Die approbierte Originalversion dieser Dissertation ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

DISSERTATION

State Event Modelling and State Event Handling in System Simulation – Alternative Methods, Comparison and Benchmarks

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften
unter der Leitung von

Ao. Univ. Prof. Dipl.-Ing. Dr. techn. Felix Breitenecker
E101
Institute für Analysis und Scientific Computing

eingereicht an der Technischen Universität Wien
Fakultät für Mathematik und Geoinformation

von

Dipl.-Ing. Rouzbeh Karim
Matrikelnummer: 9427229
Währingerstrasse 202/4; 1180 Wien

Wien, am 04 Mai 2016

In memory of my father
George Karim.

Abstract

This thesis deals with analysis, comparison and some further developments of methods for state event determination in hybrid and variable-structure systems – from view of mathematics, and from view of case studies in mechatronics.
The thesis first reviews ‘classical’ methods for state event location, where a zero search for the event superimposes the ODE solver.

In the following the author concentrates on generic methods for state event location, which integrate the zero search into the ODE solver algorithm. Here, the approximation of the state vector at an event is based on the step-size calculation until the event, using a reformulation of the ODE solver. A ‘direct’ approach reformulates an explicit ODE solver by integration of the zero search for the step-size until the event, resulting in an ‘extended’ zero search, an implicit algorithm. The thesis therefore concentrates on an ‘implicit’ generic approach, which integrates the zero search onto an implicit ODE solver. This strategy modifies the zero search for the implicit solver algorithm appropriately by integration the zero search for the step-size until the event.

The theoretical part of the thesis continues with event location in DAE systems. For hybrid systems, described by semi-explicit DAEs, the author presents an extended strategy: the zero search for the step-size until the event is implemented into the multidimensional zero search for the algebraic states and for the system states. This method can also be used for fully-implicit systems, after index reduction of the system.

The last part of the theoretical part of the thesis analyses an alternative method for state event location, the Henon method. There, independent variable and one dependent variable are ‘exchanged’, so that no zero search for the event location is necessary, but the system becomes (much) more complicated.

The practical part of the theses analyses the compared and developed strategies for event location with three case studies from mechatronics: bouncing ball (hybrid ODE system), filament pendulum (DAE system with variable structure), and rotor – stator dynamics (hybrid DAE system).

Kurzfassung

Die Dissertation befasst sich mit Analyse, Vergleich und teilweiser Weiterentwicklung von Methoden zur Bestimmung von Zustandsvektoren bei Zustandsereignissen in einem hybriden System bzw. in einem System mit variabler Struktur – aus mathematisch-numerischer Sicht, und aus Sicht von mechatronischen Fallstudien.

Die Arbeit diskutiert zunächst einige „klassische“ Methoden zur Approximation des Zustandsvektors bei einem Ereignis: ein dem ODE/DAE – Solver übergelagertes Nullstellenverfahren versucht die Schrittweite bis zum Ereignis zu approximieren.

Diese klassischen Methoden, bei fast allen Systemsimulatoren im Einsatz, sind an sich nicht generisch und teilweise ineffizient. Daher entwickelt, analysiert und diskutiert der Autor „generische“ Methoden, die die Nullstellensuche zur Ereignisbestimmung mit dem ODE – Solver verbinden. Die Approximation eines Zustandsvektors bei einem Zustandsereignis basiert dabei auf der Berechnung der Schrittweite bis zum Ereignis aus einem reformulierten ODE Solver. Ein erster Ansatz ist eine Reformulierung eines expliziten Solvers durch Integration der Nullstellensuche, was zum einem „erweiterten“ Nullstellenproblem und damit zu einem impliziten Algorithmus führt.

In der Folge beschäftigt sich die Arbeit daher auch mit der Reformulierung eines impliziten Solvers durch Integration der Nullstellensuche. Diese Vorgangsweise behält die numerischen Vorteile des impliziten Solvers bei, und die Nullstellensuche des impliziten Solvers braucht nur „geeignet“ modifiziert zu werden. Die Arbeit entwickelt und vergleicht für diese Strategie verschiedene Vorgangsweisen für die Approximation von Schrittweite und Zustandsvektor.

Der theoretische Teil der Arbeit schließt dann die Betrachtung von DAE-Systemen mit Zustandsereignissen an. Für hybride Systeme, die von semi-expliziten DAEs beschrieben werden, kann die Strategie der Erweiterung eines impliziten ODE - Solvers erfolgreich fortgesetzt werden. Bei voll-impliziten DAE-Systemen kann nach Indexreduktion dieselbe Strategie verwendet werden.

Der theoretische Teil der Arbeit schließt mit der Analyse und Bewertung einer alternativen Methode ab. Diese nach dem Enwickler Henon - Methode genannte Vorgangsweise vertauscht die unabhängige mit der das Zustandsereignis bestimmenden abhängigen Variablen und benötigt damit kein Nullstellenverfahren zur Bestimmung des Zustandsvektors bei einem Ereignis, sie arbeitet aber mit einem üblicherweise (wesentlich) komplizierterem ODE-System..

Der praktische Teil der Arbeit untersucht die verglichenen und weiterentwickelten Methoden an drei mechtronischen Fallstudien: Bouncing Ball (hybride ODE-Beschreibung), Filament Pendulum (DAE-System mit variabler Struktur), und Rotor-Stator – Dynamik (hybride DAE-Beschreibung) und versucht einen Vergleich und eine Bewertung der Methoden aus Anwendungssicht.

Acknowledgments

Many thanks to my mother Ahue my sister Maryam and brother Behzad Karim who have motivated and supported me.
Many thanks to my supervisor Ao. Univ. Prof. Dipl.-Ing. Dr. techn. Felix Breitenecker for his mentoring, advice and management, consultancy and his frankness, Ao. Univ. Prof. Dipl.-Ing. Dr. techn. Horst Ecker for his supervision, Stefanie Winkler for the additional proofreading and recommendations as well as, Dipl.-Ing. Dipl.-Ing Dr.techn. BSc. Andreas Körner, Dipl.-Ing. BSc. Irene Hafner and colleagues.

Table of Contents

1 Introduction 8
  1.1 Hybrid and Variable Structure Systems 8
  1.2 Events in Hybrid Systems 16
  1.3 Approximation Procedure of Event Location in Hybrid System 19
2 State Event Handling in System of ODEs 21
  2.1 Event Location Approximation via Reformulation of Solver Formula 21
    2.1.1 Reformulation of Explicit Solver Formula 21
    2.1.2 Reformulation of Implicit Solver Formula 24
  2.2 Event Location Approximation via Reformulation of Solver Formula for Root-Finding Methods 26
    2.2.1 Reformulation of Explicit Solver Formula for Root-Finding Methods 26
    2.2.2 Reformulation of Implicit Solver Formula for Root-Finding Methods 27
3 State Event Handling in System of DAEs 31
  3.1 Index-Reduction 31
  3.2 Event Location Approximation in DAEs Transformed to System of ODEs 33
  3.3 Event Location Approximation in System of Semi-Explicit DAEs 34
    3.3.1 Event Location of Algebraic Variable 34
    3.3.2 Event Location of State Variable 37
4 Henon’s Method 40
  4.1 Henon’s Method in System of ODEs 40
  4.2 Henon’s Method in System of Semi-Explicit DAEs 44
5 Implementations of Event Location Approximation Methods in Hybrid Systems of ODEs and DAEs 49
  5.1 Bouncing Ball with Hybrid System of ODEs 49
    5.1.1 Model Description 49
    5.1.2 Reformulation of Explicit Euler’s Formula 52
    5.1.3 Reformulation of Implicit Trapezodial Formula for Newton’s Method 53
    5.1.4 Henon’s Method 53
    5.1.5 Simulation Comparisions 54
  5.2 Filament Pendulum with Variable Structure System of DAEs 58
    5.2.1 Model Description 58
    5.2.2 Reformulation of Explicit Euler’s Formula 66
    5.2.3 Reformulation of Implicit Trapezodial Formula for Newton’s Method 68
    5.2.4 Henon’s Method 69
    5.2.5 Simulation Comparisions 71
  5.3 Rotor and Stator with Hybrid System of DAEs 75
    5.3.1 Model Description 75
    5.3.2 Reformulation of Explicit Euler’s Formula 94
    5.3.3 Reformulation of Implicit Trapezodial Formula for Newton’s Method 97
    5.3.4 Henon’s Method 98
    5.3.5 Simulation Comparisions 102
6 Conclusion 108
7 Appendices 110
8 List of Figures 118
9 Nomenclature 119
10 References 125
11 Curriculum Vitae 136

1 Introduction

This chapter introduces a basic concept of variable structure systems, hybrid systems, state events and approximation procedure of an event location in hybrid or variable structure systems.

1.1 Hybrid and Variable Structure Systems

A variable structure system[1] contains of different sub systems with particular mapping, behaviours, characteristics and/or dimensions. The sub systems are activated or deactivated via time or state events. A variable structure system can be considered as a composite system comprising continuous-time combined with the discrete-time systems which are known as "hybrid systems". The switching system, which triggers the structure transitions, is considered as a discrete time system whereas the selectable systems are regarded as continuous time systems. A switching system can be defined by conditional expressions[2] demonstrated as discontinuity in system behaviour or it is established as an event indicator to control the system modification and system properties. Figure 1 shows a variable structure system with different selectable discrete and continuous time systems of ODEs and DAEs with event detection- and handling units and with its endogenous and exogenous components for connecting to other systems.

---

[1] The change in behavior of a function can be determined via change in function mapping $\mathrm{f} : D_{\mathrm{x}} \rightarrow D_{\mathrm{y}}$. The mapping of the various functions of systems of ODEs or DAEs may be defined by the alteration of function relationship, dimensions and spaces. These are defined as "variable structure system".

[2] Constraint, algebraic equations or inequalities can be specified by conditional expressions.

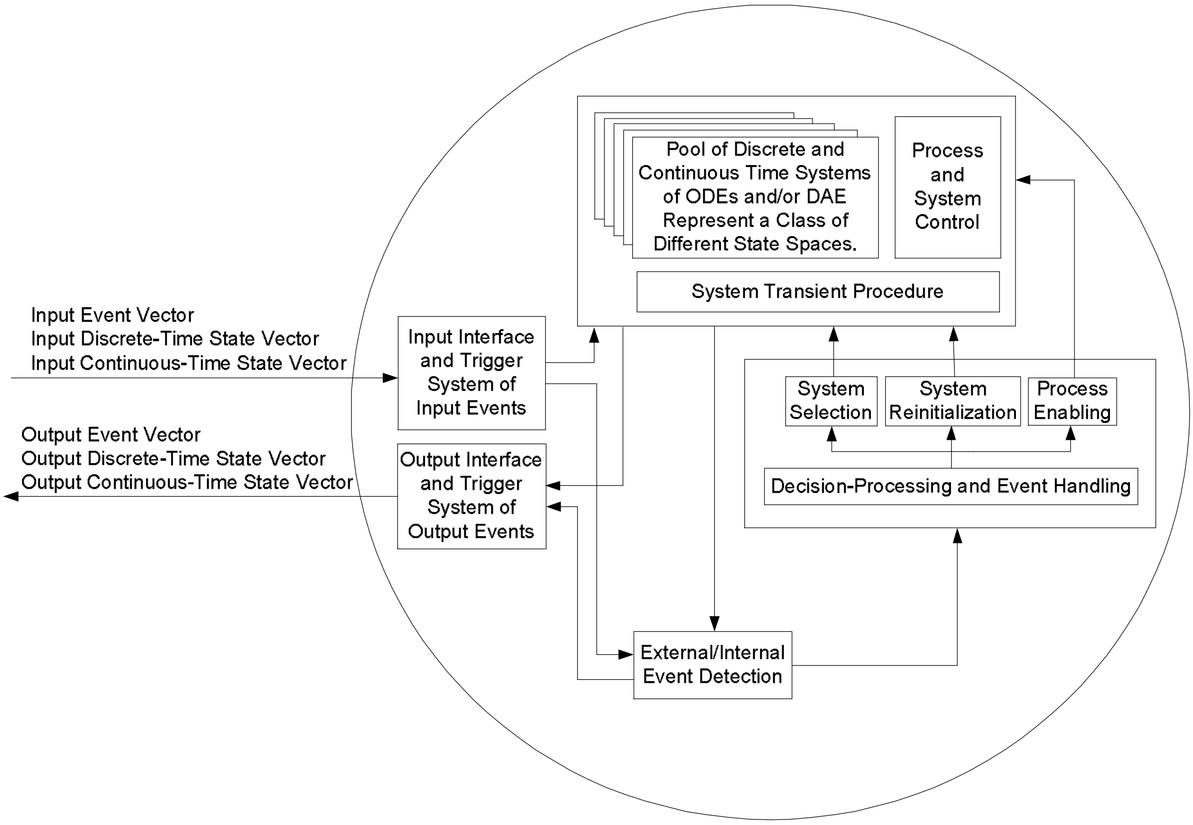
Input Event Vector
Input Discrete-Time State Vector
Input Continuous-Time State Vector
Output Event Vector
Output Discrete-Time State Vector
Output Continuous-Time State Vector
Input Interface and Trigger System of Input Events
Output Interface and Trigger System of Output Events
Pool of Discrete and Continuous Time Systems of ODEs and/or DAE Represent a Class of Different State Spaces.
Process and System Control
System Transient Procedure
System Selection
System Reinitialization
Process Enabling
Decision-Processing and Event Handling
External/Internal Event Detection

Figure 1: A Variable Structure System with Switching System and Interfaces

A variable structure system can be modelled by a hybrid automaton. A hybrid automaton [JEK02] [RAH01] H is a collection of different sets as follows

$$H = \left(L, D, E, X, Y, Z, X_0, F, P, Q, R, C, \Psi\right) \tag{1}$$

in which,

- L is a finite set of discrete states, modes, locations or nodes. A state is activated or deactivated according to the appropriate conditional expressions, which is shown on an arrow of the appropriate automaton diagram.
- D contains the domains of various state spaces on various discrete states or modes.
- E is a collection of arrows, whereas in a hybrid automata, the arrows can be considered as discrete transitions.
- X presents a finite set of continuous-time state variables.
- Y presents a finite set of algebraic variables.

- Z presents a finite set of discrete-time state variables.
- X_0 shows a set of initial states.
- F is a set of vector functions.
- P indicates a set of modification parameter vectors.
- R is a reset map. This is a map, which describes the actions at state transitions. These actions are the initialization of the state variables, activation of the other systems, changing the parameters etc.
- C is a set of conditions or guards. They can be shown on the edges between the different nodes of the automaton graph of the hybrid model.
- Q is defined as a set of switching variables. The validity of the conditional functions and equations is defined by the elements of the set C. They can be quantified e.g. by logical, integer or Boolean variables which can be considered as switching variables. The disabling or enabling of the state transitions can be defined by the switching variables Q or directly by guards C.
- Ψ is finite set of variable flags, logical variables or signals related to the parallel switching of hybrid systems. They can also be interpreted as the events of the synchronous or asynchronous transitions on a parallel hybrid automata.

In a network of the hybrid systems, a system may interact with other hybrid systems and vice versa. The interactions of sub-systems and components of different blocks require input and output connections or ports. The inputs of the system in Figure 1 are sets of the continuous-time variables U_x, discrete-time variables U_z and event variables U_q. These can be also collected in a hybrid automata as follows

$$H = (U_x, U_z, U_q, L, D, E, X, Y, Z, X_0, F, P, Q, R, C, \Psi). \tag{2}$$

The output variables can be affected the external systems. These variables are the sets of output continuous-, discrete time- and/or output event variables V_x, V_z and V_q. With these sets, the model collection can be expanded as follows:

$$H = (U_x, U_z, U_q, L, D, E, X, Y, Z, X_0, F, P, Q, R, C, \Psi, V_x, V_z, V_q). \tag{3}$$

The state graph is a graphical representation of an automaton describing variable structure systems. In the state graph, the arrows or edges can be denoted as events. In addition, certain graph outputs and other actions, such as triggers, initialization or resets, can be shown using arrows. Each system of ODEs or DAEs can be marked out with the corresponding node. A transition from one node to another can be interpreted as a sign of a system switching and change in system definition, properties, and structure.

The following forms show a system of DAEs of a variable structure system

$$\dot{\vec{x}}(t) = \vec{f}(t, \vec{x}(t), \vec{y}(t), \vec{u}(t), \vec{z}(t), \vec{p}(t)) \tag{4}$$

$$\vec{g}(t, \vec{x}(t), \vec{y}(t), \vec{u}(t), \vec{z}(t), \vec{p}(t)) = \vec{0} \tag{5}$$

where $\vec{x}$ represents a continuous time state vector and $\dot{\vec{x}}$ is its derivative, $\vec{y}$ is an algebraic variable vector, $\vec{u}$ is an input or control vector and t stands for the time[3]. $\vec{f}$ demonstrates a system vector function, $\vec{g}$ shows an algebraic vector equation and $\vec{z}$ is a discrete time state vector. Vector $\vec{p}$ indicates the system parameters. The event function is represented by h as follows:

$$h(\vec{x}(t), \vec{y}(t), \vec{u}(t), \vec{z}(t), \vec{p}(t)). \tag{6}$$

Figure 2 shows an example of a state graph with discrete states s_1 and s_2. The state transition can be controlled by an event function. The system transitions in Figure 2 can be described by a logical variable

$$q := \begin{cases} T & h(\vec{x}(t)) \geq 0 \\ F & h(\vec{x}(t)) < 0 \end{cases}$$

---

[3] The variable t is the independent variable. Hence the dependent variables are shown with the independent variable t. The structure of system can be changed and it depends on magnitudes of the outputs of the switching system in time t. The other interpretation of using index t in system indicates that the variables, systems or initial values may be valid in particular time slot and there is a mechanism, which causes a change in system, e.g. change in system structure, initial values and activating or deactivating systems etc.

and the discrete states can be described by the following logical expression:

$$\left(q\wedge\left(s_2\leftrightarrow\left(\dot{\vec{x}}(t)=\vec{f}_2(\vec{x}(t))\right)\right)\wedge\neg s_1\wedge\neg\left(\dot{\vec{x}}(t)=\vec{f}_1(\vec{x}(t))\right)\right)\vee\left(\neg q\wedge\left(s_1\leftrightarrow\left(\dot{\vec{x}}(t)=\vec{f}_1(\vec{x}(t))\right)\right)\wedge\neg s_2\wedge\neg\left(\dot{\vec{x}}(t)=\vec{f}_2(\vec{x}(t))\right)\right)$$

In Figure 2, the event indicator q is true then a transition to discrete state s_2 is done and if q is false a transition to s_1 occurs. The arrows serve the system transitions when the transition conditions are fulfilled. The transitions described by a reset map can be simplified and optimized by a transient truth table and Karnaugh map[4]. The event function in Figure 2 is defined as $\mathrm{h}: D \rightarrow \mathbf{R}$, $\vec{x} \in D=\{D_1 \cup D_2\} \subseteq \mathbf{R}^n$, $n \in \mathbf{N}$ and it is coupled with the systems $\dot{\vec{x}}(t)=\vec{f}_i(\vec{x}(t))$ with mapping $\vec{f}_i : D_i \rightarrow \mathbf{R}^n$, $i \in \{1,2\}$ and solution $\vec{x} : I_t \subseteq \mathrm{R}^+ \rightarrow D$, $t \in I_t$.

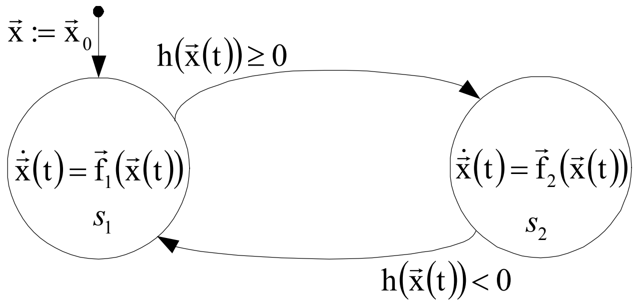
$\vec{x} := \vec{x}_0$
$h(\vec{x}(t)) \geq 0$
$\dot{\vec{x}}(t)=\vec{f}_1(\vec{x}(t))$
s_1
$\dot{\vec{x}}(t)=\vec{f}_2(\vec{x}(t))$
s_2
$h(\vec{x}(t)) < 0$

Figure 2: State Graph of Hybrid System

Figure 3 depicts an example of a hybrid automaton. This model can be considered as a hybrid system with finite or infinite system transition sequences. The system transition depends on the definitions of the conditional expression of the event function $h(\vec{x}(t),\vec{p})$ as well as definitions of state spaces, mapping of system functions and simulation time. In Figure 3, the system parameter vector $\vec{p} \in P = \{\vec{p}_0, \vec{p}_{12}, \vec{p}_{21}\}$ can be changed in each transition. The modifications of the system parameters in Figure 3 affect the structures of the differential equations and event function.

---

[4] A transient truth table gives a different point of view e.g. for building a state machine with logical structure transitions. A transient truth table contains all combinations of conditional expressions, input and output variables, including the variable values of the former step. Also, the labels of present and past nodes can be given in the tables.

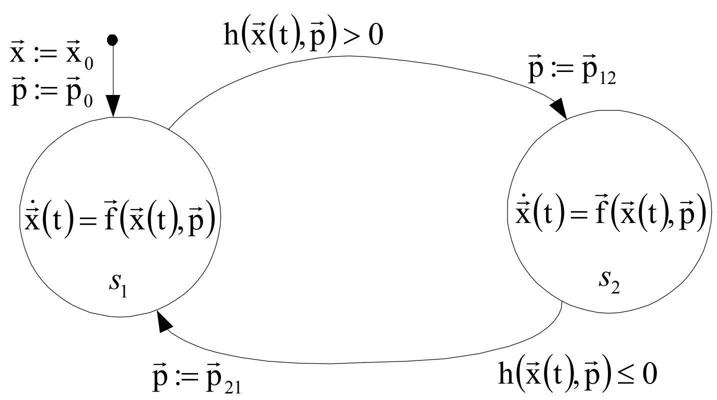
$\vec{x} := \vec{x}_0$
$\vec{p} := \vec{p}_0$
$h(\vec{x}(t), \vec{p}) > 0$
$\vec{p} := \vec{p}_{12}$
$\dot{\vec{x}}(t) = \vec{f}(\vec{x}(t), \vec{p})$
s_1
$\dot{\vec{x}}(t) = \vec{f}(\vec{x}(t), \vec{p})$
s_2
$\vec{p} := \vec{p}_{21}$
$h(\vec{x}(t), \vec{p}) \le 0$

Figure 3: Parameter Modifications in Hybrid System

In Figure 4 the algebraic vector equation[5] $\vec{g}_i(\vec{x}(t), \vec{y}(t))$ is coupled with differential equation $\dot{\vec{x}}(t) = \vec{f}_i(\vec{x}(t), \vec{y}(t))$ in which $\vec{f}_i : D_i \times D_{yi} \to \mathbf{R}^n$, $\vec{g}_i : D_i \times D_{y_i} \to \mathbf{R}^{n_y}$, $i \in \{1,2\}$. Here $\vec{x} \in D = \{D_1 \cup D_2\} \subseteq \mathbf{R}^n$ is defined as a state vector and $\vec{y} \in D_y = \{D_{y_1} \cup D_{y_2}\} \subseteq \mathbf{R}^{n_y}$ as a vector of algebraic variables with constant dimensions $n, n_y \in \mathbf{N}$ whereas the solutions are $\vec{x} : I_t \to D$ and $\vec{y} : I_t \to D_y$, $I_t \subseteq \mathbf{R}^+$. In this figure, the vector functions and the algebraic vector equations change in each transition. The event function $h(\vec{x}(t), \vec{y}(t))$ with mapping $h : D \times D_y \to \mathbf{R}$ is coupled with appropriate differential equations and is defined on each arrow by a conditional expression.

The event function $h(\vec{x}(t), \vec{y}(t))$ in Figure 4 can be expressed by a topological space $(\mathbf{R}, T_h)$ with topology

$$T_h = \{\mathbf{R}, \varnothing, \{h(\vec{x}(t), \vec{y}(t)) \mid h(\vec{x}(t), \vec{y}(t)) > 0\}, \{h(\vec{x}(t), \vec{y}(t)) \mid h(\vec{x}(t), \vec{y}(t)) \le 0\}\} \tag{7}$$

on $\mathbf{R}$. The zero-locus of an event function $h(\vec{x}(t), \vec{y}(t))$ in time domain is defined by a set D_e.

$$\exists (t_e, \vec{x}_e^T, \vec{y}_e^T)^T : \left((t_e, \vec{x}_e^T, \vec{y}_e^T)^T \in D_e := \left\{ (t, \vec{x}^T, \vec{y}^T)^T \mid h(\vec{x}(t), \vec{y}(t)) = 0 \right\} \right) \tag{8}$$

---

[5] It is assumed that the algebraic vector equation contains n_y components. Each component has many solutions. The graph of an equation can be shown by its solutions in $\mathbf{R}^n \times \mathbf{R}^{n_y}$ coordinate system. An algebraic equation is represented implicitly by mapping $D_i \times D_{y_i}$ into $\mathbf{R}$ and it may transform in explicit form.

An event vector $\left(t_e, \vec{x}_e^T, \vec{y}_e^T\right)^T$ shows a point in space $\mathbf{R}^+ \times \mathbf{R}^n \times \mathbf{R}^{n_y}$ and it is an element of zero-locus set D_e.

Zero location of an event function can be considered as an event. The event function at zero location can be seen as an equation called event equation. The root of an event equation $h(\vec{x}(t), \vec{y}(t)) = 0$ is demonstrated in vector form by

$$\left(\vec{x}_e^T, \vec{y}_e^T\right)^T := \left(\vec{x}^T(t_e), \vec{y}^T(t_e)\right)^T.$$

The vector $\left(\vec{x}_e^T, \vec{y}_e^T\right)^T$ which shows the magnitudes of dependent variables at an event is named transient vector[6].

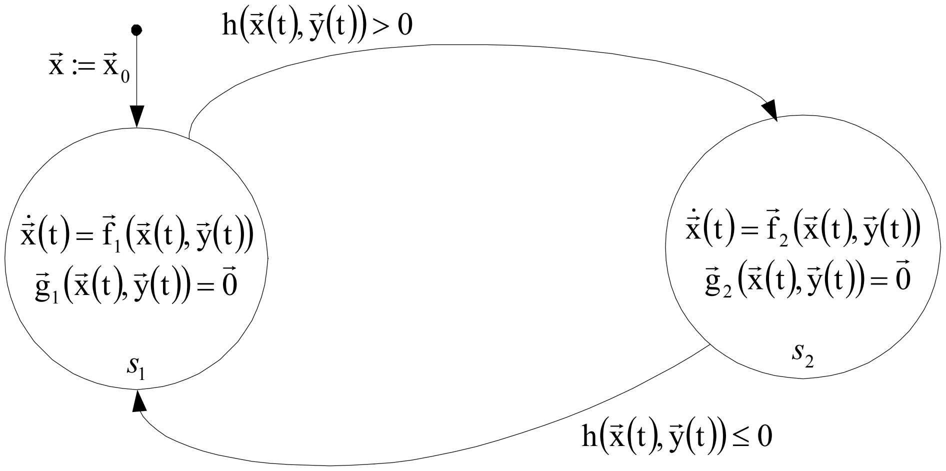
$h(\vec{x}(t), \vec{y}(t)) > 0$
$\vec{x} := \vec{x}_0$
$\dot{\vec{x}}(t) = \vec{f}_1(\vec{x}(t), \vec{y}(t))$
$\vec{g}_1(\vec{x}(t), \vec{y}(t)) = \vec{0}$
s_1
$\dot{\vec{x}}(t) = \vec{f}_2(\vec{x}(t), \vec{y}(t))$
$\vec{g}_2(\vec{x}(t), \vec{y}(t)) = \vec{0}$
s_2
$h(\vec{x}(t), \vec{y}(t)) \leq 0$

Figure 4: Modifications of DAEs in Hybrid System

Figure 5 is another illustration of a hybrid system. The state graph for this hybrid automaton is given for intervals $t \in I_t \subset \mathbf{R}^+$ for DAE solutions

$$\vec{x}_i : I_t \to D_i, \; \vec{y}_i : I_t \to D_{y_i}$$

---

[6] $\vec{x}_e$ is transient state vector and $\vec{y}_e$ is transient algebraic variable vector. A transient vector is considered as an initial vector for the initialization of the next system of ODEs or DAEs at the switching event in hybrid or variable structure systems.

with different dimensions $n_i, n_{y_i}, n_{p_i} \in \mathbf{N}$, different state vectors $\vec{x}_i \in D_i \subseteq \mathbf{R}^{n_i}$, different algebraic variable vectors $\vec{y}_i \in D_{y_i} \subseteq \mathbf{R}^{n_{y_i}}$ different parameter vectors $p_i \in D_{p_i} \subseteq \mathbf{R}^{n_{p_i}}$ different mapping of the system vector functions

$$\vec{x}_i(t) = \vec{f}_i(\vec{x}_i(t), \vec{y}_i(t), \vec{p}_i),\ \vec{f}_i : D_i \times D_{y_i} \times D_{p_i} \to \mathbf{R}^{n_i}$$

and algebraic vector equations

$$\vec{g}_i(\vec{x}_i(t), \vec{y}_i(t), \vec{p}_i) = \vec{0},\ \vec{g}_i : D_i \times D_{y_i} \times D_{p_i} \to \mathbf{R}^{n_{y_i}}$$

in each node as well as different event functions

$$h_i(\vec{x}_i(t), \vec{y}_i(t), \vec{p}_i),\ h_i : D_i \times D_{y_i} \times D_{p_i} \to \mathbf{R}$$

on each edge. For auxiliary index j with $\{i, j \in \{1,2\}, i \neq j\}$ there are as well different initialization state vectors

$$\vec{x}_i := \vec{f}_{ji}(\vec{x}_j(t), \vec{y}_j(t), \vec{p}_j)$$

and different initialization parameter vectors

$$\vec{p}_i := \vec{\theta}_{ji}(\vec{x}_j(t), \vec{y}_j(t)).$$

Each transition is accomplished at a different point of time and the hybrid system is initialized with a different initialization vector with different dimensions. Hence, this model is considered as a variable structure system with a total structure change in each transition. The magnitudes of the different state functions can vary and influence the system structure and therefore determine the dynamic of the system.

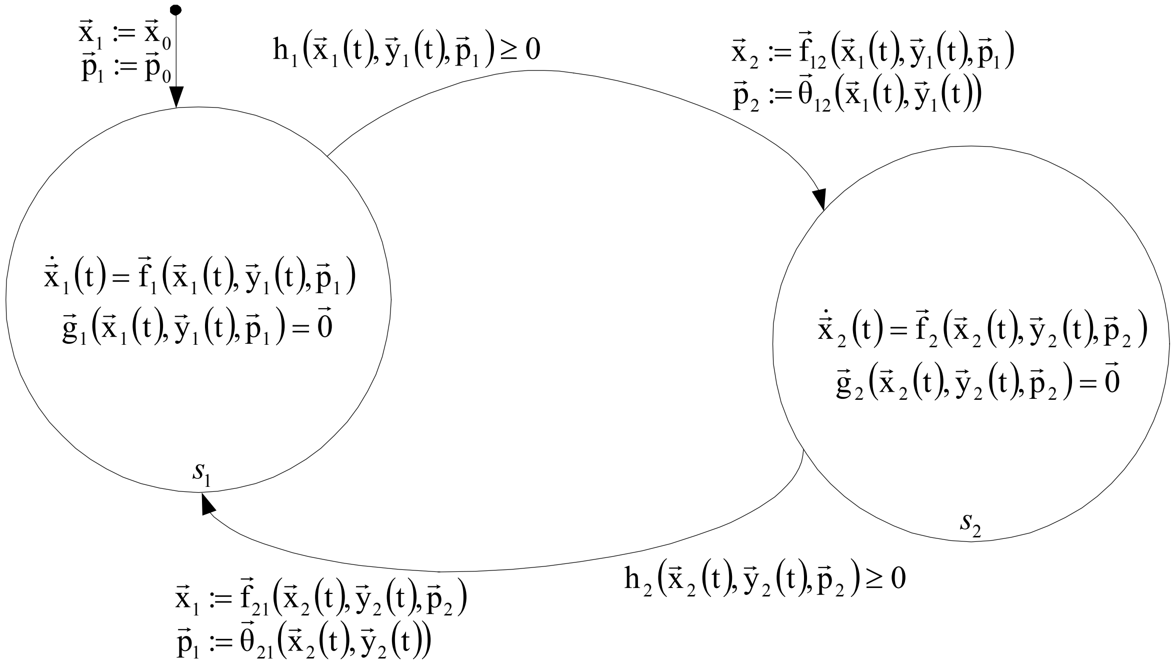
$\vec{x}_1 := \vec{x}_0$
$\vec{p}_1 := \vec{p}_0$
$h_1(\vec{x}_1(t), \vec{y}_1(t), \vec{p}_1) \geq 0$
$\vec{x}_2 := \vec{f}_{12}(\vec{x}_1(t), \vec{y}_1(t), \vec{p}_1)$
$\vec{p}_2 := \vec{\theta}_{12}(\vec{x}_1(t), \vec{y}_1(t))$
$\dot{\vec{x}}_1(t) = \vec{f}_1(\vec{x}_1(t), \vec{y}_1(t), \vec{p}_1)$
$\vec{g}_1(\vec{x}_1(t), \vec{y}_1(t), \vec{p}_1) = \vec{0}$
s_1
$\dot{\vec{x}}_2(t) = \vec{f}_2(\vec{x}_2(t), \vec{y}_2(t), \vec{p}_2)$
$\vec{g}_2(\vec{x}_2(t), \vec{y}_2(t), \vec{p}_2) = \vec{0}$
s_2
$h_2(\vec{x}_2(t), \vec{y}_2(t), \vec{p}_2) \geq 0$
$\vec{x}_1 := \vec{f}_{21}(\vec{x}_2(t), \vec{y}_2(t), \vec{p}_2)$
$\vec{p}_1 := \vec{\theta}_{21}(\vec{x}_2(t), \vec{y}_2(t))$

Figure 5: Total Structure Modification in Hybrid System

The designed state spaces of the hybrid system can run as sequential states or XOR decomposition. In this case, only one state space can be activated at any time. If the switching conditions of the hybrid system in a XOR mode are triggered to pass to several nodes at the same time, the hybrid system can lead into an undefined state. Then the system could not make a decision which transition structure should be used.

In some cases, hybrid automata may be treated as parallel or AND decomposition e.g. in a multithread system.

1.2 Events in Hybrid Systems

In a hybrid system, an event indicates a transition to a system with the various characteristics and behaviours as previous system. An event may be associated with change in mapping and structures of the system of vector function, equations, constraints, inputs, outputs, dimensions, variables, initial values and parameters.

An event can be classified as a state event or a time event. The events can be divided into two categories which are random and deterministic events[7]. An event can be modelled via system signals, magnitudes, specifications and characteristics which are associated with the event. For the definition of limitations of an event, conditional operators can be applied. In the variable structure system, an event expression can contain a time condition. Also a system

---

[7] The simulation models in chapter 5 indicate the deterministic state events.

switching by a conditional expression dependent on time can be defined. This dependency indicates a time event. An event condition with a time variable comparison points to a time event. For example in ODEs $\dot{x}(t) = f_i(x(t))$, $i \in \{0,1,\ldots,m\} \subset \mathbf{N}$ a time event in serial consecutive systems can be expressed by

$$\forall i > 1 \exists t \in \left]t_{e_i} - \varepsilon, t_{e_i} + \varepsilon\right[\subset I_t : \left(\left(t < t_{e_i}\right) \leftrightarrow \left(\dot{x}(t) = f_{i-1}(x(t))\right)\right) \vee \left(\left(t_{e_i} \le t\right) \leftrightarrow \left(\dot{x}(t) = f_i(x(t))\right)\right) \tag{9}$$

and in a parallel systems by

$$\forall i > 1 \exists t \in \left]t_{e_i} - \varepsilon, t_{e_i} + \varepsilon\right[\subset I_t : \left(\left(t < t_{e_i}\right) \rightarrow \left(\dot{x}(t) = f_{i-1}(x(t))\right) \vee \left(t_{e_i} \le t\right) \rightarrow \left(\dot{x}(t) = f_i(x(t))\right)\right)$$

in which the ODE solution $x : I_t \rightarrow D$ is defined on interval

$$I_t = \bigcup_{i=0}^{m} I_{t_i} \text{ with } I_{t_i} = \left[t_{e_i}, t_{e_{i+1}}\right[\subseteq \mathbf{R}^+$$

and the ODE mapping function is

$$f_i : D_i \rightarrow R_i ,\ x \in D = \bigcup_{i=0}^{m} D_i \subseteq \mathbf{R} ,\ R = \bigcup_{i=0}^{m} R_i \subseteq \mathbf{R} .$$

Hence, a time event happens if the time variable t crosses the time limit

$$t_{e_i} : \left(t_{e_i} \in I_t\right) \wedge \left(\forall i > 0 : t_{e_{i-1}} < t_{e_i}\right).$$

If a system switching is defined by an event expression dependent on state variables, then the switching occurrence can be interpreted as a state event. For specification of a state event, the appropriate function can be expressed with logical, relational, conditional and arithmetical operators (e.g. and, or, not, greater than, less than, equal, if, else) or combinations of these operators. The output of the switching system can be demonstrated by mapping the switching expression onto other auxiliary variable e.g. in form of a logical, integer or Boolean variable which is named here switching variable[8].

In a consecutive switching system with $\mathrm{m} \in \mathbf{N}$ switched ODEs

$$\dot{x}(t) = f_i(x(t)),\ x \in \bigcup_{i=1}^{m} D_i$$

a state event

$$x\left(t_{e_\ell}\right) = a_{e_\ell},\ \ell \in \{1,2,\ldots,m\}$$

is specified as intersection of threshold value

$$x = a_{e_\ell},\ a_{e_\ell} \in \left\{a_{e_1}, a_{e_2}, \ldots, a_{e_m}\right\} \subset D_i$$

with state variable $x(t)$ at $t = t_{e_\ell}$

$$t_{e_\ell} : \left(t_{e_\ell} \in \left\{t_{e_1}, t_{e_2}, \ldots, t_{e_m}\right\} \subset I_t\right) \wedge \left(t_{e_0} < t_{e_1} < t_{e_2} \cdots < t_{e_m}\right)$$

described by sum of the ℓ integrals.

$$x\left(t_{e_\ell}\right) = x_0 + \int_{t_{e0}}^{t_{e1}} f_1(x(t))dt + \int_{t_{e1}}^{t_{e2}} f_2(x(t))dt + \cdots + \int_{t_{e_{\ell-1}}}^{t_{e_\ell}} f_\ell(x(t))dt = x_0 + \sum_{i=1}^{\ell} \int_{t_{ei-1}}^{t_{ei}} f_i(x(t))dt$$

(10)

---

[8] An event function $h(\vec{x}(t))$ can be associated with an additional variable as a state event indicator. E.g. the logical expressions in page 12 quantify the state event occurrences using logical variable.

1.3 Approximation Procedure of Event Location in Hybrid System

In all the examples shown in Figures 2, 3, 4, 5, before a transition from discrete state s_1 to s_2 and vice versa, the transient state vector has to be computed. The transition vector is used to avoid imprecise interruption of current data processing at the moment of state transition.

Figure 6 shows a flowchart of a computer program for computing the state vector of a hybrid model using an ODE solver system with fixed step-size. The algorithm contains an approximation system of the transient state vector at the state event in one direction from the first system of ODEs to the second ODEs. In this flowchart, the procedures are shown in the rectangles and the decisions are shown in parallelograms. Top of the flowchart shows the first procedure for variable initializations. In each iteration, the first parallelogram checks, whether the event happens or not. At first, if the event is not captured, then the right branch of algorithm runs, thus the first solver algorithm runs. In each repetition loop, the iteration index and the simulation time increase. Then the simulation time is checked each time passing the last decision block whether it has reached its limit or not. If the simulation time is not finished then it proceeds one more time and the algorithm repeats the computation of the approximation of the state vector of the first system. Hence, this process continues until an event is detected by the first decision block. If an event is captured, then the last step-size before the state event is predicted in the middle block using an iterative algorithm. With this prediction, it is possible to approximate the transient state vector. Now, the ODE solver of the second model can be initialized with the approximated transient state vector. After this initialization, the algorithm runs approximations of the state vector of the next system using appropriate ODE solver. The flowchart shows a transition in one direction. Hence, the simulation of the second system continues until the simulation time is up.

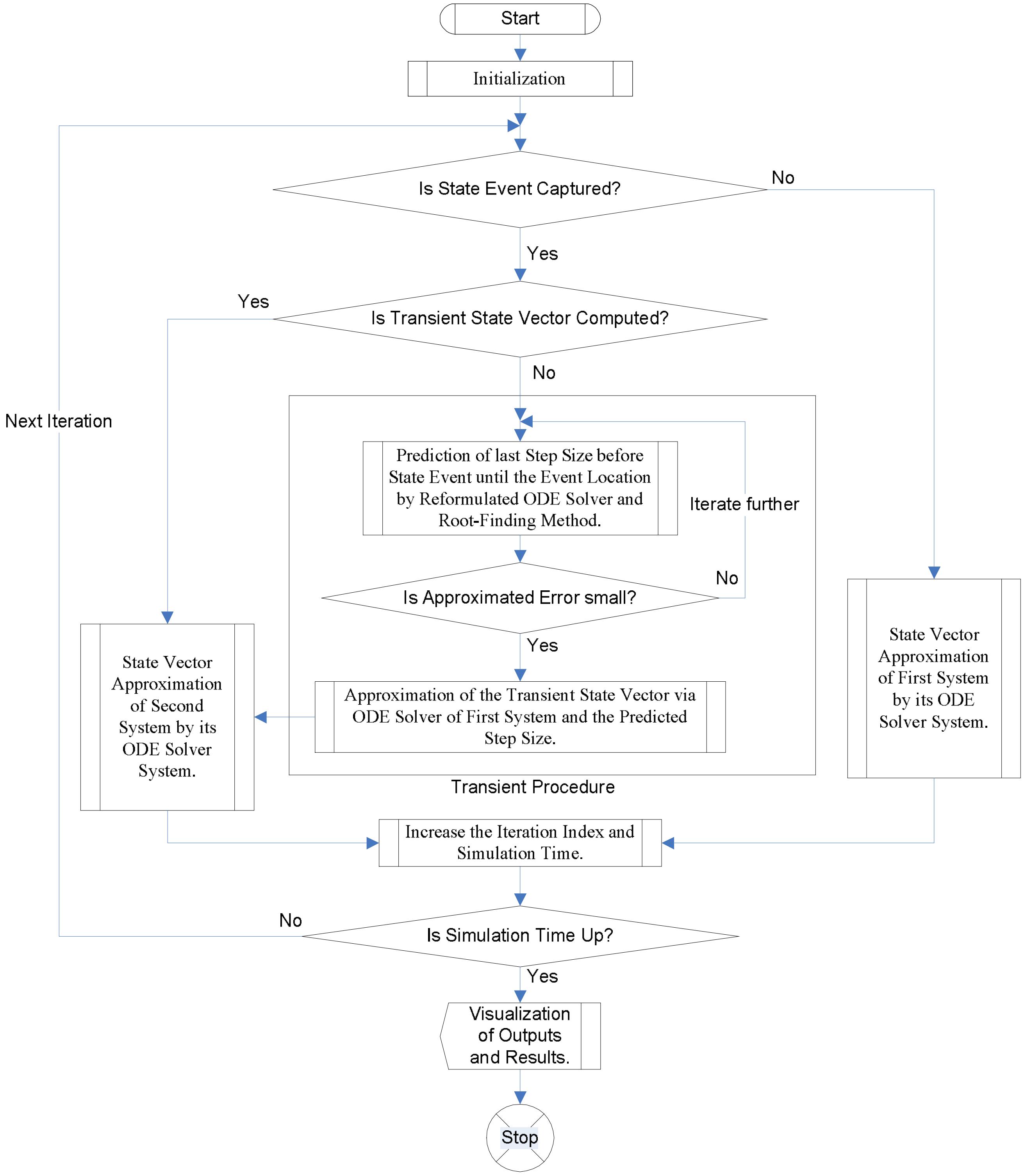
Start
Initialization
Is State Event Captured?
No
Yes
Yes
Is Transient State Vector Computed?
No
Next Iteration
Prediction of last Step Size before State Event until the Event Location by Reformulated ODE Solver and Root-Finding Method.
Iterate further
No
Is Approximated Error small?
Yes
State Vector Approximation of First System by its ODE Solver System.
State Vector Approximation of Second System by its ODE Solver System.
Approximation of the Transient State Vector via ODE Solver of First System and the Predicted Step Size.
Transient Procedure
Increase the Iteration Index and Simulation Time.
No
Is Simulation Time Up?
Yes
Visualization of Outputs and Results.
Stop

Figure 6: Approximation Procedure of Event Location in Hybrid System

2 State Event Handling in System of ODEs

This chapter presents the state event handling via solving reformulated ODE solver equation regarding the event step-size. First, the magnitude of the independent variable at the state event is computed. Then the transient initial conditions[9] can be approximated by recompilation of the computed magnitudes in the solver algorithm at the state event.

Subchapter 2.1 provides a solution by reformulation solver equations and 2.2 achieves a solution via applying the root-finding methods on nonlinear systems of equations to the system of reformulated solver formulas for approximation of transient initial conditions.

2.1 Event Location Approximation via Reformulation of Solver Formula

Subchapters 2.1.1 and 2.1.2 present the approximation methods of transient states using reformulated explicit and implicit solver formulas.

2.1.1 Reformulation of Explicit Solver Formula

The idea behind this method is to solve the reformulated explicit solver equation[10], which includes computing the step-size at the state event using the old approximated values and the computed step-size to approximate the transient state vector.

The system of ODEs is considered as a non-autonomous continuous system[11] $\dot{\vec{x}}(t)=\vec{f}(t,\vec{x}(t))$ with the state vector $\vec{x}\in D\subset\mathbf{R}^n$. The system of ODEs $\dot{\vec{x}}(t)=\vec{f}(t,\vec{x}(t))$ is defined by the function $\vec{f}:I_t\times D\rightarrow\mathbf{R}^n$ with solution $\vec{x}:I_t\rightarrow D$. There is an ODE component $\dot{x}_i(t)=f_i(t,\vec{x}(t))$ involved in state events in which $x_i\in U_i:(U_i\subset\mathbf{R})\subset D$, $i\in\{1,2,\ldots,n\}$, $n\in\mathbf{N}$ exists. The differential equation $\dot{x}_i(t)=f_i(t,\vec{x}(t))$ contains the mapping

---

[9] Transient magnitudes and/or transient state vector can be considered as transient initial conditions in a hybrid system. These magnitudes are computed after event detection and applied at the event location for initialization the next switched system.

[10] After determination of an event in hybrid or variable structure systems, the solver system switches to a system of "event location approximation" (Figure 6). The approximation system considers the reformulated solver formula as an equation and handles it as such equation. The equation is solved for the event step-size and then the computed event step-size is used to approximate transient state vector. Hence, the term "solver equation" is given for the approximation system of transient state vector at switching event.

[11] For simplification of system description, an input vector is not given in system definition. The methodology of state event handling with an input vector is the same, but the input variables and their magnitudes should be considered in der calculation.

$\mathrm{f}_{\mathrm{i}} : I_{\mathrm{t}} \times D \rightarrow \mathbf{R}$ with the differentiable solution $\mathrm{x}_{\mathrm{i}} : I_{\mathrm{t}} \rightarrow U_{\mathrm{i}}$ for the finite interval $I_{\mathrm{t}} = [\mathrm{t}_0, \mathrm{t}_\sigma] \subset \mathbf{R}^+$, $\mathrm{t} \in I_{\mathrm{t}}$.

The Euler's solver with step-size $\Delta \mathrm{t}$ for approximation of the state variable is shown in the following formula

$$\hat{\mathrm{x}}_{\mathrm{i,j}} = \hat{\mathrm{x}}_{\mathrm{i,j-1}} + \Delta \mathrm{t}\, \mathrm{f}_{\mathrm{i}}\left(\mathrm{t}_{\mathrm{j-1}}, \hat{\vec{\mathrm{x}}}_{\mathrm{j-1}}\right). \tag{11}$$

By setting $\Delta \mathrm{t} = \mathrm{t}_{\mathrm{j}} - \mathrm{t}_{\mathrm{j-1}}$ the formula (11) changes to

$$\hat{\mathrm{x}}_{\mathrm{i,j}} = \hat{\mathrm{x}}_{\mathrm{i,j-1}} + \left(\mathrm{t}_{\mathrm{j}} - \mathrm{t}_{\mathrm{j-1}}\right) \mathrm{f}_{\mathrm{i}}\left(\mathrm{t}_{\mathrm{j-1}}, \hat{\vec{\mathrm{x}}}_{\mathrm{j-1}}\right). \tag{12}$$

For plotting of the approximated state variable $\hat{\mathrm{x}}_{\mathrm{i,j}}$, each computed value $\hat{\mathrm{x}}_{\mathrm{i,j}}$ in t_{j} can be connected to the neighbouring values graphically. If the curve of the discrete time state variable $\hat{\mathrm{x}}_{\mathrm{i,j}}$ crosses the event threshold value $\mathrm{x}_{\mathrm{i}} = \mathrm{a}_{\mathrm{e}}$, then the last value before the event crossing point can be used to find the location time t_{e}. It is assumed that the magnitude of the discrete time state variable hits exactly the event threshold line $\mathrm{x}_{\mathrm{i}} = \mathrm{a}_{\mathrm{e}}$ at step j. Hence, define $\mathrm{x}_{\mathrm{i,j}} := \mathrm{a}_{\mathrm{e}}$ and consider it as an initial value. On the other hand, the value of the independent variable at the state event is presented by $\mathrm{t}_{\mathrm{j}} := \mathrm{t}_{\mathrm{e}}$. Hence, the zero crossing of the event function can be determined by quantifier t_{e} using the reformulated initial value problem as follows

$$\left(\mathrm{h}(\mathrm{t}, \vec{\mathrm{x}}(\mathrm{t})) := -\mathrm{a}_{\mathrm{e}} + \mathrm{x}_{\mathrm{i}}(\mathrm{t}) = 0\right) :\Leftrightarrow \left(\exists \mathrm{t}_{\mathrm{e}} : -\mathrm{a}_{\mathrm{e}} + \mathrm{x}_{\mathrm{i,j-1}} + \int_{\mathrm{t}_{\mathrm{j-1}}}^{\mathrm{t}_{\mathrm{e}}} \mathrm{f}_{\mathrm{i}}(\mathrm{t}, \vec{\mathrm{x}}(\mathrm{t}))\, \mathrm{dt} = 0\right). \tag{13}$$

It is assumed, that the event function $\mathrm{h}(\mathrm{t}, \vec{\mathrm{x}}(\mathrm{t}))$ is bijective on a small interval $\lfloor \mathrm{t}_{\mathrm{j-1}}, \mathrm{t}_{\mathrm{e}} \rfloor \subseteq I_{\mathrm{t}}$ and there t_{e} exist, which points to the zero location

$$0 = \mathrm{h}(\mathrm{t}_{\mathrm{e}}, \vec{\mathrm{x}}(\mathrm{t}_{\mathrm{e}}))$$

thus

$$t_e \leftrightarrow 0 .$$[12]

Hence, according to the reformulated initial value problem in (13), the reformulated explicit Euler's formula for threshold $\hat{x}_{i,j} := a_e$ is

$$h\left(\tilde{t}_e\right) = -a_e + \hat{x}_{i,j-1} + \left(\tilde{t}_e - t_{j-1}\right) f_i\left(t_{j-1}, \hat{\hat{x}}_{j-1}\right) = 0 .$$

(14)

The demanded value is $\tilde{t}_e$ and the other past values t_{j-1} and $x_{i,j-1}$ are known, hence, $\tilde{t}_e$[13] is computable from the event expression (14) as follows

$$\tilde{t}_e = t_{j-1} + \left| \frac{a_e - \hat{x}_{i,j-1}}{f_i\left(t_{j-1}, \hat{\hat{x}}_{j-1}\right)} \right|$$

(15)

if

$$\left| f_i\left(t_{j-1}, \hat{\hat{x}}_{j-1}\right) \right| \neq 0$$

is valid.

---

[12] The preposition $t_e \leftrightarrow 0$ is a minimalistic local expression of zero location in a hybrid system with consideration of an event function with a transition between two systems. For a system with domain D, it is assumed, that the viewpoint of an observer is restricted to its own system. If the observer stands out of D and observes the universe (big picture), which contains the various parallel systems, with synchronous and/or asynchronous state events, then instead of $t_e \leftrightarrow 0$ the preposition $\exists t_e : t_e \rightarrow 0$ is considered.

[13] Approximating event time t_e is carried out by preceding explicit solver magnitudes; therefor approximated value $\tilde{t}_e$ is relative to magnitude t_{j-1} and errors of preceding values affect $\tilde{t}_e$. In this dissertation, it is assumed that the local and global errors of solver solutions have infinitesimal magnitudes and the solver solutions track the exact solutions and do not exceed the fault margin and the computation methods of $\tilde{t}_e$ are applied to such systems.

After computing of $\tilde{t}_e$ by formula (15), the transient initial conditions for the system of ODE solver at the state event can be computed. The approximation of the transient state vector is given for step-size $\Delta\tilde{t}_e = \tilde{t}_e - t_{j-1}$ using explicit Euler's solver (16).

$$\hat{\vec{x}}_e = \hat{\vec{x}}_{j-1} + \Delta\tilde{t}_e \, \vec{f}\left(t_{j-1}, \hat{\vec{x}}_{j-1}\right)$$

(16)

Hence $\hat{\vec{x}}_e$ is an approximated initial vector at the system-switching event. If the approximated state vector component $\hat{x}_{e,i}$ shows a small deviation from the threshold value at the state event which indicates $\left|\hat{x}_{e,i} - a_e\right| < \varepsilon$, then the threshold value $x_i = a_e$ instead of $\hat{x}_{e,i}$ can be applied as initial value for the next system.

2.1.2 Reformulation of Implicit Solver Formula

This part presents an event location approximation concept for an implicit solver of an autonomous continuous ODE in the form $\dot{x}(t) = f(x(t))$ with a single variable $x \in D \subset \mathbf{R}$, mapping $f : D \to \mathbf{R}$ and differentiable solution $x : I_t \subset R^+ \to D$.

In this subchapter, an implicit Euler's solver is used for the approximation of the state variable. The solver formula is given by (17), at which j is the index of the state variable.

$$\hat{x}_j = \hat{x}_{j-1} + \Delta t \, f(\tilde{x}_j)$$

(17)

The solver computations and approximations of the state variable $x(t)$ continue until the event threshold magnitude $x = a_e$ is crossed.

An event function for an autonomous ODE for threshold value $x = a_e$ and t_e is given using a reformulated initial value problem as follows

$$(h(x(t)) := -a_e + x(t) = 0) :\Leftrightarrow \left(\exists t_e : -a_e + x_{j-1} + \int_{t_{j-1}}^{t_e} f(x(t))\,dt = 0 \right).$$

(18)

Hence, the solver formula (17) should be established according to the reformulated initial value problem (18), which is represented by (19). For computation of the independent variable t at the state event, the step-size at the state event can be given by $\Delta\tilde{t}_e := \tilde{t}_e - t_{j-1}$. The known previous values $\hat{x}_{j-1}$ and t_{j-1} are considered as constant values. In addition, the current approximated value $\hat{x}_j$ and the predicted state value $\tilde{x}_j$ are set equal to the threshold value $x = a_e$. Thus the resulting equation for implicit Euler's solver for computing t_e is

$$h(\tilde{t}_e) = -a_e + \hat{x}_{j-1} + (\tilde{t}_e - t_{j-1}) f(a_e) = 0.$$

(19)

If $f(a_e)$ is not zero and if the solver shows a small error, then the equation (19) can be solved for $\tilde{t}_e$ as follows

$$\tilde{t}_e = t_{j-1} + \left| \frac{a_e - \hat{x}_{j-1}}{f(a_e)} \right|.$$

(20)

Hence, the event location for the implicit Euler's method on t-axis is computed using formula (20).

For non-autonomous systems, the functional part of implicit Euler's solver (17) is contains the independent variable t_j e.g. $\hat{x}_j = \hat{x}_{j-1} + \Delta t\, f(t_j, \tilde{x}_j)$. In this case the variable t_j is replaced with $\tilde{t}_e$ and reformulated implicit Euler's equation can be solved for $\tilde{t}_e$. The equation

$$-a_e + \hat{x}_{j-1} + (\tilde{t}_e - t_{j-1}) f(\tilde{t}_e, a_e) = 0$$

may not be solved analytically. In such cases, the root-finding methods can be applied.

2.2 Event Location Approximation via Reformulation of Solver Formula for Root-Finding Methods

Subchapters 2.2.1 and 2.2.2 introduce the approximation of the transient state vector by applying the root-finding methods. The reformulation of the explicit solver formula to use the root-finding method is shown in subchapter 2.2.1. The usage of this method on high order explicit solvers makes sense regarding system simulations. Subchapter 2.2.2 focuses on the approximation of the transient state vector via reformulating implicit solver systems using root-finding methods.

2.2.1 Reformulation of Explicit Solver Formula for Root-Finding Methods

This subchapter presents a method for approximation of the transient state vector at the state event for the non-autonomous ODE system $\dot{\vec{x}}(t)=\vec{f}(t,\vec{x}(t))$. In the following, the same ODE system as described in 2.1.1 is used.

The root-finding methods[14] compute approximately the magnitude of the independent variable at the state event by appropriate reformulated explicit solver formula e.g. reformulated explicit Euler's method. For a root-finding method, it is assumed, that at the state event, all past values e.g. $\hat{\vec{x}}_{j-1}$, t_{j-1} are known as constant values and the present value $\hat{x}_{i,j}$ is assumed to be approximately equal to the threshold value $\hat{x}_{i,j} \approx a_e$, which then set $\hat{x}_{i,j} := a_e$. The computed value of location t_e is given by the last step-size $\Delta\tilde{t}_e$ at the state event which is defined as $\tilde{t}_e = \Delta\tilde{t}_e + t_{j-1}$. The event function is defined by the reformulation of the initial value problem as in axiom (13). The reformulated solver method is considered as an event function $\hat{h}(\Delta t_e)$ defined by

$$\hat{h}(\Delta\tilde{t}_e) := -a_e + \hat{x}_{i,j-1} + \Delta\tilde{t}_e \, f_i\left(t_{j-1}, \hat{\vec{x}}_{j-1}\right). \tag{21}$$

For some ODE solver, the event function $\hat{h}(\Delta\tilde{t}_e)$ can be a nonlinear function. The equation (22) shows the event function (21) at zero-crossing location, where it may be solved using the root-finding methods.

---

[14] These are e.g. bisection, fixed-point iteration, Illinois, Newton, Pegasus, regula falsi, secant etc.

$$\hat{h}\left(\Delta\tilde{t}_e\right) = 0$$

(22)

Hence, the event equation (22) can be solved approximately by the root-finding methods to compute the event step-size[15] $\Delta\tilde{t}_e$. For finding of the root location, the iterative Newton's method with the iteration indicator[16] k can be applied as follows

$$\Delta\tilde{t}_{e,k} = \Delta\tilde{t}_{e,k-1} - \left(\frac{d}{d\Delta t}\hat{h}\left(\Delta\tilde{t}_{e,k-1}\right)\right)^{-1}\left(\hat{h}\left(\Delta\tilde{t}_{e,k-1}\right)\right).$$

(23)

In each iteration the magnitudes of $\hat{h}\left(\Delta\tilde{t}_{e,k}\right)$ and $\frac{d}{d\Delta t}\hat{h}\left(\Delta\tilde{t}_{e,k}\right)$ are evaluated using the current value $\Delta\tilde{t}_{e,k}$. The transient state vector $\hat{\vec{x}}_e$ in the system of explicit Euler's solver (16) can be approximated using the known previous ODE solver values e.g. t_{j-1}, $\hat{\vec{x}}_{j-1}$ and the computed event step-size $\Delta\tilde{t}_e := \Delta\tilde{t}_{e,k}$.

2.2.2 Reformulation of Implicit Solver Formula for Root-Finding Methods

In this subchapter, a non-autonomous continuous system of the differential equation $\dot{\vec{x}}(t) = \vec{f}(t, \vec{x}(t))$ is given with $\vec{x} \in D \subset \mathbf{R}^n$, $n \in \mathbf{N}$, the mapping $\vec{f} : I_t \times D \to \mathbf{R}^n$ and the solution $\vec{x} : I_t \to D$ for a finite interval $I_t = [t_0, t_\sigma] \subset \mathbf{R}^+$. The state variable $x_i \in U_i : (U_i \subset \mathbf{R}) \subset D$, $i \in \{1, 2, \ldots n\}$ is assumed as a variable which is part of the state event. The ODE of $x_i(t)$ is considered as the last[17] ODE of $\dot{\vec{x}}(t) = \vec{f}(t, \vec{x}(t))$. Hence it is assumed that $i = n$, thus a vector

---

[15] An event step-size is a solver step-size. By means of an event step-size, concerned initial magnitudes and using solver algorithm, the event location can be approximated.

[16] In this dissertation, the iterations of solver algorithms are specified by the index variable j and the iterations of the root-finding methods are specified by index k.

[17] The state variable, which is involved in state event, can have an arbitrary position in state vector and it may be handled with methods in chapters 2, 3 and 4. In addition, a change of system axes to an applicable order can be realized in the rectangular Cartesian coordinate system in cyclic or in anti-cyclic form. Hence, the influence of a coordinate change on signs of the resulted system should be contemplated.

$$\vec{\chi} \in \mathbf{R}^{(n-1)}, \ \vec{\chi}(t) = \{(\vec{x}(t) \mid \vec{x}(t) \setminus x_n(t))\}$$

with $n-1$ state variables is defined.
Therefore

$$\vec{\chi}(t) = (\chi_1(t), \chi_2(t), \ldots, \chi_{n-1}(t))^T$$

and the event function

$$h_n(t, \vec{x}(t)) := -a_e + x_n(t) = 0$$

are defined with the state variable that is involved in the state event[18]. It is demonstrated as the last equation of the reformulated initial value problem in (24).

In axiom (24) the event location is conditioned upon intersection of the state variable $x_n(t)$ with $x_n = a_e$ at t_e in system of equations by the reformulated initial value problem.

$$\left(\vec{h}(t, \vec{x}(t)) = \vec{0}\right) :\Leftrightarrow \left(\exists t_e : \begin{pmatrix} -\vec{\chi}(t_e) \\ -a_e \end{pmatrix} + \vec{x}_{j-1} + \int_{t_{j-1}}^{t_e} \vec{f}(t, \vec{x}(t))\,dt = \vec{0} \right)$$

(24)

The discrete time solver algorithm in this section is the implicit trapezoidal solver and can be given by the following formula

$$\hat{\vec{x}}_j = \hat{\vec{x}}_{j-1} + \frac{\Delta t}{2}\left(\vec{f}\left(t_j, \tilde{\vec{x}}_j\right) + \vec{f}\left(t_{j-1}, \hat{\vec{x}}_{j-1}\right)\right).$$

(25)

The solver approximation works in two stages. First, the initial vector $\tilde{\vec{x}}_j$ is predicted[19] and then the implicit solver algorithm is computed to approximate the state vector $\hat{\vec{x}}_j$ based on the predicted vector $\tilde{\vec{x}}_j$.

---

[18] An occurrence can be characterized by intersection of a threshold value line with a state variable curve. This occurrence can be defined as a “state event” and the state variable, which is participated in the occurrence, is named the state variable that is “involved in state event”.

Formula (26) presents the event vector function for the trapezoidal solver (25) fitting to the reformulated initial value problem in definition (24). In the event vector function (26), for computation of the state event coordinates, the state variable $\tilde{x}_{n,j}$ is approximately set equal to the appropriate threshold value $\tilde{x}_{n,j} := a_e$. After this setting, the number of state variables of the state vector $\tilde{\vec{x}}_k$ is reduced to $n-1$. In addition, the step-size $\Delta\tilde{t}_{e,k}$ in (26) at the state event is considered as an unknown variable and is included into the set of variables. Thus, the $n-1$ state variables symbolized with $\tilde{\vec{\chi}}_k$ together with the unknown step-size $\Delta\tilde{t}_{e,k}$ result in n unknown variables. Then the event vector function (26) has n variables and n functions therefore the root-finding method can be applied. It is assumed, that the variable t_j in (26) is equal to its appropriate predicted event value $t_j := \tilde{t}_{e,k} = \Delta\tilde{t}_{e,k} + t_{j-1}$. In the root-finding process, $\Delta\tilde{t}_e$ is updated in every iteration loop regarding index k, in which (for a root-finding algorithm) t_{j-1} is a constant value. The event vector function is given by (26) using the variable set $\{\Delta\tilde{t}_{e,k}, \tilde{\vec{\chi}}_k\}$, the set of constant values $\{t_{j-1}, \hat{\vec{x}}_{j-1}, a_e\}$ and $\tilde{x}_{n,k} := a_e$.

$$\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k}, \tilde{\vec{\chi}}_k\right) := \begin{pmatrix} -\tilde{\vec{\chi}}_k \\ -a_e \end{pmatrix} + \hat{\vec{x}}_{j-1} + \frac{\Delta\tilde{t}_{e,k}}{2}\left(\vec{f}\left(\Delta\tilde{t}_{e,k} + t_{j-1}, \tilde{\vec{\chi}}_k, a_e\right) + \vec{f}\left(t_{j-1}, \hat{\vec{x}}_{j-1}\right)\right)$$

(26)

Vector function (26) can be summarized as following system of equations

$$\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k}, \tilde{\vec{\chi}}_k\right) = \vec{0}.$$

(27)

After the detection of the crossing of the threshold value $x_n = a_e$ with $x_{n,j}$, the step-size magnitude and the state vector $\vec{\chi}(t_e)$ can be computed by a root-finding methods of nonlinear systems of equations. The Newton's method for predicting Δt_e and $\vec{\chi}(t_e)$ at the state event for equations (27) is applied as follows

---

[19] The prediction stage is accomplished using root-finding methods for nonlinear systems of equations applied on system of reformulated implicit solver formulas.

$$\begin{pmatrix} \Delta\tilde{t}_{e,k} \\ \tilde{\vec{\chi}}_{k} \end{pmatrix} = \begin{pmatrix} \Delta\tilde{t}_{e,k-1} \\ \tilde{\vec{\chi}}_{k-1} \end{pmatrix} - \left(J_{\hat{\vec{h}}}\left(\Delta\tilde{t}_{e,k-1}, \tilde{\vec{\chi}}_{k-1}\right)\right)^{-1} \left(\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k-1}, \tilde{\vec{\chi}}_{k-1}\right)\right).$$

(28)

In Newton's method (28)

$$J_{h}\left(\Delta\tilde{t}_{e,k-1}, \tilde{\vec{\chi}}_{k-1}\right) \neq 0$$

is the appropriate Jacobian matrix. After the prediction phase, the event magnitudes are the event step-size $\Delta t_{e} \approx \Delta\tilde{t}_{e} := \Delta\tilde{t}_{e,k}$ and the transient state vector

$$\left(\vec{\chi}\left(t_{e}\right)^{T}, x_{i}\left(t_{e}\right)\right)^{T} \approx \tilde{\vec{x}}_{e} := \left(\tilde{\vec{\chi}}_{k}^{T}, a_{e}\right)^{T}.$$

The predicted transient state vector $\tilde{\vec{x}}_{e}$ and the predicted step-size $\Delta\tilde{t}_{e}$ are used in the implicit solver formula (25). Thus, the approximated transient state vector $\hat{\vec{x}}_{e}$ is computed as shown in (29).

$$\hat{\vec{x}}_{e} = \hat{\vec{x}}_{j-1} + \frac{\Delta\tilde{t}_{e}}{2}\left(\vec{f}\left(\Delta\tilde{t}_{e} + t_{j-1}, \tilde{\vec{x}}_{e}\right) + \vec{f}\left(t_{j-1}, \hat{\vec{x}}_{j-1}\right)\right)$$

(29)

3 State Event Handling in System of DAEs

DAEs can be classified by its characteristics. The general classification can be divided into two main categories, linear and nonlinear systems. Another classification can be given by means of the algebraic equations of the DAEs. There are three types of DAEs: Fully-implicit systems with hidden algebraic equations, semi-explicit systems and transformed explicit systems. Changing the type of DAEs may be possible. A fully-implicit system may be transformed into a semi-explicit system or an explicit system. Another classification can be given using the index of DAEs.

This chapter focuses on the approximation methods of transient vector for DAEs for following cases:

- DAEs transformed into ODEs

  In this case, a system of DAEs is transformed into a system of ODEs and then the approximation methods of transient state vectors for the system of ODEs can be applied.

- Semi-explicit DAEs

  In this case, either the system of semi-explicit DAEs is transformed into a system of ODEs as above or the transient vector in the semi-explicit DAEs is approximated without index-reduction by reformulation of implicit solver formula for root-finding methods of nonlinear systems of equations.

3.1 Index-Reduction

The fully-implicit DAEs can be shown as follows

$$\vec{F}\left(t,\vec{x}(t),\dot{\vec{x}}(t)\right)=\vec{0}. \tag{30}$$

The Jacobian matrix $\left(\frac{\partial\vec{F}}{\partial\dot{\vec{x}}}\left(t,\vec{x}(t),\dot{\vec{x}}(t)\right)\right)$ of equation (30) is singular, therefore it cannot be solved for $\dot{\vec{x}}(t)$ and it may contain hidden algebraic equations. The domains of the dependent variables of (30) are $\vec{x}\in D_x$, $\dot{\vec{x}}\in D_{\dot{x}}$ with $D_x, D_{\dot{x}}\subset\mathbf{R}^n$. The vector function of equation (30) indicates the mapping

$$\vec{F}: I_t\times D_x\times D_{\dot{x}}\to\mathbf{R}^n$$

for a finite interval I_t with $t \in I_t \subset \mathbf{R}^+$ and solution $\mathbf{x} : I_t \to D_x$.

If equation (30) transforms into an explicit system [SIC05] and if its Jacobian matrix fulfills

$$\det\left(\frac{\partial \vec{F}}{\partial \dot{\vec{x}}}\left(t, \vec{x}(t), \dot{\vec{x}}(t)\right)\right) \neq 0$$

and is locally unique in the surroundings of the solution, then equation (30) can be solved locally for $\dot{\vec{x}}(t)$ in terms of the other variables. After transforming into an explicit system, the ODEs $\dot{\vec{x}}(t) = \vec{f}(t, \vec{x}(t))$ can be handled with ODE solvers.

The equations in DAEs (30) may contain algebraic equations. In some cases, it is possible to extract the algebraic equations from the fully-implicit systems by some algebraic operations [AGI12], if possible, the result can be formulated as a semi-explicit system. The semi-explicit DAEs including the algebraic vector equation $\vec{g}(t, \vec{x}(t), \vec{y}(t))$ and the algebraic variable vector $\vec{y}(t)$ are given by

$$\begin{aligned} \dot{\vec{x}}(t) &= \vec{f}(t, \vec{x}(t), \vec{y}(t)) \\ \vec{0} &= \vec{g}(t, \vec{x}(t), \vec{y}(t)). \end{aligned} \tag{31}$$

The semi-explicit DAEs (31) are defined by following mappings

$$\vec{f} : I_t \times D_x \times D_y \to \mathbf{R}^n, \; \vec{g} : I_t \times D_x \times D_y \to \mathbf{R}^{n_y}.$$

The algebraic variable vector is given by $\vec{y} \in D_y \subset \mathbf{R}^{n_y}$ with $n, n_y \in \mathbf{N}$ and the solutions are

$$\vec{x} : I_t \to D_x, \; \vec{y} : I_t \to D_y.$$

The semi-explicit DAEs may be transformed to ODEs by differentiating its algebraic equations. The smallest necessary number of differentiation steps of the constraint equation, with respect to the independent variable t along the solution of the algebraic variable for transformation into the ODEs, is considered as the differential index of the DAEs. An example is the equation $0 = g(t, \vec{x}(t), y(t))$ with index 1 and algebraic variable $y(t)$. After the first differentiation of $g(t, \vec{x}(t), y(t))$ with respect to t, the following equation is suspected

$$\frac{\partial g}{\partial x_1}(t, \vec{x}(t), y(t)) \frac{d}{dt} x_1(t) + \cdots + \frac{\partial g}{\partial x_n}(t, \vec{x}(t), y(t)) \frac{d}{dt} x_n(t) + \frac{\partial g}{\partial y}(t, \vec{x}(t), y(t)) \frac{d}{dt} y(t) + \frac{d}{dt} g(t, \vec{x}(t), y(t)) = 0 \cdot \tag{32}$$

In equation (32) $\frac{d}{dt} y(t)$ can be given in from of an ODE as follows

$$\dot{y}(t) = -\left(\frac{\partial g}{\partial y}(t, \vec{x}(t), y(t))\right)^{-1} \left(\frac{\partial g}{\partial x_1}(t, \vec{x}(t), y(t)) f_1(t, \vec{x}(t), y(t)) + \cdots + \frac{\partial g}{\partial x_n}(t, \vec{x}(t), y(t)) f_n(t, \vec{x}(t), y(t)) + \frac{d}{dt} g(t, \vec{x}(t), y(t))\right) \tag{33}$$

in which $\frac{\partial g}{\partial y}(t, \vec{x}(t), y(t))$ should be nonsingular.

A solution of the transformed equation may not always be equivalent to the solution of algebraic equation [AGI12]. Hence, a numerical approximation of solution of the DAEs transformed into ODEs may not always yield the same result as a numerical approximation of its semi-explicit system.

3.2 Event Location Approximation in DAEs Transformed to System of ODEs

A transformation from fully-implicit DAEs to semi-explicit DAEs may be given by certain algebraic operations and from semi-explicit DAEs to ODEs by index-reduction. After this transformation, the ODEs can be solved using ODE solvers. Then the appropriate state event location can be approximated applying methods explained in subchapters 2.1.1, 2.2.1 and 2.2.2. Hence, in the same manner, the last step-size before the state event can be computed. In this way, the transient vector of a hybrid or variable structure system of DAEs transformed into ODEs can be approximated using the initial values and computed step-size.

3.3 Event Location Approximation in System of Semi-Explicit DAEs

This subchapter presents root-finding method applications for approximating event locations of systems of semi-explicit DAEs applied in following two cases:

- The subchapter 3.3.1 demonstrates the approximation for an event location associated with an algebraic variable.
- Subchapter 3.3.2 presents the root-finding application for approximating the state event location in case of an event defined by a state variable.

3.3.1 Event Location of Algebraic Variable

In this subchapter, an event is defined for the system of semi-explicit DAEs (31) by intersecting of set

$$\exists P_{\mathrm{y_i}}{}^{20},\ P_{\mathrm{y_i}} \subset \left\{I_{\mathrm{t}} \times D_{\mathrm{x}} \times D_{\mathrm{y}}\right\} : \left(P_{\mathrm{y_i}} := \left\{\left(\mathrm{t}, \vec{\mathrm{x}}^{\mathrm{T}}, \vec{\mathrm{y}}^{\mathrm{T}}\right)^{\mathrm{T}} \mid \mathrm{y_i} \perp p_{\mathrm{y_i}}, p_{\mathrm{y_i}} : \mathrm{y_i} - \mathrm{a_e} = 0\right\}\right)$$

with trajectory set $\exists T_{\mathrm{y_i}} : (\mathrm{t}, \mathrm{y_i})^{\mathrm{T}} \in T_{\mathrm{y_i}} \subseteq \left(I_{\mathrm{t}} \times U_{\mathrm{i}}\right)$, in which the algebraic variable $\mathrm{y_i} \in U_{\mathrm{i}} : \left(U_{\mathrm{i}} \subset \mathbf{R}\right) \subset D_{\mathrm{y}}$, $\mathrm{i} \in \{1, 2, \ldots \mathrm{n_y}\}$, $\mathrm{n_y} \in \mathbf{N}$ is involved in the event. Then the magnitude of $\mathrm{y_i(t)}$ at the event is $\mathrm{a_e} = \mathrm{y_i}(\mathrm{t_e}) \in \left(P_{\mathrm{y_i}} \cap T_{\mathrm{y_i}}\right)$.

It is assumed, that the algebraic variable $\mathrm{y_i(t)}$ and its appropriate equation establish the last equation of the semi-explicit DAEs (31). At the state event, $\mathrm{y_{n_y}}(\mathrm{t_e})$ is considered as a constant value $\mathrm{y_{n_y}}(\mathrm{t_e}) := \mathrm{a_e}$, thus the dimension of the algebraic variable vector $\vec{\mathrm{y}}(\mathrm{t_e})$ is reduced to $\mathrm{n_y} - 1$. The prediction vector of the algebraic variables is shown with

$$\tilde{\vec{\gamma}} : \left(\tilde{\vec{\gamma}} \in \mathbf{R}^{\mathrm{n_y}-1} \wedge \left(\left(\gamma_1(\mathrm{t}), \gamma_2(\mathrm{t}), \ldots, \gamma_{\mathrm{n_y}-1}(\mathrm{t})\right)^{\mathrm{T}} = \left\{\left(\vec{\mathrm{y}}(\mathrm{t}) \mid \vec{\mathrm{y}}(\mathrm{t}) \setminus \mathrm{y_{n_y}}(\mathrm{t})\right)\right\}\right)\right).$$

The event function at the intersection plane is described by

$$\mathrm{h_n}\left(\mathrm{t}, \vec{\mathrm{x}}(\mathrm{t}), \vec{\gamma}(\mathrm{t})\right) := \mathrm{g_n}\left(\mathrm{t}, \vec{\mathrm{x}}(\mathrm{t}), \vec{\gamma}(\mathrm{t})\right) = -\mathrm{a_e} + \mathrm{y_{n_y}}(\mathrm{t}) = 0$$

---

[20] $P_{\mathrm{y_i}}$ is a set which contains all points of plane $p_{\mathrm{y_i}} : \mathrm{y_i} - \mathrm{a_e} = 0$. The plane $p_{\mathrm{y_i}}$ is normal to the $\mathrm{y_i}$ -axis.

and the zero crossing can be defined at t_e using following axiom

$$\left(\vec{h}\left(t,\vec{x}(t),\vec{y}(t)\right)=\vec{0}\right):\Leftrightarrow\left(\exists t_e:\begin{pmatrix}-\vec{x}(t_e)+\vec{x}_{j-1}+\displaystyle\int_{t_{j-1}}^{t_e}\vec{f}\left(t,\vec{x}(t),\vec{y}(t)\right)dt\\ \vec{g}\left(t_e,\vec{x}(t_e),\vec{\gamma}(t_e)\right)\end{pmatrix}=\vec{0}\right).$$

(34)

The implicit solver system consists of two stages. The first stage is a prediction of the algebraic variable and the state vectors. The second stage is the approximation of the state vector. Formula (35) presents an implicit numerical concept for semi-explicit DAEs, in which the vectors $\widetilde{\vec{x}}_j$ and $\widetilde{\vec{y}}_j$ show the predicted magnitudes whereas $\hat{\vec{x}}_j$ shows the approximated vector.

$$\begin{pmatrix}\hat{\vec{x}}_j\\ \vec{0}\end{pmatrix}=\begin{pmatrix}\hat{\vec{x}}_{j-1}+\dfrac{\Delta t}{2}\left(\vec{f}\left(t_j,\widetilde{\vec{x}}_j,\widetilde{\vec{y}}_j\right)+\vec{f}\left(t_{j-1},\hat{\vec{x}}_{j-1},\widetilde{\vec{y}}_{j-1}\right)\right)\\ \vec{g}\left(t_j,\widetilde{\vec{x}}_j,\widetilde{\vec{y}}_j\right)\end{pmatrix}$$

(35)

The implicit trapezoidal method and algebraic equation in formula (35) are reformulated in (36) as an event vector function. At the state event, the variables in vector function (36) are specified by iteration index k.

$$\hat{\vec{h}}\left(\Delta\widetilde{t}_{e,k},\widetilde{\vec{x}}_k,\widetilde{\vec{\gamma}}_k\right):=\begin{pmatrix}-\widetilde{\vec{x}}_k+\hat{\vec{x}}_{j-1}+\dfrac{\Delta\widetilde{t}_{e,k}}{2}\left(\vec{f}\left(\Delta\widetilde{t}_{e,k}+t_{j-1},\widetilde{\vec{x}}_k,\widetilde{\vec{\gamma}}_k\right)+\vec{f}\left(t_{j-1},\hat{\vec{x}}_{j-1},\widetilde{\vec{y}}_{j-1}\right)\right)\\ \vec{g}\left(\Delta\widetilde{t}_{e,k}+t_{j-1},\widetilde{\vec{x}}_k,\widetilde{\vec{\gamma}}_k\right)\end{pmatrix}$$

(36)

The event vector function (36) is presented as a system of equations in (37) according to reformulated initial value problem (34) for predicting the event step-size, transient state vector and transient algebraic variable vector.

Thus, after event identification, e.g. when the algebraic variable crossing its concerned threshold value, the data processing should be switching from solver iteration algorithm to root-finding algorithm. The root-finding methods of nonlinear systems of equations can be applied on following equation

$$\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k}, \tilde{\vec{x}}_k, \tilde{\vec{\gamma}}_k\right) = \vec{0}\,.$$

(37)

The root location in system of (37) can be determined using the Newton's method as shown in (38). The vector $\left(\Delta\tilde{t}_{e,k}, \tilde{\vec{x}}_k^T, \tilde{\vec{\gamma}}_k^T\right)^T$ stands for the prediction vector and contains $n + n_y$ variables. The set $\left\{t_{j-1}, \hat{\vec{x}}_{j-1}, \hat{\vec{\gamma}}_{j-1}, a_e\right\}$ contains known constants and $J_h\left(\Delta\tilde{t}_k, \tilde{\vec{x}}_k, \tilde{\vec{\gamma}}_k\right)$ is the Jacobian matrix of (37).

$$\begin{pmatrix} \Delta\tilde{t}_{e,k} \\ \tilde{\vec{x}}_k \\ \tilde{\vec{\gamma}}_k \end{pmatrix} = \begin{pmatrix} \Delta\tilde{t}_{e,k-1} \\ \tilde{\vec{x}}_{k-1} \\ \tilde{\vec{\gamma}}_{k-1} \end{pmatrix} - J_{\hat{h}}^{-1}\left(\Delta\tilde{t}_{e,k-1}, \tilde{\vec{x}}_{k-1}, \tilde{\vec{\gamma}}_{k-1}\right)\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k-1}, \tilde{\vec{x}}_{k-1}, \tilde{\vec{\gamma}}_{k-1}\right)$$

(38)

If the Jacobian matrix of the vector function (36) is not singular and Newton's method (38) approximately converged then function (37) is close to the zero vector and

$$\left(\lim_{k\to+\infty}\left(\max_{1\le\alpha\le n+n_y}\left|\hat{h}_\alpha\left(\Delta\tilde{t}_{e,k-1}, \tilde{\vec{x}}_{k-1}, \tilde{\vec{\gamma}}_{k-1}\right)\right|\right)\right) \le \varepsilon$$

for $\varepsilon > 0$ infinitesimal. In this case, the resulting predicted transient initial variables are

$$\Delta\tilde{t}_e := \Delta\tilde{t}_{e,k},\ \tilde{t}_e := t_{j-1} + \Delta\tilde{t}_e,\ \tilde{\vec{x}}_e := \tilde{\vec{x}}_k,\ \left(\vec{\gamma}(t_e)^T, y_i(t_e)\right)^T \approx \tilde{\vec{y}}_e := \left(\tilde{\vec{\gamma}}_k^T, a_e\right)^T.$$

They can be initialized in solver system (35). The transient state vector can be approximated as follows

$$\begin{pmatrix} \hat{\vec{x}}_e \\ \vec{0} \end{pmatrix} = \begin{pmatrix} \hat{\vec{x}}_{j-1} + \dfrac{\Delta\tilde{t}_e}{2}\left(\vec{f}\left(\tilde{t}_e, \tilde{\vec{x}}_e, \tilde{\vec{y}}_e\right) + \vec{f}\left(t_{j-1}, \hat{\vec{x}}_{j-1}, \tilde{\vec{y}}_{j-1}\right)\right) \\ \vec{g}\left(\tilde{t}_e, \tilde{\vec{x}}_e, \tilde{\vec{y}}_e\right) \end{pmatrix}.$$

(39)

3.3.2 Event Location of State Variable

In this subchapter, the state event is defined by a state variable $x_i \in U_i : (U_i \subset \mathbf{R}) \subset D_x, i \in \{1,2,\ldots n\}$, $n \in \mathbf{N}$. The state event $x_i(t_e) = a_e$ is an element of the intersection of trajectory set $\exists T_{x_i} : (t, x_i)^T \in T_{x_i} \subseteq (I_t \times U_i)$ and set P_{x_i}.

$$\exists P_{x_i} \subset \{I_t \times D_x \times D_y\} : \left(P_{x_i} := \left\{(t, \vec{x}^T, \vec{y}^T)^T \mid p_{x_i} \perp x_i, p_{x_i} : x_i - a_e = 0\right\}\right) \wedge \left(x_i(t_e) \in (P_{x_i} \cap T_{x_i})\right).$$

The position of $x_i(t)$ and its equation in semi-explicit DAEs (31) is assumed to be at the last component of the state vector. Since the state variable $x_n(t)$ crosses the threshold value $x_n = a_e$ at the state event, $x_n(t)$ is set equal to the appropriate threshold value $x_n(t_e) := a_e$ and it is considered as constant. The resulting state vector remains with $n-1$ variables defined as

$$\vec{\chi} : \left(\vec{\chi} \in \mathbf{R}^{n-1} \wedge \left((\chi_1(t), \chi_2(t), \ldots, \chi_{n-1}(t))^T = \{(\vec{x}(t) \mid \vec{x}(t) \setminus x_n(t))\}\right)\right).$$

The event occurs at $a_e = x_n(t_e)$ and is specified by the reformulated initial value problem

$$-a_e + x_{n,j-1} + \int_{t_{j-1}}^{t_e} f_n(t, \vec{x}(t), \vec{y}(t))\,dt = 0$$

with quantifier t_e as part of system (40). Axiom (40) defines the zero crossing of the event vector function for t_e via system of integrations and algebraic equations conditioned by $x_i(t_e) - a_e = 0$.

$$\left(\vec{h}(t, \vec{x}(t), \vec{y}(t)) = \vec{0}\right) :\Leftrightarrow \left(\exists t_e : \left(\begin{matrix} \begin{pmatrix} -\vec{\chi}(t_e) \\ -a_e \end{pmatrix} + \vec{x}_{j-1} + \int_{t_{j-1}}^{t_e} \vec{f}(t, \vec{x}(t), \vec{y}(t))\,dt \\ \vec{g}(t_e, \vec{\chi}(t_e), \vec{y}(t_e)) \end{matrix} \right) = \vec{0} \right)$$

(40)

The event vector function (41) presents an numerical system including the reformulated trapezoidal method for variable predictions at the state event with iteration index k.

$$\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k}, \tilde{\vec{\chi}}_k, \tilde{\vec{y}}_k\right) := \begin{pmatrix} -\begin{pmatrix} \tilde{\vec{\chi}}_k \\ a_e \end{pmatrix} + \hat{\vec{x}}_{j\text{-}1} + \dfrac{\Delta\tilde{t}_{e,k}}{2}\left(\vec{f}\left(\Delta\tilde{t}_{e,k} + t_{j-1}, \tilde{\vec{\chi}}_k, \tilde{\vec{y}}_k\right) + \vec{f}\left(t_{j-1}, \hat{\vec{x}}_{j\text{-}1}, \tilde{\vec{y}}_{j\text{-}1}\right)\right) \\ \vec{g}\left(\Delta\tilde{t}_{e,k} + t_{j-1}, \tilde{\vec{\chi}}_k, \tilde{\vec{y}}_k\right) \end{pmatrix} \tag{41}$$

In event vector function (41) the vectors $\hat{\vec{x}}_{j\text{-}1}$, $\tilde{\vec{y}}_{j\text{-}1}$ and magnitudes of t_{j-1} and $\hat{x}_{n,j}$ are considered as known constant values. The vectors $\tilde{\vec{\chi}}_k$, $\tilde{\vec{y}}_k$ and variable $\Delta\tilde{t}_{e,k}$ are demanded values, therefore the prediction vector

$$\left(\Delta\tilde{t}_{e,k}, \tilde{\vec{\chi}}_k^{\,T}, \tilde{\vec{y}}_k^{\,T}\right)^T$$

has dimension $n + n_y$. The system of equations in (42) can be solved by root-finding algorithms.

$$\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k}, \tilde{\vec{\chi}}_k, \tilde{\vec{y}}_k\right) = \vec{0} \tag{42}$$

As root-finding method, the Newton's method can be applied as follows

$$\begin{pmatrix} \Delta\tilde{t}_{e,k} \\ \tilde{\vec{\chi}}_k \\ \tilde{\vec{y}}_k \end{pmatrix} = \begin{pmatrix} \Delta\tilde{t}_{e,k-1} \\ \tilde{\vec{\chi}}_{k-1} \\ \tilde{\vec{y}}_{k-1} \end{pmatrix} - J_{\hat{\vec{h}}}^{-1}\left(\Delta\tilde{t}_{e,k-1}, \tilde{\vec{\chi}}_{k-1}, \tilde{\vec{y}}_{k-1}\right)\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k-1}, \tilde{\vec{\chi}}_{k-1}, \tilde{\vec{y}}_{k-1}\right). \tag{43}$$

The predicted event values are

$$\Delta\tilde{t}_e := \Delta\tilde{t}_{e,k},\ \tilde{t}_e := t_{j-1} + \Delta\tilde{t}_{e,k},\ \vec{y}(t_e) \approx \tilde{\vec{y}}_e := \tilde{\vec{y}}_k,\ \left(\vec{\chi}(t_e)^T, x_i(t_e)\right)^T \approx \tilde{\vec{x}}_e := \left(\tilde{\vec{\chi}}_k^{\,T}, a_e\right)^T.$$

Hence the step-size $\Delta\tilde{t}_e$, the transient state vector $\tilde{\vec{x}}_e$ and the transient algebraic variable vector $\tilde{\vec{y}}_e$ can be set in formula (35). Therefore the transient state vector $\hat{\vec{x}}_e$ can be approximated as shown in (39).

4 Henon's Method

The Henon's method presents the approximation of state event location using a transformation via exchanging the independent variable with the variable involved[21] in the state event. The Henon's method [MHO82] presents a numerical approximation of intersection location of dynamic system trajectories with a surface of section[22]. The reference [BEK12] points to the changing of independent variable with dependent variable for a state event approximation by Henon's method.

This chapter focuses on formulating the Henon's method with transformed initial value problem and integral intervals for approximating the transient state vector in hybrid or variable structure systems of ODEs and/or semi-explicit DAEs using Henon's method.

4.1 Henon's Method in System of ODEs

The Henon's method can be applied to an autonomous system $\dot{\vec{x}}(t) = \vec{f}(\vec{x}(t))$ with the mapping $\vec{f} : D \subset \mathbf{R}^n \to \mathbf{R}^n$, $\vec{x} \in D$ and the solution $\vec{x} : I_t \subset \mathbf{R}^+ \to \mathbf{R}^n$. It is assumed, that a state variable involved in the state event, is given by $x_i \in U_i : (U_i \subset \mathbf{R}) \subset D$ with $i \in \{1,2,\ldots n\}$, $n \in \mathbf{N}$. The ODE component $\dot{x}_i(t) = f_i(\vec{x}(t))$ is considered as the last ODE $i = n$.

In a normal case, before the crossing of the threshold value $x_n = a_e$ by $x_n(t)$, the form (44) can be applied.

$$\frac{d}{d\tau}\vec{x}(t) = \Gamma \, \vec{f}(\vec{x}(t))$$

(44)

In this case, Γ is assigned by

$$\Gamma := \frac{dt}{d\tau}$$

(45)

---

[21] The singular point of an event function can be defined as an occurrence of an event by intersection of a threshold value or a plane as described in subchapters 2, 3.3.2 and 3.3.1.
A variable, which crosses the intersection plane or threshold value, is called here a "variable" that is "involved in event".

[22] Surface of section is presented by Henri Poincare. It is a subspace, which is used for observation of intersection of periodic motion orbits with section of subspace. The map of the intersections is named Poincare's map.

thus the system of ODEs (44) can be simplified as follows

$$\frac{d}{dt}\vec{x}(t)=\vec{f}(\vec{x}(t)).$$

(46)

In differential equation (46) t is the independent variable and $\vec{x}(t)$ is the dependent state vector. It is assumed that the state vector of (46) is computed at step t_j via integral (47) with the initial values $\vec{x}(t_{j-1})$ and $t_{j-1} \in I_t$.

$$\vec{x}(t_j)=\vec{x}(t_{j-1})+\int_{t_{j-1}}^{t_j}\vec{f}(\vec{x}(t))\,dt$$

(47)

The intersection of the variable $x_n(t)$ with the threshold value $x_n = a_e$ is defined by an event function

$$h(\vec{x}(t)) := -a_e + x_n(t) = 0.$$

(48)

Consider the function (48) and the two solutions $\vec{x}_{j-1} := \vec{x}(t_{j-1})$ and $\vec{x}_j := \vec{x}(t_j)$ in system (47). It is assumed, that the continuous function (48) contains a bijective mapping for short interval $\lfloor t_{j-1}, t_j \rfloor \subseteq I_t$. The state event

$$|x_n(t_e)| \in \lfloor |x_n(t_{j-1})|, |x_n(t_{j-1})| + \Delta x_n \rfloor,\ \Delta x_n := |(x_n(t_j) - x_n(t_{j-1}))|$$

exists if and only if there is a $t_e \in \lfloor t_{j-1}, t_j \rfloor$ with $x_{e,n} := x_n(t_e) = a_e$ hence $x_{e,n} \leftrightarrow t_e$.

The Henon’s method defines transformation (50) on ODE system via exchanging the independent variable t with the dependent variable $x_n(t)$. In ODEs (50), the function Γ is given by

$$\Gamma = \frac{1}{f_n\left(x_n, x_1(x_n), \ldots, x_{n\text{-}1}(x_n)\right)}.$$

(49)

Hence x_n is considered as an independent variable and $t(x_n)$ is regarded as a dependent variable [BEK12]. After event detection, the transformed system (50) is given via multiplying all ODEs with Γ except for the equation of x_n which is given as the last ODE. With assumption

$$f_n\left(x_n, x_1(x_n), \ldots, x_{n\text{-}1}(x_n)\right) \neq 0,$$

the last ODE $\frac{d}{dt}x_n(t) = f_n(\vec{x}(t))$ is changed into

$$\frac{d}{dx_n}t(x_n) = \frac{1}{f_n\left(x_n, x_1(x_n), \ldots, x_{n-1}(x_n)\right)} \ .$$

This transformation offers a new system of ODEs as follows

$$\begin{aligned}
&\frac{d}{dx_n}x_1(x_n) = \Gamma\, f_1\left(x_n, x_1(x_n), \ldots, x_{n-1}(x_n)\right) \\
&\frac{d}{dx_n}x_2(x_n) = \Gamma\, f_2\left(x_n, x_1(x_n), \ldots, x_{n-1}(x_n)\right) \\
&\vdots \\
&\frac{d}{dx_n}t(x_n) = \Gamma.
\end{aligned}$$

(50)

The transformed system (50) can be represented by system of ODEs (52), in which the dependent state vector is defined by

$$\vec{\chi}(x_n) := \left(\vec{X}(x_n)^T, t(x_n)\right)^T$$

in which $\vec{\chi} \in \mathbf{R}^n$, $\vec{X} \in \mathbf{R}^{n\text{-}1}$ with $\vec{X} := \{(\vec{x} \mid \vec{x} \setminus x_n)\}$. The independent variable x_n is represented by τ, thus $\tau := x_n$ then $\chi_n(\tau) := t(x_n)$. The functions of ODE system (50) are represented by

$$
\begin{aligned}
&\Theta_1(\tau,\vec{\chi}(\tau)) := \Gamma f_1(\tau,\vec{\chi}(\tau))\\
&\Theta_2(\tau,\vec{\chi}(\tau)) := \Gamma f_2(\tau,\vec{\chi}(\tau))\\
&\quad\vdots\\
&\Theta_n(\tau,\vec{\chi}(\tau)) := \Gamma.
\end{aligned}
$$

(51)

Thus, the system of ODE (50) is given as follows

$$\frac{d}{d\tau}\vec{\chi}(\tau) = \vec{\Theta}(\tau,\vec{\chi}(\tau)).$$

(52)

The state vector $\vec{\chi}(\tau)$ with the independent variable τ is illustrated via initial values in an integral form by (53) for computation of $t_e := t(a_e)$.

$$\vec{\chi}(a_e) = \vec{\chi}_{j-1} + \int_{\tau_{j-1}}^{\tau_{j-1}+|\tau_{j-1}-a_e|} \vec{\Theta}(\tau,\vec{\chi}(\tau))\,d\tau$$

(53)

The solver step-size of (53) is $\Delta\tau = \left|\tau_{j-1}, \tau_{j-1} + |\tau_{j-1} - a_e|\right|$. The integration interval[23] of (53) is

$$\left\lfloor \tau_{j-1}, \tau_{j-1} + |\tau_{j-1} - a_e| \right\rfloor.$$

According to the inverse function theorem, the domain of a one-to-one continuous function is the range of its inverse function and vice versa. Hence, if the range of the integration of $\dot{\chi}_n(\tau)$ by (53) gives the domain of the integration of $\dot{x}_n(t)$ and vice versa, then the initial values of the integration (53) results in the value of the appropriate interval

$$\left\lfloor x_{n,j-1}, x_{n,j-1} + |x_{n,j-1} - a_e| \right\rfloor \leftrightarrow \left[t_{j-1}, t_e\right],$$

---

[23] In some applications, the numerical integrations are carried out in several steps using an adaptive step-size system. The integration is not always satisfied if system is not bijective and has a strong nonlinearity.

thus, the range of the appropriate function of integral of $\chi_n(\tau)$ in (53) results in $t_e := \tilde{t}(a_e)$ as the event location in time.

After computation of the location of the independent variable $t_e := \chi_n(a_e)$, using (53), the transient state vector $\vec{x}_e$ on interval $\lfloor t_{j-1}, t_e \rfloor$ is computed. The computation of the transient state vector at the state event is demonstrated in the integral form as follows

$$\vec{x}(t_e) = \vec{x}_{j-1} + \int_{j-1}^{t_e} \vec{f}(\vec{x}(t))\,dt .$$

(54)

4.2 Henon's Method in System of Semi-Explicit DAEs

This subchapter introduces the Henon's method for the autonomous semi-explicit DAEs in form of

$$\frac{d}{dt}\vec{x}(t) = \vec{f}(\vec{x}(t))$$

$$0 = g(\vec{x}(t)).$$

(55)

The ODE part of the semi-explicit DAEs (55) has the same dimension as the system of ODEs (44).

An auxiliary state variable $x_{n+1}(t)$ is defined equal to the rhs. of the algebraic equation in DAEs (55). The variable $x_{n+1}(t)$ is applied to assess the root of the algebraic equation of (55) by the intersection of

$$x_{n+1}(t) = g(\vec{x}(t))$$

(56)

with $x_{n+1} = a_e$. The function in (56) is defined with the mapping $g : D \to \mathbf{R}$ and $\vec{x} \in D \subset \mathbf{R}^n$. Thus the event function is defined by

$$h(\vec{x}(t)) := -a_e + g(\vec{x}(t)).$$

(57)

The derivation of the event function (57) using the chain rule gives a transformation, which explains its variation with consideration of all dimensions. This transformation is illustrated by differentiation in (58).

$$\frac{d}{dt}h(\vec{x}(t)) = \frac{\partial h}{\partial x_1}(\vec{x}(t))\frac{d}{dt}x_1(t) + \frac{\partial h}{\partial x_2}(\vec{x}(t))\frac{d}{dt}x_2(t) + \cdots + \frac{\partial h}{\partial x_n}(\vec{x}(t))\frac{d}{dt}x_n(t)$$

(58)

The form (58) can be written as

$$\frac{d}{dt}h(\vec{x}(t)) = \left(\overrightarrow{\mathrm{grad}}(h(\vec{x}(t)))\right)^T \vec{f}(\vec{x}(t)).$$

(59)

The derivative of $h(\vec{x}(t))$ is assigned to $f_{n+1}(\vec{x}(t))$ as follows

$$f_{n+1}(\vec{x}(t)) := \frac{d}{dt}h(\vec{x}(t))$$

(60)

thus the corresponding ODE is

$$\frac{d}{dt}x_{n+1}(t) = f_{n+1}(\vec{x}(t)).$$

(61)

The differentiation of $x_{n+1}(t)$ with respect to t is given by the ODE $\dot{x}_{n+1}(t) = f_{n+1}(\vec{x}(t))$ and together with $\dot{\vec{x}}(t) = \vec{f}(\vec{x}(t))$ it is a system of ODEs. Hence, with partial differentiation of the algebraic equation, the system of semi-explicit DAEs (55) is transformed into a system of ODEs (62).

$$\begin{aligned}
&\frac{d}{dt}x_1(t) = f_1(\vec{x}(t)) \\
&\frac{d}{dt}x_2(t) = f_2(\vec{x}(t)) \\
&\quad\vdots \\
&\frac{d}{dt}x_n(t) = f_n(\vec{x}(t)) \\
&\frac{d}{dt}x_{n+1}(t) = f_{n+1}(\vec{x}(t))
\end{aligned}$$

(62)

After crossing the threshold value $x_n = a_e$ by $g(\vec{x}(t))$ via solver algorithm, starts the computation algorithm of the event location. Hence, the system (62) is transformed into a system of ODEs (64) under the consideration of the independent variable t as dependent variable and exchanging t with the dependent variable $x_{n+1}(t)$. Thus, in (64) the variable x_{n+1} is regarded as independent variable. In addition, all equations of ODEs (62) except the last one are multiplied by

$$\Gamma = \frac{1}{f_{n+1}(x_1(x_{n+1}), x_2(x_{n+1})\ldots, x_n(x_{n+1}))}.$$

(63)

The ODE of the state variable $x_{n+1}(t)$ in system of ODEs (62) changes into

$$\frac{d}{dx_{n+1}}t(x_{n+1}) = \frac{1}{f_{n+1}(x_1(x_{n+1}), x_2(x_{n+1})\ldots, x_n(x_{n+1}))}.$$

The whole transformation with assumption

$$f_{n+1}(x_1(x_{n+1}), x_2(x_{n+1})\ldots, x_n(x_{n+1})) \neq 0$$

is represented by the following ODE system

$$\begin{aligned}
&\frac{d}{dx_{n+1}} x_1(x_{n+1}) = \Gamma f_1(x_1(x_{n+1}), x_2(x_{n+1}) \ldots, x_n(x_{n+1})) \\
&\frac{d}{dx_{n+1}} x_2(x_{n+1}) = \Gamma f_2(x_1(x_{n+1}), x_2(x_{n+1}) \ldots, x_n(x_{n+1})) \\
&\quad\vdots \\
&\frac{d}{dx_{n+1}} x_n(x_{n+1}) = \Gamma f_n(x_1(x_{n+1}), x_2(x_{n+1}) \ldots, x_n(x_{n+1})) \\
&\frac{d}{dx_{n+1}} t(x_{n+1}) = \Gamma.
\end{aligned}$$

(64)

The system of ODEs (64) has $n+1$ ODEs and $n+1$ state variables. For clearer description, the state vector of (64) is defined in ODE system (66) by $\vec{\chi}(\tau)$ in which $\vec{\chi} \in \mathbf{R}^{n+1}$ and the independent variable is $\tau := x_{n+1}$. The vector $\vec{\chi}(\tau)$ is given by

$$\vec{\chi}(\tau) := \left(\vec{x}(x_{n+1})^T, t(x_{n+1})\right)^T$$

in which, $\chi_{n+1}(\tau) := t(x_{n+1})$. The vector function of (64) can be given by $\vec{\Theta}(\tau, \vec{\chi}(\tau))$.

$$\begin{aligned}
&\Theta_1(\vec{\chi}(\tau)) := \Gamma f_1(\vec{\chi}(\tau)) \\
&\Theta_2(\vec{\chi}(\tau)) := \Gamma f_2(\vec{\chi}(\tau)) \\
&\quad\vdots \\
&\Theta_{n+1}(\vec{\chi}(\tau)) := \Gamma
\end{aligned}$$

(65)

Then the system of ODEs (64) is established as follows

$$\frac{d}{d\tau} \vec{\chi}(\tau) = \vec{\Theta}(\tau, \vec{\chi}(\tau)).$$

(66)

The integral of $\dot{\chi}_{n+1}(\tau)$ in (66) is considered as inverse function of the function of $x_{n+1}(t)$, which can be interpreted using the differentiation formula of the inverse functions. Thus for computation of the location of the independent variable t_e, the interval $\left[\tau_{j-1}, \tau_{j-1} + \left|\tau_{j-1} - a_e\right|\right]$ is used for integral of (66) as follows

$$\vec{\chi}(a_e) = \vec{\chi}_{j-1} + \int_{\tau_{j-1}}^{\tau_{j-1}+|\tau_{j-1}-a_e|} \vec{\Theta}(\tau, \vec{\chi}(\tau))\,d\tau\,.$$

(67)

After computation of the value $t_e := \tilde{t}(a_e)$ using (67), the approximation of the transient state vector at the state event can be done by the solver algorithm of DAEs (55) for interval $\lfloor t_{j-1}, t_e \rfloor$. The Henon's method can be considered with some extensions[24].

---

[24] The Henon's method may be extended for DAEs (55) with the algebraic equations, in which for each algebraic equation an additional variable can be defined. Then the appropriate ODEs can be found as in (61), hence the result is a system of ODEs and can be solved using the method of subchapter 4.2.
In case of semi-explicit DAEs as presented in (31), the definition of the new variables might not be necessary, if the system has an algebraic variable that is involved in the state event. In this case, the index reduction method can be applied and the system transformation to ODEs and then the Henon's method can be used.
Another consideration is the application of the implicit solver algorithm for approximation of the event location and transient state vector in Henon's method.

5 Implementations of Event Location Approximation Methods in Hybrid Systems of ODEs and DAEs

This chapter presents algorithms, simulations and results of the approximation of the transient vector at the state event for the variable structure filament pendulum and hybrid systems of a bouncing ball and a rotor-stator-system.

In this chapter, the simulations and results of the three subjects of the previous chapters are illustrated. These are the algorithms of the state event handling by reformulating solver formula, reformulating solver formula for the root-finding method and the Henon's method.

5.1 Bouncing Ball with Hybrid System of ODEs

The hybrid model of bouncing ball is known from [MMT04]. The following subchapter contains the presentation and description of the development and results of the simulation algorithms of the bouncing ball hybrid system including approximation algorithms of the transient state vector at the state event.

5.1.1 Model Description

The first part of the simulation algorithm describes the kinematic relations of a falling ball. The two important differential equations are velocity and acceleration. If the ball falls from a height $y(0)$ then the velocity $v(t)$ can be defined as

$$\dot{y}(t) = v(t). \tag{68}$$

The state of the ball acceleration $a(t)$ can be given as:

$$\dot{v}(t) = a(t). \tag{69}$$

The second part of the simulation algorithm is concerned with the gravitational force without drag force for falling objects, which is given by Newton's second law

$$ma(t) = -mg\,.$$

(70)

If a ball falls and hits the ground, then a mass spring damper model can be used to describe the dynamic behaviour of the ball. The contact force of the ball is given by

$$f_e(t) = -k(y(t) - r) + f_d(t)$$

(71)

where k is the spring constant of the ball model and r is the radius of the ball. The damping force term is $f_d(t) = -cv(t)$, in which c is the damping constant. In order to compare simulations of the bouncing ball including system of event location approximations, the bouncing ball model is simplified and the damping characteristic of the ball is eliminated[25]. The simplified contact model of the ball is given by

$$ma(t) = -k(y(t) - r) - mg\,.$$

(72)

The hybrid model is elaborated by conditional ODEs in (73). It is associated with transient conditional relations. The transient conditional relations $y(t) > r$ and $y(t) \leq r$ are presented by event function $h(t) := y(t) - r$ in (73). The first differential equation in (73) is a model for the free-falling ball. The second differential equation is the contact model of the ball.

$$\begin{aligned} m\dot{v}(t) &= -mg && h(t) > 0 \\ m\dot{v}(t) &= -k(y(t) - r) - mg && h(t) \leq 0 \end{aligned}$$

(73)

The state transition depends on the altitude of the centre of the ball. If the altitude of the centre of the ball is lower than the radius of the ball, then the system switches into the spring mass term of the contact model. If the altitude of the ball is larger than the radius of the ball,

---

[25] The simulation of hybrid bouncing ball system with eliminated damping force term results in non-converged periodic solutions. The aim of eliminating the damping force term in the hybrid bouncing ball system is to verify the approximation methods of event locations in quasi-marginal stable state.
The response of each system of the modified hybrid bouncing ball system is diverged if no switching system is used. The convergence of the system responses depends on the simulation model, solver algorithm, solver and system parameters as well as initial values.

then only gravity acts on the ball. Table 1 presents a reset map for the hybrid system of the bouncing ball.

Transition Conditions	**Active States** s
$h(t) > 0$	Free-Falling (Disjunction of Spring Component)
$h(t) \leq 0$	Contacting with the Ground (Junction of Spring Mass Components)

Table 1: Reset Map of Bouncing Ball System

Figure 7 shows a state graph of the hybrid system of the bouncing ball.

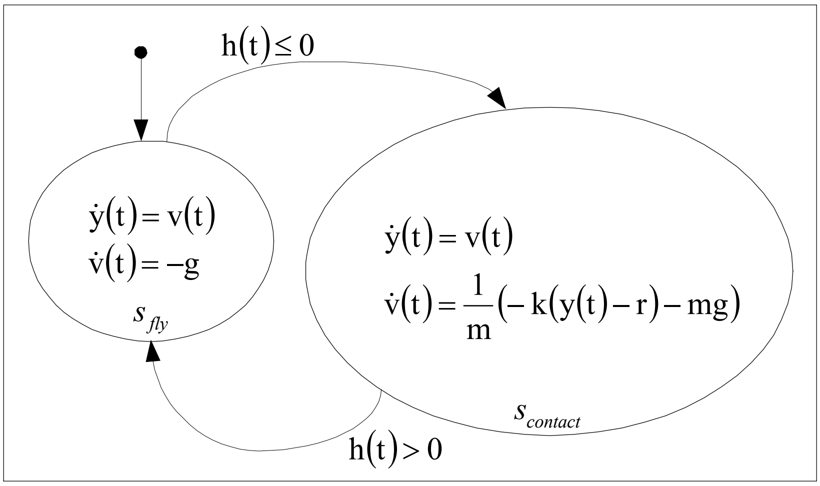

Figure 7: State Graph of Bouncing Ball System

The simulation model of the bouncing ball is represented with the state vector $\vec{x}(t) := (y(t), v(t))^T$ as follows:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -g \end{aligned} \qquad h(t) > 0$$

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= \frac{1}{m}(-k(x_1(t)-r)-mg) \end{aligned} \qquad h(t) \leq 0 .$$

(74)

5.1.2 Reformulation of Explicit Euler’s Formula

The simulation of the bouncing ball model is realized by implementing the Euler’s solver for the hybrid system (74) as follows

$$\begin{aligned} \hat{x}_{1,j} &= \hat{x}_{1,j-1} + \Delta t \hat{x}_{2,j-1} \\ \hat{x}_{2,j} &= \hat{x}_{2,j-1} + \Delta t(-g) \end{aligned} \qquad h(t) > 0$$

$$\begin{aligned} \hat{x}_{1,j} &= \hat{x}_{1,j-1} + \Delta t \hat{x}_{2,j-1} \\ \hat{x}_{2,j} &= \hat{x}_{2,j-1} + \Delta t\left(\frac{1}{m}\left(-k\left(\hat{x}_{1,j-1} - r\right) - mg\right)\right) \end{aligned} \qquad h(t) \leq 0.$$

(75)

The nominal ball radius r is considered as the threshold value $a_e := r$. An event is defined by event function $h(t_j) := -r + \hat{x}_{1,j} = 0$ as intersection of the approximated ball altitude $\hat{x}_{1,j}$ with $x_1 = r$ at $t_j := \tilde{t}_e \approx t_e$.

$$h(\tilde{t}_e) = -r + \hat{x}_{1,j-1} + (\tilde{t}_e - t_{j-1})\hat{x}_{2,j-1} = 0$$

(76)

The computation of the location of t_e in the time domain is determined by $h(\tilde{t}_e) = 0$ according to formula (15) for a transition from falling to bouncing and vice versa.

$$\tilde{t}_e = t_{j-1} + \left| \frac{r - \hat{x}_{1j-1}}{\hat{x}_{2j-1}} \right|$$

(77)

After computation of formula (77), the approximation of the transient state vector is evaluated according to explicit Euler’s solver (16) by means of $\Delta\tilde{t}_e = \tilde{t}_e - t_{j-1}$ applied to numerical hybrid system (75).

5.1.3 Reformulation of Implicit Trapezodial Formula for Newton's Method

The simulation algorithm in this part is implemented according to the subjects of subchapter 2.2.2. For simulation, the trapezoidal solver is implemented with two prediction and approximation steps. The prediction step is developed via Newton's method applied on reformulated trapezoidal solver formula. The approximation step of the state vector is realized by initializing of the solver algorithm with the predicted state vector.

The computation of prediction and approximation steps of the state vector continues until detection of a state event. Then the simulation algorithm switches to the approximation algorithm of the transient state vector at state event. The prediction of the transient state vector at state event is accomplished by Newton's method with iteration index k. In event vector functions (78), the state variable $x_{1,j}$ is set to the threshold value $x = a_e := r$, $\hat{x}_{1,j} = r$ and the variables $\Delta\tilde{t}_{e,k}$ and $\tilde{x}_{2,k}$ are demanded variables.

$$\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k}, \tilde{x}_{2,k}\right) = \begin{pmatrix} r - \hat{x}_{1,j-1} - \dfrac{\Delta\tilde{t}_{e,k}}{2}\left(\tilde{x}_{2,k} + \hat{x}_{2,j-1}\right) \\ \tilde{x}_{2,k} - \hat{x}_{2,j-1} - \dfrac{\Delta\tilde{t}_{e,k}}{2}\left(-2g\right) \end{pmatrix} \qquad h(t) > 0$$

$$\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k}, \tilde{x}_{2,k}\right) = \begin{pmatrix} r - \hat{x}_{1,j-1} - \dfrac{\Delta\tilde{t}_{e,k}}{2}\left(\tilde{x}_{2,k} + \hat{x}_{2,j-1}\right) \\ \tilde{x}_{2,k} - \hat{x}_{2,j-1} - \dfrac{\Delta\tilde{t}_{e,k}}{2}\left[\left(-g\right) + \left(\dfrac{1}{m}\left(-k\left(\hat{x}_{1,j-1} - r\right) - m \cdot g\right)\right)\right] \end{pmatrix} \qquad h(t) \leq 0$$

(78)

After computation of the prediction values according to Newton's method (28), the trapezoidal solver (29), which is realized for bouncing ball model, is reinitialized with $\tilde{\vec{x}}_e := \left(r, \tilde{x}_{2,k}\right)^T$ and $\Delta\tilde{t}_e := \Delta\tilde{t}_{e,k}$ and runs for one step calculation to approximate the transient state vector $\hat{\vec{x}}_e$ at the state event.

5.1.4 Henon's Method

In order to simulate Henon's transformation, the function Γ is defined as $\Gamma = \dfrac{1}{x_2(x_1)}$. The Henon's transformation for this simulation is shown as follows:

$$\begin{aligned} &\frac{d}{dx_1} t(x_1) = \frac{1}{x_2(x_1)} \\ &\frac{d}{dx_1} x_2(x_1) = \frac{-g}{x_2(x_1)} \end{aligned} \qquad h(t) > 0$$

$$\begin{aligned} &\frac{d}{dx_1} t(x_1) = \frac{1}{x_2(x_1)} \\ &\frac{d}{dx_1} x_2(x_1) = \frac{1}{m} \frac{(-k(x_1 - r) - mg)}{x_2(x_1)} \end{aligned} \qquad h(t) \le 0 .$$

(79)

Via setting $\vec{\chi}(\tau) := (t(\tau), x_2(\tau))^T$ and $\tau := x_1$, the systems of ODEs in (79) can be represented via a system of ODEs (52). The computation of $\tilde{t}_e$ is achieved for the selected system using ODE45 according to formula (53) on the interval $\lfloor \hat{\tau}_{j-1}, \hat{\tau}_{j-1} + |\hat{\tau}_{j-1} - r| \rfloor$.

Then the solver algorithm is reinitialized according to the initial values of (54). The approximation of the transient state vector $\hat{\vec{x}}(\tilde{t}_e)$ is realized by ODE45 solver.

5.1.5 Simulation Comparisions

The hypothetical simulation parameters are chosen in SI units as demonstrated in Table 2.

Parameter	**Description**
$h_0 := 5m$	initial ball position
$v_0 := -2.5\,m/s$	initial velocity
$m := 200kg$	mass of ball
$r := 2.5m$	radius of ball
$k := 1.0e + 4\,N/m$	spring constant
$g := 9.80665\,m/s^2$	acceleration of gravity of earth's surface

Table 2: Bouncing Ball Simulation Parameters

The approximated solutions and trajectory of the modified bouncing ball hybrid system are demonstrated in Figures 8 and 9.

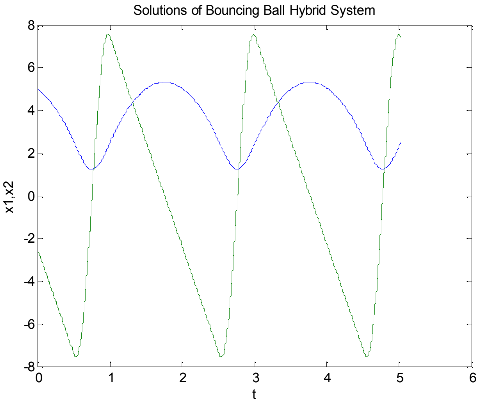

Figure 8: Approximated Solutions of Modified Bouncing Ball Hybrid System

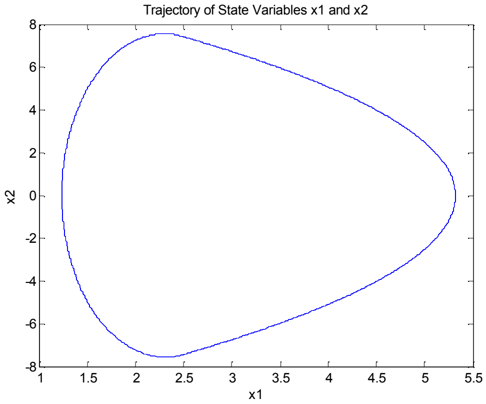

Figure 9: Trajectory of State Variables of Modified Bouncing Ball Hybrid System

Table 3 illustrates the simulation results approximating of the ball location at the state event using reformulation of explicit Euler's formula, reformulation of implicit trapezoidal formula for Newton's root-finding, Henon and Matlab adaptive methods[26].

The modified hybrid bouncing ball system without damping force term and without approximation methods of event location is simulated via Euler's solver with different simulators. These simulations show that using Euler's solver produces an instability in system responses with increasing the peaks of ball altitude $h(t)$, if the damping term is eliminated from bouncing ball model.

In spite of the fact that instability produces in response of system, the approximation method "reformulation of Euler's solver formula" in Table 4 shows a relative small magnitude for the local error of the approximated state variable at the switching event.

An additional application is programmed based on Simulink without zero-crossing algorithm.

---

[26] The adaptive zero-crossing reduces the step-size of solver iterations, until the setting values of the Matlab/Simulink configuration parameters are exceeded [MHV10].

Methods	**Reformulation of Explicit Euler's Formula**	**Reformulation of Implicit Trapezoidal Formula for Newton's Method**	**Henon**
Solver Algorithm	Implemented Euler's Solver	Implemented Trapezoidal Solver Mixed Newton Method	Programmed Henon Using Matlab ODE45
Type	Explicit	Implicit	Explicit
Programming Environment	Matlab	Matlab	Matlab
Threshold Value a_e	2.5	2.5	2.5
$x_1(\tilde{t}_e)$	2.500000	2.500000	2.499999
Local Error: $\left\|\hat{x}_1(\tilde{t}_e) - r\right\|$	0	0	5.394569e-10
Fixed Step-Size	1e-05	1e-05	1e-05
$\Delta\tilde{\iota}_e$	8.710344e-07	7.552201e-06	7.416839e-06
$\tilde{t}_e$	5.032600e-01	5.032575e-01	5.032574e-01

Methods	**Simulink**	**Adaptive (Matlab)**
Solver Algorithm	Simulink ODE4	Matlab ODE45
Type	Explicit	Explicit
Programming Environment	Simulink	Matlab
Threshold Value a_e	2.5	2.5
$x_1(\tilde{t}_e)$	[2.499981, 2.500056]	2.499999
Local Error: $\left\|\hat{x}_1(\tilde{t}_e) - r\right\|$	-	8.215650e-14
Fixed Step-Size	1e-05	1e-05
$\Delta\tilde{\iota}_e$	-	7.552201e-06
$\tilde{t}_e$	[5.0325e-01, 5.0326e-01]	5.032575e-01

Table 3: First Event Comparisons of Bouncing Ball Simulations at a Transition from Free-Fall to Bouncing State

The approximated values of the state event $x_1(t_e)$ in Table 3 show that the reformulation of implicit trapezoidal formula for root-finding method has an accurate magnitude regarding the event threshold value $a_e = 2.5$.

The simulation via reformulation of solver formula is coded according to the contents of subchapters 2.1.1 and 5.1.2. The root-finding method is programmed according to the developed method in subchapters 2.2.2 and 5.1.3. The Henon's method is programmed in Matlab, matches Matlab ODE45 and is implemented according to subchapters 4.1 and 5.1.4. Two other simulations are realized, one in Simulink without zero-crossing setting and the other in Matlab using ODE45 with adaptive method.

Table 4 demonstrates the state event approximations of $x_1(\tilde{t}_e)$ for different methods at the various events.

Henon's method is realized using ODE45 feature in Matlab environment. The system response in this implementation shows a difficulty[27] evaluating the eleventh state event and the magnitudes of $\vec{\chi}(\tau)$ show an unstable behavior.

The reformulation of implicit trapezoidal formula for Newton's root-finding method has a stable procedure and shows low local errors.

	Methods			
Error $\lvert x_1(\tilde{t}_e) - r \rvert$	**Reformulation of Explicit Euler's Formula**	**Reformulation of Implicit Trapezoidal Formula for Newton's Method**	**Henon**	**Adaptive Matlab**
Error at First Event	0	0	5.394569e-10	8.215650e-14
Error at Fifth Event	3.108624e-15	0	3.033617e-10	4.138911e-13
Error at Tenth Event	8.881784e-16	0	1.524347e-08	9.099387e-13
Error at Twentieth Event	1.199040e-14	0	-	1.498801e-12

Table 4: Errors of Approximated Magnitudes of State Variable $x_1(t)$ at Various State Events in Hybrid System of the Bouncing Ball

---

[27] At the eleventh state event, the approximated magnitudes of $\chi_2(\tau) := x_2(x_1)$ become close to zero where the magnitudes of $\frac{d}{d\tau}\chi_1(\tau) = \frac{1}{\chi_2(\tau)}$ increase periodically and cause the instability in system responses and consequently in system simulation.

5.2 Filament Pendulum with Variable Structure System of DAEs

The model of filament pendulum is inspired by [BEC08] and [CGN07]. Two models establish the variable structure pendulum: one is the mathematical pendulum and the other is the free-fall model.

This subchapter deals with modelling of event handling of the variable structure system of the filament pendulum as well as development and implementation of the simulation algorithms for approximation of the transient vector.

5.2.1 Model Description

The structure of the filament pendulum system may be changed from a filament pendulum to a free-fall system and vice versa, if certain conditions are fulfilled. A pendulum changes its position if it is not at its equilibrium state. The motion conditions of a pendulum can be fulfilled by choosing certain initial values, which can cause the pendulum to launch out of its equilibrium state.

The angular velocity of the pendulum is expressed by

$$\omega_p(t) = \frac{d}{dt}\varphi_p(t). \tag{80}$$

The length of the arc of the angle $\varphi_p(t)$ is defined as

$$s(t) = L \cdot \varphi_p(t). \tag{81}$$

The tangential velocity can be found from the derivative of the arc of the angle with respect to time:

$$v_t(t)=\frac{d}{dt}s(t)=L\frac{d}{dt}\varphi_p(t)=L\omega_p(t).$$

(82)

The tangential acceleration of the pendulum is defined by

$$a_t(t)=\frac{d^2}{dt^2}s(t)=L\frac{d^2}{dt^2}\varphi_p(t).$$

(83)

The state of the angular acceleration without damping component is

$$mL^2\frac{d^2}{dt^2}\varphi_p(t)=mgL\sin(\varphi_p(t)).$$

(84)

The inertia moment for a filament pendulum is

$$J = mL^2$$

(85)

and using differential equation (84) substituting mL^2 with J one gets

$$J\frac{d^2}{dt^2}\varphi_p(t)-mgL\sin(\varphi_p(t))=0.$$

(86)

The moment relation including the damping term can be amended in the following form:

$$J\frac{d^2}{dt^2}\varphi_p(t)-mgL\sin(\varphi_p(t))+c_p\frac{d}{dt}\varphi_p(t)=0.$$

(87)

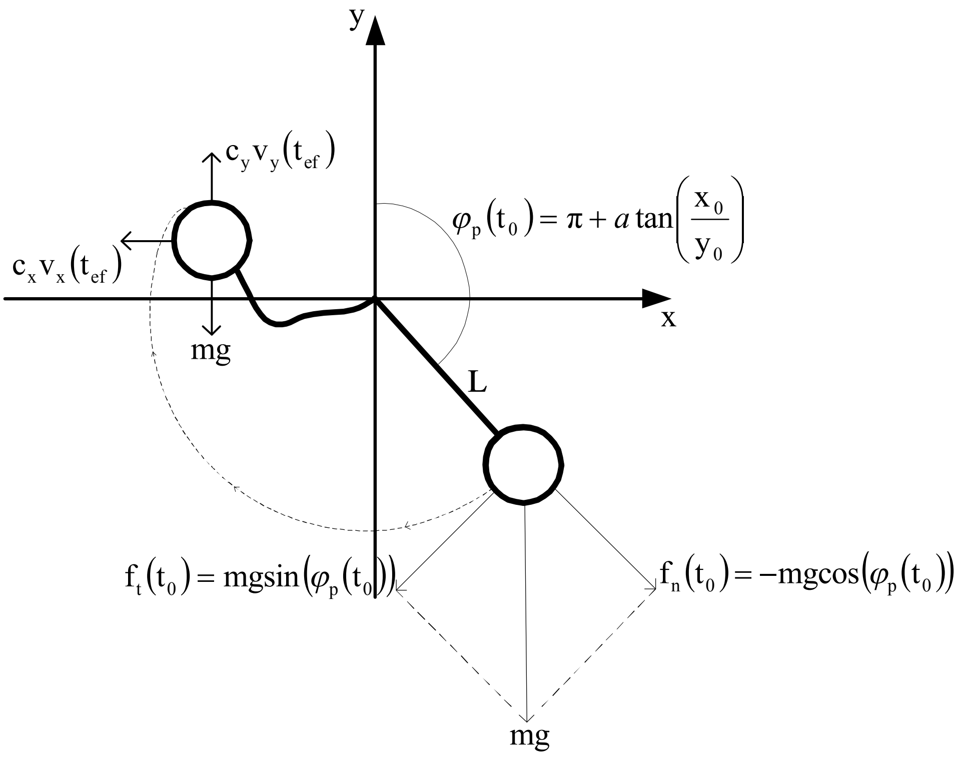
y
x
$c_y v_y(t_{ef})$
$c_x v_x(t_{ef})$
mg
$\varphi_p(t_0) = \pi + a\tan\left(\frac{x_0}{y_0}\right)$
L
$f_t(t_0) = mg\sin(\varphi_p(t_0))$
$f_n(t_0) = -mg\cos(\varphi_p(t_0))$
mg

Figure 10: Forces in Free-Fall and Mathematical Pendulum Systems

To leave the pendulum state and switch to the free-fall state, a tension force condition of the filament must be fulfilled. Hence, if the filament loses its tension and the pendulum leaves its track, then the simulation process must be switched from the pendulum structure to the free-fall structure. The tension force of the filament $F_a(t)$ can be obtained from the radial force. The radial acceleration is given as:

$$a_r(t) = L\omega_p^2(t). \tag{88}$$

The radial force of the circular motion is

$$F_a(t) = -W\cos(\varphi_p(t)) + ma_r(t), \tag{89}$$

thus, a criterion of the model switching is defined by the tension of filament $F_a(t)$ which can be calculated using (90).

$$F_a(t) = -mg\cos\varphi_p(t) + mL\omega_p^2(t)$$

(90)

If the filament pendulum aborts its harmonic oscillation, then three cases can be assumed. Either the pendulum leaves its orbit in an outward[28] direction, stays within its pendulum radius[29] or reaches its equilibrium state.

If the tension of filament $F_a(t)$ is vanished, then the path of the pendulum motion has to change to the free-fall path. This means that the ODEs of free-fall motion are valid, therefore the velocity along the x-axis in free-fall system is defined by:

$$\dot{x}(t) = v_x(t).$$

(91)

The force equation can be given with the mass acceleration and damping model:

$$m\dot{v}_x(t) = ma_x(t) = -c_x v_x(t).$$

(92)

The velocity along the y-axis is represented by

$$\dot{y}(t) = v_y(t).$$

(93)

The force equation in y-direction is explained by

$$m\dot{v}_y(t) = ma_y(t) = -W - c_y v_y(t) = -mg - c_y v_y(t).$$

(94)

---

[28] This case can be occurred if the pendulum starts to move with high magnitude of radial acceleration, which induces a high magnitude of centrifugal force and the filament is ripped.

[29] A transition from free-fall to pendulum state can occur with a bouncing characteristic. The bouncing behavior is not considered in the simulation model and in a transition from free-fall to pendulum state, an inelastic behavior of filament pendulum is assumed.

The damping force $c_y v_y(t)$ in differential equation (94) has a negative sign. Another term is the weight force mg, which also has a negative sign.

A criterion for a transition of the simulation data processing from the free-fall to the pendulum algorithm can be given by the distance between pendulum and centre.

$$r(t) = \sqrt{x^2(t) + y^2(t)} \tag{95}$$

An automated event handling[30] is controlled by a switching system. The switching possibilities are shown in Table 5. The state transitions are defined by the states of $F_a(t)$ and $r(t)$. In each system, the appropriate condition triggers the system transition. Table 5 shows the events using two logical variables $q_{ef}(t)$ and $q_{ep}(t)$. The logical variables are defined as

$$q_{ep}(t) := \begin{cases} T & r(t) = L \\ F & r(t) < L \end{cases}$$

and

$$q_{ef}(t) := \begin{cases} T & F_a(t) = 0 \\ F & F_a(t) > 0. \end{cases}$$

Table 5 describes that for a transition from the pendulum to the free-fall state, only condition

$$h_{ef}(t) := F_a(t) = -mg\cos\varphi_p(t) + mL\omega_p^2(t) = 0$$

has to be valid and for a transition from the free-fall to the pendulum state, only the condition

$$h_{ep}(t) := r(t) - L = \left|\sqrt{x^2(t) + y^2(t)}\right| - L = 0$$

has to be true.

---

[30] Appendix A1 shows the non-automated event-handling modus. In this case, separate simulations run consecutively.

Transition Conditions	Logical Variables		State Transition	
	$q_{\mathrm{ef}}(\mathrm{t})$	$q_{\mathrm{ep}}(\mathrm{t})$	**Previous State**	**Active State**
$\mathrm{h}_{\mathrm{ef}}(\mathrm{t})=0$	*T*	*Not Used*	Pendulum System	Free-Fall System
$\mathrm{h}_{\mathrm{ep}}(\mathrm{t})=0$	*Not Used*	*T*	Free-Fall System	Pendulum System

Table 5: Reset Map of Variable Structure Filament Pendulum

The rhs. of Figure 11 shows the free-fall coordinate system. The system transition from pendulum to free-fall occurs, if $q_{\mathrm{ef}}(\mathrm{t})$ is true for

$$\mathrm{t}_{\mathrm{ef}} : \left(\mathrm{t}_{\mathrm{ef}} \leftrightarrow \left(\mathrm{h}_{\mathrm{ef}}(\mathrm{t}) = \text{-}\mathrm{mg}\cos\varphi_{\mathrm{p}}(\mathrm{t}) + \mathrm{mL}\omega_{\mathrm{p}}^{2}(\mathrm{t}) = 0\right)\right),$$

then the initial values at the free-fall event are defined as

$$\begin{aligned}
&\mathrm{x}\left(\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}})\right) := \mathrm{L}\sin\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}}) \\
&\mathrm{y}\left(\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}})\right) := \mathrm{L}\cos\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}}) \\
&\mathrm{v}_{\mathrm{x}}(\mathrm{t}_{\mathrm{ef}}) := \frac{\mathrm{d}}{\mathrm{dt}}\mathrm{x}\left(\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}})\right) = \frac{\partial \mathrm{x}}{\partial \varphi}\left(\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}})\right)\frac{\mathrm{d}}{\mathrm{dt}}\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}}) = \mathrm{L}\omega_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}})\cos\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}}) \\
&\mathrm{v}_{\mathrm{y}}(\mathrm{t}_{\mathrm{ef}}) := \frac{\mathrm{d}}{\mathrm{dt}}\mathrm{y}\left(\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}})\right) = \frac{\partial \mathrm{y}}{\partial \varphi}\left(\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}})\right)\frac{\mathrm{d}}{\mathrm{dt}}\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}}) = -\mathrm{L}\omega_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}})\sin\varphi_{\mathrm{p}}(\mathrm{t}_{\mathrm{ef}}).
\end{aligned}$$

(96)

The left part of Figure 11 shows the pendulum system. When $q_{\mathrm{ep}}(\mathrm{t})$ becomes true then the pendulum differential equations should be initialized at

$$\mathrm{t}_{\mathrm{ep}} : \left(\mathrm{t}_{\mathrm{ep}} \leftrightarrow \left(\mathrm{r}(\mathrm{t}) - \mathrm{L}\right) = 0\right)$$

with the following initial values:

$$\lambda(t_{ep}) := \frac{x(t_{ep})}{y(t_{ep})}$$

$$\varphi_p(t_{ep}) := \pi + \mathrm{atan}(\lambda(t_{ep}))$$

$$\omega_p(t_{ep}) := \frac{\partial\varphi_p}{\partial\lambda}(t_{ep})\frac{\partial\lambda}{\partial x}(t_{ep})\frac{d}{dt}x(t_{ep}) + \frac{\partial\varphi_p}{\partial\lambda}(t_{ep})\frac{\partial\lambda}{\partial y}(t_{ep})\frac{d}{dt}y(t_{ep}) = \frac{1}{r(t_{ep})^2}\left(y(t_{ep})v_x(t_{ep}) - x(t_{ep})v_y(t_{ep})\right).$$

(97)

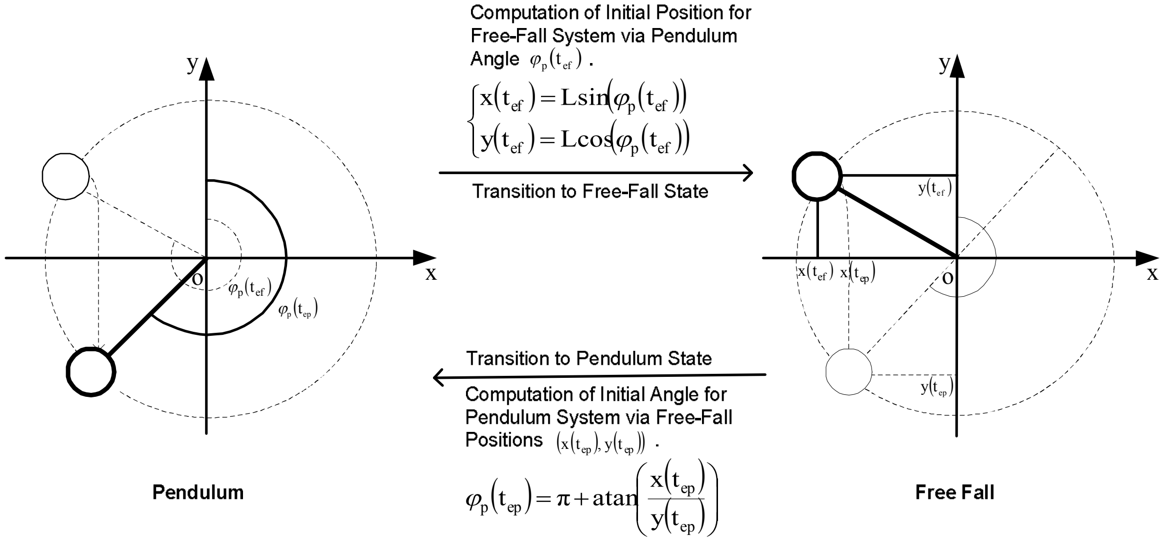
Computation of Initial Position for Free-Fall System via Pendulum Angle $\varphi_p(t_{ef})$.
$\begin{cases} x(t_{ef}) = L\sin(\varphi_p(t_{ef})) \\ y(t_{ef}) = L\cos(\varphi_p(t_{ef})) \end{cases}$
Transition to Free-Fall State
Transition to Pendulum State
Computation of Initial Angle for Pendulum System via Free-Fall Positions $(x(t_{ep}), y(t_{ep}))$.
$\varphi_p(t_{ep}) = \pi + \mathrm{atan}\left(\frac{x(t_{ep})}{y(t_{ep})}\right)$
y
x
o
$\varphi_p(t_{ef})$
$\varphi_p(t_{ep})$
$y(t_{ef})$
$x(t_{ef})$
$x(t_{ep})$
$y(t_{ep})$
Pendulum
Free Fall

Figure 11: Initializations at State Transitions for Free-Fall and Pendulum Systems[31]

The initial values of the state vector at an event should be prepared automatically before starting data processing of the next system. The simulation model of the filament pendulum is completed with the initialization terms. The supplemented variables of structure filament pendulum are shown in Figure 12.

---

[31] The system transitions are given by two different transformations. The position of an event for a transition from free-fall to pendulum is in the low-half part of the coordinate system and for a transition from pendulum to free-fall it is in the high half part of the coordinate system.

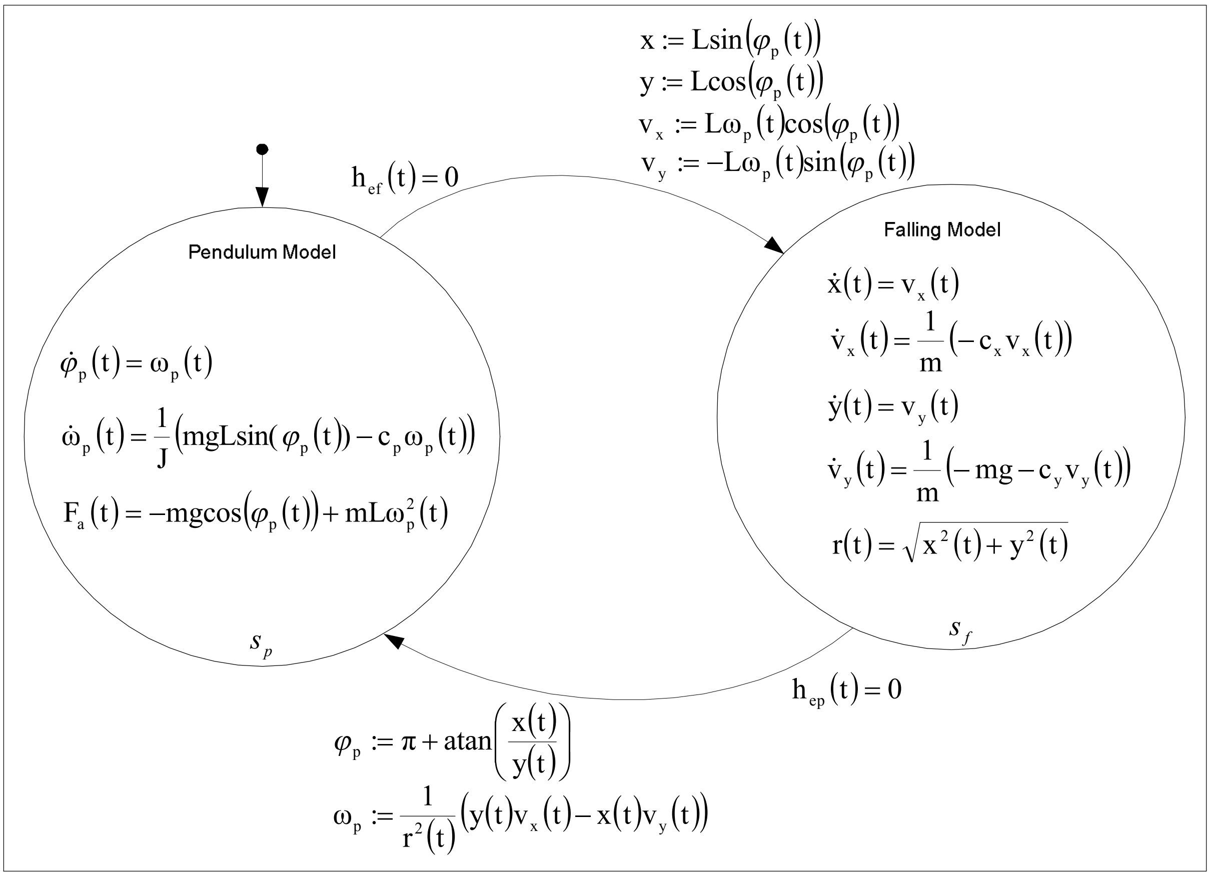
$x := L\sin(\varphi_p(t))$
$y := L\cos(\varphi_p(t))$
$v_x := L\omega_p(t)\cos(\varphi_p(t))$
$v_y := -L\omega_p(t)\sin(\varphi_p(t))$
$h_{ef}(t) = 0$
Pendulum Model
$\dot{\varphi}_p(t) = \omega_p(t)$
$\dot{\omega}_p(t) = \frac{1}{J}(mgL\sin(\varphi_p(t)) - c_p\omega_p(t))$
$F_a(t) = -mg\cos(\varphi_p(t)) + mL\omega_p^2(t)$
s_p
Falling Model
$\dot{x}(t) = v_x(t)$
$\dot{v}_x(t) = \frac{1}{m}(-c_x v_x(t))$
$\dot{y}(t) = v_y(t)$
$\dot{v}_y(t) = \frac{1}{m}(-mg - c_y v_y(t))$
$r(t) = \sqrt{x^2(t) + y^2(t)}$
s_f
$h_{ep}(t) = 0$
$\varphi_p := \pi + \mathrm{atan}\left(\frac{x(t)}{y(t)}\right)$
$\omega_p := \frac{1}{r^2(t)}(y(t)v_x(t) - x(t)v_y(t))$

Figure 12: State Graph of Variable Structure System of Semi-Explicit DAEs for Filament Pendulum

For simplification, the state vector for free-fall is defined by

$$\vec{x}(t) = (x_1(t), x_2(t), x_3(t), x_4(t))^T := (x(t), v_x(t), y(t), v_y(t))^T$$

and the algebraic variable is defined by $y_1(t) := r(t)$. For the pendulum model the state vector is defined by

$$\vec{x}(t) = (x_1(t), x_2(t))^T := (\varphi_p(t), \omega_p(t))^T$$

and the algebraic variable is $y_1(t) := F_a(t)$.

So, the variable structure system of semi-explicit DAEs can be given as follows

$$
\begin{array}{ll}
\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= \frac{1}{m}\left(-c_x x_2(t)\right) \\
\dot{x}_3(t) &= x_4(t) \\
\dot{x}_4(t) &= \frac{1}{m}\left(-mg - c_y x_4(t)\right) \\
y_1(t) &= \sqrt{x_1^2(t) + x_3^2(t)}
\end{aligned} & h_{ef}(t) = 0 \\
\\
\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= \frac{1}{J}\left(mgL\sin\left(x_1(t)\right) - c_p{}' x_2(t)\right) \\
y_1(t) &= -mg\cos\left(x_1(t)\right) + mLx_2^2(t)
\end{aligned} & h_{ep}(t) = 0\,.
\end{array}
\tag{98}
$$

5.2.2 Reformulation of Explicit Euler's Formula

The state event handling of a variable structure system of semi-explicit DAEs requires an index-reduction procedure for transformation DAEs into systems of ODEs. This transformation is explained in subchapter 3.1. Hence, the radius of the pendulum (95) and the tension of filament (90) are differentiated by the chain rule with respect to the independent variable t. The differentiation of $y_1(t)$ for the free-fall system with respect to t is represented by

$$
\frac{dy_1(t)}{dt} = \frac{\partial y_1}{\partial x_1}\left(x_1(t), x_3(t)\right)\frac{dx_1(t)}{dt} + \frac{\partial y_1}{\partial x_3}\left(x_1(t), x_3(t)\right)\frac{dx_3(t)}{dt} = \frac{\left(x_1(t)x_2(t) + x_3(t)x_4(t)\right)}{\sqrt{x_1^2(t) + x_3^2(t)}}
\tag{99}
$$

and the differentiation of $y_1(t)$ for pendulum system with respect to t is

$$
\begin{aligned}
\frac{dy_1(t)}{dt} &= \frac{\partial y_1}{\partial x_1}\left(x_1(t), x_2(t)\right)\frac{dx_1(t)}{dt} + \frac{\partial y_1}{\partial x_2}\left(x_1(t), x_2(t)\right)\frac{dx_2(t)}{dt} \\
&= mg\sin\left(x_1(t)\right)x_2(t) + \frac{2mLx_2(t)}{J}\left(mgL\sin\left(x_1(t)\right) - c_p x_2(t)\right).
\end{aligned}
\tag{100}
$$

The semi-explicit systems of DAEs (98) are transformed to the systems of ODEs as follows:

$$
\begin{aligned}
&\dot{x}_1(t)=x_2(t)\\
&\dot{x}_2(t)=\frac{1}{m}\left(-c_x x_2(t)\right)\\
&\dot{x}_3(t)=x_4(t) && h_{ef}(t)=0\\
&\dot{x}_4(t)=\frac{1}{m}\left(-mg-c_y x_4(t)\right)\\
&\dot{y}_1(t)=\frac{x_1(t)x_2(t)+x_3(t)x_4(t)}{\sqrt{x_1^2(t)+x_3^2(t)}}
\end{aligned}
$$

$$
\begin{aligned}
&\dot{x}_1(t)=x_2(t)\\
&\dot{x}_2(t)=\frac{1}{J}\left(mgL\sin\left(x_1(t)\right)-c_p x_2(t)\right) && h_{ep}(t)=0\,.\\
&\dot{y}_1(t)=mg\sin\left(x_1(t)\right)x_2(t)+\frac{2mLx_2(t)}{J}\left(mgL\sin\left(x_1(t)\right)-c_p x_2(t)\right)
\end{aligned}
$$

(101)

The computation of the location t_e for a transition from free-fall to pendulum state is realized using $\hat{y}_{1,j}=a_e:=L$ by solving the reformulated explicit Euler's equation of ODE (99) according to formula (15).

$$\widetilde{t}_e=t_{j-1}+\left|\frac{\left(L-\hat{y}_{1j-1}\right)\sqrt{\hat{x}_{1,j-1}^2+\hat{x}_{3,j-1}^2}}{\hat{x}_{1,j-1}\hat{x}_{2,j-1}+\hat{x}_{3,j-1}\hat{x}_{4,j-1}}\right|$$

(102)

The computation of event location t_e along t for a state transition from pendulum to free-fall is done by solving the reformulated explicit Euler's equation of ODE (100) with respect to $a_e:=0$ and $\hat{y}_{1,j}=a_e$ as follows

$$\widetilde{t}_e=t_{j-1}+\left|\frac{\hat{y}_{1j-1}}{mg\sin\left(\hat{x}_{1,j-1}\right)\hat{x}_{2,j-1}+\frac{2mL\hat{x}_{2,j-1}}{J}\left(mgL\sin\left(\hat{x}_{1,j-1}\right)-c_p\hat{x}_{2,j-1}\right)}\right|.$$

(103)

The transient state vector can be approximated via Euler's method with the computed event step-size $\Delta\tilde{t}_e := \tilde{t}_e - t_{j-1}$ and the appropriate initial values.

5.2.3 Reformulation of Implicit Trapezodial Formula for Newton's Method

In this subchapter, the trapezoidal solver with prediction and approximation phases is used as an implicit solver. The variable structure system of semi-explicit DAEs (98) is taken in its original form without index-reduction. The root-finding method is applied two times for each system, once for computing the state vectors in prediction correction routine in the implicit trapezoidal solver algorithm until an event is detected, and once to predict the transient vector at the state event in the event location approximation algorithm after the event detection.

For the prediction of the state event location by the root-finding method, it is assumed that the variable $\tilde{y}_{1,j}$ reaches its appropriate threshold values at step j. Hence, at state event $\tilde{y}_{1,j}$ is considered either as constant value L for free-fall to pendulum transition or as zero in case of pendulum to free-fall transition. The unknown state variables at step j are predicted by the Newton's method with iteration index k at the event location environment. In the root-finding iteration algorithm, the other variables with index $j-1$ are considered as constant known values. The hybrid event vector functions for the variable structure filament pendulum are shown as follows

$$\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k},\tilde{\vec{x}}_k\right)=\begin{pmatrix} \tilde{x}_{1,k}-\hat{x}_{1,j-1}-\dfrac{\Delta\tilde{t}_k}{2}\left(\hat{x}_{2,j-1}+\tilde{x}_{2,k}\right) \\ \tilde{x}_{2,k}-\hat{x}_{2,j-1}+\dfrac{\Delta\tilde{t}_k c_x}{2m}\left(\hat{x}_{2,j-1}+\tilde{x}_{2,k}\right) \\ \tilde{x}_{3,k}-\hat{x}_{3,j-1}-\dfrac{\Delta\tilde{t}_k}{2}\left(\hat{x}_{4,j-1}+\tilde{x}_{4,k}\right) \\ \tilde{x}_{4,k}-\hat{x}_{4,j-1}+\dfrac{\Delta\tilde{t}_k}{2m}\left(2mg+c_y\left(\hat{x}_{4,j-1}+\tilde{x}_{4,k}\right)\right) \\ L-\sqrt{\tilde{x}_{1,k}^2+\tilde{x}_{3,k}^2} \end{pmatrix}=\vec{0} \qquad h_{ef}(t)=0$$

$$\hat{\vec{h}}\left(\Delta\tilde{t}_{e,k},\tilde{\vec{x}}_k\right)=\begin{pmatrix} \tilde{x}_{1,k}-\hat{x}_{1,j-1}-\dfrac{\Delta\tilde{t}_k}{2}\left(\hat{x}_{2,j-1}+\tilde{x}_{2,k}\right) \\ \tilde{x}_{2,k}-\hat{x}_{2,j-1}-\dfrac{\Delta\tilde{t}_k}{2J}\left(mgL\left(\sin\left(\hat{x}_{1,j-1}\right)+\sin\left(\tilde{x}_{1,k}\right)\right)-c_p\left(\hat{x}_{2,j-1}+\tilde{x}_{2,k}\right)\right) \\ mg\cos\left(\tilde{x}_{1,k}\right)-mL\tilde{x}_{2,k}^2 \end{pmatrix}=\vec{0} \qquad h_{ep}(t)=0.$$

(104)

The step-size and variable predictions at the state event are realized for each domain according to Newton’s method for nonlinear systems of equations (38) using event vector function (104). The transient vectors are approximated by the system of implicit trapezoidal solvers (39).

5.2.4 Henon’s Method

The Henon transformation is realized via systems of ODEs (105) according to Henon’s transformation (64).

$$
\begin{aligned}
&\frac{d}{dy_1}x_1(y_1)=\frac{x_2(y_1)\sqrt{x_1^2(y_1)+x_3^2(y_1)}}{x_1(y_1)x_2(y_1)+x_3(y_1)x_4(y_1)}\\
&\frac{d}{dy_1}x_2(y_1)=\frac{-c_x x_2(y_1)\sqrt{x_1^2(y_1)+x_3^2(y_1)}}{mx_1((y_1)x_2(y_1)+x_3(y_1)x_4(y_1))}\\
&\frac{d}{dy_1}x_3(y_1)=\frac{x_4(y_1)\sqrt{x_1^2(y_1)+x_3^2(y_1)}}{x_1(y_1)x_2(y_1)+x_3(y_1)x_4(y_1)} \qquad\qquad h_{ef}(t)=0\\
&\frac{d}{dy_1}x_4(y_1)=\frac{-(mg+c_y x_4(y_1))\sqrt{x_1^2(y_1)+x_3^2(y_1)}}{m((y_1)x_2(y_1)+x_3(y_1)x_4(y_1))}\\
&\frac{d}{dy_1}t(y_1)=\frac{\sqrt{x_1^2(y_1)+x_3^2(y_1)}}{x_1(y_1)x_2(y_1)+x_3(y_1)x_4(y_1)}
\end{aligned}
$$

$$
\begin{aligned}
&\frac{d}{dy_1}x_1(y_1)=\frac{x_2(y_1)}{mg\sin(x_1(y_1))x_2(y_1)+\frac{2mLx_2(y_1)}{J}(mgL\sin(x_1(y_1))-c_p x_2(y_1))}\\
&\frac{d}{dy_1}x_2(y_1)=\frac{(mgL\sin(x_1(y_1))-c_p x_2(y_1))}{Jmg\sin(x_1(y_1))x_2(y_1)+2mLx_2(y_1)(mgL\sin(x_1(y_1))-c_p x_2(y_1))} \qquad h_{ep}(t)=0\\
&\frac{d}{dy_1}t(y_1)=\frac{1}{mg\sin(x_1(y_1))x_2(y_1)+\frac{2mLx_2(y_1)}{J}(mgL\sin(x_1(y_1))-c_p x_2(y_1))}
\end{aligned}
$$

(105)

For a transition from free-fall to pendulum, the state event handling is given for the state vector

$$\vec{\chi}(\tau)=(\chi_1(\tau),\chi_2(\tau),\chi_3(\tau),\chi_4(\tau),\chi_5(\tau))^T := (x_1(y_1),x_2(y_1),x_3(y_1),x_4(y_1),t(y_1))^T$$

for the interval

$$\left[\tau_{j-1},\tau_{j-1}+|\tau_{j-1}-a_e|\right]=\left[\tau_{j-1},\tau_{j-1}+|\tau_{j-1}-L|\right]$$

in which $y_1 := r$ is the independent variable τ. In case of a transition from pendulum to free-fall state, the state vector of transformation system is defined by

$$\vec{\chi}(\tau)=(\chi_1(\tau),\chi_2(\tau),\chi_3(\tau))^T := (x_1(y_1),x_2(y_1),t(y_1))^T$$

and $y_1 := F_a$ is defined as independent variable τ. The numerical computation is applied for the interval

$$\left[\tau_{j-1}, \tau_{j-1} + \left|\tau_{j-1} - a_e\right|\right] = \left[\tau_{j-1}, \tau_{j-1} + \left|\tau_{j-1} - 0\right|\right],$$

hence the step-size at the state event $\Delta\tilde{t}_e$ is computed using systems of ODEs (105) with ODE45 and then for approximation of the transient state vector at the state event, ODE45 is applied on the interval $\left[t_{j-1}, \tilde{t}_e\right]$.

5.2.5 Simulation Comparisions

The simulation comparisons are given for hypothetical system parameters, which are shown in Table 6.

Parameter	**Description**
$x_0 := 0.9\text{m}$	initial position along x-axis
$y_0 := -0.43\text{m}$	initial position along y-axis
$\varphi_0 := \pi + \operatorname{atan}(x_0/y_0)$	pendulum initial angle
$\omega_0 := 8\,\text{rad/s}$	initial angular velocity
$L := 1\text{m}$	filament length
$g := 9.80665\,\text{m/s}^2$	acceleration of gravity of earth's surface
$c_p := 0.97\,\text{N}\cdot\text{m}\cdot\text{s/rad}$	damping coefficient of pendulum system
$c_x := 0.95\,\text{N}\cdot\text{s/m}$	damping coefficient of free-fall system along x-axis
$c_y := 0.95\,\text{N}\cdot\text{s/m}$	damping coefficient of free-fall system along y-axis
$J := 0.75\text{kg}\cdot\text{m}^2$	mass moment of inertia
$m := 0.75\text{kg}$	pendulum mass

Table 6: Filament Pendulum Simulation Parameters

The results of the computer simulations are illustrated on Figures 13 and 14. These figures show the approximation of the positions and the trajectory of the pendulum.

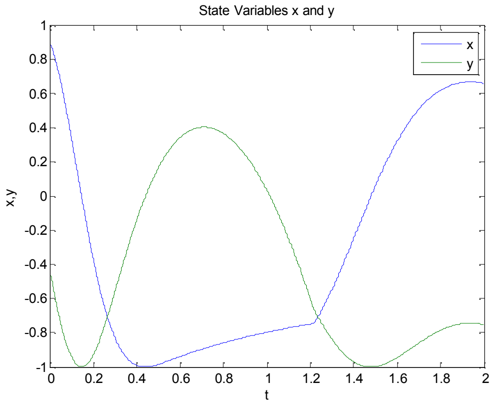

Figure 13: Simulation of Filament Pendulum Positions

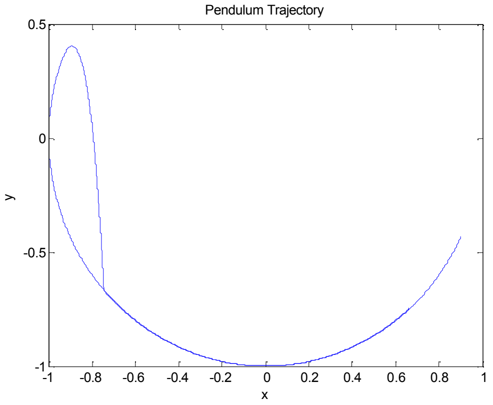

Figure 14: Simulation of Filament Pendulum Trajectory

The following table shows the comparison of the results of the simulations for different approximation methods of the transient magnitudes of the variable $y_1(t_e) := F_a(t_e)$ at the first transition from pendulum state to free-fall state.

Methods	Reformulation of Explicit Euler's Formula	Reformulation of Implicit Trapezoidal Formula for Newton's Method Application	Henon
Solver Algorithm	Implemented Euler's Solver	Implemented Trapezoidal Solver Mixed Newton Method	Programmed Henon Using Matlab ODE45
Type	Explicit	Implicit	Explicit
Programming Environment	Matlab	Matlab	Matlab
Threshold Value a_{e1}	0	0	0
$y_1(\tilde{t}_e)$	-5.456111e-016	-4.440892e-016	3.057221e-08
Local Error of $y_1(\tilde{t}_e)$	5.456111e-016	4.440892e-016	3.057221e-08
Fixed Step-Size	1e-05	1e-05	1e-05
$\Delta\tilde{t}_e$	1.363987e-06	1.628527e-06	9.522046e-06
$\tilde{t}_e$	5.600113e-01	5.600216e-01	5.600195e-01

Methods	Simulink	Adaptive (Matlab)
Solver Algorithm	ODE4	ODE45
Type	Explicit	Explicit
Programming Environment	Simulink	Matlab
Threshold Value a_{e_1}	0	0
$y_1(\tilde{t}_e)$	[-2.041575e-04, 2.036859e-04]	-1.443289e-13
Local Error of $y_1(\tilde{t}_e)$	-	1.443289e-13
Fixed Step-Size	1e-05	1e-05
$\Delta\tilde{t}_e$	-	9.522518e-06
$\tilde{t}_e$	[5.60130e-01, 5.60140e-01]	5.600195e-01

Table 7: First Event Comparisons of Filament Pendulum Simulation at the Transition from Pendulum to Free-Fall State

An approximation of the event location by "reformulation of solver formula" in DAEs demands a transformation of the DAEs to appropriate ODEs using an index-reduction method. The solver algorithm of DAEs transformed into ODEs has not always the same results as semi-explicit DAEs. This is known as the drift-off effect. In Table 7, the local error of algebraic variable $y_1(t_e)$ at the state event using Euler's solver in system of DAEs transformed in ODEs (101) is 5.456111e-16. That is the difference of the threshold value $a_{e_1} := 0$ to the approximated value $\hat{y}_1(\tilde{t}_e)$. This result can be verified directly by computing the constraint equation (90) using the approximated values $\hat{\varphi}(\tilde{t}_{ep})$ and $\hat{\omega}(\tilde{t}_{ep})$. In this case, the error is 1.265872e-04. This verification shows a larger error value[32].

The prediction of $y_1(t_e)$ is developed using reformulation of implicit trapezoidal formula (104) in root-finding method (38). Table 7 shows the magnitude of the variable $y_1(\tilde{t}_e)$ at the first state event, in which the value of $y_1(\tilde{t}_e)$ using root-finding algorithm has the lowest deviation from the threshold value $a_{e_1} = 0$. The local error in this manner is 4.440892e-16. An implicit solver with the root-finding prediction stage has additional costs for developing and programming the root-finding algorithm. Further, the simulation result depends on convergence location of the prediction variables and the iterations may take higher run time if the roots of the algorithm converge very slowly.

Table 8 shows the errors of the approximated magnitudes of $|F_a(t)|$ and $|r(t)|$ at the first and the second events.

	Methods			
	Reformulation of Explicit Euler's Formula	**Reformulation of Implicit Trapezoidal Formula for Newton's Method**	**Henon**	**Adaptive Matlab**
Error $\left\|F_a(\tilde{t}_e)\right\|$ **at First Event**	5.456111e-16	4.440892e-016	3.057221e-08	1.443289e-13
Error $\left\|r(\tilde{t}_e) - L\right\|$ **at Second Event**	0	0	2.869926e-12	1.709743e-14

Table 8: Errors of Approximated Magnitudes of Algebraic Variables at the State Events in Variable Structure System of ODEs for Filament Pendulum

---

[32] The index reduction method is applied in DAEs for two methods "reformulation of solver formula" and "Henon's method". Hence, the using of index reduction method may result in a drift-off effect and higher error magnitudes. The verifications of approximated magnitudes in constraint equations show relative greater local errors of approximated magnitudes of variable $y_1(\tilde{t}_e)$ at first event in variable structure systems of DAEs than "reformulation of implicit solver system for root-finding methods" in filament pendulum simulations.

5.3 Rotor and Stator with Hybrid System of DAEs

The non-dimensional rotor-stator contact model is described by [SPE07]. The fundamental design of the rotor model consists of a Jeffcott rotor equation [EKD93]. The subject of this section refers to the simulation of a non-dimensional rotor-stator hybrid system and the approximation of the locations of the transient vector[33] at the state event.

5.3.1 Model Description

The main difference between a rotor-stator contact model and a Jeffcott rotor model lays in the impact force terms and the stator model. The model, which is simulated in this section, describes both deflection behaviour and force interactions including impact force of the rotor-stator before and after collision.

The precise construction of a rotor is very difficult in practice. Faults during machining or construction can create an eccentricity radius of the rotor axis. The rotor axis must be centred on its disc, and an eccentricity distance from the centre of its disc can produce a centrifugal force. This force deflects the rotor from its central position, and it can lead to large deflections at the critical speed. The rotor deflection depends on the rotor speed and its mass, and on the parameters of the rotor-stator spring damper deflection model.

The rotor model is given by nonhomogeneous differential equations (106). The left bracket presents the motion properties with damping stiffness components. The middle bracket defines the non-balance excitation. This term includes the angular velocity ω, mass of rotor m_r and the eccentricity radius e. The bracket on the rhs. of (106) contains the impact force terms, which are projected with the coordinate transformation matrix with angle $\varphi(t)$ on the x- and y-axes:

$$\begin{bmatrix} m_r\ddot{x}_r(t)+c_r\dot{x}_r(t)+k_r x_r(t) \\ m_r\ddot{y}_r(t)+c_r\dot{y}_r(t)+k_r y_r(t) \end{bmatrix} = \begin{bmatrix} m_r e\omega^2\cos(\omega t) \\ m_r e\omega^2\sin(\omega t) \end{bmatrix} + \begin{bmatrix} -\cos(\varphi(t)) & \sin(\varphi(t)) \\ -\sin(\varphi(t)) & -\cos(\varphi(t)) \end{bmatrix} \begin{bmatrix} f_{cn}(t) \\ f_{ct}(t) \end{bmatrix}. \tag{106}$$

The variables $x_r(t)$ and $y_r(t)$ are the rotor position variables, which can be depicted on the x- and y-axes, m_r is the rotor mass, and the parameters c_r and k_r are the approximated rotor

---

[33] A transient vector contains state vector and algebraic variable vector. Transient vector is computed at a switching event for initialization of transition systems or states in hybrid or variable structure systems of DAEs.

damper and stiffness coefficients. The stator equations of motion in x and y directions are specified by:

$$\begin{bmatrix} m_s\ddot{x}_s(t)+c_s\dot{x}_s(t)+k_sx_s(t) \\ m_s\ddot{y}_s(t)+c_s\dot{y}_s(t)+k_sy_s(t) \end{bmatrix} = -\begin{bmatrix} -\cos(\varphi(t)) & \sin(\varphi(t)) \\ -\sin(\varphi(t)) & -\cos(\varphi(t)) \end{bmatrix} \begin{bmatrix} f_{cn}(t) \\ f_{ct}(t) \end{bmatrix}. \tag{107}$$

The parameters c_s and k_s are stator damping and stiffness coefficients whereas m_s is the stator mass. The impact force in the stator equation is similar to the impact force in the rotor equation but in opposite direction. The system structure can be changed, in which the impact forces $f_{cn}(t)$ and $f_{ct}(t)$ in the rotor-stator equations can be coupled or decoupled according to states of switching variables. The contact force consists of two forces $f_{cn}(t)$ and $f_{ct}(t)$ in the radial and tangential directions.

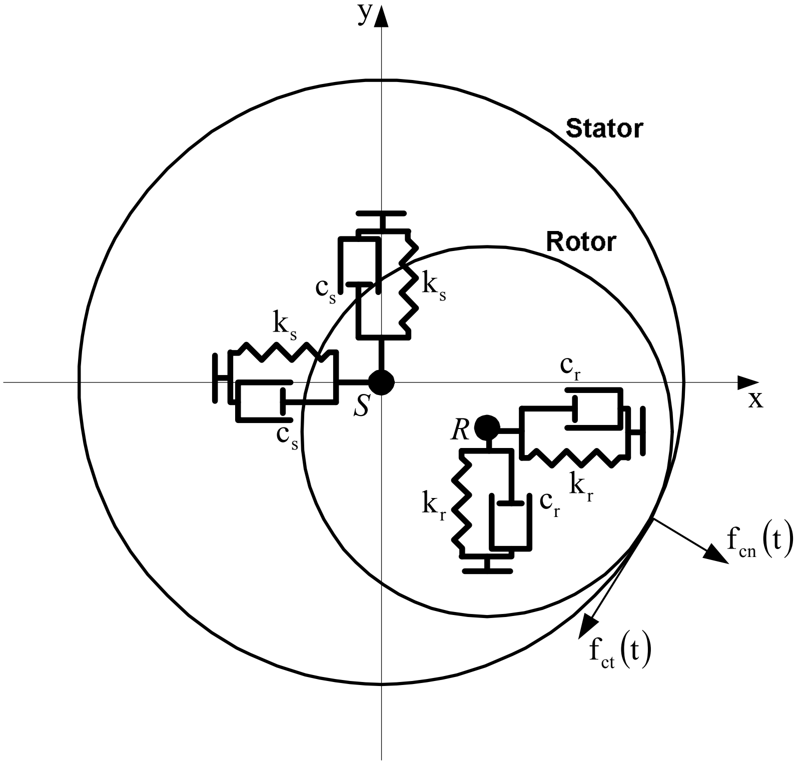

Figure 15: Tangential and Radial Forces of Rotor-Stator in Contact Location

The rotor-stator system has algebraic equations which control the activation of the two switching variables $q_c(t)$ and $q_t(t)$. The radial force

$$f_n(t) = k_h d(t) + c_h \dot{d}(t)$$

represents a stiffness damping characteristic which is defined by the switching model via

$$f_{cn}(t) = q_c(t) f_n(t)$$

as follows:

$$f_{cn}(t) = q_c(t)\left(k_h d(t) + c_h \dot{d}(t)\right) \qquad q_c(t) := \begin{cases} 0 & d(t) \geq 0 \\ 1 & f_n(t) \leq 0. \end{cases}$$

(108)

The variable $d(t)$ is the radial indentation and the parameters k_h and c_h are the stiffness and damping coefficients in the radial force model.

The switching variable $q_t(t)$ gives the direction of the tangential force $f_t(t)$ as follows

$$f_t(t) = q_t(t) \mu f_{cn}(t) \qquad q_t(t) := \begin{cases} -1 & v_t(t) < 0 \\ 1 & v_t(t) > 0 \end{cases}$$

(109)

in which μ is the friction coefficient and $v_t(t)$ is the tangential velocity.

The radial and tangential velocity equations presented in [SPE07] are

$$\begin{bmatrix} \dot{d}(t) \\ v_t(t) \end{bmatrix} = \begin{bmatrix} 0 \\ \omega \frac{D_r}{2} \end{bmatrix} + \begin{bmatrix} \cos(\varphi(t)) & \sin(\varphi(t)) \\ -\sin(\varphi(t)) & \cos(\varphi(t)) \end{bmatrix} \begin{bmatrix} \dot{x}_{rs}(t) \\ \dot{y}_{rs}(t) \end{bmatrix}$$

(110)

where the parameter D_r is the rotor diameter, ω is the angular velocity, and $v_t(t)$ the tangential impact velocity.

The rotor-stator system may show two manufacturing problems: The first problem is the eccentricity radius e, which was mentioned at the beginning of this chapter. The other one is the position of the rotor centre in relation to the stator centre, where the rotor centre has an offset from the stator. In Figure 16 this offset is illustrated with x_0 and y_0.

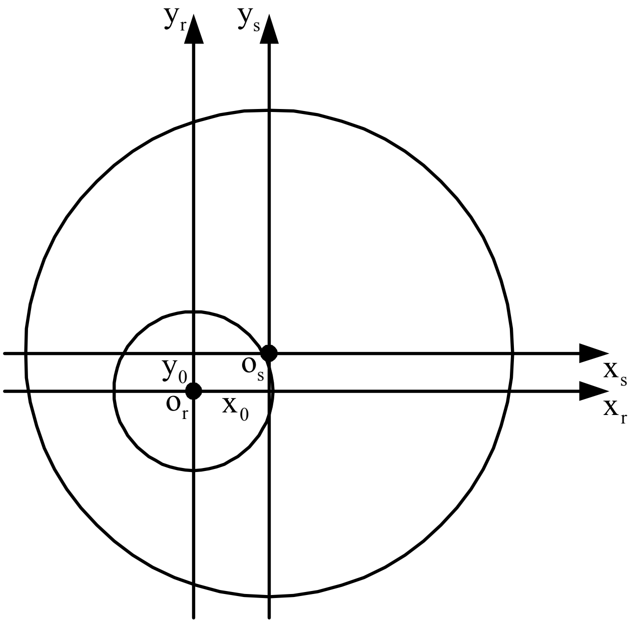

Figure 16: Rotor and Stator Offsets

If the stator is in a deflected position, then it should be considered in the computation of $x_s(t)$ and $y_s(t)$. Figure 17 shows the new position of $x_s(t)$ and $y_s(t)$ as well as the offset between the stator and rotor.

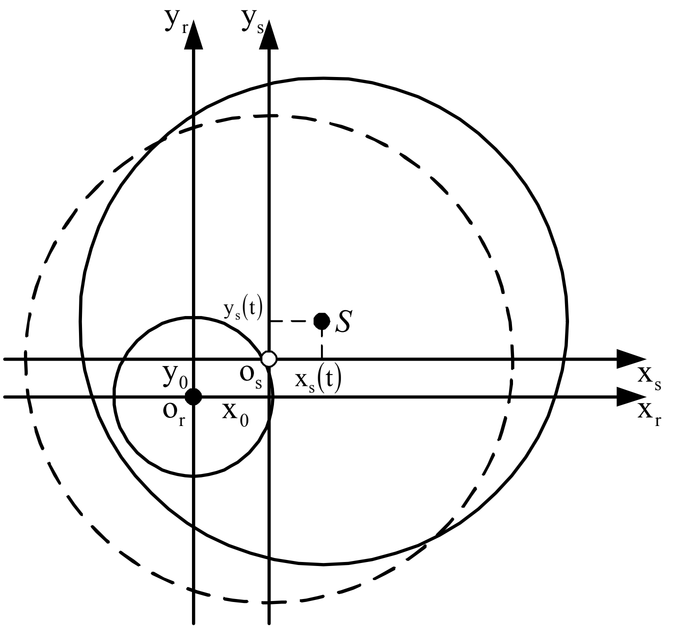

Figure 17: Stator Deflection

The positions of the rotor $x_r(t)$ and $y_r(t)$, the eccentricity radius e as well as the stator centre are shown in Figure 18.

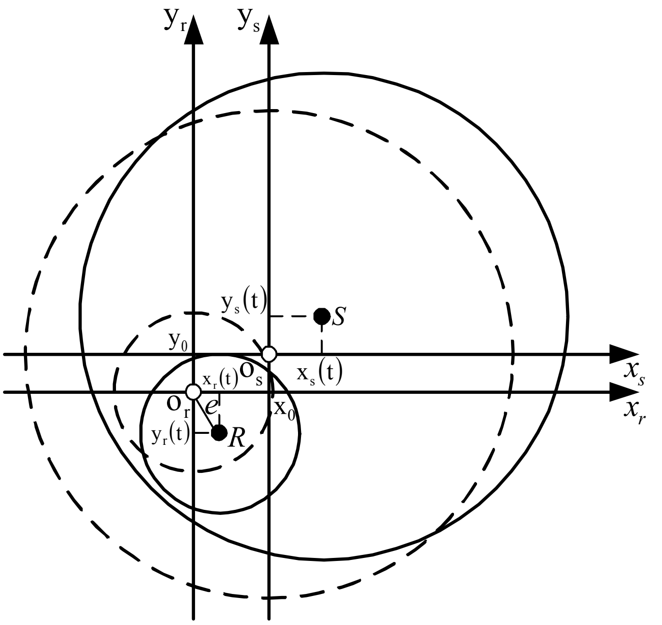

Figure 18: Rotor Eccentricity Radius and Rotor-Stator Deflections

The relative deflections $x_{rs}(t)$ and $y_{rs}(t)$ between the rotor and stator are given by

$$x_{rs}(t) = x_r(t) - (x_s(t) + x_0)$$

(111)

$$y_{rs}(t) = y_r(t) - (y_s(t) + y_0)$$

(112)

whereas x_0 and y_0 are the rotor-stator offsets. The radial distance between rotor and stator is

$$r_{rs}^2(t) = x_{rs}^2(t) + y_{rs}^2(t)$$

(113)

and the radial indentation or intrusion depth [SPE07] is obtained from the radial distance between rotor and stator

$$d(t) = r_{rs}(t) - c.$$

(114)

Parameter c is the rotor-stator radial clearance.

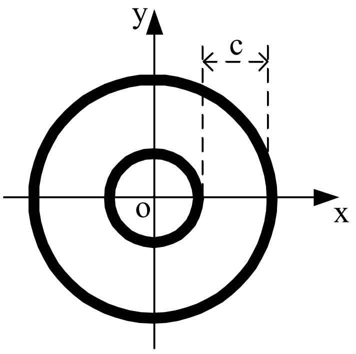

Figure 19: Radial Rotor to Stator Clearance

The contact angle can be computed by

$$\varphi(t) = \operatorname{atan}\left(\frac{y_{rs}(t)}{x_{rs}(t)}\right).$$

(115)

The rotor-stator contact model, which is introduced by the differential equations (106) and (107), is a hybrid system. In order to avoid numerical problem, a dimensionless[34] hybrid system should be applied. Hence, in the next step, the transformation from a dimensional system to a dimensionless system is demonstrated. The construction of a dimensionless model is given by replacing the derivative terms. It starts with the angular velocity $2\pi f t = \omega t = 2\pi\iota$ of the rotor. The derivation $\omega dt = 2\pi d\iota$ with respect to time and the assumption of constant frequency f leads to the norming factor $\frac{2\pi}{\omega}$ as follows

$$dt = \frac{2\pi}{\omega} d\iota .$$

(116)

Variable t is the independent variable of the initial system and ι is the independent variable of the dimensionless system. The first and second derivation operators for the transformations of the dimensional derivative terms into the dimensionless derivative terms with respect to ι are

$$(\ )' = \frac{d}{d\iota}(\ ) = \frac{2\pi}{\omega}\left(\frac{d}{dt}(\ )\right)$$

(117)

and

$$(\ )'' = \frac{d}{d\iota}\left(\frac{d}{d\iota}(\ )\right) = \frac{2\pi}{\omega}\left(\frac{d}{dt}\left(\frac{2\pi}{\omega}\left(\frac{d}{dt}(\ )\right)\right)\right) = \left(\frac{2\pi}{\omega}\right)^2\left(\frac{d^2}{d^2t}(\ )\right).$$

(118)

---

[34] In a dimensionless system, all variables should be changed into scalar variables. A simulation of the dimensionless system can be used for interpretations of the system behaviors, effects of parameter modifications, results and measured values.

The following two operators can perform the transformations of the first and second order dimensionless derivative terms into dimensional derivative terms. The first order derivative operator can be obtained from (117)

$$\dot{(\ )} = \frac{d}{dt}(\ ) = \frac{\omega}{2\pi}\left(\frac{d}{d\iota}(\ )\right) = \frac{\omega}{2\pi}(\ )'$$

(119)

and the second order derivative operator from (118)

$$\ddot{(\ )} = \frac{d^2}{d^2t}(\ ) = \frac{\omega}{2\pi}\left(\frac{d}{d\iota}\left(\frac{\omega}{2\pi}\left(\frac{d}{d\iota}(\ )\right)\right)\right) = \left(\frac{\omega}{2\pi}\right)^2(\ )''.$$

(120)

The different variables and constants are converted into dimensionless variables or constants by dividing these magnitudes by the length of clearance.

$$X_r(\iota) = \frac{x_r(\iota)}{c},\ Y_r(\iota) = \frac{y_r(\iota)}{c},\ X_s(\iota) = \frac{x_s(\iota)}{c},\ Y_s(\iota) = \frac{y_s(\iota)}{c},\ X_{rs}(\iota) = \frac{x_{rs}(\iota)}{c},\ Y_{rs}(\iota) = \frac{y_{rs}(\iota)}{c},$$
$$R_{rs}(\iota) = \frac{r_{rs}(\iota)}{c},\ D(\iota) = \frac{d(\iota)}{c},\ X_0 = \frac{x_0}{c},\ Y_0 = \frac{y_0}{c},\ E = \frac{e}{c}.$$

(121)

The operators (117), (118) and the relations in (121) are used to transform the dimensional terms in the rotor-stator hybrid system into the dimensionless terms. After the substitutions the rotor system is presented by

$$\begin{bmatrix} m_r\left(\frac{\omega}{2\pi}\right)^2 cX_r''(\iota) + c_r\left(\frac{\omega}{2\pi}\right)cX_r'(\iota) + k_r cX_r(\iota) \\ m_r\left(\frac{\omega}{2\pi}\right)^2 cY_r''(\iota) + c_r\left(\frac{\omega}{2\pi}\right)cY_r'(\iota) + k_r cY_r(\iota) \end{bmatrix} = \begin{bmatrix} m_r cE\omega^2\cos(2\pi\iota) \\ m_r cE\omega^2\sin(2\pi\iota) \end{bmatrix} + \begin{bmatrix} -\cos(\varphi(\iota)) & \sin(\varphi(\iota)) \\ -\sin(\varphi(\iota)) & -\cos(\varphi(\iota)) \end{bmatrix}\begin{bmatrix} f_{cn}(\iota) \\ f_{ct}(\iota) \end{bmatrix}$$

(122)

and the stator system is

$$\begin{bmatrix} m_s\left(\frac{\omega}{2\pi}\right)^2 cX''(\iota)_s + c_s\left(\frac{\omega}{2\pi}\right) cX_s'(\iota) + k_s cX_s(\iota) \\ m_s\left(\frac{\omega}{2\pi}\right)^2 cY_s''(\iota) + c_s\left(\frac{\omega}{2\pi}\right) cY_s'(\iota) + k_s cY_s(\iota) \end{bmatrix} = -\begin{bmatrix} -\cos(\varphi(\iota)) & \sin(\varphi(\iota)) \\ -\sin(\varphi(\iota)) & -\cos(\varphi(\iota)) \end{bmatrix} \begin{bmatrix} f_{cn}(\iota) \\ f_{ct}(\iota) \end{bmatrix}.$$

(123)

The radial and tangential forces[35] are obtained in the same way:

$$f_{cn}(\iota) = q_c(\iota) f_n(\iota) = q_c(\iota)\left(ck_h D(\iota) + cc_h\left(\frac{\omega}{2\pi}\right) D'(\iota)\right),$$

(124)

$$f_{ct}(\iota) = q_t(\iota)\mu f_{cn}(\iota).$$

(125)

The parameters of the dimensionless differential equations of rotor-stator are defined as follows:

$$\Omega_r^2 = \frac{k_r}{m_r},\ \Omega_s^2 = \frac{k_s}{m_s},\ \Omega_h^2 = \frac{k_h}{\bar{m}},\ \eta = \frac{\omega}{\Omega_r}\ \ A_r = \frac{2\pi}{\eta},$$

$$A_s = \frac{2\pi}{\eta}\frac{\Omega_s}{\Omega_r},\ A_h = \frac{2\pi}{\eta}\frac{\Omega_h}{\Omega_r},\ \zeta_r = \frac{c_r}{2\sqrt{k_r m_r}},\ \zeta_s = \frac{c_s}{2\sqrt{k_s m_s}},\ \zeta_h = \frac{c_h}{2\sqrt{k_h \bar{m}}},\ \bar{m} = \frac{m_r m_s}{m_r + m_s}.$$

(126)

The dimensionless equations of motion can be given by the substitution of the parameters in (126) into the rotor-stator hybrid systems (122) and (123). After these replacements, the dimensionless differential equations of the rotor-stator system can be given as follows:

---

[35] The variable D is dimensionless radial intrusion depth. The coefficients c_h, k_h, μ and c are explained in pages 77 and 80.

$$\begin{bmatrix} X_r''(\iota)+2\zeta_r A_r X_r'(\iota)+A_r^2 X_r(\iota) \\ Y_r''(\iota)+2\zeta_r A_r Y_r'(\iota)+A_r^2 Y_r(\iota) \end{bmatrix} = \begin{bmatrix} (2\pi)^2 E\cos(2\pi\iota) \\ (2\pi)^2 E\sin(2\pi\iota) \end{bmatrix} + \begin{bmatrix} q_c\left(A_h^2 D(\iota)+2\zeta_h A_h D'(\iota)\right)\left(-\cos(\varphi(\iota))+q_t\mu\sin(\varphi(\iota))\right) \\ q_c\left(A_h^2 D(\iota)+2\zeta_h A_h D'(\iota)\right)\left(-\sin(\varphi(\iota))-q_t\mu\cos(\varphi(\iota))\right) \end{bmatrix}$$

(127)

$$\begin{bmatrix} X_s''(\iota)+2\zeta_s A_s X_s'(\iota)+A_s^2 X_s(\iota) \\ Y_s''(\iota)+2\zeta_s A_s Y_s'(\iota)+A_s^2 Y_s(\iota) \end{bmatrix} = -\begin{bmatrix} q_c\left(A_h^2 D(\iota)+2\zeta_h A_h D'(\iota)\right)\left(-\cos(\varphi(\iota))+q_t\mu\sin(\varphi(\iota))\right) \\ q_c\left(A_h^2 D(\iota)+2\zeta_h A_h D'(\iota)\right)\left(-\sin(\varphi(\iota))-q_t\mu\cos(\varphi(\iota))\right) \end{bmatrix}.$$

(128)

The computations of further quantities are necessary to complete the simulation model. The dimensionless relative deflection between the rotor and the stator along the x- and y-axes can be computed from the updated position values of the rotor-stator dimensionless differential equations:

$$X_{rs}(\iota)=X_r(\iota)-\left(X_s(\iota)+X_0\right)$$

(129)

$$Y_{rs}(\iota)=Y_r(\iota)-\left(Y_s(\iota)+Y_0\right).$$

(130)

The dimensionless relative radial deflection is described by

$$R_{rs}(\iota)=\sqrt{X_{rs}^2(\iota)+Y_{rs}^2(\iota)}$$

(131)

and the angle of deflection is presented as follows

$$\varphi(\iota)=\operatorname{atan}\left(\frac{Y_{rs}(\iota)}{X_{rs}(\iota)}\right).$$

(132)

The model simulation is put together by the dimensionless differential equations and the dimensional algebraic equations. For this purpose, the position coordinates of the rotor and stator $x_r(\iota)$, $y_r(\iota)$, $x_s(\iota)$, $y_s(\iota)$ are computed from the dimensionless values $X_r(\iota)$, $Y_r(\iota)$, $X_s(\iota)$, $Y_s(\iota)$, hence

$$x_r(\iota)=cX_r(\iota),\ y_r(\iota)=cY_r(\iota),\ x_s(\iota)=cX_s(\iota),\ y_s(\iota)=cY_s(\iota),\ r_{rs}(\iota)=cR_{rs}(\iota),$$

so the intrusion depth can be given by

$$d(\iota)=c\left(R_{rs}(\iota)-1\right)$$

(133)

and the velocities by

$$\begin{bmatrix}\dot{d}(\iota)\\ v_t(\iota)\end{bmatrix}=\begin{bmatrix}0\\ \omega\dfrac{D_r}{2}\end{bmatrix}+c\frac{\omega}{2\pi}\begin{bmatrix}\cos(\varphi(\iota)) & \sin(\varphi(\iota))\\ -\sin(\varphi(\iota)) & \cos(\varphi(\iota))\end{bmatrix}\begin{bmatrix}X'_{rs}(\iota)\\ Y'_{rs}(\iota)\end{bmatrix}.$$

(134)

The contact force in radial direction is given by

$$f_{cn}(\iota)=q_c(\iota)f_n(\iota)=q_c(\iota)\left(k_h d(\iota)+c_h\dot{d}(\iota)\right)$$

(135)

and in tangential direction by

$$f_{ct}(\iota)=q_t(\iota)\mu f_{cn}(\iota).$$

(136)

The conditional expressions (137) and (138) form the switching system.

$$q_c(\iota):=\begin{cases}1 & (d(\iota)\geq 0)\wedge(f_n(\iota)>0)\\ 0 & (f_n(\iota)\leq 0)\end{cases}$$

(137)

$$q_t(\iota) := \begin{cases} -1 & v_t(\iota) < 0 \\ 1 & v_t(\iota) > 0 \end{cases}$$

(138)

The dimensionless switching variables are converted to dimensional variables of $d(\iota)$, $f_n(\iota)$, and $v_t(\iota)$ and then appropriate switching variables $q_c(\iota)$ and/or $q_t(\iota)$ are evaluated in order to control the switching process. Figure 20 shows an abstract simulation concept for the hybrid system of DAEs with dimensional and dimensionless variables.

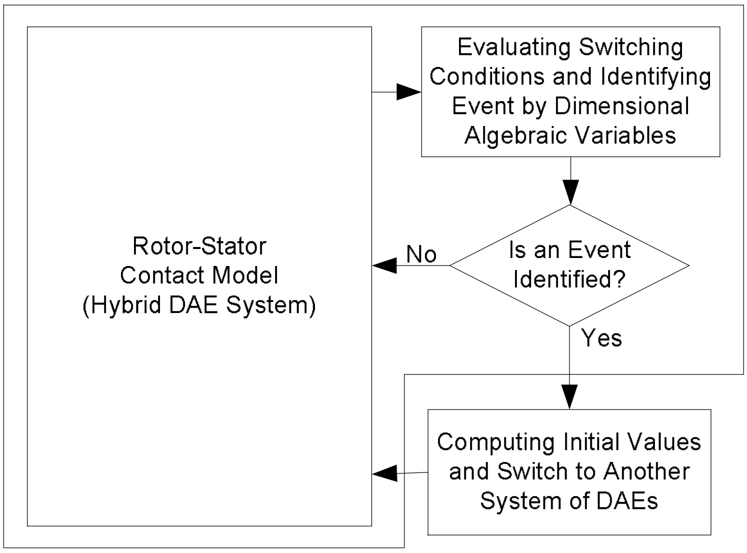
Rotor-Stator
Contact Model
(Hybrid DAE System)
Evaluating Switching Conditions and Identifying Event by Dimensional Algebraic Variables
Is an Event Identified?
No
Yes
Computing Initial Values and Switch to Another System of DAEs

Figure 20: Simulation Concept[36] Based on Dimensional and Dimensionless Variables

State graphs can be used for the analysis and development of the switching process. First, the state graphs of the switching variables and then the state graphs of the system transitions are demonstrated. The state graphs of the switching variable $q_c(\iota)$ and its switching conditions are depicted in the following figure.

---

[36] Applying dimensionless model and nonlinear transformation between dimensionless and dimensional switching variables result in non-qualitative simulation approach.

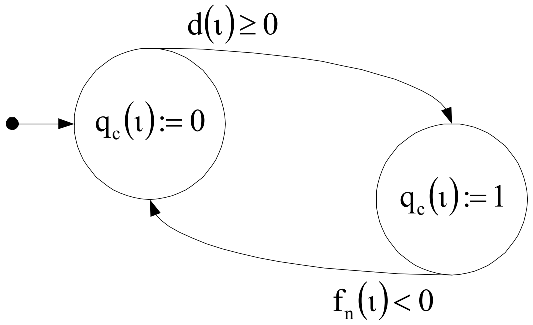

Figure 21: State Graph of Rotor-Stator Contact by Switching Variable $q_c(\iota)$

The switching conditions of the radial force structure are presented in [BEK10] as follows:

$$f_{cn}(\iota) = q_c(\iota) f_n(\iota) = q_c(\iota)\left(k_h d(\iota) + c_h \dot{d}(\iota)\right) \qquad q_c(\iota) := \begin{cases} 1 & (f_n(\iota) > 0) \wedge (d(\iota) \geq 0) \\ 0 & (f_n(\iota) \leq 0). \end{cases} \tag{139}$$

The switching variable $q_c(\iota)$ in (139) is determined by two conditional expressions with two variables $d(\iota)$ and $f_n(\iota)$. The state graph of $q_c(\iota)$ is illustrated in Figure 22.

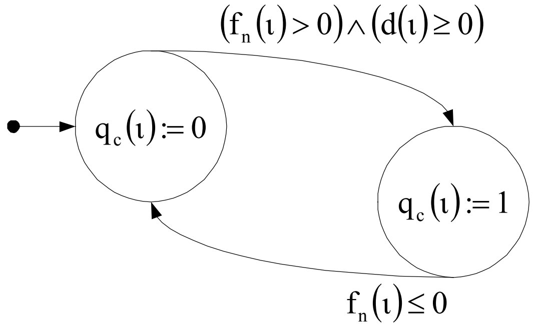

Figure 22: Extended Form of State Graph of Switching Variable $q_c(\iota)$

The rhs. of the rotor-stator differential equations have the interrelated terms of the radial force $f_n(\iota)$ and the radial intrusion depth $d(\iota)$. The radial force structure is valid if the radial force is greater than zero and simultaneously the radial impact depth is greater or equal to zero.

The condition for disconnecting of the rotor from the stator is modified according to the logical negation[37] of the condition

$$\left(\mathrm{f}_{\mathrm{n}}(\iota)>0\right)\wedge\left(\mathrm{d}(\iota)\geq 0\right)$$

by

$$\neg\left(\left(\mathrm{f}_{\mathrm{n}}(\iota)>0\right)\wedge\left(\mathrm{d}(\iota)\geq 0\right)\right)=\neg\left(\mathrm{f}_{\mathrm{n}}(\iota)>0\right)\vee\neg\left(\mathrm{d}(\iota)\geq 0\right)=\left(\mathrm{f}_{\mathrm{n}}(\iota)\leq 0\right)\vee\left(\mathrm{d}(\iota)<0\right).$$

The reset map of the state transitions is illustrated in Table 9:

Conditions		**Corresponding Logical Values**		**Connecting Transition Condition** $\left(\mathrm{f}_{\mathrm{n}}(\iota)>0\right)\wedge\left(\mathrm{d}(\iota)\geq 0\right)$	**Disconnecting Transition Condition** $\left(\mathrm{f}_{\mathrm{n}}(\iota)\leq 0\right)\vee\left(\mathrm{d}(\iota)<0\right)$	$\mathrm{q}_{\mathrm{c}}(\iota)$	**Active State**
$\mathrm{f}_{\mathrm{n}}(\iota)$	$\mathrm{d}(\iota)$	$q_{\mathrm{f}_{\mathrm{n}}}^{\mathrm{L}}(\iota)$	$q_{\mathrm{d}}^{\mathrm{L}}(\iota)$				
$\mathrm{f}_{\mathrm{n}}(\iota)<0$	$\mathrm{d}(\iota)<0$	*F*	*F*	*F*	*T*	0	Disjunction
$\mathrm{f}_{\mathrm{n}}(\iota)<0$	$\mathrm{d}(\iota)\geq 0$	*F*	*T*	*F*	*T*	0	Disjunction
$\mathrm{f}_{\mathrm{n}}(\iota)\geq 0$	$\mathrm{d}(\iota)<0$	*T*	*F*	*F*	*T*	0	Disjunction
$\mathrm{f}_{\mathrm{n}}(\iota)\geq 0$	$\mathrm{d}(\iota)\geq 0$	*T*	*T*	*T*	*F*	1	Contact

Table 9: Reset Map of Structure Switching of Rotor-Stator System using Logical Variables $q_{\mathrm{f}}^{\mathrm{L}}(\iota)$ and $q_{\mathrm{d}}^{\mathrm{L}}(\iota)$

According to the previous table, the contact between rotor and stator can take place if both corresponding logical auxiliary variables $q_{\mathrm{f}_{\mathrm{n}}}^{\mathrm{L}}(\iota)$ and $q_{\mathrm{d}}^{\mathrm{L}}(\iota)$ are true, so the disconnection of the rotor and the stator is valid if either $q_{\mathrm{f}_{\mathrm{n}}}^{\mathrm{L}}(\iota)$, $q_{\mathrm{d}}^{\mathrm{L}}(\iota)$ or both of them are false. In this dissertation, the disconnecting conditions of the radial force are defined and developed as follows

---

[37] The rotor stator contact model is simulated for two first events and the constraint equations (140) and (139) have the same outcomes.
The logical negation of a switching expression is not valid in all applications. E.g. in filament pendulum system the negation rule cannot given because the switching system has two separate switching conditions and each condition is concerned to its own system.

$$f_{cn}(\iota) = q_c(\iota) f_n(\iota) = q_c(\iota)\left(k_h d(\iota) + c_h \dot{d}(\iota)\right) \qquad q_c(\iota) := \begin{cases} 1 & (f_n(\iota) > 0) \wedge (d(\iota) \geq 0) \\ 0 & (f_n(\iota) \leq 0) \vee (d(\iota) < 0) \end{cases} \tag{140}$$

in which the logical compliment of the transition condition $(f_n(\iota) > 0) \wedge (d(\iota) \geq 0)$ is given by the separation condition $(f_n(\iota) \leq 0) \vee (d(\iota) < 0)$.

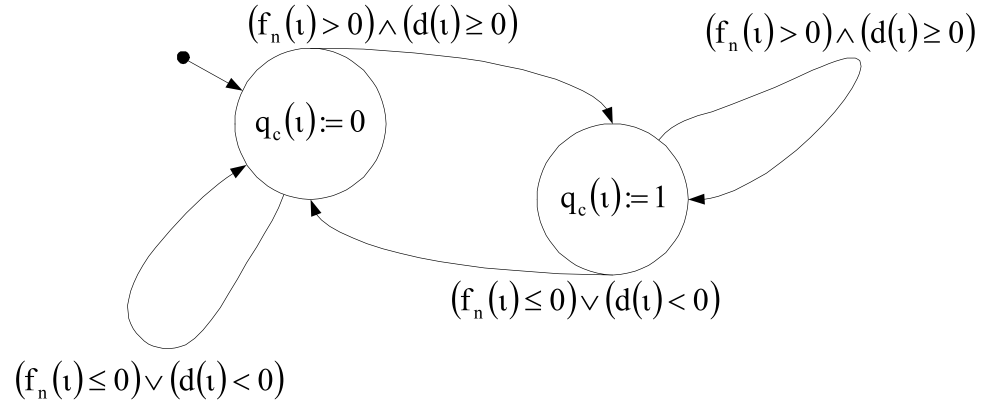
$(f_n(\iota) > 0) \wedge (d(\iota) \geq 0)$
$(f_n(\iota) > 0) \wedge (d(\iota) \geq 0)$
$q_c(\iota) := 0$
$q_c(\iota) := 1$
$(f_n(\iota) \leq 0) \vee (d(\iota) < 0)$
$(f_n(\iota) \leq 0) \vee (d(\iota) < 0)$

Figure 23: State Graph of Rotor-Stator Contact by Switching Variable $q_c(\iota)$ According to Table 7

The state graph of the switching variable $q_t(\iota)$ is illustrated in Figure 24.

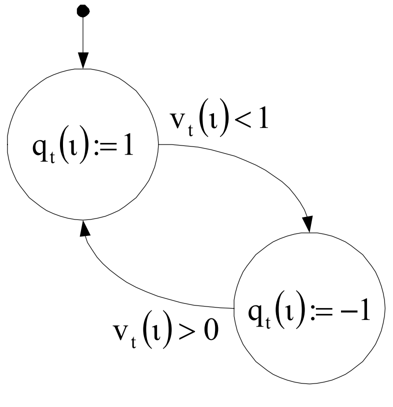

Figure 24: State Graph of Tangential Force by Switching Variable $q_t(\iota)$

It is assumed that both the static and kinetic friction are described approximately by formula (125) with friction coefficient μ, then the transient conditions of the switching variable $q_t(\iota) \in \{-1, 1\}$ can be developed by conditional expressions in (141) as follows:

$$f_t(\iota) = q_t(\iota)\mu f_{cn}(\iota) \qquad q_t(\iota) := \begin{cases} -1 & v_t(\iota) < 0 \\ 1 & v_t(\iota) > 0 \\ q_t(\iota^-) & v_t(\iota) = 0 \end{cases}$$

(141)

whereas $f_{cn}(\iota)$ and $f_t(\iota)$ show the radial and tangential forces. For the tangential velocity $v_t(\iota)$ three different values are possible $v_t(\iota) > 0$, $v_t(\iota) < 0$ and $v_t(\iota) = 0$. The tangential force depends on the radial force, the switching variable $q_t(\iota)$ and the friction coefficient. Here it is assumed that the switching variable $q_t(\iota)$ holds its previous value $q_t(\iota) := q_t(\iota^-)$, if the tangential impact velocity is equal to zero and the conditional relation $(f_n(\iota) > 0) \wedge (d(\iota) \geq 0)$ is valid. The modified state graph is shown in Figure 25.

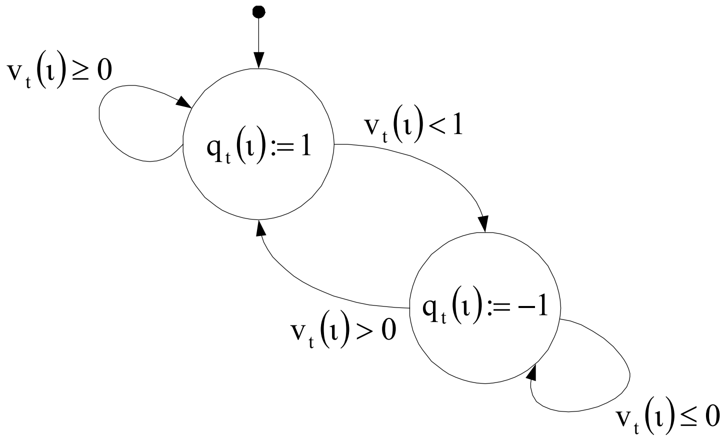

Figure 25: Modified State Graph of Switching Variable $q_t(\iota)$

The system structure switches if a change in the variables $q_c(\iota)$ and $q_t(\iota)$ occurs. The switching variable $q_c(\iota)$ defines the contact state, and the variable $q_t(\iota)$ defines the changing direction of the friction force state. The state graph of the rotor-stator system is demonstrated with the conditional expressions (141) and (140) in Figure 26.

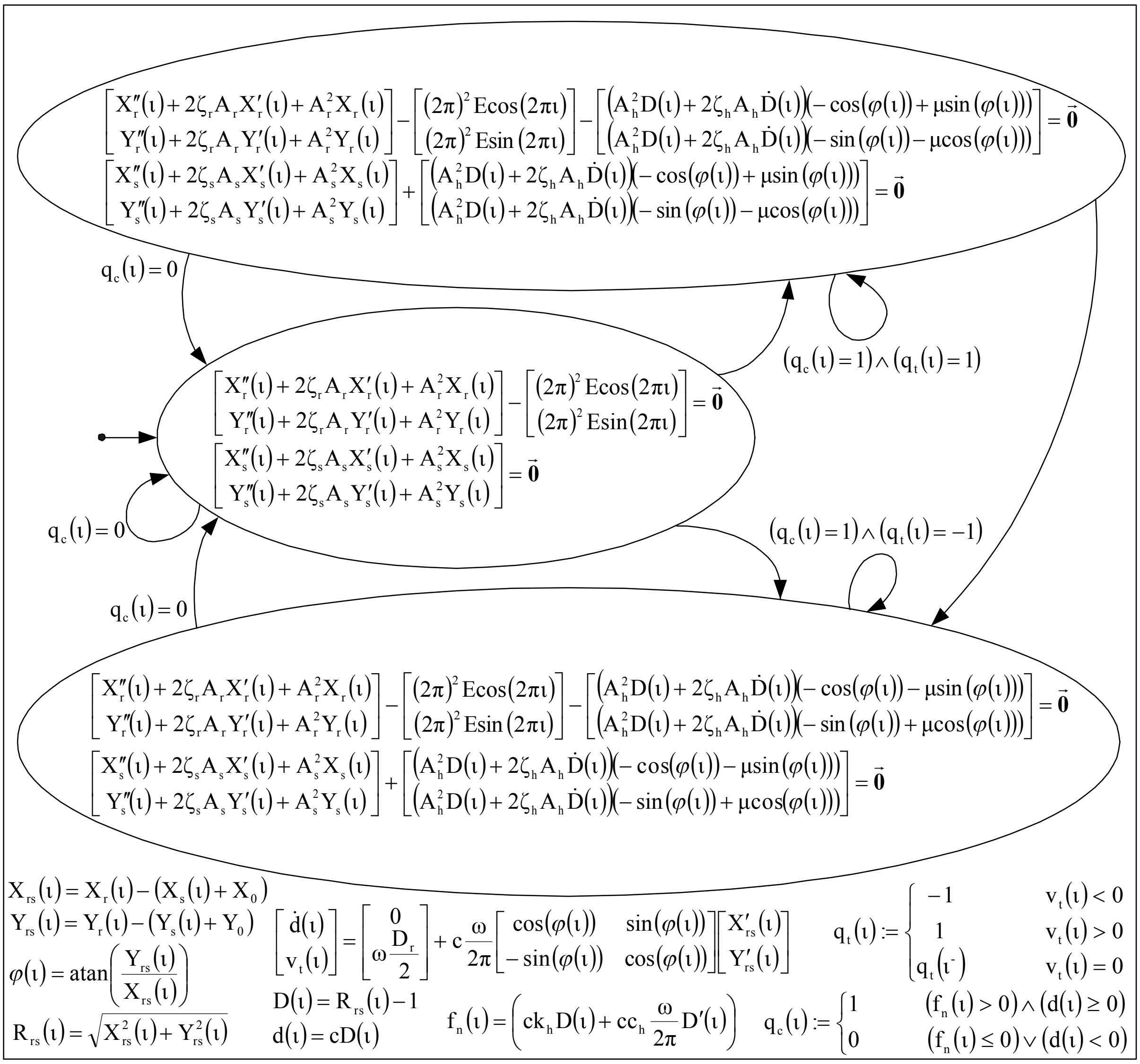
$\begin{bmatrix} X_r''(\iota)+2\zeta_r A_r X_r'(\iota)+A_r^2 X_r(\iota) \\ Y_r''(\iota)+2\zeta_r A_r Y_r'(\iota)+A_r^2 Y_r(\iota) \end{bmatrix} - \begin{bmatrix} (2\pi)^2 E\cos(2\pi\iota) \\ (2\pi)^2 E\sin(2\pi\iota) \end{bmatrix} - \begin{bmatrix} (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\cos(\varphi(\iota))+\mu\sin(\varphi(\iota))) \\ (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\sin(\varphi(\iota))-\mu\cos(\varphi(\iota))) \end{bmatrix} = \vec{0}$
$\begin{bmatrix} X_s''(\iota)+2\zeta_s A_s X_s'(\iota)+A_s^2 X_s(\iota) \\ Y_s''(\iota)+2\zeta_s A_s Y_s'(\iota)+A_s^2 Y_s(\iota) \end{bmatrix} + \begin{bmatrix} (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\cos(\varphi(\iota))+\mu\sin(\varphi(\iota))) \\ (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\sin(\varphi(\iota))-\mu\cos(\varphi(\iota))) \end{bmatrix} = \vec{0}$
$q_c(\iota)=0$
$(q_c(\iota)=1)\wedge(q_t(\iota)=1)$
$\begin{bmatrix} X_r''(\iota)+2\zeta_r A_r X_r'(\iota)+A_r^2 X_r(\iota) \\ Y_r''(\iota)+2\zeta_r A_r Y_r'(\iota)+A_r^2 Y_r(\iota) \end{bmatrix} - \begin{bmatrix} (2\pi)^2 E\cos(2\pi\iota) \\ (2\pi)^2 E\sin(2\pi\iota) \end{bmatrix} = \vec{0}$
$\begin{bmatrix} X_s''(\iota)+2\zeta_s A_s X_s'(\iota)+A_s^2 X_s(\iota) \\ Y_s''(\iota)+2\zeta_s A_s Y_s'(\iota)+A_s^2 Y_s(\iota) \end{bmatrix} = \vec{0}$
$q_c(\iota)=0$
$(q_c(\iota)=1)\wedge(q_t(\iota)=-1)$
$q_c(\iota)=0$
$\begin{bmatrix} X_r''(\iota)+2\zeta_r A_r X_r'(\iota)+A_r^2 X_r(\iota) \\ Y_r''(\iota)+2\zeta_r A_r Y_r'(\iota)+A_r^2 Y_r(\iota) \end{bmatrix} - \begin{bmatrix} (2\pi)^2 E\cos(2\pi\iota) \\ (2\pi)^2 E\sin(2\pi\iota) \end{bmatrix} - \begin{bmatrix} (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\cos(\varphi(\iota))-\mu\sin(\varphi(\iota))) \\ (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\sin(\varphi(\iota))+\mu\cos(\varphi(\iota))) \end{bmatrix} = \vec{0}$
$\begin{bmatrix} X_s''(\iota)+2\zeta_s A_s X_s'(\iota)+A_s^2 X_s(\iota) \\ Y_s''(\iota)+2\zeta_s A_s Y_s'(\iota)+A_s^2 Y_s(\iota) \end{bmatrix} + \begin{bmatrix} (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\cos(\varphi(\iota))-\mu\sin(\varphi(\iota))) \\ (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\sin(\varphi(\iota))+\mu\cos(\varphi(\iota))) \end{bmatrix} = \vec{0}$
$X_{rs}(\iota)=X_r(\iota)-(X_s(\iota)+X_0)$
$Y_{rs}(\iota)=Y_r(\iota)-(Y_s(\iota)+Y_0)$
$\varphi(\iota)=\operatorname{atan}\left(\frac{Y_{rs}(\iota)}{X_{rs}(\iota)}\right)$
$R_{rs}(\iota)=\sqrt{X_{rs}^2(\iota)+Y_{rs}^2(\iota)}$
$\begin{bmatrix} \dot{d}(\iota) \\ v_t(\iota) \end{bmatrix} = \begin{bmatrix} 0 \\ \omega\frac{D_r}{2} \end{bmatrix} + c\frac{\omega}{2\pi}\begin{bmatrix} \cos(\varphi(\iota)) & \sin(\varphi(\iota)) \\ -\sin(\varphi(\iota)) & \cos(\varphi(\iota)) \end{bmatrix}\begin{bmatrix} X_{rs}'(\iota) \\ Y_{rs}'(\iota) \end{bmatrix}$
$D(\iota)=R_{rs}(\iota)-1$
$d(\iota)=cD(\iota)$
$f_n(\iota)=\left(ck_h D(\iota)+cc_h\frac{\omega}{2\pi}D'(\iota)\right)$
$q_t(\iota):=\begin{cases} -1 & v_t(\iota)<0 \\ 1 & v_t(\iota)>0 \\ q_t(\iota^-) & v_t(\iota)=0 \end{cases}$
$q_c(\iota):=\begin{cases} 1 & (f_n(\iota)>0)\wedge(d(\iota)\geq 0) \\ 0 & (f_n(\iota)\leq 0)\vee(d(\iota)<0) \end{cases}$

Figure 26: Rotor-Stator State Graph

In the simplified simulation system, the direction of $v_t(\iota)$ does not change. By setting convenient initial values, the variable $v_t(\iota)$ stays greater than zero, so the simplified automaton is modified as follows

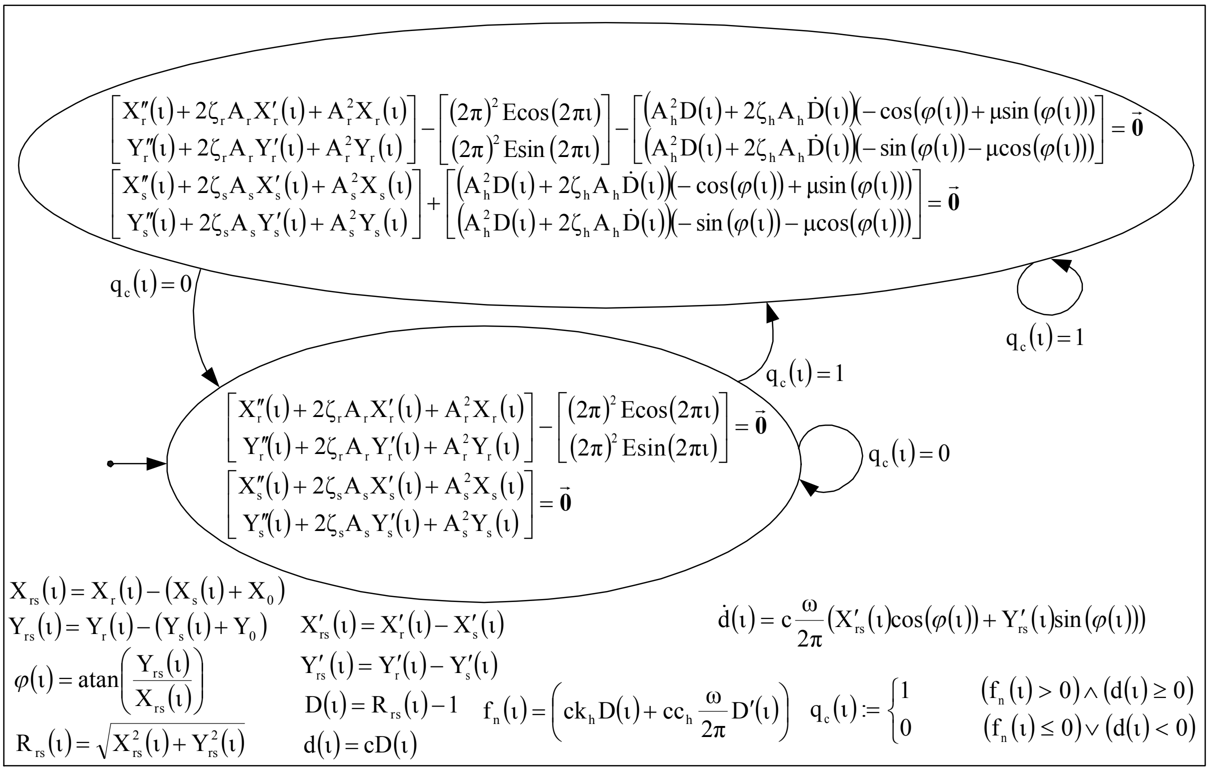
$\begin{bmatrix} X_r''(\iota)+2\zeta_r A_r X_r'(\iota)+A_r^2 X_r(\iota) \\ Y_r''(\iota)+2\zeta_r A_r Y_r'(\iota)+A_r^2 Y_r(\iota) \end{bmatrix} - \begin{bmatrix} (2\pi)^2 E\cos(2\pi\iota) \\ (2\pi)^2 E\sin(2\pi\iota) \end{bmatrix} - \begin{bmatrix} (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\cos(\varphi(\iota))+\mu\sin(\varphi(\iota))) \\ (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\sin(\varphi(\iota))-\mu\cos(\varphi(\iota))) \end{bmatrix} = \vec{\mathbf{0}}$
$\begin{bmatrix} X_s''(\iota)+2\zeta_s A_s X_s'(\iota)+A_s^2 X_s(\iota) \\ Y_s''(\iota)+2\zeta_s A_s Y_s'(\iota)+A_s^2 Y_s(\iota) \end{bmatrix} + \begin{bmatrix} (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\cos(\varphi(\iota))+\mu\sin(\varphi(\iota))) \\ (A_h^2 D(\iota)+2\zeta_h A_h \dot{D}(\iota))(-\sin(\varphi(\iota))-\mu\cos(\varphi(\iota))) \end{bmatrix} = \vec{\mathbf{0}}$
$q_c(\iota)=0$
$q_c(\iota)=1$
$q_c(\iota)=1$
$\begin{bmatrix} X_r''(\iota)+2\zeta_r A_r X_r'(\iota)+A_r^2 X_r(\iota) \\ Y_r''(\iota)+2\zeta_r A_r Y_r'(\iota)+A_r^2 Y_r(\iota) \end{bmatrix} - \begin{bmatrix} (2\pi)^2 E\cos(2\pi\iota) \\ (2\pi)^2 E\sin(2\pi\iota) \end{bmatrix} = \vec{\mathbf{0}}$
$\begin{bmatrix} X_s''(\iota)+2\zeta_s A_s X_s'(\iota)+A_s^2 X_s(\iota) \\ Y_s''(\iota)+2\zeta_s A_s Y_s'(\iota)+A_s^2 Y_s(\iota) \end{bmatrix} = \vec{\mathbf{0}}$
$q_c(\iota)=0$
$X_{rs}(\iota)=X_r(\iota)-(X_s(\iota)+X_0)$
$Y_{rs}(\iota)=Y_r(\iota)-(Y_s(\iota)+Y_0)$
$\varphi(\iota)=\operatorname{atan}\left(\frac{Y_{rs}(\iota)}{X_{rs}(\iota)}\right)$
$R_{rs}(\iota)=\sqrt{X_{rs}^2(\iota)+Y_{rs}^2(\iota)}$
$X_{rs}'(\iota)=X_r'(\iota)-X_s'(\iota)$
$Y_{rs}'(\iota)=Y_r'(\iota)-Y_s'(\iota)$
$D(\iota)=R_{rs}(\iota)-1$
$d(\iota)=cD(\iota)$
$\dot{d}(\iota)=c\frac{\omega}{2\pi}(X_{rs}'(\iota)\cos(\varphi(\iota))+Y_{rs}'(\iota)\sin(\varphi(\iota)))$
$f_n(\iota)=\left(ck_h D(\iota)+cc_h\frac{\omega}{2\pi}D'(\iota)\right)$
$q_c(\iota):=\begin{cases} 1 & (f_n(\iota)>0)\wedge(d(\iota)\geq 0) \\ 0 & (f_n(\iota)\leq 0)\vee(d(\iota)<0) \end{cases}$

Figure 27: Rotor-Stator State Graph for $v_t(\iota) >> 0$

The state vector is defined by

$$\vec{x}(\iota) := (X_r(\iota), X_r'(\iota), Y_r(\iota), Y_r'(\iota), X_s(\iota), X_s'(\iota), Y_s(\iota), Y_s'(\iota))^T .$$

The algebraic variable vector is defined by

$$\vec{y}(\iota) := (d(\iota), f_n(\iota))^T .$$

The extended variable vector is

$$\vec{x}_{ext} := (D(\iota), \dot{D}(\iota), \varphi(\iota))^T .$$

Thus, the system of differential equations of the noncontact case is established for $q_c(\iota)=0$ by (142).

$$
\begin{aligned}
&\dot{x}_1(\iota) = x_2(\iota)\\
&\dot{x}_2(\iota) = -\left(2\zeta_r A_r x_2(\iota) + A_r^2 x_1(\iota)\right) + (2\pi)^2 E\cos(2\pi\iota)\\
&\dot{x}_3(\iota) = x_4(\iota)\\
&\dot{x}_4(\iota) = -\left(2\zeta_r A_r x_4(\iota) + A_r^2 x_3(\iota)\right) + (2\pi)^2 E\sin(2\pi\iota)\\
&\dot{x}_5(\iota) = x_6(\iota)\\
&\dot{x}_6(\iota) = -\left(2\zeta_s A_s x_6(\iota) + A_s^2 x_5(\iota)\right)\\
&\dot{x}_7(\iota) = x_8(\iota)\\
&\dot{x}_8(\iota) = -\left(2\zeta_s A_s x_8(\iota) + A_s^2 x_7(\iota)\right)
\end{aligned}
$$

(142)

The system of ODEs (143) is defined for the contact simulation of rotor with stator in case of $q_c(\iota) = 1$.

$$
\begin{aligned}
&\dot{x}_1(\iota) = x_2(\iota)\\
&\dot{x}_2(\iota) = -\left(2\zeta_r A_r x_2(\iota) + A_r^2 x_1(\iota)\right) + (2\pi)^2 E\cos(2\pi\iota) + \left(A_h^2 x_{ext_1}(\iota) + 2\zeta_h A_h x_{ext_2}(\iota)\right)\left(-\cos\left(x_{ext_3}(\iota)\right) + \mu\sin\left(x_{ext_3}(\iota)\right)\right)\\
&\dot{x}_3(\iota) = x_4(\iota)\\
&\dot{x}_4(\iota) = -\left(2\zeta_r A_r x_4(\iota) + A_r^2 x_3(\iota)\right) + (2\pi)^2 E\sin(2\pi\iota) + \left(A_h^2 x_{ext_1}(\iota) + 2\zeta_h A_h x_{ext_2}(\iota)\right)\left(-\sin\left(x_{ext_3}(\iota)\right) - \mu\cos\left(x_{ext_3}(\iota)\right)\right)\\
&\dot{x}_5(\iota) = x_6(\iota)\\
&\dot{x}_6(\iota) = -\left(2\zeta_s A_s x_6(\iota) + A_s^2 x_5(\iota)\right) - \left(A_h^2 x_{ext_1}(\iota) + 2\zeta_h A_h x_{ext_2}(\iota)\right)\left(-\cos\left(x_{ext_3}(\iota)\right) + \mu\sin\left(x_{ext_3}(\iota)\right)\right)\\
&\dot{x}_7(\iota) = x_8(\iota)\\
&\dot{x}_8(\iota) = -\left(2\zeta_s A_s x_8(\iota) + A_s^2 x_7(\iota)\right) - \left(A_h^2 x_{ext_1}(\iota) + 2\zeta_h A_h x_{ext_2}(\iota)\right)\left(-\sin\left(x_{ext_3}(\iota)\right) - \mu\cos\left(x_{ext_3}(\iota)\right)\right)
\end{aligned}
$$

(143)

The switching of systems depends on the magnitudes of the constraint equations. The radial intrusion is given by

$$
y_1(\iota) = c x_{ext_1}(\iota)
$$

(144)

and the radial force[38] is

$$
y_2(\iota) = c k_h x_{ext_1}(\iota) + c c_h \frac{\omega}{2\pi} x_{ext_2}(\iota).
$$

(145)

---

[38] The dimensional radial force (124) is rewritten in formula (145) for $q_c(\iota) = 1$.

Hence, $y_1(\iota)$ and $y_2(\iota)$ are part of the switching conditions and represent the variables $d(\iota)$ and $f_n(\iota)$. The variables $x_{ext_1}(\iota)$, $x_{ext_2}(\iota)$ and $x_{ext_3}(\iota)$ are auxiliary variables.

$$x_{ext_1}(\iota)=\sqrt{(x_1(\iota)-(x_5(\iota)+x_0))^2+(x_3(\iota)-(x_7(\iota)+y_0))^2}-1 \tag{146}$$

$$x_{ext_2}(\iota)=((x_2(\iota)-x_6(\iota))\cos(x_{ext_3}(\iota))+(x_4(\iota)-x_8(\iota))\sin(x_{ext_3}(\iota))) \tag{147}$$

$$x_{ext_3}(\iota)=\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right) \tag{148}$$

The rhs. of dimensionless intrusion depth (146) can be inserted into (144). Hence the dimensional intrusion depth (144) can be written as

$$y_1(\iota)=c\left(\sqrt{(x_1(\iota)-(x_5(\iota)+x_0))^2+(x_3(\iota)-(x_7(\iota)+y_0))^2}-1\right). \tag{149}$$

The contact angle $x_{ext_3}(\iota)$ in equation (147) can be substituted with its equivalent in (148). Then the dimensionless intrusion depth x_{ext_1} and radial impact velocity x_{ext_2} in (145) can be substituted with their equivalents in (146) and (147), which results in

$$y_2(\iota)=ck_h x_{ext_1}+cc_h\frac{\omega}{2\pi}\left((x_2(\iota)-x_6(\iota))\cos\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right)+(x_4(\iota)-x_8(\iota))\sin\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right)\right). \tag{150}$$

5.3.2 Reformulation of Explicit Euler's Formula

For approximation of the event location for a transition from the noncontact case to the contact case, the transient condition $(y_2(\iota)>0)\wedge(y_1(\iota)\geq 0)$ has to be valid. Thus, the derivative of the radial indentation (149) is established by the chain rule, which is demonstrated in appendix A2.1. This transformation is shown as follows

$$\dot{y}_1(\iota) = c\frac{(x_1(\iota)-(x_5(\iota)+x_0))(x_2(\iota)-x_6(\iota))+(x_3(\iota)-(x_7(\iota)+y_0))(x_4(\iota)-x_8(\iota))}{\left(\sqrt{(x_1(\iota)-(x_5(\iota)+x_0))^2+(x_3(\iota)-(x_7(\iota)+y_0))^2}\right)}.$$

(151)

The transformation of radial force (150) to an ODE is demonstrated by (152). This transformation is illustrated in A2.2.

$$\begin{aligned}
\dot{y}_2(\iota) = {} & ck_h \frac{(x_1(\iota)-(x_5(\iota)+x_0))(x_2(\iota)-x_6(\iota))+(x_3(\iota)-(x_7(\iota)+y_0))(x_4(\iota)-x_8(\iota))}{\left(\sqrt{(x_1(\iota)-(x_5(\iota)+x_0))^2+(x_3(\iota)-(x_7(\iota)+y_0))^2}\right)} \\
& + cc_h \frac{\omega}{2\pi}\Bigg[\Big(\big(-(2\zeta_r A_r x_2(\iota)+A_r^2 x_1(\iota))+(2\pi)^2 E\cos(2\pi\iota)\big)+(2\zeta_s A_s x_6(\iota)+A_s^2 x_5(\iota))\Big)\cos\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right) \\
& + \Big(\big(-(2\zeta_r A_r x_4(\iota)+A_r^2 x_3(\iota))+(2\pi)^2 E\sin(2\pi\iota)\big)+(2\zeta_s A_s x_8(\iota)+A_s^2 x_7(\iota))\Big)\sin\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right) \\
& + \frac{(x_4(\iota)-x_8(\iota))(x_1(\iota)-(x_5(\iota)+x_0))-(x_2(\iota)-x_6(\iota))(x_3(\iota)-(x_7(\iota)+y_0))}{(x_1(\iota)-(x_5(\iota)+x_0))^2}\frac{1}{\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)^2+1} \\
& \left(-(x_2(\iota)-x_6(\iota))\sin\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right)+(x_4(\iota)-x_8(\iota))\cos\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right)\right)\Bigg]
\end{aligned}$$

(152)

The radial force $y_2(\iota)$ and the radial distance $y_1(\iota)$ whereas $(y_2(\iota)>0)\wedge(y_1(\iota)\geq 0)$ do not cross their threshold values synchronously. In the simulation, the variable $y_2(\iota)$ triggers after $y_1(\iota)$. Thus at the first switching the transient condition $(y_2(\iota)>0)\wedge(y_1(\iota)\geq 0)$ is sensitive on condition $y_2(\iota)>0$ if $y_1(\iota)\geq 0$ is fulfilled. Then the event location on t-axis for the first transition is given via the discrete form of ODE (152) according to formula (15) as follows

$$\tilde{\iota}_e = \iota_{j-1} + \left|\frac{-\hat{y}_{2,j-1}}{f_{y_2}\left(\iota_{j-1}, \hat{\vec{x}}_{j-1}\right)}\right|.$$

(153)

In formula (153) $f_{y_2}\left(\iota_{j-1}, \hat{\vec{x}}_{j-1}\right)$ is the discrete form of the rhs. of the derivative (152) given by Euler's algorithm, which is shown as follows

$$
\begin{aligned}
f_{y_2}\left(\iota_{j-1}, \hat{\vec{x}}_{j-1}\right) = {} & ck_h \frac{\left(\hat{x}_{1,j-1} - \left(\hat{x}_{5,j-1} + x_0\right)\right)\left(\hat{x}_{2,j-1} - \hat{x}_{6,j-1}\right) + \left(\hat{x}_{3,j-1} - \left(\hat{x}_{7,j-1} + y_0\right)\right)\left(\hat{x}_{4,j-1} - \hat{x}_{8,j-1}\right)}{\left(\sqrt{\left(\hat{x}_{1,j-1} - \left(\hat{x}_{5,j-1} + x_0\right)\right)^2 + \left(\hat{x}_{3,j-1} - \left(\hat{x}_{7,j-1} + y_0\right)\right)^2}\right)} \\
& + cc_h \frac{\omega}{2\pi}\left[\left(\left(-\left(2\zeta_r A_r \hat{x}_{2,j-1} + A_r^2 \hat{x}_{1,j-1}\right) + (2\pi)^2 E\cos\left(2\pi\iota_{j-1}\right)\right) + \left(2\zeta_s A_s \hat{x}_{6,j-1} + A_s^2 \hat{x}_{5,j-1}\right)\right)\cos\left(\operatorname{atan}\left(\frac{\hat{x}_{3,j-1} - \left(\hat{x}_{7,j-1} + y_0\right)}{\hat{x}_{1,j-1} - \left(\hat{x}_{5,j-1} + x_0\right)}\right)\right)\right. \\
& + \left(\left(-\left(2\zeta_r A_r \hat{x}_{4,j-1} + A_r^2 \hat{x}_{3,j-1}\right) + (2\pi)^2 E\sin\left(2\pi\iota_{j-1}\right)\right) + \left(2\zeta_s A_s \hat{x}_{8,j-1} + A_s^2 \hat{x}_{7,j-1}\right)\right)\sin\left(\operatorname{atan}\left(\frac{\hat{x}_{3,j-1} - \left(\hat{x}_{7,j-1} + y_0\right)}{\hat{x}_{1,j-1} - \left(\hat{x}_{5,j-1} + x_0\right)}\right)\right) \\
& + \frac{\left(\hat{x}_{4,j-1} - \hat{x}_{8,j-1}\right)\left(\hat{x}_{1,j-1} - \left(\hat{x}_{5,j-1} + x_0\right)\right) - \left(\hat{x}_{2,j-1} - \hat{x}_{6,j-1}\right)\left(\hat{x}_{3,j-1} - \left(\hat{x}_{7,j-1} + y_0\right)\right)}{\left(\hat{x}_{1,j-1} - \left(\hat{x}_{5,j-1} + x_0\right)\right)^2} \frac{1}{\left(\frac{\hat{x}_{3,j-1} - \left(\hat{x}_{7,j-1} + y_0\right)}{\hat{x}_{1,j-1} - \left(\hat{x}_{5,j-1} + x_0\right)}\right)^2 + 1} \\
& \left.\left(-\left(x_2(\iota) - x_6(\iota)\right)\sin\left(\operatorname{atan}\left(\frac{\hat{x}_{3,j-1} - \left(\hat{x}_{7,j-1} + y_0\right)}{\hat{x}_{1,j-1} - \left(\hat{x}_{5,j-1} + x_0\right)}\right)\right) + \left(x_4(\iota) - x_8(\iota)\right)\cos\left(\operatorname{atan}\left(\frac{\hat{x}_{3,j-1} - \left(\hat{x}_{7,j-1} + y_0\right)}{\hat{x}_{1,j-1} - \left(\hat{x}_{5,j-1} + x_0\right)}\right)\right)\right)\right].
\end{aligned}
$$

(154)

For the second event, a state transition from the contact case to the noncontact case, depends on the disjunction condition $\left(y_2(\iota) \leq 0\right) \vee \left(y_1(\iota) < 0\right)$. In this case, an event occurs if the variables $y_2(\iota)$ or $y_1(\iota)$ crosses the threshold value of zero. These are also not synchronous events. Here, the variable $y_1(\iota)$ intersects with the threshold value before $y_2(\iota)$. Hence, t_e in this case is computed according to the formula (15) for ODE (151) as follows

$$
\tilde{\iota}_e = \iota_{j-1} + \left| \frac{\hat{y}_{1,j-1}\left(\sqrt{\left(\hat{x}_{1,j-1} - \left(\hat{x}_{5,j-1} + x_0\right)\right)^2 + \left(\hat{x}_{3,j-1} - \left(\hat{x}_{7,j-1} + y_0\right)\right)^2}\right)}{c\left(\left(\hat{x}_{1,j-1} - \left(\hat{x}_{5,j-1} + x_0\right)\right)\left(\hat{x}_{2,j-1} - \hat{x}_{6,j-1}\right) + \left(\hat{x}_{3,j-1} - \left(\hat{x}_{7,j-1} + y_0\right)\right)\left(\hat{x}_{4,j-1} - \hat{x}_{8,j-1}\right)\right)} \right|.
$$

(155)

Thus, at the second event the variable $y_2(\iota)$ does not trigger the system transition. However the transformation of constraint equation (150) to its appropriate ODE, for determination of $y_2(\iota)$ regarding a state transition from contact case to the noncontact case, is demonstrated in A2.2 by ODE (168).

5.3.3 Reformulation of Implicit Trapezodial Formula for Newton's Method

An approximation of the transient vector for a transition from the noncontact state to the contact state is processed after the crossing of the threshold value. Thus, at state event, the event variable is set equal to its threshold value $\tilde{y}_{2,j} := a_e = 0$. The system of event equations[39] is established according to system (37) for reformulated trapezoidal formula by event vector function (156). In order to compute the prediction values, Newton's method[40] is used as shown in (38).

$$\hat{\tilde{h}}\left(\Delta\tilde{\iota}_{e,k}, \tilde{\vec{x}}_k, \tilde{y}_1\right) = \begin{pmatrix} -\tilde{x}_{1,k} + \hat{x}_{1,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(\hat{x}_{2,j-1} + \tilde{x}_{2,k}\right) \\ -\tilde{x}_{2,k} + \hat{x}_{2,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(\left(-\left(2\zeta_r A_r \hat{x}_{2,j-1} + A_r^2 \hat{x}_{1,j-1}\right) + (2\pi)^2 E\cos\left(2\pi\iota_{j-1}\right)\right) + \left(-\left(2\zeta_r A_r \tilde{x}_{2,k} + A_r^2 \tilde{x}_{1,k}\right) + (2\pi)^2 E\cos\left(2\pi\left(\iota_{j-1} + \Delta\tilde{t}_{e,k}\right)\right)\right)\right) \\ -\tilde{x}_{3,j} + \hat{x}_{3,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(\hat{x}_{4,j-1} + \tilde{x}_{4,k}\right) \\ -\tilde{x}_{4,k} + \hat{x}_{4,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(\left(-\left(2\zeta_r A_r \hat{x}_{4,j-1} + A_r^2 \hat{x}_{3,j-1}\right) + (2\pi)^2 E\sin\left(2\pi\iota_{j-1}\right)\right) + \left(-\left(2\zeta_r A_r \tilde{x}_{4,k} + A_r^2 \tilde{x}_{3,k}\right) + (2\pi)^2 E\sin\left(2\pi\left(\iota_{j-1} + \Delta\tilde{t}_{e,k}\right)\right)\right)\right) \\ -\tilde{x}_{5,k} + \hat{x}_{5,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(\hat{x}_{6,j-1} + \tilde{x}_{6,k}\right) \\ -\tilde{x}_{6,k} + \hat{x}_{6,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(-\left(2\zeta_s A_s \hat{x}_{6,j-1} + A_s^2 \hat{x}_{5,j-1}\right) - \left(2\zeta_s A_s \tilde{x}_{6,k} + A_s^2 \tilde{x}_{5,k}\right)\right) \\ -\tilde{x}_{7,k} + \hat{x}_{7,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(\hat{x}_{8,j-1} + \tilde{x}_{8,k}\right) \\ -\tilde{x}_{8,k} + \hat{x}_{8,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(-\left(2\zeta_s A_s \hat{x}_{8,j-1} + A_s^2 \hat{x}_{7,j-1}\right) - \left(2\zeta_s A_s \tilde{x}_{8,k} + A_s^2 \tilde{x}_{7,k}\right)\right) \\ -\tilde{y}_{1,k} + c\left(\sqrt{\left(\tilde{x}_{1,k} - \left(\tilde{x}_{5,k} + x_0\right)\right)^2 + \left(\tilde{x}_{3,k} - \left(\tilde{x}_{7,k} + y_0\right)\right)^2} - 1\right) \\ -a_e + k_h c\left(\sqrt{\left(\tilde{x}_{1,k} - \left(\tilde{x}_{5,k} + x_0\right)\right)^2 + \left(\tilde{x}_{3,k} - \left(\tilde{x}_{7,k} + y_0\right)\right)^2} - 1\right) + cc_h \frac{\omega}{2\pi}\left(\left(\tilde{x}_{2,k} - \tilde{x}_{6,k}\right)\cos\left(\operatorname{atan}\left(\tilde{x}_{ext_3,k}\right)\right) + \left(\tilde{x}_{4,k} - \tilde{x}_{8,k}\right)\sin\left(\operatorname{atan}\left(\tilde{x}_{ext_3,k}\right)\right)\right) \end{pmatrix}$$

(156)

Afterwards the transient vector is approximated using the components of the prediction vector

$$\left(\tilde{t}_e, \tilde{\vec{x}}_e^T, \tilde{\vec{y}}_e^T\right)^T := \left(t_{j-1} + \Delta\tilde{t}_k, \tilde{\vec{x}}_k^T, \tilde{y}_{1,k}, a_e\right)^T$$

according to formula (39).

---

[39] For computation of event location at the state event, the event vector function is reformulated as a system of equation.

[40] System (156) is written in short forms with variable $\tilde{x}_{ext_3,k}$, hence for numerical processing of (156) this variable must be replaced with its equivalent $\tilde{x}_{ext_3,k} := \operatorname{atan}\left(\frac{\tilde{x}_{3,k} - \left(\tilde{x}_{7,k} + y_0\right)}{\tilde{x}_{1,k} - \left(\tilde{x}_{5,k} + x_0\right)}\right)$.

For a state vector prediction of a transition from the contact case to the noncontact case, the system of equation (157) is applied.

$$\hat{\vec{h}}\left(\Delta\tilde{\iota}_k,\tilde{\vec{x}}_k,\tilde{y}_{2,k}\right)=\left(\hat{h}_1\left(\Delta\tilde{\iota}_k,\tilde{\vec{x}}_k,\tilde{y}_{2,k}\right),\hat{h}_2\left(\Delta\tilde{\iota}_k,\tilde{\vec{x}}_k,\tilde{y}_{2,k}\right),\ldots,\hat{h}_8\left(\Delta\tilde{\iota}_k,\tilde{\vec{x}}_k,\tilde{y}_{2,k}\right),\hat{h}_{y_1}\left(\Delta\tilde{\iota}_k,\tilde{\vec{x}}_k,\tilde{y}_{2,k}\right),\hat{h}_{y_2}\left(\Delta\tilde{\iota}_k,\tilde{\vec{x}}_k,\tilde{y}_{2,k}\right)\right)^T=\vec{0}$$

(157)

The variable $\tilde{y}_{1,j}$ in (157) is replaced with its threshold value $\tilde{y}_{1,j}:=a_e=0$. The prediction of root location of the system of equations (157) is done with the Newton's method (38). The elements of (157) are defined in A2.3.

After the prediction phase, the approximation of the transient vector is realized by computing trapezoidal method (39) using the predicted magnitudes.

5.3.4 Henon's Method

The semi-explicit DAEs of the rotor-stator are transformed to ODEs via the derivatives of both algebraic equations (149) and (150). These transformations are demonstrated by ODEs (151), (152) and (168).

According to the transition condition explained in 5.3.2, the variable $y_2(\iota)$ influences the structure transition from the first noncontact case to the contact case. Hence, the variable y_2 is set as an independent variable and the independent variable $\iota(y_2)$ as dependent. The Henon's transformation is given according to the presented method in subchapter 4.2 for the rotor-stator noncontact to contact case with

$$\Gamma=\frac{1}{f_{y2}\left(\iota(y_2),\vec{x}(y_2)\right)}$$

as follows

$$
\begin{aligned}
\frac{d}{dy_2}x_1(y_2) &= \frac{x_2(y_2)}{f_{y2}(\imath(y_2),\vec{x}(y_2))}\\
\frac{d}{dy_2}x_2(y_2) &= \frac{-\left(2\zeta_r A_r x_2(y_2)+A_r^2 x_1(y_2)\right)+(2\pi)^2 E\cos(2\pi\imath(y_2))}{f_{y2}(\imath(y_2),\vec{x}(y_2))}\\
\frac{d}{dy_2}x_3(y_2) &= \frac{x_4(y_2)}{f_{y2}(\imath(y_2),\vec{x}(y_2))}\\
\frac{d}{dy_2}x_4(y_2) &= \frac{-\left(2\zeta_r A_r x_4(y_2)+A_r^2 x_3(y_2)\right)+(2\pi)^2 E\sin(2\pi\imath(y_2))}{f_{y2}(\imath(y_2),\vec{x}(y_2))}\\
\frac{d}{dy_2}x_5(y_2) &= \frac{x_6(y_2)}{f_{y2}(\imath(y_2),\vec{x}(y_2))}\\
\frac{d}{dy_2}x_6(y_2) &= \frac{-\left(2\zeta_s A_s x_6(y_2)+A_s^2 x_5(y_2)\right)}{f_{y2}(\imath(y_2),\vec{x}(y_2))}\\
\frac{d}{dy_2}x_7(y_2) &= \frac{x_8(y_2)}{f_{y2}(\imath(y_2),\vec{x}(y_2))}\\
\frac{d}{dy_2}x_8(y_2) &= \frac{-\left(2\zeta_s A_s x_8(y_2)+A_s^2 x_7(y_2)\right)}{f_{y2}(\imath(y_2),\vec{x}(y_2))}\\
\frac{d}{dy_2}y_1(y_2) &= \frac{f_{y_1}(\vec{x}(y_2))}{f_{y2}(\imath(y_2),\vec{x}(y_2))}\\
\frac{d}{dy_2}\imath(y_2) &= \frac{1}{f_{y2}(\imath(y_2),\vec{x}(y_2))}.
\end{aligned}
\tag{158}
$$

In system of ODEs (158), $f_{y2}(\imath(y_2),\vec{x}(y_2))$ stands for the rhs. of ODE (152) with independent variable y_2. This term is shown as follows

$$
\begin{aligned}
f_{y2}(\imath(y_2),\vec{x}(y_2)) = {} & ck_h \frac{\left(x_1(y_2)-\left(x_5(y_2)+x_0\right)\right)\left(x_2(y_2)-x_6(y_2)\right)+\left(x_3(y_2)-\left(x_7(y_2)+y_0\right)\right)\left(x_4(y_2)-x_8(y_2)\right)}{\left(\sqrt{\left(x_1(y_2)-\left(x_5(y_2)+x_0\right)\right)^2+\left(x_3(y_2)-\left(x_7(y_2)+y_0\right)\right)^2}\right)}\\
& + cc_h\frac{\omega}{2\pi}\Bigg[\left(\left(-\left(2\zeta_r A_r x_2(y_2)+A_r^2 x_1(y_2)\right)+(2\pi)^2 E\cos(2\pi\imath)\right)+\left(2\zeta_s A_s x_6(y_2)+A_s^2 x_5(y_2)\right)\right)\cos\left(\operatorname{atan}\left(\frac{x_3(y_2)-\left(x_7(y_2)+y_0\right)}{x_1(y_2)-\left(x_5(y_2)+x_0\right)}\right)\right)\\
& + \left(\left(-\left(2\zeta_r A_r x_4(y_2)+A_r^2 x_3(y_2)\right)+(2\pi)^2 E\sin(2\pi\imath)\right)+\left(2\zeta_s A_s x_8(y_2)+A_s^2 x_7(y_2)\right)\right)\sin\left(\operatorname{atan}\left(\frac{x_3(y_2)-\left(x_7(y_2)+y_0\right)}{x_1(y_2)-\left(x_5(y_2)+x_0\right)}\right)\right)\\
& + \frac{\left(x_4(y_2)-x_8(y_2)\right)\left(x_1(y_2)-\left(x_5(y_2)+x_0\right)\right)-\left(x_2(y_2)-x_6(y_2)\right)\left(x_3(y_2)-\left(x_7(y_2)+y_0\right)\right)}{\left(x_1(y_2)-\left(x_5(y_2)+x_0\right)\right)^2}\frac{1}{\left(\frac{x_3(y_2)-\left(x_7(y_2)+y_0\right)}{x_1(y_2)-\left(x_5(y_2)+x_0\right)}\right)^2+1}\\
& \left(-\left(x_2(y_2)-x_6(y_2)\right)\sin\left(\operatorname{atan}\left(\frac{x_3(y_2)-\left(x_7(y_2)+y_0\right)}{x_1(y_2)-\left(x_5(y_2)+x_0\right)}\right)\right)+\left(x_4(y_2)-x_8(y_2)\right)\cos\left(\operatorname{atan}\left(\frac{x_3(y_2)-\left(x_7(y_2)+y_0\right)}{x_1(y_2)-\left(x_5(y_2)+x_0\right)}\right)\right)\right)\Bigg].
\end{aligned}
\tag{159}
$$

The rhs. of ODE (151) is defined by formula (160) for ODE system (158) as follows

$$f_{y_1}(\vec{x}(y_2)) = c\frac{(x_1(y_2)-(x_5(y_2)+x_0))(x_2(y_2)-x_6(y_2))+(x_3(y_2)-(x_7(y_2)+y_0))(x_4(y_2)-x_8(y_2))}{\left(\sqrt{(x_1(y_2)-(x_5(y_2)+x_0))^2+(x_3(y_2)-(x_7(y_2)+y_0))^2}\right)}.$$

(160)

The independent variable of the ODE system (158) is $\tau := y_2$ and the dependent variable is defined by $\chi_{10}(\tau) := \iota(\tau)$, hence the state vector in Henon's transformed system is

$$\vec{\chi}(\tau) := (x_1(\tau), x_2(\tau), x_3(\tau), x_4(\tau), x_5(\tau), x_6(\tau), x_7(\tau), x_8(\tau), y_1(\tau), \iota(\tau))^T.$$

The computation of $\tilde{\iota}_e$ is given according to formula (67) with $a_e := 0$ by ODE45 applied on system of ODEs in (158). In the next stage, the approximated transient state vector is computed on interval $\lfloor \iota_{j-1}, \tilde{\iota}_e \rfloor$ using solver algorithm ODE45.

At the second event, a transition from the contact case to the noncontact case depends on the event variable y_1. Hence, the rhs. of the ODE (151) is applied for setting

$$\Gamma = \frac{1}{f_{y_1}(\vec{x}(y_1))}.$$

In this case, the variable y_1 is realized as independent variable and the variable $\iota(y_1)$ as dependent variable. The second event is triggered by the disjunction condition $(y_2(\iota) \leq 0) \vee (y_1(\iota) < 0)$ and the variable $y_1(\iota)$ fulfilled its conditional relation earlier than $y_2(\iota)$. The Henon's transformation is given according to the method described in subchapter 4.2 as follows

$$
\begin{aligned}
&\frac{d}{dy_1}x_1(y_1)=\frac{x_2(y_1)}{f_{y_1}(\vec{x}(y_1))}\\
&\frac{d}{dy_1}x_2(y_1)=\frac{-\left(2\zeta_r A_r x_2(y_1)+A_r^2 x_1(y_1)\right)+(2\pi)^2 E\cos\left(2\pi\iota(y_1)+\left(A_h^2 x_{ext_1}(y_1)+2\zeta_h A_h x_{ext_2}(y_1)\right)\left(-\cos\left(x_{ext_3}(y_1)\right)+\mu\sin\left(x_{ext_3}(y_1)\right)\right)\right)}{f_{y_1}(\vec{x}(y_1))}\\
&\frac{d}{dy_1}x_3(y_1)=\frac{x_4(y_1)}{f_{y_1}(\vec{x}(y_1))}\\
&\frac{d}{dy_1}x_4(y_1)=\frac{-\left(2\zeta_r A_r x_4(y_1)+A_r^2 x_3(y_1)\right)+(2\pi)^2 E\sin(2\pi\iota(y_1))+\left(A_h^2 x_{ext_1}(y_1)+2\zeta_h A_h x_{ext_2}(y_1)\right)\left(-\sin\left(x_{ext_3}(y_1)\right)-\mu\cos\left(x_{ext_3}(y_1)\right)\right)}{f_{y_1}(\vec{x}(y_1))}\\
&\frac{d}{dy_1}x_5(y_1)=\frac{x_6(y_1)}{f_{y_1}(\vec{x}(y_1))}\\
&\frac{d}{dy_1}x_6(y_1)=\frac{-\left(2\zeta_s A_s x_6(y_1)+A_s^2 x_5(y_1)\right)-\left(A_h^2 x_{ext_1}(y_1)+2\zeta_h A_h x_{ext_2}(y_1)\right)\left(-\cos\left(x_{ext_3}(y_1)\right)+\mu\sin\left(x_{ext_3}(y_1)\right)\right)}{f_{y_1}(\vec{x}(y_1))}\\
&\frac{d}{dy_1}x_7(y_1)=\frac{x_8(y_1)}{f_{y_1}(\vec{x}(y_1))}\\
&\frac{d}{dy_1}x_8(y_1)=\frac{-\left(2\zeta_s A_s x_8(y_1)+A_s^2 x_7(y_1)\right)-\left(A_h^2 x_{ext_1}(y_1)+2\zeta_h A_h x_{ext_2}(y_1)\right)\left(-\sin\left(x_{ext_3}(y_1)\right)-\mu\cos\left(x_{ext_3}(y_1)\right)\right)}{f_{y_1}(\vec{x}(y_1))}\\
&\frac{d}{dy_1}\iota(y_1)=\frac{1}{f_{y_1}(\vec{x}(y_1))}\\
&\frac{d}{dy_1}y_2(y_1)=\frac{f_{y2}(\iota(y_1),\vec{x}(y_1))}{f_{y_1}(\vec{x}(y_1))}.
\end{aligned}
\tag{161}
$$

The structure of $f_{y2}(\iota(y_1)\vec{x}(y_1))$ is the same as the rhs. of the ODE (168) (appendix A2.2). The computation process of ODEs (161) is given for interval $\left\lfloor \hat{\tau}_{j\text{-}1}, \hat{\tau}_{j-1}+\left|\hat{\tau}_{j-1}-a_e\right|\right\rfloor$ with $a_e=0$ and $\hat{\tau}_{j-1}:=\hat{y}_{j-1}$. The system of ODEs (161) is treated by ODE45 according to initial values of (67) with the independent variable $\tau:=y_1$ by state vector

$$\vec{\chi}(\tau):=\left(x_1(\tau),x_2(\tau),x_3(\tau),x_4(\tau),x_5(\tau),x_6(\tau),x_7(\tau),x_8(\tau),\iota(\tau),y_2(\tau)\right)^T.$$

In the next stage, the approximation of the transient state vector regarding the interval $\left\lfloor \iota_{j\text{-}1}, \iota_{j-1}+\left|\iota_{j-1}-\iota_e\right|\right\rfloor$ is carried out by the solver ODE45.

5.3.5 Simulation Comparisions

For simulations, the hypothetical parameters of the rotor-stator contact model are defined in Table 10.

Parameter	**Description**
$x_0 := 0.0239\text{m}$	offset between rotor and stator in x-direction
$y_0 := 0.0000001\text{m}$	offset between rotor and stator in y-direction
$D_r := 2\text{m}$	rotor diameter
$e := 0.00999\text{m}$	eccentricity radial offset of rotor
$c := 0.024\text{m}$	radial clearance between rotor and stator
$m_r := 8\text{kg}$	mass of rotor
$m_s := 20\text{kg}$	mass of stator
$c_r := 150\,\text{N}\cdot\text{s/m}$	rotor damping coefficient
$k_r := 800000\,\text{N/m}$	rotor stiffness coefficient
$c_s := 20000\,\text{N}\cdot\text{s/m}$	stator damping coefficient
$k_s := 700000\,\text{N/m}$	stator stiffness coefficient
$c_h := 7\,\text{N}\cdot\text{s/m}$	contact damping coefficient
$k_h := 100000\,\text{N/m}$	contact stiffness coefficient
$\omega := 101\,\text{rad}/s$	angular velocity
$\mu := 0.002$	friction coefficient

Table 10: Rotor-Stator Simulation Parameters

The simulation of the rotor-stator model provides the following results, which are shown in Figures 28, 29, 30, 31.

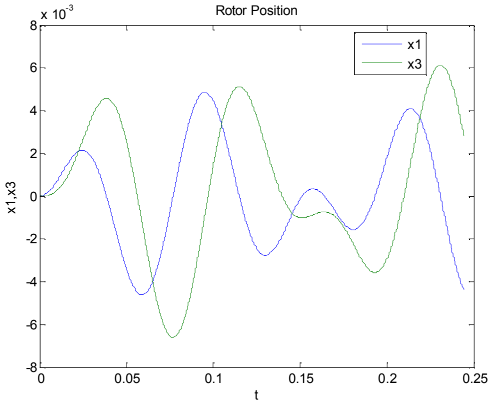

Figure 28: Simulation of Rotor Position

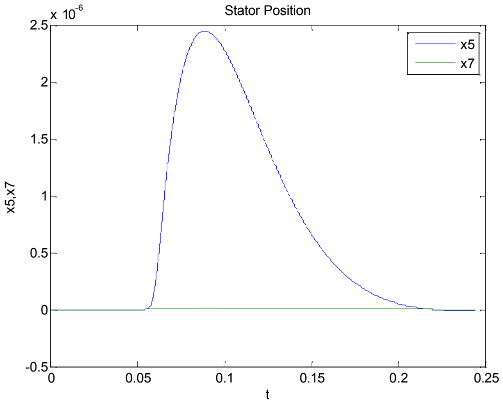

Figure 29: Simulation of Stator Position

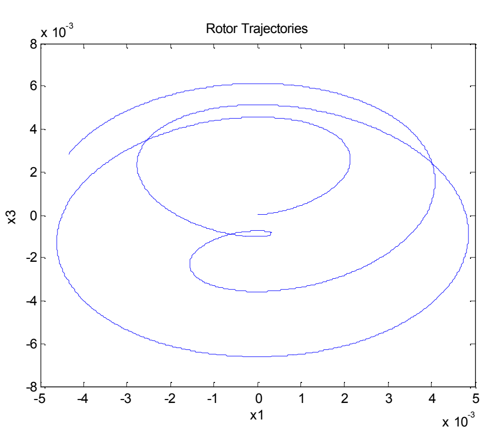

Figure 30: Simulation of Rotor Trajectory

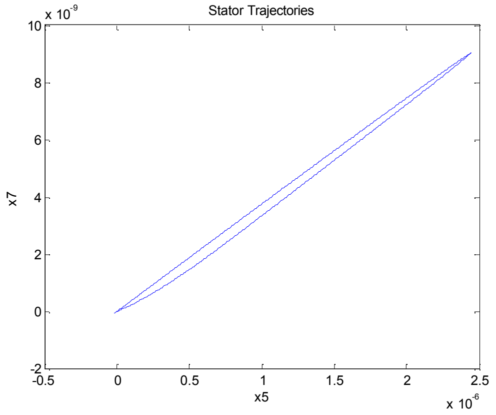

Figure 31: Simulation of Stator Trajectory

Figures 28 and 29 show the rotor and stator positions along x- and y-axes and Figure 30 and 31 show the rotor and stator trajectories.

The simulation comparisons are demonstrated in Table 11.

Methods	**Reformulation of Explicit Euler's Formula**	**Reformulation of Implicit Trapezoidal Formula for Newton's Method**	**Henon**
Solver Algorithm	Implemented Euler's Solver	Implemented Trapezoidal Solver Mixed Newton Method	Programmed Henon Using Matlab ODE45
Type	Explicit	Implicit	Explicit
Programming Environment	Matlab	Matlab	Matlab
Threshold Value a_e	0	0	0
$y_2\left(\tilde{t}_e\right)$	0	7.216449e-016	-8.816886e-009
Local Error of $y_2\left(\tilde{t}_e\right)$	0	7.216449e-016	8.816886e-009
Fixed Step-Size	1e-005	1e-005	1e-005
$\Delta\tilde{t}_e$	5.455252e-006	1.894331e-006	7.897663e-006
$\tilde{t}_e$	5.459545e-002	5.460189e-002	5.459789e-002

Methods	**Simulink**	**Adaptive (Matlab)**
Solver Algorithm	Simulink ODE4	Matlab ODE45
Type	Explicit	Explicit
Programming Environment	Simulink	Matlab
Threshold Value a_e	0	0
$y_2\left(\tilde{t}_e\right)$	[-5.340435e-002, 1.085920e-002]	6.505906e-014
Local Error of $y_2\left(\tilde{t}_e\right)$	-	6.505906e-014
Fixed Step-Size	1e-005	1e-005
$\Delta\tilde{t}_e$	-	7.896551e-006
$\tilde{t}_e$	[5.459e-002, 5.46e-002]	5.459789e-002

Table 11: First Event Comparisons of Rotor-Stator Simulations at the Transition from Noncontact Operating to Contact State

The simulation results in Table 12 show local errors for approximated variables involved in first and second events.

	Methods			
	Reformulation of Explicit Euler's Formula	**Reformulation of Implicit Trapezoidal Formula for Newton's Method**	**Henon**	**Adaptive Matlab**
Error $\left\|y_2\left(\tilde{t}_e\right)\right\|$ **at First Event**	0	7.216449e-16	8.816886e-09	6.505906e-14
Error $\left\|y_1\left(\tilde{t}_e\right)\right\|$ **at Second Event**	2.346380e-20	0	5.918564e-12	0

Table 12: Errors of Computed Algebraic Variables $y_1\left(\tilde{t}_e\right)$ and $y_1\left(\tilde{t}_e\right)$ Involved in the First and Second Events in Hybrid System of Rotor-Stator

Table 13 shows the summary of simulations, which are implemented related to different types of systems.

Systems	**Reformulation of (Explicit Euler) Solver Formula**	**Reformulation of (Implicit Trapezoidal) Solver Formula for Newton's Method**	**Henon (Using ODE45)**
Autonomous ODEs	Bouncing Ball with ODEs	Bouncing Ball with ODEs	Bouncing Ball with ODEs
Non-Autonomous ODEs	Rotor-Stator with DAEs transformed in ODEs	-	Rotor-Stator with DAEs transformed in ODEs
Semi-Explicit Autonomous DAEs	Filament Pendulum using Index-Reduction	Filament Pendulum Directly on Semi-Explicit DAEs[41]	Filament Pendulum using Index-Reduction
Semi-Explicit Non-Autonomous DAEs	Rotor-Stator using Index-Reduction	Rotor-Stator Directly on Semi-Explicit DAEs	Rotor-Stator using Index-Reduction

Table 13: Summary of Different Simulations Related to System Types

---

[41] Reformulation of the implicit solver for root-finding method is applied on the semi-explicit DAEs without an index reduction procedure. In this method, the root-finding system outputs should be converged in root location.

Table 14 shows the programming cost of the different algorithms. The root-finding method contains the solver algorithm, Newton’s method with Jacobian matrix and shows higher implementation cost than other algorithms.

Subjects	**Reformulation of (Explicit Euler) Solver Formula**	**Reformulation of (Implicit Trapezoidal) Solver Formula for Newton’s Method**	**Henon’s Method**	**Adaptive**	**Without Zero-Crossing**
Programming Cost	Low for simple models.	High	-	-	Low

Table 14: Implementation Costs of Different Event Location Approximation Methods

Table 15 demonstrates the approximated execution times of different simulations from start until the end of the approximation procedures of the transient vectors at the first events.

	Simulations		
Methods	**Bouncing Ball**	**Filament Pendulum**	**Rotor-Stator**
Reformulation of Explicit Euler’s Formula	1.936829e-01	8.841572e+01	2.873217
Reformulation of Implicit Trapezoidal Formula for Newton’s Method	3.065886	1.225275e+02	5.889020
Henon (ODE45 Matlab)	1.080181e+01	1.189660e+01	1.531918
Adaptive (ODE45 Matlab)	8.599733	1.142153e+01	1.095217
Without Zero-Crossing (ODE4 Simulink)	4.399873e+01	2.122344e+01	4.391379

Table 15: Comparisons of Approximated Execution Time for First Event

Table 16 shows certain system benchmarks of different approximation methods of transient vectors.

Methods	Advantage	Disadvantage
Reformulation of Solver Formula	• Low Local Error. • Simple Implementation for Simple Solver.	• Simple Solver Induce often High Global Error Magnitudes. • Index-Reduction Procedure for DAEs Is Inevitable. • Instability if Functional Part of the Solver in Reformulated Formula Is Singular.
Reformulation of Solver Formula for Root-Finding Methods	• Low Local Error. • Handling of Semi-Explicit DAEs without Index-Reduction.	• High Implementation Cost. • Root-Finding Method May not Converge. • Instability of Newton's Method if Jacobian Matrix Is Singular. • High Execution Time for some Applications
Henon (ODE45 Matlab)	Applicable on Embedded Solvers.	• Instability if Denominator of Γ Is Singular. • Restricted Application for Non-autonomous System[42]. • Index-Reduction Procedure for DAEs Should be Given.

Table 16: Advantages and Disadvantages of Various Methods

---

[42] If the range of the integration of the ODE in Henon's transformation is not equivalent to the domain of appropriate ODE involved in state event, then the Henon's method shows an inaccurate result. This may be given for non-autonomous systems. The behavior of the function of the event location approximation procedure at the event environment should be one to one and onto and it should be continuous on the approximation interval.

6 Conclusion

A problem of hybrid and variable structure dynamic systems is the precise approximation of the transient state vector, algebraic variables and event location at the system-switching event.

This dissertation seeks to find answers to the problem of approximation of transient state vectors, algebraic vectors and event locations of the hybrid and variable structure systems of ODEs and/or DAEs at the state events or system switching. The main aims of this thesis are research and development of methods and algorithms, which are characterized as approximate solutions of hybrid or variable structure systems at the surface of crossing sections. The approximated solutions can be used for reduction of local errors in hybrid and variable structure systems at the switching event.

The results of this research can be classified in two categories, “reformulation of solver formula” and “reformulation of solver formula for root-finding methods”. Another focus is on the “Henon’s method”. These methods are developed on certain explicit or implicit solver algorithms.

The main outcome of this thesis is as follows:

- The “reformulation of solver formula” is developed for the approximation of a transient state vector. This method consists of solving the reformulated solver equation regarding the event step-size at the state event and approximation of a transient state vector by using the solver formula. This method can be applied on certain explicit solver for variable structure systems of ODEs and DAEs transformed into ODEs.
- Another result has been obtained for variable structure systems of ODEs and DAEs transformed into ODEs using the root-finding method on a reformulated explicit solver formula for approximation of transient state vector.
- Another conclusion is the Method development for approximating the transient state vector via application of the root-finding method of nonlinear systems of equations on reformulated implicit solver system equations. This method is specified for implicit solver and implemented with two-procedure prediction and approximation stages for variable structure systems of ODEs or DAEs transformed into ODEs.

- The Method for approximation of a transient vector in hybrid system of semi-explicit DAEs has been developed in two prediction and approximation procedures. This method handles hybrid system of semi-explicit DAEs without index-reduction via application of the root-finding method of nonlinear systems of equations with reformulated system of implicit solver.
- Henon’s method is an important part of this dissertation. The research on Henon’s method includes implementations of approximation of the transient state vector of variable structure systems of ODEs and DAEs transformed into ODEs for embedded solver systems.

In this dissertation, altogether fifteen simulation programs were developed. Nine simulation programs were implemented for verification of three methods: “Reformulation of solver formula, reformulation of solver formula for root-finding method and Henon’s methods. Six additional simulations were implemented for comparisons with developed methods in which three of them were realized with Matlab adaptive method and three further simulations without event location approximations.

The simulation models are “bouncing ball”, “filament pendulum” and “rotor-stator contact” models. The bouncing ball is a hybrid system of ODEs. The rotor-stator contact model is a hybrid system of semi-explicit DAEs and the filament pendulum is a variable structure system of semi-explicit DAEs.

The simulations via two methods reformulation of solver formula and reformulation of solver formula for root-finding method were realized by algorithm developments and coding of the entire solver systems; whereas Henon’s method was realized by developing the algorithms on embedded ODE45 of Matlab.

The simulation results show that the root-finding method composed with implicit trapezoidal solver has a relatively low local error for approximated magnitude of event location in hybrid semi-explicit DAEs. The root-finding Method has higher programming costs.

Henon’s method is one with certain flexibility regarding coupling for embedded solver systems.

The simulation results and relative local errors of approximated event variables are listed in various tables. The results are comparable with the results given by Matlab using adaptive zero-crossing and without zero-crossing.

7 Appendices

A1: Separate Simulations of Hybrid Models in Different Frames

A1.1: Simulation of a Hybrid Model of Filament Pendlum

The different structures of a hybrid system can be simulated separately. In the following figure, the pendulum simulation model is shown.

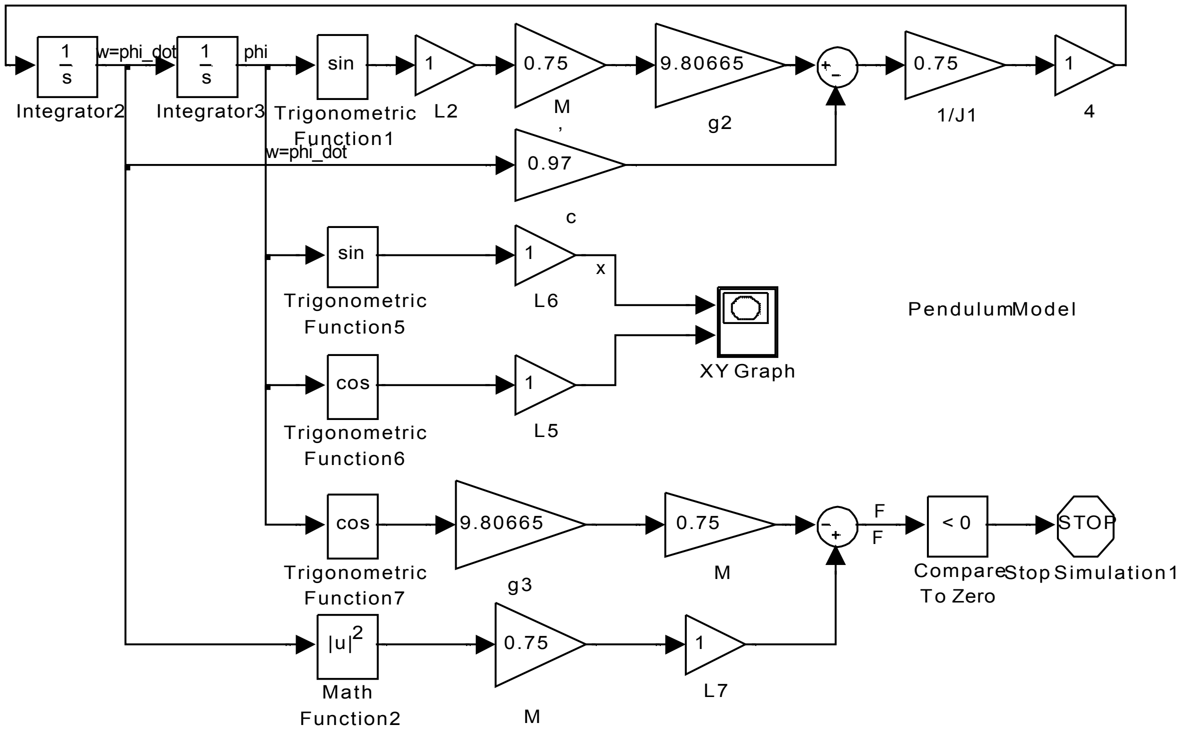

Figure 32: Pendulum Simulation Model in Simulink

This model runs as long as the pendulum attraction force is greater or equal to zero. If the tension force decreases below zero $F_a(t) < 0$, then the simulation stops.

Before starting the simulation of the free-fall model, its variables have to be initialized manually by the computed initialization variables via the pendulum model. The free-fall model is shown as follows:

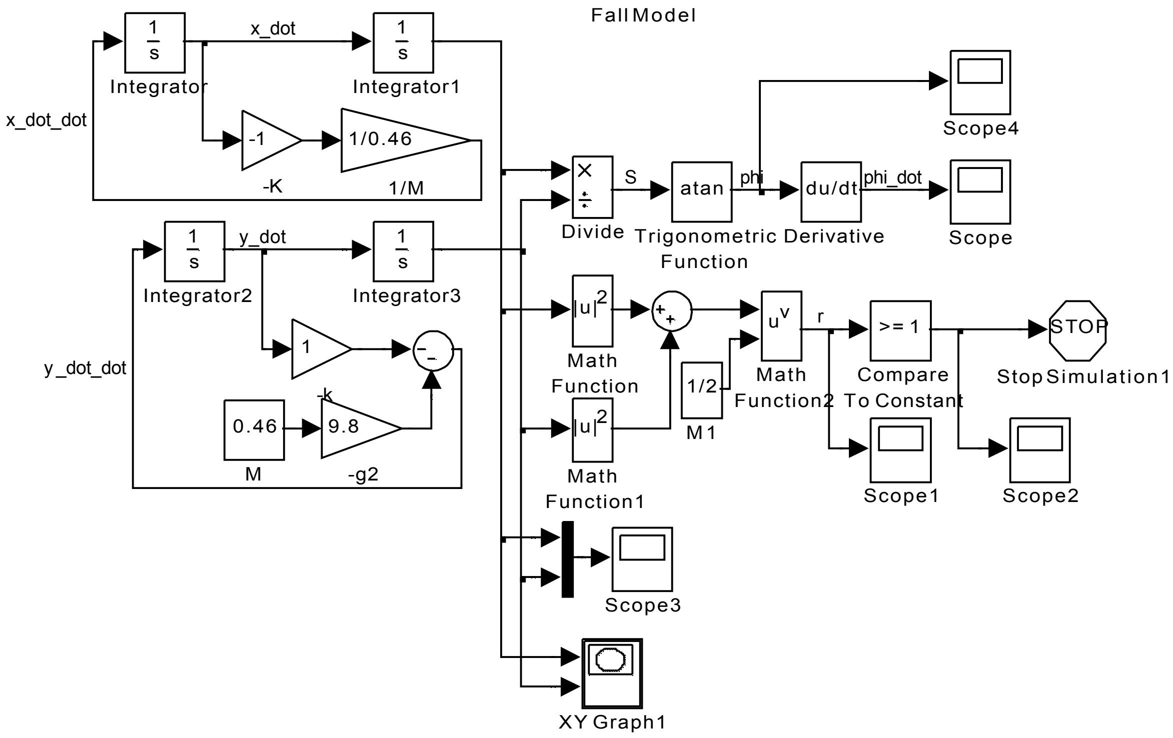

Figure 33: Free-Fall Simulation Model in Simulink

The coordinates of the pendulum position, after a completed free-fall simulation, deflects from its equilibrium point, so the values of the last free-fall coordinates can be used to reinitialize the pendulum model, and the pendulum simulation would restart.

Due to the automation of the event handling the two models can be simulated automatically after each other. In this case, at the state event, the transient state vector and the location of the independent variable are approximated and the initializations are carried out automatically.

The following figures show the three separate simulation results of the variable structure filament pendulum model. Figure 34 shows the pendulum trajectory from the start until the free-fall event. Figure 35 shows the free-fall simulation when the pendulum deflects from its orbit.

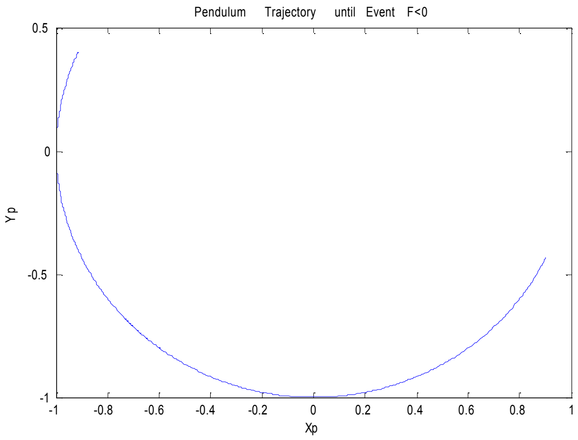

Figure 34: Pendulum Swinging after Start

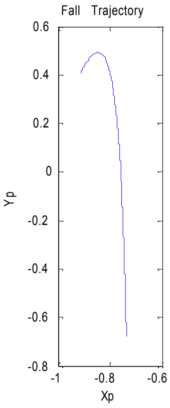

Figure 35: Free-Fall after Swinging

During the free-fall, when the pendulum achieves its maximum radius, then the simulation must be stopped and the pendulum swinging simulation should be started. The result of this simulation is illustrated in Figure 36.

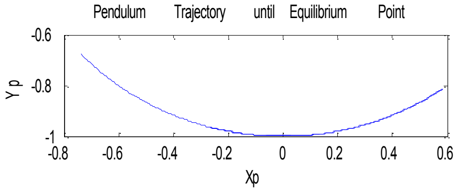

Figure 36: Pendulum Swinging after Free-Fall

A2: Rotor-Stator DAEs transformed into ODEs

A2.1:

The transformation of the algebraic equation (149) to an ODE is realized by the chain rule.

$$
\begin{aligned}
&\frac{d}{d\iota} y_1(\iota) = c \frac{d}{d\iota} x_{ext_1}(\iota) \\
&= c\left(\frac{\partial x_{ext_1}}{\partial x_1}\left(x_1(\iota), x_3(\iota), x_5(\iota), x_7(\iota)\right) \frac{d}{d\iota} x_1(\iota) + \frac{\partial x_{ext_1}}{\partial x_3}\left(x_1(\iota), x_3(\iota), x_5(\iota), x_7(\iota)\right) \frac{d}{d\iota} x_3(\iota) \right. \\
&\left. + \frac{\partial x_{ext_1}}{\partial x_5}\left(x_1(\iota), x_3(\iota), x_5(\iota), x_7(\iota)\right) \frac{d}{d\iota} x_5(\iota) + \frac{\partial x_{ext_1}}{\partial x_7}\left(x_1(\iota), x_3(\iota), x_5(\iota), x_7(\iota)\right) \frac{d}{d\iota} x_7(\iota) \right)
\end{aligned}
\tag{162}
$$

The derivative of algebraic equation (149) is shown by

$$
\dot{y}_1(\iota) = c \frac{\left(x_1(\iota) - (x_5(\iota) + x_0)\right)\left(\dot{x}_1(\iota) - \dot{x}_5(\iota)\right) + \left(x_3(\iota) - (x_7(\iota) + y_0)\right)\left(\dot{x}_3(\iota) - \dot{x}_7(\iota)\right)}{\left(\sqrt{\left(x_1(\iota) - (x_5(\iota) + x_0)\right)^2 + \left(x_3(\iota) - (x_7(\iota) + y_0)\right)^2}\right)}
\tag{163}
$$

with substituting of $\dot{x}_1(\iota)$, $\dot{x}_3(\iota)$, $\dot{x}_5(\iota)$ and $\dot{x}_7(\iota)$ with their equivalents the following ODE results

$$
\dot{y}_1(\iota) = c \frac{\left(x_1(\iota) - (x_5(\iota) + x_0)\right)\left(x_2(\iota) - x_6(\iota)\right) + \left(x_3(\iota) - (x_7(\iota) + y_0)\right)\left(x_4(\iota) - x_8(\iota)\right)}{\left(\sqrt{\left(x_1(\iota) - (x_5(\iota) + x_0)\right)^2 + \left(x_3(\iota) - (x_7(\iota) + y_0)\right)^2}\right)}.
\tag{164}
$$

A2.2:

The chain rule on algebraic equation (150) is demonstrated by (165).

$$
\begin{aligned}
&\frac{d}{d\iota}y_2(\iota)\\
&=ck_h\left(\frac{\partial x_{ext_1}}{\partial x_1}(x_1(\iota),x_3(\iota),x_5(\iota),x_7(\iota))\frac{d}{d\iota}x_1(\iota)+\frac{\partial x_{ext_1}}{\partial x_3}(x_1(\iota),x_3(\iota),x_5(\iota),x_7(\iota))\frac{d}{d\iota}x_3(\iota)\right.\\
&\left.+\frac{\partial x_{ext_1}}{\partial x_5}(x_1(\iota),x_3(\iota),x_5(\iota),x_7(\iota))\frac{d}{d\iota}x_5(\iota)+\frac{\partial x_{ext_1}}{\partial x_7}(x_1(\iota),x_3(\iota),x_5(\iota),x_7(\iota))\frac{d}{d\iota}x_7(\iota)\right)\\
&+cc_h\frac{\omega}{2\pi}\left(\frac{\partial x_{ext_2}}{\partial x_1}(x_1(\iota),x_2(\iota),x_3(\iota),x_4(\iota),x_5(\iota),x_6(\iota),x_7(\iota),x_8(\iota))\frac{d}{d\iota}x_1(\iota)+\frac{\partial x_{ext_2}}{\partial x_2}(x_1(\iota),x_2(\iota),x_3(\iota),x_4(\iota),x_5(\iota),x_6(\iota),x_7(\iota),x_8(\iota))\frac{d}{d\iota}x_2(\iota)\right.\\
&+\frac{\partial x_{ext_2}}{\partial x_3}(x_1(\iota),x_2(\iota),x_3(\iota),x_4(\iota),x_5(\iota),x_6(\iota),x_7(\iota),x_8(\iota))\frac{d}{d\iota}x_3(\iota)+\frac{\partial x_{ext_2}}{\partial x_4}(x_1(\iota),x_2(\iota),x_3(\iota),x_4(\iota),x_5(\iota),x_6(\iota),x_7(\iota),x_8(\iota))\frac{d}{d\iota}x_4(\iota)\\
&+\frac{\partial x_{ext_2}}{\partial x_5}(x_1(\iota),x_2(\iota),x_3(\iota),x_4(\iota),x_5(\iota),x_6(\iota),x_7(\iota),x_8(\iota))\frac{d}{d\iota}x_5(\iota)+\frac{\partial x_{ext_2}}{\partial x_6}(x_1(\iota),x_2(\iota),x_3(\iota),x_4(\iota),x_5(\iota),x_6(\iota),x_7(\iota),x_8(\iota))\frac{d}{d\iota}x_6(\iota)\\
&\left.+\frac{\partial x_{ext_2}}{\partial x_7}(x_1(\iota),x_2(\iota),x_3(\iota),x_4(\iota),x_5(\iota),x_6(\iota),x_7(\iota),x_8(\iota))\frac{d}{d\iota}x_7(\iota)+\frac{\partial x_{ext_2}}{\partial x_8}(x_1(\iota),x_2(\iota),x_3(\iota),x_4(\iota),x_5(\iota),x_6(\iota),x_7(\iota),x_8(\iota))\frac{d}{d\iota}x_8(\iota)\right)
\end{aligned}
\tag{165}
$$

Hence (165) can be written as follows

$$
\begin{aligned}
&\dot{y}_2(\iota)=ck_h\frac{(x_1(\iota)-(x_5(\iota)+x_0))(x_2(\iota)-x_6(\iota))+(x_3(\iota)-(x_7(\iota)+y_0))(x_4(\iota)-x_8(\iota))}{\left(\sqrt{(x_1(\iota)-(x_5(\iota)+x_0))^2+(x_3(\iota)-(x_7(\iota)+y_0))^2}\right)}\\
&+cc_h\frac{\omega}{2\pi}\left[(\dot{x}_2(\iota)-\dot{x}_6(\iota))\cos\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right)+(\dot{x}_4(\iota)-\dot{x}_8(\iota))\sin\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right)\right.\\
&+\left(\frac{(\dot{x}_3(\iota)-\dot{x}_7(\iota))(x_1(\iota)-(x_5(\iota)+x_0))-(\dot{x}_1(\iota)-\dot{x}_5(\iota))(x_3(\iota)-(x_7(\iota)+y_0))}{(x_1(\iota)-(x_5(\iota)+x_0))^2}\frac{1}{\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)^2+1}\right)\\
&\left.\left(-(x_2(\iota)-x_6(\iota))\sin\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right)+(x_4(\iota)-x_8(\iota))\cos\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right)\right)\right].
\end{aligned}
\tag{166}
$$

The derivatives of $\dot{x}_1(\iota)$, $\dot{x}_3(\iota)$, $\dot{x}_5(\iota)$ and $\dot{x}_7(\iota)$ can be replaced by their equivalents but the derivatives of $\dot{x}_2(\iota)$, $\dot{x}_4(\iota)$, $\dot{x}_6(\iota)$ and $\dot{x}_8(\iota)$ have hybrid equations, thus, they are replaced in (166) with their equivalents for two cases $q_c(\iota)=0$ according to ODEs (142) and

$q_c(\iota)=1$ according to ODEs (143). Therefore, the ODE (166) should be used once before contacting for $q_c(\iota)=0$

$$
\begin{aligned}
\dot{y}_2(\iota) = {} & ck_h \frac{(x_1(\iota)-(x_5(\iota)+x_0))(x_2(\iota)-x_6(\iota))+(x_3(\iota)-(x_7(\iota)+y_0))(x_4(\iota)-x_8(\iota))}{\left(\sqrt{(x_1(\iota)-(x_5(\iota)+x_0))^2+(x_3(\iota)-(x_7(\iota)+y_0))^2}\right)} \\
& + cc_h \frac{\omega}{2\pi}\left[\left(\left(-\left(2\zeta_r A_r x_2(\iota)+A_r^2 x_1(\iota)\right)+(2\pi)^2 E\cos(2\pi\iota)\right)+\left(2\zeta_s A_s x_6(\iota)+A_s^2 x_5(\iota)\right)\right)\cos\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right)\right. \\
& + \left(\left(-\left(2\zeta_r A_r x_4(\iota)+A_r^2 x_3(\iota)\right)+(2\pi)^2 E\sin(2\pi\pi)\right)+\left(2\zeta_s A_s x_8(\iota)+A_s^2 x_7(\iota)\right)\right)\sin\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right) \\
& + \left(\frac{(x_4(\iota)-x_8(\iota))(x_1(\iota)-(x_5(\iota)+x_0))-(x_2(\iota)-x_6(\iota))(x_3(\iota)-(x_7(\iota)+y_0))}{(x_1(\iota)-(x_5(\iota)+x_0))^2}\frac{1}{\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)^2+1}\right) \\
& \left.\left(-(x_2(\iota)-x_6(\iota))\sin\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right)+(x_4(\iota)-x_8(\iota))\cos\left(\operatorname{atan}\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)\right)\right)\right]
\end{aligned}
$$

(167)

and once more in contact case for $q_c(\iota)=1$ as follows

$$
\begin{aligned}
&\dot{y}_2(\iota)=ck_h\,\frac{(x_1(\iota)-(x_5(\iota)+x_0))(x_2(\iota)-x_6(\iota))+(x_3(\iota)-(x_7(\iota)+y_0))(x_4(\iota)-x_8(\iota))}{\left(\sqrt{(x_1(\iota)-(x_5(\iota)+x_0))^2+(x_3(\iota)-(x_7(\iota)+y_0))^2}\right)}\\
&+cc_h\,\frac{\omega}{2\pi}\{[(-(2\zeta_r A_r x_2(\iota)+A_r^2 x_1(\iota))+(2\pi)^2 E\cos(2\pi\iota)+(A_h^2(\sqrt{(x_1(\iota)-(x_5(\iota)+x_0))^2+(x_3(\iota)-(x_7(\iota)+y_0))^2}-1)\\
&+2\zeta_h A_h((x_2(\iota)-x_6(\iota))\cos(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))+(x_4(\iota)-x_8(\iota))\sin(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))))\\
&(-\cos(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))+\mu\sin(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))))\\
&+(2\zeta_s A_s x_6(\iota)+A_s^2 x_5(\iota))+(A_h^2(\sqrt{(x_1(\iota)-(x_5(\iota)+x_0))^2+(x_3(\iota)-(x_7(\iota)+y_0))^2}-1)\\
&+2\zeta_h A_h((x_2(\iota)-x_6(\iota))\cos(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))+(x_4(\iota)-x_8(\iota))\sin(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))))\\
&(-\cos(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))+\mu\sin(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))))]\cos(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))\\
&+[(-(2\zeta_r A_r x_4(\iota)+A_r^2 x_3(\iota))+(2\pi)^2 E\cos(2\pi\iota)+(A_h^2(\sqrt{(x_1(\iota)-(x_5(\iota)+x_0))^2+(x_3(\iota)-(x_7(\iota)+y_0))^2}-1)\\
&+2\zeta_h A_h((x_2(\iota)-x_6(\iota))\cos(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))+(x_4(\iota)-x_8(\iota))\sin(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))))\\
&(-\sin(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))-\mu\cos(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))))\\
&+(2\zeta_s A_s x_8(\iota)+A_s^2 x_7(\iota))+(A_h^2(\sqrt{(x_1(\iota)-(x_5(\iota)+x_0))^2+(x_3(\iota)-(x_7(\iota)+y_0))^2}-1)\\
&+2\zeta_h A_h((x_2(\iota)-x_6(\iota))\cos(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))+(x_4(\iota)-x_8(\iota))\sin(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))))\\
&(-\sin(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))-\mu\cos(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))))]\sin(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))\\
&+\left(\frac{(x_4(\iota)-x_8(\iota))(x_1(\iota)-(x_5(\iota)+x_0))-(x_2(\iota)-x_6(\iota))(x_3(\iota)-(x_7+y_0))}{(x_1-(x_5+x_0))^2}\,\frac{1}{\left(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}\right)^2+1}\right)\\
&(-(x_2(\iota)-x_6(\iota))\sin(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))+(x_4(\iota)-x_8(\iota))\cos(\operatorname{atan}(\frac{x_3(\iota)-(x_7(\iota)+y_0)}{x_1(\iota)-(x_5(\iota)+x_0)}))))\}.
\end{aligned}
$$

(168)

A2.3:

Form (169) gives the components of the event vector function, for a transition from rotor-stator contact case to noncontact case. In this system, the extended variable $\tilde{x}_{ext_3,k}$ must be substitute with its equivalent $\tilde{x}_{ext_3,k} := \mathrm{atan}\left(\frac{\tilde{x}_{3,k} - \left(\tilde{x}_{7,k} + y_0\right)}{\tilde{x}_{1,k} - \left(\tilde{x}_{5,k} + x_0\right)}\right)$. In simulation, form (169) is used according to Newton’s root-finding method (38).

$$
\begin{aligned}
&\hat{h}_1\left(\Delta\tilde{t}_k, \vec{\tilde{x}}_k, \tilde{y}_{2,k}\right) = -\tilde{x}_{1,k} + \hat{x}_{1,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(\hat{x}_{2,j-1} + \tilde{x}_{2,k}\right) \\
&\hat{h}_2\left(\Delta\tilde{t}_k, \vec{\tilde{x}}_k, \tilde{y}_{2,k}\right) = -\tilde{x}_{2,k} + \hat{x}_{2,j-1} \\
&+ \frac{\tilde{\Delta}\tilde{\iota}_{e,k}}{2}\Big[-\left(2\zeta_r A_r \hat{x}_{2,j-1} + A_r^2 \hat{x}_{1,j-1}\right) + (2\pi)^2 E\cos\left(2\pi\iota_{j-1}\right) + \left(A_h^2 \hat{x}_{ext_1,j-1} + 2\zeta_h A_h \hat{x}_{ext_2,j-1}\right)\left(-\cos\left(\hat{x}_{ext_3,j-1}\right) + \mu\sin\left(\hat{x}_{ext_3,j-1}\right)\right) \\
&- \left(2\zeta_r A_r \tilde{x}_{2,k} + A_r^2 \tilde{x}_{1,k}\right) + (2\pi)^2 E\cos\left(2\pi\left(\iota_{j-1} + \Delta\tilde{t}_{e,k}\right)\right) + \left(A_h^2 \tilde{x}_{ext_1,k} + 2\zeta_h A_h \tilde{x}_{ext_2,k}\right)\left(-\cos\left(\tilde{x}_{ext_3,k}\right) + \mu\sin\left(\tilde{x}_{ext_3,k}\right)\right)\Big] \\
&\hat{h}_3\left(\Delta\tilde{t}_k, \vec{\tilde{x}}_k, \tilde{y}_{2,k}\right) = -\tilde{x}_{3,k} + \hat{x}_{3,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(\hat{x}_{4,j-1} + \tilde{x}_{4,k}\right) \\
&\hat{h}_4\left(\Delta\tilde{t}_k, \vec{\tilde{x}}_k, \tilde{y}_{2,k}\right) = -\tilde{x}_{4,k} + \hat{x}_{4,j-1} \\
&+ \frac{\Delta\tilde{\iota}_{e,k}}{2}\Big[-\left(2\zeta_r A_r \hat{x}_{4,j-1} + A_r^2 \hat{x}_{3,j-1}\right) + (2\pi)^2 E\sin\left(2\pi\pi_{j-1}\right) + \left(A_h^2 \hat{x}_{ext_1,j-1} + 2\zeta_h A_h \hat{x}_{ext_2,j-1}\right)\left(-\sin\left(\hat{x}_{ext_3,j-1}\right) - \mu\cos\left(\hat{x}_{ext_3,j-1}\right)\right) \\
&- \left(2\zeta_r A_r \tilde{x}_{4,k} + A_r^2 \tilde{x}_{3,k}\right) + (2\pi)^2 E\sin\left(2\pi\left(\iota_{j-1} + \Delta\tilde{t}_{e,k}\right)\right) + \left(A_h^2 \tilde{x}_{ext_1,k} + 2\zeta_h A_h \tilde{x}_{ext_2,k}\right)\left(-\sin\left(\tilde{x}_{ext_3,k}\right) - \mu\cos\left(\tilde{x}_{ext_3,k}\right)\right)\Big] \\
&\hat{h}_5\left(\Delta\tilde{t}_k, \vec{\tilde{x}}_k, \tilde{y}_{2,k}\right) = -\tilde{x}_{5,k} + \hat{x}_{5,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(\hat{x}_{6,j-1} + \tilde{x}_{6,k}\right) \\
&\hat{h}_6\left(\Delta\tilde{t}_k, \vec{\tilde{x}}_k, \tilde{y}_{2,k}\right) = -\tilde{x}_{6,k} + \hat{x}_{6,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\Big[-\left(2\zeta_s A_s \hat{x}_{6,j-1} + A_s^2 \hat{x}_{5,j-1}\right) - \left(A_h^2 \hat{x}_{ext_1,j-1} + 2\zeta_h A_h \hat{x}_{ext_2,j-1}\right)\left(-\cos\left(\hat{x}_{ext_3,j-1}\right) + \mu\sin\left(\hat{x}_{ext_3,j-1}\right)\right) \\
&- \left(2\zeta_s A_s \tilde{x}_{6,k} + A_s^2 \tilde{x}_{5,k}\right) - \left(A_h^2 \tilde{x}_{ext_1,k} + 2\zeta_h A_h \tilde{x}_{ext_2,k}\right)\left(-\cos\left(\tilde{x}_{ext_3,k}\right) + \mu\sin\left(\tilde{x}_{ext_3,k}\right)\right)\Big] \\
&\hat{h}_7\left(\Delta\tilde{t}_k, \vec{\tilde{x}}_k, \tilde{y}_{2,k}\right) = -\tilde{x}_{7,k} + \hat{x}_{7,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\left(\hat{x}_{8,j-1} + \tilde{x}_{8,k}\right) \\
&\hat{h}_8\left(\Delta\tilde{t}_k, \vec{\tilde{x}}_k, \tilde{y}_{2,k}\right) = -\tilde{x}_{8,k} + \hat{x}_{8,j-1} + \frac{\Delta\tilde{\iota}_{e,k}}{2}\Big[-\left(2\zeta_s A_s \hat{x}_{8,j-1} + A_s^2 \hat{x}_{7,j-1}\right) - \left(A_h^2 \hat{x}_{ext_1,j-1} + 2\zeta_h A_h \hat{x}_{ext_2,j-1}\right)\left(-\sin\left(\hat{x}_{ext_3,j-1}\right) - \mu\cos\left(\hat{x}_{ext_3,j-1}\right)\right) \\
&- \left(2\zeta_s A_s \tilde{x}_{8,k} + A_s^2 \tilde{x}_{7,k}\right) - \left(A_h^2 \tilde{x}_{ext_1,k} + 2\zeta_h A_h \tilde{x}_{ext_2,k}\right)\left(-\sin\left(\tilde{x}_{ext_3,k}\right) - \mu\cos\left(\tilde{x}_{ext_3,k}\right)\right)\Big] \\
&h_{y_1}\left(\Delta\tilde{t}_k, \vec{\tilde{x}}_k, \tilde{y}_{2,k}\right) = c\left(\sqrt{\left(\tilde{x}_{1,k} - \left(\tilde{x}_{5,k} + x_0\right)\right)^2 + \left(\tilde{x}_{3,k} - \left(\tilde{x}_{7,k} + y_0\right)\right)^2} - 1\right) \\
&h_{y_2}\left(\Delta\tilde{t}_k, \vec{\tilde{x}}_k, \tilde{y}_{2,k}\right) = -\tilde{y}_{2,k} + k_h c\left(\sqrt{\left(\tilde{x}_{1,k} - \left(\tilde{x}_{5,k} + x_0\right)\right)^2 + \left(\tilde{x}_{3,k} - \left(\tilde{x}_{7,k} + y_0\right)\right)^2} - 1\right) \\
&+ cc_h \frac{\omega}{2\pi}\left(\left(\tilde{x}_{2,k} - \tilde{x}_{6,k}\right)\cos\left(\mathrm{atan}\left(\tilde{x}_{ext_3,k}\right)\right) + \left(\tilde{x}_{4,k} - \tilde{x}_{8,k}\right)\sin\left(\mathrm{atan}\left(\tilde{x}_{ext_3,k}\right)\right)\right)
\end{aligned}
\tag{169}
$$

8 List of Figures

Figure 1: A Variable Structure System with Switching System and Interfaces 9
Figure 2: State Graph of Hybrid System 12
Figure 3: Parameter Modifications in Hybrid System..13
Figure 4: Modifications of DAEs in Hybrid System..14
Figure 5: Total Structure Modification in Hybrid System..16
Figure 6: Approximation Procedure of Event Location in Hybrid System...............................20
Figure 7: State Graph of Bouncing Ball System...51
Figure 8: Approximated Solutions of Modified Bouncing Ball Hybrid System 55
Figure 9: Trajectory of State Variables of Modified Bouncing Ball Hybrid System..............55
Figure 10: Forces in Free-Fall and Mathematical Pendulum Systems.....................................60
Figure 11: Initializations at State Transitions for Free-Fall and Pendulum Systems...............64
Figure 12: State Graph of Variable Structure System of Semi-Explicit DAEs for Filament Pendulum...65
Figure 13: Simulation of Filament Pendulum Positions...72
Figure 14: Simulation of Filament Pendulum Trajectory...72
Figure 15: Tangential and Radial Forces of Rotor-Stator in Contact Location........................76
Figure 16: Rotor and Stator Offsets 78
Figure 17: Stator Deflection...79
Figure 18: Rotor Eccentricity Radius and Rotor-Stator Deflections..79
Figure 19: Radial Rotor to Stator Clearance...80
Figure 20: Simulation Concept Based on Dimensional and Dimensionless Variables............86
Figure 21: State Graph of Rotor-Stator Contact by Switching Variable $q_c(\iota)$........................87
Figure 22: Extended Form of State Graph of Switching Variable $q_c(\iota)$.................................87
Figure 23: State Graph of Rotor-Stator Contact by Switching Variable $q_c(\iota)$ According to Table 7...89
Figure 24: State Graph of Tangential Force by Switching Variable $q_t(\iota)$.............................89
Figure 25: Modified State Graph of Switching Variable $q_t(\iota)$ 90
Figure 26: Rotor-Stator State Graph...91
Figure 27: Rotor-Stator State Graph for $v_t(\iota) >> 0$ 92
Figure 28: Simulation of Rotor Position...103
Figure 29: Simulation of Stator Position 103
Figure 30: Simulation of Rotor Trajectory 103
Figure 31: Simulation of Stator Trajectory...103
Figure 32: Pendulum Simulation Model in Simulink...110
Figure 33: Free-Fall Simulation Model in Simulink...111
Figure 34: Pendulum Swinging after Start 112
Figure 35: Free-Fall after Swinging 112
Figure 36: Pendulum Swinging after Free-Fall..112

9 Nomenclature

a	Acceleration
a_e	Threshold value
a_n	Radial acceleration
a_r	Radial acceleration
a_t	Tangential acceleration
A_h	Interpreted as the dimensionless abbreviation factor of the contact model
A_r	Interpreted as the dimensionless abbreviation factor of the rotor
A_s	Interpreted as the dimensionless abbreviation factor of the stator
c	Radial clearance between rotor and stator
c	Damping constant of ball
c_r	Rotor damping coefficient
c_s	Stator damping coefficient
c_h	Damping coefficient of contact system
c_x	Damping coefficient of free-fall equation along x-axis
c_y	Damping coefficient of free-fall equation along y-axis
c_p	Pendulum damping coefficient
C	Set of conditional guards
d	Radial intrusion depth
D	Dimensionless radial intrusion depth
D	Domain of a function
D_r	Rotor diameter
e	Error
e	Eccentricity radial offset of rotor
e	Event index
E	Set of arrows of transition topology
E	Dimensionless value of e
f	Function
f	Force

f_{cn}	Switched radial intrusion force
f_{ct}	Switched tangential intrusion force
f_n	Radial intrusion force
f_n	Weight force in radial direction
f_t	Weight force in tangential direction
$\vec{f}$	System vector function
F	Set of the vector functions
F	Logical value false
F_a	Tension force
$\vec{F}$	Fully-implicit DAEs
g	Gravity acceleration
g	An algebraic equation
$\vec{g}$	System algebraic equations (algebraic vector equation)
h	Event function
$\vec{h}$	System event function (event vector function)
H	Hybrid automaton
i	Index of state variable involved state event
I	Interpretation
I	Interval defined for independent variable
j	Solver iteration index
J	Moment of inertia
J	Jacobian matrix
k	Iteration index of root-finding method
k_h	Stiffness coefficient of contact model
ℓ	Index
L	Set of discrete states
L	Length of pendulum
m	Mass of ball
m	Mass of pendulum
m_r	Mass of rotor
m_s	Mass of stator

M_g	Moment of the gravitational force
M_d	Damping moment
n	Dimension of state vector
$\mathbf{N}$	Field of natural number
P	Set of modification parameters of hybrid automaton
P	Event plane, intersection plane with state variable
$\vec{\mathrm{p}}$	Parameter vector
q	Switching variable
q	Logical variable
Q	Set of the switching variable
r	Radius of ball
r	Distance between centre of coordinate system and centre of pendulum
r_{rs}	Relative radial distance between rotor and stator
R	Reset map
R_{rs}	Dimensionless value of r_{rs}
$\mathbf{R}$	Field of real numbers
s	Arc length of angle φ
s	Discrete state, node, location
Sy/Asy	Set of synchronous or asynchronous variables for parallel processes
t	Independent variable
t	Time
T	Logical value true
$\vec{\mathrm{u}}$	Input vector
U	Subset or subspace as a domain of a function
U_q	Set of input event variables
U_x	Set of the input continuous variables
U_z	Set of input discrete time state variables
v	Ball velocity
v_t	Rotor-stator tangential impact velocity
v_y	Pendulum velocity in free-fall position along y-axis
v_x	Pendulum velocity in free-fall position along x-axis

V_e	Set of output event variables
V_x	Set of output continuous time state variables
V_z	Set of output discrete time state variables
W	Weight
x	State variable
x_i	State variable which is involved state event with index i
x_0	Offset between rotor and stator in x-direction
x_r	Position of rotor in x-direction
x_s	Position of stator in x-direction
x_{rs}	Relative deflection between rotor and stator in x-direction
$\vec{x}$	Continuous time state vector
$\dot{\vec{x}}$	Derivative of $\vec{x}$ with respect to independent variable time t
X	Finite set of continuous states
X_0	Set of the initial states
X_0	Dimensionless value of offset x_0 along x-axis
x_{ext}	Extended variable in rotor-stator model
x_p	Pendulum coordinate axis
X_r	Dimensionless value of rotor position x_r along x-axis
X_s	Dimensionless value of stator position x_s along x-axis
X_{rs}	Dimensionless value of relative deflection x_{rs} along x-axis
y	Algebraic variable
y	Ball altitude
y	Position on y-axis
y_0	Offset between rotor and stator along y-axis
y_r	Position of rotor along y-axis
y_s	Position of stator along y-axis
y_{rs}	Relative deflection between rotor and stator on y-axis
$\vec{y}$	Algebraic variable vector
Y	Set of the algebraic variables
Y_0	Dimensionless value of offset y_0 along y-axis

Y_r	Dimensionless value of rotor position y_r along y-axis
Y_s	Dimensionless value of stator position y_s along y-axis
Y_{rs}	Dimensionless value of relative deflection y_{rs} along y-axis
$\vec{z}$	Discrete time state vector
Z	Set of the discrete state variables
β	Elasticity constant of the ball
γ	Auxiliary algebraic variable
$\vec{\gamma}$	Auxiliary algebraic variable vector
Γ	Auxiliary function in Henon’s method
δ	Auxiliary variable
Δt	Step-size
Δt_e	Step-size at the state event (event step-size)
ε	Upper tolerance value
ζ_r	Damping ratio of dimensionless equation of rotor motion
ζ_s	Damping ratio of dimensionless equation of stator motion
ζ_h	Damping ratio of dimensionless contact model equation
Θ	Auxiliary function
$\vec{\Theta}$	Auxiliary vector function
ι	Independent variable in dimensionless system
λ	Tangent ratio
μ	Friction coefficient of impact model
π	Ratio of circle circumference to its diameter (pi)
τ	Auxiliary independent variable
φ	Contact angle
φ_p	Pendulum angle
χ	Auxiliary variable
$\vec{\chi}$	Auxiliary state vector
ω	Angular velocity
ω_p	Pendulum angular velocity
Ω_r	Interpreted as dimensionless natural frequency of rotor
Ω_s	Interpreted as dimensionless natural frequency of stator

Ω_h	Interpreted as dimensionless natural frequency of contact model
$(\ )_e$	Event variable, variable magnitude at an event, threshold value
$\dot{(\ )}$	First derivative with respect to independent variable t
$\ddot{(\ )}$	Second derivative with respect to independent variable t
$(\ )'$	First derivation with respect to independent variable ι
$(\ )''$	Second derivation with respect to independent variable ι
$(\ )_0$	Initial value
$\tilde{(\ )}$	Predicted value
$\hat{(\ )}$	Approximated value
$\vec{(\ )}$	Vector

10 References

[ACK03] **A. Chutinan, B. H. Krogh**. *Fellow IEEE: Computational Techniques for Hybrid System Verification*. IEEE Transaction on Automatic Control, Vol. 48. 2003.

[ABT04] **A. Beiset**. *Theory and Problems of Applied Physics*. 4th Edition, Schaum's Outline Series, 2004.

[AGI12] **A. Geletu**. *Inroduction to Differential Algebraic Equations*. Ilmenau University of Technology, Department of Simulation and Optimal Processes (SOP). Slides, 2011/2012.

[AKL83] **A. Kawamura**. *Lipschitz Continuous Initial Value Problem is Polynomal Space Complete.* (in answer to question raised in 1983).

[AOW83] **A. V. Oppenheim, A. S. Willsky, I. T. Young**. *Signal and System*. Prentice-Hall, Inc. 1983.

[ASE02] **A. V. Savkin, R. J. Evans**. *Hybrid Dynamical Systems: Controller and Sensor Switching Problems*. Birkhäuse Boston c/o Springer-Verlag, 2002.

[AKG02] **A. Kharab, R. B. Guenther**. *An Introduction to Numerical Methods: a Matlab Approach.* Chapman & Hall/CRC, 2002.

[ASN06] **A. Steinbrecher**. *Numerical Solution of Quasi-Linear Differential-Algebraic Equations and Industrial Simulation of Multibody Systems*. Dissertation, Technische Universität Berlin, Fakultät II – Mathematik und Naturwissenschaften der Technischen Universität Berlin, 2006.

[BDS05] **B. D. Schutter**. *Models for Hybrid Systems. Delft Centre for Systems and Control.* Delft University of Technology. 2005. http://www.dii.unisi.it/hybrid/school/slides/02-deschutter.pdf

[BSC03] **B. Sedghi**. *Control Design of Hybrid Systems Via Dehybridizatio*n. *These No.2859*. Ecole Polytechnique Federale de Lausanna, 2003.

[CGN07] **C. N. Geusen, A. Nordweg**. *Objektorientierte Modellierung und Simulation technischer Systeme. Vertiefungsveranstaltung im Studiengebiet SSG*. TU Berlin Fraunhofer Institute Rechnerarchitekture und Sofrwaretechnik, 2006/2007.

[CBT99] **C. Bendtsen, P. G. Thomsen**. *Numerical Solution of Differential Algebraic Equations, Technical Report*. Technical University of Denmark, Department of Mathematical Modelling, Denmark IMM-REP, 1999.

[CHM80] **C. J. Harris, J. F. Miller**. *Stability of Linear Systems, Some Aspects of Kinematic Similarity*. Academic Press Inc. 1980.

[CTI05] **C. J. Tomlin, S. Islander**. *Hybrid Systems, Analysis, and Control, Lecture Notes 4, Existence of Execution*. Stanford University Spring Quarter. 2004/2005.
http://www.stanford.edu/class/aa278a/

[CTN08] **C. Tischendorf**. *Numerik Differential Algebraischer Gleichungen*. Skript zur Vorlesung, , Universität Köln, 2007/2008.

[CWG70] **C. W. Gear**. *The Simultaneous Numerical Solution of Differential Algebraic Equations*. Stanford University, Computer Science Department, California, 1970.

[DHV06] *Dymola Help Versions (6.0b, Dymola- Dynamic Modeling Laboratory)*. Dynasim AB, 2006.
www.dynasim.se

[DKC02] **D. Kincaid, W. Cheney**. *Numerical Analysis: Mathematics of Scientific Computing*. 3rd Edition, Integre Technical Publishing Co, 2002.

[DYG72] **D. M. Young, R. T. Gregory**. *A Survey of Numerical Mathematics, Volum I.* Addison-Wesley Publishing Company, Richard S. Varga, 1972.

[DWA07] **D. Weiß**. *Allgemeine Lineare Verfahren für Differential-Algebraische Gleichungen mit Index 2*. Dissertation, Hundt Druck GmbH, Köln, 2007.

[EHL80] **E. Hairer, C. Lubich, M. Roche**. *Lecture Notes in Mathematics*. Springer-Verlag, 1980.

[EKA79] **E. Kreyszig**. *Advanced Engineering Mathematics*. 4th Edition, 1979.

[EKD93] **E. Krämer**. *Dynamic of Rotor and Foundation*. Springer-Verlag, 1993.

[FBE10] **F. Breitenecker, H. Ecker, A. Körner, B. Heinzl, M. Rössler**. *ARGRSIM S280, State Events und Strukturdynamische Systeme in Modellbildung und Simulation, Modelbildung und Simulation der Rotor-Stator-Dynamik Strukturdynamische Modellansätze mit Kontaktmodellen. Klassifikation und Evaluierung von Features in Simulation in Hinsicht auf Physical Modelling, State Events und Strukturdynamische Systeme*. TU-Wien, 2010.

[FBE12] **F. Breitenecker, H. Ecker, B. Heinzl, A. Körner, M. Rößler, N. Popper**. *Change of Independent Variable for State Event Detection in System Simulation, Evaluation with Argesim Benchmarks*. Vienna University of Technology, Institute for Analysis and Scientific Computing, Austria Institute for Computer Aided Automation, Institute for Mechanics and Mechatronics, Austria dwh Simulation Services, Vienna Austria, 2012.

[FBH65] **F. B. Hildebrand**. *Introduction to Numerical Analysis*. McGraw-Hill Book Company Inc.1965.[FEC09] **F. Ayres, E. Mendelson**. *Calculus*. 5th Edition, Schaum’s Otlines Series, Mc Graw Hill, 2009.

[FBM09] **F. Breitenecker, F. Miksch, G. Zauner, P. Einzinger, B. Glock**. *S257 Modellbildung und Simulation in Gesundheitsökonomie, HTA und WBM, Teil 1, Fehlerquellen im Modellierungsprozess*. Veranstaltung 2009. http://www.argesim.org/index.php?id=81&tx_ttnews[pointer]=1&tx_ttnews[tt_news]=46&tx_ttnews[backPid]=80&cHash=aed638dccd

[FSA70] **F. S. Acton**. *Numerical Methods That Works*. Harper & Row, Publishers, Inc. 1970.

[FSN89] **F. Scheid**. *Numerical Analysis*. Schaum's Outline Series McGraw-Hill, 1989.

[FSY78] **F. Szidarovszky, S. Yakowitz**. *Principles and Procedures of Numerical Analysis.* Plenum Press, New York, 1978.

[GDB08] **G. Dahlquist, A. Björck**. *Numerical Methods in Scientific Computing, Volume 1*. Society for Industrial and Applied Mathematics, 2008.

[DGH10] **D. F. Griffiths, D. J. Higham**. *Numerical Methods for Ordinary Differential Equations, Initial Values Problems*. Springer-Verlag London Limited, 2010.

[GJR82] **G. Jordan-Engeln, F. Reutter**. *Numerische Mathematik für Ingenieure.* Wissenschaftsverlag, Hain-Druck GmbH, dritte überarbeitete und erweiterte Auflage, 1982.

[GKT90] **G. Keogh**. *The Numerical Solution of Ordinary and Algebraic Differential Equations using One Step Methods*. M. Sc. Thesis, School of Mathematical Sciences, 1990.

[GST88] **G. Sewell**. *The Numerical Solution of Ordinary and Partial Differential Equations*. Academic Press. 1988.

[HLS87] **H. Leipholz**. *Stability Theory, An Introduction to the Stability of Dynamic Systems and Rigid Bodies.* 2nd Edition, John Wiley & Sons Ltd. and B. G. Teubner, 1987.

[HMA91] **H. M. Anita**. *Numerical Methods for Scientists and Engineers*. 2^{nd} Edition, Birkhäuser-Verlag. 1991.

[HSK11] **H. R. Schwarz, N. Köckler**. *Numerische Mathematik.* Vieweg + Teubner Verlag, 8. Auflage, 2011.

[HSE73] **H. Selder**. *Einführung in die Numerische Mathematik für Ingenieure*. Carl Hanser Verlag München, 1973.

[HST07] **H. Stöcker**. *Taschenbuch der Physik.* Verlag Harri Deutsch, 2007.

[IBS01] **I. N. Bronstein, K. A. Semendjsjew, G. Musiol, H. Mühlig**. *Taschenbuch der Mathematik.* Verlag Harri Deutsch, 2001.

[IHC06] **I. M. Haddad, V. S. Chellaboina, S. G. Nersesov**. *Impulsive and Hybrid Dynamical Systems: Stability, Dissopativity and Control*. Published by Princeton University Press, 2006.

[IJJ87] **I. Jacques, C. Judd**. *Numerical Analysis*. Chapman and Hall. 1987.

[JBD85] **J. Becker, H. Dreyes, W. Haacke, R. Nabert**. *Numerische Mathematik für Ingenieure*. B. G. Teubner Stuttgart, 2., überarbeitete Auflage, 1985.

[JEK02] **J. M. Esposito, V. Kumar**. *A Hybrid Systems Framework for Multi-robot Control and Programming*. GRASP Laboratory, University of Pennsylvania, 2002.

[JFE02] **J. F. Epperson**. *An Introduction to Numerical Methods and Analysis*. John Wiley & Sons, Inc. 2002.

[JHE97] **J. Herzberger**. *Einführung in das wissenschaftliche Rechnen für Informatiker, Mathematiker und Naturwissenschaftler*. Addison Wesley Longman Verlag GmbH, 1997.

[JHM92] **J. H. Mathews**. *Numerical Methods for Mathematics, Science and Engineering.* Prentice Hall, Inc. A Simon and Schuster Company, 1992.

[JLW73] **J. L. Willems**. *Stabilität dynamischer Systeme*. R. Oldenbourg-Verlag, 1973.

[JMH09] **J. M. Heinzle**. *Gewöhnliche Differentialgleichungen Ordinary Differential Equations. Unterlage zur Vorlesung Mathematische Methoden der Physik I.* University of Vienna. Version, 2009.

[JOP81] **J. M. Ortega, W. G. Poole**. *An Introduction to Numerical Methods for Differential Equations.* Pitman Publishers, 1981.

[JRA10] **J. Rang**. *Advanced Methods for ODE and DAEs*. TU Braunschweig, Institute of Scientific Computing, Sommersemester, 2010.

[JZH00] **J. Zhang, K. Henrik Johnson, J. Lygeros and S. Sastry**. *Zeno Hybrid System.* 2000.
http://robotics.eecs.berkeley.edu/~sastry/pubs/PDFs%20of%20Pubs2000-2005/Publications%20of%20Postdocs/Zhang/ZhangZenoHybridSystems2001.pdf

[KBC96] **K. E. Brenan, S. L. Campbell, L. R. Petzold**. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM, Society for Industrial and Applied Mathematics, 1996.

[KMM01] **K. W. Morton, D. F. Mayers**. *Numerical Solution of Partial Differential Equations.* Cambridge Univ. Press, 2001.

[LPM07] **L. Papula**. *Mathematik für Ingenieure und Naturwissenschaftler, Band 2.* Friedr. Vieweg & Sohn Verlag, GWV Fachverlage GmbH Wiesbaden, 2007.

[LRP89] **L. R. Petzold**. *Recent Developments in the Numerical Solution of Differential/Algebraic Systems*. Computing and Mathematics Research Division, L-316, Lawrence National Laboratory, Livermore, CA 94550, USA, 1989.

[LTB99] **L. T. Biegler**. *Differential Algebraic Equations (DAEs).* Carnegie Mellon University, Chemical Engineering Department, Pittsburgh, 1999.

[MBM04] **M. Bollhöfer, V. Mehrmann**. *Numerische Mathematik, Eine Projektorientierte Einführung für Ingenieure, Mathematiker und Naturwissenschaftler*. Friedr. Vieweg & Sohn Verlag/GWV Fachverlag GmbH, Wiesbaden, 2004.

[MEJ99] **M. Egerstedt , K. Johnsson, J. Lygeros , S. Sastry**. *Behavior Based Robotics Using Regulaized Hybrid Automata, Optimization and Systems Theory*. Royal Institute of Technology SE-100 44 Stockholm, Sweden. Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA 94720, USA, 1999.

[MGI06] **M. Gerdin**. *Identifikation and Estimation for Models Described by Differential Algebraic Equations*. Dissertation, Linköpings Universitet, 2006.

[MHB09] **M. H. Bourgeois**. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechners*. Vieweg + Teubner | GWV Fachverlag GmbH, 3. Aktualisierte Auflage, Wiesbaden, 2009.

[MHN11] **M. Hermann**. *Numerische Mathematik.* Oldenbourg-Verlag München. 3., überarbeitete und erweiterte Auflage, 2011.

[MHO82] **M. Henon**. *On The Numerical Computation of Poincare Maps*. North Holland Publishing Company, C.N.R.S. Observation de Nice. France, 1982.

[MHSCH] **M. Heemels**. *Solution Concepts and Well-Posedness of Hybrid Systems*. Department of Mechanical Engineering Technical University Eindhoven, 2005.

[MHV10] *Matlab Help version 7.10.0.,* (R2010a).

[MLM98] **M. D. Lemmon, K. X. He, I. Markovsky**. *A Tutorial Introduction to Supervisory Hybrid Systems, Technical Report of the ISIS Group*. Department of Electrical Engineering, University of Notre Dame. 1998. http://www.nd.edu/~isis/techreports/isis-98-004.pdf

[MMT04] **D. Michael M. Tiller**. *Introduction to Physical Modelling with Modelica*. Kluwer Academic Publishers, Second Printing, 2004.

[MPHL2] **M. Prandini**. *Hybrid Systems – Lecture n. 2, Execution of a hybrid automaton: Zeno and blocking hybrid automata*, 2009. http://robotics.eecs.berkeley.edu/~sastry/pubs/PDFs%20of%20Pubs2000-2005/Publications%20of%20Postdocs/Zhang/ZhangDynamicalSystems2000.pdf

[MRS89] **M. R. Spiegel**. *Allgemeine Mechanik, Theorie und Anwendung*. McGraw-Hill Company, 1989.

[LWA08] **L. Wunderlich**. *Analysis and Numerical Solution of Structured and Switched Differential-Algebraic Systems*. Dissertation, Fakultät II Mathematik und Naturwissenschaften der Technischen Universität Berlin, 2008.

[OEC08] **O. Enge-Rosenblatt, C. Clauss, P. Schwarz, F. Breitenecker, C. Nytsch-Geusen**. *Comparisons of Different Modelica-Based Simulators, Using Benchmarks Tasks*. Fraunhofer Institute for Integrated Circuits Branch, Lab Design Automation Zaunerstrsasse 38, 01069 Dresden Germany, Vienna University of Technology, Fraunhofer Institute for Computer Architecture and Software Technology, Berlin Germany, 2008.

[PAT94] **P. A. Tipler**. *Physik*. Spektrum Akademischer Verlag, 1994.

[PBH96] **P. N. Brown, A. C. Hindmarsh**. *Consistent Initial Condition Calculation for Differential-Algebraic Systems.* Lawrence Livermore National Laboratory, University of Minnesota, 1996.

[PFP06] **P. Fritzon**. *Principles of Object-Oriented Modelling and Simulation with Modelica 2.1*. IEEE Press Wiley Interscience Publication, 2006.

[PHW08] **P. Hamann, L. Wunderlich**. *Analysis and Numerical Simulation of Hybrid Differential-Algebraic Equations*. Technische Universität Berlin, Institut für Mathematik, Outline, 2008.

[PMW96] **P. M. E. J. Wijckmans**. *Conditioning of Differential Algebraic Equations and Numerical Solution of Multibody Dynamics*. Thesis Technische Universiteit Eindhoven, 1996.[PHS05] **H. Stachel**. *Skriptum zur Vorlesung Geometrische Kinematik, für die Studienrichtung Maschinenbau*. TU Wien, 2004/2005.

[RBF01] **R. L. Burden, J. D. Faires**. *Numerical Analysis.* 7th Edition, Brooks/Cole Thomson Learning, 2001.

[RBG06] **R. Bronson. Gabriel Costa.** *Differential Equations*. 3rd Edition, McGraw-Hill Companies Inc. 2006.

[RFH07] **R. W. Freund, R. H. W. Hoppe**. *Stoer/Bulirsch: Numerische Mathematik 1.* Springer-Verlag, Berlin Heidelberg, 10., neu bearbeitete Auflage, 2007.

[RGN02] **R. Gasch, R. Nordmann, H. Pfütner**. *Rotordynamik.* Springer-Verlag, 2., Auflage, 2002.

[RPN10] **R. Plato**. *Numerische Mathematik Kompakt, Grunflagenwissen für Studium und Praxis.* Vieweg+Teubner, GWV Fachverlage GmbH, Wiesbaden, 2010.

[RSW05] **R. Schaback, H. Wendland**. *Numerische Mathematik.* Springer-Verlag, Berlin Heidelberg New York, 5., vollständige neu bearbeitete Auflage, 2005.

[RAH01] **R. Atterer**. *Hauptseminar Design hybrider eingebetteter Systeme Teil 4: Hybrid Automaten*. Technische Universität München, Fakultät für Informatik 2001.

[RKA13] **R. Karim**. *Approximation of Event Coordinates of a Multifunction and Multivariable Algebraic Model by Newton Method*. Eurosim 2013, Wales, England, 2013.

[RWT79] **R. Wait**. *The Numerical Solution of Algebraic Equations*. John Wiley & Sons, Ltd, 1979.

[SAB04] **S. A. Bruin**. *Seminar, Theory of Differential Algebraic Equations, Numerische Integration of DAE's*. Technische Universiteit Eindhoven, 2004.

[SCF03] **S. Schulz**. *Four Lectures on Differential-Algebraic Equations*. Berlin, 2003.

[SIC05] **S. Ilie**. *Computational Complixity of Numerical Solutions of Initial Value Problems for Differential Algebraic Equations*. Dissertation, University of Western Ontario, London, Ontario, 2005.

[SLL09] **S. Lopschutz, M. Lipson**. *Linear Algebra*. 4th Edition, McGraw-Hill Companies, Inc. 2009.

[SOF88] **S. O. Fatunla**. *Numerical Methods for Initial Value Problems in Ordinary Differential Equations*. Academic Press Inc. 1988.

[SPE07] **S. Popprath, H. Ecker**. *Nonlinear dynamics of a rotor contacting an elastically suspended stator*. Journal of Sound and Vibration, 2007.

[TEN06] **T. Ernst, A. Nordwig**. *Mosilab: Mosila Modelbeschreibungssprache*. 2006. http://swt.cs.tu-berlin.de/lehre/mosim/ws0607/unterlagen/mosim-2006-11-28-ModelicaConference05.pdf

[TJK04] **T. J. Koo**. *EECE 396-1 Hybrid and Embedded Systems Computation.* Institute for Software Integrated Systems, Department of Electrical Engineering and Computer Science, Vanderbilt University, 2004.

[TPC08] **T. Pulecchi, F. Casella**. *HyAuLib: Modelling Hybrid Automata in Modelica*. Politecnico di Milano, Dipartimento di Elettronica e Informazione Piazza Leonardo da Vinci 32, 20133 Milano, Italy, 2008.

[TSM09] **T. Stykel**. *Model Reduction of Differential Algebraic Equations*. DFG Research Center Matheon, Technische Universität Berlin, 2009.

[WBP93] **W. Boehm, H. Prautzsch**. *Numerical Methods*. Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 1993.

[WBR08] **W. Bahmen, A. Reusken**. *Numerik für Ingenieure und Naturwissenschaftler*. Springer-Verlag, Berlin Heidelberg, 2008.

[WGC85] **W. Gander.** *Computermathematik*. Birkhäuser-Verlag, Basel, 1985.

[WGK03] **W. Greiner**. *Klassische Mechanik I, Kinematik und Dynamik der Punktteilchen Relativität*. Verlag Harri Deutsch, 2003.

[WGD03] **W. Greiner**. *Klassische Mechanik, Teilchensysteme, Lagrange-Hamiltonische Dynamik, Nichtlineare Phänomene II*. Verlag Harri Deutsch, 2003.

[WHS67] **W. Hahn**. *Stability of Motion, Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellung*. Springer-Verlag, 1967.

[WMP06] **W. Mack, M. Plöchl**. *Stabilitätsprobleme Bewegter Systeme. Vorlesungsskriptum zur LVA-Nr. 309.023.* TU-Wien, Studienjahr, 2005/2006.

[WPW01] **W. Preuss, G. Wenisch**. *Lehre und Übungsbuch Numerische Mathematik mit Softwareuntersstützung*. Carl Hanser Verlag München Wien, 2001.

11 Curriculum Vitae

Personal Details

Rouzbeh Karim
Born July 22.1966, in Tehran/Iran
Address: Währingerstr. 202/4, 1180-Vienna/Austria
rouzbeh_karim@hotmail.com
rouzbehkarim66@gmail.com

Education

2012 - 2016	Vienna University of Technology, Doctoral Study with Emphasis in field of State Event Modelling and State Event Handling in Hybrid Systems.
2012	Vienna University of Technology, Change in Emphasis of Doctoral Study.
2007 - 2012	Vienna University of Technology, Doctoral Study in Field of Hybrid System Modelling and Simulation in Mechatronics.
2005-2006	Vienna University of Technology, Research in field of Numerical Solution of Systems Including Extended Kalman Filter and Simulation of Soccer Robots.
2005	Vienna University of Technology, Inscription in Doctoral Program.
2005	Vienna University of Technology, Completion of Studies for the Diploma Program in Electrical Engineering with Concentration in Computer Technology.
2004-2005	Vienna University of Technology, Diploma Thesis: Approximate In-and Output Linearization of Nonlinear Plants by Neuronal Network and Control Itself.
2003-2004	Vienna University of Technology, Computer Laboratory Project: Implementation of Multithread-Socket-Programs in C Language for a Client-Server Communication between distributed Computers.
1999-2005	Vienna University of Technology, Study Electrical Engineering with Concentration in Computer Technology.
1998	Vienna University of Technology, Validation of Foreign Certificate after Recognition of Graduations of Supplementary Lectures in field of Automation and Control.

1995-1998	Vienna University of Technology, Study Electrical Engineering with Concentration in Automation and Control.
1994-1995	Technische Universität Wien, Study Informatik.
1992	Azad University of Tehran, Diploma of Completion of Bachelor Studies for the Electrical Engineering in the Branch of Study Electronics.
1992	Azad University of Tehran, Diploma Thesis: Circuit Analysis of Two Type Televisions and Providing Television Labor Lecture Notes.
1989-1992	Azad University of Tehran, Bachelor Studies Electrical Engineering in the Branch of Study Electronics.
1989	University Entrance Exam for Bachelor Studies in the field of Electrical Engineering in the Branch of Study Electronics.
1989	Technical and Vocational College Nr.2 Teheran, Diploma of Completion of Associate Course of Studies in Field of Electrical Engineering in the Brach of Study Technician-Electronics.
1986-1989	Technical and Vocational College Nr.2 Teheran, Studies in Field of Electrical Engineering in the Branch of Study Technician-Electronics.
1986	Entrance Exam for Associate Course of Studies in Field of Electrical Engineering Technician-Electronics.

Work Experience

2014/10-2014/12	Hardware Engineer, llynx Electronic GmbH, Vienna. Analysis of Functionality of the Switching Power Supply for LED Circuits and Programing of Microcontroller in C Language.
2012/04-2013/06	Hardware Developer and Programmer, Sigmatek, GmbH & Co KG, Vienna/Salzburg. Hardware Simulating with LTSpice and Hardware Development with PAD in Field of Industrial Computer and Software Development for TI AM3359 PRU (DSP) with PASM Language under Code Compose Studio 5.3 for Data Communication Between VARAN Master-Slave Units. Analyzing and Computing Reaction Time of a Distributed System as well as Implementing CRC Procedure for Distributed System.
2006/10-2007/09	Automation and Control Engineer, Process Control Engineering (PCE) GesmbH, Vienna. Drawing Automation Circuit Diagram in Pulp and Paper Industry with WSCAD. Consulting in Field of Database and Control System.

2002/03-2003/06	Scientist (Freelancer), AIT GmbH (Traffic Telematics), Vienna. Researching, Development, Simulating and Implementing Algorithms for Kalman Filter System, Neuronal Networks as well as a Fourier Transform GUI Using Matlab, Simulink, C and MSVC++ Tools.
2000/02-2001/09	Hardware Developer (Trainee), Siemens Wien. Programming Graphic Mixer Unit for Fujitsu Monitor, a Reuse Round-Robin Arbiter in VHDL and Providing Reuse Programming Style for VHDL.
1994/03-1994/08	Automation Engineer, Tarsam, Iran/Teheran/Karaj. Automation of Vacuum forming Packing Machine for Food Industry. Identifying Processes, Sensors and Actuators of a Vacuum Forming Packing Machine. Implementing Logical Plan, PLC Program to Control Processes as well as Design an Electronic Circuit for Test of PLC Inputs and Outputs.
1992	Electronic Technician (Freelance), Iran/Teheran. Design and Development of a Digital Electronic Cuircuits for Speed Measurment of Peykan Vehicle. Design and Computation of Logic and Cuircuit of a Judo Scoreboard.

Extra-Curricular Courses

2014	Academic Centre of Vienna, Project Management, Modern Management
2013	WIFI- Wien, Object Oriented Analysis and Design with UML.
2013	WIFI-Wien, Android Application Development, Apps for Android Development.

Papers

2012	Parameter Approximation of Multiple Inputs and Multiple Outputs of Dynamic System Using Least Square Method. www.textfeld.ac.at.
2013	Approximation of Event Coordinates of a Multifunction and Multivariable Algebraic Model by Newton's Method. EUROSIM 2013.

Personal Skills

Competences	Systems Analyzing, Design, Modelling, Programming, Simulating, Developing and Implementing.

Hobbies	Music Composing, Playing Synthesizer and Keyboard, Painting, Sculpting, Swimming and Skiing, Electronic Kits and Assembling Sets, Live Music, Concerts etc.
Other Interest	Science, Technology, Art, Diving, Climbing, Hopkido, Jugging, Yoga, Relaxation, Nature, Fun, Silence etc.