State B
L*der(x1)= -x1*R1 -R2*C*der(x2) - x2 + U;
C*(R2+Ron+Rmin)*der(x2)= x1*(Ron+Rmin) - x2;

equation //State C
VDr= (R2*C*der(x2)) + x2;
if VDr< 0 then
L*der(x1)= -x1*R1 - VDr - U;
C*der(x2)= x1 - VDr*1e-5;
else
L*der(x1)= -x1*R1 - VDr*1e-5 - U;
C*der(x2)= x1 - VDr;
end if;

```

#### Simulation

To simulate the system, using solver DASSL,  $1e-6$  as min step size, 0,08 as max step size, 1.0 for relative tolerance and 0 ... 0,2 as simulation interval. Plot  $x_1$  and  $x_2$  for each state is shown by figure 5.37. It took 1s to simulate state A, 0,2s to simulate state B and 13,4s to simulate state C.

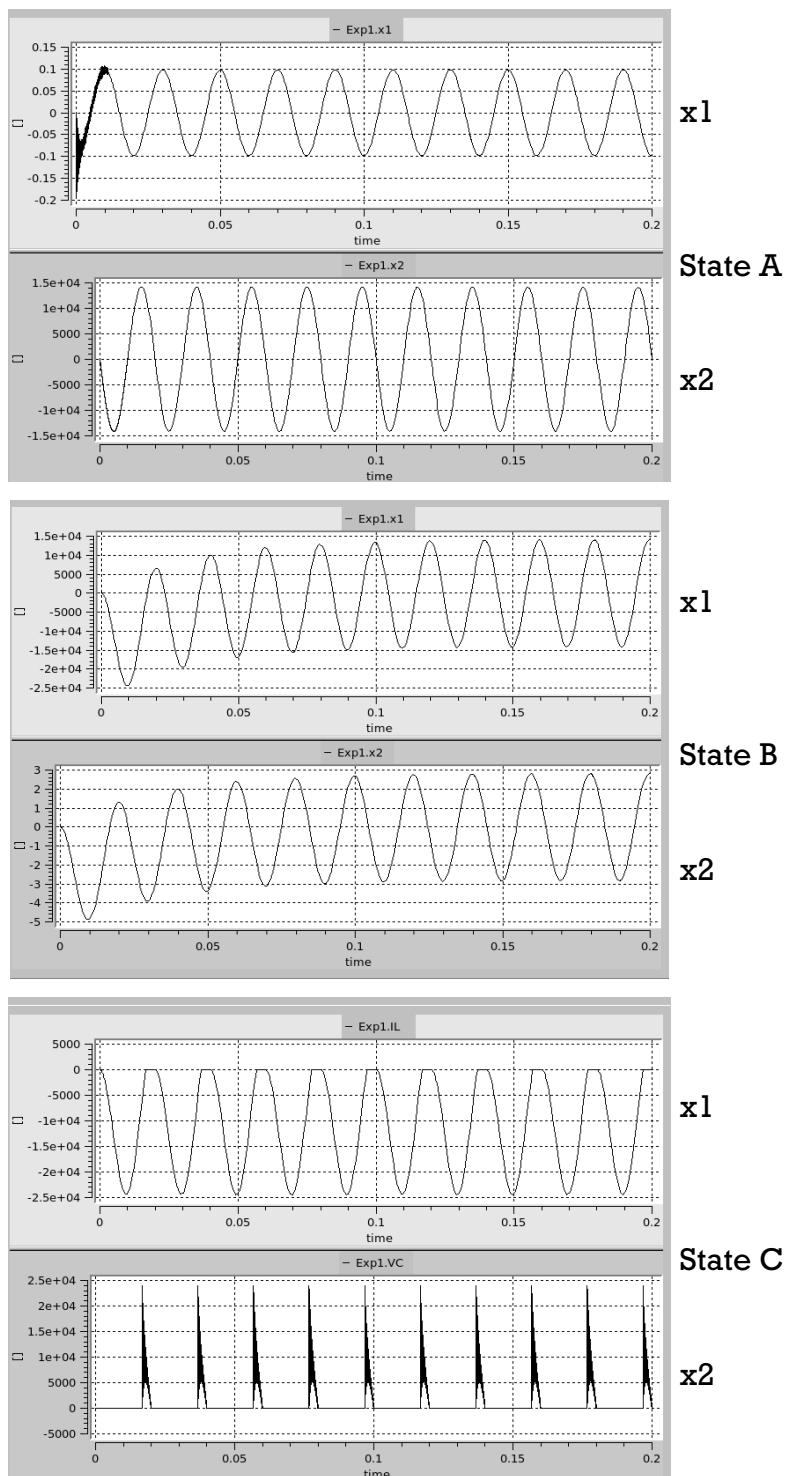


Figure 5.37 Plot  $x_1$  and  $x_2$  steady states (mosilab)

### 5.3.4.1.2 Classical Simulation

#### Design of Model

For design of the model, using the exact differential equation with modelica function `der(x)` as  $dx/dt$  in the equation. Time dependent switch using algorithm below:

```

equation
 t_red = mod(time, 1E-1);
 k=((1e+8)-(1e-4))/TRF;
algorithm
 if
 (0<=t_red) and (t_red<TRF) then
 Trap:=(1e-4) + k*t_red;
 elseif
 (TRF<=t_red) and (t_red<(5e-2)) then
 Trap:=1e+8;
 elseif
 ((5e-2)<=t_red) and (t_red<((5e-2)+TRF)) then
 Trap:=(1e+8) - k*(t_red - (5e-2));
 elseif
 ((5e-2)+TRF<=t_red) and (t_red<(1e-1)) then
 Trap:=1e-4;
 else
 Trap:=-5;
 end if;

```

#### Simulation

To simulate the system, using solver DASSL,  $1e-6$  as min step size, 0,08 as max step size, 1.0 for relative tolerance and 0 ... 0,3 as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.38. It took 9,1s to simulate this task.

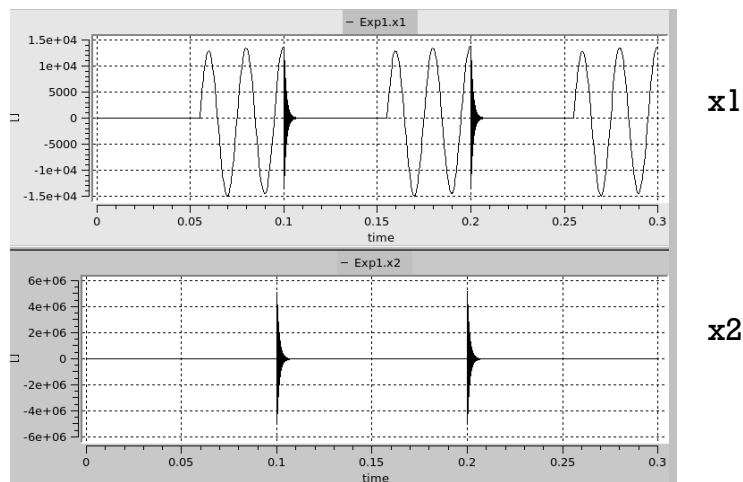


Figure 5.38 Plot  $x_1$  and  $x_2$  classical simulation (mosilab)

### 5.3.4.1.3 Different Diode Models

#### Design of Model

For design of the model, using the exact differential equation with modelica function  $\text{der}(x)$  as  $dx/dt$  in the equation. The equation for diode A, diode B and diode C was written below:

```

equation //Diode A
L*der(x1)=-x1*R1 -R2*C*der(x2) - x2 - U;
C*der(x2)= x1 - ids*(exp(((R2*C*der(x2))+x2)/VT)-1);

equation //Diode B
L*der(x1)=-x1*R1 -R2*C*der(x2) - x2 - U;
VD= (R2*C*der(x2)) + x2;
CTR = VD/VT;
if CTR > maxexp then
C*der(x2)= x1 - ids*(exp(maxexp*(1+CTR-maxexp))-1) + (VD/R);
else
C*der(x2)= x1 - ids*(exp(CTR)-1) + (VD/R);
end if;

equation //Diode C
L*der(x1)=-x1*R1 -R2*C*der(x2) - x2 - U;
VD= (R2*C*der(x2)) + x2;
VT= ((30 * sin(2*3.14159*100*time))+310)*8.61734681e-5;
CTR = VD/VT;
if CTR > maxexp then
C*der(x2)= x1 - ids*(exp(maxexp*(1+CTR-maxexp))-1) + (VD/R);
else
C*der(x2)= x1 - ids*(exp(CTR)-1) + (VD/R);
end if;

```

#### Simulation

To simulate the system, using solver DASSL,  $1e-8$  as min step size,  $1e-5$  as max step size,  $1e-8$  for relative tolerance and 0 ... 0,2 as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.39. It took 47,9s to simulate diode A, 100,9s to simulate diode B and 110,7s to simulate diode C.

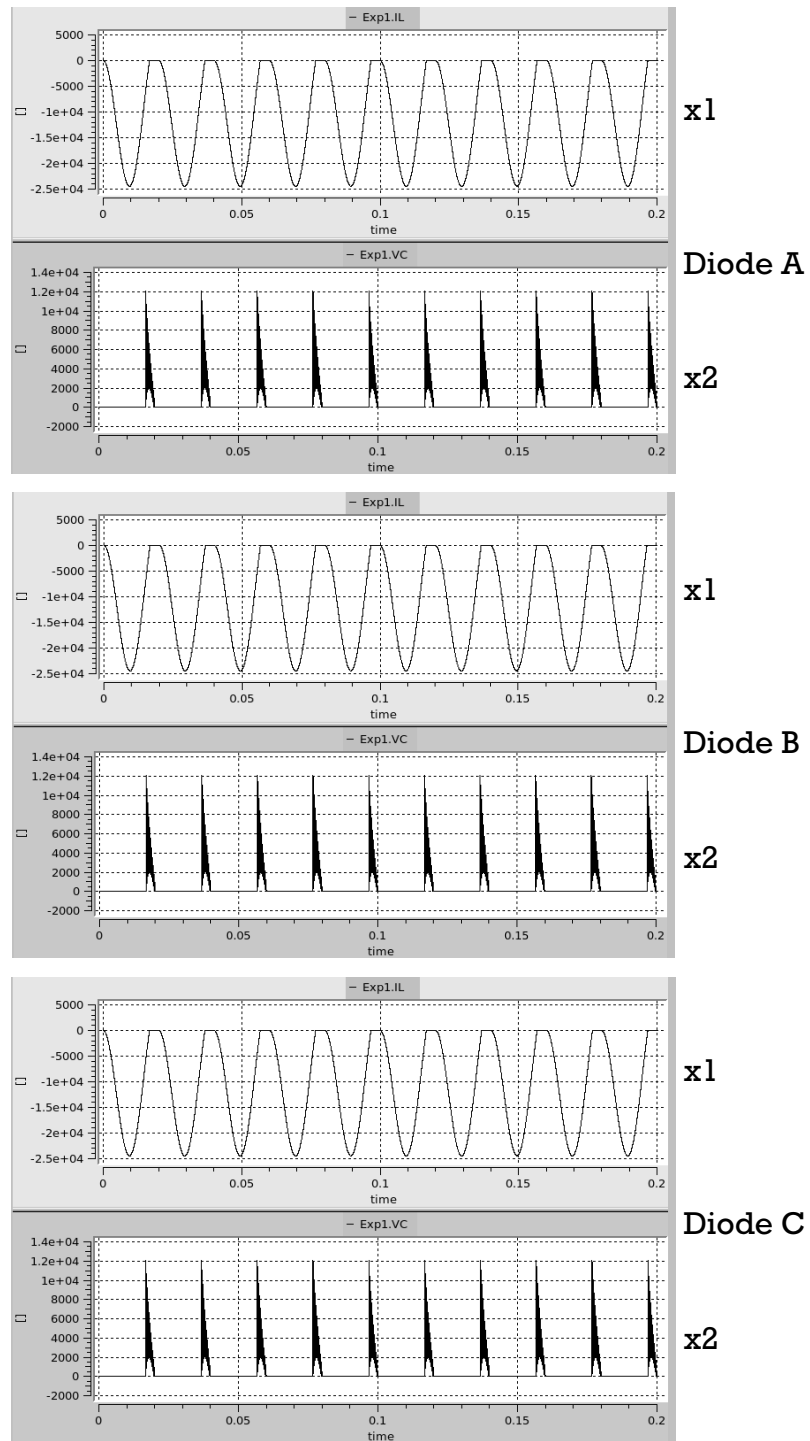


Figure 5.39 Plot  $x1$  and  $x2$  different diode models (mosilab)



#### 5.3.4.1.4 Influence of Simulation Algorithms

##### Design of Model

For design of the model, using the exact differential equation with modelica function  $\text{der}(x)$  as  $dx/dt$  in the equation. The equation for this task was written below:

*equation*

$$L * \text{der}(x1) = -x1 * R1 - VD - x2 - U;$$

$$C * \text{der}(x2) = x1 - \text{ids} * (\exp(VD/VT) - 1) - (VD / (Ron + Rmin));$$

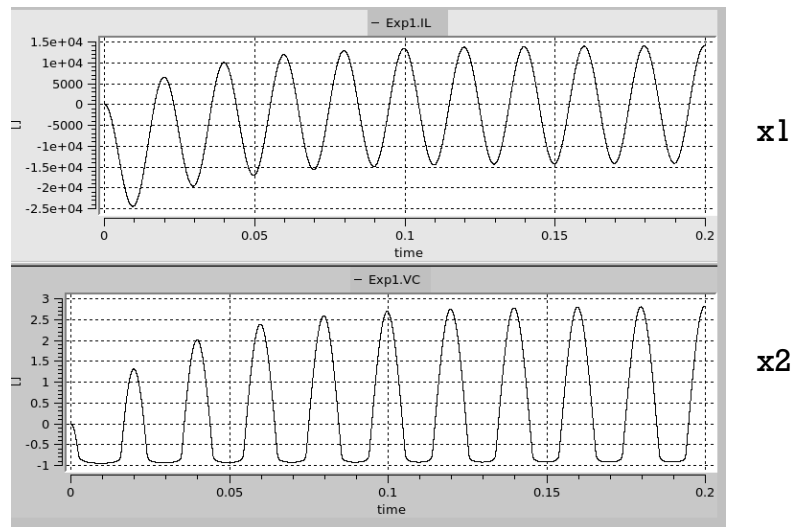


Figure 5.40 Plot x1 and x2 influence of dimulation algorithms (mosilab)

##### Simulation

To simulate the system, using solver DASSL, 1000 as numbers of interval,  $1e-8$  for relative tolerance and  $0 \dots 0,2$  as simulation interval. Plot x1 and x2 for this task is shown by figure 5.40. it took 9,7s to simulate this task.

For the calculation of condition of massmatrices can't be done by mosilab, because mosilab didn't have  $\text{cond}()$ ,  $\text{norm}()$  and  $\text{inv}()$  in their core system.

For whole calculation and simulation, using Mosilab version 3.1 on Notebook Dell Latitude D630 Intel Centrino Duo.

#### 5.3.4.2 Statechart

The same as previous, only state C, task B, diode B and diode C that can be simulated with statechart mode.

### 5.3.4.2.1 Steady States

#### Design of Model

For design of the model, using the exact differential equation with modelica function  $\text{der}(x)$  as  $dx/dt$  in the equation. The statechart for state C was written below:

```

Equation //statechart equation
s1 = if VDr >= 0 then true else false;
s2 = if VDr < 0 then true else false;
statechart //statechart algorithm
state C20MosilabSC_idealchartSC extends State;
 annotation(extent=[-103,103; 46,-46]);
 State State1 annotation(extent=[-92,60; -79,56]);
 State State2 annotation(extent=[-51,59; -38,55]);
 State Initial (isInitial=true) annotation(extent=[-85,71; -83,69]);
 transition Initial->State1 action
 A:= 1;
 end transition annotation(points=[-84,69; -84,60]);
 transition State1->State2 event s2 action
 A:= -1;
 end transition annotation(points=[-79,56; -51,56]);
 transition State2->State1 event s1 action
 A:= 1;
 end transition annotation(points=[-51,57; -79,57]);
end C20MosilabSC_idealchartSC;

```

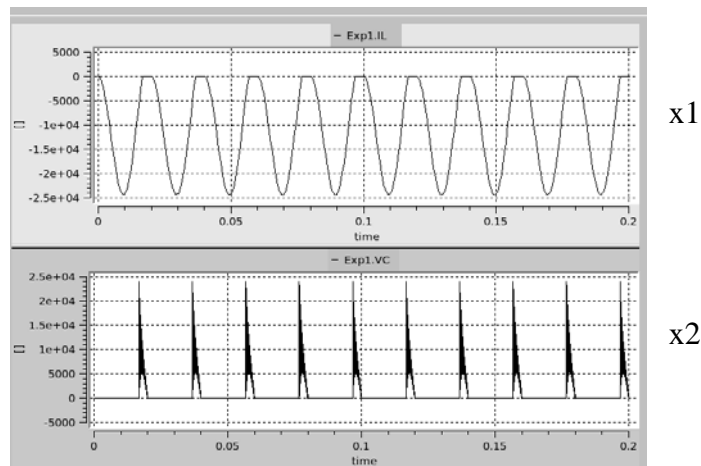


Figure 5.41 Plot x1 and x2 steady states (mosilab statechart)

#### Simulation

To simulate the system, using solver DASSL,  $1e-6$  as min step size, 0,08 as max step size, 1.0 for relative tolerance and 0 ... 0,2 as simulation interval.

Plot  $x_1$  and  $x_2$  for each state is shown by figure 5.41. It took 14,1s to simulate state C.

### 5.3.4.2.2 Classical Simulation

#### Design of Model

For design of the model, using the exact differential equation with modelica function  $\text{der}(x)$  as  $dx/dt$  in the equation. The statechart algorithm was the same as previous in steady states. The statechart equation for this task was written below:

```
equation
s1 = if Trap > 1e-4 then true else false;
s2 = if Trap <= 1e-4 then true else false; if
```

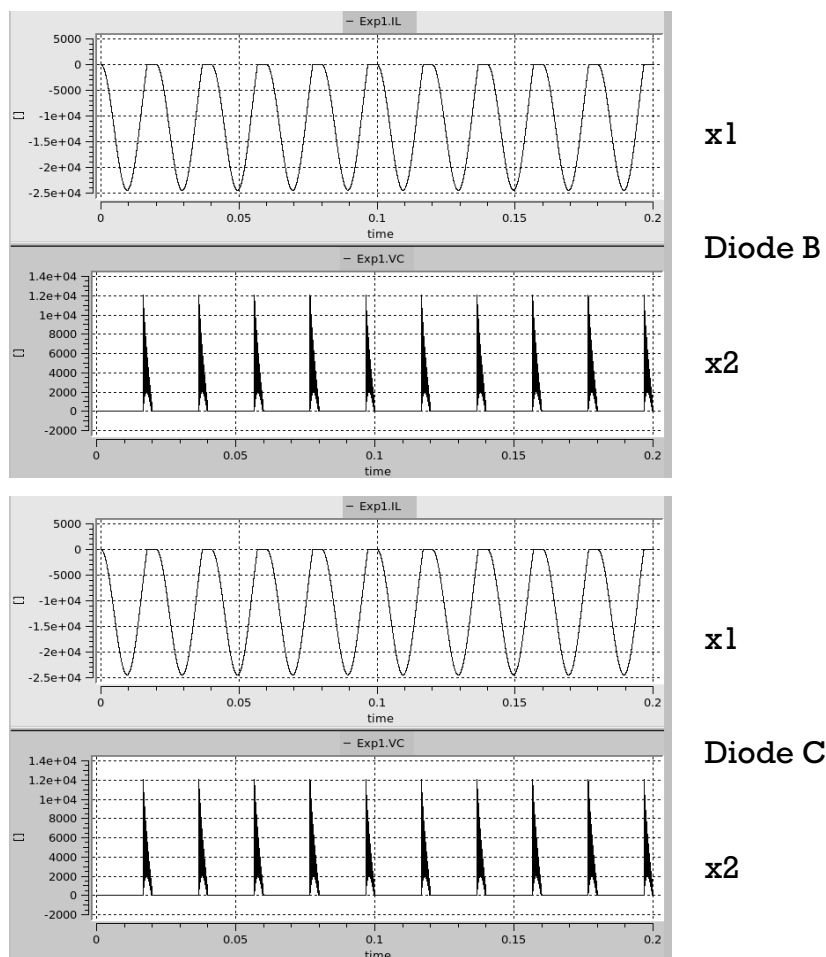


Figure 5.42 Plot  $x_1$  and  $x_2$  different diode models (mosilab statechart)

## Simulation

To simulate the system, using solver DASSL,  $1e-6$  as min step size, 0,08 as max step size, 1.0 for relative tolerance and 0 ... 0,3 as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.38. It took 8,8s to simulate this task.

### 5.3.4.2.3 Different Diode Models

#### Design of Model

For design of the model, using the exact differential equation with modelica function  $\text{der}(x)$  as  $dx/dt$  in the equation. The statechart algorithm was the same as previous in steady states. The statechart equation for diode B and diode C was written below:

```
equation //Diode B and C
s2 = if CTR>maxexp then true else false;
s1 = if CTR<=maxexp then true else false;
```

#### Simulation

To simulate the system, using solver DASSL,  $1e-8$  as min step size,  $1e-5$  as max step size,  $1e-8$  for relative tolerance and 0 ... 0,2 as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.42. It took 105,2s to simulate diode B and 116,8s to simulate diode C.

For whole calculation and simulation, using Mosilab version 3.1 on Notebook Dell Latitude D630 Intel Centrino Duo.

## 5.3.5 SimulationX

### 5.3.5.1 Hybrid Model

#### 5.3.5.1.1 Steady States

##### Design of Model

For design of the model using only gain, add/substract and integrator block for differential equation and signal generator for the sinus voltage. Relational changeover switch was controlled by 2 signal ( $s_1 < s_2$ ) for switching



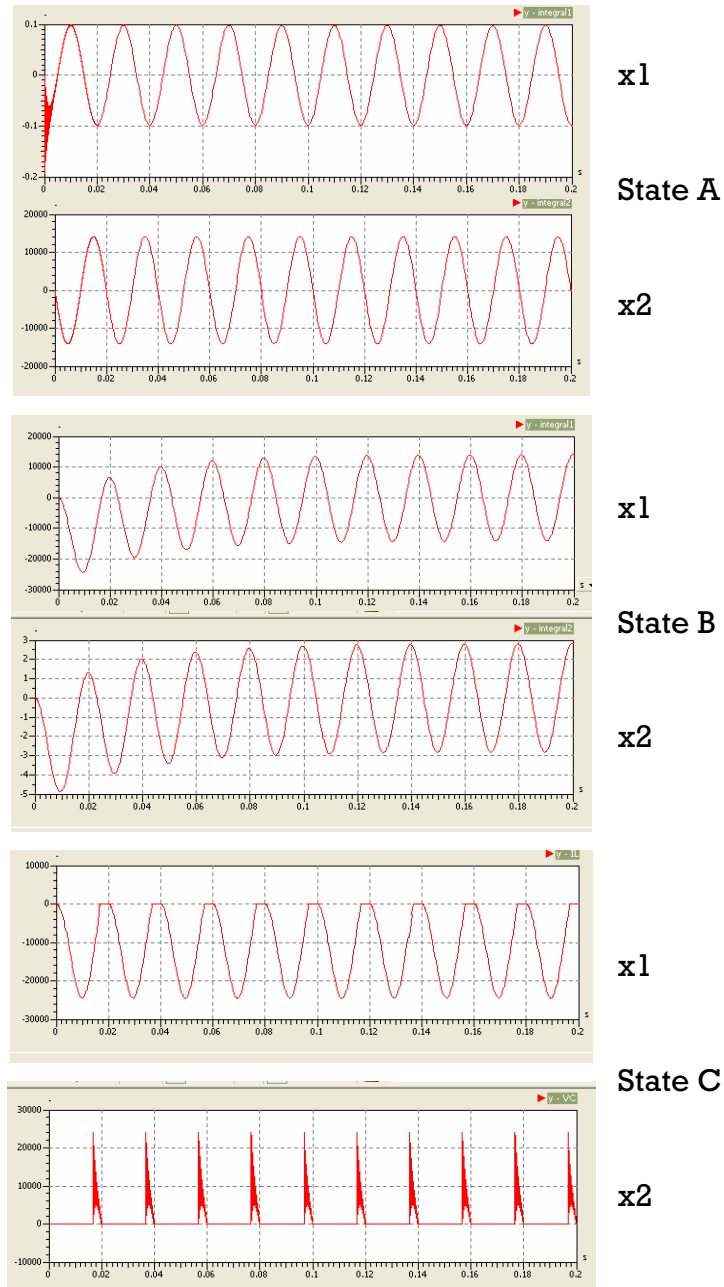


Figure 5.44 Plot  $x_1$  and  $x_2$  steady states (simulationX)

### Simulation

To simulate the system, using solver BDF-method,  $1e-14$  as min step size,  $1e-8$  as min output step size,  $1e-5$  for absolute tolerance,  $1e-5$  for relative tolerance and 0 ... 0,2 as simulation interval. Plot  $x_1$  and  $x_2$  for each state is shown by figure 5.44. it took 0,7020s to simulate state A, 0,11181s to simulate state B and 9,1124s to simulate state C.

### 5.3.5.1.2 Classical Simulation

#### Design of Model

For design of the model using only gain, add/substract and integrator block for differential equation and signal generator for the sinus voltage. Time dependent switch was built by type designer block using modelica code and relational changeover switch block. Modelica code was written below:

```

algorithm
 if
 (0<=t_red) and (t_red<TRF) then
 Trap:=(1e-4) + k*t_red;
 elseif
 (TRF<=t_red) and (t_red<(5e-2)) then
 Trap:=1e+8;
 elseif
 ((5e-2)<=t_red) and (t_red<((5e-2)+TRF)) then
 Trap:=(1e+8) - k*(t_red - (5e-2));
 elseif
 ((5e-2)+TRF<=t_red) and (t_red<(1e-1)) then
 Trap:=1e-4;
 else
 Trap:=-5;
 end if;
 end if;

equation
 t_red = mod(time, 1E-1);
 k=((1e+8)-(1e-4))/TRF;

```

Relational changeover switch block was controlled by 2 signal (s1<=s2) for switching differential equation in this model. The model for the system was shown in figure 5.45.

#### Simulation

To simulate the system, using solver BDF-method, 1e-16 as min step size, 1e-8 as min output step size, 1e-5 for absolute tolerance, 1e-5 for relative tolerance and 0 ... 0,3 as simulation interval. Plot x1 and x2 for this task is shown by figure 5.46. It took 7,7191s to simulate this task.

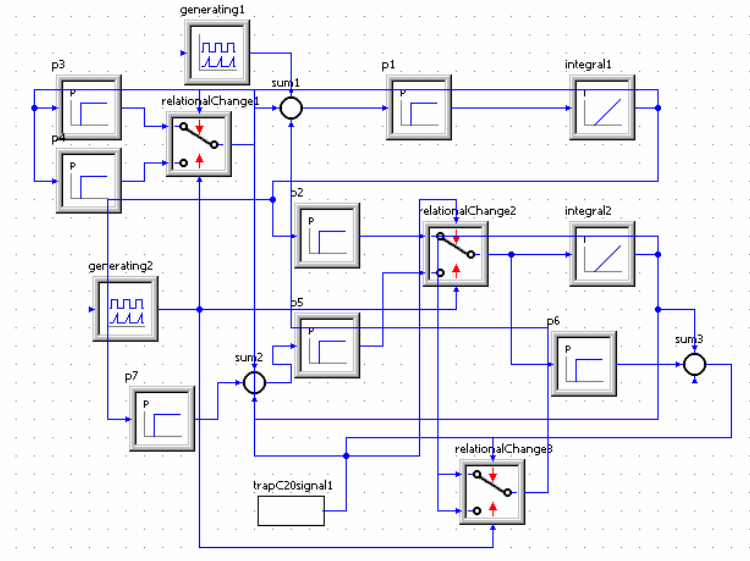


Figure 5.45 Model of the system classical simulation (simulationX)

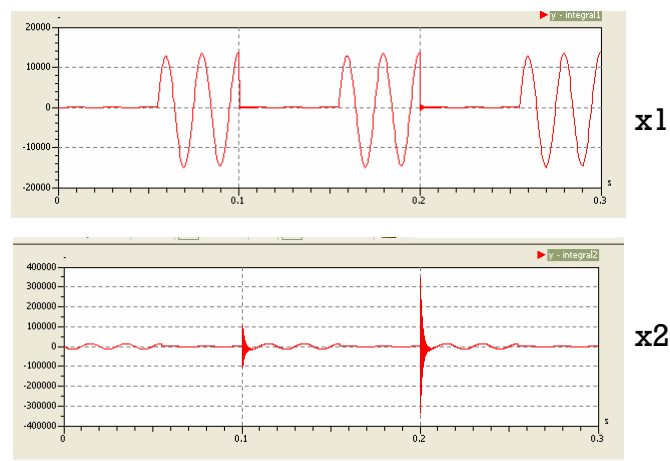


Figure 5.46 Plot x1 and x2 classical simulation (simulationX)

### 5.3.5.1.3 Different Diode Models

#### Design of Model

For design of the model for diode A using only gain, add/substract and integrator block for differential equation , signal generator for the sinus voltage and exponent block for exponential function. For diode B, the design is similar with diode A but with addition relational changeover switch block that was controlled by 2 signal( $s1 > s2$ ) for switching differential equation in diode B. For diode C, the design based on diode B, with changing 1 gain block ( $V_D/V_T$ ) with 2 signal generator(as sinus and constant), 1



add/subtract block, 1 gain block and 1 function block (as division) to define the function of  $V_T$  ( $V_D/V_T(t)$ ). The model for diode A, diode B and diode C was shown in figure 5.47.

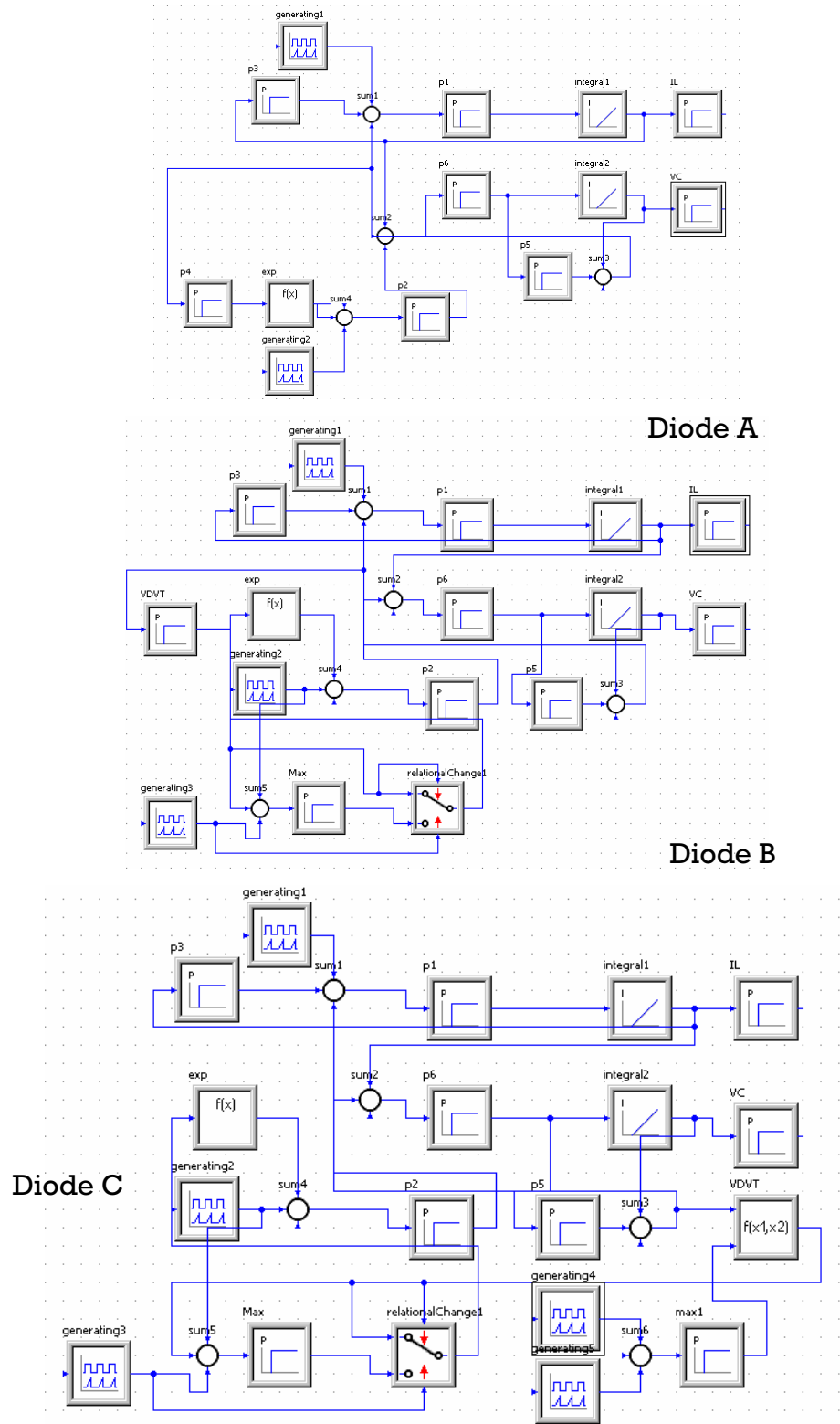
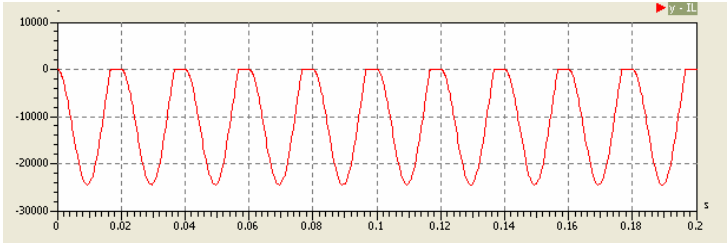
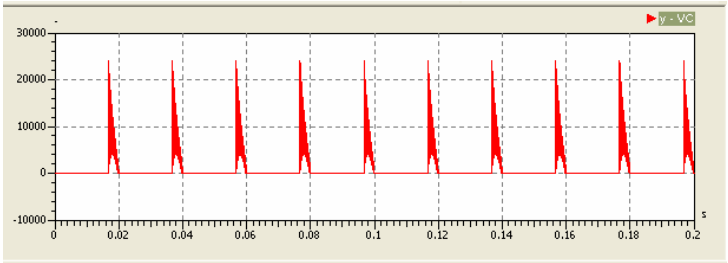


Figure 5.47 The model of the system different diode modes (simulationX)

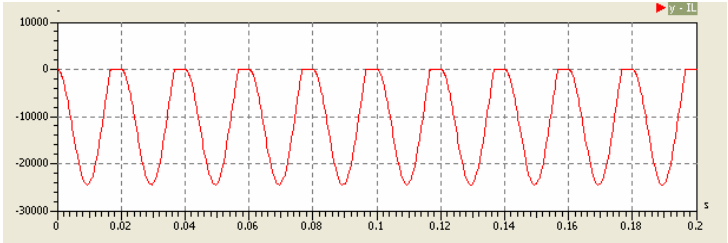


x1

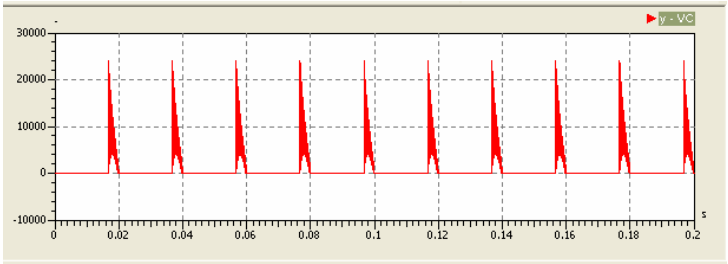


Diode A

x2

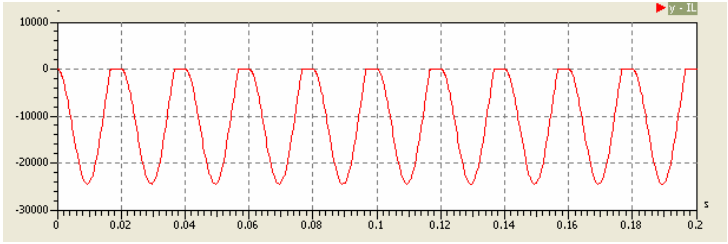


x1

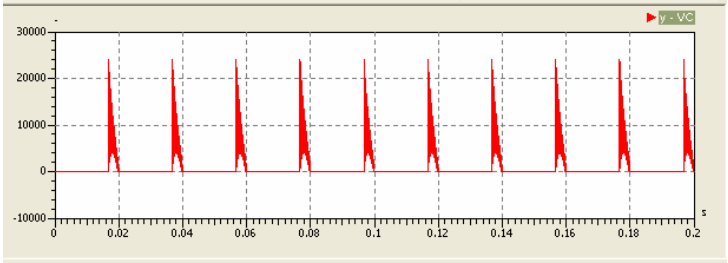


Diode B

x2



x1



Diode C

x2

Figure 5.48 Plot x1 and x2 different diode models (simulationX)

## Simulation

To simulate the system, using solver BDF-method,  $1e-18$  as min step size,  $1e-8$  as min output step size,  $1e-5$  for absolute tolerance,  $1e-5$  for relative tolerance and  $0 \dots 0,2$  as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.48. it took 8,9803s to simulate diode A, .10,0866s to simulate diode B and 1307,6718s to simulate diode C.

### 5.3.5.1.4 Influence of Simulation Algorithms

#### Design of Model

For design of the model using only gain, add/subtract and integrator block for differential equation and signal generator for the sinus voltage. The model of the system was shown in figure 5.49.

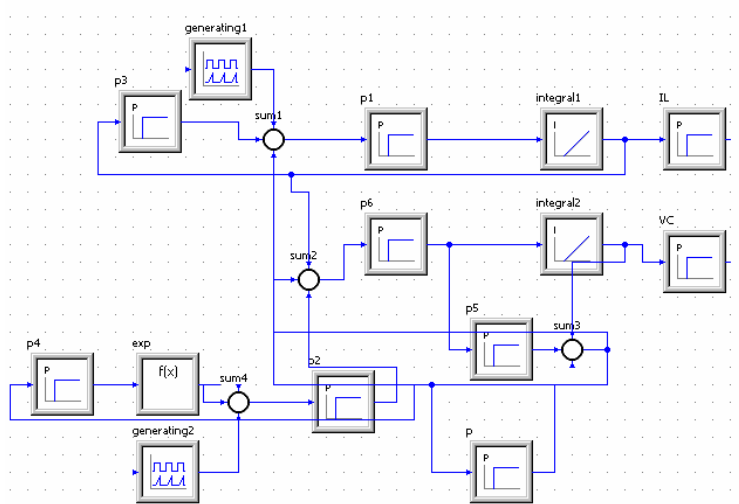


Figure 5.49 Model of the system influence of simulation algorithms  
(simulationX)

## Simulation

To simulate the system, using solver BDF-method,  $1e-8$  as min step size,  $1e-4$  as min output step size,  $1e-5$  for absolute tolerance,  $1e-5$  for relative tolerance and  $0 \dots 0,2$  as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.50. it took 560,5021s to simulate this task.

For calculation of condition of massmatrices, can't be done by simulationX because simulationX didn't have function that can calculate the condition of massmatrices.

For whole calculation and simulation, using SimulationX version 2.0 on PC Intel Pentium D, 2 x 2,66 GHz.

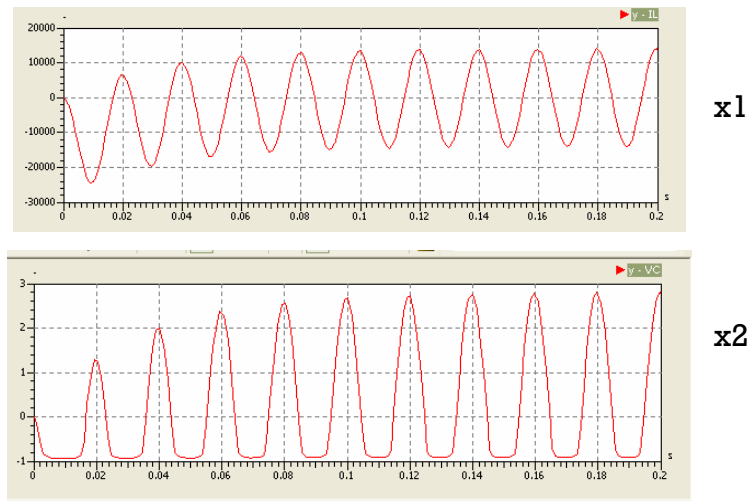


Figure 5.50 Plot x1 and x2 influence of simulation algorithms (simulationX)

### 5.3.5.2 Electrical Model

#### 5.3.5.2.1 Steady States

##### Design of Model

For design of the model based on figure 5.2 using resistor, inductor, capacitor and sine voltage source. For state C, ideal diode was used. The model for state A, state B and state C was shown in figure 5.51.

##### Simulation

To simulate the system, using solver BDF-method,  $1e-14$  as min step size,  $1e-8$  as min output step size,  $1e-5$  for absolute tolerance,  $1e-5$  for relative tolerance and 0 ... 0,2 as simulation interval. Plot x1 and x2 for each state is shown by figure 5.44. It took 0,6697s to simulate state A, 0,0844s to simulate state B and 16,2727s to simulate state C.

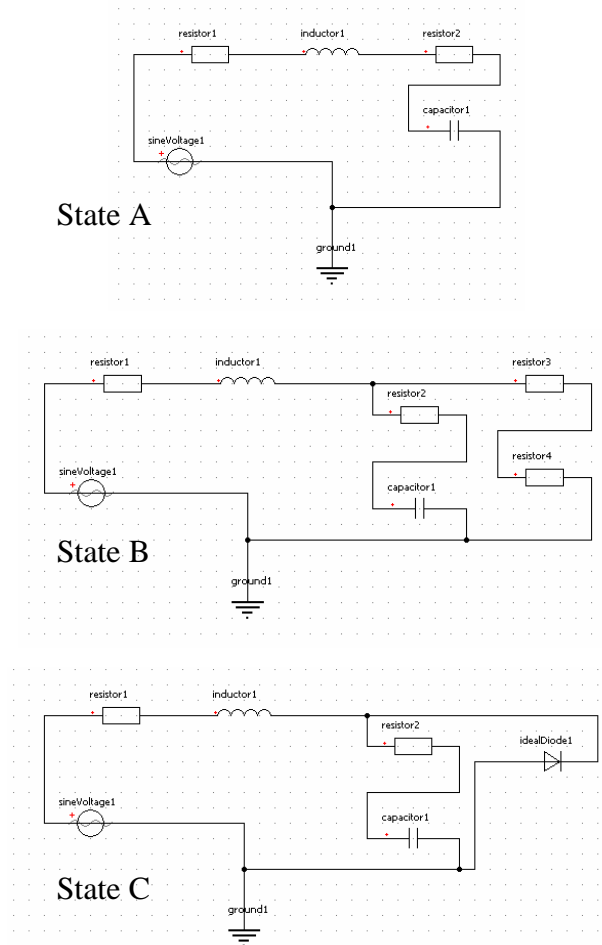


Figure 5.51 Model of the system steady states (simulationX electrical)

### 5.3.5.2.2 Classical Simulation

#### Design of Model

For design of the model based on state B changing 1 resistor with 1 ideal switch. The ideal switch as time dependent switch was controlled by type designer block using modelica code that was used on hybrid model and relational changeover switch to assign when ideal switch open or closed. The model of the system was shown by figure 5.52

#### Simulation

To simulate the system, using solver BDF-method,  $1e-14$  as min step size,  $1e-8$  as min output step size,  $1e-5$  for absolute tolerance,  $1e-5$  for relative tolerance and 0 ... 0,3 as simulation interval. Plot x1 and x2 for this task is shown by figure 5.53. It took 9,3952s to simulate this task.

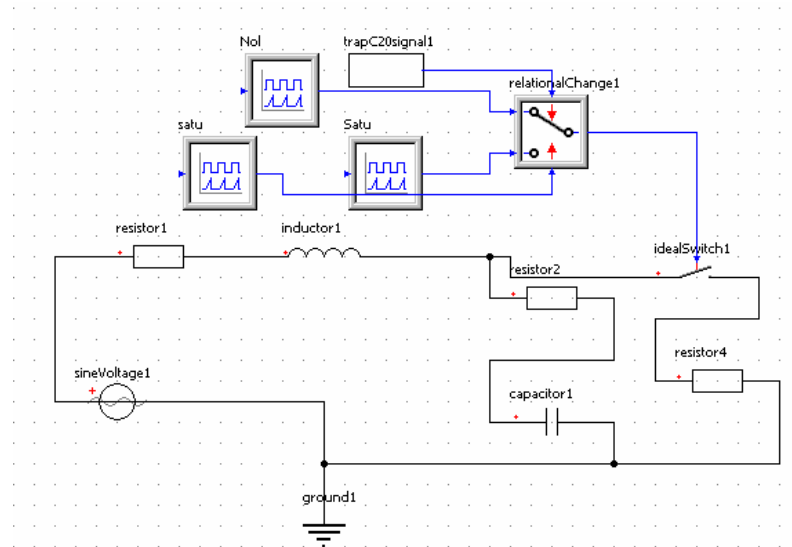


Figure 5.52 Model of the system classical simulation (simulationX electrical)

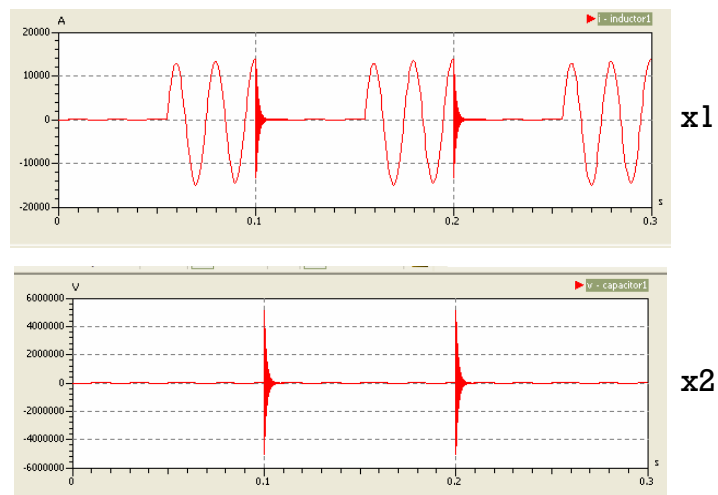


Figure 5.53 Plot x1 and x2 classical simulation (simulationX electrical)

### 5.3.5.2.3 Different Diode Models

#### Design of Model

For design of the model based on state C with different diode model, Model diode A is the same as state C using ideal diode, For model diode B using semiconductor diode and model diode C can't be done by simulationX, because simulationX have only 2 type of diode. The model for diode B was shown in figure 5.54.

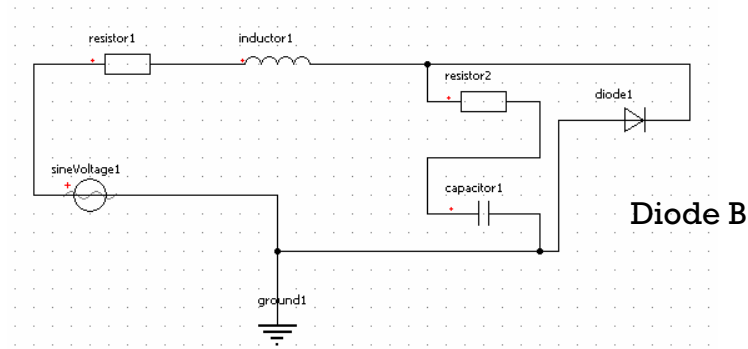


Figure 5.54 The model of the system for diode B (simulationX electrical)

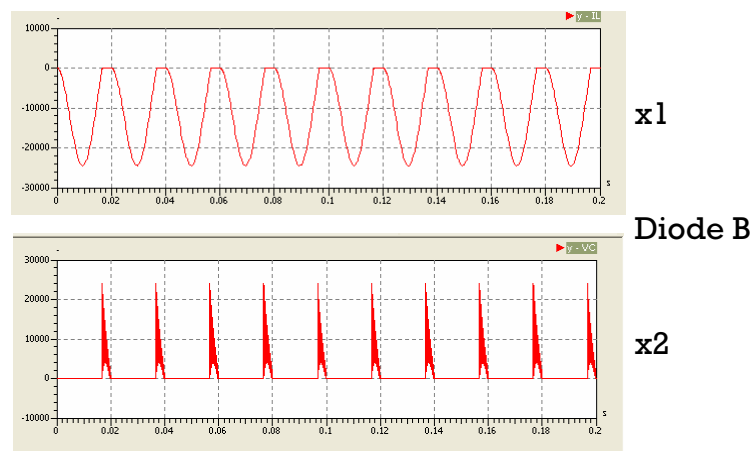


Figure 5.55 Plot x1 and x2 different for diode B (simulationX electrical)

### Simulation

To simulate the system, using solver BDF-method,  $1e-14$  as min step size,  $1e-8$  as min output step size,  $1e-5$  for absolute tolerance,  $1e-5$  for relative tolerance and 0 ... 0,2 as simulation interval. Plot x1 and x2 for this task is shown by figure 5.55. It took 8,6296s to simulate diode B.

#### 5.3.5.2.4 Influence of Simulation Algorithms

##### Design of Model

For design of the model based on figure 5.1 with all switch closed. The model of the system was shown in figure 5.56.

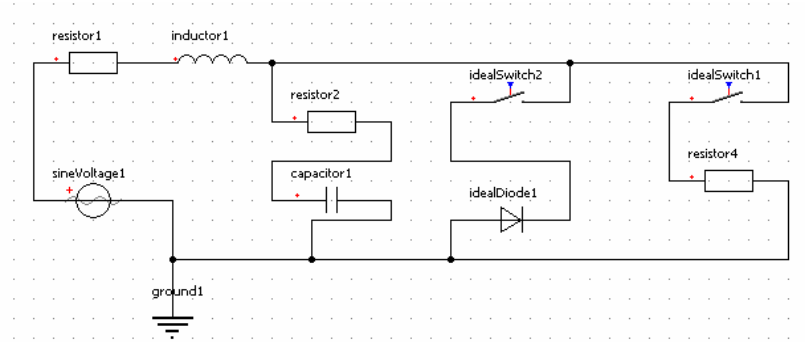


Figure 5.56 Model of the system influence of simulation algorithms  
(simulationX)

### Simulation

To simulate the system, using solver BDF-method,  $1e-14$  as min step size,  $1e-8$  as min output step size,  $1e-5$  for absolute tolerance,  $1e-5$  for relative tolerance and  $0 \dots 0,2$  as simulation interval. Plot  $x_1$  and  $x_2$  for this task is shown by figure 5.57. it took  $0,3801s$  to simulate this task.

For whole calculation and simulation, using SimulationX version 2.0 on PC Intel Pentium D,  $2 \times 2,66$  GHz.

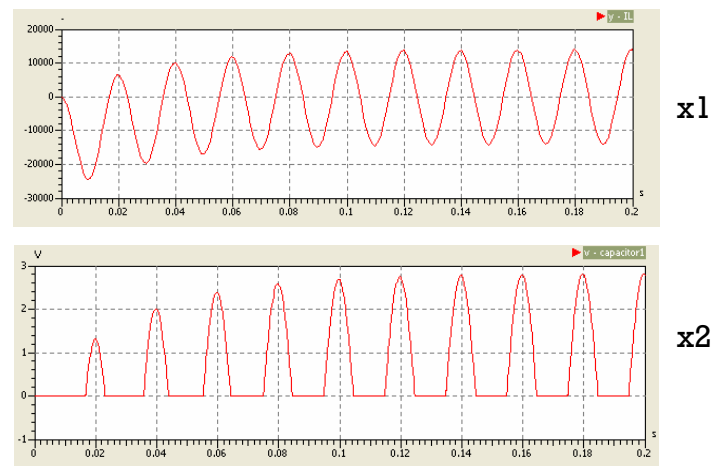


Figure 5.57 Plot  $x_1$  and  $x_2$  influence of simulation algorithms (simulationX)



## 6. Comparison

### 6.1 Table of Result

Based on the design and solutions of each comparison discussed in the previous chapter, the result can be tabulated as follow:

#### 6.1.1. Comparison 3

As analyzed in chapter III, table 6-1 shows which software is able to complete the task.

Table 6-1 The Result of the simulation software completing the task for comparison 3

| Task | Matlab | Simulink           |                    | Dymola             |                    |                    |                    | Mosilab            |                    | SimulationX        |                    |
|------|--------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
|      | T      | M                  | S                  | M                  | S                  | E                  | T                  | T                  | S                  | M                  | E                  |
| A    | V      | $\mathbf{X}^{(1)}$ | $\mathbf{X}^{(1)}$ | $\mathbf{V}^{(2)}$ | $\mathbf{V}^{(2)}$ | $\mathbf{V}^{(2)}$ | $\mathbf{V}^{(2)}$ | $\mathbf{X}^{(3)}$ | $\mathbf{X}^{(3)}$ | $\mathbf{V}^{(5)}$ | $\mathbf{V}^{(5)}$ |
| B    | V      | V                  | V                  | V                  | V                  | V                  | V                  | V                  | V                  | V                  | V                  |
| C    | V      | V                  | V                  | V                  | V                  | V                  | V                  | $\mathbf{X}^{(4)}$ | $\mathbf{X}^{(4)}$ | V                  | V                  |

Note: T = text mode

M= hybrid model

S= stateflow/stategraph/statechart mode

E= Electrical model

- (1) Simulink is unable to complete task A, this task can be done only by using matlab code.
- (2) Doing the task using the same method “calling the eigenValue function”
- (3) Mosilab is unable to complete task A, due to the lack of function to calculate the eigenvalue
- (4) Mosilab is unable to complete the task, because it does not have any function to create a phase simulation (xy plot, phase plot)

- (5) Doing the task with the same method, step one is to simulate the system and then proceed to tab analysis (natural frequencies and mode shapes).

Table 6-2 described the list of the time needed to simulate the task for comparison 3.

Table 6-2 The list of the time needed to simulate the task for comparison 3

| Task | Matlab   | Simulink |          | Dymola      |                  |                  |                  |
|------|----------|----------|----------|-------------|------------------|------------------|------------------|
|      | T        | M        | S        | M           | S                | E                | T                |
| A    | 0,1015   | <b>X</b> | <b>X</b> | 0,5         | = <sup>(1)</sup> | = <sup>(1)</sup> | = <sup>(1)</sup> |
| B    | 4,3442   | 4        | 3        | 0,047       | 0,063            | 0,047            | 0,047            |
| C    | 1,075    | 9,5      | 2        | 0,025       | 0,047            | 0,015            | 0,031            |
| Task | Mosilab  |          |          | SimulationX |                  |                  |                  |
|      | T        |          | S        | M           |                  | E                |                  |
| A    | <b>X</b> |          | <b>X</b> | 0,0723      |                  | 0,0133           |                  |
| B    | 1,3      |          | 8,5      | 1,2528      |                  | 0,8461           |                  |
| C    | <b>X</b> |          | <b>X</b> | 0,145       |                  | 0,11             |                  |

Note: All value is in second (s)

- (1) Because it is using the same method, therefore the-timing would also be the same

### 6.1.2. Comparison 5

As analyzed in the chapter IV, table 6-3 shows which software is able to complete the task.

Table 6-3 The Result of the simulation software completing the task for  
comparison 5

| Task | Matlab | Simulink |   | Dymola |   |                         |   | Mosilab |   | SimulationX |                         |
|------|--------|----------|---|--------|---|-------------------------|---|---------|---|-------------|-------------------------|
|      | T      | M        | S | M      | S | E                       | T | T       | S | M           | E                       |
| A    | V      | V        | V | V      | V | <b>X</b> <sup>(1)</sup> | V | V       | V | V           | <b>X</b> <sup>(2)</sup> |
| B    | V      | V        | V | V      | V | <b>X</b> <sup>(1)</sup> | V | V       | V | V           | <b>X</b> <sup>(2)</sup> |
| C    | V      | V        | V | V      | V | <b>X</b> <sup>(1)</sup> | V | V       | V | V           | <b>X</b> <sup>(2)</sup> |
| D    | V      | V        | V | V      | V | <b>X</b> <sup>(1)</sup> | V | V       | V | V           | <b>X</b> <sup>(2)</sup> |

Note: (1)&(2) Because comparison 5 itself does not have any electrical circuit, therefore no electrical model can be applied for this comparison.

Table 6-4 described the list of the time needed to simulate the task for comparison 5.

Table 6-4 The list of the time needed to simulate the task for comparison 5

| Task | Matlab   | Simulink |          | Dymola      |          |          |          |
|------|----------|----------|----------|-------------|----------|----------|----------|
|      | T        | M        | S        | M           | S        | E        | T        |
| A    | 0,8903   | 0,5      | 9        | 0,047       | 0,062    | <b>X</b> | 0,047    |
| B    | <b>X</b> | <b>X</b> | <b>X</b> | <b>X</b>    | <b>X</b> | <b>X</b> | <b>X</b> |
| C    | <b>X</b> | <b>X</b> | <b>X</b> | <b>X</b>    | <b>X</b> | <b>X</b> | <b>X</b> |
| D    | 10,7892  | 3        | 14       | 0,204       | 0,25     | <b>X</b> | 0,187    |
| Task | Mosilab  |          |          | SimulationX |          |          |          |
|      | T        |          | S        | M           |          | E        |          |
| A    | 0,1      |          | 0,3      | 0,0582      |          | <b>X</b> |          |
| B    | <b>X</b> |          | <b>X</b> | <b>X</b>    |          | <b>X</b> |          |
| C    | <b>X</b> |          | <b>X</b> | <b>X</b>    |          | <b>X</b> |          |
| D    | 1,3      |          | 2,3      | 0,434       |          | <b>X</b> |          |

Note: - All value is in second (s)

- Task B and C does not have time for the simulation, because these tasks are only the result of data from task A.

### 6.1.3. Comparison 20

As analysed in chapter V, table 6-5 shows which software is able to complete the task.

Table 6-5 The Result of the simulation software completing the task for comparison 20

| Task | Matlab         | Simulink           |                    | Dymola             |                    |                    |                    | Mosilab            |                    | SimulationX        |                     |
|------|----------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|---------------------|
|      | T              | M                  | S                  | M                  | S                  | E                  | T                  | T                  | S                  | M                  | E                   |
| A-sa | V              | V                  | $\mathbf{X}^{(3)}$ | V                  | $\mathbf{X}^{(5)}$ | V                  | V                  | V                  | $\mathbf{X}^{(5)}$ | V                  | V                   |
| sb   | V              | V                  | $\mathbf{X}^{(3)}$ | V                  | $\mathbf{X}^{(5)}$ | V                  | V                  | V                  | $\mathbf{X}^{(5)}$ | V                  | V                   |
| sc   | $V_{ns}^{(1)}$ | V                  | $\mathbf{X}^{(3)}$ | V                  | $\mathbf{X}^{(5)}$ | V                  | V                  | V                  | $\mathbf{X}^{(5)}$ | V                  | V                   |
| B    | V              | V                  | V                  | V                  | V                  | V                  | V                  | V                  | V                  | $V_{diff}^{(10)}$  | V                   |
| C-da | $V_{ns}^{(1)}$ | V                  | $\mathbf{X}^{(3)}$ | V                  | V                  | $\mathbf{X}^{(6)}$ | V                  | V                  | V                  | V                  | $\mathbf{X}^{(6)}$  |
| db   | $V_{ns}^{(1)}$ | V                  | $\mathbf{X}^{(3)}$ | V                  | V                  | V                  | V                  | V                  | V                  | V                  | V                   |
| dc   | $V_{ns}^{(1)}$ | V                  | $\mathbf{X}^{(3)}$ | V                  | V                  | $V_{diff}^{(7)}$   | V                  | V                  | V                  | V                  | $\mathbf{X}^{(11)}$ |
| D    | $V_{ns}^{(1)}$ | V                  | $\mathbf{X}^{(3)}$ | V                  | $\mathbf{X}^{(5)}$ | $V_{diff}^{(8)}$   | V                  | V                  | $\mathbf{X}^{(5)}$ | V                  | $V_{diff}^{(12)}$   |
| cond | V              | $\mathbf{X}^{(2)}$ | $\mathbf{X}^{(2)}$ | $\mathbf{X}^{(4)}$ | $\mathbf{X}^{(4)}$ | $\mathbf{X}^{(4)}$ | $\mathbf{X}^{(4)}$ | $\mathbf{X}^{(9)}$ | $\mathbf{X}^{(9)}$ | $\mathbf{X}^{(9)}$ | $\mathbf{X}^{(9)}$  |

Note: sa=state A

sb= stateB

sc= State C

da= diode A

db= diode B

dc= diode C

ns= not satisfied

cond= condition. Only the equation in explicit form that its condition can be calculated

diff= the plot is different than others.

- (1) Using ode15i can simulate the task, but the result isn't satisfactory.
- (2) Simulink can't calculate the condition, this task can only be done by using matlab code.
- (3) Only the equation that have switching state and in explicit form can be modelled in stateflow model. But stateflow have restriction, which does not allow algebraic loop inside their stateflow, that's why only task B's condition can be calculated.
- (4) Dymola can calculate indirect condition number with equation:  

$$\text{cond} = \text{norm}(A,p) * \text{norm}(\text{inv}(A,p))$$
 But in this task, dymola can't calculate condition because dymola can't inverse the system matrix in it. Probably the matrix is too stiff.
- (5) Only the equation that have switching state and in explicit form can be modelled in stategraph model in dymola and statechart mode in mosilab
- (6) Diode A is the same as state C
- (7) Using heating diode made the plot different than others
- (8) The plot is slightly different than others. This plot is going to have a more stable result than the plot from other model and textual mode
- (9) Mosilab and simulationX can't calculate condition number, because they don't have any function for this task.
- (10) The plot from hybrid model simulationX slightly different than others, although using the same signal block in electrical model simulationX.
- (11) SimulationX does not have other type of diode in electrical model  
 - They only have 2 types: ideal diode and semiconductor diode, therefore there is no diode C for this task.
- (12) The plot is different than others and compared to the plot in number (8), this plot is already stable from the beginning, much more like in the tesxt book theory than the result in-actual situation

Below is Table 6-6 described the list of the time needed to simulate the task for comparison 20.

Table 6-6 The list of the time needed to simulate the task for comparison 20

| Task | Matlab   | Simulink |          | Dymola      |          |          |          |
|------|----------|----------|----------|-------------|----------|----------|----------|
|      | T        | M        | S        | M           | S        | E        | T        |
| A-sa | 29,6353  | 1        | <b>X</b> | 0,046       | <b>X</b> | 0,031    | 0,031    |
| sb   | 1,1946   | 0,5      | <b>X</b> | 0,031       | <b>X</b> | 0,031    | 0,031    |
| sc   | 1,4376   | 19,5     | <b>X</b> | 0,234       | 0,234    | 0,188    | 0,203    |
| B    | 50,5118  | 5,5      | 1,5      | 0,203       | 0,235    | 0,703    | 0,219    |
| C-da | 0,8612   | 153      | <b>X</b> | 2,66        | <b>X</b> | <b>X</b> | 2,5      |
| db   | 1,0165   | 183      | <b>X</b> | 5,42        | 5,08     | 4,08     | 3,77     |
| dc   | 0,9585   | 213      | <b>X</b> | 6,14        | 6,11     | 4,83     | 4,39     |
| D    | 0,8593   | 255      | <b>X</b> | 0,141       | <b>X</b> | 0,094    | 0,125    |
| cond | 0,19     | <b>X</b> | <b>X</b> | <b>X</b>    | <b>X</b> | <b>X</b> | <b>X</b> |
| Task | Mosilab  |          |          | SimulationX |          |          |          |
|      | T        |          | S        | M           |          | E        |          |
| A-sa | 1        |          | <b>X</b> | 0,7020      |          | 0,6697   |          |
| sb   | 0,2      |          | <b>X</b> | 0,1118      |          | 0,0844   |          |
| sc   | 13,4     |          | 14,1     | 9,1124      |          | 16,2727  |          |
| B    | 9,1      |          | 8,8      | 7,7191      |          | 9,3952   |          |
| C-da | 47,9     |          | <b>X</b> | 8,9803      |          | <b>X</b> |          |
| db   | 100,9    |          | 105,2    | 10,0866     |          | 8,6296   |          |
| dc   | 110,7    |          | 116,8    | 1307,6718   |          | <b>X</b> |          |
| D    | 9,7      |          | <b>X</b> | 560,5021    |          | 0,3801   |          |
| cond | <b>X</b> |          | <b>X</b> | <b>X</b>    |          | <b>X</b> |          |

Note: - All value is in second (s)

- The task that don't have time result (X), refer to table 6-5.

## 6.2 Advantage and disadvantage

### 6.2.1 Matlab

#### **Advantage:**

- User friendly
- Powerful package
- Can do all calculation of matrix, such as eigenvalue, norm, condition, etc..
- Can do all the task that given
- Very complete documentation inside their help sections
- When error occurred, the error message is very detail that user can immediately knew what the problem is.
- Provide workspace, that user can overlook the data from simulation
- Provide many free literatures in internet.
- Provide step by step tutorial, that a new user can understand the algorithm code immediately.
- Provide internet newscenter which function as a forum for user so they can correspondence with other matlab user around the world to seek answers should they run into problems while using the software.

#### **Disadvantage:**

- cost expensive
- Have only 1 type modelling: textual mode.
- One of the solvers is not working very well, ode15i is the only solver that can solve differential equation in implicit form.
- Take up so many memory from the compute, around 125 Kb in stand by mode

### 6.2.2 Simulink

#### **Advantage:**

- User friendly
- Powerful package
- Very complete documentation inside their help sections
- When error occurred, the error message is very detail that user can immediately knew what the problem is.
- Have 2 type of modelling: hybrid and stateflow
- Provide workspace, that user can overlook the data from simulation
- Provide many free literatures in internet.
- Provide step by step tutorial, that a new user can understand the algorithm code immediately.
- Provide internet newscenter which function as a forum for user so they can correspondence with other simulink user around the world to seek answers should they run into problems while using the software.

#### **Disadvantage:**

- cost expensive
- Can't do all calculation of matrix, such as eigenvalue, norm, condition, etc..
- Give restriction in simulink mode, can't do the task if there is any signal source inside the subsystem block.
- Give restriction in stateflow mode, can't do the task if there is any loop inside the stateflow block such as algebraic loop.
- Take up so many memory from the computer around 125 Kb in stand by mode

### 6.2.3 Dymola

#### **Advantage:**

- User friendly
- Powerful package
- Very fast simulation



- Can do some calculation of matrix, such as eigenvalue, norm, inverse etc..
- When error occurred, the error message is very detail that user can immediately knew what the problem is.
- Very complete description for their model in the dymola library
- Doesn't have algebraic loop restriction in their stategraph mode.
- Have 4 type of modelling: hybrid, electrical, stategraph and textual mode.
- Have many variation of electrical model in their library.
- Provide free literatures in internet.
- Provide tutorial, that a new user can understand the algorithm of code.
- Provide dymola forum that if user got question about their problem, can send a question to this place and get the answer from user of dymola from TU Kaiserslautern.
- Provide demo version in their website

**Disadvantage:**

- Have problem to inverse stiff matrix.
- The documentation inside their help sections is not complete
- Do not provide workspace
- Take up average memory from the computer around 65 Kb in stand by mode
- New user have difficulty when encounters textual mode in dymola, because user must understand first modelica language.

#### 6.2.4 Mosilab

**Advantage:**

- User friendly
- When error occurred, the error message is very detail that user can immediately knew what the problem is.
- Doesn't have algebraic loop restriction in their statechart mode.
- Have 2 type of modelling: textual and statechart.

- Provide getting started and tutorial documentation, which enable new user to understand how to use mosilab.
- Provide free software from their website.

**Disadvantage:**

- The package is not so powerful
- Can't calculate all matrix calculation.
- Do not provide documentation inside their help sections
- Do not provide workspace
- Can only works in operating linux system
- Before software installation, user must provide first a few small programs that support mosilab.
- New user will have difficulty in software installation. User must be an expert of linux first.
- New user will have difficulty when they encounter textual mode and statechart in mosilab, because user must understand first modelica language.

### 6.2.5 SimulationX

**Advantage:**

- User friendly
- Powerful package
- When error occurred, the error message is so detail that user can immediately knew what the problem is.
- Complete description for their model in the simulationX library
- Have 2 types of modelling: hybrid and electrical.
- Have variation of electrical model in their library.
- Provide free literatures in internet.
- Provide tutorial-which enable new user to understand the algorithm-code.
- Provide ITI helpdesk that if user got question about their problem, can send a question to this place and get the answer from ITI expert.

- Provide free student version in their website.
- Provide type designer block, that user can define their code for new element type.
- Provide list data in the form txt file, user can overlook the data from simulation

**Disadvantage:**

- Can't calculate all matrix calculation, only eigenvalues and eigenvector.
- Complicated view of connector between block. So many tangled and tousled between one connector with other connector.
- Not so complete documentation inside their help sections
- Take up a lot of time to execute the software
- Take up many memory from the computer around 85 Kb in stand by mode
- New user will have difficulty when they encounters type designer block, because user must understand first modelica language.

## 7. Conclusion and Suggestion

### 7.1 Conclusion

Base on result, advantage and disadvantage above, the conclusion can be made as follow:

2. Matlab is the only simulation software that can do all calculation of matrix.
3. Dymola is the fastest simulation software.
4. Dymola have the most variation type of modelling needed in this thesis.
5. Despite Matlab can do all task that was given, they have quite few not satisfied result in comparison 20. This was because of using solver ode15i. The only solver that can solve the equation in implicit form,
6. The longest time simulation is by simulationX to simulate task diode C in comparison 20 = 1307,6718s.
7. The fastest time simulation is by dymola electrical model to simulate task d in comparison 3 = 0,015s.
8. Mosilab in comparison 3 can only do task B. The other tasks are impossible, because mosilab didn't have any function to calculate eigenvalue and to plot y function x ( $y(x)$ ).
9. To measure the time simulation in matlab by using matlab built function tic toc.
10. Measuring the time in simulink and stateflow, by looking computer clock.
11. Dymola, mosilab and simulationX provide time simulation in their software. Dymola in their message window, mosilab in their simulation process window and simulationX in their output bar window.
12. Only matlab that have not satisfied result when encounter equation in implicit form.

13. The different plot between equation model and electrical model in task D comparison 20 because of in equation model is only the approximation equation for task D.
14. The different plot between electrical model by dymola and simulationX in task D comparison 20 because of different algorithm and equation of ideal diode, between the two simulation software.
15. The simulation in dymola for task diode C electrical model heating diode can only be done by dymola, because inside the equation of this diode, they have 2 special dymola built in function, exlin and pow, which only available in dymola.
16. Stateflow, stategraph and statechart can only model the equation that has switching state.
17. Stateflow can only model 1 task in comparison 20, because they have restriction which does not allow any algebraic loop as input or event in their stateflow block
18. The stiffer the equation is, the longer time required to simulate the equation.
19. Ranking time simulation from fastest to slowest between type of modelling in dymola are:
  - a. Textual mode
  - b. Electrical model
  - c. Hybrid model
  - d. Stategraph model
20. As written above, this ranking is also valid for other simulation software.
21. Matlab, Simulink and Dymola have limitation for their step size time in simulation. Matlab limit is  $2.22045e-014$ , simulink limit is  $2.842170943040401e-014$  and dymola limit is  $5.939787e-013$ . Smaller than that matlab and simulink will automatically set to this limit, but dymola will give failed and pop up an error message.

22. Mosilab and simulationX don't have step size limitation in their core system.
23. Modelling the system in modelica code is the most efficient and easiest way to model the system based on equation.
24. Only simulink have real time simulation by setting the stop time to infinite. The other simulation software didn't have this feature.
25. Type designer block in simulationX is very useful feature for expert user to define their code in new element type.
26. Only simulink that have if, switch case, while loop and for loop block in their library.
27. Stateflow, stategraph and statechart are based on Petri nets theory and very useful as controller in the system.
28. Extracting data in task b and c in comparison 5 from plot result in Dymola, Mosilab and simulationX, by pointing mouse cursor in the plot result.
29. Matlab and simulink have workspace to extract data from simulation.
30. Only simulationX that automatically calculate eigenvalue from every simulation.
31. User must be a linux expert first to install mosilab in their computer.

## 7.1 Suggestion

Based on result, advantage, disadvantage and conclusion above, the suggestion can be made as follow:

1. Installation of mosilab can be made easier, like in windows, with step-by-step installation wizard.
2. User shouldn't need to provide initial small programs to support mosilab, before installation. Mosilab should have provides them in the first place for the user.
3. Matlab should add anoother solver that can solve problem in implicit form, not only ode15i, maybe in the future; there will be ode23i, ode45i, etc.

4. Dymola, mosilab, simulationX should add feature like workspace in matlab/simulink, it will make easier for user to see the data result from their simulation.
5. Dymola and matlab/simulink should reduce their limitation in step size time.
6. Simulink should delete algebraic loop restriction in their stateflow mode.
7. Dymola, mosilab, simulationX should add complete features of all matrix calculation.
8. Mosilab should add  $y(x)$  plot in their core system.
9. SimulationX should add more models in their library to reduce user using type designer block.
10. Dymola, mosilab and simulation should add if, for loop, while loop, switch case block in their library.
11. Matlab and simulink should add feature for automatic time simulation measurement.
12. Dymola, mosilab and simulationX should add real time simulation feature (setting stop time to infinite).
13. Simulink should add electrical model in their library.
14. SimulationX should add statechart and digital model in their library.
15. For future reference, other comparison can be made with different simulation software such as vensim, jsim, anylogic, java, C/C++ etc...

## REFERENCE

- [1] <http://en.wikipedia.org/wiki/Simulation>
- [2] [http://en.wikipedia.org/wiki/Computer\\_simulation](http://en.wikipedia.org/wiki/Computer_simulation)
- [3] <http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html>
- [4] Dabney, James B. : Mastering Simulink / James B. Dabney ; Thomas L. Harman. - Upper Saddle River, NJ : Prentice Hall, 2004. - XIX, 376 S. . - ISBN 0-13-142477-7
- [5] Fritzson, Peter : Principles of object-oriented modeling and simulation with Modelica 2.1 / Peter Fritzson. - New York, NY : Wiley, 2004. - XLII, 897 S. . - ISBN 0-471-47163-1
- [6] Tiller, Michael : Introduction to physical modeling with Modelica / Michael Tiller. - Boston, Mass. [u.a.] : Kluwer Academic Publ., 2001. - XXII, 344 S. . - (Kluwer international series in engineering and computer science ; 615 ). - ISBN 0-7923-7367-7
- [7] <http://www.mosilab.de/>
- [8] [http://www.itl.de/news/topics\\_e.htm](http://www.itl.de/news/topics_e.htm)
- [9] Nathan O. Sokal and Alan D. Sokal, Class E - A New Class of High-Efficiency Tuned Single-Ended Switching Power Amplifiers, IEEE Journal of Solid-State Circuits, Vol. SC-10, No. 3, June 1975, pp. 168-176.
- [10] Julio C. Mandojana, Kelly J. Herman and Robert E. Zulinski, A Discrete/Continuous Time-Domain Analysis of a Generalized Class E Amplifier, IEEE Transactions on Circuits and Systems, Vol. 37, No. 8, August 1990, pp. 1057-1060
- [11] <http://www.argesim.org/comparisons/index.html>
- [12] <file:///C:/Programme/Dymola/Modelica/Library/Modelica%202.2.1/help/>
- [13] <http://en.wikipedia.org/wiki/Diode>
- [14] [http://de.wikipedia.org/wiki/Kondition\\_%28Mathematik%29](http://de.wikipedia.org/wiki/Kondition_%28Mathematik%29)



Hanselman, Duane and Littlefield, Bruce: Matlab Bahasa Komputasi Teknis. Alih Bahasa: Jozep Edyanto, Yogyakarta: Andi, 2002. ISBN: 979-533-753-X.

Hanselman, Duane C. : Mastering MATLAB 7 / Duane Hanselman ; Bruce Littlefield. - Internat. ed. . - Upper Saddle River, NJ : Pearson Prentice Hall, 2005. - XI, 852 S. . - ISBN 0-13-185714-2

[http://apcmag.com/how\\_to\\_dual\\_boot\\_windows\\_xp\\_and\\_linux\\_xp\\_installed\\_first.htm](http://apcmag.com/how_to_dual_boot_windows_xp_and_linux_xp_installed_first.htm)

<http://www.dynasim.se/index.htm>

[http://en.wikipedia.org/wiki/Petri\\_net](http://en.wikipedia.org/wiki/Petri_net)

[http://en.wikipedia.org/wiki/State\\_diagram](http://en.wikipedia.org/wiki/State_diagram)

<http://www.facstaff.bucknell.edu/mastascu/eLessonsHtml/Diodes/Diode1.html>

<http://www.mathworks.com/matlabcentral/newsreader/>

<http://www.modelica-forum.com/forums/index.php?showforum=4>

<http://www.modelica.org/>

[http://en.wikipedia.org/wiki/Matrix\\_norm](http://en.wikipedia.org/wiki/Matrix_norm)

[http://en.wikipedia.org/wiki/Inverse\\_matrix](http://en.wikipedia.org/wiki/Inverse_matrix)

<http://en.wikipedia.org/wiki/Eigenvalue>

<http://www.psychocats.net/ubuntu/installingsoftware>

## APPENDIX

### SOURCE CODE

#### Comparison 3

##### Matlab

```
function Aout=A(t)
global VDC L1 C2 L3 C4 RL TRF
Aout=[0 -1/L1 0 0; 1/C2 -1/(C2*R(t)) -1/C2 0;
0 1/L3 -RL/L3 -1/L3; 0 0 1/C4 0];
End
```

```
function R_out = R(t)
global TRF
TRF=1e-15;
k=((5e+6)-(5e-2))/TRF;
t_red=mod(t, (10e-6));
if(0<=t_red)&(t_red<TRF)
 R_out=(5e-2)+k*t_red;
elseif(TRF<=t_red)&(t_red<(5e-6))
 R_out=5e+6;
elseif((5e-6)<=t_red)&(t_red<((5e-6)+TRF))
 R_out=(5e+6)-k*(t_red-(5e-6));
elseif((5e-6)+TRF<=t_red)&(t_red<(10e-6))
 R_out=5e-2;
else
 R_out=-5;
End
```

```
function dx=deq(t,x)
global VDC L1 C2 L3 C4 RL TRF
b=[VDC/L1; 0; 0; 0];
dx=(A(t)*x)+b;
end
```

```
tic
global VDC L1 C2 L3 C4 RL TRF
TRF= 1e-15; L1= 79.9e-6; VDC= 5; C2= 17.9e-9;
L3= 232e-6; C4= 9.66e-9; RL= 52.4;
R0ff= eig(A(TRF))
R0n= eig (A(0))
Toc
```

#### Dymola & Mosilab

##### Modelica Text

```
model C3Dymola_textv2
constant Real L1 = 79.9E-6; constant Real C2 = 17.9E-9;
constant Real L3 = 232.0E-6; constant Real C4 = 9.66E-9;
constant Real VDC = 5; constant Real RL = 52.4;
constant Real TRF = 1E-15;
Real x1; Real x2; Real x3; Real x4;
Real Rt; Real t_red; Real IRT; Real VRL; Real k;
equation
```

```

t_red = mod(time, 10E-6);
k=((5e+6)-(5e-2))/TRF;
algorithm
if
 (0<=t_red) and (t_red<TRF) then
 Rt:=(5e-2) + k*t_red;
elseif
 (TRF<=t_red) and (t_red<(5e-6)) then
 Rt:=5e+6;
elseif
 ((5e-6)<=t_red) and (t_red<((5e-6)+TRF)) then
 Rt:=(5e+6) - k*(t_red - (5e-6));
elseif
 ((5e-6)+TRF<=t_red) and (t_red<(10e-6)) then
 Rt:=5e-2;
else
 Rt:=-5;
end if;
equation
 L1*der(x1)= -x2 + VDC;
 C2*der(x2)= x1 - (x2/Rt) - x3;
 L3*der(x3)= x2 - (RL*x3) - x4;
 C4*der(x4)= x3;
IRT = x2/Rt; VRL = RL*x3;
end C3Dymola_textv2;

```

## MOSILAB STATECHART

```

model C3MosilabState
constant Real L1 = 79.9E-6; constant Real C2 = 17.9E-9;
constant Real L3 = 232.0E-6; constant Real C4 = 9.66E-9;
constant Real VDC = 5; constant Real RL = 52.4;
constant Real TRF = 1E-15;
event discrete Boolean s1(start=false), s2(start=false);
Real x1; Real x2; Real x3; Real x4;
Real Rt; Real t_red; Real IRT; Real VRL; Real k;
equation
t_red = mod(time, 10E-6);
k=((5e+6)-(5e-2))/TRF;
algorithm
if
 (0<=t_red) and (t_red<TRF) then
 Rt:=(5e-2) + k*t_red;
elseif
 (TRF<=t_red) and (t_red<(5e-6)) then
 Rt:=5e+6;
elseif
 ((5e-6)<=t_red) and (t_red<((5e-6)+TRF)) then
 Rt:=(5e+6) - k*(t_red - (5e-6));
elseif
 ((5e-6)+TRF<=t_red) and (t_red<(10e-6)) then
 Rt:=5e-2;
else
 Rt:=-5;
end if;

```

```

equation
s1 = if Rt>=5e+6 then true else false;
s2 = if Rt<=5e-2 then true else false;
L1*der(x1)=-x2 + VDC;
C2*der(x2)= x1 - (x2/Rt) - x3;
L3*der(x3)= x2 - (RL*x3) - x4;
C4*der(x4)= x3;
IRT = x2/Rt; VRL = RL*x3;
statechart
state C3MosilabStateSC extends State;
 annotation(extent=[-104,104; 44,-43]);
 State State1 annotation(extent=[-90,63; -77,59]);
 State State2 annotation(extent=[-51,62; -38,58]);
 State Initial (isInitial=true) annotation(extent=[-82,74; -80,72]);
 transition Initial->State1 action
 Rs:=5e+6;
 end transition annotation(points=[-82,72; -82,63]);
 transition State1->State2 event s2 action
 Rs:= 5e-2;
 end transition annotation(points=[-77,59; -51,59]);
 transition State2->State1 event s1 action
 Rs:= 5e+6;
 end transition annotation(points=[-51,60; -77,60]);
end C3MosilabStateSC;
end C3MosilabState;

```

## COMPARISON 5

### MATLAB

```

function [t,y]=C5
tic
global p d
tstart = 0; tfinal = 5;
y0= [4.2 0.3]; C = [2.7E+6 0.4 3.5651205 5.5];
p=5.8; d=1;
options = odeset('reltol',1e-11,'Events',@events);
tout = tstart; yout = y0;
teout = []; yeout = []; ieout = [];
while tout(length(tout))<5
% Call ODE Solver
FUN = @(t,y)F(t,y,C);
[t,y,te,ye,ie] = ode15s(FUN,[tstart tfinal],y0,options);
nt = length(t);
if y(nt)>=5.8
 p=2.5; d=-1; C = [2.7E+6 -0.3 3.5651205 2.73]; end
if y(nt)<=2.5
 p=5.8; d=1; C = [2.7E+6 0.4 3.5651205 5.5]; end
tout = [tout; t(2:nt)]; yout = [yout; y(2:nt,:)];
teout = [teout; te]; yeout = [yeout; ye];
ieout = [ieout; ie];
% Set the new initial conditions
y0=[y(nt,1) y(nt,2)];
tstart=t(nt);
options = odeset(options);
end
y1=yout(1:end,1);

```

```

plot(tout,y1);
A=teout
B=y1(length(y1))
toc
% -----
function dydt = F(t, y, C)
dydt(1,1) = C(1) * (y(2) + C(2) - y(1));
dydt(2,1) = C(3) * (C(4) - y(2));
% -----
function [value,isterminal,direction] = events(t,y)
global p d
value = y(1)- [p;0];
isterminal = [1;1];
direction = [d;1];

```

## DYMOLA & MOSILAB

### Modelica Text

```

model C5Dymola_text2
constant Real c1 = 2.7E+6; Real c2(start=0.4); constant Real c3 = 3.5651205; Real
c4(start=5.5);
Real y1(start=4.2); Real y2(start=0.3);
algorithm
 when (y1>=5.8) then
 c2:=-0.3; c4:=2.73;
 end when;
 when (y1<=2.5) then
 c2:=0.4; c4:=5.5;
 end when;
equation
 der(y1)= c1*(y2 + c2 - y1);
 der(y2)= c3*(c4 - y2);
end C5Dymola_text2;

```

### MOSILAB STATECHART

```

model C5MosilabState
constant Real c1 = 2.7E+6; Real c2(start=0.4); constant Real c3 = 3.5651205; Real
c4(start=5.5);
Real y1(start=4.2); Real y2(start=0.3);
event discrete Boolean s1(start=false), s2(start=false);
equation
s2 = if y1 >= 5.8 then true else false;
s1 = if y1 <= 2.5 then true else false;
der(y1)=c1*(y2+c2-y1);
der(y2)=c3*(c4-y2);
statechart
 state C5MosilabStateSC extends State;
 annotation(extent=[-104,105; 45,-43]);
 State State1 annotation(extent=[-90,63; -81,59]);
 State State2 annotation(extent=[-58,62; -45,58]);
 State Initial (isInitial=true) annotation(extent=[-82,74; -80,72]);
 transition Initial->State1
 end transition annotation(points=[-82,72; -82,63]);
 transition State1->State2 event s2 action

```

```

c2:= -0.3; c4:= 2.73;
end transition annotation(points=[-81,59; -77,60; -58,60]);
transition State2->State1 event s1 action
c2:= 0.4; c4:= 5.5;
end transition annotation(points=[-58,59; -77,59; -81,59]);
end C5MosilabStateSC;
end C3MosilabState;

```

## Comparison 20

### MATLAB

#### STEADY STATES

```

function Source = U(t)
Source= 14142.135623731 * sin((2*pi*50*t)+pi);
end

```

```

function dxdt= deqx(t,x)
global L C R1 R2
A=[-(R1+R2)/L -1/L; 1/C 0];
b=[U(t)/L; 0];
dxdt=(A*x)+b;
end

```

```

function dxdt= deqxB(t,x)
global L C R1 R2 Rmin Ron
A=[-((R1*R2)+((R1+R2)*(Ron+Rmin)))/(L*(R2+Ron+Rmin)) -
(Ron+Rmin)/(L*(R2+Ron+Rmin));
(Ron+Rmin)/(C*(R2+Ron+Rmin)) -1/(C*(R2+Ron+Rmin))];
b=[U(t)/L; 0];
dxdt=(A*x)+b;
end

```

```

tic
global L C R1 R2 %State A
L= 3.18e-3; C= 22.1e-9; R1= 0.1; R2= 5;
[tsol,xsol]=ode23s('deqx',[0 2e-1],[0;0]);
x1=xsol(1:end,1); x2=xsol(1:end,2);
plot(tsol,x1);
toc

```

```

tic
global L C R1 R2 Ron Rmin %StateB
L= 3.18e-3; C= 22.1e-9; R1= 0.1; R2= 5;
Ron=1e-4; Rmin=1e-4;
[tsol,xsol]=ode23s('deqxB',[0 2e-1],[0;0]);
x1=xsol(1:end,1); x2=xsol(1:end,2);
plot(tsol,x1);
toc

```

```

function C20StateC_exponent %StateC
tic
global L C R1 R2 d R
Ron = 1e-4; Rmin= 1e-4; L= 3.18e-3; C= 22.1e-9;
R1= 0.1; R2= 5; d= -1; R=1e-5;
x0= [0;0]; xp0=[0;0];

```

```

options = odeset('Events',@events,'RelTol', 1e-7, 'AbsTol', 1e-7);
tstart=0; tfinal=0.2; tout = 0; xout = x0';
teout = []; xeout = []; ieout = [];
FUN=@(t,x,xp)f1(t,x,xp);
while tout(length(tout))<0.2
% Call ODE Solver
[t,x,te,xe,ie]=ode15i(FUN,[tstart tfinal],x0,xp0,options);
nt = length(t);
tout = [tout; t(2:nt)];
xout = [xout; x(2:nt,:)];
dx1=xout(1:end,1); dx2=xout(1:end,2);
dx1dtall= diff(dx1); dx2dtall= diff(dx2);
nx= length(dx2);
dx2i=dx2dtall(nx-2); x2i= dx2(nx-1);
ctrl= ((R2*C*dx2i) + x2i);
if ctrl<0
 FUN=@(t,x,xp)f2(t,x,xp); d=-1 ;end
if ctrl>0
 FUN=@(t,x,xp)f1(t,x,xp); d=1 ;end
% Set the new initial conditions
x0=[x(nt,1); x(nt,2)];
xp0=[dx1dtall(nx-1); dx2dtall(nx-1)];
tstart=t(nt);
options = odeset(options);
end
x1=xout(1:end,1); x2=xout(1:end,2);
plot(tout,x1);
toc
% -----
function dxdt = f1(t,x,xp)
global L C R1 R2 R
% VD<0
dxdt = [- (U(t)/L)-((x(1)*R1)/L)-(((R2*C*xp(2))+x(2))/L)
 (x(1)/C)-(((R2*C*xp(2))+x(2))*R)/C)]
% -----
function dxdt = f2(t,x,xp)
global L C R1 R2 R
% VD>=0
dxdt = [- (U(t)/L)-((x(1)*R1)/L)-(((R2*C*xp(2))+x(2))*R)/L)
 (x(1)/C)-(((R2*C*xp(2))+x(2))+R)/C)]
% -----
function [value,isterminal,direction] = events(t,x,xp)
global R2 C d
value = ((R2*C*xp(2))+ x(2)) - [0;0];
isterminal = [1;1];
direction = [0;d];

```

## CLASSICAL SIMULATION

```

function T_out = T(t)
persistent TRF
TRF=5e-3;
k=((1e+8)-(1e-4))/TRF;
t_red=mod(t, (1e-1));
if(0<=t_red)&&(t_red<TRF)
 T_out=(1e-4)+k*t_red;

```

```

elseif(TRF<=t_red)&&(t_red<(5e-2))
 T_out=1e+8;
elseif((5e-2)<=t_red)&&(t_red<((5e-2)+TRF))
 T_out=(1e+8)-k*(t_red-(5e-2));
elseif((5e-2)+TRF<=t_red)&&(t_red<(1e-1))
 T_out=1e-4;
else
 T_out=-5;
end

function C20TaskBV2_withevents
tic
global L C R1 R2 Rmin Ron p
Ron = 1e-4; Rmin= 1e-4; L= 3.18e-3; C= 22.1e-9;
R1= 0.1; R2= 5; p=1e-4; x0= [0 0];
options = odeset('Events',@events);
tstart=0; tfinal=0.3; tout = 0; xout = x0;
FUN=@(t,x)deqtaskb1(t,x);
while tout(length(tout))<0.3
% Call ODE Solver
[t,x,te,xe,ie]=ode23s(FUN,[tstart tfinal],x0,options);
nt = length(t);
tout = [tout; t(2:nt)];
xout = [xout; x(2:nt,:)];
Tu= t(nt);
if T(Tu)<=1e-4
 p=1e+8; FUN=@(t,x)deqtaskb2(t,x); end
if T(Tu)>=1e+8
 p=1e-4; FUN=@(t,x)deqtaskb1(t,x); end
% Set the new initial conditions
x0=[x(nt,1) x(nt,2)];
tstart=t(nt);
options = odeset(options);
end
x1=xout(1:end,1); x2=xout(1:end,2);
plot(tout,x2);
toc
% -----
function dxdt1= deqtaskb1(t,x)
global L C R1 R2
dxdt1(1,1) = (-x(1)*(R1+R2)/L)-(x(2)/L)+(U(t)/L);
dxdt1(2,1) = (x(1)/C);
% -----
function dxdt2= deqtaskb2(t,x)
global L C R1 R2 Rmin Ron
dxdt2(1,1) = (-x(1)*((R1*R2)+((R1+R2)*(Ron+Rmin)))/(L*(R2+Ron+Rmin)))-
(x(2)*(Ron+Rmin)/(L*(R2+Ron+Rmin)))+(U(t)/L);
dxdt2(2,1) = (x(1)*(Ron+Rmin)/(C*(R2+Ron+Rmin)))-(x(2)/(C*(R2+Ron+Rmin)));
% -----
function [value,isterminal,direction] = events(t,x)
global p
value = T(t)- p + [0;0];
isterminal = [1;1];
direction = [0;0];

```



## DIFFERENT DIODE MODELS

```

function C20StateC %Diode A
tic
global L C R1 R2 Rmin Ron ids VT
Ron = 1e-4; Rmin= 1e-4; L= 3.18e-3; C= 22.1e-9;
R1= 0.1; R2= 5; ids= 1e-6; VT= 0.04; maxexp=15; R=1e+8;
[tsol,xsol]=ode15i(@f,[0 2e-1],[0;0],[0;0]);
x1=xsol(1:end,1); x2=xsol(1:end,2);
plot(tsol,x1);
toc
% -----
function dxdt = f(t,x,xp)
global L C R1 R2 Rmin Ron ids VT R
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-((R2*C*xp(2))/L)-(x(2)/L)
 (x(1)/C)-((ids*(exp(((R2*C*xp(2))-x(2))/VT)-1))/C)];

function C20StateC_exponent %Diode B
tic
global L C R1 R2 Rmin Ron ids VT maxexp R
Ron = 1e-4; Rmin= 1e-4; L= 3.18e-3; C= 22.1e-9;
R1= 0.1; R2= 5; ids= 1e-6; VT= 0.04; maxexp=15; R=1e+8;
x0= [0;0]; xp0=[0;0];
options = odeset('Events',@events,'Refine',100);
tstart=0; tfinal=0.2; tout = 0; xout = x0';
teout = []; xeout = []; ieout = [];
FUN=@(t,x,xp)f1(t,x,xp);
while tout(length(tout))<0.2
% Call ODE Solver
[t,x,te,xe,ie]=ode15i(FUN,[tstart tfinal],x0,xp0,options);
nt = length(t);
tout = [tout; t(2:nt)];
xout = [xout; x(2:nt,:)];
dx1=xout(1:end,1); dx2=xout(1:end,2);
dx1dtall= diff(dx1); dx2dtall= diff(dx2);
nx= length(dx2);
dx2i=dx2dtall(nx-2); x2i= dx2(nx-1);
ctrl= ((R2*C*dx2i) + x2i)/VT;
if ctrl<maxexp
 FUN=@(t,x,xp)f2(t,x,xp); end
if ctrl>maxexp
 FUN=@(t,x,xp)f1(t,x,xp); end
% Set the new initial conditions
x0=[x(nt,1); x(nt,2)];
xp0=[dx1dtall(nx-1); dx2dtall(nx-1)];
tstart=t(nt);
options = odeset(options);
end
x1=xout(1:end,1); x2=xout(1:end,2);
plot(tout,x1);
toc
% -----
function dxdt = f1(t,x,xp)
global L C R1 R2 Rmin Ron ids VT R
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-((R2*C*xp(2))/L)-(x(2)/L)

```

```

 (x(1)/C)-((ids*(exp(((R2*C*xp(2))-x(2))/VT)-1))/C)+(((R2*C*xp(2))-x(2))/(R*C)))]
% -----
function dxdt = f2(t,x,xp)
global L C R1 R2 Rmin Ron ids VT maxexp R
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-((R2*C*xp(2))/L)-(x(2)/L)
 (x(1)/C)-((ids*(exp(maxexp*(1+(((R2*C*xp(2)) + x(2)/VT))-maxexp))-1))/C)-
 (((R2*C*xp(2))-x(2))/(R*C))]
% -----
function [value,isterminal,direction] = events(t,x,xp)
global R2 C VT maxexp
value = (((R2*C*xp(2))+ x(2))/VT) - maxexp + [0;0];
isterminal = [1;1];
direction = [0;0];

function VTout = VT(t) %VT(t) for Diode C
VTout= ((30 * sin(2*pi*100*t))+310)*8.61734681e-5;
end

function C20StateC_exponent_temp %Diode C
tic
global L C R1 R2 Rmin Ron ids maxexp R
Ron = 1e-4; Rmin= 1e-4; L= 3.18e-3; C= 22.1e-9;
R1= 0.1; R2= 5; ids= 1e-6; VT= 0.04; maxexp=15; R=1e+8;
x0= [0;0]; xp0=[0;0];
options = odeset('Events',@events);
tstart=0; tfinal=0.2; tout = 0; xout = x0';
teout = []; xeout = []; ieout = [];
FUN=@(t,x,xp)f1(t,x,xp);
while tout(length(tout))<0.2
% Call ODE Solver
[t,x,te,xe,ie]=ode15i(FUN,[tstart tfinal],x0,xp0,options);
nt = length(t);
tout = [tout; t(2:nt)];
xout = [xout; x(2:nt,:)];
dx1=xout(1:end,1); dx2=xout(1:end,2);
dx1dtall= diff(dx1); dx2dtall= diff(dx2);
nx= length(dx2);
dx2i=dx2dtall(nx-2); x2i= dx2(nx-1);
ctrl= ((R2*C*dx2i) + x2i)/VT;
if ctrl<maxexp
 FUN=@(t,x,xp)f2(t,x,xp); end
if ctrl>maxexp
 FUN=@(t,x,xp)f1(t,x,xp); end
% Set the new initial conditions
x0=[x(nt,1); x(nt,2)];
xp0=[dx1dtall(nx-1); dx2dtall(nx-1)];
tstart=t(nt);
options = odeset(options);
end
x1=xout(1:end,1); x2=xout(1:end,2);
plot(tout,x1);
toc
% -----
function dxdt = f1(t,x,xp)
global L C R1 R2 Rmin Ron ids R
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-((R2*C*xp(2))/L)-(x(2)/L)

```

```

 (x(1)/C)-((ids*(exp(((R2*C*xp(2))-x(2))/VT(t))-1))/C)+(((R2*C*xp(2))-x(2))/(R*C)))]
% -----
function dxdt = f2(t,x,xp)
global L C R1 R2 Rmin Ron ids maxexp R
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-((R2*C*xp(2))/L)-(x(2)/L)
 (x(1)/C)-((ids*(exp(maxexp*(1+(((R2*C*xp(2)) + x(2)/VT(t)))-maxexp))-1))/C)-
 (((R2*C*xp(2))-x(2))/(R*C)))]
% -----
function [value,isterminal,direction] = events(t,x,xp)
global R2 C maxexp
value = (((R2*C*xp(2))+ x(2))/VT(t)) - maxexp + [0;0];
isterminal = [1;1];
direction = [0;0];

```

## INFLUENCE OF SIMULATION ALGORITHMS

```

function C20_subsystem3
tic
global L C R1 R2 Rmin Ron ids VT
Ron = 1e-4; Rmin= 1e-4; L= 3.18e-3; C= 22.1e-9;
R1= 0.1; R2= 5; ids= 1e-6; VT= 0.04;
[tsol,xsol]=ode15i(@f,[0 2e-1],[0;0],[0;0]);
x1=xsol(1:end,1);
x2=xsol(1:end,2);
plot(tsol,x2);
toc
% -----
function dxdt = f(t,x,xp)
global L C R1 R2 Rmin Ron ids VT
dxdt = [-(U(t)/L)-((x(1)*R1)/L)-((R2*C*xp(2))/L)-(x(2)/L)
 (x(1)/C)-((ids*(exp(((R2*C*xp(2))-x(2))/VT)-1))/C)-((R2*C*xp(2))/(C*(Ron+Rmin)))-
 (x(2)/(C*(Ron+Rmin)))]

```

## DYMOLA & MOSILAB

### Modelica Text

```

model C20DymolaStateA_text
constant Real R1 = 0.1; constant Real R2 = 5; constant Real Rmin = 1E-4;
constant Real L = 3.18E-3; constant Real C = 22.1E-9;
Real x1; Real x2; Real U;
equation
 U = 14142.135623731 * sin((2*3.14159*50*time) + 3.14159);
 L*der(x1)= -x1*(R1+R2) - x2 + U;
 C*der(x2)= x1;
end C20DymolaStateA_text;

model C20DymolaStateB_text
constant Real R1 = 0.1; constant Real R2 = 5; constant Real Rmin = 1E-4;
constant Real Ron = 1E-4; constant Real L = 3.18E-3; constant Real C = 22.1E-9;
Real x1; Real x2; Real U;
equation
 U = 14142.135623731 * sin((2*3.14159*50*time) + 3.14159);
 L*der(x1)= -x1*R1 -R2*C*der(x2) - x2 + U;
 C*(R2+Ron+Rmin)*der(x2)= x1*(Ron+Rmin) - x2;
end C20DymolaStateB_text;

```

```

model C20DymolaStateC_Ideal_text
constant Real R1 = 0.1; constant Real R2 = 5; constant Real Rmin = 1E-4;
constant Real Ron = 1E-4; constant Real L = 3.18E-3; constant Real C = 22.1E-9;
Real x1; Real x2; Real U; Real VC; Real IL; Real VD; Real VDr; Real ID;
equation
 U = 14142.135623731 * sin((2*3.14159*50*time) + 3.14159);
 VDr= (R2*C*der(x2)) + x2;
 if VDr< 0 then
 L*der(x1)= -x1*R1 - VDr - U;
 C*der(x2)= x1 - VDr*1e-5;
 else
 L*der(x1)= -x1*R1 - VDr*1e-5 - U;
 C*der(x2)= x1 - VDr;
 end if;
 VD= x2-x1; ID= x1; IL= -x1; VC= -x2 + x1;
end C20DymolaStateC_Ideal_text;

```

## CLASSICAL SIMULATION

```

model C20DymolaStateB_taskB_text
constant Real R1 = 0.1; constant Real R2 = 5; constant Real Rmin = 1E-4;
constant Real Ron = 1E-4; constant Real L = 3.18E-3; constant Real C = 22.1E-9;
constant Real TRF = 5e-3; Real x1; Real x2; Real U; Real t_red; Real k; Real Trap;
equation
 t_red = mod(time, 1E-1);
 k=((1e+8)-(1e-4))/TRF;
algorithm
 if
 (0<=t_red) and (t_red<TRF) then
 Trap:=(1e-4) + k*t_red;
 elseif
 (TRF<=t_red) and (t_red<(5e-2)) then
 Trap:=1e+8;
 elseif
 ((5e-2)<=t_red) and (t_red<((5e-2)+TRF)) then
 Trap:=(1e+8) - k*(t_red - (5e-2));
 elseif
 ((5e-2)+TRF<=t_red) and (t_red<(1e-1)) then
 Trap:=1e-4;
 else
 Trap:=-5;
 end if;
equation
 U = 14142.135623731 * sin((2*3.14159*50*time) + 3.14159);
 if Trap <= 1e-4 then
 L*der(x1)= -x1*R1 -R2*C*der(x2) - x2 + U;
 C*(R2+Ron+Rmin)*der(x2)= x1*(Ron+Rmin) - x2;
 else
 L*der(x1)= -x1*(R1+R2) - x2 + U;
 C*der(x2)= x1;
 end if;
end C20DymolaStateB_taskB_text;

```

## DIFFERENT DIODE MODELS

model C20DymolaStateC\_text\_simple

constant Real R1 = 0.1; constant Real R2 = 5; constant Real Rmin = 1E-4;  
constant Real Ron = 1E-4; constant Real L = 3.18E-3; constant Real C = 22.1E-9;  
constant Real ids = 1e-6; constant Real VT = 0.04;

Real x1; Real x2; Real U; Real VC; Real IL; Real VD; Real ID;

equation

U = 14142.135623731 \* sin( (2\*3.14159\*50\*time) + 3.14159);

L\*der(x1)= -x1\*R1 -R2\*C\*der(x2) - x2 - U;

C\*der(x2)= x1 - ids\*(exp(((R2\*C\*der(x2))+x2)/VT)-1);

VD= (R2\*C\*der(x2)) + x2;

ID= x1 - (C\*der(x2));

IL= -x1; VC= -x2;

end C20DymolaStateC\_text\_simple;

model C20DymolaStateC\_text\_simple\_exponentv2

constant Real R1 = 0.1; constant Real R2 = 5; constant Real Rmin = 1E-4;

constant Real Ron = 1E-4; constant Real L = 3.18E-3; constant Real C = 22.1E-9;

constant Real ids = 1e-6; constant Real VT = 0.04; constant Real maxexp = 15; constant Real R = 1e+8;

Real x1; Real x2; Real U; Real VC; Real IL; Real VD; Real ID; Real CTR;

equation

U = 14142.135623731 \* sin( (2\*3.14159\*50\*time) + 3.14159);

L\*der(x1)= -x1\*R1 -R2\*C\*der(x2) - x2 - U;

VD= (R2\*C\*der(x2)) + x2;

CTR = VD/VT;

if CTR > maxexp then

C\*der(x2)= x1 - ids\*(exp(maxexp\*(1+CTR-maxexp))-1) + (VD/R);

else

C\*der(x2)= x1 - ids\*(exp(CTR)-1) + (VD/R);

end if;

ID= x1 - (C\*der(x2)); IL= -x1; VC= -x2;

end C20DymolaStateC\_text\_simple\_exponentv2;

model C20DymolaStateC\_text\_simple\_exponentv2\_temp

constant Real R1 = 0.1; constant Real R2 = 5; constant Real Rmin = 1E-4;

constant Real Ron = 1E-4; constant Real L = 3.18E-3; constant Real C = 22.1E-9;

constant Real ids = 1e-6; constant Real maxexp = 15; constant Real R = 1e+8;

Real x1; Real x2; Real U; Real VC; Real IL; Real VD; Real ID; Real CTR; Real VT;

equation

U = 14142.135623731 \* sin( (2\*3.14159\*50\*time) + 3.14159);

L\*der(x1)= -x1\*R1 -R2\*C\*der(x2) - x2 - U;

VD= (R2\*C\*der(x2)) + x2;

VT= ((30 \* sin(2\*3.14159\*100\*time))+310)\*8.61734681e-5;

CTR = VD/VT;

if CTR > maxexp then

C\*der(x2)= x1 - ids\*(exp(maxexp\*(1+CTR-maxexp))-1) + (VD/R);

else

C\*der(x2)= x1 - ids\*(exp(CTR)-1) + (VD/R);

end if;

ID= x1 - (C\*der(x2)); IL= -x1; VC= -x2;

end C20DymolaStateC\_text\_simple\_exponentv2\_temp;

## INFLUENCE OF SIMULATION ALGORITHMS

```

model C20DymolaStateB_subsystem3_text
constant Real R1 = 0.1; constant Real R2 = 5; constant Real Rmin = 1E-4;
constant Real Ron = 1E-4; constant Real L = 3.18E-3; constant Real C = 22.1E-9;
constant Real ids = 1e-6; constant Real VT = 0.04;
Real x1; Real x2; Real U; Real VC; Real IL; Real VD; Real ID;
equation
 U = 14142.135623731 * sin((2*3.14159*50*time) + 3.14159);
 VD= (R2*C*der(x2)) + x2;
 L*der(x1)= -x1*R1 -VD - x2 - U;
 C*der(x2)= x1 - ids*(exp(VD/VT)-1) - (VD/(Ron+Rmin));
ID= x1 - (C*der(x2)); IL= -x1; VC= -x2;
end C20DymolaStateB_subsystem3_text;

```

## MOSILAB STATECHART

```

model C20MosilabSC_idealchart
constant Real R1 = 0.1; constant Real R2 = 5; constant Real Rmin = 1E-4;
constant Real Ron = 1E-4; constant Real L = 3.18E-3; constant Real C = 22.1E-9;
event discrete Boolean s1(start=false), s2(start=false);
Real x1; Real x2; Real U; Real VC; Real IL; Real VD; Real VDr; Real ID; Integer A;
equation
 U = 14142.135623731 * sin ((2*3.14159*50*time) + 3.14159);
 VDr= (R2*C*der(x2)) + x2;
 s1 = if VDr >= 0 then true else false;
 s2 = if VDr < 0 then true else false;
 if A<0 then
 L*der(x1)= -x1*R1 - VDr - U;
 C*der(x2)= x1 - VDr*1e-5;
 else
 L*der(x1)= -x1*R1 - VDr*1e-5 - U;
 C*der(x2)= x1 - VDr;
 end if;
 VD= x2-x1; ID= x1; IL= -x1; VC= -x2+x1;
statechart
 state C20MosilabSC_idealchartSC extends State;
 annotation(extent=[-103,103; 46,-46]);
 State State1 annotation(extent=[-92,60; -79,56]);
 State State2 annotation(extent=[-51,59; -38,55]);
 State Initial (isInitial=true) annotation(extent=[-85,71; -83,69]);
 transition Initial->State1 action
 A:= 1;
 end transition annotation(points=[-84,69; -84,60]);
 transition State1->State2 event s2 action
 A:= -1;
 end transition annotation(points=[-79,56; -51,56]);
 transition State2->State1 event s1 action
 A:= 1;
 end transition annotation(points=[-51,57; -79,57]);
 end C20MosilabSC_idealchartSC;
end C20MosilabSC_idealchart;

```

## CLASSICAL SIMULATION

```

model C20MosilabStateB_TaskB_chart

```

```

constant Real R1 = 0.1; constant Real R2 = 5; constant Real Rmin = 1E-4; constant Real Ron =
1E-4;
constant Real L = 3.18E-3; constant Real C = 22.1E-9; constant Real TRF = 5e-3;
event discrete Boolean s1(start=false), s2(start=false);
Real x1; Real x2; Real U; Real t_red; Real k; Real Trap; Integer A;
equation
t_red = mod(time, 1E-1);
k=((1e+8)-(1e-4))/TRF;
algorithm
if
 (0<=t_red) and (t_red<TRF) then
 Trap:=(1e-4) + k*t_red;
elseif
 (TRF<=t_red) and (t_red<(5e-2)) then
 Trap:=1e+8;
elseif
 ((5e-2)<=t_red) and (t_red<((5e-2)+TRF)) then
 Trap:=(1e+8) - k*(t_red - (5e-2));
elseif
 ((5e-2)+TRF<=t_red) and (t_red<(1e-1)) then
 Trap:=1e-4;
else
 Trap:=-5;
end if;
equation
U= 14142.135623731*sin((2*3.14159*50*time) + 3.14159);
s1 = if Trap>1e-4 then true else false;
s2 = if Trap<=1e-4 then true else false;
if A > 0 then
 L*der(x1)= -x1*R1 - R2*C*der(x2) - x2 + U;
 C*(R2+Ron+Rmin)*der(x2)= x1*(Ron+Rmin) - x2;
else
 L*der(x1)= -x1*(R1+R2) - x2 + U;
 C*der(x2)= x1;
end if;
statechart
state C20MosilabStateSC extends State;
 annotation(extent=[-104,104; 44,-43]);
 State State1 annotation(extent=[-90,63; -77,59]);
 State State2 annotation(extent=[-51,62; -38,58]);
 State Initial (isInitial=true) annotation(extent=[-82,74; -80,72]);
 transition Initial->State1 action
 A:=-1;
 end transition annotation(points=[-82,72; -82,63]);
 transition State1->State2 event s2 action
 A:=1;
 end transition annotation(points=[-77,59; -51,59]);
 transition State2->State1 event s1 action
 A:=-1;
 end transition annotation(points=[-51,60; -77,60]);
end C20MosilabStateSC;
end C20MosilabStateB_TaskB_chart;

```

## DIFFERENT DIODE MODELS

```

model C20MosilabStateC_simple_exponent_chart
constant Real R1= 0.1; constant Real R2= 5; constant Real Rmin= 1E-4; constant Real Ron= 1E-
4;
constant Real L= 3.18E-3; constant Real C= 22.1E-9; constant Real ids= 1e-6; constant Real
VT= 0.04;
constant Real maxexp= 15; constant Real R = 1e+8;
Real x1; Real x2; Real U; Real VC; Real IL; Real VD; Real ID; Real CTR; Integer A;
event discrete Boolean s1(start=false), s2(start=false);
equation
U= 14142.135623731*sin((2*3.14159*50*time) + 3.14159);
VD= (R2*C*der(x2)) + x2;
L*der(x1)= -x1*R1 - VD - x2 - U;
CTR= VD/VT;
s2= if CTR>maxexp then true else false;
s1= if CTR<=maxexp then true else false;
if A>0 then
C*der(x2)= x1 - ids*(exp(maxexp*(1+CTR-maxexp))-1) + (VD/R);
else
C*der(x2)= x1 - ids*(exp(CTR)-1) + (VD/R);
end if;
ID= x1 - (C*der(x2)); IL= -x1; VC= -x2;
statechart
state C20MosilabStateSC extends State;
annotation(extent=[-104,104; 44,-43]);
State State1 annotation(extent=[-90,63; -77,59]);
State State2 annotation(extent=[-51,62; -38,58]);
State Initial (isInitial=true) annotation(extent=[-82,74; -80,72]);
transition Initial->State1 action
A:=-1;
end transition annotation(points=[-82,72; -82,63]);
transition State1->State2 event s2 action
A:=1;
end transition annotation(points=[-77,59; -51,59]);
transition State2->State1 event s1 action
A:=-1;
end transition annotation(points=[-51,60; -77,60]);
end C20MosilabStateSC;
end C20MosilabStateC_simple_exponent_chart;

model C20MosilabStateC_simple_exponent_temp_chart
constant Real R1= 0.1; constant Real R2= 5; constant Real Rmin= 1E-4; constant Real Ron= 1E-
4;
constant Real L= 3.18E-3; constant Real C= 22.1E-9; constant Real ids= 1e-6;
constant Real maxexp= 15; constant Real R = 1e+8;
Real x1; Real x2; Real VT; Real U; Real VC; Real IL; Real VD; Real ID; Real CTR; Integer A;
event discrete Boolean s1(start=false), s2(start=false);
equation
U= 14142.135623731*sin((2*3.14159*50*time) + 3.14159);
VD= (R2*C*der(x2)) + x2;
L*der(x1)= -x1*R1 - VD - x2 - U;
VT= ((30 * sin(2*3.14159*100*time))+310)*8.61734681e-5;
CTR= VD/VT;
s2= if CTR>maxexp then true else false;

```



```

s1 = if CTR<=maxexp then true else false;
if A>0 then
C*der(x2)= x1 - ids*(exp(maxexp*(1+CTR-maxexp))-1) + (VD/R);
else
C*der(x2)= x1 - ids*(exp(CTR)-1) + (VD/R);
end if;
ID= x1 - (C*der(x2)); IL= -x1; VC= -x2;
statechart
 state C20MosilabStateSC extends State;
 annotation(extent=[-104,104; 44,-43]);
 State State1 annotation(extent=[-90,63; -77,59]);
 State State2 annotation(extent=[-51,62; -38,58]);
 State Initial (isInitial=true) annotation(extent=[-82,74; -80,72]);
 transition Initial->State1 action
 A:=-1;
 end transition annotation(points=[-82,72; -82,63]);
 transition State1->State2 event s2 action
 A:=1;
 end transition annotation(points=[-77,59; -51,59]);
 transition State2->State1 event s1 action
 A:=-1;
 end transition annotation(points=[-51,60; -77,60]);
 end C20MosilabStateSC;
end C20MosilabStateC_simple_exponent_temp_chart;

```