Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.

http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.

http://www.ub.tuwien.ac.at/eng

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

DIPLOMARBEIT

Splitting-Verfahren für nichtlineare Evolutionsgleichungen

Ausgeführt am Institut für

Analysis und Scientific Computing
der Technischen Universität Wien

unter der Anleitung von Prof. Winfried Auzinger

durch

Michael Quell

Döblergasse 4/12
1070 WIEN

24.01.2018

Unterschrift

Abstract

This thesis motivates and summarizes the construction of splitting schemes for evolution equations. Further asymptotic correct local error estimators are proposed, to adapt the step-size in time propagation and reduce total computation time. Commutator free Magnus-type integrators are compared to splitting off the time variable. The Methods are tested by three applications: Models for solar cells, reaction-diffusion system by Gray-Scott and Schrödinger wave equations.

Keywords: Evolution equations, splitting methods, order conditions, local error estimators, commutator free Magnus integrators,

I declare by my signature that this assignment is my own work and that I have correctly acknowledged the work of others. This assignment is in accordance with University guidance on good academic conduct.

Vienna, January 24, 2018

Michael Quell

Acknowledgments

I wish to thank various people for their contribution to this thesis. First I would like to acknowledge the support provided by my family during the hard times of my study. I am particularly grateful for the assistance given by Dr. Benson Muite in the beginning. Special thanks should be given to Professor Winfried Auzinger, my research project supervisor for his professional guidance and valuable support and to Dr. Othmar Koch for his useful and constructive recommendations on this project. I would also like to thank Dr. Harald Hofstätter for his technical advice.

Contents

1 Introduction **7**

2 Operator Splitting Methods **8**

3 Order Conditions **10**
- 3.0.1 Solving order conditions 11

4 Local Error Estimators **13**
- 4.1 Defect based 13
  - 4.1.1 Efficient evaluation of the defect 14
- 4.2 Embedded formulae 16
- 4.3 Palindromic schemes 16
- 4.4 MilneDevice 17
- 4.5 Time-adaptive strategies 18

5 Commutator free exponential Time propagators / Magnus **19**
- 5.1 Magnus-type integrators 19
- 5.2 Commutator free Magnus integrators 20
  - 5.2.1 Local error and defect. Order conditions and defect based local error estimate 20
  - 5.2.2 Efficient evaluation of the defect 22
- 5.3 Splitting off the time variable “ freezing ” 22
  - 5.3.1 Solution 23

6 Examples and Results **25**
- 6.1 Methods used 25
  - 6.1.1 Magnus 25
  - 6.1.2 Splitting 25
  - 6.1.3 Hardware 25
- 6.2 Solar cells: Application of splitting and Magnus-type integrators . 26
  - 6.2.1 Implementation 26

- 6.2.2 Matrix exponential 26
- 6.2.3 Results (2 electrons) 27
- 6.2.4 Results (8 electrons) 35
- 6.3 Gray-Scott equation 42
  - 6.3.1 Implementation 45
  - 6.3.2 Visual results 46
- 6.4 Schrödinger Equations 47
  - 6.4.1 The cubic NLS 47
  - 6.4.2 Quantum control 50

7 Spectral Method **56**

Chapter 1

Introduction

In many fields in science researchers model systems over time. This leads to the class of differential equations called evolution equations. Often the their problems are to complex to be linear, therefore the nonlinear case is studied too. The thesis starts in Chapter 2 with an motivation to use splitting methods on solving nonlinear evolution equations. Then is followed by a Chapter 3 about getting the order conditions and constructing splitting schemes. Explicit algorithms are not given, though they are referenced. The next Chapter 4 gives 4 different methods to estimate the local posterior error of the splitting schemes. The Chapter 5 analyzes a a special kind of evolution equations, where the numerical results and numerical comparisons are than displayed in the next Chapter 6. Three types of evolution equations are discussed

- linear time dependent right hand side in Section 6.2,
- nonlinear parabolic equations in Section 6.3 and
- wave equations in Section 6.4.

In all the three cases it is investigated how well the splitting methods perform, especially in 6.2 they are compared to the Magnus-type integrators. It is also tried to answer the question if the computational time can be improved using adaptive step size selection algorithms based on an estimated local error. In the last Chapter 7 the underlying concept of the space discretization is explained in Section 6.3 and 6.4.

Chapter 2

Operator Splitting Methods

We consider evolution equations of the form

$$\partial_t u(t) = F(u(t)), \quad t \geq 0, \quad \text{and } u(0) \text{ given.} \tag{2.1}$$

In many cases splitting the right-hand side $F(u)$ into two ore more components leads to an easy way to achieve accurate and efficient results. [23] Let the right-hand side be

$$F(u) = A(u) + B(u) + (C(u) + \ldots). \tag{2.2}$$

In many cases one can integrate all the sub-problems exactly, reducing the complexity and leading to simple and explicit numerical algorithms. A collections of splitting schemes. I would recommend and which was used here, is authored by W. Auzinger and O. Koch and can be found at

`http://www.asc.tuwien.ac.at/~winfried/splitting`

this will be referenced by [5]. This collection lists only schemes, where no assumptions were made on the operators, though for special cases one may find optimized schemes somewhere else. First we will discuss the case, when the right hand side is split only in to two components, as the notation is shorter and everything can easily be expanded to more terms. This results in an evolution equation (2.1) where the right-hand side is split into two operators,

$$\partial_t u(t) = F(u(t)) = A(u(t)) + B(u(t)), \quad t \geq 0. \tag{2.3}$$

A single step of a multiplicative splitting scheme, starting from u, having s stages and over a step of length h, is given by

$$\mathscr{S}(h,u) = \mathscr{S}_s(h, \mathscr{S}_{s-1}(h, \ldots, \mathscr{S}_1(h,u))) \approx \mathscr{E}_F(h,u)\,, \tag{2.4a}$$

with

$$\mathscr{S}_j(h,u) = \mathscr{E}_B(b_j h, \mathscr{E}_A(a_j h, u))\,, \tag{2.4b}$$

with appropriate coefficients a_j, b_j. $\mathscr{E}_F$ denotes the exact flow associated with the given evolution equation (2.3). For the simplest case $s = 1$ and $a_1 = b_1 = 1$, one can easily verify by Taylor expansion of the solution that this yields an scheme of order 1. The computational effort to achieve high accuracy is lower for higher order methods. There are general two ways to create higher order schemes, either generating and solving a system of polynomial order conditions or by composition of lower order methods. The composition is not discussed here. For a scheme of order p the local error of a splitting step, which is denoted by

$$\mathscr{S}(h,u) - \mathscr{E}_F(h,u) =: \mathscr{L}(h,u) = \mathscr{O}(h^{p+1}), \tag{2.5}$$

is of order $p+1$ for $h \to 0$.

Chapter 3

Order Conditions

As stated in [3] it is sufficient, to assume a linear right hand side F to construct the order conditions ,

$$\frac{d}{dt}u(t) = Fu(t) = (A+B)u(t), \quad u(0)\text{given.} \tag{3.1}$$

This gives for (2.4)

$$\mathscr{S}(h) = \mathscr{S}_s(h, \mathscr{S}_{s-1}(h, \dots, \mathscr{S}_1(h))) \approx \mathscr{E}_F(h) = \mathrm{e}^{hF}, \tag{3.2}$$

with

$$\mathscr{S}_j(h) = \mathrm{e}^{hB_j}\mathrm{e}^{hA_j}, \quad A_j = a_jA \quad ,B_j = b_jB, \quad j = 1, \dots, s. \tag{3.3}$$

In this case the local error (2.5) is of the form (2.5) with a linear operator $\mathscr{L}(h)$. The Taylor expansion of $\mathscr{L}(h)u$ at $h = 0$ up to order $p+1$ gives,

$$\mathscr{L}(h)u = \sum_{q=1}^{p+1} \frac{h^q}{q!}\frac{d^q}{dh^q}\mathscr{L}(0)u + \mathscr{O}(h^{p+2}). \tag{3.4}$$

If

$$0 = \frac{d}{dh}\mathscr{L}(0) = \frac{d^2}{dh^2}\mathscr{L}(0) = \dots = \frac{d^p}{dh^p}\mathscr{L}(0) \quad \Leftrightarrow \quad \mathscr{L}(h) = \mathscr{O}(h^{p+1}), \tag{3.5}$$

then the method is of order p and the leading local error term is given by

$$\frac{h^{p+1}}{(p+1)!}\frac{d^{p+1}}{dh^{p+1}}\mathscr{L}(0), \tag{3.6}$$

whereas

$$\frac{d^q}{dh^q}\mathscr{L}(0) = \sum_{|\mathbf{k}|=q} \binom{n}{\mathbf{k}} \cdot \prod_{j=1}^{s}\sum_{l=0}^{k_j}\binom{k_j}{l} B_j^l A_j^{k_j-l} - (A+B)^q, \tag{3.7}$$

with $\mathbf{k} = (k_1, k_2, \ldots, k_s) \in \mathbb{N}^s$ and $|\mathbf{k}| = k_1 + k_2 + \cdots + k_s$. The righthandside of (3.7) can be written [9] as linear combination of higher order commutators of A and B. Consider the ring

$$\mathbb{C}\langle A, B\rangle \tag{3.8}$$

in the not commuting powerseries of A and B and with the Lie-bracket

$$[A, B] := AB - BA, \tag{3.9}$$

which will also sometimes be reffered as commutator of A and B, gives an free Lie algebra. The representation in the Lyndon–Shirshov basis [3] of the free Lie algebra generated by A and B yield non-redundant order conditions. To compare to methods of the same order, one considers the leading error term (3.6) of the local error representation,

$$\frac{h^{p+1}}{(p+1)!} \frac{d^{p+1}}{dh^{p+1}} \mathscr{L}(0) = \sum_{k=1}^{l_{p+1}} \kappa_{p+1,k} K_{p+1,k}, \tag{3.10}$$

with l_{p+1} commutators $K_{p+1,k}$ of order $p+1$,and computes

$$\left(\sum_{k=1}^{l_{p+1}} |\kappa_{p+1,k}|^2 \right)^{\frac{1}{2}}. \tag{3.11}$$

or, in fact easier to compute and local error measure (LEM)

$$LEM := \left(\sum_{k=1}^{l_{p+1}} |\lambda_{p+1,k}|^2 \right)^{\frac{1}{2}}. \tag{3.12}$$

The $\lambda_{p+1,k}$ are the coefficients generated for the order condition for $p+1$. Because the LEM varies over several orders of magnitudes (3.11) and (3.12) are equally reasonable; See [9] for details.

3.0.1 Solving order conditions

The number of order conditions grows exponentially, with the number of stages. Higher order schemes found at [5] are numeric solutions to the order conditions and than refined with software floats and Newton method to full double precision. If the number of coefficients is larger than the number of equations the LEM (3.12) is minimized using state of the art solvers or plain Monte Carlo simulation, though the minimum given by the Monte Carlo search is not guaranteed to be the global minimum. An alternative is to reduce the number of order condition by considering special symmetries. For example one considers symmetric schemes,

$$a_j = a_{s+1-j} \text{ and } b_j = b_{s-j}, \text{ with } b_s = 0, \tag{3.13}$$

which are a standard choice for long time integration. Another possibility is to exchange the roles of A and B, leading to reflected coefficients,

$$a_j = b_{s+1-j} \text{ and } b_j = a_{s+1-j}, \tag{3.14}$$

or so-called palindromic schemes.

Chapter 4

Local Error Estimators

In this chapter various asymptotically correct local error estimators will be proposed. The local error estimator is denoted by

$$\mathscr{P}(h,u) \approx \mathscr{L}(h,u), \tag{4.1}$$

which has the same order as the local error and $\mathscr{P}(h,u) - \mathscr{L}(h,u) = \mathscr{O}(h^{p+2})$. The estimate will be used to adapt the step size.

4.1 Defect based

In [4, 6, 7, 8], asymptotically correct local error estimators based on the defect which is defined by plugging the splitting operator $\mathscr{S}$ into (2.1)

$$\mathscr{D}(h,u) = \frac{d}{dh}\mathscr{S}(h,u) - F\mathscr{S}(h,u), \tag{4.2}$$

have been constructed and analysed. This idea is based on the integral representation

$$\mathscr{L}(h,u) = \int_0^h \underbrace{\partial_2\mathscr{E}_F(h-\tau,\mathscr{S}(\tau,u))\mathscr{D}(\tau,u)}_{=:\ell(\tau)}\,d\tau, \tag{4.3}$$

The notation for the Fréchet derivative of the flux $\mathscr{E}_F(h,u)$ is, $\partial_2\mathscr{E}_F(h,u)$. One can verify by repeating differentiation and evaluation for $h = 0$ that for a scheme of order $p \geq 1$ it is equivalent that

$$\begin{aligned} 0 &= \frac{d}{dh}\mathscr{L}(0,u) = \frac{d^2}{dh^2}\mathscr{L}(0,u) = \ldots = \frac{d^p}{dh^p}\mathscr{L}(0,u) && (4.4) \\ \Leftrightarrow \quad 0 &= \mathscr{D}(0,u) = \frac{d}{dh}\mathscr{D}(0,u) = \ldots = \frac{d^{p-1}}{dh^{p-1}}\mathscr{D}(0,u) && (4.5) \end{aligned}$$

I.e. the validity of the p-th order conditions guarantees that the first $p-1$ derivatives of $\mathscr{D}$ vanish at $t=0$. By Taylor expansion of (4.3) and ignoring higher order terms ,

$$\mathscr{L}(h,u) = \int_0^h l(\tau)d\tau \approx \int_0^h \frac{\tau^p}{p!}\ell^{(p)}(0)d\tau = \frac{\tau^{p+1}}{(p+1)!}\ell^{(p)}(0). \tag{4.6}$$

The notation $\approx$ stands for asymptotic equivalence. Reversing the Taylor expansion gives,

$$\frac{\tau^{p+1}}{(p+1)!}\ell^{(p)}(0) \approx \frac{h}{p+1}\ell(h) = \frac{h}{p+1}\mathscr{D}(h,u). \tag{4.7}$$

This leads to a local error estimator involving a single evaluation of the defect,

$$\mathscr{P}(h,u) = \tfrac{h}{p+1}\mathscr{D}(h,u) \approx \mathscr{L}(h,u)\,. \tag{4.8}$$

This algorithm works generally for splittings of any order into an arbitrary number of operators if Fréchet derivatives of the subflows are available, see [3].

4.1.1 Efficient evaluation of the defect

The defect can be computed simultaneously with the the update of the solution u. Considering a scheme $\mathscr{S}$ with s stages and let $d = \mathscr{D}(h,u)$. Hence algorithms for splitting in 2 , 3 and n operators are given, which can be easily deduced from [8]. Splitting into two operators gives equation (2.3). The sub flows of the operators are given by $\mathscr{E}_A$ and $\mathscr{E}_B$.

```
$d = 0$
for $k = 1 : s$
    $d = d + a_k A(u)$
    $d = \partial_2 \mathscr{E}_A(a_k h, u) \cdot d$
    $u = \mathscr{E}_A(a_k h, u)$
    $d = \partial_2 \mathscr{E}_B(b_k h, u) \cdot d$
    $u = \mathscr{E}_B(b_k h, u)$
    $d = d + \begin{cases} b_k B(u), & k < s \\ (b_k - 1)B(u), & k = s \end{cases}$
end
$d = d - A(u)$
```

In case of splitting (2.1) into three operators,

$$\partial_t u(t) = F(u(t)) = A(u(t)) + B(u(t)) + C(u(t)), \quad t \geq 0. \tag{4.9}$$

The coefficients of splitting scheme are named a_j, b_j and c_j with respect to the name of their operator.

```
$d = 0$
for $k = 1 : s$
    $d = d + a_k A(u)$
    $d = \partial_2 \mathscr{E}_A(a_k h, u) \cdot d$
    $u = \mathscr{E}_A(a_k h, u)$
    $d = d + b_k B(u)$
    $d = \partial_2 \mathscr{E}_B(b_k h, u) \cdot d$
    $u = \mathscr{E}_B(b_k h, u)$
    $d = \partial_2 \mathscr{E}_C(c_k h, u) \cdot d$
    $u = \mathscr{E}_C(c_k h, u)$
    $d = d + \begin{cases} c_k C(u), & k < s \\ (c_k - 1) C(u), & k = s \end{cases}$
end
$d = d - A(u) - B(u)$
```

The general case using n operators (2.1) yields,

$$\partial_t u(t) = F(u(t)) = A_1(u(t)) + A_2(u(t)) + \cdots + A_n(u(t)), \quad t \geq 0. \tag{4.10}$$

One may already guessed the algorithm, but here it is anyway

```
$d = 0$
for $k = 1 : s$
    for $j = 1 : n - 1$
        $d = d + a_{j_k} A_j(u)$
        $d = \partial_2 \mathscr{E}_{A_j}(a_{j_k} h, u) \cdot d$
        $u = \mathscr{E}_{A_j}(a_{j_k} h, u)$
    end
    $d = \partial_2 \mathscr{E}_{A_n}(a_{n_k} h, u) \cdot d$
    $u = \mathscr{E}_{A_n}(a_{n_k} h, u)$
    $d = d + \begin{cases} a_{n_k} A_n(u), & k < s \\ (a_{n_k} - 1) A_n(u), & k = s \end{cases}$
end
$d = d - \sum_{j=1}^{n-1} A_j(u)$
```

4.2 Embedded formulae

Consider a pair of two schemes with orders p, and $p+1$. The scheme $\mathscr{S}$ of order p is called the worker, which propagates the solution in time and the scheme $\hat{\mathscr{S}}$ of order $p+1$ is the controller. An estimate for the local error is given by the distance between the solutions of the different schemes.

$$\mathscr{P}(h,u) = \mathscr{S}(h,u) - \hat{\mathscr{S}}(h,u) \approx \mathscr{L}(h,u) \tag{4.11}$$

For computational efficiency it is desired that most parts of the computation from the worker can be reused for the controller. To setup those pairs choose first an efficient controller of order $p+1$ and $\hat{s}$ stages. Denote the coefficients of the controller $\hat{\mathscr{S}}$ as $\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{\hat{s}}$ and $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_{\hat{s}}$. The worker is only a scheme of order p and therefore does not need to fulfill as many order conditions. This makes it possible to choose the worker $\mathscr{S}$ to have $s \leq \hat{s}$ stages and coefficients $a_1, a_2, \ldots, a_s$ and $b_1, b_2, \ldots, b_s$ where $a_i = \hat{a}_i$ and $b_i = \hat{b}_i$ for the first k coefficients of the scheme. In [9], it is described how to choose a 'good' $\mathscr{S}$ from a set of possible candidates.

4.3 Palindromic schemes

The local error estimator is based on the adjoint scheme $\mathscr{S}^*$ of $\mathscr{S}$ which is satisfies,

$$\mathscr{S}^*(h,u) = \mathscr{S}^{-1}(-h,u). \tag{4.12}$$

The leading error term of $\mathscr{S}$ and its adjoint $\mathscr{S}^*$ are identical up to the factor $(-1)^p$ see [9] and [14],

$$\mathscr{L}(h,u) = \qquad C(u)\,h^{p+1} + \mathscr{O}(h^{p+2}), \tag{4.13}$$

$$\mathscr{L}^*(h,u) = (-1)^p C(u)\,h^{p+1} + \mathscr{O}(h^{p+2}), \tag{4.14}$$

If p is odd, the averaged scheme

$$\overline{\mathscr{S}}(h,u) = \frac{1}{2}(\mathscr{S}(h,u) + \mathscr{S}^*(h,u)), \tag{4.15}$$

is of order $p+1$. The local error estimator is, defined in the same way as for an embedded pair

$$\mathscr{P}(h,u) = \mathscr{S}(h,u) - \overline{\mathscr{S}(h,u)} = \frac{1}{2}(\mathscr{S}(h,u) - \mathscr{S}^*(h,u)). \tag{4.16}$$

Palindromic schemes, or 'reflected schemes' first proposed in [9] in the terminology of [3], are characterized by $b_j = a_{s+1-j}$, $j = 1, \cdots, s$, i.e.,

$$\begin{aligned}&(a_1, b_1, a_2, b_2, \ldots, a_{s-1}, b_{s-1}, a_s, b_s)\\ =\ &(a_1, a_s, a_2, a_{s-1}, \ldots, a_{s-1}, a_2, a_s, a_1).\end{aligned} \tag{4.17}$$

Assume a scheme of order p is given, and consider a single splitting step of the form (2.4). Swapping the roles of A and B, i.e., replacing (2.4) by

$$\hat{\mathscr{S}}(h,u) = \hat{\mathscr{S}}_s(h, \hat{\mathscr{S}}_{s-1}(h, \ldots, \hat{\mathscr{S}}_1(h,u))), \tag{4.18a}$$

with

$$\hat{\mathscr{S}}_j(h,v) = \mathscr{E}_A(b_j h, \mathscr{E}_B(a_j h, v)), \tag{4.18b}$$

also results in a scheme of order p, because no special information about the nature of A or B enters the order conditions. If $\mathscr{S}$ is palindromic then

$$\mathscr{S}(-h, \hat{\mathscr{S}}(h,u)) = u, \quad \text{which gives} \quad \hat{\mathscr{S}}(h,u) = \mathscr{S}^*(h,u). \tag{4.19}$$

The computational costs for the local error estimator are the same as for the basic integrator. Palindromic schemes are not only interesting because, as stated in 3.0.1 the number of order condition is lower, but also because [9] discovered that they tend to have minimal LEMs among sets of equally costly schemes. The reason for this is not know, but would be interesting for further investigations.

4.4 MilneDevice

Consider two schemes $(\mathscr{S}, \hat{\mathscr{S}})$ [9], with the same order p and s stages. Now let the local errors $\mathscr{L}, \hat{\mathscr{L}}$ be related via

$$\mathscr{L}(h,u) = C(u)h^{p+1} + \mathscr{O}(h^{p+2}) \tag{4.20a}$$

$$\hat{\mathscr{L}}(h,u) = \gamma C(u)h^{p+1} + \mathscr{O}(h^{p+2}) \tag{4.20b}$$

with $\gamma \neq 1$. The additive scheme

$$\overline{\mathscr{S}}(h,u) = -\frac{\gamma}{1-\gamma}\mathscr{S}(h,u) + \frac{1}{1-\gamma}\hat{\mathscr{S}}(h,u) \tag{4.21}$$

is of order $p+1$, and a local error estimator is given by

$$\mathscr{P}(h,u) = \mathscr{S}(h,u) - \overline{\mathscr{S}}(h,u) = -\frac{1}{1-\gamma}(\mathscr{S}(h,u) - \hat{\mathscr{S}}(h,u)). \tag{4.22}$$

This is more general than Section 4.3, on the other hand, the construction works in practice only for lower-order schemes, because (4.20b) , (4.20b) is a rather restrictive requirement, which corresponds to $\gamma = -1$

4.5 Time-adaptive strategies

After estimating the local error (4.1) $\mathscr{P}(h,u)$, it is used to choose the step size h. Given a local error tolerance TOL, one may want to maximize step size h. The following formula is used to estimate the optimal step size for the next step (see [15, 22]).

$$h = h_0 \min\left\{\alpha_{\max}, \max\left\{\alpha_{\min}, \alpha\left(\frac{\mathrm{TOL}}{\mathscr{P}(h_0,u)}\right)^{\frac{1}{p+1}}\right\}\right\} \tag{4.23}$$

The notation h_0 is used for the step size currently used. The parameters $\alpha_{\min}$ and $\alpha_{\max}$ limit the change rate of the step size, α is a safety factor to reduce overestimating the step size due to numerical errors. If the estimated error is larger than the given tolerance the step is rejected and a new guess is made with the previously estimated error. In the test the parameters were chosen as,

$$\alpha = 0.9, \quad \alpha_{\min} = 0.25, \quad \alpha_{\max} = 4.0. \tag{4.24}$$

More advanced algorithms, e.g. combining the last two step sizes (h_0, h_{-1}) and estimated errors (4.25) to predict the step size have been investigated but did not show a significant change in efficiency or number of rejected steps.

$$h = h_0 \min\left\{\alpha_{\max}, \max\left\{\alpha_{\min}, \alpha\left(\frac{\mathrm{TOL}}{\mathscr{P}(h_0,u)}\right)^{\frac{\beta_1}{p+1}}\left(\frac{\mathrm{TOL}}{\mathscr{P}(h_{-1},u)}\right)^{\frac{\beta_2}{p+1}}\left(\frac{h_0}{h_{-1}}\right)^{-\alpha_1}\right\}\right\} \tag{4.25}$$

with

$$\alpha = 1.0, \quad \alpha_{\min} = 0.25, \quad \alpha_{\max} = 4.0 \quad \text{and} \quad \alpha_1 = \beta_1 = \beta_2 = 0.25. \tag{4.26}$$

Chapter 5

Commutator free exponential Time propagators / Magnus

For a non-autonomous linear system of differential equations

$$\frac{d}{dt}u(t) = A(t)u(t), \quad t \in [t_0, t_{end}], \quad u(t_0) = u_0 \text{ given,} \tag{5.1}$$

the formal representation of the solution is given by the Magnus expansion [14],

$$u(t_n + \tau_n) = e^{\Omega(t_n+\tau_n)}u(t_n), \quad t_n, t_n + \tau_n \in [t_0, t_{end}] \tag{5.2}$$

$$\Omega(t_n + \tau_n) = \int_{t_n}^{t_n+\tau_n} A(\sigma)d\sigma + \frac{1}{2}\int_{t_n}^{t_n+\tau_n}\int_{t_n}^{\sigma_1}[A(\sigma_1), A(\sigma_2)]d\sigma_2 d\sigma_1 + \tag{5.3}$$

$$\frac{1}{6}\int_{t_n}^{t_n+\tau_n}\int_{t_n}^{\sigma_1}\int_{t_n}^{\sigma_2}[[A(\sigma_1), A(\sigma_2)], A(\sigma_3)] + [A(\sigma_1), [A(\sigma_2), A(\sigma_3)]]d\sigma_3 d\sigma_2 d\sigma_1 + \dots$$

This has been extensively studied in the literature.

5.1 Magnus-type integrators

Truncation of the infinite sum (5.3) and suitable quadrature of the integrals gives the class of Magnus-type integrators,

$$\mathrm{e}^{\Omega(t_n+\tau_n)} \approx \mathrm{e}^{\Omega_n}. \tag{5.4}$$

Second-order Magnus-type integrator (exponential midpoint scheme)

$$\Omega_n \approx \tau_n A\left(t_n + \frac{\tau_n}{2}\right) \tag{5.5}$$

Fourth-order Magnus-type integrator

$$\Omega_n \approx \frac{1}{6}\tau_n \left(A(\tau_n) + 4A\left(t_n + \frac{\tau_n}{2}\right) + A(t_n+\tau_n)\right) - \frac{1}{12}\tau_n^2 [A(t_n), A(t_n+\tau_n)] \quad (5.6)$$

For higher order one gets nasty commutators expressions which may alter the structure of the matrix. This leads to the following alternative approach.

5.2 Commutator free Magnus integrators

To avoid the computational costs of the commutators, commutator free exponential time integrators were proposed [1, 2]. They are constructed as compositions of matrix exponentials of linear combinations of A evaluated at certain times. The scheme is characterized by the order p, the number of evaluation points (nodes) K of A and the number of matrix exponentials J. Therefore we have the vector

$c = \begin{pmatrix} c_1 \\ \vdots \\ c_K \end{pmatrix} \in \mathbb{R}^K$ of evaluation points and the matrix $a = \begin{pmatrix} a_{11} & \dots & a_{1K} \\ \vdots & \ddots & \vdots \\ a_{J1} & \dots & a_{JK} \end{pmatrix} \in \mathbb{R}^{J\times K}$

with the corresponding coefficients.
One step with Magnus form $u_n \approx u(t_n)$ to u_{n+1} with stepsize τ_n

$$\begin{aligned} u_{n+1} &= \mathscr{S}_{t_n}(\tau) u_n \\ \mathscr{S}_{t_n}(\tau) &= \mathrm{e}^{\Omega_n} = \mathrm{e}^{\tau B_J(t_n,\tau)} \cdots \mathrm{e}^{\tau B_1(t_n,\tau)} \\ B_j(t_n,\tau) &= a_{j1} A(t_n + c_1\tau) + \cdots + a_{jK} A(t_n + c_K\tau), \quad j \in \{1,\dots,J\} \end{aligned}$$

From now on assume $t_n = 0$, to shorten the notation.

5.2.1 Local error and defect. Order conditions and defect based local error estimate

The main idea is the same as for splitting, see also (2.5) for the local error, (4.2) for the defect and (4.8) for the local error estimator. The local error and defect are now defined by,

$$\mathscr{L}(\tau) = \mathscr{S}(\tau) - \mathscr{E}(\tau), \quad (5.7)$$

$$\mathscr{D}(\tau) = \frac{d}{d\tau}\mathscr{S}(\tau) - A(t_0+\tau)\mathscr{S}(\tau), \quad (5.8)$$

with the exact flow $\mathscr{E}$ associated with the given equation (5.1). The local error satisfies

$$\frac{d}{d\tau}\mathscr{L}(\tau) = A(\tau)\mathscr{L}(\tau) + \mathscr{D}(\tau), \quad (5.9)$$

and

$$\mathscr{L}(\tau)=\int_0^\tau \Pi(\tau,\sigma)\,\mathscr{D}(\sigma)\,\mathrm{d}\sigma\,, \quad \text{with} \quad \Pi(\tau,\sigma)=\mathscr{E}(\tau)\,\mathscr{E}(-\sigma)\,. \tag{5.10}$$

A scheme of order p is characterized by the property as stated before in (3.5) $\mathscr{L}(\tau)=\mathscr{O}(\tau^{p+1})$, or equivalently $\mathscr{L}(0)=0$ and

$$\tfrac{\mathrm{d}^q}{\mathrm{d}\tau^q}\mathscr{L}(\tau)\big|_{\tau=0}=0\,, \quad q=1,\cdots,p\,. \tag{5.11a}$$

Since from (3.7) and (5.8) we inductively obtain

$$\tfrac{\mathrm{d}^q}{\mathrm{d}\tau^q}\mathscr{L}(\tau)=\sum_{k=0}^{q-1}\tbinom{q-1}{k}\,\tfrac{\mathrm{d}^{q-1-k}}{\mathrm{d}\tau^{q-1-k}}A(\tau)\,\tfrac{\mathrm{d}^k}{\mathrm{d}\tau^k}\mathscr{L}^{(k)}(\tau)+\tfrac{\mathrm{d}^{q-1}}{\mathrm{d}\tau^{q-1}}\mathscr{D}(\tau)\,,$$

and as for the splitting case order conditions (5.11a) are equivalent to

$$\tfrac{\mathrm{d}^{q-1}}{\mathrm{d}\tau^{q-1}}\mathscr{D}(\tau)\big|_{\tau=0}=0\,, \quad q=1,\cdots,p\,, \tag{5.11b}$$

and a scheme of order p is characterized by

$$\mathscr{D}(\tau)=\mathscr{O}(\tau^p)\,. \tag{5.11c}$$

To compute the defect (5.8) for this step one must be able to evaluate

$$\begin{aligned}
\frac{d}{d\tau}\mathscr{S}(\tau) &= \left(\frac{d}{d\tau}e^{\tau B_J(\tau)}\right)e^{\tau B_{J-1}(\tau)}\cdots e^{\tau B_1(\tau)}+\cdots+ && (5.12)\\
&+\ e^{\tau B_J(\tau)}\cdots e^{\tau B_2(\tau)}\left(\frac{d}{d\tau}e^{\tau B_1(\tau)}\right) && (5.13)\\
\frac{d}{d\tau}e^{\tau B_j(\tau)} &= \sum_{m=0}^{\infty}\frac{1}{(m+1)!}ad^m_{\Omega_j(\tau)}\left(\Omega_j'(\tau)\right)e^{\tau B_j(\tau)}. && (5.14)
\end{aligned}$$

with the notation

$$\begin{aligned}
\Omega_j(\tau) &= \tau B_j(\tau),\\
\Omega_j'(\tau) &= B_j(\tau)+\tau B_j'(\tau),\\
B_j'(\tau) &= a_{j1}c_1A'(c_1\tau)+\cdots+a_{jK}c_KA'(c_K\tau),
\end{aligned}$$

$$\begin{aligned}
ad^j_\Omega(X) &= [\Omega, ad^{j-1}_\Omega(X)], \quad j\geq 1, && (5.15)\\
ad^0_\Omega(X) &= X. && (5.16)
\end{aligned}$$

It is not possible to compute the infinite sum (5.14), since we only consider a method of order p, the sum is truncated,

$$\frac{d}{d\tau}e^{\tau B_1(\tau)}\approx\sum_{m=0}^{p-1}\frac{1}{(m+1)!}ad^m_{\Omega_j(\tau)}\left(\Omega_j'(\tau)\right)e^{\tau B_j(\tau)}. \tag{5.17}$$

5.2.2 Efficient evaluation of the defect

For commutator free Magnus integrators it is also possible to evaluate the defect parallel to the propagtaion of the basic solution see 4.1.1 for the splitting case. The notation stays the same, $d = \mathscr{D}(\tau)$. The main computational effort is the evaluation of the matrix exponentials in (5.20) and (5.21).

```
for $j = 1 : J$
    $B = \sum_{k=1}^{K} a_{j,k} A(c_k \tau)$                                   (5.18)
    $B_t = \sum_{k=1}^{K} c_k a_{j,k} A_t(c_k \tau)$                           (5.19)
    $u = \mathrm{e}^{\tau B} u$                                                 (5.20)
    $d = \begin{cases} \sum_{m=1}^{p-1} \frac{1}{(m+1)!} ad_{\tau B}^{m}(B + \tau B_t)u, & j = 1 \\ d + \mathrm{e}^{\tau B} d + \sum_{m=1}^{p-1} \frac{1}{(m+1)!} ad_{\tau B}^{m}(B + \tau B_t)u, & j > 1 \end{cases}$   (5.21)
end
$d = d - A(\tau)u$                                                               (5.22)
```

Define $A_t := \frac{d}{dt} A(t)$. The sum over the commutators (5.15) in the update of the defect is implemented using only matrix vector multiplications. There might be some potential to optimize it, since commutators may be bad for numeric evaluation.

5.3 Splitting off the time variable “ freezing ”

To remove the time dependence of the system matrix $A(t)$ from (5.1), one can also use a splitting approach. A new variable v is introduced, which will represent time and giving the autonomous system,

$$\frac{d}{dt} u(t) = A(v) u(t), \tag{5.23a}$$

$$\frac{d}{dt} v(t) = 1, \tag{5.23b}$$

or equivalently,

$$\frac{d}{dt} \begin{pmatrix} u(t) \\ v(t) \end{pmatrix} = \underbrace{\begin{pmatrix} A(v) & 0 \\ 0 & 0 \end{pmatrix}}_{\mathscr{B}} \begin{pmatrix} u(t) \\ v(t) \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{\mathscr{A}}, \tag{5.24}$$

with initial condition $(u_0,t_0)^T)$. The splitting operators in this section will be written as $\mathscr{A},\mathscr{B}$ as A is already used for the system matrix. Splitting the right hand side in $\mathscr{A}(u,v)=(0,1)^T$ and $\mathscr{B}(u,v)=(A(v)u,0)^T$ simplifies the computation, because now the matrix for the exponential is time independent. For this kind of splitting where one of the operators is simple like $\mathscr{A}$ it might be interesting to find optimized schemes, which exploit the special structure of $\mathscr{A}$.

Every splitting scheme with s stages applied to (5.1), may also be interpreted as a commutator free Magnus integrator (CFMI) from Section 5.2 with $s=K=J$ and

$$c=\left(a_1,a_1+a_2,\dots,\sum_{i=0}^{s}a_i\right)^T \text{ and } a=diag(b_1,\dots,b_s). \tag{5.25}$$

As an example for the splitting scheme `Strang` $p=s=2$ with coefficients $a_1=a_2=\frac{1}{2}$ and $b_1=1,b_2=0$ (5.25) yields

$$c=\left(\frac{1}{2},1\right)^T \text{ and } a=diag(1,0). \tag{5.26}$$

This simplifies to $c=\left(\frac{1}{2}\right)$ and $a=(1)$, which is the same as exponential midpoint rule in (5.5) further named `Magnus2`. To proof that one can always interpret a CFMI as a splitting scheme may be more difficult. This is an interesting question for further investigations.

5.3.1 Solution

We note the initial value of $(u,v)^T=(u_0,v_0)^T$. For operator $\mathscr{A}$ the exact solution is

$$\begin{pmatrix}u(t)\\v(t)\end{pmatrix}=\begin{pmatrix}u_0\\v_0+t\end{pmatrix}, \tag{5.27}$$

therefore for the evaluation of the defect based error estimator from Section 4.1 one needs,

$$\partial_2\mathscr{E}_{\mathscr{A}}\left(t,\begin{pmatrix}u\\v\end{pmatrix}\right)=Id. \tag{5.28}$$

For operator $\mathscr{B}$ the solution is

$$\begin{pmatrix}u(t)\\v(t)\end{pmatrix} = \exp\left(t\begin{pmatrix}A(v)&0\\0&0\end{pmatrix}\right)\begin{pmatrix}u_0\\v_0\end{pmatrix}=\begin{pmatrix}\exp\left(tA(v)\right)u_0\\v_0\end{pmatrix}, \tag{5.29}$$

and for the defect based error estimator

$$\partial_2\mathscr{E}_{\mathscr{B}}\left(t,\begin{pmatrix}u\\v\end{pmatrix}\right)=\begin{pmatrix}\exp\left(tA(v)\right)&\frac{\partial}{\partial v}\exp\left(tA(v)\right)u\\0&1\end{pmatrix}, \tag{5.30}$$

with differential of the matrix exponential

$$\frac{\partial}{\partial v}\exp\left(tA(v)\right)u \quad = \quad \sum_{m=0}^{\infty}\frac{1}{(m+1)!}ad_{tA(v)}^{m}\left(t\frac{\partial}{\partial v}A(v)\right)\exp\left(tA(v)\right)u. \tag{5.31}$$

The infinite sum will be approximated as in (5.17) by a partial sum up to $m = p-1$ if p is the order of the scheme.

Chapter 6

Examples and Results

6.1 Methods used

6.1.1 Magnus

The Magnus schemes are named `Magnus` where as the first number afterwards gives the order of the scheme. `Magnus2` `Magnus4` are schemes with the minimal number of matrix exponentials for their order. For the `Magnus4Optimized` scheme, the local error constant is smaller at cost of an additional matrix exponential, The coefficients for this three are taken from [2]. The two sixth order schemes `Magnus6_4` and `Magnus6_6` which were also considered, are taken from [1]. The number of matrix exponentials can be found in Table 6.3.

6.1.2 Splitting

The splitting schemes used in the comparisons are all from [5] as mentioned earlier in Section 2. Embedded splitting pairs of order *p* are named with the pattern “`EMB p+1/p ...` ”. Similar Palindromic schemes are named in the manner of “`PP p/p+1 ...` ” and Milne pairs have some where the string “`milne` ” inside the name. If the coefficients are complex “ `c` ” will be the last character in the name. The additional tag “ `D` ” is added when the defect based error estimator from Section 4.1 is used with the splitting scheme.

6.1.3 Hardware

The computations where done on a PC running Ubuntu 16.04 with an Intel(R) Core(TM) i7-2600 CPU @3.4 GHz (4 cores) and 16 GB ram. The Julia language is still in development, the provided code was written in Julia version v0.5 .

6.2 Solar cells: Application of splitting and Magnus-type integrators

Modeling oxide solar cells physicist get a time-dependent Hermitian matrix

$$H\colon \mathbb{R} \to \mathbb{C}^{d\times d} \tag{6.1}$$

to describe the movement and interaction of electrons within a Hubbard-type model in solid state physics. The explicit time-dependency originates from an external electric field associated with a photon. The matrix H is Hermitian, though formulating the equation like 5.1 we get

$$\frac{d}{dt}u(t) = A(t)u(t) = -\mathrm{i}H(t)u(t), \quad t \in [t_0, t_{end}], \quad u(t_0) = u_0 \text{ given}, \tag{6.2}$$

with $A(t)$ is anti-Hermitian, which means $a_{ij} = -\overline{a_{ji}}$.

Section 6.2.3 discusses the simple case with 2 electrons, and Section 6.2.4 the more complex case with 8 electrons in a line.

6.2.1 Implementation

The implementation is done in Julia [11] a high-level, high-performance dynamic programming language for technical computing. In Julia you are able to call Python and C code directly, which enables you to acces lots of software libraries directly. The code is setup in such a way that one can easily change the scheme. The main computational effort lies in the evaluation of the matrix exponentials.

6.2.2 Matrix exponential

The system Matrix in 6.2.3 has only dimension 4, for witch the native implemantion in julia is used. In the physically interesting cases ,the system matrix $A(t)$ is a large sparse matrix. The system in 6.2.4 roughly about 0.2% of the 4900^2 elements are non zero. Expokit [24] is used to compute the matrix exponential. Expokit evaluates $\mathrm{e}^{tA}u$ without computing the exponential itself. Expokit is written in FORTRAN, that means a wrapper for the code is needed. H. Hofstätter provided the interface to call Expokit directly from Julia. We choose Expokit over the native implementation, because its enormous performance increase for sparse matrices of about 5700 compared to the native implementation of expm in Julia, see Table 6.2. This allows to complete the computations in an adequate time. Also in the full case the performance increase is significant as Table 6.1 reports. In those Tables "time" means the time to compute the matrix exponential of either a random matrix in $\mathbb{C}^{4900\times 4900}$ or (6.6) and apply it to a vector in $\mathbb{C}^{4900}$ measured in seconds. "err" is the L_2 norm of the difference between initial data and computing one step forward and backward with step size $h = 0.1$ The native implemantion in julia relies on the scaling and squaring algorithm from [16] to compute first the exponential and then uses a matrix vector multiplication.

	time	error
Julia's expm	95.965	1.711e-15
Expokit	1.022	2.984e-08

Table 6.1: full matrix ($n = 4900$)

	time	error
Julia's expm	130.001	2.002e-15
Expokit	0.022	2.184e-09

Table 6.2: sparse matrix ($n = 4900$)

From Tables 6.1 and Table 6.2 one can easily see the major disadvantage of Expokit, it is that it can only guarantee accuracy of $\sqrt{eps}$, where $eps \approx 1e-16$ is the machine epsilon. This demands further investigations to improve this.

As stated before in 6.2.1 the main computational effort goes into the computation of the matrix exponentials. That means for the splitting schemes A and B are chosen that the number of matrix exponentials is minimal. Following Table 6.3 shows how many times the matrix exponential has to be computed per step. The Magnus schemes are from [1, 2] and the others from [5]. The error estimator for Magnus schemes is always the defect based one, which generally takes $2s-1$ evaluations. One may determine that for embedded schemes the additional evaluations for the error estimator are smaller than for the defect based ones.

6.2.3 Results (2 electrons)

In this simple case, for 2 electrons and 2 states the Hamiltonian $H \in \mathbb{C}^{4\times 4}$ reads,

$$H(t) = \begin{pmatrix} v_{11}+v_{22} & -v_{12}(t) & -v_{21}(t) & 0 \\ -v_{12}(t)* & 2v_{11}+U & 0 & v_{21}(t) \\ -v_{21}(t)* & 0 & 2v_{22}+U & v_{12}(t) \\ 0 & v_{21}(t)* & v_{12}(t)* & v_{11}+v_{22} \end{pmatrix} \quad * \ldots \text{complex comjungate.}$$

Method	# matrix exponentials without errorestimator	# matrix exponentials with errorestimator
`Magnus2`	1	1
`Magnus4`	2	3
`Magnus4Optimized`	3	5
`Magnus6_4`	4	7
`Magnus6_6`	6	11
`BM 11-6 PRK D`	10	19
`A 10-6 D`	9	17
`Symm-Milne-32`	2	4
`EMB 4/3 AK p`	5	8
`EMB 4/3 M/AK`	5	7
`EMB 5/4 AK (ii)`	8	11
`EMB 4/3 AK s`	5	7
`PP 3/4 A`	3	6
`PP 5/6 A`	8	16

Table 6.3: Number of matrix exponential computed per step.

The initial condition is choosen $u_0 = (1,0,0,0)^T$ and the other parameters are given by,

$$v_{11} = 0, \quad v_{22} = 1, \quad v_{12}(t) = e^{\mathrm{i}\omega(t)}, \quad v_{21}(t) = e^{-\mathrm{i}\omega(t)}, \tag{6.3a}$$

$$\omega(t) = \frac{1}{10} e^{-\frac{1}{6}(t-6)^2} \cos\left(\frac{7\pi}{4}(t-6)\right) \quad \text{and } U = 3. \tag{6.3b}$$

The interesting time for this simulation is between $t_0 = 0.0$ and $t_{end} = 8.0$.

Reference solution

Starting with a step size of $t_0 - t_{end}$ the step size is halved every iteration, as long as the distance between the solution at t_{end} and the one computed with the previous step size is decreasing. For this purpose `PP 3/4 A` was used. It took 15 refinements to the final step size, $2.441e-04$ with the minimal distance of $1.735e-12$ to the previously computed one. Also a the values of the solution at $t_0 + (t_{end} - t_0) \cdot 2^{-i}, i \in 1,2,\ldots,15$ are computed to be used as a reference solution for estimating the local error and its orders. The following tables show the results.

Tables

First we present some tables to show general, that the schemes have the predicted order. Then second order methods are compared to determine their efficiency and later on 4-th order methods are compared.

- Tables 6.4 and 6.5 show the global error at the end of the integration interval. The used step size is in the column h, err is the L_2 norm of the error and p is the experimental order, which is computed by the logarithm of the quotient from the error of two consecutive step sizes.

	h	err	p
1	$8.000e+00$	$1.030e-01$	
2	$4.000e+00$	$1.328e-01$	-0.37
3	$2.000e+00$	$1.179e-01$	0.17
4	$1.000e+00$	$8.571e-02$	0.46
5	$5.000e-01$	$3.764e-02$	1.19
6	$2.500e-01$	$8.160e-03$	2.21
7	$1.250e-01$	$2.065e-03$	1.98
8	$6.250e-02$	$5.180e-04$	2.00
9	$3.125e-02$	$1.296e-04$	2.00
10	$1.563e-02$	$3.241e-05$	2.00
11	$7.813e-03$	$8.103e-06$	2.00
12	$3.906e-03$	$2.026e-06$	2.00
13	$1.953e-03$	$5.064e-07$	2.00
14	$9.766e-04$	$1.266e-07$	2.00
15	$4.883e-04$	$3.165e-08$	2.00
16	$2.441e-04$	$7.914e-09$	2.00

Table 6.4: Global Error at $t = 8.0$; `Magnus2`

	h	err	p
1	$8.000e+00$	$9.939e-02$	
2	$4.000e+00$	$1.209e-01$	-0.28
3	$2.000e+00$	$1.043e-01$	0.21
4	$1.000e+00$	$4.252e-02$	1.29
5	$5.000e-01$	$7.685e-03$	2.47
6	$2.500e-01$	$2.362e-03$	1.70
7	$1.250e-01$	$5.951e-04$	1.99
8	$6.250e-02$	$1.490e-04$	2.00
9	$3.125e-02$	$3.728e-05$	2.00
10	$1.563e-02$	$9.320e-06$	2.00
11	$7.813e-03$	$2.330e-06$	2.00
12	$3.906e-03$	$5.825e-07$	2.00
13	$1.953e-03$	$1.456e-07$	2.00
14	$9.766e-04$	$3.641e-08$	2.00
15	$4.883e-04$	$9.103e-09$	2.00
16	$2.441e-04$	$2.277e-09$	2.00

Table 6.5: Global Error at $t = 8.0$; `Symm-Milne-32`

- In Table 6.6 the properties of the local error can be seen. Collumn err1 shows the local error in L_2 norm and err2 the error of the defect based error estimator also in the L_2 norm, which is the same as the error of the by the error estimator corrected solution. The orders are the ones one would expect for `Strang` $p = 2$, $p+1$ and $p+2$.

	h	err1	p1	err2	p2
1	$8.000e+00$	$1.030e-01$		$5.653e+01$	
2	$4.000e+00$	$2.357e-02$	2.13	$2.407e-01$	7.88
3	$2.000e+00$	$2.372e-03$	3.31	$1.601e-02$	3.91
4	$1.000e+00$	$4.896e-04$	2.28	$1.313e-03$	3.61
5	$5.000e-01$	$6.205e-05$	2.98	$8.883e-05$	3.89
6	$2.500e-01$	$9.476e-06$	2.71	$7.760e-06$	3.52
7	$1.250e-01$	$1.202e-06$	2.98	$4.605e-07$	4.07
8	$6.250e-02$	$1.495e-07$	3.01	$2.654e-08$	4.12
9	$3.125e-02$	$1.856e-08$	3.01	$1.572e-09$	4.08
10	$1.563e-02$	$2.307e-09$	3.01	$9.534e-11$	4.04
11	$7.813e-03$	$2.875e-10$	3.00	$5.865e-12$	4.02
12	$3.906e-03$	$3.588e-11$	3.00	$3.636e-13$	4.01
13	$1.953e-03$	$4.481e-12$	3.00	$2.273e-14$	4.00
14	$9.766e-04$	$5.599e-13$	3.00	$1.516e-15$	3.91
15	$4.883e-04$	$6.998e-14$	3.00	$1.415e-16$	3.42
16	$2.441e-04$	$8.749e-15$	3.00	$2.221e-16$	-0.65
17	$1.221e-04$	$1.099e-15$	2.99	$1.110e-16$	1.00

Table 6.6: Local error and error of the defect based error estimator for `Magnus2`

- In Table 6.7 the Methods `Magnus2` and `Magnus2` are compared, using equidistant time stepping for different step sizes. It is shown that `Magnus2` is approximately twice as fast as `Symm-Milne-32`, because as stated in 6.3 the number of matrix exponentials is twice. The time is averaged over 100 runs and the unit is in seconds. The error for `Symm-Milne-32` is much smaller for the same step size, because of a much smaller local error constant, `Symm-Milne-32` has 0.05 and `Magnus2` has 0.6.

Method	h	err	time
`Magnus2`	$1e-01$	$1.324e-03$	0.005
`Symm-Milne-32`	$1e-01$	$3.812e-04$	0.004
`Magnus2`	$1e-02$	$1.328e-05$	0.015
`Symm-Milne-32`	$1e-02$	$3.818e-06$	0.029
`Magnus2`	$1e-03$	$1.328e-07$	0.141
`Symm-Milne-32`	$1e-03$	$3.818e-08$	0.277
`Magnus2`	$1e-04$	$1.335e-09$	1.383
`Symm-Milne-32`	$1e-04$	$3.889e-10$	2.785

Table 6.7: Global error at $t = 8.0$ for equidistant time-stepping, various stepsizes

- The costs for the error estimator vary significantly as Table 6.8 shows. For `Magnus2` there is no additional matrix exponential to be evaluated, whereas the error estimator for `Symm-Milne-32` is as expensive as the basic integrator.

Method	10000 steps without err est	10000 steps with err est
`Magnus2`	0.175	0.300
`Symm-Milne-32`	0.345	0.728

Table 6.8: Effort for the error estimator

- To determine if adaptive step size selection is faster than without, Table 6.9 shows that for `Magnus2` it is a draw, but for `Symm-Milne-32` it is definitely not. It uses about half the steps as needed for `Magnus2`. The columns "# steps adap" and "# steps equi" give the number of computed time steps, for the adaptive time stepping and the equidistant time stepping. The last two columns "time adap" and "time equi" show the computation time measured in seconds for both variants.

Method	# steps adap	# steps equi	time adap	time equi
TOL = 10^{-8}				
`Magnus2`	1613	2487	0.046	0.046
`Symm-Milne-32`	757	1195	0.059	0.044
TOL = 10^{-10}				
`Magnus2`	7490	11544	0.204	0.204
`Symm-Milne-32`	3515	5523	0.259	0.194
TOL = 10^{-12}				
`Magnus2`	34772	53581	0.947	0.986
`Symm-Milne-32`	16322	25637	1.189	1.036

Table 6.9: Comparison of the efficiency of `Strang` (`Magnus2`) and `Symm-Milne-32`. The tolerances were 10^{-8} (top), 10^{-10} (middle) and 10^{-12} (bottom), respectively. For equidistant time the minimal stepsize was used

- In Table 6.10 it is shown that `Magnus4Optimized` has the lowest error constant and `Magnus4` is the fastest. Step sizes for this methods smaller than 10^{-3} seem not to be beneficial as the global error increases again due to the effect of rounding error.

Method	h	err	time
`Magnus4`	$1e-01$	$3.300e-06$	0.004
`Magnus4Optimized`	$1e-01$	$1.525e-07$	0.006
`EMB 4/3 AK s`	$1e-01$	$3.395e-06$	0.009
`EMB 4/3 M/AK`	$1e-01$	$8.947e-07$	0.010
`Magnus4`	$1e-02$	$3.336e-10$	0.035
`Magnus4Optimized`	$1e-02$	$1.478e-11$	0.057
`EMB 4/3 AK s`	$1e-02$	$3.424e-10$	0.075
`EMB 4/3 M/AK`	$1e-02$	$9.100e-11$	0.079
`Magnus4`	$1e-03$	$3.317e-12$	0.369
`Magnus4Optimized`	$1e-03$	$3.125e-12$	0.609
`EMB 4/3 AK s`	$1e-03$	$7.132e-12$	0.663
`EMB 4/3 M/AK`	$1e-03$	$4.356e-12$	0.840
`Magnus4`	$1e-04$	$9.105e-12$	3.832
`Magnus4Optimized`	$1e-04$	$2.192e-11$	5.944
`EMB 4/3 AK s`	$1e-04$	$3.859e-11$	5.734
`EMB 4/3 M/AK`	$1e-04$	$2.683e-11$	7.007

Table 6.10: Global error at $t = 8.0$ for equidistant time-stepping, various stepsizes

- To compare the efficiency of the error estimator look at Tabel 6.11, where you can see that the defect based error estimator for the Magnus schemes cost more than the basic integrator itself, for the embedded splitting scheme the cost for the error estimator are only 40% of the basic integrator.

Method	10000 steps without err est	10000 steps with err est
`Magnus4`	0.384	0.842
`Magnus4Optimized`	0.616	1.579
`EMB 5/4 AK (ii)`	1.369	1.911

Table 6.11: Effort for the error estimator

- Table 6.12 shows that for all Magnus schemes the adaptive time step selection is slower than computing with a fixed step size. The `EMB 5/4 AK (ii)` is for larger tolerances slightly better, and for smaller tolerances it could be called a tie. The fasted method over all is the `Magnus4Optimized`, which also needs fewest steps. The number of rejected steps (not displayed) is up to 4 because in the beginning the initial step size given was to big.

Method	# steps adap	# steps equi	time adap	time equi
TOL = 10^{-8}				
`Magnus4`	175	233	0.045	0.020
`Magnus4Optimized`	74	99	0.036	0.008
`EMB 5/4 AK (ii)`	93	128	0.024	0.033
TOL = 10^{-10}				
`Magnus4`	440	585	0.086	0.023
`Magnus4Optimized`	172	240	0.033	0.014
`EMB 5/4 AK (ii)`	227	321	0.053	0.045
TOL = 10^{-12}				
`Magnus4`	1105	1469	0.098	0.057
`Magnus4Optimized`	422	599	0.070	0.037
`EMB 5/4 AK (ii)`	562	807	0.115	0.112

Table 6.12: Comparison of the efficiency of `Magnus4`, `Magnus4Optimized` and `Emb 5/4 AK (ii)` The tolerances were 10^{-8} (top), 10^{-10} (middle) and 10^{-12} (bottom), respectively. For equidistant time the minimal stepsize was used

6.2.4 Results (8 electrons)

Now for 8 electrons in a line the system matrix $H \in \mathbb{C}^{4900\times 4900}$ is given by following definition,

$$
\begin{aligned}
f(t) &= \cos\left(a e^{\left(\frac{-(t-t_0)^2)}{2\sigma^2}\right)}(\cos(\omega(t-t_0)) - \cos(\omega(-t_0)))\right) - 1 \qquad (6.4)\\
g(t) &= \sin\left(a e^{\left(\frac{-(t-t_0)^2}{2\sigma^2}\right)}(\cos(\omega(t-t_0)) - \cos(\omega(-t_0)))\right) \qquad (6.5)\\
H(t) &= H_{\text{diag}} + H_{\text{off}_1} + f(t)\cdot H_{\text{off}_1} + \mathrm{i}\cdot g(t)\cdot H_{\text{off}_2} \qquad (6.6)
\end{aligned}
$$

where H_{diag} is a diagonal matrix H_{off_1} is symmetric and H_{off_2} is skew symmetric. As a result $H(t)$ is also hermitian. The scalar parameters for f and g are given by,

$$U = 5, \quad a = 2, \quad \omega = 7, \quad t_0 = 2, \quad \sigma = 1. \qquad (6.7)$$

Reference Solution

Starting with a step size of $t_0 - t_{end}$ the step size is halved every iteration, as long as the distance between the solution at t_{end} and the one computed with the previous step size is decreasing. In this case `PP 56 A` was used. It took 11 refinements to the final step size, $7.813e-3$ with the minimal distance of $1.870e-8$ to the previously computed one. Also a the values of the solution at $t_0 + (t_{end} - t_0) \cdot 2^{-i}, i \in 1, 2, \ldots, 11$ are computed to be used as a reference solution for estimating the local error and its orders.

Tables

The tables in this section are similar to them shown in 6.2.3, but for a much larger system. First there are some tables to show general that the splitting works, eq that the schemes have the predicted order, than second order methods are compared to determine their efficiency and later on 4-th order methods are compared.

- Tables 6.13, 6.14 and 6.15 show the global error, with respect to the reference solution at the end of the integration interval. The used step size is in the column h, err is the error and p is the experimental order, which is computed by the logarithm of the quotient from the error of two consecutive step sizes. For `Emb 5/4 A k (ii)` the global order is barley visibly because the step size region before the numerical rounding errors is very small.

	h	err	p
1	$8.000e+00$	$1.058e+00$	
2	$4.000e+00$	$1.358e+00$	-0.36
3	$2.000e+00$	$1.167e+00$	0.22
4	$1.000e+00$	$1.407e+00$	-0.27
5	$5.000e-01$	$1.380e+00$	0.03
6	$2.500e-01$	$6.350e-01$	1.12
7	$1.250e-01$	$1.894e-01$	1.75
8	$6.250e-02$	$4.847e-02$	1.97
9	$3.125e-02$	$1.217e-02$	1.99
10	$1.563e-02$	$3.044e-03$	2.00
11	$7.813e-03$	$7.593e-04$	2.00
12	$3.906e-03$	$1.881e-04$	2.01

Table 6.13: Global Error at $t = 8.0$; `Magnus2`

	h	err	p
1	$8.000e+00$	$1.417e+00$	
2	$4.000e+00$	$1.276e+00$	0.15
3	$2.000e+00$	$1.405e+00$	-0.14
4	$1.000e+00$	$1.383e+00$	0.02
5	$5.000e-01$	$9.244e-01$	0.58
6	$2.500e-01$	$3.767e-01$	1.30
7	$1.250e-01$	$5.665e-02$	2.73
8	$6.250e-02$	$1.423e-02$	1.99
9	$3.125e-02$	$3.560e-03$	2.00
10	$1.563e-02$	$8.885e-04$	2.00
11	$7.813e-03$	$2.203e-04$	2.01
12	$3.906e-03$	$5.328e-05$	2.05

Table 6.14: Global Error at $t = 8.0$; `Symm-Milne-32`

	h	err	p
1	$8.000e+00$	$1.082e+00$	
2	$4.000e+00$	$1.458e+00$	-0.43
3	$2.000e+00$	$1.367e+00$	0.09
4	$1.000e+00$	$1.257e+00$	0.12
5	$5.000e-01$	$5.125e-01$	1.29
6	$2.500e-01$	$5.653e-02$	3.18
7	$1.250e-01$	$4.152e-04$	7.09
8	$6.250e-02$	$3.194e-05$	3.70
9	$3.125e-02$	$8.233e-06$	1.96
10	$1.563e-02$	$6.869e-06$	0.26
11	$7.813e-03$	$6.798e-06$	0.02
12	$3.906e-03$	$6.789e-06$	0.00

Table 6.15: Global Error at $t = 8.0$; `EMB 5/4 AK (ii)`

- In Table 6.16 the properties of the local error can be seen. Collumn err1 shows the local error and err2 the error of the defect based error estimator, which is the error of the by the error estimator corrected solution. The orders are the ones one would expect for `Strang` $p = 2$, $p+1$ and $p+2$.

	h	err1	p1	err2	p2
1	$8.000e+00$	$1.058e+00$		$1.666e+03$	
2	$4.000e+00$	$1.368e+00$	-0.37	$1.007e+01$	7.37
3	$2.000e+00$	$1.379e+00$	-0.01	$8.609e+01$	-3.10
4	$1.000e+00$	$9.787e-01$	0.49	$9.053e+00$	3.25
5	$5.000e-01$	$2.639e-01$	1.89	$4.638e-01$	4.29
6	$2.500e-01$	$4.377e-02$	2.59	$5.061e-02$	3.20
7	$1.250e-01$	$6.941e-03$	2.66	$3.533e-03$	3.84
8	$6.250e-02$	$8.973e-04$	2.95	$2.139e-04$	4.05
9	$3.125e-02$	$1.110e-04$	3.02	$1.299e-05$	4.04
10	$1.563e-02$	$1.370e-05$	3.02	$7.990e-07$	4.02
11	$7.813e-03$	$1.699e-06$	3.01	$5.003e-08$	4.00
12	$3.906e-03$	$2.111e-07$	3.01	$3.971e-09$	3.66

Table 6.16: Local error and error of the defect based error estimator for `Magnus2`

- In the Table 6.17 it is shown that `Magnus2` is approximately twice as fast as `Symm-Milne-32`, because as stated in 6.3 the number of matrix exponentials is twice. The error for `Symm-Milne-32` is much smaller for the same step size.

Method	h	error	time
`Magnus2`	$1e-01$	$1.228e-01$	1.931
`Symm-Milne-32`	$1e-01$	$3.630e-02$	3.741
`Magnus2`	$1e-02$	$1.244e-03$	18.223
`Symm-Milne-32`	$1e-02$	$3.607e-04$	36.254
`Magnus2`	$1e-03$	$1.393e-05$	184.817
`Symm-Milne-32`	$1e-03$	$5.180e-06$	363.343

Table 6.17: Global error at $t = 8.0$ for equidistant time-stepping, various stepsizes

- The costs for the error estimator vary dramatically as Table 6.18 shows. For `Magnus2` there is no additional matrix exponential to be evaluated, whereas the error estimator for `Symm-Milne-32` is as expensive as the basic integrator.

Method	1000 steps without err est	1000 steps with err est
`Magnus2`	22.762	27.459
`Symm-Milne-32`	46.037	90.545

Table 6.18: Effort for the error estimator

- The main question is if the computation with error estimating and adaptive step size selection is faster than without. Table 6.19 gives the answer for `Magnus2` this is correct with an speed up of a factor 2, but for `Symm-Milne-32` it is not. It uses about half the steps as needed for `Magnus2`.

Method	# steps adap	# steps equi	time adap	time equi
TOL = 10^{-4}				
`Magnus2`	275	705	7.928	16.335
`Symm-Milne-32`	147	360	15.300	16.505
TOL = 10^{-6}				
`Magnus2`	1263	3269	35.013	74.313
`Symm-Milne-32`	618	1637	57.627	74.398
TOL = 10^{-8}				
`Magnus2`	5846	15166	161.191	345.653
`Symm-Milne-32`	2917	7701	280.121	352.199

Table 6.19: Comparison of the efficiency of `Strang` (`Magnus2`) and `Symm-Milne-32` The tolerances were 10^{-4} (top), 10^{-6} (middle) and 10^{-8} (bottom), respectively. For equidistant time the minimal stepsize was used

- Figure 6.1 shows the adaptive chosen step size for `Magnus2`. If the step size is chosen too small, in this case five orders of magnitude wrong, the adaptive selection algorithm nearly instantly corrects this. The wrong initial step size gives you a total of additional 9 steps to compute. The minimal step size occurs around $t = 2$, because there the changes of the matrix are the strongest in time.

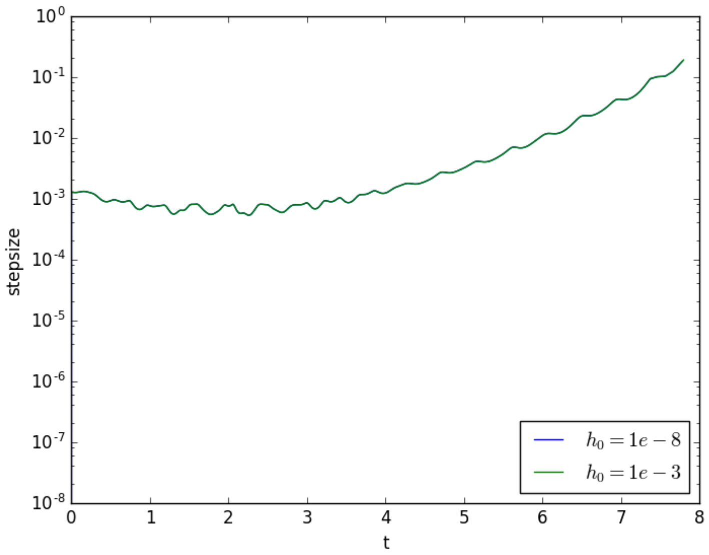

Figure 6.1: Step size over time for `Magnus2` with TOL = 10^{-8} with different initial step sizes.

- In the Table 6.20 it is shown that `Magnus4Optimized` has the lowest error constant and `Magnus4` is the fastest, but due to the numerical inaccuracy all compared methods have the same error for step sizes smaller than 0.01.

Method	h	err	time
`Magnus4`	$1e-01$	$5.469e-04$	4.165
`Magnus4Optimized`	$1e-01$	$4.153e-05$	6.669
`EMB 4/3 AK s`	$1e-01$	$4.626e-04$	8.165
`EMB 4/3 M/AK`	$1e-01$	$1.236e-04$	9.376
`Magnus4`	$1e-02$	$4.730e-06$	40.363
`Magnus4Optimized`	$1e-02$	$4.731e-06$	66.643
`EMB 4/3 AK s`	$1e-02$	$4.736e-06$	74.517
`EMB 4/3 M/AK`	$1e-02$	$4.774e-06$	91.337
`Magnus4`	$1e-03$	$1.587e-06$	428.143
`Magnus4Optimized`	$1e-03$	$1.638e-06$	699.416
`EMB 4/3 AK s`	$1e-03$	$1.582e-06$	728.632
`EMB 4/3 M/AK`	$1e-03$	$6.338e-06$	909.764

Table 6.20: Global error at $t = 8.0$ for equidistant time-stepping, various stepsizes.

- Table 6.21 shows that for the Magnus-type schemes the evaluation of the sum of commutators already has not negligible costs. The embedded scheme proofs again the the costs of the error estimator are cheap compared to the other methods.

Method	1000 steps without err est	1000 steps with err est
`Magnus4`	51.934	102.975
`Magnus4Optimized`	85.633	200.748
`EMB 5/4 AK (ii)`	177.413	244.238

Table 6.21: Effort for the error estimator

- Table 6.22 shows that for `Magnus4` and `EMB 5/4 AK (ii)` it is a tie or doing adaptive time stepping is slightly better. Though the fasted method over all is the `Magnus4Optimized`. Also worth wile to mention is that for tolerance 10^{-8} the number of steps used in `EMB 5/4 AK (ii)` is significant higher than before, this is probably the result that the error estimator is the difference of two solution, which are limited in their accuracy by $\sqrt{eps}$. This leads to many rejections of steps. For the both Magnus schemes the defect error estimator which is scaled by the step size still gives an accurate estimate.

Method	# steps adap	# steps equi	time adap	time equi
TOL = 10^{-4}				
`Magnus4`	70	148	8.138	7.425
`Magnus4Optimized`	39	77	10.639	6.365
`EMB 5/4 AK (ii)`	35	67	15.556	13.181
TOL = 10^{-6}				
`Magnus4`	172	375	18.017	18.930
`Magnus4Optimized`	76	156	17.634	13.325
`EMB 5/4 AK (ii)`	84	167	27.678	29.217
TOL = 10^{-8}				
`Magnus4`	430	952	43.826	47.752
`Magnus4Optimized`	169	345	33.952	28.826
`EMB 5/4 AK (ii)`	658	1382	224.973	245.597

Table 6.22: Comparison of the efficiency of `Magnus4`, `Magnus4Optimized` and `Emb 5/4 AK (ii)` The tolerances were 10^{-4} (top), 10^{-6} (middle) and 10^{-8} (bottom), respectively. For equidistant time the minimal stepsize was used. Note that for TOL=10^{-8} the accuracy of the matrix exponetial is the limiting factor.

6.3 Gray-Scott equation

As a parabolic example, we study the Gray-Scott system (see [13]) modeling a two-component reaction-diffusion process, in two, $\mathbf{x} \in \mathbb{R}^2$, or three, $\mathbf{x} \in \mathbb{R}^3$, dimensions

$$\partial_t u(\mathbf{x},t) = c_u \Delta u(\mathbf{x},t) - u(\mathbf{x},t)\, v^2(\mathbf{x},t) + \alpha(1 - u(\mathbf{x},t)), \tag{6.8a}$$

$$\partial_t v(\mathbf{x},t) = c_v \Delta v(\mathbf{x},t) + u(\mathbf{x},t)\, v^2(\mathbf{x},t) - \beta\, v(\mathbf{x},t). \tag{6.8b}$$

The nonlinear term describes the reaction $u + 2v \to 3v$ of two chemical components which are described by $(u(\mathbf{x},t), v(\mathbf{x},t))$. α is the rate of fresh u added and β is the amount of v removed. This has beed discussed in [10]. For this model it is natural to have periodic boundary conditions. The system is a model for pattern formation with dynamical behavior. Splitting only in two components (6.9) will not allow to solve the nonlinear term B exactly,

$$\underbrace{\begin{pmatrix} c_u\Delta - \alpha & 0 \\ 0 & c_v\Delta - \beta \end{pmatrix} U(\mathbf{x},t) + \begin{pmatrix} \alpha \\ 0 \end{pmatrix}}_{=A} + \underbrace{\begin{pmatrix} -u(\mathbf{x},t)v^2(\mathbf{x},t) \\ u(\mathbf{x},t)v^2(\mathbf{x},t) \end{pmatrix}}_{=B}. \tag{6.9}$$

The implementation solves this by using numerical integration for, example a Runge–Kutta integrator. In Table 6.23 the global error and experimental order for `Strang` is shown.

	h	err	p
1	$1.000e+00$	$1.831e-03$	1.77
2	$5.000e-01$	$5.363e-04$	1.94
3	$2.500e-01$	$1.393e-04$	1.98
4	$1.250e-01$	$3.518e-05$	1.99
5	$6.250e-02$	$8.817e-06$	1.99
6	$3.125e-02$	$2.205e-06$	1.99
7	$1.563e-02$	$5.515e-07$	1.99
8	$7.813e-03$	$1.378e-07$	2.00
9	$3.906e-03$	$3.446e-08$	2.00
10	$1.953e-03$	$8.613e-09$	2.00
11	$9.765e-04$	$2.145e-09$	2.04

Table 6.23: Global error at $t = 1.0$; `Strang`

Though splitting into 3 operators according to

$$\underbrace{\begin{pmatrix} c_u\Delta-\alpha & 0 \\ 0 & c_v\Delta-\beta \end{pmatrix} U(\mathbf{x},t)+\begin{pmatrix} \alpha \\ 0 \end{pmatrix}}_{=A}+\underbrace{\begin{pmatrix} 0 \\ u(\mathbf{x},t)v^2(\mathbf{x},t) \end{pmatrix}}_{=B}-\underbrace{\begin{pmatrix} u(\mathbf{x},t)v^2(\mathbf{x},t) \\ 0 \end{pmatrix}}_{=C}.$$

allows all the subflows to be integrated exactly, but the computation time is higher [10]. For example compare Table 6.23 with 6.24, this shows that the error for the same step size is approximately twice as big. This also corresponds to the computed LEM see [5] for both of this schemes, the value of the LEM for `Strang` is 0.6 and `Strang ABC` has LEM 1.5.

	h	err	p
1	$1.000e+00$	$3.401e-03$	2.00
2	$5.000e-01$	$8.502e-04$	2.00
3	$2.500e-01$	$2.125e-04$	2.00
4	$1.250e-01$	$5.313e-05$	2.00
5	$6.250e-02$	$1.328e-05$	2.00
6	$3.125e-02$	$3.321e-06$	1.99
7	$1.563e-02$	$8.302e-07$	1.99
8	$7.813e-03$	$2.075e-07$	1.99
9	$3.906e-03$	$5.189e-08$	1.99
10	$1.953e-03$	$1.297e-08$	1.99
11	$9.766e-04$	$3.250e-09$	1.97

Table 6.24: Global error at $t = 1.0$; `Strang ABC`

See Table 6.25 where `PP 3/4 A 3 c` is the only *ABC*-splitting scheme. This table also shows that a second order scheme needs a lot more steps than a third order scheme. The fewest steps were made by the *ABC*-splitting scheme. The fastest for the larger tolerance was the equidistant `Emb 4/3 A c` and for the smaller tolerance equidistant `PP 3/4 A c`. Time adaptivity is of no use in this simulation as the solution has rapid changes all the time on different places, which will not allow larger bigger step sizes in time.

Method	# steps adap	# steps equi	time adap	time equi
TOL = 10^{-5}				
`Milne 2/2 c (i),`	406	486	57.04	28.21
`Emb 4/3 A c,`	67	79	17.72	11.46
`PP 3/4 A c,`	116	135	23.02	12.99
`PP 3/4 A 3 c,`	65	67	26.74	13.23
TOL = 10^{-8}				
`Milne 2/2 c (i),`	4691	5625	878.72	503.93
`Emb 4/3 A c,`	516	612	174.30	128.19
`PP 3/4 A c,`	929	1107	195.87	106.79
`PP 3/4 A 3 c,`	555	645	244.41	138.38

Table 6.25: Comparison of the efficiency of `Milne 2/2 c (i)`, `Emb 4/3 A c`, `PP 3/4 A c` and `PP 3/4 A 3 c` for (6.8). The tolerances were 10^{-5} (top) and 10^{-8} (bottom), respectively.

For parabolic equations it is essential that the splitting coefficients have non-negative real part. This leads for higher order schemes $p > 2$ to complex coefficients. Coefficients from [5] can be identified to have complex entries by " c " at the end of their name. The full error analysis was done in [10].

6.3.1 Implementation

The code is written in FORTRAN, using MPI library and 2DECOMP&FFT [21], a library to distribute the data efficient to multiple nodes (computers) and compute the FFT. This allows the code to be run on large clusters as the Vienna Scientific Cluster (VSC) [27].

This high performance computing (hpc) system has 1314 nodes with 2 AMD Opteron "Magny Cours 6132HE" with 8 cores each running at 2,2 GHz. The memory of one node is 8 x 4 GB ECC DDR3 RAM and an 16 GB SSD. The system is connected by 2 x Gigabit Ethernet LAN and 1 x Infiniband-QDR. The operating system is Scientific Linux 6.0 with Intel MPI and Open MPI. Figure 6.2 shows that if double the number of cores where used to run the simulation the time is nearly halved, which would be the ideal case, dotted line. See section 6.3.2 for the 3-dimensional case to get the used parameters.

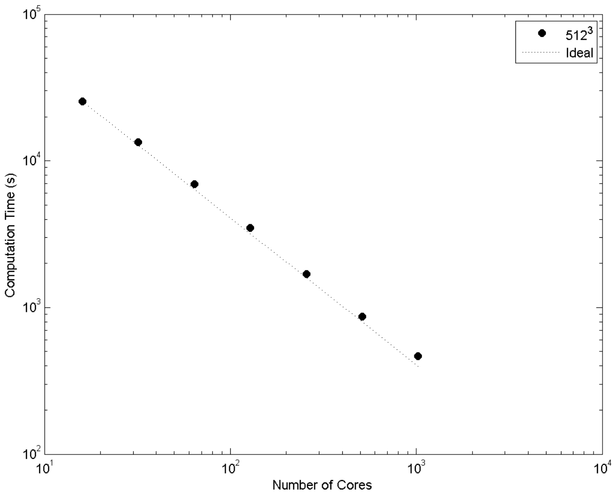

Figure 6.2: Scaling the number of cores.

To generate the pictures shown in 6.3.2 ParaView 4.1 [25] was used. This is a multi functional software to analyze and generate pictures from scientific data.

With this tool it is possible to use co-processing to render the visualization in situ without the need to write the raw data to disc, which can be very time consuming.

6.3.2 Visual results

The free parameters in (6.8) are chosen as

$$c_u = 0.04, \quad c_v = 0.005, \quad \alpha = 0.038, \quad \beta = 0.076. \tag{6.10}$$

2 dimensions

In 2-dimension $\mathbf{x} = (x,y) \in [-4\pi, 4\pi]^2$ we prescribe the initial condition as

$$u(x,y,0) = 0.5 + \exp(-1-(x^2+y^2)), \quad v(x,y,0) = 0.1 + \exp(-1-(x^2+y^2)). \tag{6.11}$$

A visualization of the solution component v at $t = 0,\ 2000$ and 4000 is shown in Figure 6.3.

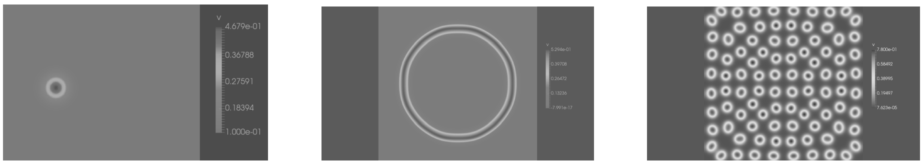

Figure 6.3: Solution component v at $t = 0$ (left), $t = 2000$ (middle) and $t = 4000$ (right) for (6.8).

3 dimensions

In 3 spatial dimensions $\mathbf{x} = (x,y,z) \in [-4\pi, 4\pi]^3$ we prescribe the initial condition

$$\begin{aligned} u(x,y,z,0) &= 0.5 + \exp(-1-(x^2+y^2+z^2)), \qquad (6.12)\\ v(x,y,z,0) &= 0.1 + \exp(-1-(x^2+y^2+z^2)). \qquad (6.13) \end{aligned}$$

In Figure 6.4 we show the component v computed by a complex embedded 4/3 splitting pair from [19] with an underlying spatial discretization with 512^3 basis functions and a tolerance of 10^{-5}. The solution is plotted at times $t = 2500$, $t = 3000$, $t = 4000$, and $t = 5000$.

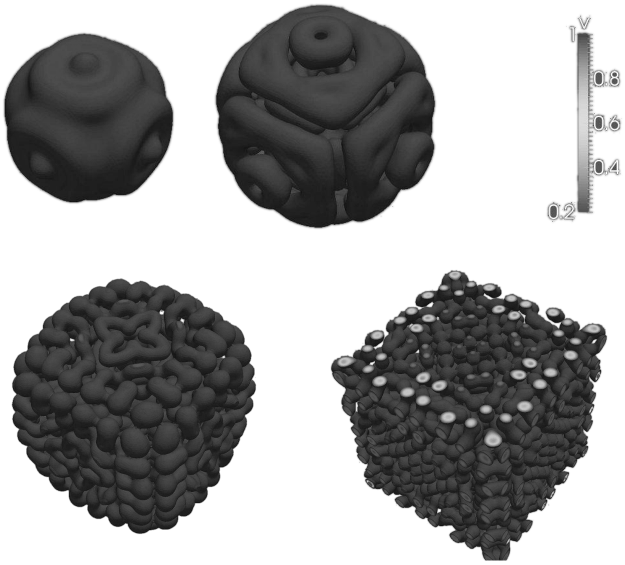

Figure 6.4: Solution component v for (6.8) in 3D at times $t = 2500, 3000, 4000, 5000$.

6.4 Schrödinger Equations

Splitting also works for wave equations, though one should choose only schemes with real coefficients. The standard form of the non relativistic Schrödinger equation for a single particle moving in an electric field is given by,

$$\mathrm{i}\hbar\partial_t\psi(\mathbf{x},t) = \left(-\frac{\hbar^2}{2m}\Delta + V(\mathbf{x},t)\right)\psi(\mathbf{x},t) \tag{6.14}$$

with $\hbar$ the Planck constant, m the particle's mass and V the potential energy. The notation for the unknown function is changed form u to ψ, because it is standard in literature about Schrödinger equations. There are two examples considered first the cubic nonlinear Schrödinger (NLS) equation 6.4.1 and a semi classical example in 6.4.2 with an time dependent potential.

6.4.1 The cubic NLS

The cubic nonlinear Schrödinger equation [26] is given by

$$\mathrm{i}\,\partial_t\psi(\mathbf{x},t) = -\tfrac{1}{2}\Delta\psi(\mathbf{x},t) + \kappa\,|\psi(\mathbf{x},t)|^2\,\psi(\mathbf{x},t)\,, \tag{6.15a}$$

$$\psi(\mathbf{x},0) = \psi_0(\mathbf{x})\,, \qquad \mathbf{x}\in\mathbb{R}^n\,, \quad t>0\,, \tag{6.15b}$$

where we set $\kappa = -1$. To reduce computing complexity and because all interesting effects are available, the 1-dimensional case is considered $\mathbf{x} = x \in \mathbb{R}$ with the initial condition chosen as the sum of two solitons,

$$\psi_0(x) = \sum_{j=1}^{2} \frac{a_j e^{-ib_j x}}{\cosh(a_j(2x - c_j))}, \qquad x \in [-16, 16]. \tag{6.15c}$$

with $a_1 = 2.0$, $a_2 = 2.0$, $b_1 = 1.0$, $b_2 = 3.0$, $c_1 = 5.0$, $c_2 = -5.0$.

The cubic NLS has been treated earlier in [4], a theoretical analysis is given in [20]. The two solitons which eventually cross [18], posing a challenge for an adaptive step-size selection algorithm. The two solitons cross approximately at $t = 2.3$ as Figure 6.5 shows. The step size drops by a factor of 5 during the collision and afterwards regains to previous level. This kind of behaviour for the solution is necessary that the adaptive algorithms are faster than using an equidistant time integrator to achieve the same global error at t_{end} as Table 6.26 shows.

The last column in Table 6.26, ladled with err, shows that the global error strongly correlates with the prescribed tolerance. The scheme `PP 5/6 A` is the best for low tolerances and it beats the equidistant step size solution every time. The schemes with the defect based error estimator often fail to beat the equidistant step size solution.

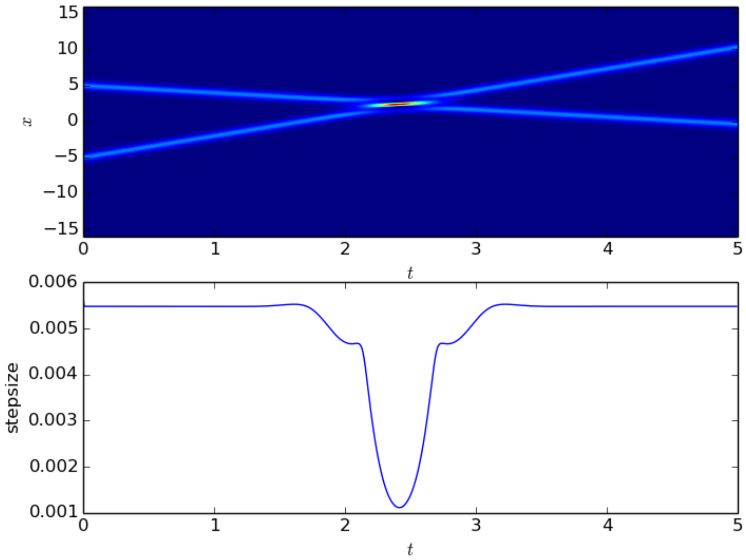

Figure 6.5: Solution of (6.15) (top), and stepsizes generated by an adaptive procedure based on `PP 5/6 A` with tolerance 10^{-10} (bottom).

Method	# steps adap	# steps equi	time adap	time equi	err
TOL = 10^{-5}					
`Emb 4/3 AK p`	275	586	0.123	0.130	$6.646e-05$
`PP 3/4 A`	438	772	0.150	0.106	$7.608e-05$
`PP 5/6 A`	154	394	0.136	0.149	$3.447e-04$
`Emb 4/3 AK p D`	265	469	0.131	0.103	$1.848e-04$
`PP 3/4 A D`	438	776	0.185	0.126	$7.526e-05$
`PP 5/6 A D`	188	581	0.188	0.263	$3.945e-05$
TOL = 10^{-8}					
`Emb 4/3 AK p`	1536	3159	0.655	0.617	$6.971e-08$
`PP 3/4 A`	2478	2858	0.758	0.427	$3.719e-07$
`PP 5/6 A`	503	1292	0.392	0.477	$1.114e-07$
`Emb 4/3 AK p D`	1495	1619	0.647	0.316	$9.817e-07$
`PP 3/4 A D`	2478	2865	0.924	0.438	$3.717e-07$
`PP 5/6 A D`	567	1391	0.449	0.561	$6.386e-08$
TOL = 10^{-10}					
`Emb 4/3 AK p`	4854	10382	1.975	2.091	$6.609e-10$
`PP 3/4 A`	7837	6766	2.406	1.060	$1.174e-08$
`PP 5/6 A`	1136	2433	0.945	0.975	$1.786e-09$
`Emb 4/3 AK p D`	4728	3870	2.163	0.862	$3.089e-08$
`PP 3/4 A D`	7837	6766	2.506	1.069	$1.174e-08$
`PP 5/6 A D`	1174	2491	1.122	1.074	$1.553e-09$

Table 6.26: Comparison of the efficiency of `Emb 4/3 AK p`, `PP 3/4 A`, and `PP 5/6 A` and their their corresponding defect based error estimators for Nonlinear Schroedinger. The tolerances were 10^{-5} (top), 10^{-8} (middle) and 10^{-10} (bottom), respectively. For equidistant time the optimal stepsize to get the same up to a 1% margin global error at $t_{end} = 5.0$ was used.

In Figure 6.6 the Table 6.26 is used to generate a work precision diagram. On the x-axis the compuation time in seconds is drawn and on the y-axis the L_2 norm of the global error at t_{end}. Lines of the same color belong to same scheme, full lines are with adaptive time integration and the dashed ones for equidistant time integration. `Strang` scheme is added to show off the superiority of the other schemes. The `Emb 4/3 AK p` is the best third order scheme, the adaptive algorithm is also faster than the equidistant strategy.

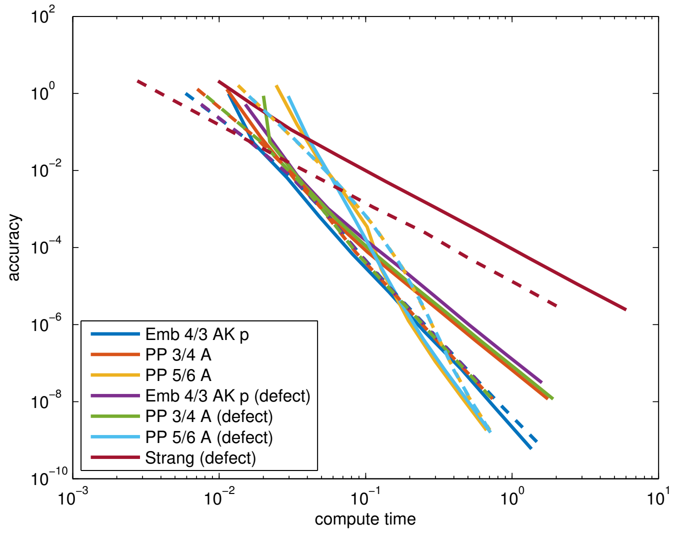

Figure 6.6: Work precission diagram of (6.15).

6.4.2 Quantum control

A model of quantum control of atomic systems discussed in [17] introduces a potential which explicitly depends on time to control the permeability:

$$i\partial_t \psi(\mathbf{x},t) = \varepsilon \Delta \psi(\mathbf{x},t) + \varepsilon^{-1} V(\mathbf{x},t)\, \psi(\mathbf{x},t)\,, \tag{6.16a}$$

$$\psi(\mathbf{x},0) = \psi_0(\mathbf{x})\,, \qquad \mathbf{x} \in \mathbb{R}^n\,, \quad t > 0\,, \tag{6.16b}$$

with potential $V(\mathbf{x},t)$. The potential is the sum of an time independent and a time dependent one, which effects permeability. For this example it is also chosen the 1-dimensional case, where $\mathbf{x} = x \in \mathbb{R}$. The initial state and potential are given by

$$\begin{aligned}
\psi_0(x) &= (\delta\pi)^{-\frac{1}{4}} e^{\frac{ik_0(x-x_0)}{\delta} - \frac{(x-x_0)^2}{2\delta}} \\
V(x,t) &= V_0(x) + \underbrace{\rho(3t-1)\rho(\sin(2\pi(x-t)))}_{=:E(x,t)} \\
V_0(x) &= \rho(4x)\sin(20\pi x) \\
\rho(x) &= \begin{cases} e^{\frac{-1}{1-x^2}}, & |x| < 1 \\ 0, & \text{otherwise,} \end{cases}
\end{aligned}$$

where $x_0 = -0.3$, $k_0 = 0.1$, $\delta = 10^{-3}$ and $\varepsilon = 2^{-8}$. The space was discretized with 2048 points on the interval $(-1,1)$, with periodic boundary conditions.

It is clearly visible by comparing Figure 6.7 and Figure 6.8 that the time dependent potential allows larger amounts to pass through. This shows that the anticipated effect can be reached.

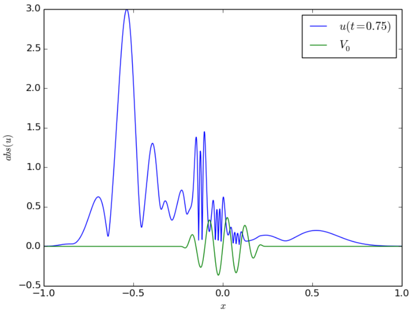

Figure 6.7: $V(x,t) = V_0(x)$

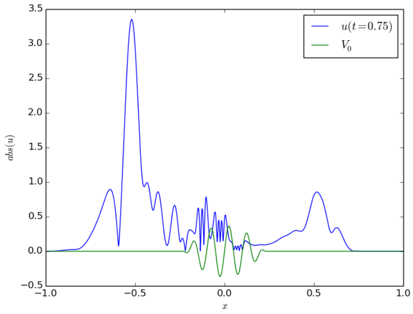

Figure 6.8: $V(x,t) = V_0(x) + E(x,t)$

Figure 6.9 shows the space time plot of the solution with the time dependent potential. It shows that during the interaction of the solition and the barrier the step size decreases and in the end increases again. Changing the semiclassical

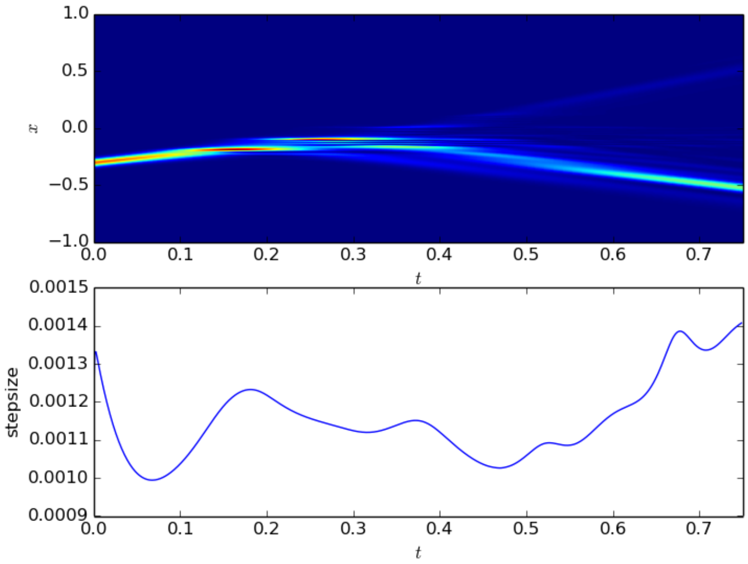

Figure 6.9: Solution (top) and step sizes (bottom) generated for (6.16) with $\varepsilon = 2^{-8}$ for `PP 5/6 A` and tolerance 10^{-10}.

parameter ε to 10^{-5} gives after an initial drop by factor 10 oscillating step sizes, see Figure 6.10. This results lead to further investigation on the parameter ε and the step size selection strategy.

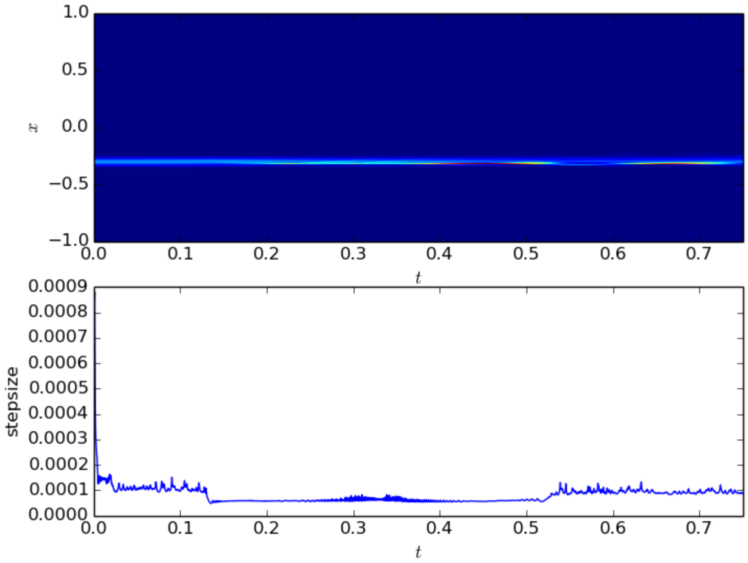

Figure 6.10: Solution (top) and step sizes (bottom) generated for (6.16) with $\varepsilon = 10^{-5}$ for `PP 5/6 A` and tolerance 10^{-10}.

To compare the step size selection strategies from 4.5 the change ratio is plotted in Figure 6.11, where ‘Simple’ refers to (4.23) and ‘2 Steps’ to (4.25). The change ratio in the first step is the highest because the algorithm needs to find the sutible stepsize. The 2 Step algorithm seems to have a smoother sequence of step sizes, especially in the case with small ε. Table 6.27 shows that the simple step size selection algorithm 4.23 is better than 4.25 in this situation. When forced to reach the given local error tolerance 10^{-10} by a relative tolerance of 10^{-1},

Selection startegy	ε	steps	rejected steps	rejections per steps
Simple	$3.91e-03$	655	5	0.0076
	$1.00e-05$	10386	12	0.0012
2 Steps	$3.91e-03$	662	16	0.0242
	$1.00e-05$	10489	101	0.0096

Table 6.27: Comparing different step size selection methods `PP 5/6 A`

$$\text{err} < \text{TOL} \quad \text{and} \quad \frac{\text{err}}{\text{TOL}} - 1 < \text{reltol} \tag{6.17}$$

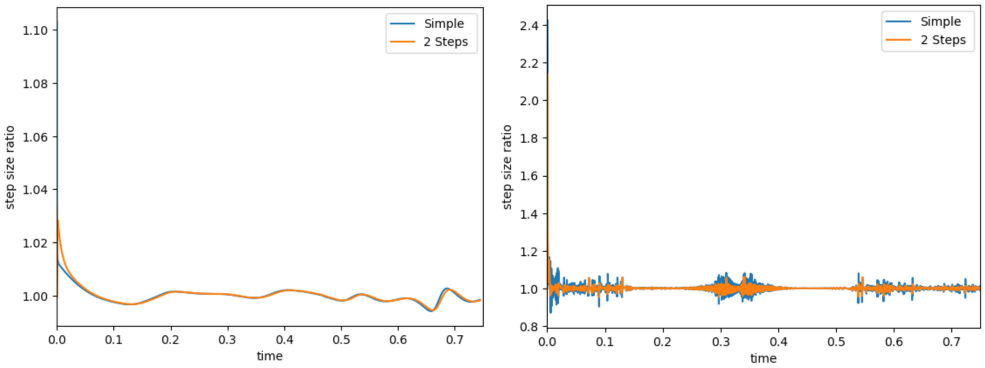

Figure 6.11: Comparing selection strategies for `PP 5/6 A` with tolerance 10^{-10}. Left $\varepsilon = 2^{-8}$ and right $\varepsilon = 10^{-5}$.

the number of rejected steps increases, as Table 6.28 shows. The two-step step size selection algorithm performs now better. But for the smaller value of ε both strategies fail, because they end up in an cycle of insufficient step sizes. Decreasing the relative tolerance to 10^{-2} drastically deteriorates the number of rejected steps for the two-step step size selection strategy, see Table 6.29

Selection startegy	ε	steps	rejected steps	rejections per steps
Simple	$3.91e-03$	587	1408	2.3986
	$1.00e-05$	∞	∞	—
2 Steps	$3.91e-03$	591	989	1.6734
	$1.00e-05$	∞	∞	—

Table 6.28: Comparing different step size selection methods `PP 5/6 A` forced relative tolerance 10^{-1}.

Selection startegy	ε	steps	rejected steps	rejections per steps
Simple	$3.91e-03$	586	1538	2.6246
	$1.00e-05$	∞	∞	—
2 Steps	$3.91e-03$	586	4605	7.8584
	$1.00e-05$	∞	∞	—

Table 6.29: Comparing different step size selection methods `PP 5/6 A` forced relative tolerance 10^{-2}.

Investigating small ε

Table 6.30 and 6.31 show for different values of ε, how the error estimator and the size of the first step correlate. For a given ε and given tolerance 10^{-10} the adaptive step size selecting strategy has been applied 100 times, for the step starting from $t_0 = 0$. The result of this procedure is seen in the column labeled h. The column to the right tries to measure if the step size converged to a specific value or oscillates by giving the ratio of h and the previously step size h_{-1}, 1.0 is the value for convergence. The last two columns give the by the error estimator estimated error and the exact error to the reference solution computed by $\frac{h}{100}$ with `PP 5/6 A`. The error estimator is in the range of the exact error. Though the error estimator for small epsilon sometimes leads to too large step sizes and errors but the number of rejected steps is only slightly higher than in cases with larger ε.

ε	h	h/h_{-1}	err est	err
$3.91e-03$	$1.347e-03$	$1.000e+00$	$5.314e-11$	$5.698e-11$
$1.00e-05$	$4.165e-03$	$4.000e+00$	$2.177e-07$	$3.894e-07$

Table 6.30: 1 step `PP 5/6 A`

ε	h	h/h_{-1}	err est	err
$3.91e-03$	$1.334e-03$	$1.000e+00$	$5.053e-11$	$5.414e-11$
$1.00e-05$	$3.906e-03$	$4.000e+00$	$1.774e-07$	$3.130e-07$

Table 6.31: 1 step `PP 5/6 A D`

Further investigations on the parameter ε yielded that for some values of ε the step size selection leads to oscillations. The step size change is limited by a factor of 4, see 4.5 and Figure 6.12. The instant change from $\frac{1}{4}$ to 4 and back are the result of swapping from the too large step size to the smaller one or vice versa. The defect based error estimator has more problems with small ε. Reducing the change rate to 2 gives Figure 6.13. Comparing both Figures shows that the problem is not solved by this measure.

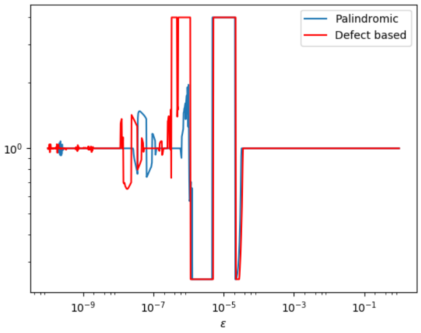

Figure 6.12: Comparing h/h_{-1} for `PP 5/6 A` and `PP 5/6 A D` with tolerance 10^{-10} and changerate limited by 4.

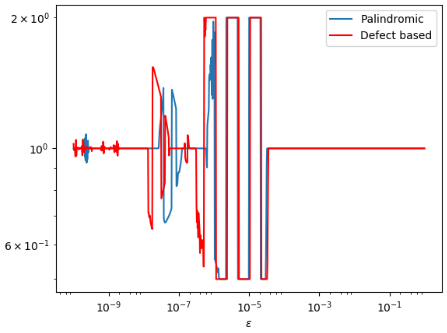

Figure 6.13: Comparing h/h_{-1} for `PP 5/6 A` and `PP 5/6 A D` with tolerance 10^{-10} and changerate limited by 2.

Chapter 7

Spectral Method

The spectral method is used to solve partial differential equations, by representing the solution using global ansatz functions. Typical ansatz function are sin and cos for periodic boundary conditions or Chebyshev polynomials else.

Consider the NLS equation (6.15). The spacial dimension is set to 1 for simplicity. In higher dimensions the arguments follow the same. Since the L^2-function ψ can be assumed to be negligible outside an interval which can be scaled down to $[-\pi,\pi]$, without restriction of generality,

$$\begin{aligned}
\mathrm{i}\,\partial_t\psi(x,t) &= -\tfrac{1}{2}\Delta\psi(x,t)+\kappa\,|\psi(x,t)|^2\,\psi(x,t)\,, && (7.1)\\
\psi(x,0) &= \psi_0(x)\,, \qquad x\in[-\pi,\pi]\,, \quad t>0\,, && (7.2)\\
u(-\pi,t) &= u(\pi,t)\,, \qquad t\in[0,t_{end}] && (7.3)
\end{aligned}$$

Trigonometric Interpolation First the desired function is semidiscretized in space, by using an cut of Fourier series.

$$\psi(x,t)\approx\psi_k(x,t)=\sum_{j=-\frac{k}{2}}^{\frac{k}{2}-1}c_j(t)\mathrm{e}^{\mathrm{i}jx}, \quad x\in[-\pi,\pi], \quad k \text{ even} \tag{7.4}$$

The $c_j(t)$ satisfy,

$$c_j(t)=\int_{-\pi}^{\pi}\psi(x,t)\mathrm{e}^{\mathrm{i}jx}dx. \tag{7.5}$$

In this case, we consider collocation on an equidistant grid

$$x_j=j\frac{2\pi}{k}, \quad j=-\frac{k}{2},-\frac{k-1}{2},\dots,\frac{k}{2}-1, \tag{7.6}$$

which yields the set of equations,

$$\mathrm{i}\,\partial_t\psi_k(x_j,t)=-\tfrac{1}{2}\Delta\psi_k(x_j,t)+\kappa\,|\psi_k(x_j,t)|^2\,\psi_k(x,t), \quad j=-\frac{k}{2},-\frac{k-1}{2},\dots,\frac{k}{2}-1. \tag{7.7}$$

Discrete Fourier Transform Let $\mathscr{F}_k : \mathbb{C}^k \to \mathbb{C}^k$ denote the discrete Fourier transform of length k,

$$\hat{\phi} = \mathscr{F}_k \phi \text{ with } \hat{\phi}_l = \sum_{j=-\frac{k}{2}}^{\frac{k}{2}-1} \mathrm{e}^{-\mathrm{i} l j 2\frac{\pi}{k}} \phi_j, \quad l = -\frac{k}{2}, -\frac{k-1}{2}, \ldots, \frac{k}{2} - 1. \tag{7.8}$$

The inverse transform $\mathscr{F}_k^{-1}$ is given by

$$\phi = \mathscr{F}_k^{-1} \hat{\phi} \text{ with } \phi_l = \frac{1}{k} \sum_{j=-\frac{k}{2}}^{\frac{k}{2}-1} \mathrm{e}^{\mathrm{i} l j 2\frac{\pi}{k}} \hat{\phi}_j, \quad l = -\frac{k}{2}, -\frac{k-1}{2}, \ldots, \frac{k}{2} - 1. \tag{7.9}$$

Though mathematicians like it more if $\mathscr{F} = \mathscr{F}^{-1}$, but this is computational extensive because than you would have to scale the data twice with a factor $\frac{1}{\sqrt{k}}$ instead once with $\frac{1}{k}$. Both of the transforms can be implemented with computational complexity $\mathscr{O}(k \log k)$ via the fast Fourier transform (FFT) [12]. For the Fourier transform generally holds,

$$\mathscr{F} \frac{\partial}{\partial x} \phi(x) = \mathrm{i} x \mathscr{F} \phi(x) \tag{7.10}$$

The derivative in x direction in the problem is changed to a multiplication after the Fourier transformation. This changes the PDE to an ODE, in t

$$\mathrm{i} \partial_t \psi_k = -\frac{1}{2} \mathscr{F}_k^{-1} (\mathrm{i} x)^2 \mathscr{F}_k \psi_k + |\psi_k|^2 \psi_k \tag{7.11}$$

The approximation error at $t = 0$ for any $s \in \mathbb{N}^+$ is bounded by

$$\|\psi - \psi_k\|_{L_2} \leq C k^{-s} \|\partial_x^s \psi\|_{L_2} \tag{7.12}$$

if $\partial_x^s \psi(\cdot, 0) \in L_2$, which can be shown by Parseval's formula and partial integration. The error bounds for all $t > 0$ is given by,

$$\|\psi(t) - \psi_k(t)\|_{L_2} \leq C k^{-s} (1 + t) \max_{\tau \in [0,t]} \|\partial_x^s \psi(\tau)\|_{L_2} \tag{7.13}$$

if the exact solution fulfills $\partial_x^{s+2} \psi(\cdot, t) \in L_2$.

List of Figures

6.1 Step size over time for `Magnus2` with TOL = 10^{-8} with different initial step sizes. 40

6.2 Scaling the number of cores. 45

6.3 Solution component v at $t = 0$ (left), $t = 2000$ (middle) and $t = 4000$ (right) for (6.8). 46

6.4 Solution component v for (6.8) in 3D at times $t = 2500$, 3000, 4000, 5000. 47

6.5 Solution of (6.15) (top), and stepsizes generated by an adaptive procedure based on `PP 5/6 A` with tolerance 10^{-10} (bottom). . . 48

6.6 Work precission diagram of (6.15). 50

6.7 $V(x,t) = V_0(x)$ 51

6.8 $V(x,t) = V_0(x) + E(x,t)$ 51

6.9 Solution (top) and step sizes (bottom) generated for (6.16) with $\varepsilon = 2^{-8}$ for `PP 5/6 A` and tolerance 10^{-10}. 51

6.10 Solution (top) and step sizes (bottom) generated for (6.16) with $\varepsilon = 10^{-5}$ for `PP 5/6 A` and tolerance 10^{-10}. 52

6.11 Comparing selection strategies for `PP 5/6 A` with tolerance 10^{-10}. Left $\varepsilon = 2^{-8}$ and right $\varepsilon = 10^{-5}$ 53

6.12 Comparing h/h_{-1} for `PP 5/6 A` and `PP 5/6 A D` with tolerance 10^{-10} and changerate limited by 4. 55

6.13 Comparing h/h_{-1} for `PP 5/6 A` and `PP 5/6 A D` with tolerance 10^{-10} and changerate limited by 2. 55

List of Tables

6.1 full matrix $(n = 4900)$ 27
6.2 sparse matrix $(n = 4900)$ 27
6.3 Number of matrix exponential computed per step. 28
6.4 Global Error at $t = 8.0$; `Magnus2` 29
6.5 Global Error at $t = 8.0$; `Symm-Milne-32` 30
6.6 Local error and error of the defect based error estimator for `Magnus2` 31
6.7 Global error at $t = 8.0$ for equidistant time-stepping, various stepsizes 32
6.8 Effort for the error estimator 32
6.9 Comparison of the efficiency of `Strang (Magnus2)` and `Symm-Milne-32`. The tolerances were 10^{-8} (top), 10^{-10} (middle) and 10^{-12} (bottom), respectively. For equidistant time the minimal stepsize was used 33
6.10 Global error at $t = 8.0$ for equidistant time-stepping, various stepsizes 34
6.11 Effort for the error estimator 34
6.12 Comparison of the efficiency of `Magnus4`, `Magnus4Optimized` and `Emb 5/4 AK (ii)` The tolerances were 10^{-8} (top), 10^{-10} (middle) and 10^{-12} (bottom), respectively. For equidistant time the minimal stepsize was used 35
6.13 Global Error at $t = 8.0$; `Magnus2` 36
6.14 Global Error at $t = 8.0$; `Symm-Milne-32` 37
6.15 Global Error at $t = 8.0$; `EMB 5/4 AK (ii)` 37
6.16 Local error and error of the defect based error estimator for `Magnus2` 38
6.17 Global error at $t = 8.0$ for equidistant time-stepping, various stepsizes 38
6.18 Effort for the error estimator 39
6.19 Comparison of the efficiency of `Strang (Magnus2)` and `Symm-Milne-32` The tolerances were 10^{-4} (top), 10^{-6} (middle) and 10^{-8} (bottom), respectively. For equidistant time the minimal stepsize was used . 39

6.20 Global error at $t = 8.0$ for equidistant time-stepping, various stepsizes. . . . 40
6.21 Effort for the error estimator . . . 41
6.22 Comparison of the efficiency of `Magnus4`, `Magnus4Optimized` and `Emb 5/4 AK (ii)` The tolerances were 10^{-4} (top), 10^{-6} (middle) and 10^{-8} (bottom), respectively. For equidistant time the minimal stepsize was used. Note that for TOL=10^{-8} the accuracy of the matrix exponetial is the limiting factor. . . . 42
6.23 Global error at $t = 1.0$; `Strang` . . . 43
6.24 Global error at $t = 1.0$; `Strang ABC` . . . 44
6.25 Comparison of the efficiency of `Milne 2/2 c (i)`, `Emb 4/3 A c`, `PP 3/4 A c` and `PP 3/4 A 3 c` for (6.8). The tolerances were 10^{-5} (top) and 10^{-8} (bottom), respectively. . . . 44
6.26 Comparison of the efficiency of `Emb 4/3 AK p`, `PP 3/4 A`, and `PP 5/6 A` and their their corresponding defect based error estimators for Nonlinear Schroedinger. The tolerances were 10^{-5} (top), 10^{-8} (middle) and 10^{-10} (bottom), respectively. For equidistant time the optimal stepsize to get the same up to a 1% margin global error at $t_{end} = 5.0$ was used. . . . 49
6.27 Comparing different step size selection methods `PP 5/6 A` . . . 52
6.28 Comparing different step size selection methods `PP 5/6 A` forced relative tolerance 10^{-1}. . . . 53
6.29 Comparing different step size selection methods `PP 5/6 A` forced relative tolerance 10^{-2}. . . . 53
6.30 1 step `PP 5/6 A` . . . 54
6.31 1 step `PP 5/6 A D` . . . 54

Bibliography

[1] A. Alverman and H. Fehske. High-order commutator-free exponential time-propagation of driven quantum systems. *J. Comput. Phys.*, 230:5930–5956, 2011.

[2] A. Alverman, H. Fehske, and P.B. Littlewood. Numerical time propagation of quantum systems in radiation fields. *New J. Phys.*, 14:105008, 2012.

[3] W. Auzinger and W. Herfort. Local error structures and order conditions in terms of Lie elements for exponential splitting schemes. *Opuscula Math.*, 34:243–255, 2014.

[4] W. Auzinger, H. Hofstätter, O. Koch, and M. Thalhammer. Defect-based local error estimators for splitting methods, with application to Schrödinger equations, Part III: The nonlinear case. *J. Comput. Appl. Math.*, 273:182–204, 2014.

[5] W. Auzinger and O. Koch. Coefficients of various splitting methods. `http://www.asc.tuwien.ac.at/~winfried/splitting/`.

[6] W. Auzinger, O. Koch, and M. Thalhammer. Defect-based local error estimators for splitting methods, with application to Schrödinger equations, Part I: The linear case. *J. Comput. Appl. Math.*, 236:2643–2659, 2012.

[7] W. Auzinger, O. Koch, and M. Thalhammer. Defect-based local error estimators for splitting methods, with application to Schrödinger equations, Part II: Higher-order methods for linear problems. *J. Comput. Appl. Math.*, 255:384–403, 2013.

[8] W. Auzinger, O. Koch, and M. Thalhammer. Defect-based local error estimators for high-order splitting methods involving three linear operators. *Numer. Algorithms*, 70:61–91, 2015.

[9] Winfried Auzinger, Harald Hofstätter, David Ketcheson, and Othmar Koch. Practical splitting methods for the adaptive integration of nonlinear evolution

equations. part I: Construction of optimized schemes and pairs of schemes. *BIT Numerical Mathematics*, 57:55–74, 2017.

[10] Winfried Auzinger, Othmar Koch, and Michael Quell. Adaptive high-order splitting methods for systems of nonlinear evolution equations with periodic boundary conditions. *Numerical Algorithms*, pages 1–23, 2016.

[11] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

[12] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[13] P. Gray and S.K. Scott. *Chemical Waves and Instabilities*. Clarendon, Oxford, 1990.

[14] E. Hairer, Ch. Lubich, and G. Wanner. *Geometric Numerical Integration*. Springer-Verlag, Berlin–Heidelberg–New York, 2002.

[15] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I*. Springer-Verlag, Berlin–Heidelberg–New York, 1987.

[16] Nicholas J Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM Journal on Matrix Analysis and Applications*, 26(4):1179–1193, 2005.

[17] Arieh Iserles, Karolina Kropielnicka, and Pranav Singh. On the discretisation of the semiclassical schrödinger equation with time-dependent potential. *Cambridge Numerical Analysis Report NA2015/02. Cambridge, UK: Cambridge Numerical Analysis Group, Department of Applied Mathematics and Theoretical Physics, Cambridge University*, 2015.

[18] M.S. Ismail and T.R. Taha. Numerical simulation of coupled nonlinear Schrödinger equation. *Math. Comput. Simulation*, 56:547–562, 2001.

[19] O. Koch, Ch. Neuhauser, and M. Thalhammer. Embedded split-step formulae for the time integration of nonlinear evolution equations. *Appl. Numer. Math.*, 63:14–24, 2013.

[20] O. Koch, Ch. Neuhauser, and M. Thalhammer. Error analysis of high-order splitting methods for nonlinear evolutionary Schrödinger equations and application to the MCTDHF equations in electron dynamics. *M2AN Math. Model. Numer. Anal.*, 47:1265–1284, 2013.

[21] Ning Li and Sylvain Laizet. 2decomp & fft-a highly scalable 2d decomposition library and fft interface. In *Cray User Group 2010 conference*, pages 1–13, 2010.

[22] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C — The Art of Scientific Computing*. Cambridge University Press, Cambridge, U.K., 1988.

[23] Blanes S., F. Casas, and A. Murua. Splitting and composition methods in the numerical integration of differential equations. *Bol. Soc. Esp. Mat. Apl.*, 45:87–143, 2008.

[24] Roger B Sidje. Expokit: a software package for computing matrix exponentials. *ACM Transactions on Mathematical Software (TOMS)*, 24(1):130–156, 1998.

[25] Amy Henderson Squillacote and James Ahrens. *The paraview guide*, volume 366. Kitware, 2007.

[26] C. Sulem and P.-L. Sulem. *The Nonlinear Schrödinger Equation*. Appl. Math. Sciences. Springer-Verlag, New York, 1999.

[27] `http://vsc.ac.at/`. Vienna Scientific Cluster. 2011.