Evaluation of 'State of the art' numerical integration schema

Cristiana Maria Rodrigues de Azevedo

Dissertação para a obtenção do grau de Mestre em Engenharia Química pela Faculdade de Engenharia da Universidade do Porto

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

tese realizada sob a orientação da

Doutora Maria Joana Monteiro de Carvalho Peres

e co-orientação de

Dipl. Ing. Moritz von Stosch

Porto, July 2011

Agradecimentos

À Doutora Joana Peres pelo seu olhar crítico e opinião prática sobre todos os assuntos.

E ao *quasi*-Doutor Moritz von Stosch por tudo.

One night Sine and Cosine are hanging out in a bar, having a good time. After a while they realize that their mate, Exp, is sitting lonesome in a corner, making a rather sad face. So Sine decides to go over there and he asks him: "Hey mate, how's it going? You have to mingle, integrate yourself!"The Exp answers: " As if that would make a difference!"

Resumo

Equações Diferenciais Ordinárias (ODEs) de primeira ordem são utilizadas frequentemente em Engenharia Química. Enquanto que a sua integração analítica é muitas vezes, impossível, encontrar uma solução numérica é fazível. Diferentes casos requerem aplicações de diferentes soluções, e por isso, de diversos algoritmos para a solução numérica de ODEs de primeira ordem, que se encontram prontamente aplicáveis. Aquando da utilização destes algoritmos, é um pré-requisito conhecer as suas limitações.

Na presente tese, foi colectado um conjunto de métodos numéricos para ODEs de primeira ordem, implementados para MATLAB, i.e. (i) metodologias instaladas no próprio software (ode15s, ode23s, ode23t, ode23, ode45 e ode113); e (ii) outros métodos sem custo disponíveis na internet (euler_forward, euler_modified, euler_backward, heun, opt_rk2, rk4, ab2, adams_pc4, rkf45, rkv56, vs_pc4, ode87 e rock4), são comparados na análise de dois casos de estudo (A e B). A comparação é baseada em critérios de avaliação que tomam em conta a qualidade e os custos computacionais associados. Para o primeiro caso de estudo, A, um sistema linear de ODEs, moderadamente "stiff", com solução analítica conhecida, vários cenários foram estudados à cerca de mudanças aplicadas na metodologia e na condição inicial do caso de estudo). Pode-se concluir que:

i) O método rkv56 tem a melhor performance em termos de qualidade, mas associada a um custo computacional relativamente elevado;

ii) Fornecer informação adicional sob a forma do Jacobiano não teve impacto na qualidade nem no custo computacional da solução, exceptuando para o método ode23s onde o número de avaliações de funções diminuiu significativamente.

iii) Fornecer um valor passo inicial para os métodos de passo adaptativo teve pouca influẽncia na solução numérica.

iv) No caso de metodologias de passo fixo (euler_forward, euler_modified, euler_backward, heun, opt_rk2, rk4, ab2 e adams_pc4) terem de ser aplicadas para e.g. ajustar o tamanho do passo a certas frequẽncias amostrais, os métodos heun e rk4 são boas escolhas na medida em que podem ser facilmente adaptadas, e produzem resultados muito superiores aos dos métodos euler_forward ou euler_backward.

v) De igual modo, no caso dos métodos de passo fixo, os Erros Locais Estimados (LEEs) fornecem um limite superior aos Erros Locais (TLE), mas sendo a ordem de magnitude dos LEEs bastante superior.

vi) Para as metodologias de passo adaptativo (ode15s, ode23s, ode23t, ode23, ode45, ode113, rkf45, rkv56, vs_pc4, ode87 e rock4), os LEE não providenciam sempre um limite superior aos TLE, mas a ordem dos LEE é bastante próxima da ordem dos TLE.

vii) No caso dos métodos "stiff"(ode15s, ode23s, ode23t e rock4) o custo computacional das soluções não foi inferior que o correspondente para os métodos não-"stiff", confirmando que o caso de estudo A é moderadamente "stiff".

viii) Perturbações na condição inicial causam (no presente caso de estudo) erros muito superiores a erros devido à integração numérica. Contudo, a metodologia contribui para a propagação do erro.

ix) Certos métodos (rock4, rkv56, rkf45, vs_pc4 e ode87) não providenciam a solução em pontos do tempo especificados a priori, sendo necessário usar uma interpolação. Ao aplicar a interpolação cúbica de Hermite a soluções numericas com alta exactidão (sob tolerâncias rigorosas), o erro devido à interpolação é superior ao erro causado pela integração.

No caso do segundo caso de estudo, B, um sistema de ODEs não-linear, sem solução analítica conhecida, cenários semelhantes foram estudados. A partir dos resultados pode-se concluir que:

i) O método rkv56 teve, mais uma vez, uma alta performance, mas com custos computacionais bastante mais elevados que para a maioria das restantes metodologias.

ii) O custo computacional dos métodos "stiff", é em geral, significativamente inferior ao dos métodos não-"stiff", sendo tal uma prova que o caso de estudo B é "stiff".

iii) No caso em que a função do Jacobiano foi fornecida (apenas aplicável aos métodos "stiff"), foram observados menos avaliações de funções, e por isso tempos computacionais inferiores (devido ao facto de o Jacobiano não ter que ser calculado através de Diferenças Finitas). No entanto, não houve mudança quanto à qualidade da solução numérica.

iv) No exercício de perturbação dos valores iniciais, as contribuições para o erro não puderam ser especificadas, devido a não haver solução analítica conhecida. No entanto, o comportamento oscilatório do método ode23t pode ser mais uma vez observado. Este comportamento está de acordo com resultados observados para o caso de estudo B, em que o ode23t apresentou um comportamento que pode ser interpretado como propagação de erro.

Palavras chave: ODE; stiff; método numérico; solução analítica; perturbação valores iniciais; exactidão

Abstract

First-order Ordinary Differential Equations are frequently encountered in chemical engineering. While their analytical integration is, many times, not possible, the numerical solution may still remain feasible. Different cases require the application of different solutions and thus various algorithms for the numerical solution of first order ODEs, that are readily applicable, can be found. When relying on these algorithms, it is a prerequisite to know for their limitations.

In this thesis, a number of first order ODE solution schema that were encountered for the MATLAB environment, i.e. (i) schema that come with the software (ode15s, ode23s, ode23t, ode23, ode45 and ode113); and (ii) also others that are provided free of cost in the internet (euler_forward, euler_modified, euler_backward, heun, opt_rk2, rk4, ab2, adams_pc4, rkf45, rkv56, vs_pc4, ode87 and rock4), are compared for two case studies. The comparison is based on evaluation criteria that take the quality of the solution and the associated computational cost into account. For the first test case, a linear mildly stiff ODEs system with known analytical solution, various scenarios were studied (concerning changes that were applied to the solution schema and the initial condition of test case). It can be concluded that:

i) The rkv56 has the best performance in terms of quality, but its associated computational cost is relatively high.

ii) Providing additional information in form of the Jacobian is found to have no impact on the quality nor on the computational cost of the solution, except for the ode23s where the number of function evaluations decreased significantly.

iii) Providing an initial step size value to the solvers was found to have little influence on the numerical solution.

iv) In the case that the fixed step size solvers (euler_forward, euler_modified, euler_backward, heun, opt_rk2, rk4, ab2 and adams_pc4) have to be applied to e.g. fit the step sizes to a certain sampling frequency, the heun or the rk4 schema should be the matter of choice since they can be relatively easy adapted and give far by far better results than the euler_forward or the euler_backward schema.

v) Also, in the case of the fixed step size solvers, the Local Error Estimates (LEEs) provide an upper bound to the True Local Error (TLE) but the order of the LEE is much higher.

vi) For the adaptive step size solvers (ode15s, ode23s, ode23t, ode23, ode45, ode113, rkf45, rkv56, vs_pc4, ode87 and rock4), the LEE does not always provide an upper bound to the TLE, but the order of the LEE is close to the one of the TLE.

vii) In case of the stiff solvers (ode15s, ode23s, ode23t and rock4), the computational cost of the solution was not lower than for the non-stiff solvers, another evidence that this test case is mildly stiff.

viii) In case of the disturbed initial value, it could be observed that in this particular case the error caused through the disturbance is much higher than the one caused by the numerical integration. However the numerical integration contributes to error propagation.

ix) Some of the the numerical schema (rock4, rkv56, rkf45, vs_pc4 and ode87) do not provide the solution at the prior specified time instances, wherefore the utilization of an interpolation schema becomes necessary. When applying a Hermite interpolation schema it was observed that when the numerical solution should have a high accuracy (Stringent integration tolerances) then the error that is due to the interpolation is higher than the one that is caused by the integration.

In case of the second test, a non-linear, stiff and time-variant ODE system, without known analytical solution, identical scenarios were studied. From the results it could be concluded that:

i) The rkv56 also performed very well, but the computational cost was much greater than the one observed for most of the other solvers.

ii) The computational cost of the stiff solvers in general was significantly lower than the one of the non-stiff solvers, which is another proof that this case is stiff.

iii) In case that additionally a Jacobian function was provided (only applicable for the stiff solvers), lower computational times and lower number of function evaluations were observed (since the Jacobian does not have to be calculated through Finite Differences), however there was no change to the quality of the numerical solution.

iv) For the disturbed initial values the contributions to the error could not be specified since the analytical solution is unknown. Nevertheless for the ode23t oscillatory behavior could be observed. This behavior was in accordance with results made for the other cases, where the ode23t showed a behavior that can be interpreted as error propagation.

Keywords: ODE; stiff; numerical scheme; analytical solution; disturbance initial values; accuracy

Index

List of Figures

List of Tables

Nomenclature

A	Jacobian matrix
A, B and C	Chemical species in teste case B
ae	Vector absolute tolerance values
EPA	Criteria for the error propagation analysis
f	ODE system
h	Step size
$k_1, ...k_{11}$	Variable integration coefficients or Kinetic parameters in Test Case B
n	Dimension of Real value space or number of state variables
p	Order of the numerical integration method
q	Integer parameter for the LEE
R	Real value space
re	Scalar relative tolerance value
t	Time
T	End of integration interval
W	Number of time instances
x	State of the ODE system
y	State of the ODE systems in Test Cases A and B
$z_{experimental}$	Experimentally measured values

Greek Symbols

κ	Condition Number
κ_{rel}	Relative Condition Number
$\lambda_1, ..., \lambda_3$	Eigenvalues in Test Case B
τ	Ratio between the tolerances

Indicies

0	Index that designates initial condition
5%	Index that specifies a disturbance of +5%
ana	Index that specifies the solution was obtained analytically
dis	Index that specifies that the solution was disturbed
i	Index that specifies the state
j	Index that specifies the time instant
n	Current time instance
new	Index that refers to a new numerical solution
num	Index that specifies the solution was obtained numerically
old	Index that refers to an old numerical solution

Abreviations

$AbsTol$	Absolute Tolerance
IVP	Initial Value Problem
LEE	Local Error Estimate
$MaxRelE2$	Maximum Relative Error 2
ODE	Ordinary Differential Equation
$RE1$	Relative Error 1
$RE2$	Relative Error 2
$RelTol$	Relative Tolerance
TLE	True Local Error

Chapter 1

Introduction

Mathematic is the language of Nature.

In all fields of science, i.e. physics, biology, chemistry or engineering, there is the constant need to express the behavior of a system by a manageable language, Belardini & al. (2005); Burghes and Borrie (1981); Butcher and Wanner (1996); Daun & al. (2008); Gladwell & al. (2003); Krogh (1973); Oster and Perelson (1974). Given that the system is "big" enough, so that uncertainty is not a problem, one can use a deterministic framework (e.g. Differential Equations (DEs)) to represent the behavior (else, i.e. uncertainty plays a major role and thus DEs are unreliable, stochastics would have to be considered (Daun & al., 2008)).

In chemical engineering it is many times tented to model complete processes. For this purpose the conservation laws (material, momentum and energy balances) are usually applied. On the macroscopic scale these balances are based on DEs, since the system can be assumed to be sufficiently large. If the system can further be assumed to be homogeneous it can be described through Ordinary Differential Equations (ODEs) (else, i.e. the system is heterogeneous, Partial Differential Equations (PDEs) will be needed for its description). In fact a huge number of process models is based on ODEs (in Science direct about 58000 journal entries are found for the keywords "ordinary differential equations" and "process models" and still about 19000 for "ordinary differential equations" and "chemical engineering").

While it is most times not possible to solve ODEs analytically (Gladwell & al., 2003), not even with the help of computers by so called "symbolic manipulation" (MAPLE or MATLAB), the solution through numerical schema still remains feasible. Much research has been going on in this direction and a number of very efficient schema have been proposed Brugnano and Trigiante (1998); Butcher and Wanner (1996); Hairer and Wanner (1993); Petzold (1981); Rentrop (1985); Shampine (1981); Shampine and Gladwell (1996) mostly for first-order ODEs. However different cases require the application of different solutions, i.e. ODE systems might be stiff or non-stiff, or in other cases the application of either fixed step size or adaptive steps size methods might be advantageous. Many of these schema found its way into commercial mathematical software but also free (of cost) solutions are provided e.g. in the internet. When relying on these numerical integration schema, it is valuable to know for their limitations and/or their advantageous. For the assessment of the 'capabilities' of a numerical integration algorithm, several defined test cases exist Enright & al. (1975); Hull & al. (1972); Krogh (1973); Seinfeld and Lapidus (1973). The most wildly used test case families of non-stiff and stiff ODEs, can be found in Enright & al. (1975) and (Hull & al., 1972) which belong to the same research group. Several families of test cases proposed by these authors still remain challenging today. Corresponding to Enright & al. (1975), Shampine (1981) suggested some extensions and/or corrections to the equations and conditions for their solution, in order to make the problems even more challenging. The test cases contained in Krogh (1973) are more directed to the fields of Mechanics, and Celestial Body Orbits. In Seinfeld and Lapidus (1973) the aim of the author when assembling the test cases was to have some ODEs systems with known analytical solutions, wherefore the numerical calculation can be checked for accu-

racy. In order to be capable to compare the results, Shampine (1981) and Krogh (1973), proposed some criteria to address the quality of the numerical solution. In Krogh (1973) it was also proposed to utilize the number of total function evaluations as the computational cost that is associated to the numerical solution, since the duration time of an integration depends dramatically on the hardware.

In this thesis, several numerical algorithms for the integration of first-order differential equations (available for the MATLAB environment) will be applied to various scenarios of two test cases and the obtained numerical solution will be evaluated based on criteria that address the quality and the associated computation cost.
At first, a brief introduction to numerical integration is provided and several numerical schema for the integration of first order ODEs (that were readily available for the application with MATLAB) are briefly presented. Then criteria are defined that allow to assess the quality of the numerical solution and the associated computational cost. The integration algorithms are then applied to two test cases, namely Test Case A, which was taken from (Seinfeld and Lapidus, 1973) and Test Case B (the Robertson‘s equations) which was taken from (Enright *&* al., 1975) . Several changes to the parameters of the integration schema and a disturbance to the initial values of the ODE systems were performed and the obtained numerical solutions thereupon evaluated through the defined criteria.

Chapter 2

Numerical Integration Methods

2.1 Mathematical Preliminaries

Initial Value Problems

This thesis evaluates methodologies that aim to find a numerical approximation to the solution of first order Ordinary Differential Equations (ODEs) in the form of Initial Value Problems (IVP) such as:

$$x' = f(t, x), \quad x(t_0) = x_0, \quad 0 \leqslant t \leqslant T, \tag{2.1}$$

where x_0 is a given vector, so called initial condition or initial value (not necessarily at $t_0 = 0$), in the n-dimensional real vector space R^n, n>1, and x(t) ($\in \mathrm{R}^n$) is unknown for $0 < t \leq T$. Depending on the underlying set of equations, f, the ODEs system can be divided into the classes of stiff and non-stiff systems.

Stiff and Non-Stiff ODEs Systems

The *a priori* classification whether a system is non-stiff or stiff is not straightforward, and still subject of study (Petzold, 1983; Rentrop, 1985; Shampine, 1977; Shampine and Hiebert, 1977). Stiff ODEs systems arise in many areas, such as Chemical Kinetics, particularly when simultaneously rapid and slow reactions take place, e.g. combustion or polymerization reactions. Stiffness can also evolve from drastic changes to the steady state, e.g. set-point changes.

If the computational cost for numerical solutions (duration time) would not be of concern, stiffness would not be an issue, in the sense that any non-stiff solver can integrate a stiff ODEs system, but at a higher expense (large computational time). This fact can be used as an heuristic definition on what stiffness is, i.e.: Ashino and Vaillancourt (2009) wrote:"*Stiff equations are equations where (certain) implicit methods, in particular backward differentiation methods, perform much better than explicit ones*". Therefore stiffness can be seen as an efficiency parameter when comparing the computational cost of different numerical solution schema. It has to be kept in mind that (non-)stiffness is a property of the ODEs system and not of the numerical solution scheme at hand.

One Step Methodologies and Multistep Methodologies (Methodologies with Memory)

One Step Methods only utilize information about the solution at a single point $x = x_n$, from which the method calculates x_{n+1} at the next point $t = t_{n+1}$, which is the reason for their name. However, one step methods can be multistage, which means they evaluate the slope of the function f at several sub intervals of the step h.

Multistep Methods incorporate information about the previous $n-i$ points for the calculation of x_{n+1}, wherefore they are prone to error propagation. However those methodologies can be expected to have a lower computational

cost since these methodologies only calculate the slope once per step h. Both methodologies are exemplary shown in Figure 2.1 .

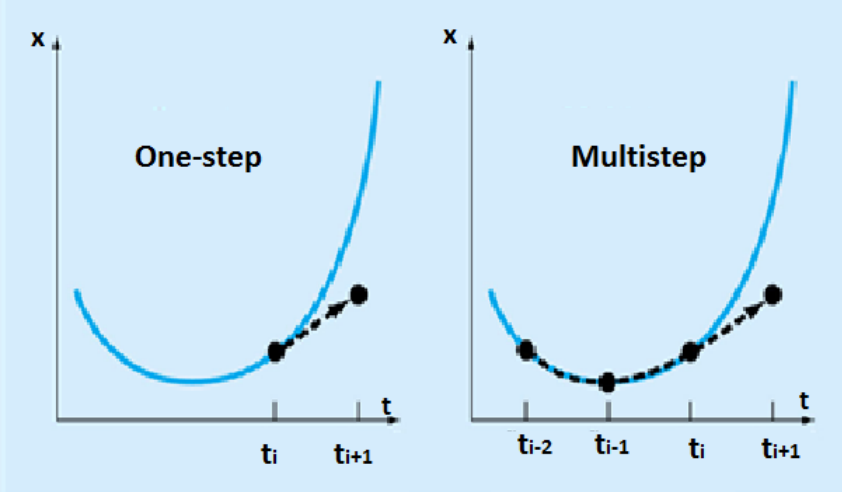

Figure 2.1 Exemplary sketch of one step methods (left) and multistep methods (right).

Explicit and Implicit Methods

An explicit method is explicit in respect to the calculation of x_{n+1} in that the derivative, f, is calculated at instances of t and x which are already known, e.g. $x_{n+1} = x_n + h \cdot f(t_n, x_n)$, which is the forward Euler schema. In contrast an implicit method contains the term x_{n+1} on both sides of the equation, i.e.: $x_{n+1} = x_n + h \cdot f(t_{n+1}, x_{n+1})$, which is the backward Euler method. One must calculate x_{n+1} on the right hand side of the ODE by a recurrence formula, Ashino and Vaillancourt (2009).
One could then (erroneously) assume that an explicit solver should be preferred to an implicit one due to the higher computational costs associated with the implicit scheme. But, when integrating stiff ODEs, explicit solvers require impractically small step sizes in order to maintain the local error estimation under the tolerance requirements. Therefore, as mentioned above, implicit methodologies are used almost exclusively to integrate stiff ODEs systems.

In contrast to forward Euler and backward Euler methods, which are fully explicit and implicit methods, respectively, most methodologies rather combine explicit and implicit schema. A good example are the $Predictor - Corrector$ Methods. These schema usually incorporate an explicit formula, the *Predictor* in order to provide a first solution for x_{n+1} at t_{n+1}. Then, in general, an implicit *Corrector* scheme is applied for the evaluation of the associated prediction error (Conte and de Boor, 1981).

Fixed and Adaptive Step Size Methods

Fixed step size methods are only useful up to a certain point because they will use the same step throughout the entire integration interval, not being sensitive to error tolerances. They are ideal when dealing with ODEs with a behavior close to linearity.
Fixed step size methods will (always) converge. Still, convergence does not imply accuracy. In practice, it is advantageous to let the step size h vary so that h is taken larger when $x(t)$ does not vary rapidly and smaller when $x(t)$ changes rapidly (Ashino and Vaillancourt, 2009).
A class of methods, that can vary the step size automatically, is referred to as Embedded Pairs (e.g., of *Runge-Kutta*) Methods. These methods estimate the solution x_{n+1} twice with two different order methods and then

calculate a local error estimate (LEE) associated with the difference between both solutions. If the LEE is smaller than a given tolerance, the method will increase the step size h. On the other hand, if the LEE is above the tolerance, the method will decrease the step size and repeat the calculation of x_{n+1} until the above condition is obeyed.

In the following the numerical integration schema for the solution of first order differential equations (that are dealt with in this thesis) will be classified, as shown in Figure 2.2.

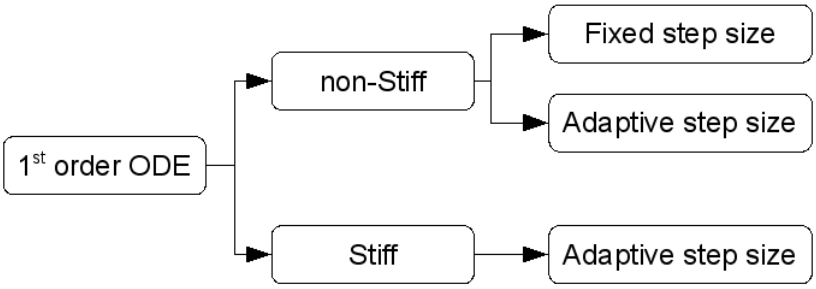

Figure 2.2 Schematic sketch of the section structure.

2.2 Non-Stiff Solvers

2.2.1 Fixed Step Size Solvers

euler_forward

This code was provided by David Houcque - Northwestern University. The implemented version was slightly modified so that also ODE systems can be solved instead of only one ODE.
This is a one-step, fully explicit method of first order, also know as Taylor algorithm of order one or Runge–Kutta method of order one. Its solution is given by:

$$x_{n+1} = x_n + h \cdot f(t_n, x_n), \tag{2.2}$$

where x_n is a vector of the state values at point n, h is the step size, f is the ODE system which is evaluated at time t_n and x_n, in order to obtain the solution at x_{n+1}. This method many times finds application when experimental data, $z_{experimental}$, are used as inputs to $f(t, x, z_{experimental})$ since it is easy to adapt the method to the sampling frequency of the experimental data.

euler_backward

This code also was provided by David Houcque - Northwestern University. The implemented version was slightly modified so that also ODE systems can be solved instead of only one ODE. This is a multistep, fully implicit first order method which is also referred to as implicit euler method or Backward Differential Formula 1 ($BDF1$). The formula is given by:

$$\begin{aligned} x_{new} &= x_n + h \cdot f(t_n, x_n) \\ x_{n+1} &= x_n + h \cdot f(t_n, x_{new}) \end{aligned} \tag{2.3}$$

Since the backward euler method is implicit, the scheme of equation 2.3 at first calculates an estimate of x_{new} using the forward euler method. Then, in a second step, x_{new} is utilized to calculate x_{n+1}. This method, therefore,

should almost be twice as computational intensive as the forward euler method. This method can also be easily adapted for the case that experimental data, $z_{experimental}$, are used as inputs to $f(t, x, z_{experimental})$.

euler_modified

Two codes were provided for the modified Euler method. The first, addressed here, is called euler_modified and was provided by David Houcque - Northwestern University. His implemented version was slightly modified so that also ODE systems can be solved instead of only one ODE. The second code will be addressed in the section below. This is a multistep, second order method, which is also known as improved euler method, heun method, trapezoidal rule or Runge-Kutta method of order two.
Its solution schema is given by:

$$\begin{aligned} x_{new} &= x_n + h \cdot f(t_n, x_n) \\ x_{n+1} &= x_n + \tfrac{h}{2} \cdot (f(t_n, x_n) + f(t_{n+1}, x_{new})) \end{aligned} \tag{2.4}$$

This method, for the same step size, can be expected to be more accurate than the two methods given above since it is of second order. Further it can be expected that almost the same computational effort is required for the solution than as for the backward euler method. Therefore it should be applied whenever experimental data, $z_{experimental}$, are used as inputs to $f(t, x, z_{experimental})$.

heun

This method is the same as the modified euler, but this code was provided by Bryan Bradie - Newport University. Even though the method is the same the implementation is different, wherefore both codes are kept.

opt_rk2

This code was also provided by Bryan Bradie - Newport University. The file name comes from the complete name optimal-Runge-Kutta of second order. The equations are given by:

$$\begin{aligned} f_n &= f(t_n, x_n) \\ x_{new} &= x_n + \tfrac{2 \cdot h}{3} \cdot f_n \\ x_{n+1} &= x_n + \tfrac{h}{4} \cdot (f_n + 3 \cdot f(t_n + \tfrac{2 \cdot h}{3}, x_{new})) \end{aligned} \tag{2.5}$$

Its schema is similar to the modified euler or heun schema, just that in this case the function is evaluated at two-thirds of the step length and accordingly weighted differently.

rk4

This code as well was provided by Bryan Bradie - Newport University. The abbreviation rk4 stands for Runge-Kutta of fourth order. The solution schema is given by:

$$\begin{aligned} k_1 &= f(t_n, x_n) \\ k_2 &= f(t_n + \tfrac{h}{2}, x_n + \tfrac{k_1}{2}) \\ k_3 &= f(t_n + \tfrac{h}{2}, x_n + \tfrac{k_2}{2}) \\ k_4 &= f(t_n + h, x_n + k_3) \\ x_{n+1} &= x_n + (k_1 + k_2 + k_3 + k_4) \cdot h/6 \end{aligned} \tag{2.6}$$

where it can naturally be expected that the accuracy of the solution obtained with the same step size is better than for the above mentioned which have lower order.

ab2

This code was also provided by Bryan Bradie - Newport University. The abbreviation relates to an Adams Bashford second order method, which uses in this case the opt_rk2 method for the initialization. The Adams Bashford schema reads as:

$$\begin{aligned} f'_{new} &= f(t_n, x_n) \\ x_{n+1} &= x_i + \tfrac{3 \cdot h}{2} \cdot f'_{new} - \tfrac{h}{2} \cdot f_{old} \end{aligned} \tag{2.7}$$

This method is explicit and multistep.

adams_pc4

This code also was provided by Bryan Bradie - Newport University. The code name comes from the Adams predictor-corrector fourth order method and uses in this case the classical rk4 method (equations 2.6) to generate the starting values. The Adams predictor-corrector schema reads as:

$$\begin{aligned} t_n = t_{n+1}, x_n = x_{n+1} \\ f_{new} = f(t_n, x_n) \\ x_{new} = x_{n+1} + \tfrac{h}{24} \cdot (55 \cdot f_{new} - 59 \cdot f_{new-1} + 37 \cdot f_{new-2} - 9 \cdot f_{new-3}) \\ f^{(0)}_{new} = f(t_n + h, x_{new}) \\ x^{(k)}_{n+1} = x_n + \tfrac{h}{24} \cdot (9 \cdot f^{(0)}_{new} + 19 \cdot f_{new} - 5 \cdot f_{new-1} + f_{new-2}) \end{aligned} \tag{2.8}$$

2.2.2 Adaptive Step Size Methods

ode23

The solver ode23 of the *MATLAB ODE suite* is an implementation of a four-stage pair of embedded explicit Runge–Kutta methods of orders two and three with error control. The higher-order solution, namely third order, is used to produce the solution (Ashino and Vaillancourt, 2009), so called local extrapolation, and the second order is used to control the error. The four stages according to Ashino and Vaillancourt (2009) are: "

$$\begin{aligned} k_1 &= h \cdot f(t_n, x_n) \\ k_2 &= h \cdot f(t_n + \tfrac{h}{2}, x_n + \tfrac{k_1}{2}) \\ k_3 &= h \cdot f(t_n + \tfrac{3}{2} \cdot h, x_n + \tfrac{3}{4} \cdot k_2) \\ k_4 &= h \cdot f(t_n + h, x_n + (\tfrac{2}{9}) \cdot k_1 + (\tfrac{1}{3}) \cdot k_2 + (\tfrac{4}{9}) \cdot k_3) \end{aligned} \tag{2.9}$$

The first three stages produce the solution at the next time step:

$$x_{n+1} = x_n + (\frac{2}{9}) \cdot k_1 + (\frac{1}{3}) \cdot k_2 + (\frac{4}{9}) \cdot k_3 \tag{2.10}$$

and coefficient $k_4$is used for local error estimate calculation:

$$LEE = \frac{-5}{72} k_1 + \frac{1}{12} k_2 + \frac{1}{9} k_3 - \frac{1}{8} k_4 \tag{2.11}$$

However, this is really a three-stage method since the first step at x_{n+1} is the same as the last step at x_n, that is $k_1^{[n+1]} = k_4^{[n]}$."

An alternative way to present the information, namely the coefficients, that are comprised by equation 2.9, is through a so called Butcher Tableau. For the present method, the ode23, the link between the equations and the Butcher Tableau (shown in Table 2.1) is obvious. For an extensive explanation see Ashino and Vaillancourt (2009).

Table 2.1 Butcher Tableau for ode23.

	c	A			
k_1	0	0			
k_2	1/2	1/2	0		
k_3	3/2	0	3/4	0	
k_4	1	2/9	1/3	4/9	0
y_{n+1}	b^T	2/9	1/3	4/9	0
LEe		-5/72	1/12	1/9	-1/8

ode45

The solver ode45 of the *MATLAB ODE suite* is an implementation of the explicit Dormand-Prince (5,4)7 embedded Runge-Kutta formula (where the number 5 means that the solution is advanced with fifth order, the number 4 relates to the calculation of the local error estimate and the number 7 means that the method utilizes seven variable parameters (Ashino and Vaillancourt, 2009)). The coefficients $k_{1,\dots,7}$ of the ode45 are displayed in a Butcher tableau, Table 2.2, for a further extensive explanation of the method see (Ashino and Vaillancourt, 2009).

Table 2.2 Butcher Tableau for ode45.

	c	A						
k_1	0	0						
k_2	1/5	1/5	0					
k_3	3/10	3/40	9/40	0				
k_4	4/5	44/45	-56/15	32/9	0			
k_5	8/9	$\frac{19372}{6561}$	$\frac{-25360}{2187}$	$\frac{64448}{6561}$	$\frac{-212}{729}$	0		
k_6	1	$\frac{9017}{3168}$	$\frac{-355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$\frac{-5103}{18656}$	0	
k_7	1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$\frac{-2187}{6784}$	$\frac{11}{84}$	
$\hat{y}_{n+1}$	$\hat{b}_T$	$\frac{5179}{57600}$	0	$\frac{777}{16695}$	$\frac{393}{640}$	$\frac{-92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$
y_{n+1}	b^T	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$\frac{-2187}{6784}$	$\frac{11}{84}$	0
$b^T - \hat{b}_T$		$\frac{71}{57600}$	0	$\frac{-71}{16695}$	$\frac{71}{1920}$	$\frac{-17253}{339200}$	$\frac{22}{525}$	$\frac{-1}{40}$
$y_{n+0.5}$		$\frac{5783653}{57600000}$	0	$\frac{466123}{1192500}$	$\frac{-41347}{1920000}$	$\frac{16122321}{339200000}$	$\frac{-7117}{20000}$	$\frac{183}{10000}$

ode113

The code ode113 (comprised in the *MATLAB ODE suite*) is an adaptive step size, variable order method which uses an Adams–Bashfourth–Moulton predictor-corrector of first to twelfth order, (Ashino and Vaillancourt, 2009). Since this is a predictor corrector method of variable order, the complexity and length of the implementation is high, therefore see Ashino and Vaillancourt (2009) for further details.

rkf45

This code was provided by Bryan Bradie - Newport University. His implementation does not allow the user to specify at which time instances, other then the end of the integration interval, the ODEs solution is calculated. Further, the only parameters that can be changed by the user are the absolute tolerance, the minimum step size and the maximum step size. In the test cases studied here, the values of minimum and maximum step size were set at 1E-10 and 1E-1 respectively.
The method is a six-stage Runge–Kutta–Fehlberg pair RKF(4,5) with local error estimate. This method is presented in the form of a Butcher Tableau, see Table 2.3. The estimated local error is obtained using the last row of the table. Again further details can be found in Ashino and Vaillancourt (2009).

Table 2.3 Butcher Tableau for rkf45.

	c	A					
k_1	0						
k_2	1/4	1/4					
k_3	3/8	3/32	9/32				
k_4	12/13	1932/2197	-7200/2197	7296/2197			
k_5	1	439/216	-8	3680/513	-845/4104		
k_6	1/2	-8/27	2	-3544/2565	1859/4104	-11/40	
y_{n+1}		25/216	0	1408/2565	2197/4104	-1/5	0
LEe		16/135	0	6656/12825	28561/56430	-9/50	2/55

rkv56

This code was provided by Bryan Bradie - Newport University. Again, his implementation does not allow the user to specify at which time instances, other then the end of the integration interval, the ODEs solution is calculated. Therefore the implementation was extended, in a second stage (see section 4.1.7), and the extended code is called rkv56_t. Again, the only parameters that can be changed by the user are the absolute tolerance, the minimum step size and the maximum step size. In the test cases studied, the values of minimum and maximum step size were set at, respectively, 1E-10 and 1E-1.
The eightstage Runge-Kutta-Verner pair RKV(5,6) of order 5 and 6 is presented in a Butcher Tableau, see Table 2.4.

Table 2.4 Butcher Tableau for rkv56.

	c	A							
k_1	0	0							
k_2	1/6	1/6	0						
k_3	$\frac{4}{15}$	$\frac{4}{75}$	$\frac{16}{75}$	0					
k_4	2/3	5/6	-8/3	5/2	0				
k_5	5/6	$\frac{-165}{64}$	$\frac{55}{6}$	$\frac{-425}{64}$	$\frac{85}{96}$	0			
k_6	1	$\frac{12}{5}$	-8	$\frac{4015}{612}$	$\frac{-11}{36}$	$\frac{88}{255}$	0		
k_7	$\frac{1}{15}$	$\frac{-8263}{15000}$	$\frac{124}{75}$	$\frac{-643}{680}$	$\frac{-81}{250}$	$\frac{2484}{10625}$	0		
k_8	1	$\frac{3501}{1720}$	$\frac{-300}{43}$	$\frac{297275}{52632}$	$\frac{-319}{2322}$	$\frac{24068}{84065}$	0	$\frac{3850}{26703}$	
y_{n+1}	b^T	$\frac{13}{160}$	0	$\frac{2375}{5984}$	$\frac{5}{16}$	$\frac{12}{85}$	$\frac{3}{44}$		
$\hat{y}_{n+1}$	$\hat{b}_T$	$\frac{3}{40}$	0	$\frac{875}{2244}$	$\frac{23}{72}$	$\frac{264}{1955}$	0	$\frac{125}{11592}$	$\frac{43}{616}$

vs_pc4

This code was, as well, provided by Bryan Bradie - Newport University. His implementation does not allow the user to specify at which time instances, other then the end of the integration interval, the ODEs solution is calculated. In this case, the only parameters that can be changed by the user are, as well, the absolute tolerance, the minimum step size and the maximum step size. In the test cases studied, the values of minimum and maximum step size were set at 1E-10 and 1E-1 respectively.
The code name comes from the full name: Variable Step Adams Predictor Corrector of fourth order. This method is very similar to the fixed step adams_pc4 method but for this method the local error estimate is compared to a user given absolute tolerance which is used to control the step size. The method is initialized, i.e. the starting values are generated, using the classical fourth order Runge-Kutta method, see equations 2.6, after which the

following solution schema is applied:

$$\begin{gathered} t_n = t_{n+1}, x_n = x_{n+1} \\ f_{new} = f(t_n, x_n) \\ x_{new} = x_{n+1} + \frac{h}{24} \cdot (55 \cdot f_{new} - 59 \cdot f_{new-1} + 37 \cdot f_{new-2} - 9 \cdot f_{new-3}) \\ f_{new}^{(0)} = f(t_n + h, x_{new}) \\ x_{n+1}^{(k)} = x_n + \frac{h}{24} \cdot (9 \cdot f_{new}^{(0)} + 19 \cdot f_{new} - 5 \cdot f_{new-1} + f_{new-2}) \end{gathered} \tag{2.12}$$

ode87

This code was provided by Vasiliy Govorukhin -Dept. of Computational Mathematics of the Faculty of Mechanics, Mathematics and Computer Science, Rostov State University, Russia- through the MATLAB code-exchange. His implementation does not allow the user to specify at which time instances, other then the end of the integration interval, the ODEs solution is calculated. The method is a realization of the eighth to seventh order Dorman and Prince formulas (Ashino and Vaillancourt, 2009). The integrator advances with eighth order and requires 13 function evaluations per integration step, for further details (Hairer and Wanner, 1993). The parameters that can be changed by the user are almost identical to the ones of the *MATLAB ODE suite*.
The coefficients of the method are presented in a Butcher tableau, Table 2.5.

Table 2.5 Butcher Tableau for ode87.

	c	A												
k_1	1/18	1/18	0	0	0	0	0	0	0	0	0	0	0	0
k_2	1/12	1/48	1/16	0	0	0	0	0	0	0	0	0	0	0
k_2	1/8	1/32	0	3/32	0	0	0	0	0	0	0	0	0	0
k_3	5/16	5/16	0	$\frac{-75}{64}$	75/64	0	0	0	0	0	0	0	0	0
k_4	3/8	3/80	0	0	3/16	3/20	0	0	0	0	0	0	0	0
k_5	59/400	$\frac{29443841}{614563906}$	0	0	$\frac{77736538}{692538347}$	$\frac{-28693883}{1125000000}$	$\frac{23124283}{1800000000}$	0	0	0	0	0	0	0
k_6	$\frac{93}{200}$	$\frac{16016141}{94662911}$	0	0	$\frac{61564180}{158732637}$	$\frac{22789713}{633445777}$	$\frac{545815736}{2771057229}$	$\frac{-180193667}{1043307555}$	0	0	0	0	0	0
k_7	$\frac{5490023248}{9719169821}$	$\frac{39632708}{573591083}$	0	0	$\frac{-433636366}{683701615}$	$\frac{-421739975}{2616292301}$	$\frac{100302831}{723423059}$	$\frac{790204164}{839813087}$	$\frac{800635310}{3783071287}$	0	0	0	0	0
k_8	13/20	$\frac{246121993}{1340847787}$	0	0	$\frac{-37695042795}{15268766246}$	$\frac{-309121744}{1061227803}$	$\frac{-12992083}{490766935}$	$\frac{6005943493}{2108947869}$	$\frac{393006217}{1396673457}$	$\frac{123872331}{1001029789}$	0	0	0	0
k_9	$\frac{1201146811}{1299019798}$	$\frac{-1028468189}{846180014}$	0	0	$\frac{8478235783}{508512852}$	$\frac{1311729495}{143242823}$	$\frac{-10304129995}{1701304382}$	$\frac{-48777925059}{3047939560}$	$\frac{15336726248}{1032824649}$	$\frac{-45442868181}{3398467696}$	$\frac{3065993473}{597172653}$	0	0	0
k_{10}	1	$\frac{185892177}{718116043}$	0	0	$\frac{-3185094517}{667107341}$	$\frac{-477755414}{1098053517}$	$\frac{-703635378}{230739211}$	$\frac{5731566787}{1027545527}$	$\frac{5232866602}{850066563}$	$\frac{-4093664535}{808688257}$	$\frac{3962137247}{1805957418}$	$\frac{65686358}{487910083}$	0	0
k_{11}	1	$\frac{403863854}{491063109}$	0	0	$\frac{-5068492393}{434740067}$	$\frac{-411421997}{543043805}$	$\frac{652783627}{914296604}$	$\frac{11173962825}{925320556}$	$\frac{-13158990841}{6184727034}$	$\frac{3936647629}{1978049680}$	$\frac{-160528059}{685178525}$	$\frac{248638103}{1413531060}$	0	0
	b8	$\frac{14005451}{335480064}$	0	0	0	0	$\frac{-59238493}{1068277825}$	$\frac{181606767}{758867731}$	$\frac{561292985}{797845732}$	$\frac{-1041891430}{1371343529}$	$\frac{760417239}{1151165299}$	$\frac{118820643}{751138087}$	$\frac{-528747749}{2220607170}$	1/4
	b7	$\frac{13451932}{455176623}$	0	0	0	0	$\frac{-808719846}{976000145}$	$\frac{1757004468}{5645159321}$	$\frac{656045339}{265891186}$	$\frac{-3867574721}{1518517206}$	$\frac{465885868}{322736535}$	$\frac{53011238}{667516719}$	$\frac{2}{45}$	0

2.3 Stiff Solvers

Ode15s

The solver ode15s of the *MATLAB ODE* suite is an implicit solver intended for stiff systems. It has variable order and is based on the numerical differentiation formulas. The ode15s optionally uses the backward differentiation formulas, BDFs. Due to its complexity no further details will be provided here and the reader is referred to Shampine and Reichelt (1997).

ode23s

The solver ode23s of the *MATLAB ODE* suite is a triple of modified implicit Rosenbrock methods of orders three and two with error control for stiff systems. It advances from y_n to y_{n+1} with the second-order method (that is, without local extrapolation) and controls the local error by taking the difference between the third- and second-order numerical solutions. Further details can be found in Ashino and Vaillancourt (2009).

ode23t

The code of the ode23t (*MATLAB ODE suite*) is an implementation of the trapezoidal rule (modified euler). It is a low order method which integrates moderately stiff systems of differential equations, for details see Ashino and Vaillancourt (2009).

rock4

This code is provided by Gerd Steinebach - FH Bonn-Rhein-Sieg, through the MATLAB file exchange. His implementation does not allow the user to specify at which time instances, other then the end of the integration interval, the ODEs solution is calculated. It is an implementation of the rock4 (Abdulle, 2002) for the numerical solution of a (mildly) stiff system of first order differential equations. The parameters that can be specified by the user are identical to the ones of the ode23s of the *MATLAB ODE*.

2.3.1 Factors that might impact on the quality of the solution or on the the associated computational cost

Since the specifications on the quality of numerical solution (to the ODEs system) or the restrictions on the computational time might vary, the user is given the opportunity to adapt the numerical schema through the choice of a few parameters, in order to meet these expectations. The most important parameters in this respect are, without doubt, the values of relative and absolute tolerance for the adaptive step size solvers and the size of the step length for the fixed step size solvers. Thus changes in those parameters were studied for both test cases presented in the following. Further investigations concern the incorporation of additional information, i.e. the initial step size and the Jacobian of the ODEs, since it can be expected that the numerical solution will get the better (in terms of speed and accuracy) the more prior information is provided.
Also given by the user, but without influence on the solvers, are the initial values. They, beside the prior mentioned parameters, might have a significantly impact on the quality of the solution. Even though this is ODEs system dependent it is expected that the numerical scheme might contribute to error propagation. Therefore a disturbance is applied to the initial values of the following test cases and it is studied to what parts the disturbance and the numerical schema contribute to the error.

Chapter 3

Evaluation Criteria

In order to compare the solutions obtained through the numerical integration schema, criteria are required. The criteria introduced in the following address the accuracy and the related computational cost.

3.1 True Local Error - TLE

The True Local Error (TLE) can only be calculated if the analytical solution of the studied ODEs system is known. The TLE is defined by:

$$TLE_{i,1..W} = y_{i,1..W,numerical} - y_{i,1..W,analytical}, \tag{3.1}$$

where the index i refers to each respective state, $1..W$ are the events (e.g. time points), and analytical/numerical specify how the solution was obtained. For a time course the TLE is usually analyzed graphically due to its high dimension, however it is obvious that the TLE can also be analyzed at a specific time points only.

3.2 Local Error Estimate for Fixed Step Size Methods

In case that the analytical solution of an ODE system is unknown, the TLE cannot be calculated. In order to have an idea about the magnitude of the numerical integration error an estimation schema is usually applied. The Richardson extrapolation, or step halving method, for instance provides a Local Error Estimate LEE (see equation 3.2), which gives an upper bound for the TLE, Calvo and Montijano (1996); Shampine and Watts (1976).

$$LEE_{i,1..W} = \frac{\left|(y_{i,1..W,h} - y_{i,1..W,h/2})\right|}{(2^p - 1)}, \tag{3.2}$$

where p refers to the order of the numerical solver applied.

3.3 Local Error Estimate of Adaptive Step Size Methods

For adaptive step size methods the LEE used in this thesis is based on the Tolerance Proportionality method, see Calvo & al. (2008). Its definition is:

$$LEE_{i,1..W} = \frac{\left|(y_{(i,1..W,\tau)} - y_{(i,1..W,\delta\cdot\tau)})\right|}{(1 - \tau^{(p/q)})}, \tag{3.3}$$

where q is an integer ≥ 1, δ is the tolerance given by the user and τ is a tolerance ratio. In Calvo & al. (2008) values for $q = 5$ and $\tau = 5$ are suggested to produce LEE which are a good compromise between the accuracy of the error estimation and the guarantee that an upper bound for the TLE is provided through the LEE.

3.4 Relative Error1

Since the states can have very different scales it is advisable to scale the TLE for the calculation of an overall error value. An overall value is e.g. useful for the comparison of different schema. An intuitive way to scale the TLE is to divide each error by the respective state, such taking variations in each state along the time into account:

$$RE1 = \frac{1}{W \cdot n} \cdot \sum_{i=1}^{n} \sum_{j=1}^{W} \frac{(y_{numerical,i,j} - y_{analytical,i,j})^2}{y_{analytical,i,j}^2}. \tag{3.4}$$

The problem with this definition is that even small errors can be extremely amplified in case that one of the states, at some time, tends to zero (division by almost zero).

3.5 Relative Error2

Another way of defining an overall error value is to divide the errors of state i by the respective maximum value of the state, i.e.:

$$RE2 = \frac{1}{W \cdot n} \cdot \sum_{i=1}^{n} \sum_{j=1}^{W} \frac{(y_{numerical,i,j} - y_{analytical,i,j})^2}{max(y_{analytical,i,1..W})^2} = \frac{1}{W \cdot n} \cdot \sum_{i=1}^{n} \sum_{j=1}^{W} RE2_{i,j}. \tag{3.5}$$

By doing so one obtains an overall error value in which the different scales of the states are considered. Similar error definitions were also proposed by Shampine (1981).
Note that in case that the analytical solution is not known, the Relative Error2 is calculated using the LEE. In this case, instead of the analytical solution in the term $max(y_{analytical,i,1..W})$, the numerical solution of one of the solvers is chosen and then the calculation of all RE2 values is carried out on this basis, see equation 3.6.

$$RE2 = \frac{1}{W \cdot n} \cdot \sum_{i=1}^{n} \sum_{j=1}^{W} \frac{(LEE_{i,j})^2}{max(y_{numerical,i,1..W})^2} \tag{3.6}$$

3.6 Maximum Relative Error2

Since the Relative Error2 (and the Relative Error1) can somewhat be seen as an average value, since the overall sum of errors is divided by the number of events and states ($W \cdot n$), an additional Error Criteria is defined which is the Maximum Relative Error2. This criteria, as the name reveals, presents the maximum Relative Error2 value that is obtained, irrespectively of the states, i.e.:

$$MaxRelE2 = \max_{i} (RE2_{i,j}) \quad \forall j = 1, ..., W. \tag{3.7}$$

It can be used to analyze whether the averaged error value, i.e. $RE2$, differs significantly from a maximum, locally, obtained error.

3.7 Condition Number

The condition number (κ) studied here is to respect to the Jacobian matrix of the system of ODEs. The condition number provides a measure of the sensitivity of the solution of a system of linear equations to errors in the data (See (Ashino and Vaillancourt, 2009)).

$$\kappa(t) = \left(||A|| \cdot ||A^{-1}|| \right), \tag{3.8}$$

where A is the Jacobian matrix. This condition number was calculated directly with the function *cond* of *MATLAB*, which delivers the norm of order two, i.e., the ratio of the largest singular value of the Jacobian matrix to the smallest (absolute values). Values of condition number near one indicate a well-conditioned matrix.

3.8 Relative Condition Number

The Relative Condition Number, κ_{rel}, is a measure whether a disturbance in the initial values, is amplified or damped by the system. However it can be expected that the numerical integration scheme has an impact on the error propagation. The Relative Condition Number can be calculated along time, i.e.:

$$\kappa_{rel}(t) = \left(\frac{||y(t) - y_{dis}(t)||}{||y(t)||} \right) \cdot \left(\frac{||y(t_0)||}{||y(t_0) - y_{dis}(t_0)||} \right), \tag{3.9}$$

where the index dis refers to a disturbed system. From this definition it follows that the error in the system along time is amplified when $\kappa_{rel} > 1$ and damped when $\kappa_{rel} < 1$.

3.9 Computational Cost

The evaluation criteria mentioned so far concern only one of the two main aspects of comparison criteria, i.e. the accuracy of a numerical integration scheme. The second, which is the computational cost, reflects how efficient a method is, e.g., when comparing two methods, and one of which produces errors a few order of magnitude lower than the other, but at the cost of a much greater computational time.
Maybe, the most accurate manner to define the computational cost of an integration is through the number of function evaluations or function calls, that a scheme made during the integration. The number of function calls is independent of the computer being used, while different computers can take very different duration times. Even the same machine, under different environmental conditions may have distinct duration times. This being said, the duration time for an integration will be still considered, i.e. it is a primary criteria of comparison. This is due to the fact that: (i) except for the *MATLAB ODE suite* solvers, the information regarding the number of function calls is not easily accessible; (ii) the integration times, at least for Test Case A, are so low that, environmental changes can be disregarded.

3.9.1 First Duration

In a sequence of runs, that are performed with the same numerical scheme, the first integration run is observed to have a longer duration time than the following runs which might be due to the initialization. Therefore, the value of the first integration run will be displayed and compared to the average durations.

3.9.2 Average Duration

The average duration is calculated by taking the arithmetic mean of the duration times that are observed for five runs except for the fist run. This averaged duration time gives a reliable estimate for the computational cost.

Chapter 4

Results and Discussion

4.1 Test case A

In order to compare the performance of the numerical, ordinary differential equation solvers (see section 2), a system of linear first order differential equations with known analytical solution will be analyzed in this section. This system is chosen since (i) the analytical solution is known; (ii) it is linear, i.e. it has a constant Jacobian; and (iii) it was applied by Seinfeld and Lapidus (1973) in a similar case, i.e. for the analyses of a set of numerical ODE solvers. This system of ODEs also enables to show the importance of the proper "definition" of the relative error.

4.1.1 Test Set Equations

The system of ODEs is given by

$$y' = \underbrace{\begin{bmatrix} -0.1 & -49.9 & 0 \\ 0 & -50 & 0 \\ 0 & 70 & -120 \end{bmatrix}}_{A} \cdot y, \tag{4.1}$$

where $0 < t \leq 2$ and $y_0 = [2, 1, 2]^T$. For this condition the analytic solution is given as:

$$\begin{aligned} y_1(t) &= e^{-0.1 \cdot t} + e^{-50 \cdot t} \\ y_2(t) &= e^{-50 \cdot t} \\ y_3(t) &= e^{-50 \cdot t} + e^{-120 \cdot t} \end{aligned} \quad , \tag{4.2}$$

its time course is shown in fig. 4.1. This system is a (mildly) stiff problem in the sense that (i) the magnitude of the eigenvalues of the Jacobian differ significantly (The eigenvalues of A are $\lambda_1 = -120$, $\lambda_2 = -50$ and $\lambda_3 = -0.1$; and (ii) with increasing time the solution is firstly governed by a rapidly decaying component (i.e. due to λ_1 or λ_2,) which then becomes insignificant.

Although the system of equations is mildly stiff, this knowledge will be disregarded for the evaluation and both non-stiff and stiff solvers will be used. It can be expected that the non-stiff solvers will have a larger computational cost (excessive computation time/ function calls) for integrating the system. In turn this fact can be used as another way to demonstrate that the system is stiff.

In the following sections the behavior of the solvers (see sections 2.2 and 2.3) will be analyzed for a number of controlled changes to either the numerical solvers or the studied system.

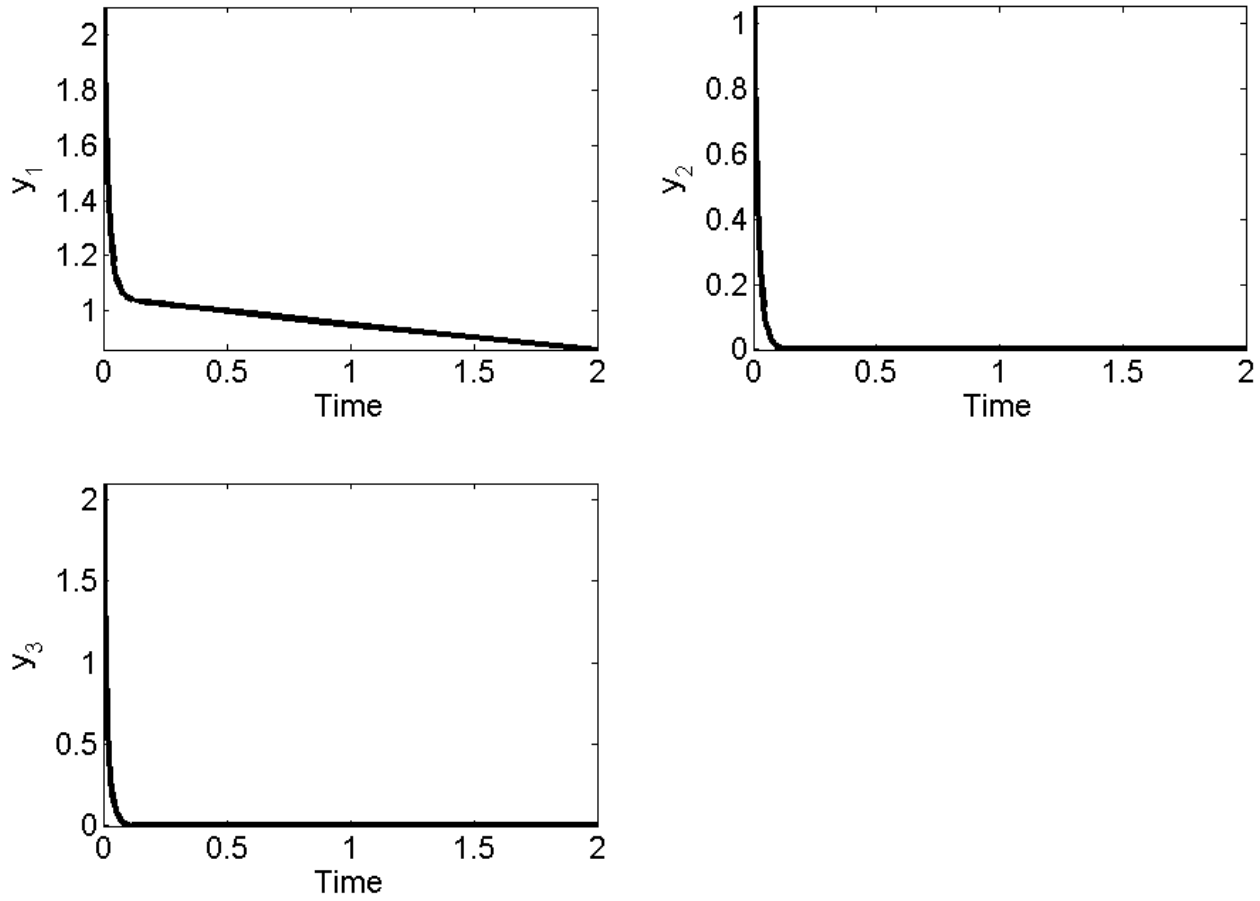

Figure 4.1 Time course of the analytic solution given through equations 4.2.

4.1.2 Analysis of the Adaptive Step Size Solvers - Changing the Relative and/or Absolute Tolerances

Changes to the tolerance values only apply to the solvers with adaptive step size, namely rock4, ode15s, ode23s, ode23t, rkf45, rkv56, vs_pc4, ode87, ode45, ode23 and ode113. The implementation of the solvers rkf45, rkv56 and vs_pc4 allow only for the manipulation of the absolute tolerance.

In case of the MATLAB solvers, two tolerance values can be chosen, i.e. a scalar relative tolerance value, *re*, and a vector of absolute tolerance values, *ae*. Then, at each step the numerical method tries to produce a $y_i(t_n)$ such that:

$$|LEe_i| \leq max(re|(y_i(t_n))|, ae_i) \tag{4.3}$$

The previous inequality is a form of *mixed error control*. In the case that *re* or all values of ae_i are zero, this means that the solver is using a pure absolute control or a pure relative control, respectively. There are difficulties that can arise when either one of the error controls is applied on their own, as highlighted by Gladwell & al. (2003). For instance, one should set a nonzero relative error tolerance value when investigating changes in the pure absolute control Gladwell & al. (2003) (e.g. default MATLAB tolerances).
In the MATLAB help a guideline for the choice of tolerance values is given, i.e.: "At relative high state values, the solver solution convergence is determined by the relative tolerance. As the state values approach zero, convergence

is then controlled by absolute tolerance. The choice of values for the relative tolerance and absolute tolerance varies depending on the problem. The default values are supposed to work for first trials of the simulation; however if you want to optimize the solution, consider that there is a trade-off between speed and accuracy. "

In table 4.1 an overview of the values for the criteria defined in section 3, obtained with the solvers mentioned above are shown for the default tolerances (relative tolerance [$RelTol$] of 1E-3 and absolute tolerance [$AbsTol$] of 1E-6) and the refined tolerances ($RelTol$ of 1E-6 and $AbsTol$ of 1E-8).

Table 4.1 Values for the evaluation criteria over numerical integration methods under default and stringent tolerances.

Criteria / Solver	Rel Tol	Abs Tol	Number events	Relative Error1	Relative Error2	Average Duration	First Duration	Success steps	Failled attempts	Functon evaluation	Partial derivative
Rock4 H.i.	3	6	603	3.60E+062	4.52E-008	1.51E-002	1.19E+000	30	2	245	2
Rock4 H.i.	6	8	603	4.91E+056	3.61E-011	3.01E-002	2.93E-001	87	0	577	4
Rock4	3	6	93	2.56E+072	3.34E-008	1.51E-002	1.19E+000	30	2	245	2
Rock4	6	8	93	1.26E+066	2.42E-014	3.01E-002	2.93E-001	87	0	577	4
ode15s	3	6	603	4.29E+060	3.14E-010	6.03E-002	9.61E-001	72	3	127	1
ode15s	6	8	603	2.97E+063	1.17E-015	8.42E-002	4.83E-001	163	2	335	1
ode23s	3	6	603	2.92E+056	2.77E-009	5.79E-002	4.03E-001	57	0	344	57
ode23s	6	8	603	2.62E+052	3.68E-013	2.28E-001	7.03E-001	391	0	2349	391
ode23t	3	6	603	1.07E+062	9.04E-010	4.04E-002	3.70E-001	94	0	119	1
ode23t	6	8	603	3.51E+056	3.50E-013	1.73E-001	4.28E-001	620	0	658	1
rkf45	3	6	516	6.44E+066	2.81E-018	7.09E-002	9.22E-002	n.i.	n.i.	n.i.	n.i.
rkf45	6	8	1272	1.62E+063	2.03E-022	1.65E-001	1.88E-001	n.i.	n.i.	n.i.	n.i.
rkv56	3	6	348	6.44E+066	3.41E-019	7.33E-002	9.96E-002	n.i.	n.i.	n.i.	n.i.
rkv56	6	8	633	6.23E+061	1.78E-023	1.27E-001	1.52E-001	n.i.	n.i.	n.i.	n.i.
vs_pc4	3	6	1173	1.60E+062	1.57E-018	4.35E-002	7.22E-002	n.i.	n.i.	n.i.	n.i.
vs_pc4	6	8	3141	2.81E+064	7.67E-022	1.03E-001	1.33E-001	n.i.	n.i.	n.i.	n.i.
ode87	3	6	147	3.18E+075	4.08E-009	2.22E-002	7.51E-002	n.i.	n.i.	n.i.	n.i.
ode87	6	8	162	2.75E+071	3.64E-015	2.41E-002	7.94E-002	n.i.	n.i.	n.i.	n.i.
ode45	3	6	603	7.43E+069	2.18E-011	3.21E-002	2.60E-001	81	2	499	n.i.
ode45	6	8	603	2.33E+067	1.69E-018	3.80E-002	2.88E-001	116	2	709	n.i.
ode23	3	6	603	6.80E+071	2.13E-009	4.24E-002	2.47E-001	113	5	355	n.i.
ode23	6	8	603	7.20E+067	7.94E-016	6.52E-002	2.69E-001	374	5	1138	n.i.
ode113	3	6	603	1.56E+071	3.06E-011	5.20E-002	7.32E-002	188	17	394	n.i.
ode113	6	8	603	1.31E+069	5.76E-016	6.20E-002	7.93E-002	228	13	470	n.i.

n.i. = no information

H.i. = Hermite interpolation, see section 4.1.7.

From the inspection of Table 4.1 the following can be observed:

i.) the values of Relative Errors1 ($RE1$) are strikingly greater than the ones of Relative Errors2 which will be discussed further below;

ii.) the stiff solvers (rock4, ode15s, ode23s and ode23t) produce, in general, Relative Errors2 ($RE2$) which are greater than those observed for the non-stiff solvers. This observation is in agreement with the fact that stiff solver due to the underlying solution schema are expected to perform slightly worse than non-stiff solvers, Burden and Faires (2005);

iii.) in contrast to what was expected, the computational costs of the stiff solvers are observed to be greater than those of the non-stiff solvers. This can be explained by the fact that the system is the stiffest at the beginning of the integration, while thereafter the system is relatively non-stiff. The latter phase is significantly longer than the stiff phase;

iv.) non-stiff solvers rkf45, rkv56 and vs_pc4, produce the most accurate results ($RE2 \approx 1E-22$, stringent tolerances) even though they are not the methodologies with the highest order (see section 4.1.7). This of course comes at the expense of higher computational times. However it can be observed that the results obtained for rkf45, rkv56 and vs_pc4 are, even under the less stringent (default) tolerances, still better than those of the other solvers under stringent tolerances while the computational cost is comparable;

v.) Solvers integrating under the stringent tolerances ($RelTol = 1E-6$, $AbsTol = 1E-8$), in general, have an average duration time which is twice as large as when integrating with the default tolerances ($RelTol = 1E-3$, $AbsTol = 1E-6$). This is due to the fact that there are more function calls observed. For the solvers rock4, ode15s and ode113 there are less failed attempts, which signifies that the estimation for a step size was rejected fewer times.

Calculation of the Relative Error

It can be seen in Table 4.1 that the values of relative Errors (i.e. Relative Error1), irrespectively from the solver chosen, are large. When inspecting the local errors (exemplarily shown for the stiff solvers) in Figure 4.2 it can be observed that their magnitude is relatively small. Therefore the values obtained for Relative Error1 are misleading, since the true magnitude of local errors is small. It can be inferred that the definition of Relative Error1, i.e. the division of the local error through the respective exact solution, is not the right approach for this system Shampine (1981). The reason is that the steady state of the solution of at least one of the ODEs tends to zero, which magnifies even the slightest difference between the analytical and numerical solution. Accounting for this effect another relative error was defined (Relative Error2), i.e. the division of the local error through the maximum (absolute) value of the analytic solution for the respective state, as defined in section 3.5. For this case much more reasonable values of relative errors (Relative Error2) can be observe in Table 4.1. These forms of relative error were also investigated by Shampine (1981).

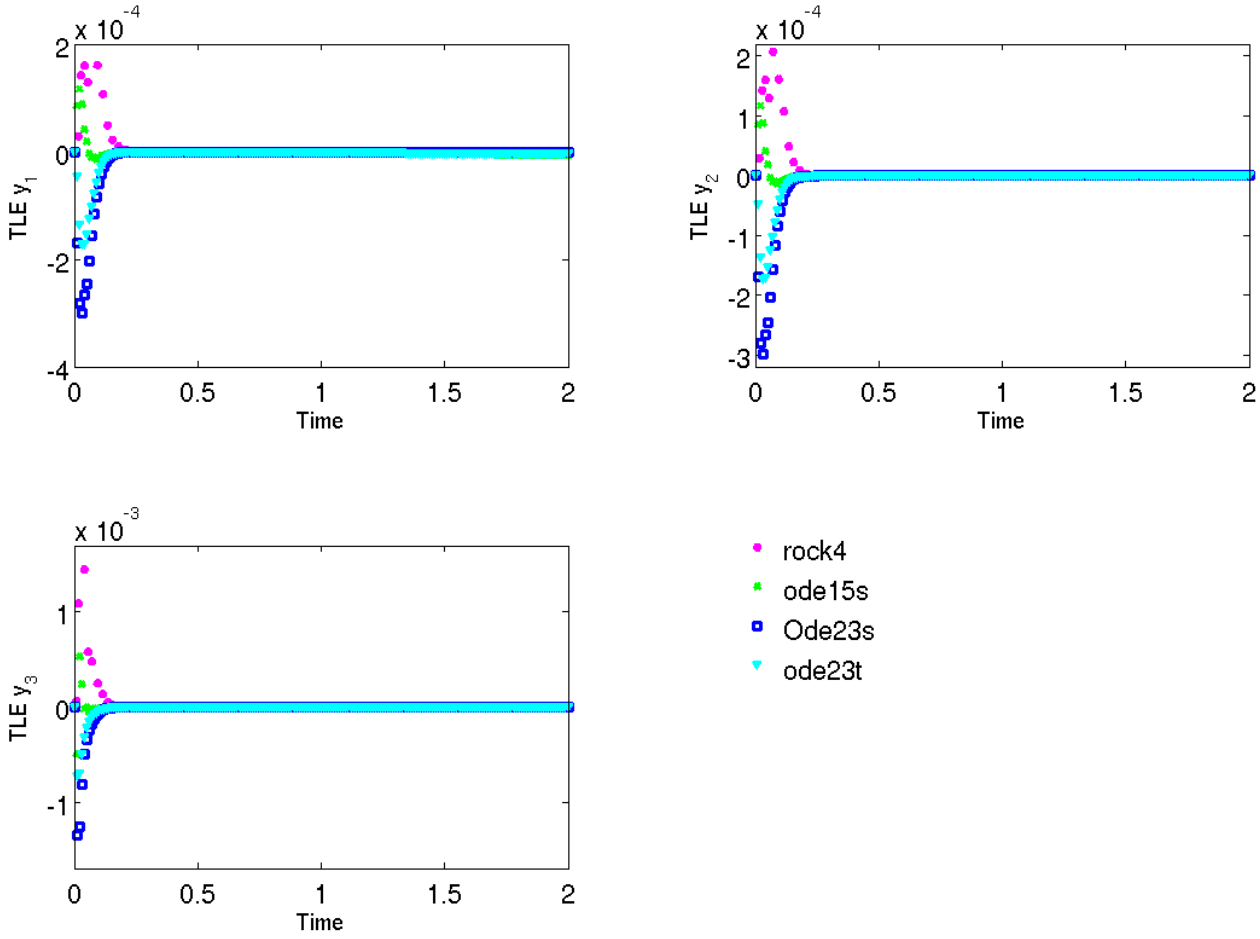

Figure 4.2 Local Errors over time obtained by the solvers rock4, ode15s, ode23 and ode23t (default tolerances).

Observations Regarding the Error Control

As expected, the exercises performed under more stringent absolute and relative tolerances produce lower relative error values than with default tolerance values. An exception occurs when the relative tolerance is set to 1E-3 and the absolute tolerance is varied (1E-6, 1E-8, 1E-10), i.e.: the values of relative errors are the same irrespectively of the different absolute tolerance values, see Table 4.2. This may be caused by the fact that the error control, expression 4.3 is governed by the relative error tolerance, *re*. This assumption is supported by the observations made for plots of True Local Errors (TLE) over time, see Figure 4.3, where, as a title of example, only the solutions obtained for $RelTol = 1E-3$ and $AbsTol = 1E-6$ and $1E-10$ were plotted. Therein it can be seen that irrespective of the state, the biggest TLEs occur at the beginning of the integration. At this stage the TLE values of both solutions ($TLE_{AbstTol} = 1E-6$, green dots and $TLE_{AbstTol} = 1E-10$, red dots) are identical, wherefore (i) one cannot see the red dots and (ii) the difference between both (blue dots) is zero. At a second stage, namely where the difference between the TLEs is nonzero and therefore the local error is controlled by the absolute tolerance, the TLEs which can be observed are much smaller than the ones of the first stage. Thus, the differences in TLE, which are due to the absolute tolerance are in comparison, negligible, i.e. they cannot be observed in Table 4.2. This observationis due to the fact that for relatively great state values, the relative tolerance governs the error control and only when the state values are close to zero (i.e. $re \cdot |(x_i(t_n))| \leq ae_i$), the error is controlled by the absolute tolerance. This means that even though the absolute tolerance does not seem to have an impact on the values of Relative Error2, the TLE is constrained at some point by the absolute tolerance. This is important since the values of the state are relatively small at the point (which is for the first time constrained by the absolute tolerance) and the solution, if unconstrained, would be significantly worse. This findings are in agreement with Gladwell & al. (2003).

Table 4.2 Relative Errors for ode45 solver at fixed relative tolerance (1E-3).

Absolute tolerances	Maximum Relative Error2	Relative Error2
Abs Tol 1 (1e-6)	1.13e-004	2.1754e-011
Abs Tol 2 (1e-8)	1.13e-004	2.1742e-011
Abs Tol 3 (1e-10)	1.13e-004	2.1742e-011

Error Propagation

Figure 4.4 displays the TLEs with relatively stringent tolerances (Relative Tolerance of 1E-6 and Absolute Tolerance of 1E-8). Therein, in the cases of solvers ode23t and ode23s, error propagation is patent for equation y_1, which will be subject to analysis in section 4.1.5. This behavior however cannot be observed in Figure 4.2 (default tolerances) which is due to the scaling of the axes that is relatively rough. Further, only for the solvers ode23t and ode23s very high local errors can be observed for equations y_2 and y_3 in Figure 4.4, which might be due to elevated gradients in the beginning of the integration interval. These findings are in accordance to the *MATLAB help* and Gladwell & al. (2003).

Local Error Estimation based on the Tolerance Proportionality

The local error estimation which is based on the tolerance proportionality equation 3.3, (see section 3.2) is supposed to give an upper bound for the magnitude of the true local error. This estimation can be of great help in the cases where an exact solution cannot be found (e.g. see section 4.2.2). An heuristic search was performed to encounter the ideal values of τ and q, which should balance (i) the minimization of the difference between TLE

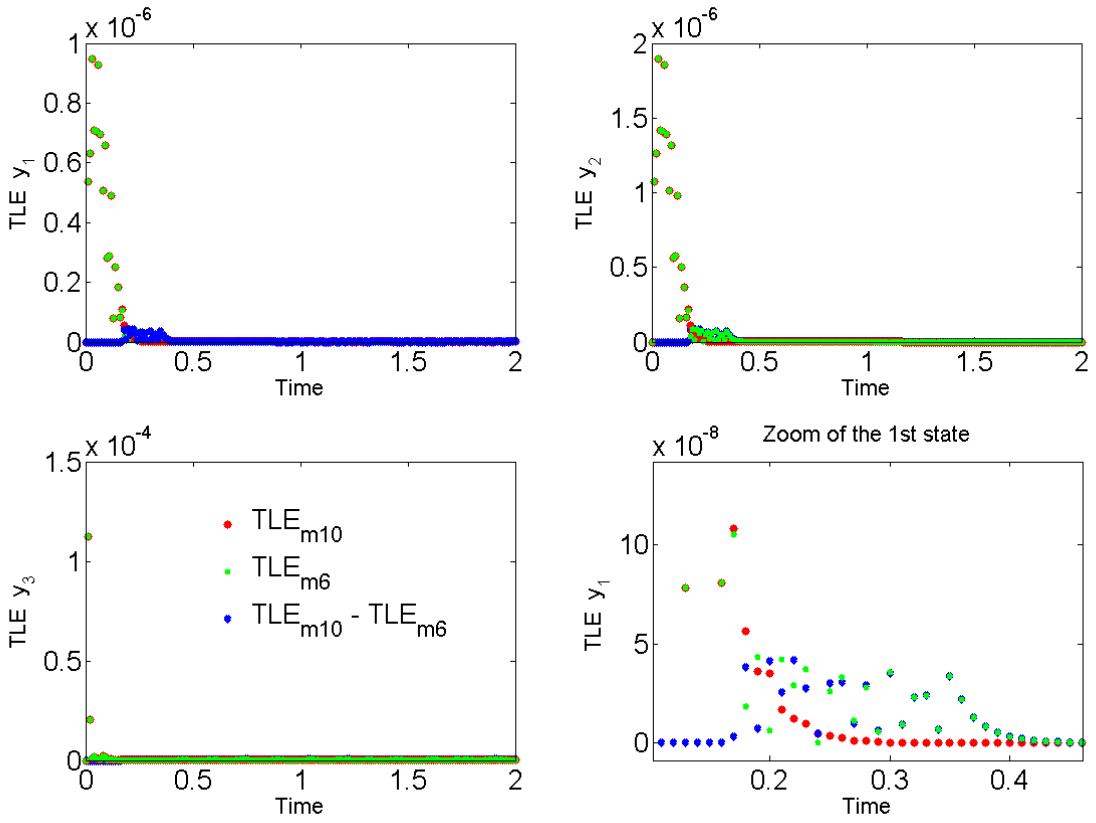

Figure 4.3 True Local Errors (TLEs) over time obtained with solver ode45. Relative tolerance fixed at $1E-3$ and Absolute tolerance values of $1E-6$ (TL_{m6}) and $1E-10$ (TL_{m10}).

and Local Error Estimate (LEE); and (ii) the guarantee that even in the worst case the LEE is an upper bound for the TLE.

Table 4.3 LEEs obtained with equation 3.3 for the relative tolerance: 1E-8; and the absolute tolerances: 1E-6 and 1E-10.

τ	q	Method LEE	Number of Events	Maximum Relative Error2	Relative Error2
10000	4	ode15s	603	1.1818e-007	3.5150e-016
10000	4	ode23s	603	2.1007e-007	1.2475e-015
10000	4	ode23t	603	1.7136e-007	8.5578e-016
10000	4	ode45	603	9.9695e-011	3.7917e-022
10000	4	ode23	603	1.1251e-007	2.3043e-016
10000	4	ode113	603	1.0901e-006	1.6662e-014
10000	4	rkv56 t	603	6.3126e-016	5.9212e-033
10000	5	ode15s	603	1.0290e-007	2.6652e-016
10000	5	ode23s	603	1.6575e-007	7.7668e-016
10000	5	ode23t	603	1.3521e-007	5.3278e-016
10000	5	ode45	603	5.6058e-011	1.1988e-022
10000	5	ode23	603	8.8775e-008	1.4346e-016
10000	5	ode113	603	9.4921e-007	1.2634e-014
10000	5	rkv56 t	603	3.1637e-016	1.4873e-033

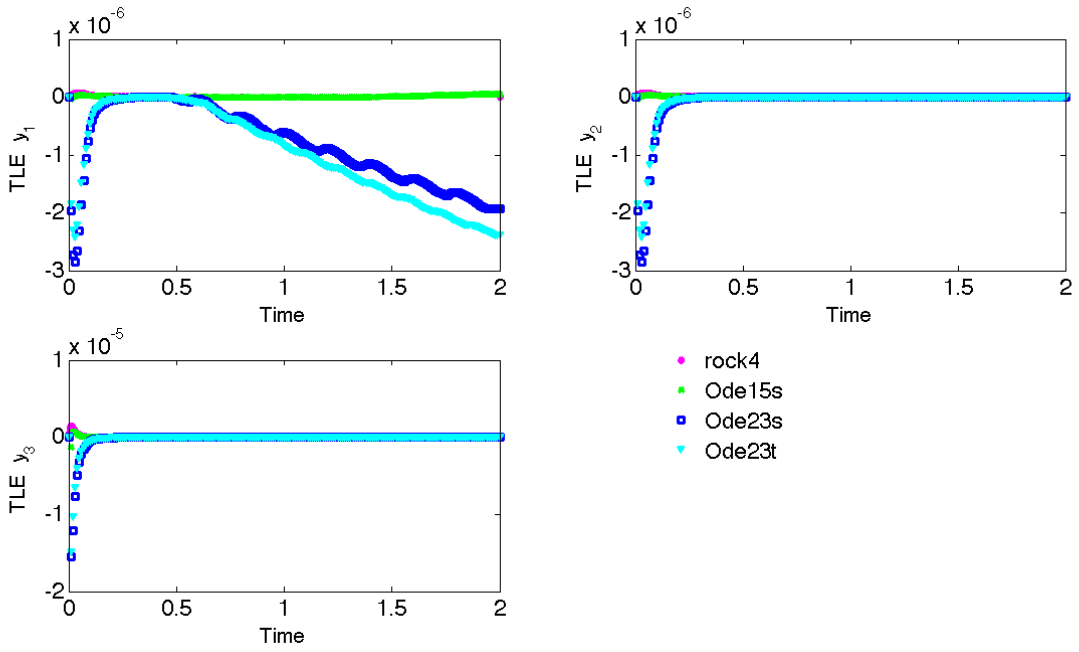

Figure 4.4 True Local Error of solvers rock4, ode15s, ode23s and ode23t (Relative tolerance: 1E-6; Absolute tolerance: 1E-8).

Table 4.4 TLE for the relative tolerance: 1E-8 and the absolute tolerance: 1E-6.

τ	q	Method TLE	Number of Events	Maximum Relative Error2	Relative Error2
1000	4	ode15s	603	5.5923e-007	7.6295e-015
1000	4	ode23s	603	6.7108e-006	1.2211e-012
1000	4	ode23t	603	5.4254e-006	8.4017e-013
1000	4	ode45	603	5.6052e-007	1.1985e-014
1000	4	ode23	603	3.4455e-006	2.1613e-013
1000	4	ode113	603	1.2102e-009	9.3796e-021
1000	4	rkv56 t	603	2.8255e-015	5.7582e-031
1000	5	ode15s	603	5.5923e-007	7.6295e-015
1000	5	ode23s	603	6.7108e-006	1.2211e-012
1000	5	ode23t	603	5.4254e-006	8.4017e-013
1000	5	ode45	603	5.6052e-007	1.1985e-014
1000	5	ode23	603	3.4455e-006	2.1613e-013
1000	5	ode113	603	1.2102e-009	9.3796e-021
1000	5	rkv56 t	603	2.8255e-015	5.7582e-031

Here values of $\tau = 5, 10, 1000$ and $q = 3, 4, 5$ were studied. The values studied in the original article by Calvo & al. (2008) are $\tau = 1.5, 2, 5, 10$ and $q = 5$, who also recommended to use $\tau = 5$ and $q = 5$. However in the studied cases the LEE did, many times, not provide an upper bound for the TLE, which e.g. can be observed for the ode45 solver when comparing the values given in Table 4.3 to the ones given in Table 4.4. Even with a relatively low value of $\tau = 5$ (see Table 4.5) which results in more crude values of LEE, the ode45 solver still produces values of Relative Error2 that are smaller than the respective Relative Error2 values in case of the TLE (see Table 4.6). Since one usually wants to obtain an upper bound of the expected magnitude of the error, the definition by Calvo & al. (2008) at least in the given case, cannot be used to its full extent. This also does not support the trust in the LEE definition.

Table 4.5 LEE obtained with equation 3.3 for the relative tolerances: 1E-8 and 5E-8; and the absolute tolerance: 1E-10.

τ	q	Method LEE	Number of Events	Maximum Relative Error2	Relative Error2
5	5	ode15s	603	8.7002e-008	4.7722e-016
5	5	ode23s	603	7.6440e-007	4.2438e-015
5	5	ode23t	603	5.7620e-007	3.0083e-015
5	5	ode45	603	4.5075e-010	7.4302e-022
5	5	ode23	603	1.9581e-008	1.4436e-018
5	5	ode113	603	3.7024e-008	4.4342e-018

Table 4.6 TLE for the relative tolerance: 5E-8 and the absolute tolerances: 1E-10.

τ	q	Method TLE	Number of Events	Maximum Relative Error2	Relative Error2
5	5	ode15s	603	3.7381e-008	1.1375e-016
5	5	ode23s	603	1.0482e-006	8.1691e-015
5	5	ode23t	603	7.9513e-007	5.8934e-015
5	5	ode45	603	1.8429e-009	9.5377e-021
5	5	ode23	603	2.2079e-008	1.8417e-018
5	5	ode113	603	1.2102e-009	9.3796e-021

4.1.3 Analysis of the Fixed Step Size Solvers - Changing the Step Size

The solvers with fixed step sizes are: euler_modified, euler_forward, euler_backward, heun, opt_rk2, rk4, ab2 and adams_pc4. In order to evaluate these solvers various experiments are performed in which the step size is varied. As a first estimate for the fixed step size, the initial step size given in literature Seinfeld and Lapidus (1973), 7E-3, was used. Additionally, two other step sizes were studied, namely ten times greater and ten times lower, so in total three numerical experiments were performed. The results are comprised in Tables 4.7 through 4.9.

Table 4.7 Values of evaluation criteria over numerical integration methods with fixed step size of 7E-3.

Method	Number of Events	Relative Error2	Average Duration	First Duration
euler_modified	861	7.27E-006	2.72E-002	7.32E-002
euler_forward	861	1.00E-004	9.70E-003	1.62E-002
euler_backward	861	9.43E-004	3.10E-002	4.03E-002
heun	861	7.27E-006	1.29E-002	4.34E-002
opt_rk2	861	7.27E-006	1.24E-002	2.12E-002
rk4	861	6.25E-009	2.44E-002	3.80E-002
ab2	861	2.31E-005	7.59E-003	1.77E-002
adams_pc4	861	8.60E-008	1.70E-002	3.74E-002

It can in general be observed from the inspection of the tables 4.7, 4.8 and 4.9 that:

i. When using a step size of 7E-2, all fixed step size solvers produce extremely high Relative Error2 values (Table 4.8). The solvers, forced to used such a large step size, were not able to produce accurate solutions which is in agreement with Burden and Faires (2005);

ii. When using relatively small step sizes, 7E-4, the fixed step solvers need 8573 steps, and therefore deliver smaller Relative Error2 values than with a step size of 7E-3 (Tables 4.9 and 4.7);

iii. Euler_modified and heun solvers produces the same Relative Error2 values (2.45E-10), as expected (see section 2.2.1). The heun solver in comparison has a lower computation time. This is due to the fact that its implementation is more efficient, i.e. one less function evaluation is made per step;

iv. It could also be observed that in the case of the opt_rk2 the Relative Error2 values were identical to the respective values from the euler_modified and heun solvers and the duration times were even lower than for euler_modified. This is unexpected since the same number of function evaluations are, in principal, made;

v. In case that the step size is adequately small it can be seen that the error values obey to the order of accuracy of the solvers.

Table 4.8 Values of evaluation criteria over numerical integration methods with fixed step size of 7E-2.

Method	Number of Events	Relative Error2	Average Duration	First Duration
euler_modified	90	2.72E+080	2.88E-003	9.51E-003
euler_forward	90	2.76E+047	1.20E-003	5.02E-003
euler_backward	90	1.21E+101	2.11E-003	6.44E-003
heun	90	2.72E+080	2.26E-003	9.52E-003
opt_rk2	90	2.72E+080	1.75E-003	7.19E-003
rk4	90	4.49E+119	3.00E-003	1.17E-002
ab2	90	1.68E+060	1.29E-003	9.10E-003
adams_pc4	90	9.42E+097	2.82E-003	2.06E-002

Table 4.9 Values of evaluation criteria over numerical integration methods with fixed step size of 7E-4.

Method	Number of Events	Relative Error2	Average Duration	First Duration
euler_modified	8574	2.45E-010	2.30E-001	2.66E-001
euler_forward	8574	7.06E-007	1.04E-001	1.21E-001
euler_backward	8574	8.21E-007	1.78E-001	1.93E-001
heun	8574	2.45E-010	1.21E-001	1.43E-001
opt_rk2	8574	2.45E-010	1.21E-001	1.42E-001
rk4	8574	1.95E-017	2.58E-001	2.73E-001
ab2	8574	1.38E-009	7.04E-002	9.38E-002
adams_pc4	8574	2.19E-016	1.62E-001	1.95E-001

Local Error Estimate based on Step Halving

For the present test case there is an analytical solution, so one can accurately calculate a set of errors associated with the solutions from the solvers. The local error estimates (LEE), see section 3.2 were calculated in order to compare the estimations to the true local errors. Based on the LEE and the TLE, Relative Error2 and Maximum Relative Error2 were calculated, which are comprised in Table 4.10. Therein it can be observed that the LEE always provides an upper bound for the TLE. However the quality of the LEE is rather poor since, in general, not even the magnitude of the TLE is precisely estimated.
By visual inspection of the LEEs, Figure 4.6, it is noticeable that the euler_backward and euler_forward methodologies (order one) produce relative high values of LEE over time, which is in agreement with the fact that these methodologies have a lower order, namely order one. When comparing the plots of LEEs to the ones of TLEs, Figure 4.5, it becomes obvious that the LEE only serves as an upper bound.

Table 4.10 TLE and LEE for fixed step size solvers (step size: 7E-3).

Method	Error Type	Number of Events	Maximum Relative Error2	Relative Error2
euler_modified	LEE	861	1.6987e+00	1.1159e-02
euler_modified	TLE	861	4.37E-002	7.27E-006
euler_forward	LEE	861	1.4796e+01	6.4860e-01
euler_forward	TLE	861	1.63E-001	1.00E-004
euler_backward	LEE	861	3.7808e+01	1.2588e+01
euler_backward	TLE	861	3.39E-001	9.43E-004
heun	LEE	861	1.6987e+00	1.1159e-02
heun	TLE	861	4.37E-002	7.27E-006
opt_rk2	LEE	861	1.6987e+00	1.1159e-02
opt_rk2	TLE	861	4.37E-002	7.27E-006
rk4	LEE	861	1.3955e-02	5.1462e-07
rk4	TLE	861	1.54E-003	6.25E-009
ab2	LEE	861	2.1018e+00	1.6180e-02
ab2	TLE	861	5.82E-002	1.29E-005
adams_pc4	LEE	861	2.6912e-01	1.8721e-04
adams_pc4	TLE	861	5.77E-003	8.60E-008

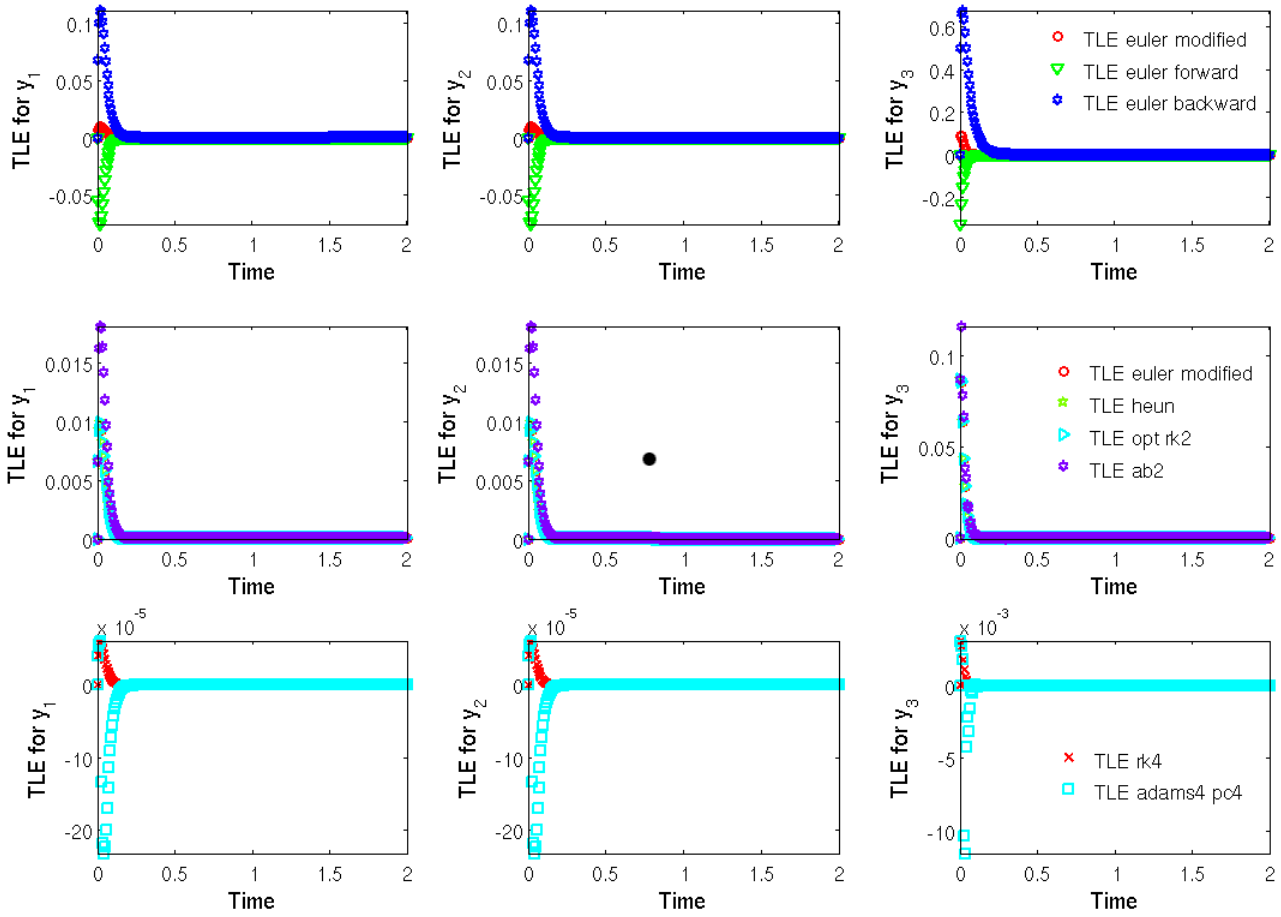

Figure 4.5 TLE for fixed step size solvers (step size 7E-3).

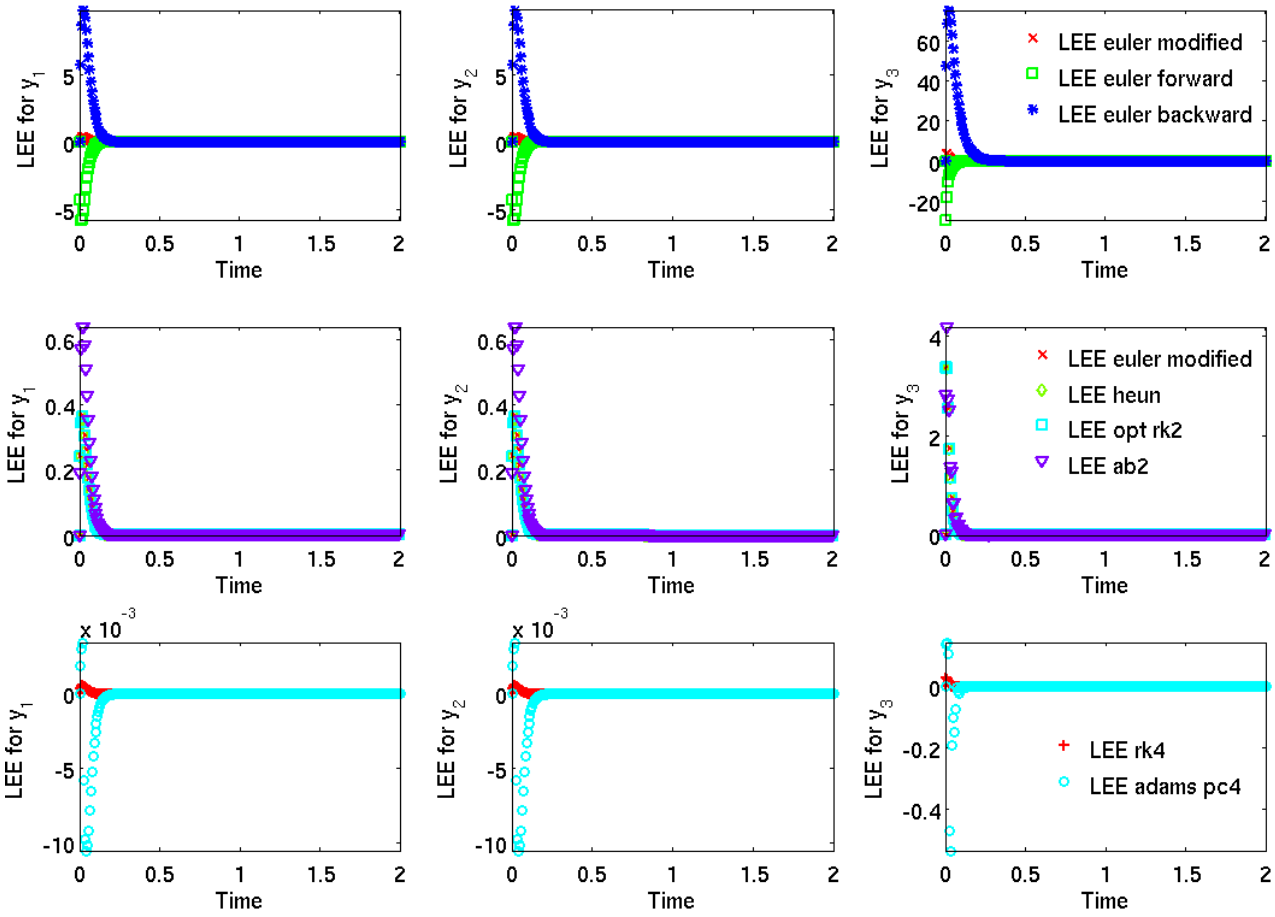

Figure 4.6 Local Error Estimates for fixed step size solvers (step size 7E-3).

4.1.4 Changing the Initial Step Size

The changes in the initial step size only apply to the solvers with adaptive step size control. The implementation of the solvers rkf45, rkv56 and vs_pc4 do not allow the manipulation of the initial step size estimation. As a first estimate for the initial step size, the value given in literature Seinfeld and Lapidus (1973), 7E-3, was used. Additionally, two other step size values were used, namely 100 times smaller and 100 times bigger. The results are comprised in Tables 4.12 through 4.11.

Table 4.11 Values of evaluation criteria over numerical integration methods (default tolerances), with given initial step size of 7E-1.

Method	Number of Events	Maximum Relative Error2	Relative Error2	Average Duration	First Duration
Rock4	60	5.64e-003	7.31e-007	8.6706e-002	1.8454e+000
ode15s	603	9.31e-004	4.50e-009	3.4913e-001	3.0527e+000
ode23s	603	8.96e-004	3.52e-009	2.7934e-001	2.5945e+000
ode23t	603	3.33e-004	7.97e-010	3.9221e-001	2.3978e+000
ode87	150	2.34e-004	4.6522e-009	1.2551e-001	3.9691e-001
ode45	603	1.05e-005	3.8531e-013	1.0474e-001	1.8309e+000
ode23	603	6.68e-004	2.1068e-009	6.0954e-002	5.7287e-001
ode113	603	1.14e-004	7.0926e-011	1.0690e-001	1.1316e-001

Table 4.12 Values of evaluation criteria over numerical integration methods (default tolerances), with given initial step size of 7E-3.

Method	Number of Events	Maximum Relative Error2	Relative Error2	Average Duration	First Duration
Rock4	63	1.99e-003	9.55e-008	1.7196e-002	2.4603e-001
ode15s	603	5.33e-004	8.02e-010	5.4925e-002	4.3683e-001
ode23s	603	9.51e-004	3.97e-009	4.8051e-002	3.7840e-001
ode23t	603	3.46e-004	8.64e-010	4.9602e-002	2.9857e-001
ode87	150	2.52e-004	4.5327e-009	2.3504e-002	6.9571e-002
ode45	603	7.58e-005	1.3254e-011	2.8040e-002	2.6030e-001
ode23	603	5.88e-004	1.7113e-009	2.9684e-002	2.4694e-001
ode113	603	1.14e-004	7.0926e-011	4.9884e-002	5.3031e-002

Table 4.13 Values of evaluation criteria over numerical integration methods (default tolerances), with given initial step size of 7E-5.

Method	Number of Events	Maximum Relative Error2	Relative Error2	Average Duration	First Duration
Rock4	78	1.75e-003	6.39e-008	1.2682e-001	1.7103e+000
ode15s	603	4.45e-004	6.56e-010	3.4297e-001	3.3228e+000
ode23s	603	7.57e-004	2.79e-009	2.8133e-001	2.6363e+000
ode23t	603	6.02e-004	1.61e-009	3.6503e-001	2.4942e+000
ode87	153	2.65e-004	4.0701e-009	4.6972e-002	1.6104e-001
ode45	603	1.22e-004	2.5361e-011	6.4440e-002	6.2094e-001
ode23	603	6.18e-004	1.9227e-009	6.1053e-002	5.7252e-001
ode113	603	2.43e-004	1.6917e-010	9.9408e-002	1.1770e-001

From the observations for Tables 4.12 through 4.11 no general trend can be inferred, i.e. the user specified initial step size does not provoke similar behaviors in the errors of the different solvers. These findings are in accordance with Krogh (1973): "Note that there is very little penalty for selecting too large (or too small) an initial step size, which is just as well since many users would be appalled if they knew how small a step is used initially."However, in the case of the rock4 and ode87 it can be seen that with decreasing step size the number of

events increases and the accuracy of the solution (Relative Error2 values) improves. Increasing accuracy of the solution (along with decreasing initial step sizes) can also be observed for some of the other solvers.

4.1.5 Disturbing the Initial Values

Through the disturbance of the initial values it is intended to investigate (i) to which parts the solvers contribute to the error propagation and (ii) how the errors introduced by the disturbance compare to the errors introduced by the numerical integration. It is clear that the latter depends on the system of equations chosen.
Disturbances in the initial values are frequent since they are usually not known theoretically wherefore experimentally measured values are considered, which come along with experimental inaccuracy. This analysis is e.g. of importance for Parameter Estimation, since (i) the initial values are usually taken from measurements and (ii) during the parameter identification from experimental data, numerical integration routines frequently find application.

The test-set system has a high condition number (2039.6, see equation 3.7), which gives reason to the expectation that the ODEs system is sensitive to disturbances (Conte and de Boor, 1981).

A perturbation of 5% was performed to the initial values, $y(t0) = [2, 1, 2]$; i.e.: $y1(t0) = [2.1, 1.05, 2.1]$ and $y2(t0) = [1.9, 0.95, 1.9]$, e.g. see Figure 4.7. Therein it can be seen that for y_1 error propagation can be observed for the integration interval. This behavior is not as clearly observed for y_2 and y_3, which is due to the fact that the states reach their equilibrium relatively fast. As a title of example, only the stiff solvers were tested, since error propagation was prior observed for those. Default tolerances were used in all cases.

In a first part, an analysis was made in which the solution of every stiff solver was compared to a “new ”analytical solution. This “new ”analytical solution was obtained with the MATLAB *dsolve* function and the respective disturbed initial values. The purpose of this exercise is to identify the magnitude of the error that is due to the numerical integration. Results are comprised in Table 4.14 in form of Maximum Relative Error2 and the Relative Error2 over the stiff solvers. Therein it can be seen that the values obtained for a positive or negative disturbance of 5% seem to be identical. This observation was already expected from the time courses presented in Figure 4.7, but it is clearly enforced when inspecting the results presented in Figure 4.8, wherein it can be seen that a disturbance of the initial values by $\mp 5\%$ seems to produce symmetrical local errors. Even though one can observe that the disturbances in the initial values propagate evenly irrespective of the direction of disturbance, this error propagation is a property of the ODEs system and thus, in that form, does not reveal any information about the numerical integration schema applied. Therefore, in what follows, only positive disturbances are further investigated.

Table 4.14 Relative Error2 values of stiff solvers calculated through the difference between the numerical solution and the analytical solution where both solutions are subject to the disturbed initial values.

Method	Disturbance of y_0	Maximum Relative Error2	Relative Error2
Rock4	+ 5%	1.39e-003	3.34e-008
Rock4	− 5%	1.39e-003	3.34e-008
ode15s	+ 5%	2.66e-004	3.14e-010
ode15s	− 5%	2.66e-004	3.14e-010
ode23s	+ 5%	6.78e-004	2.77e-009
ode23s	− 5%	6.78e-004	2.77e-009
ode23t	+ 5%	3.62e-004	9.04e-010
ode23t	− 5%	3.62e-004	9.04e-010

In a second part, an analysis was made in which the solution of every stiff solver is compared to (i) the original analytical solution (applying the undisturbed initial values); and (ii) the “new ”analytical solution referred to as

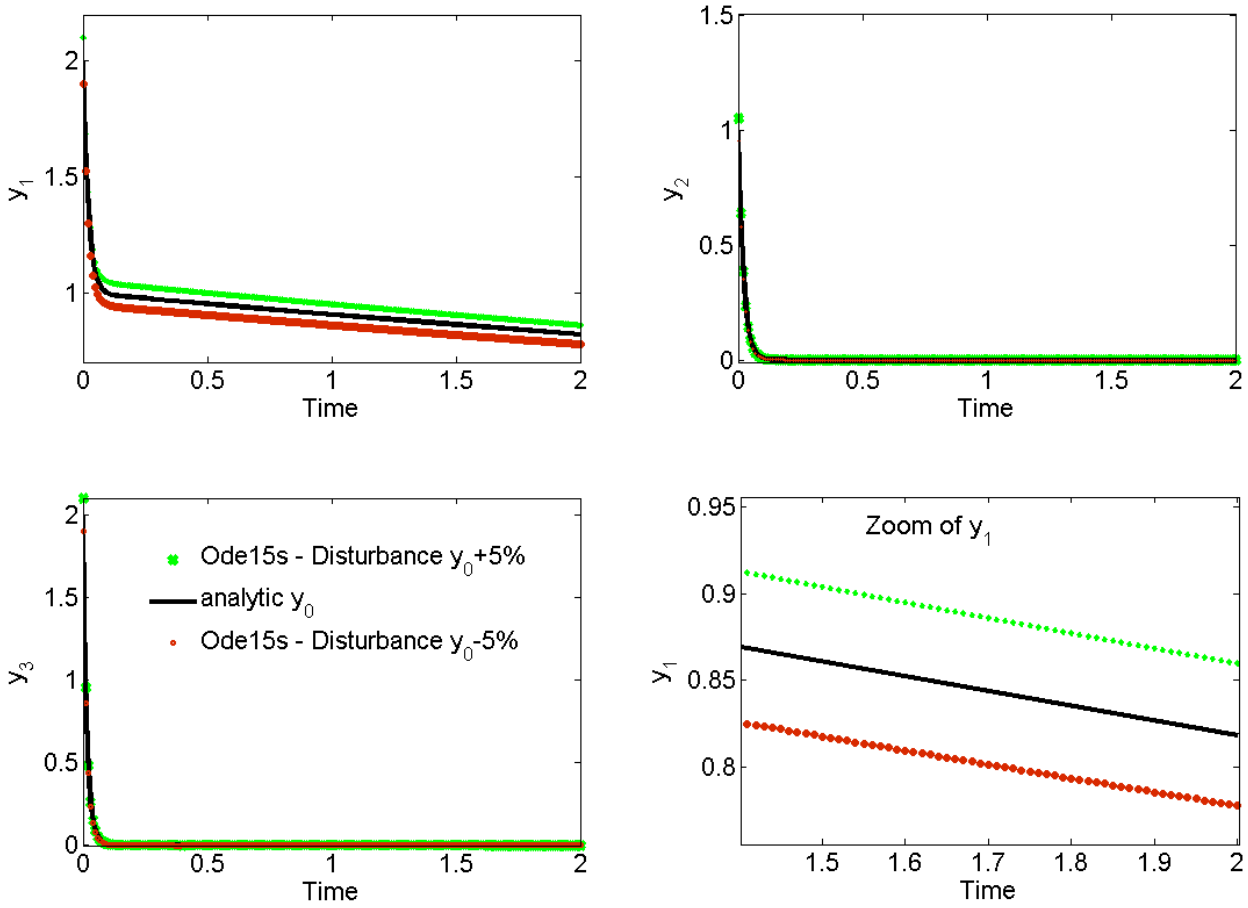

Figure 4.7 The analytical solution with original initial values and the numerical solutions obtained through the ode15s with the disturbed initial values, i.e. $y_{+5\%}(t0) = [2.1, 1.05, 2.1]$ and $y_{-5\%}(t0) = [1.9, 0.95, 1.9]$.

disturbed analytical solution (which is only presented for positive disturbances, see above). Results are shown in Table 4.15, in the form of Maximum Relative Error2 and the Relative Error2 over the stiff solvers, and the original or disturbed analytical solutions. This analysis enables the quantification of the error caused by the disturbance in the initial value when compared with the error caused by the numerical integration alone.

Table 4.15 Relative Errors2 of stiff solvers calculated through the difference between the numerical solution subject to the disturbed initial values and the analytical solution subject to the original initial values.

Method	Initial values	Maximum Relative Error2	Relative Error2
Rock4	disturbed	1.39e-003	3.34e-008
Rock4	original	5.00e-002	1.08e-003
ode15s	disturbed	2.66e-004	3.14e-010
ode15s	original	5.00e-002	1.91e-004
ode23s	disturbed	6.78e-004	2.77e-009
ode23s	original	5.00e-002	1.90e-004
ode23t	disturbed	3.62e-004	9.04e-010
ode23t	original	5.00e-002	1.90e-004

It can be observed (in Table 4.15) that for all four solvers the obtained maximum relative errors are the same, i.e. 5.00 E-002, which is exactly the initial disturbance of 5%. In case of the Relative Error2 values it can be seen that those errors which were calculated with the original initial values are several orders of magnitude greater than

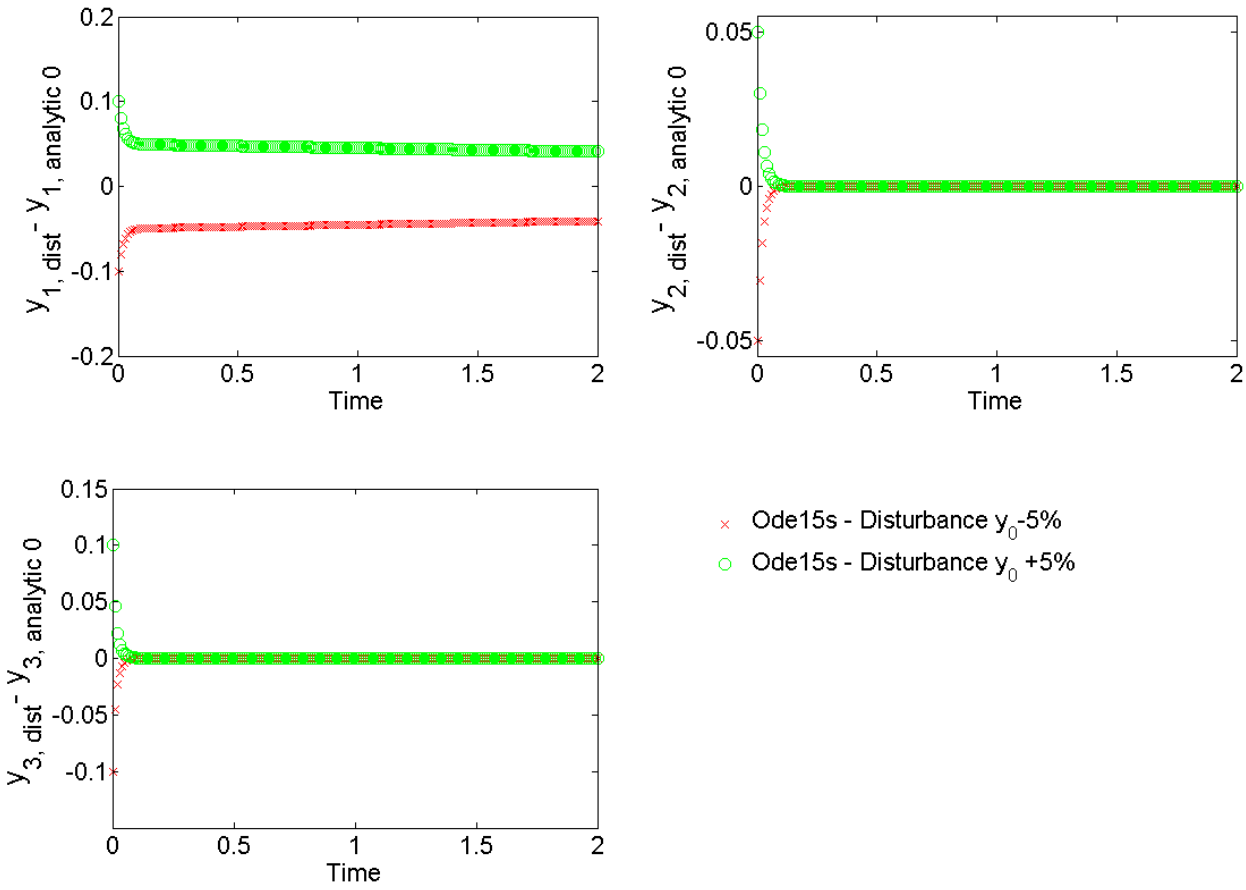

Figure 4.8 Local errors for $y1(t0)$. Local errors for $y2(t0)$. Ode15s.

those obtained with the disturbed initial values. This means that, in the given case, the errors caused through the disturbances in the initial values have a much higher magnitude than those errors caused by the numerical integration schema. However, this is as before due to the property of the ODEs system. In order to estimate how much the numerical integration schema contributes to the error propagation, the following is applied:

$$EPA = TLE_{5\%} - TLE_0 = (y_{i,num,5\%} - y_{i,ana,5\%}) - (y_{i,num,0} - y_{i,ana,0}), \tag{4.4}$$

where the 0 in the index refers to the original and the 5% refers to the disturbed initial values, and the *ana* and *num* refer to the analytical and the numerical solution, respectively. In Figure 4.9 the results obtained with equation 4.4 are plotted over time. Therein it can be seen that:

i.) only very fiew points are obtained for the solver rock4. The shown points differ significantly (about two orders of magnitude) from those of the other solvers' values, which is also confirmed by the higher Relative Error2 values shown in Table 4.14;

ii.) for the EPA of y_1 the solvers ode23s and ode23t show a slightly oscillatory behavior, which is similar to the one observed in Figure 4.4. Such a behavior might be interpreted as a mild instability. In the case of the ode23t the difference of TLEs in general seems to decrease.

iii.) in case of the solver ode15s the value of the EPA of y_1 also seems to decrease. Further it can be seen for y_2

and y_3 that the values obtained for the ode15s in comparison to the solvers ode23s and ode23t show slight bumps.

All in all the results indicate that the numerical integration schema contribute additionally to the error propagation, but with a much lower order.

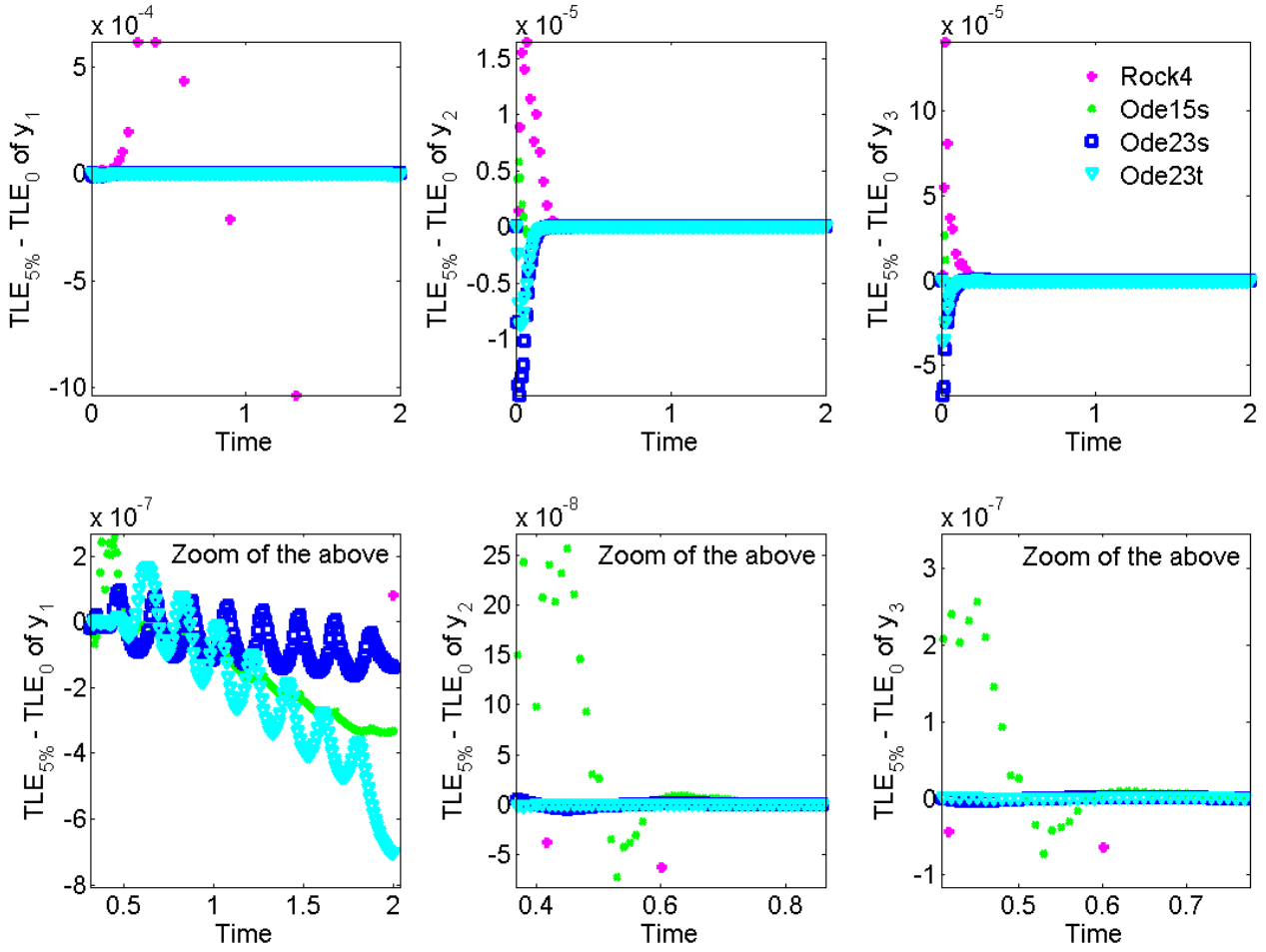

Figure 4.9 Investigation on the Error Propagation of the numerical integration routines subject to disturbances in the initial values.

Relative Condition (κ_{rel})

Another way to analyze the error propagation of a system along time is given through the relative condition number (κ_{rel}). The relative condition number (κ_{rel}) provides a measure of the damping or amplification of a disturbance of the initial condition along the integrating time. A plot of the κ_{rel} over the integrating interval can be seen in Figure 4.10. (Note that the y-axis in this case unfortunately shows only the values of one since the digits varying from one are much smaller, wherefore a black continuous line is given at the exact value, which in this case is superposed by the analytical solution.) In Figure 4.10 it can be seen that in case of the analytical solution the disturbance in the initial values maintains the same magnitude along the time. As observed before in Figures 4.9 and 4.4), the ode23s and ode23t solvers show oscillating behavior. The relative condition number of the ode23t seems to decrease, which is in agreement to the observation made before in Figure 4.9. Also the relative condition number of the ode15s is lower than one, which means that the error introduced by the disturbance in the initial value is damped but on the other hand, the error due to the numerical integration scheme increases. What is

very interesting to note is that even so the question whether error propagation occurs or not is dominated by the system of ODEs, the choice of the numerical integration schema used for the solution does have an influence on the propagation. The magnitude of this contribution might be case dependent.

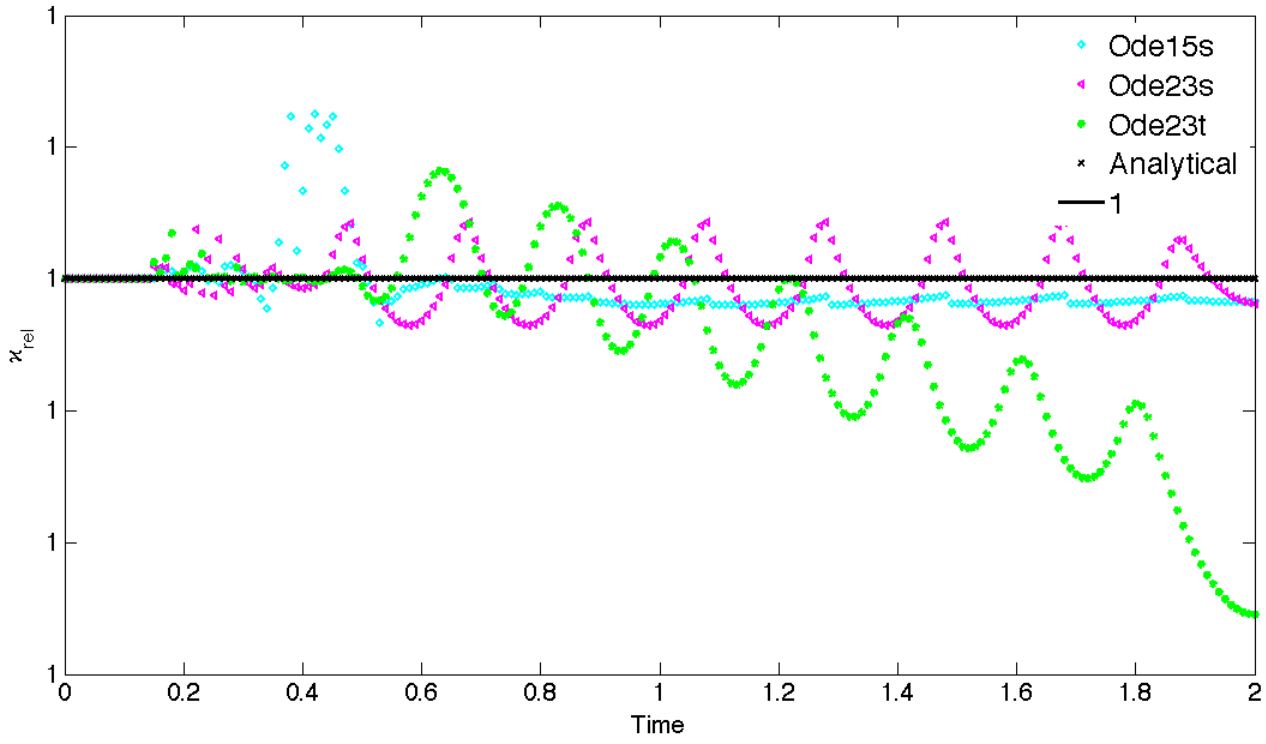

Figure 4.10 κ_{rel} for the *MATLAB ODE suite* stiff solvers.

4.1.6 Structural Knowledge Incorporation – Constant Jacobian

The *MATLAB ODE* stiff solvers (are the only solvers studied here that) allow for the incorporation of structural knowledge about the ODE system, in this particular case that the Jacobian is constant. This information does not impact on the quality of the results but on the computation time required for the integration. This is due to the fact that at each time step all implicit methodologies have to solve a set of linear or non-linear equations, requiring the numerical calculation of the Jacobian of $f(x, t)$ (see equation 2.1). Thus, in the case that the Jacobian is constant, it is worthwhile to incorporate this information a priori into the stiff solver, wherefore the numerical calculation of the Jacobian at each time step is avoided.

In Table 4.16 the statistics obtained by the stiff solvers (under default tolerances), are presented. Only one of the solvers, namely ode23s, benefits from the information about the Jacobian that is provided. In this particular case there were 176 function evaluations and 1 partial derivative versus 344 function evaluations and 57 partial derivatives in the case that no information about the Jacobian was provided. Therein the average duration time also was lower when the Jocobian was given.

The apparent lack of efficiency of the other solvers is justified by the fact that ode15s and ode23t are able to save the Jacobians and that those solvers only recalculate the Jacobians when they "believe" that it is necessary for the efficient evaluation of the implicit formulas, (Gladwell & al., 2003).

Table 4.16 Statistic information for the *MATLAB* suite stiff solvers with and without Structural knowledge incorporation, under default tolerances.

Solver	Given Jacobian	Successful steps	Failed attempts	Function evaluations	Partial derivatives	Average duration
ode15s	No	72	3	127	1	6.03E-2
ode15s	Yes	72	3	127	1	5.30E-2
ode23s	No	57	0	344	57	5.39E-2
ode23s	Yes	57	0	176	1	2.67E-2
ode23t	No	94	0	119	1	4.04E-2
ode23t	Yes	94	0	119	1	4.13E-2

4.1.7 General Observations

Additional analysis regarding the best performing solver – rkv56

Solver rkv56 has produced the lowest relative errors within a broad range of tolerances (e.g. see Table 4.1). In order to ensure that this efficiency is not influenced by the fact that the number of points given by rkv56 (which was most times lower than those of the other solvers), an extra exercise was performed. At first the TLE were graphically analyzed, see Figure 4.11. Therein it can be seen that the TLE obtained for the rkv56 is either lower or at most equal to the TLE obtained for the other solvers, ode23, ode113 and rkf45.

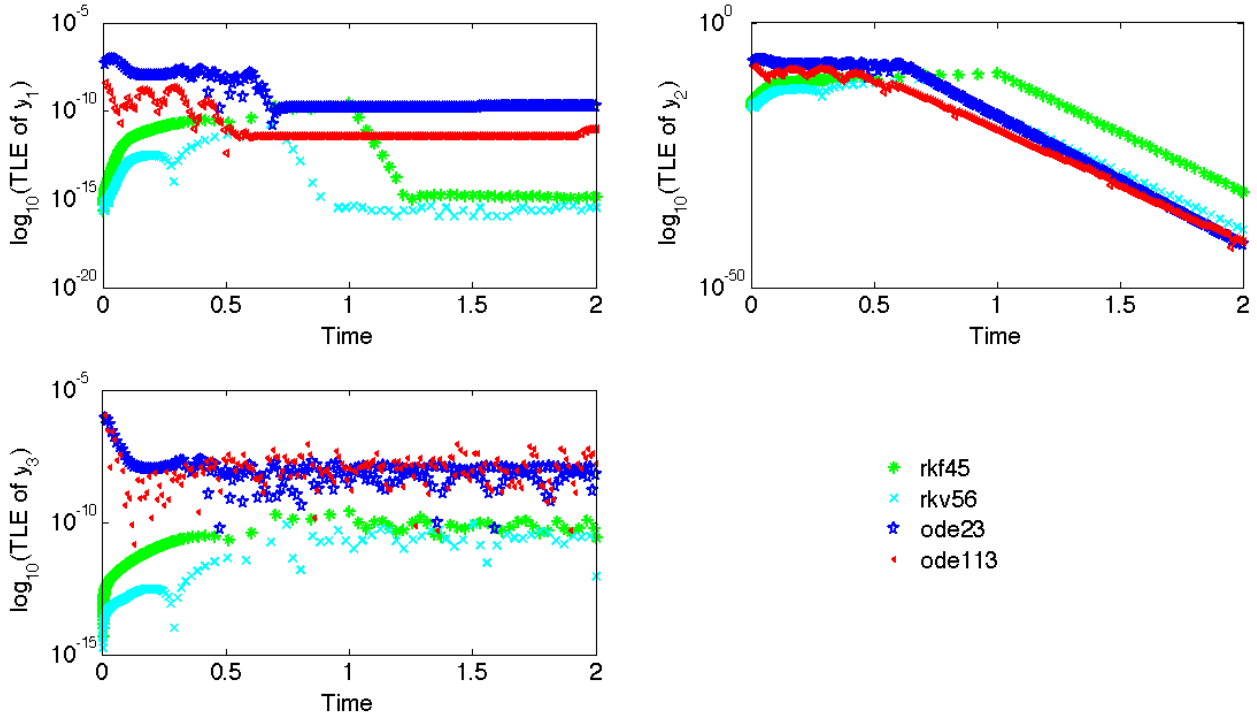

Figure 4.11 TLE for non–stiff solvers rkf45, rkv56 ode23 and ode113 under stringent tolerances.

In a second stage two other well performing solvers, namely ode45 and ode113, were set up to produce time points only at those instants given by rkv56. In Table 4.17 the relative errors for rkv56, ode45 and ode113 presented are calculated only for the same time points. The better performance of rkv56 is still outstanding, even more regarding the low average duration time.

Table 4.17 Relative errors for rkv56, ode45 and ode113 in the same time points. Default tolerances were used.

Method	Number of Events	Maximum Relative Error2	Relative Error2	Average Duration	First Duration
rkv56	348	3.96e-009	3.4088e-019	1.1921e-001	1.7933e-001
ode45	348	1.96e-004	4.0778e-010	5.7289e-002	6.3022e-001
ode113	348	1.03e-004	1.8438e-010	1.3698e-001	1.6293e-001

However the rkv56 solver has the limitation that one cannot obtain the solution at desired time instances, but instead those time instances chosen by the solver are given. In order to overcome this limitation a manipulation of the original code was made. The new code is from now on called rkv56_t. In Table 4.18 relative errors for rkv56_t are illustrated which show an even better performance than before. This increase in accuracy is due to the fact that the manipulation of the previous code involved an increase of the number of events, hence, a decrease of the step size, which ultimately lead to a higher accuracy. The prize to pay is of course the computational time, which naturally increases.

Table 4.18 Relative errors for rkv56, rkv_t ode45 and ode113 in the same time points, under default tolerances.

Method	Number of Events	Maximum Relative Error2	Relative Error2	Average Duration	First Duration
rkv56	348	3.96e-009	3.4088e-019	3.6912e-001	5.1250e-001
rkv56 t	603	2.00e-011	5.9213e-024	7.8710e-001	1.3061e+000
ode45	603	1.13e-004	2.1754e-011	2.8398e-001	2.0094e+000
ode113	603	9.74e-005	3.0628e-011	5.1477e-001	8.9307e-001

Hermite Interpolation of the numerical solution

For the solvers, rock4, rkv56, rkf45, vs_pc4, ode87 the solution of the ODE system is not obtained at predefined time instances. In case that the solution is desired at specific time points (other than the ones provided), an approximation schema would have to be applied. The most wildly employed schema in engineering is probably the one of Hermite cubic interpolations (Burden and Faires, 2005). Exemplary such a Hermite interpolation was carried out for the solver rock4. The results in terms of Relative Errors2 and others are shown in Table 4.1. It can be observed that the error values obtained in the case of interpolation are relatively high for the stringent tolerances. This implies that in this case the error which is due to the interpolation is greater than the error that is due to the numerical integration. Therefore one should avoid the Hermite interpolation and rather utilize a schema which provides the solution at the desired time instances, when a very accurate solution is desired.

4.2 Test Case B - The Robertson's Equations

This test case is chosen since (i) a chemical reaction system is represented; and (ii) the system is stiff (a. reported to be stiff (Robertson, 1966); b. the largest and the lowest time-varying eigenvalues differ significantly, see Figure 4.13), non-linear and time variant. In this case the analytical solution is, as for most systems, unknown. In fact, "if an IVP (initial value problem) arises from a practical situation, most likely you will not be able to solve it analytically yet you will be able to solve it numerically.", Gladwell & al. (2003).

4.2.1 Test Set Equations

In this section a chemical reaction of three reactants which was proposed by Robertson (1966) (wherefore this system of equations is many times, as in the following, referred to as the Robertson's equations), is addressed. The Robertson's equations describe quantitatively the depletion and generation of the species involved in the following chemical reaction:

$$\begin{aligned} A &\xrightarrow{k_1} B, & k_1 &= 0.04 \\ 2 \cdot B &\xrightarrow{k_2} B + C, & k_2 &= 3 \cdot 10^7 \\ B + C &\xrightarrow{k_3} A + C, & k_3 &= 10^{4\cdot} \end{aligned} \tag{4.5}$$

The reaction rate constants k_i quantify the speed of the chemical reaction, where each of the indexes $i = 1:3$ correspond to one chemical specie. Therefore the first reaction (involving k_1) is quite slow whereas the second reaction, in comparison, is very fast (another evidence that the system is stiff). It can be observed in Figure 4.12 that the second reaction is almost instantaneous, in the sense that almost immediately after the begin of the reaction, the concentration of B rises from 0 to $\approx$ 3.5E-5.

The time evolution of the reaction system was expressed by Robertson (1966) through the following system of ODEs:

$$\begin{aligned} y_1'(t) &= -0.04 \cdot y_1 + 10^4 \cdot y_2 \cdot y_3 \\ y_2'(t) &= 0.04 \cdot y_1 - 10^4 \cdot y_2 \cdot y_3 - 3 \cdot 10^7 \cdot y_2^2 \\ y_3'(t) &= 3 \cdot 10^7 \cdot y_2^2 \end{aligned} \tag{4.6}$$

Each ODE represents the evolution of the reactant concentration along the integration time, subject to the initial condition, $y_0 = [1, 0, 0]^T$. As indicated by the initial values there is only one reactant, namely A, at the beginning of the reaction. Note that the system described by equations 4.6 proposed by Robertson (1966) is normalized in such a manner that the sum of the values y_i at each time instant is one.

The Robertson's system of ODEs is part of a test set of ODEs assembled by Enright & al. (1975), in which the ODEs were divided into five different classes according to their properties: Linear/Non-linear, Real/Non-real eigenvalues. The system at hand belongs to the class, Non-linear with Real eigenvalues. The authors (Enright & al., 1975) decided to rescale this particular problem. In contrary Shampine (1981) recommended to apply the system for the evaluation of different numerical solution schema with its original coefficient values, since the greater magnitude of the coefficients results in a more stiff system. Further, Shampine (1981) outlines that the integration interval should reach from 0 to 40, i.e. $0 < t \leq 40$. Therefore, the original system with the original coefficients is applied, i.e. exactly the one given by equations 4.6 and the integration interval is: $0 < t \leq 40$. As can be concluded from the eigenvalues, shown in Figure 4.13, the system along the time is quite stiff, i.e. the stiffness ratio ($\frac{Re(\lambda_2)}{Re(\lambda_3)}$ see (Ashino and Vaillancourt, 2009)) is relatively high. Further, it can be observed that the condition

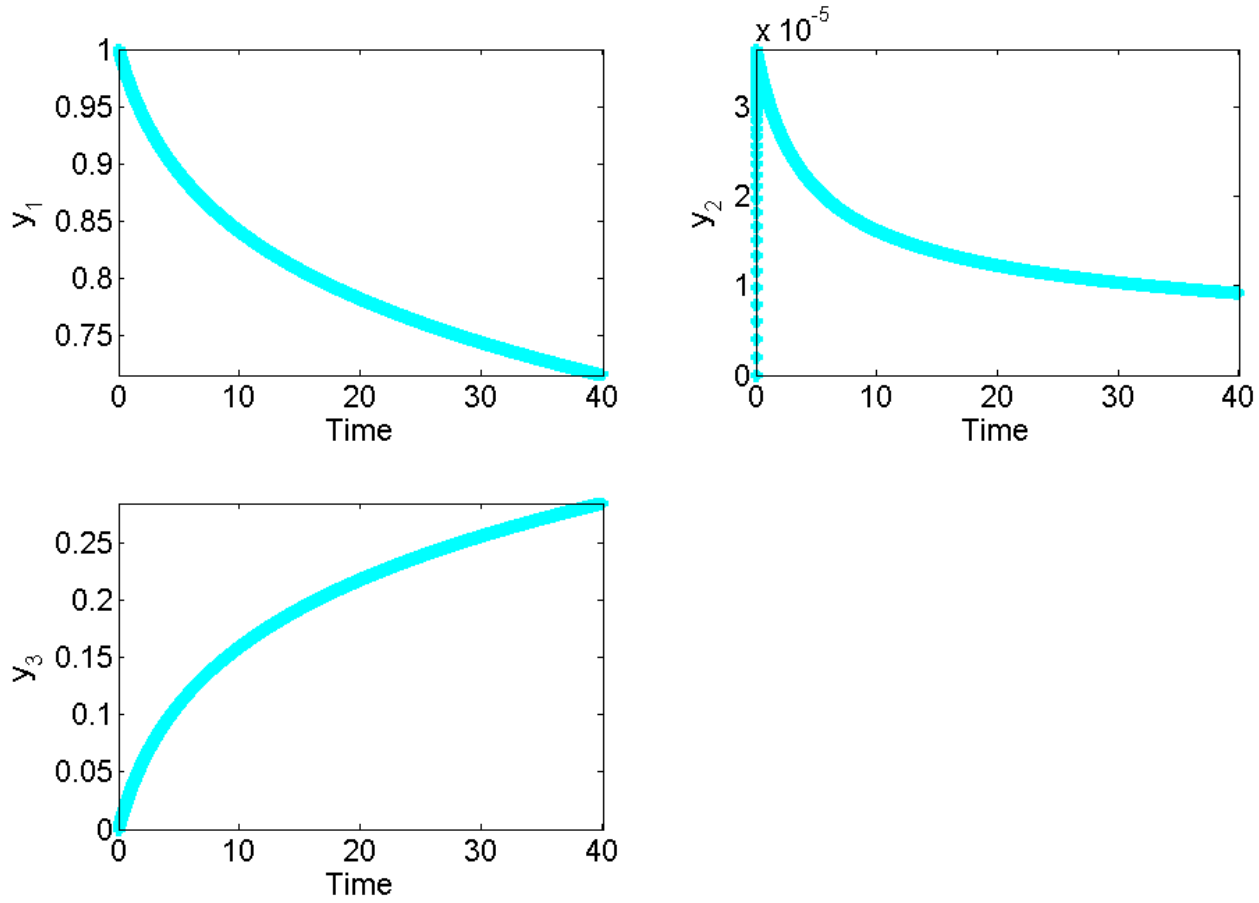

Figure 4.12 Time course for the Robertson's problem. (solver rk4. Step size=1E-4, Relative Error2 of 2.0588E-11)

number (which represents the sensitivity of the system to errors, see equation 3.8) is immense ($\approx$ 2.3E21, some of the values even tend to infinity [not plotted]), considering that a well-conditioned matrix has a condition number value of one.

4.2.2 Analysis of the Adaptive Step Size Solvers - Changing the Relative and/or Absolute Tolerances

Changes to tolerances only apply to the solvers with adaptive step size, namely rock4, ode15s, ode23s, ode23t, rkf45, rkv56, vs_pc4, ode87, ode45, ode23 and ode113. Note that the implementation of the solvers rkf45, rkv56 and vs_pc4 allow only for the manipulation of the absolute tolerance.

Local Error Estimation based on the Tolerance Proportionality

Since for the system of ODEs being investigated the analytical solution is unknown, the values of relative errors shown in Tables 4.19 and 4.20 are calculated utilizing local error estimations. The local error estimation which is based on the tolerance proportionality, (equation 3.3 in section 3.3) should provide an upper bound to the local error. For the other test case (see section 4.1.2) it was observed that (i) the LEE did not always provide an upper bound but a reasonable estimation of the magnitude of the TLE; and (ii) the best values of LEE were obtained for $q = 5$ and $\tau = 5$, which is in accordance with Calvo & al. (2008). Therefore in this case the LEE are calculated applying $q = 5$ and $\tau = 5$.

The calculation of the LEE requires the comparison of the solution y_i obtained under two different sets of tolerances

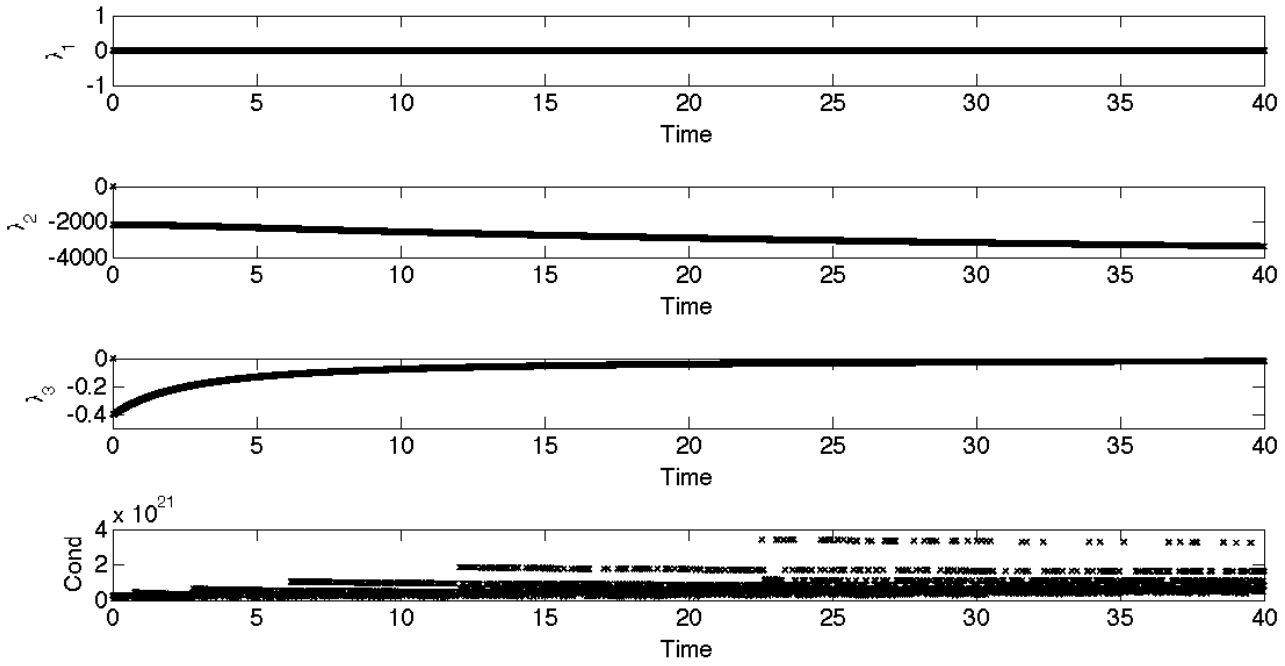

Figure 4.13 Time course of the eigenvalues and the condition number (solver: ode15s; default tolerances).

in the same time instant, t_i. Therefore only the *MATLAB suite* solvers were analyzed plus the best performing solver, rkv56, which was extended in order to deliver the output of the solution at the specified time instances (see section 4.1.7).

In Tables 4.19 and 4.20 the Relative Errors2, which are calculated based on the LEE, are displayed. In the cases of the variable order solvers, namely the ode15s and the ode113, the true relative errors are, most likely, lower, since the LEE (in order to provide an reliable upper bound) were calculated under the "worse case scenario "assumption, wherefore the lowest order was used for the calculation of the LEE for these solvers. In Table 4.19 the relative tolerance was varied in order obtain the LEE, while the absolute tolerance was fixed. (Note that the solver rkv56_t only use absolute tolerance wherefore no LEE are obtained). In this case it can be observed that the stiff solvers, in general, seem to be better than the non-stiff solver, except for the ode45. This can also be observed for the values given in Table 4.20, where the absolute tolerance was varied for the calculation of the LEE. Outstanding is again the performance of the rkv56_t solver, which comes with a higher computational cost. For the ode15s it can be seen that the Relative Error2 values are relatively low, even so order one was used for the LEE calculation, while the same cannot be stated for the ode113.

Table 4.19 LEEs obtained with equation 3.3 for the relative tolerance: 1E-8 and 5E-8; and the absolute tolerances: 1E-10.

τ	q	Method LEE	Number of events	Maximum Relative Error2	Relative Error2
5	5	ode15s	12003	4.3792e-007	3.0913e-014
5	5	ode23s	12003	2.3269e-007	3.8707e-016
5	5	ode23t	12003	2.0706e-006	5.8616e-013
5	5	ode45	12003	3.5115e-007	5.2470e-015
5	5	ode23	12003	3.5437e-006	1.1654e-012
5	5	ode113	12003	1.4975e-004	2.3764e-010

Table 4.20 LEEs obtained with equation 3.3 for the relative tolerance: 1E-8; and the absolute tolerances: 1E-10 and 5E-10.

τ	q	Method LEE	Number of events	Maximum Relative Error2	Relative Error2
5	5	ode15s	12003	1.0721e-006	2.5230e-015
5	5	ode23s	12003	2.1000e-006	4.6387e-013
5	5	ode23t	12003	6.4419e-007	2.0142e-015
5	5	ode45	12003	4.2946e-006	7.0733e-013
5	5	ode23	12003	2.7612e-005	7.1518e-011
5	5	ode113	12003	4.7455e-004	3.0003e-009
5	5	rkv56 t	12003	2.9158e-009	1.3955e-020

Analysis of the Fixed Step Size Solvers - Changing the Step Size

The changes in the step size only apply to the solvers with fixed step sizes, namely the solvers euler_modified, euler_forward, euler_backward, heun, opt_rk2, rk4, ab2 and adams_pc4. As a first estimate for the fixed step size, the initial step size given in literature Seinfeld and Lapidus (1973), 1E-5 was used. This proved to be very time consuming. Step sizes of 1E-4 and 5E-5 were found to be a good compromise between accuracy and cost.

Local Error Estimation based on Step Halving

Once there is no known analytical solution, the TLEs cannot be calculated. Instead, LEEs are calculated using the step halving method (see section 3.2), which provide a good upper bound for the TLE Shampine and Watts (1976). Based on these LEEs the relative error was calculated, see Table 4.21.

Table 4.21 Relative Errors2 obtained with the LEE over the fixed step size methods. Step size used: 1E-4.

Method	Order p	Number of events	Relative Error2
euler_modified	2	1200003	1.5971e-04
euler_forward	1	1200003	9.3241e-02
euler_backward	1	1200003	1.9007e-01
heun	2	1200003	1.5971e-04
opt_rk2	2	1200003	6.0364e-05
rk4	4	1200003	2.0588e-011
ab2	2	1200003	2.0387e-004
adams_pc4	4	1200003	1.4526e-008

Therein it can be observed that:

i) the higher the order of the methodology the better the value of the LEE (which can be expected);

ii) the modified euler and the heun method give identical results while this time, in contrast to the observations made in 4.1.3, the ones obtained with the solver opt_rk2 are better and thus not identical.

When analyzing the LEEs graphically, i.e. Figure 4.14, one can visually confirm the observation just made, namely: (i) the poor performance of the lower order methodologies; (ii) that the LEE over time for the euler_modified and heun solvers are almost superposed. Else it can be seen that, in general, the values of the LEE for the second and third state are greater than those observed for the first state, i.e., for the bottom subplot, concerning the forth order solvers, the LEE of y_1 is around the order 1E-10, while for y_2 and y_3 the order of magnitude is $\approx$ 1E-6.

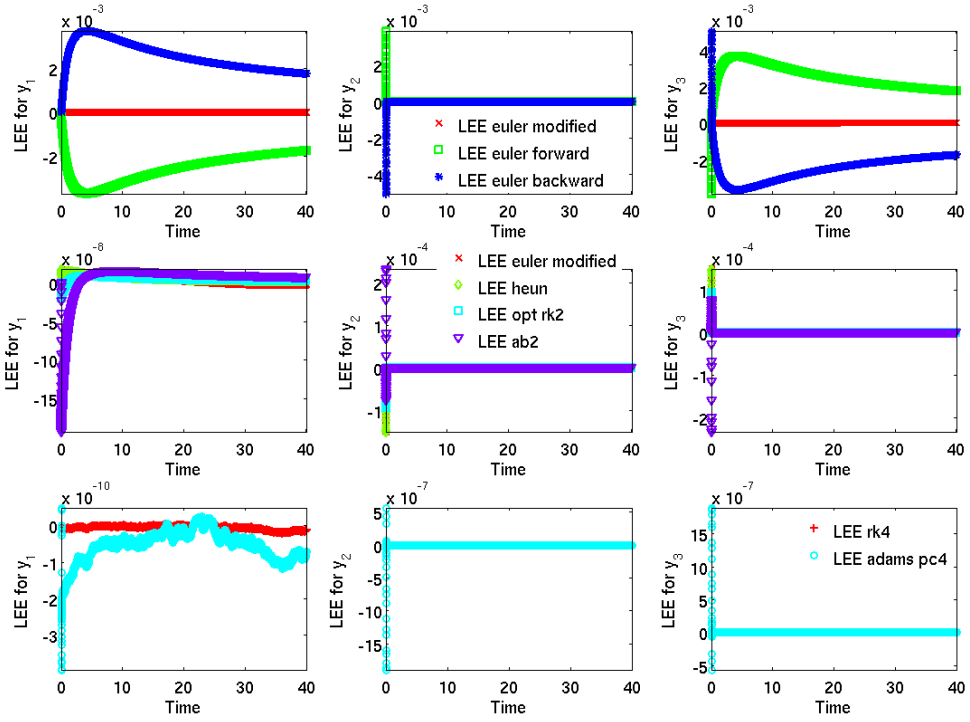

Figure 4.14 Local Error Estimation over the integrating interval for the fixed step size solvers. Step Size: 7E-3.

4.2.3 Disturbing the Initial Values

The Robertson's system of equations is normalized in such a manner that the sum of the values y_i in each time instant is one. Therefore, the initial values are not in the same manner disturbed as in the previous section, since this would violate physical constraints, e.g. $y_0' = [1, 0, 0] \cdot 1.05$, or 105% of the reactant A at $t = 0$. Instead, a disturbance in the initial condition was performed that would still respect $\sum_{i=1}^{3} y_i = 1 : y_{0,disturbed} = [0.88, 2E-5, 0.12]$. This initial condition could represent an 'poorly washed 'reactor that still contains traces from the reactants B and C.

Since the analytical solution is unknown, in the following only the relative condition number will be studied, which however reveals information not only about the ODEs system but also about the numerical integration schema, as was observed in section 4.1.5.

Relative Condition Number (κ_{rel})

The relative condition number (κ_{rel}) provides a measure whether a disturbance in the initial value along the integration is damped or amplified. In order to analyze how much each of the n states contributes to the system overall behavior, an additional κ_{rel} was calculated for each state, see Figure 4.15. Therein (for y_2), in case of the

solver ode23t, mild oscillatory behavior can be observed, which also was seen in section 4.1.5. In case of all three states, the value of κ_{rel}, calculated for each state separately, is decreasing along time, which is in agreement with the behavior of κ_{rel} shown in Figure 4.16. Therefore a disturbance in the initial condition, in this case, is damped along time. However it can be seen, that again, the numerical integration scheme contributes to error propagation.

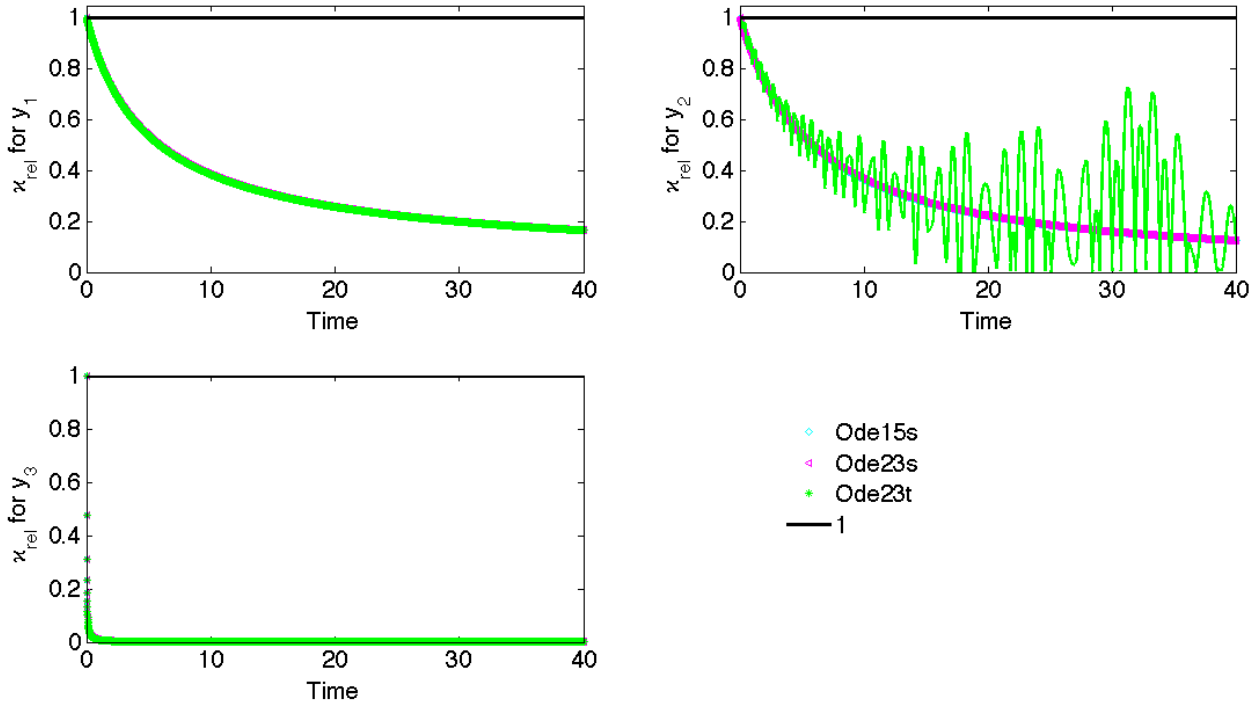

Figure 4.15 κ_{rel} for the stiff solvers decomposed in the three states. Default tolerances were used.

4.2.4 Knowledge Incorporation in form of the Jacobian

The *MATLAB suite* stiff solvers allow for the incorporation of additional knowledge about the ODE system i.e. the Jacobian. The Robertson's system of ODEs does not have a constant Jacobian, in contrast to the test case A, which was studied in the previous section. Yet it is possible and advisable to provide the Jacobian for the numerical integration (Gladwell & al., 2003), since this will allow the solver to calculate analytical Jacobian at each time step instead of calculating an approximation through finite differences (which is time consuming). However, the incorporation of the Jacobian function does not improve the quality of the results but it will cause a decrease in the number of function evaluations needed and therefore in the time required for the integration. A resume of the statistical information for the stiff solvers without and with the incorporation of the Jacobian function is displayed in Table 4.22. By inspecting the information given in Table 4.22 it can be seen that the solver which benefits the most of the incorporation is the ode23s (131 vs 67 function evaluations). This behavior was also noticeable in the incorporation of the Constant Jacobian information in the previous section (Test Set A, Table 4.22). However in contrast to the prior observations, this time, the other solvers also used the non-constant Jacobian, which is observed through the lower computational times and the lower number of function evaluations.

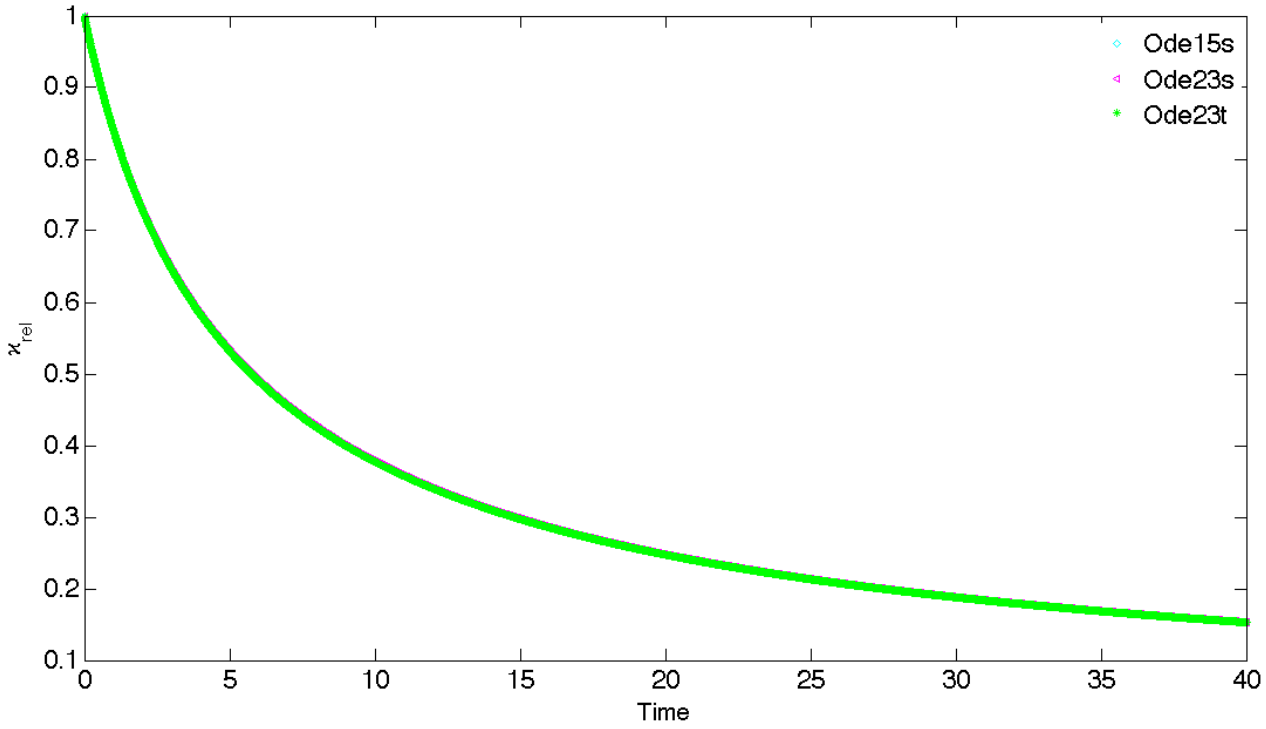

Figure 4.16 κ_{rel} for the stiff solvers, under default tolerances.

Table 4.22 Statistics concerning the Jacobian Incorporation. Results are obtained under default tolerances.

Method	Given Jacobian	Successful steps	Failed attempts	Function evaluations	Partial derivatives	Duration
ode15s	No	50	5	110	3	1.44
ode15s	Yes	50	5	98	3	1.20
ode23s	No	21	1	131	21	5.25
ode23s	Yes	21	1	67	21	1.04
ode23t	No	43	7	128	5	1.07
ode23t	Yes	43	7	108	5	0.78

4.2.5 General Observations

Duration Times for Stiff and Non Stiff Solvers

The average duration times for the integration using the stiff solvers is much smaller than those of the non-stiff solvers (adaptive step size methods), see Table 4.23. This fact underlines that the test case B system (the Robertson's equations) for the analyzed initial conditions and integration time, is stiff.

The fixed step size solvers do not control the error through tolerances, therefore a straightforward comparison concerning the duration times with the adaptive step size solvers is difficult. Still, the experiments performed with the fixed step size solvers with a step size of 5E-5 produce LEEs that are similar to those of the adaptive step size solvers using default tolerances ($RelTol = 1E-3$ and $AbsTol = 1E-6$). Therefore on the basis of the similar errors, a comparison of the computational times can be carried out. In this context it can be observed that the fixed step size solvers of higher order, namely the rk4 and the adams_pc4 (see Table 4.24), have duration times which are similar to the ones of the (non-stiff) adaptive solvers (see Table 4.23).

Table 4.23 Duration times for the adaptive step size solvers under *default* tolerances.

Method	Duration
rock4	5.64
ode15s	4.61
ode23s	2.63
ode23t	2.4
rkf45	156.76
rkv56	125
vs_pc4	267.77
ode45	43.1
ode23	11.39
ode113	41.95

Table 4.24 Duration times for the fixed step size solvers. Step sizes used: 5E-5 and 2.5E-5.

Method	Number of events	Duration
euler_modified	400001	55.01
euler_modified	800001	108.96
euler_forward	400001	25.37
euler_forward	800001	47.73
euler_backward	400001	48.88
euler_backward	800001	91.8
heun	400001	30.87
heun	800001	55.87
opt_rk2	400001	27.91
opt_rk2	800001	56.22
rk4	400001	55.66
rk4	800001	111.19
ab2	400001	16.67
ab2	800001	32.7
adams_pc4	400001	30.45
adams_pc4	800001	60.74

Further it can be seen that the integration applying the heun solver lasts approximately half the time than when integrated with the euler_modified (see Table 4.24), even though they (i) base on the same method and (ii) produce identical Relative Error2 values (as seen above, Table 4.21). Thus the implementation of the heun method seems to be superior.

Chapter 5

Conclusions

In this thesis a collection of numerical integration schema, that are all implemented in the *MATLAB* environment, were applied to two test cases, namely Test Case A (a linear, mildly-stiff system) and Test Case B - also called the Robertson's equations - (a non-linear, time-variant, stiff system). When applying changes to the solver parameters (variation of absolute tolerance and relative tolerance values, variation of the step size, incorporation of additional information i.e. initial step size or the Jacobian) and disturbances to the initial conditions of the systems, the following was observed:

i. In both test cases A and B the best performance was observed for the rkv56 solver. While this performance came with a relatively low cost in case A, see Table 4.1, in case B computational cost was much higher, e.g. section 4.2.5;

ii. In test case A, for the stiff solvers (ode15s, ode23s, ode23t and rock4), the computational cost of the solution was not lower than for the non-stiff solvers, another evidence that this test case is mildly stiff.

iii. The computational cost of the stiff solvers in the test case B, in general, was significantly lower than the one of the non-stiff solvers, which is another proof that this case is stiff.

iv. In case that it is intended to use a fixed step size solver, e.g. when measurements are incorporated into the ODE system and one wants to avoid their interpolation, it is advised to utilize either the heun scheme (fast and relatively low error) or the rk4 (even lower errors but dramatically higher computational cost, see section 4.1.3) since (a) the step size of this scheme can relatively easy be fitted to a regular sampling frequency; and (b) the quality of the numerical solution when compared to first order Euler scheme (backward or forward) is much higher.

v. Specification of the initial step size value (in adaptive solvers) has relatively low impact on the quality of the solution (Section 4.1.4). These findings are in agreement with Krogh (1973);

vi. In case of the solvers with fixed step size, the same order of error magnitude is associated with a much higher computational cost, when compare to the respective values obtained by the stiff adaptive step size solvers, i.e.: average durations $\approx$ 10-100 times higher than the duration times for stiff solvers, Tables 4.24 and 4.23. This is due to the fact that the adaptive step size solvers utilize a small step size only where necessary, while in case of the fixed step size solvers the same step size is used throughout the entire integration.

vii. While for Test Case A, the fixed step size solver opt_rk2 produces the same errors (both TLE and LEE) than the euler_modified and the heun solvers (see section 4.1.3); in Test Case B, (see Table 4.21) the LEE values obtained through the opt_rk2 solver are one order better than the respective values obtained by the euler_modified or the heun algorithm (which produce identical LEEs).

viii. For the adaptive step size solvers (ode15s, ode23s, ode23t, ode23, ode45, ode113, rkf45, rkv56, vs_pc4, ode87 and rock4), the LEE does not always provide an upper bound to the TLE, but the order of the LEE is close to the one of the TLE. Such conclusion is only possible in test case A, where there were the analytical solution.

ix. Additional information in form of the *Jacobian* can be provided to the stiff solvers. In Test Case A (*constant Jacobian*), a decrease in computation time was observed only for the ode23s, see Table 4.16. This is in agreement with Gladwell & al. (2003), because the other stiff solvers do only recalculate the Jacobian when they believe that it is necessary. In Test Case B, incorporation of the Jacobian function lead to a decrease in the number of function evaluations (and therefore a decrease in the duration) for all solvers, i.e. ode15s, ode23t and ode23s. In case of the latter, the greatest decrease in the number of function evaluations (131 vs 67) was observed, which is consistent with the five fold lower integration time (Table 4.22).

x. When a controlled disturbance was applied to the initial values, the errors obtained due to the disturbance were much greater than those errors that were caused by the numerical integration. This effect is known as error propagation and of course depends on the ODEs system. However, in the case of the numerical solution it could be observed that the numerical integration scheme contributed to the error propagation.

xi. For Test Case A, a relatively low influence of the changes in the absolute tolerance on the overall quality of the numerical solution was observed. The reason is that the numerical solution is constrained by the absolute tolerance only when the product of state and of relative tolerance values is below the one of the absolute tolerance. Thus the choice of the relative tolerance value, in general, seems to be more critical than the one of the absolute tolerance. While this should be kept in mind for practical considerations, note that none of the tolerances values should be zero (Gladwell & al., 2003).

xii. For the case that the solvers do not provide the solution at the user specified time instances (else than the end point of the integration), it was investigated, whether a Hermite cubic interpolation of the solution results into a significant increase of error. It was concluded that the error of the numerical solution increases significantly due to the interpolation in the case that the integration was performed with stringent tolerances. Therefore it is advised to use a schema for which the solution time instances can be specified in case that a very accurate solution is desired.

xiii. A good performance was expected in case of the adaptive step size solver ode87, which has orders of eight and seven, but the results obtained were at the same quality level than the one observed for the solvers of fourth and fifth order (Table 4.1).

Chapter 6

Evaluation of the realized work

6.1 Realization of the Objectives

The following listed objectives were, as planned, addressed in this thesis. All objectives were fully reached and it can be stated that to the best of my knowledge, all methodologies for the integration of first order ODEs systems (that are readily applicable) that exist for the MATLAB environment were evaluated in this thesis.

6.1.1 Background

The background can be divided into three major points:

Test Cases

An overview of the test cases that can be found in literature is given in the introduction. In this thesis two test case examples were addressed, that were taken from Seinfeld and Lapidus (1973) and Enright & al. (1975). The specific properties of these test cases are complementary and the reasons for their selection is outlined in the respective sections.

Evaluation Criteria

In the introduction an overview on literature that cover evaluation criteria is given. In section 3 the evaluation criteria applied in this thesis are presented. The main aspect for the evaluation of the numerical integration schema is the quality of the solution and the associated computational cost.

Numerical methods

A number of algorithms for the integration of first order ODEs were collected and classified in the first part of this thesis. A brief introduction on the background of numerical integration, and the respective methods is also given. Also parameters were outlined that serve to adapt the numerical solution to specifications given by the user.

6.1.2 Evaluation and Benchmarking

The collected integration methods were applied to the two test cases, Test Case A (a linear, mildly stiff, ODEs system with known analytical solution) and Test Case B (a non-linear, stiff, time-variant ODEs system with unknown analytical solution). The obtained numerical solutions were then evaluated applying the above mentioned evaluation criteria. Based on these criteria the integration methods were compared and conclusions drawn.

6.2 Other accomplished work

Two major points that are also covered in the thesis but did not only serve for the evaluation of integration methods are listed in the following.

6.2.1 Hermite cubic interpolation

Since in some cases the numerical solution is not obtained at specified time instances, an interpolation schema has to be applied. Here an Hermite cubic interpolation was applied and it was investigated to what extend the interpolation contributes to the final error. It was found that only in the case that the quality of the solution is desired with a high accuracy (stringent tolerances), the Hermite interpolation contributes significantly to the error, wherefore in this case a different integration scheme should be applied.

6.2.2 Error propagation

Error propagation that is due to disturbances in the initial values was also investigated, which however is almost completely due to properties of the ODEs system and not due to the numerical integration scheme chosen. A criteria, namely the relative condition number was applied which provides information about the fact whether a disturbance in the initial values, is along the time, damped or amplified. This is particularly interesting for the field of parameter estimation, since therein usually the initial values are measured values, which naturally come along with measurement errors. However, it could be observed that the error that is due to the initial disturbance is greater than the one that is due to the numerical integration.

6.3 Limitations and Future work

Some of the presented numerical integration schema allow the user to specify additional parameters to either incorporate more a priori available information about the ODEs system at hand, or to define other properties concerning the numerical solution. These other parameters were not investigated in this thesis.
Another shortage of this thesis is that one numerical integration algorithm, namely the solver DASPK, which is based on a Fortran routine and was encountered in the internet, could not be applied, since the computer system specifications did not allow it. In future work it would be interesting to develop a MATLAB interface for this solver and to then, in a second step, provide the interface along with the Fortran source code for individual compilation.
In the case of the integration of measured variable values into the ODEs system it was mentioned that usually fixed step size solvers find application. In this respect it would be interesting to develop an adaptive step size solver that complies to the provided sampling frequency of the measured values and rather adapts the integration method order than the step size.

6.4 Personal view

Though this thesis is in its essence a mathematical work, I think that the investigation is very fruitful to chemical engineers, in the sense that we frequently encounter problems (e.g., mass or energy balances, in the form of ODEs) where the analytical solution is not possible, but for which solution a wide range of numerical “tools ”exist. These numerical methods can (within the accuracy required) perform just as well, as if there was an exact solution as long as one knows about its limitations.

References

Abdulle, A. (2002), 'Fourth order chebyshev methods with recurrence relation', *SIAM J SCI COMPUT*, **23 (6)**, 2041–2054.

Ashino, R. and Vaillancourt, R. (2009), *Numerical Methods with Matlab*, Department of Mathematics and Statistics, University of Ottawa, Ottawa, ON, Canada.

Belardini, P., Bertoli, C., Corsaro, S. and D'Ambra, P. (2005), *The Impact of Different Stiff ODE Solvers in Parallel Simulation of Diesel Combustion.*, HPCC'05.

Brugnano, L. and Trigiante, D. (1998), 'Boundary value methods: The third way between linear multistep and runge-kutta methods', *Computers & Mathematics with Applications*, **36**(10-12), 269 – 284, URL `http://www.sciencedirect.com/science/article/pii/S089812219880028X`.

Burden, R. and Faires, J. (2005), *Numerical Analysis*, Bob Pirtle.

Burghes, D. N. and Borrie, M. S. (1981), *Modelling with differential equations / D.N. Burghes and M.S. Borrie*, E. Horwood ; Halsted [distributor], Chichester : New York :.

Butcher, J. C. and Wanner, G. (1996), 'Runge-kutta methods: some historical notes', *Applied Numerical Mathematics*, **22**(1-3), 113 – 151, URL `http://www.sciencedirect.com/science/article/pii/S0168927496000487`, special Issue Celebrating the Centenary of Runge-Kutta Methods.

Calvo, M., González-Pinto, S. and Montijano, J. I. (2008), 'Global error estimation based on the tolerance proportionality for some adaptive runge-kutta codes', *J. Comput. Appl. Math.*, **218**, 329–341.

Calvo, M. and Montijano, J. I. (1996), 'Global error estimation with adaptive explicit runge-kutta methods', *IMA Journal of Numerical Analysis*, 47–63.

Conte, S. D. and de Boor, C. (1981), *Elementary Numerical Analysis, An Algorithmic Approach*, McGraw-Hill International Editions.

Daun, S., Rubin, J., Vodovotz, Y. and Clermont, G. (2008), 'Equation-based models of dynamic biological systems', *Journal of Critical Care*, **23**(4), 585–594, URL `http://www.sciencedirect.com/science/article/pii/S0883944108000506`.

Enright, W. H., Hull, T. E. and Lindberg, B. (1975), 'Comparing numerical methods for stiff systems of o.d.e:s', *BIT Numerical Mathematics*, **15**, 10–48, URL `http://dx.doi.org/10.1007/BF01932994`, 10.1007/BF01932994.

Gladwell, I., Shampine, L. and Thompson, S. (2003), *Solving ODEs with MATLAB*, Cambridge University Press, New York, NY, USA.

Hairer, N. and Wanner (1993), *Solving Ordinary Differential Equations. Nonstiff Problems*, Springer Series in Comput. Math.

Hull, E., Enright, H., Fellen, M. and Dgwwk, E. S. (1972), 'Comparing numerical methods for ordinary differential equations', *Numer. Anal*.

Krogh, F. T. (1973), 'On testing a subroutine for the numerical integration of ordinary differential equations', *J. ACM*, **20**(4), 545–562.

Oster, G. F. and Perelson, A. S. (1974), 'Chemical reaction dynamics', *Archive for Rational Mechanics and Analysis*, **55**, 230–274, URL http://dx.doi.org/10.1007/BF00281751, 10.1007/BF00281751.

Petzold, L. (1981), 'An efficient numerical method for highly oscillatory ordinary differential equations', *SIAM J. Numer. Anal*.

Petzold, L. (1983), 'Automatic selection of methods for solving stiff and non-stiff systems of ordinary differential equations', *Siam Journal on Scientific Computing*.

Rentrop, P. (1985), 'Partitioned runge-kutta methods with stiffness detection and stepsize control', *Numerische Mathematik*, **47**, 545–564, URL http://dx.doi.org/10.1007/BF01389456, 10.1007/BF01389456.

Robertson, H. (1966), *The solution of a set of reaction rate equations. In Numerical Analysis, An Introduction*, Academic Press, London.

Seinfeld, J. and Lapidus, L. (1973), *Numerical solution of ordinary differential equations*, Academic Press.

Shampine, L. F. (1977), 'Stiffness and nonstiff differential equation solvers, ii: Detecting stiffness with runge-kutta methods', *ACM Trans. Math. Softw.*, **3**(1), 44–53.

Shampine, L. F. (1981), 'Evaluation of a test set for stiff ode solvers', *ACM Trans. Math. Softw.*, **7**(4), 409–420.

Shampine, L. F. and Gladwell, I. (1996), 'Software based on explicit rk formulas', *Applied Numerical Mathematics*, **22**(1-3), 293–308, URL http://www.sciencedirect.com/science/article/pii/S0168927496000396.

Shampine, L. F. and Hiebert, K. L. (1977), 'Detecting stiffness with the fehlberg (4, 5) formulas', *Computers & Mathematics with Applications*, **3**(1), 41–46, URL http://www.sciencedirect.com/science/article/pii/0898122177901122.

Shampine, L. F. and Reichelt, M. W. (1997), 'The matlab ode suite', *SIAM Journal on Scientific Computing*, **Vol. 18**, pp 1–22.

Shampine, L. F. and Watts, H. A. (1976), 'Global error estimates for ordinary differential equations.', *ACM Trans. Math. Softw.*, 172–186.