Andreas Frank Martitsch, BSc

Heat transport problems in magnetized plasmas solved by a two dimensional conservative finite difference method

MASTER THESIS

for obtaining the academic degree
Diplom-Ingenieur

Master Programme of
Technical Physics

Graz University of Techonology

Supervisor:
Ao.Univ.-Prof. Dipl.-Ing. Dr.phil. Martin Heyn

Co-Supervisor:
Ass.Prof. Dipl.-Ing. Dr.techn. Winfried Kernbichler

Institute of Theoretical and Computational Physics

Graz, June 5, 2012

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
date (signature)

Kurzfassung

Die numerische Lösung von Transportgleichungen ist grundlegend für das physikalische Verständnis von Fusionsplasmen. Eine makroskopische Beschreibung dieses Transportproblems ist mittels einer Konvektion-Diffusionsgleichung möglich. Schwierigkeiten in der numerischen Behandlung stammen von der Steifheit der betrachteten Differentialgleichung, die wiederum durch die Anisotropien innerhalb des Fusionsplasmas hervorgerufen wird. Realistische Werte für die Anisotropien - darunter versteht man das Verhältnis von parallelem Transport zu normalem Transport bezüglich der Magnetfeldlinen - decken acht bis zwölf Größenordnungen ab. Während der letzten Dekade wurden die bekannten und weitverbreiteten numerischen Methoden zur Lösung von partiellen Differentialgleichungen, wie beispielsweise die Finite-Differenzenmethode oder die Finite-Elemente-Methode, auf verschiedene Transportprobleme angewendet. In dieser Arbeit wird ein konservatives Finites-Differenzenschema entwickelt, wobei ein unseres Wissens nach neuer Ansatz für die Rekonstruktion der numerischen Flussfunktionen mit entsprechend hoher Ordnung herangezogen wird. Die entwickelte Methode kann direkt auf ein adaptives Gitter übertragen werden. Zusätzlich werden geeignete Zeitintegrationsmethoden für die Untersuchung von zeitabhängigen Problemen vorgestellt und analysiert. Des Weiteren werden drei Programmpakete zur Lösung von schwachbesetzten, linearen Gleichungssystemen bezüglich ihrer Effizienz und Genauigkeit getestet. Hierbei ist die richtige Wahl der Routine entscheidend für insgesamte Effizienz des entwickelten Programms, da die räumliche und zeitliche Diskretisierung der partiellen Differentialgleichungen zu linearen Gleichungssystemen mit großen, schwachbesetzten Koeffizientenmatrizen führt. Um die Funktionsfähigkeit und Stabilität des konservativen Finiten-Differenzenschemas zu überprüfen, werden einige Testszenarien ausgearbeitet und die erzielten Ergebnisse ausführlich diskutiert.

Abstract

The numerical solution of transport equations plays an essential role in the physics of fusion plasmas. In this context, a macroscopic mathematical description of the transport problem is given by a convection-diffusion equation. Difficulties in the numerical treatment mainly arise from the stiffness of the considered differential equation which is caused by the anisotropies within the plasma of fusion research devices. Realistic values for the anisotropies, i.e. the ratio of parallel to perpendicular transport with respect to the magnetic field lines, cover eight to twelve orders of magnitude, whereby the small parameters may not be neglected as they lead to essential physics. Since the last decade the well-known and wide-spread numerical techniques for the solution of partial differential equations, such as the finite difference method and the finite element method, have been applied to the transport problem. In this thesis, a conservative finite difference scheme using a high order reconstruction of the numerical flux functions is developed. The formulated method can readily be extended to an adaptive mesh which is a part of ongoing research. In addition, a selection of suitable time integration procedures for the investigation of time-dependent problems is presented and analyzed. Furthermore, three widely used library routines for the solution of sparse linear systems of equations are benchmarked with respect to performance and accuracy. The right choice of the sparse linear systems of equations solver is crucial for the overall performance of the developed code since the spatial and temporal discretization of the partial differential equations leads to linear systems of equations with large, sparse coefficient matrices. So as to prove the operability and the stability of the scheme, several test case scenarios including an application to the heat transport in a Tokamak are worked out and studied extensively.

Acknowledgements

A lot of people are involved in this thesis in its present form. I do not want to rank the individual persons, since everyone plays an important role in the development. If one of them would miss, the thesis might not look like this or would even not exist.

I want to thank Prof. Martin Heyn for agreeing to supervise my work. I thank Ass.-Prof. Winfried Kernbichler who was always available for clarifying discussions and provided a lot of ideas concerning the basic concepts as well as the computational implementation. He also gave me the opportunity to work at the institute very early and encouraged me at every stage of my studies. At this place I also would like to give thanks to Dr. Sergei Kasilov for introducing me patiently to the mathematics and physics of transport problems. For giving me a key-insight into conditions for the stability of numerical schemes I want to thank Prof. Ferdinand Schürrer.

I appreciate having had the chance to work in the plasma physics group at the Institute of Theoretical and Computational Physics with Peter Leitner, Klaus Allmaier, Miran Mulec and Gernot Kapper. In remembrance of the good time during the coffee breaks, I thank Martin Nuss, Iris Hehn, Matthias Hasewend and Klaus Lang.

For supplying me with coffee and sweets whilst putting the thesis down on paper, I thank Anna Holas. Finally, I want to thank my parents for their love, support and encouragement.

Contents

1 Sparse Linear Systems of Equations Solvers

All numerical techniques used to solve partial differential equations (PDE), such as finite-difference method (FDM), finite-element method (FEM) or finite-volume method (FVM), perform a spatial and temporal discretization of the PDE. This discretization of the PDE eventually leads to a linear system of equations. Subject to the size and refinement of the computational grid, the resulting linear systems of equations exhibit a more or less sparse structure, i.e. most entries are zero. Realistic problems consist of coefficient matrices of the size of $\sim 10^6 \times 10^6$ with about 0.05% non-zero entries. Hence, one requires about 30GB of memory including overhead to store the non-zero entries instead of 7400GB for the storage of the full coefficient matrix. For this reason a sparse storage of the coefficient matrix becomes inevitable and will be discussed in Section 1.1. Furthermore, one is interested in having a fast and numerical accurate sparse linear system of equations (SLSE) solver, because the solution of the linear systems of equations constitutes one of the most time-consuming steps. The different strategies for the factorization of asymmetric coefficient matrices are outlined in Section 1.2 and in Section 1.3 the results of the performed benchmarks are shown. The conclusions of this chapter have an impact on the choice of the right SLSE solver, which is the core element of the PDE solver developed in Chapter 2 and Chapter 3.

1.1 Sparse Matrix Storage Formats

In this section an overview of different sparse matrix storage formats is given whereby the two most widespread variants are described in detail. Before further discussion, the following terms are defined. Let n denote the number of columns or rows and τ the number of nonzero elements of a square matrix $\underline{\underline{\mathbf{A}}}$, where $\tau << n^2$. Since all considered matrices are nonsingular, n is equal to the rank of the matrix.
The book of Tewarson [1] shows some packed forms of storage like linked lists, an array of unique integers defining the position of the corresponding value in the matrix or a format where each nonzero element is compressed into an item of two storage cells, the first cell stores the row index and the second the value of the element. These storage formats differ in their memory consumption and computational effort for matrix operations. Linked lists are well suited for matrix calculations in which new nonzero elements are created or deleted, because there is no need of realignment of the remaining elements. The main disadvantage of linked lists lies in the fact that the memory requirement, $n + 3\tau$ memory locations to store $\underline{\underline{\mathbf{A}}}$, is higher than in static schemes. In comparison to the linked lists, static schemes, as the scheme III in [1,p. 8], only need 2τ memory locations to store the matrix. Most SLSE solvers comprised in known libraries, make use of two related formats, the compressed sparse column (CSC) format or the compressed sparse row (CSR) format, in order to process the large coefficient matrices. The CSC format and the CSR format are

associated with each other via the relationship

$$\mathrm{CSC}\left(\underline{\underline{\mathbf{A}}}\right) = \mathrm{CSR}\left(\underline{\underline{\mathbf{A}}}^{T}\right). \tag{1.1}$$

Equation (1.1) implies that the matrix $\underline{\underline{\mathbf{A}}}$ stored in the CSC format is equivalent to storing the transposed matrix $\underline{\underline{\mathbf{A}}}^{T}$ in the CSR format. For this reason interchanging the storage arrays for the row- and column-indices leads to a transposition of the matrix in the other format. The scheme II in [1,p. 7] represents the above-mentioned CSC format. The CSC format consists of three storage arrays for the value of elements (VE), row indices (RI) and column index pointer (CIP). Any nonzero element a_{mn} can be reconstructed from these three arrays. The values of the nonzero elements are stored in the array VE and each element of VE is linked to a corresponding element of RI which yields the row index. Furthermore, one requires the information about the column index. On that account, the elements of VE are ordered by column and the CIP array points to the indices of the elements of the VE array that match with the first element of the respective column. To illustrate this description, consider the subsequent matrix,

$$\underline{\underline{\mathbf{A}}} = \begin{pmatrix} 0 & 0 & a_{13} & 0 & 0 \\ a_{21} & 0 & 0 & a_{24} & 0 \\ 0 & 0 & a_{33} & 0 & 0 \\ a_{41} & 0 & 0 & 0 & a_{45} \\ 0 & a_{52} & 0 & 0 & 0 \end{pmatrix}, \tag{1.2}$$

which gives the CSC format representation,

$$\begin{aligned} \mathrm{VE} &= (a_{21},\, a_{41},\, a_{52},\, a_{13},\, a_{33},\, a_{24},\, a_{45}), \\ \mathrm{RI} &= (2,\quad 4,\quad 5,\quad 1,\quad 3,\quad 2,\quad 4), \\ \mathrm{CIP} &= (1,\quad 3,\quad 4,\quad 6,\quad 7). \end{aligned} \tag{1.3}$$

Thereby the three storage arrays (1.3) are filled by going consecutively through the columns of the matrix $\underline{\underline{\mathbf{A}}}$ (1.2). For each column one stores the values of the nonzero elements in the VE array and the corresponding row indices in the RI array. As mentioned, the elements of CIP array point to the first nonzero element of the respective column. The relation between the CIP array and VE array is established by means of a linear index that tags the elements of VE. Since the first column is filled with nonzero entries, the first element of the CIP array has to be one. The further entries of the CIP array are produced by incrementing the previous entry with the number of nonzero elements of the actual column.

1.2 Direct Methods for the Solution of Sparse Linear Systems of Equations

The aim of direct methods is the transformation of the coefficient matrix of a linear system of equations,

$$\underline{\underline{\mathbf{A}}} \cdot \underline{\mathbf{x}} = \underline{\mathbf{b}} \,, \tag{1.4}$$

into an upper triangular matrix,

$$\underline{\underline{\mathbf{U}}} = \left[u_{ij}\right] \quad \text{with} \quad u_{ij} = 0 \quad \text{for} \quad i > j.$$

Depending on the chosen algorithm, one obtains an equivalent system with a modified righthand side,

$$\underline{\underline{\mathbf{U}}} \cdot \underline{\mathbf{x}} = \underline{\mathbf{y}} \,, \tag{1.5}$$

which can be solved easily by a backward-substitution [2,p. 35],

$$\begin{aligned} x_{\mathrm{n}} &= \frac{y_{\mathrm{n}}}{u_{\mathrm{nn}}} \\ x_i &= \frac{1}{u_{ii}} \left[y_i - \sum_{j=i+1}^{\mathrm{n}} u_{ij}\, x_j \right] \quad \text{for} \quad i = \mathrm{n}-1, \mathrm{n}-2, \ldots, 1. \end{aligned}$$

A variant of the above-described Gaussian elimination is the LU decomposition [3,p. 178], introduced by the famous English mathematician *Alan M. Touring*, that constitutes the foundation of the library routines discussed in the later subsections. In fact, the algorithms for a sparse LU decomposition are much more complex [1,pp. 15-106] and can be found in the cited literature about the library routines. The following relations and results are obtained for a full matrix representation of $\underline{\underline{\mathbf{A}}}$; nevertheless, the basic understanding of the SLSE solvers is created within these calculations. One denotes the subsequent factorization of the coefficient matrix $\underline{\underline{\mathbf{A}}}$,

$$\underline{\underline{\mathbf{A}}} = \underline{\underline{\mathbf{L}}} \cdot \underline{\underline{\mathbf{U}}} \,, \tag{1.6}$$

as the LU decomposition of $\underline{\underline{\mathbf{A}}}$ where $\underline{\underline{\mathbf{U}}}$ is an upper triangular matrix,

$$\underline{\underline{\mathbf{U}}} = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \ldots & u_{1\mathrm{n}} \\ 0 & u_{22} & u_{23} & \ldots & u_{2\mathrm{n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & u_{\mathrm{nn}} \end{pmatrix},$$

and $\underline{\underline{\mathbf{L}}}$ is an unit lower triangular matrix,

$$\underline{\underline{\mathbf{L}}} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{\mathrm{n}1} & l_{\mathrm{n}2} & l_{\mathrm{n}3} & \dots & 1 \end{pmatrix}.$$

Using the LU decomposition (1.6), one is able to rewrite the linear system of equations (1.4),

$$\begin{aligned} \underline{\underline{\mathbf{A}}} \cdot \underline{\mathbf{x}} &= \left(\underline{\underline{\mathbf{L}}} \cdot \underline{\underline{\mathbf{U}}}\right) \cdot \underline{\mathbf{x}} \\ &= \underline{\underline{\mathbf{L}}} \cdot \underbrace{\left(\underline{\underline{\mathbf{U}}} \cdot \underline{\mathbf{x}}\right)}_{\equiv \underline{\mathbf{y}}} \\ &= \underline{\mathbf{b}}\,, \end{aligned}$$

from which the yet unknown righthand side $\underline{\mathbf{y}}$ of Equation (1.5) is apparent. Due to the special form of $\underline{\underline{\mathbf{L}}}$, the system $\underline{\underline{\mathbf{L}}} \cdot \underline{\mathbf{y}} = \underline{\mathbf{b}}$ can be solved by forward substitution [2,p. 35],

$$\begin{aligned} y_1 &= b_1 \\ y_i &= b_i - \sum_{j=1}^{i-1} l_{ij}\, y_j \quad \text{for} \quad i = 2,3,\dots,\mathrm{n}. \end{aligned}$$

A procedure for factorizing the coefficient matrix $\underline{\underline{\mathbf{A}}}$ into $\underline{\underline{\mathbf{L}}}$ and $\underline{\underline{\mathbf{U}}}$ components is *Crout's algorithm* [2,p. 36],

$$\begin{aligned} u_{ij} &= a_{ij} - \sum_{k=1}^{i-1} l_{ik}\, u_{kj} \qquad i = 1,\dots,j-1, \\ \gamma_{ij} &= a_{ij} - \sum_{k=1}^{j-1} l_{ik}\, u_{kj} \qquad i = j,\dots,\mathrm{n}, \\ u_{jj} &= \gamma_{jj}, \\ l_{ij} &= \frac{\gamma_{ij}}{\gamma_{jj}}\,. \end{aligned} \tag{1.7}$$

One substantial feature of the algorithm is that the l's and u's on the righthand side of Equation (1.7) are known when needed and there is never a lack of information. Another advantage is saving memory, because every element of the coefficient matrix a_{ij} is used only once and its storage location is overwritten by the corresponding u_{ij} or l_{ij}. The typical computational cost of such a Gaussian elimination procedure is of the order $\mathcal{O}(\frac{2\,n^3}{3})$ [4,p. 98]. As mentioned, the SLSE solver routines make use of more sophisticated approaches for the factorization of the coefficient matrices. A more detailed discussion is given in the book of Tewarson [1,pp. 83-91] including a variant of Crout's algorithm for sparse matrices with a minimization of fill-in.

So far considerations about numerical stability have been omitted. During the numerical treatment, one

recognizes a high impact of row interchanges on the solution of the linear system of equations. A strategy for the best possible row interchanges is called partial pivoting. In order to illustrate this behavior, think of subsequent set of linear equations [3,p. 86],

$$\begin{aligned} 0.400x + 99.6y &= 100 \\ 75.3x + 45.3y &= 30.0\,. \end{aligned}$$

Using three significant digits and by pivoting on 0.400, one obtains the solution $x = -1.00$ and $y = 1.01$. With respect to the actual solution $x = 1.00$ and $y = 1.00$, this yields a huge error of 200% in the x value. Pivoting on 75.3 gives by contrast the correct result. Thus, partial pivoting has to be implemented in the *Crout's algorithm*. If the absolute values of the l_{ij} are as small as possible, the least roundoff errors are obtained. This is guaranteed when the γ_{jj} in Equation (1.7) are as large as possible, which can be realized by interchanging the row with the largest γ_{ij} with the j-th row.
The proceeding subsections outline the differences between three wide-spread library routines concerning the factorization large asymmetric coefficient matrices.

1.2.1 SuperLU 4.0

SuperLU [5, 6] denotes a set of three ANSI C subroutine libraries for solving SLSE, especially for coefficient matrices with very unsymmetric structure. In this thesis, the Sequential SuperLU is only regarded because the parallel version, Multithreaded SuperLU, has not shown a better performance with respect to our demands in terms of computational speed and memory consumption. The sparse Gaussian elimination consists of two steps. In the first step a triangular factorization is computed,

$$\underline{\underline{\mathbf{P_r}}} \cdot \underline{\underline{\mathbf{D_r}}} \cdot \underline{\underline{\mathbf{A}}} \cdot \underline{\underline{\mathbf{D_c}}} \cdot \underline{\underline{\mathbf{P_c}}} = \underline{\underline{\mathbf{L}}} \cdot \underline{\underline{\mathbf{U}}},$$

where $\underline{\underline{\mathbf{D_r}}}$ and $\underline{\underline{\mathbf{D_c}}}$ are diagonal matrices to equilibrate the system, $\underline{\underline{\mathbf{P_r}}}$ and $\underline{\underline{\mathbf{P_c}}}$ indicate permutation matrices. $\underline{\underline{\mathbf{P_c}}}$ and $\underline{\underline{\mathbf{P_r}}}$ are chosen in such a way that the ordering of the columns and rows of $\underline{\underline{\mathbf{A}}}$ increases the sparsity of the computed LU factors, numerical stability and parallelism. This factorization is also possible for non-square matrices. The second step consists of solving $\underline{\underline{\mathbf{A}}} \cdot \underline{\mathbf{x}} = \underline{\mathbf{b}}$ by evaluation of

$$\begin{aligned} \underline{\mathbf{x}} &= \underline{\underline{\mathbf{A}}}^{-1} \cdot \underline{\mathbf{b}} \\ &= \left(\underline{\underline{\mathbf{D_r}}}^{-1} \cdot \underline{\underline{\mathbf{P_r}}}^{-1} \cdot \underline{\underline{\mathbf{L}}} \cdot \underline{\underline{\mathbf{U}}} \cdot \underline{\underline{\mathbf{P_c}}}^{-1} \cdot \underline{\underline{\mathbf{D_c}}}^{-1} \right)^{-1} \cdot \underline{\mathbf{b}} \\ &= \left(\underline{\underline{\mathbf{D_c}}} \cdot \left(\underline{\underline{\mathbf{P_c}}} \cdot \left(\underline{\underline{\mathbf{U}}}^{-1} \cdot \left(\underline{\underline{\mathbf{L}}}^{-1} \cdot \left(\underline{\underline{\mathbf{P_r}}} \cdot \left(\underline{\underline{\mathbf{D_r}}} \cdot \underline{\mathbf{b}} \right) \right) \right) \right) \right) \right). \end{aligned}$$

At the beginning the rows of $\underline{\mathbf{b}}$ are scaled by $\underline{\underline{\mathbf{D_r}}}$; and furthermore, the rows of $\underline{\underline{\mathbf{D_r}}} \cdot \underline{\mathbf{b}}$ are permuted by $\underline{\underline{\mathbf{P_r}}}$. The multiplications with $\underline{\underline{\mathbf{L}}}^{-1}$ and $\underline{\underline{\mathbf{U}}}^{-1}$, respectively, indicate solving triangular systems of equations.

The pivoting strategy of Sequential SuperLU to determine the row permutation $\underline{\underline{\mathbf{P_r}}}$ is called threshold pivoting [6]. Starting from a coefficient matrix $\underline{\underline{\mathbf{A}}}$ where the first i-1 columns are factorized, one looks for the pivot for column i. The element a_{mi} denotes the largest entry on or below the diagonal of the partially factored $\underline{\underline{\mathbf{A}}}$, $a_{mi} = \max_{j \geq i} |a_{ji}|$. The threshold $0 < u \leq 1$ determines the pivot in column i according to the condition $|a_{ii}| \geq u \cdot |a_{mi}|$. If this condition is fulfilled, the diagonal entry a_{ii} is chosen as the pivot; otherwise a_{mi} is used. The classical partial pivoting strategy is equivalent to $u = 1$. In this case a_{mi} or an equally large value will be selected as the pivot. Smaller values of u are preferable if a pre-ordered matrix exists so that choosing diagonal pivots is good for sparsity or parallelism. Naturally, this bears the risk of less numerical stability. Setting $u = 0$ means that the pivots on the diagonal will be chosen unless they are zero.

1.2.2 SuiteSparse 3.6.0

SuiteSparse is a collection of libraries for numerical problems concerning sparse matrices. UMFPACK, a part of SuiteSparse, offers solver routines for unsymmetric and symmetric SLSEs [7, 8, 9, 10], which is based on the Unsymmetric-pattern MulitFrontal method. The main step consists of the factorization of $(\underline{\underline{\mathbf{P}}} \cdot \underline{\underline{\mathbf{A}}} \cdot \underline{\underline{\mathbf{Q}}})$, $(\underline{\underline{\mathbf{P}}} \cdot \underline{\underline{\mathbf{R}}} \cdot \underline{\underline{\mathbf{A}}} \cdot \underline{\underline{\mathbf{Q}}})$ or $(\underline{\underline{\mathbf{P}}} \cdot \underline{\underline{\mathbf{R}}}^{-1} \cdot \underline{\underline{\mathbf{A}}} \cdot \underline{\underline{\mathbf{Q}}})$ into the product $\underline{\underline{\mathbf{L}}} \cdot \underline{\underline{\mathbf{U}}}$, where $\underline{\underline{\mathbf{P}}}$ and $\underline{\underline{\mathbf{Q}}}$ are permutation matrices, and $\underline{\underline{\mathbf{R}}}$ is a diagonal matrix of row scaling factors. $\underline{\underline{\mathbf{P}}}$ and $\underline{\underline{\mathbf{Q}}}$ aim at reduction of fill-in. $\underline{\underline{\mathbf{P}}}$, additionally, is designed to enhance the numerical accuracy. After the column pre-ordering that reduces fill-in, UMFPACK scales and analyzes the matrix in order to determine the strategy, symmetric or unsymmetric, for pre-ordering of rows and columns. Pivots with zero Markowitz cost are eliminated and placed in the LU factors [11,p. 73]. Then the remaining submatrix $\underline{\underline{\mathbf{S}}}$ is analyzed. In the unsymmetric case COLAMD [12, 13, 14, 15] is used to calculate the column pre-ordering of $\underline{\underline{\mathbf{S}}}$ which yields, in a first step, the symmetric permutation of the matrix $(\underline{\underline{\mathbf{S}}} \cdot \underline{\underline{\mathbf{S}}}^T)$ without evaluating this product. The Cholesky factors of $(\underline{\underline{\mathbf{S}}} \cdot \underline{\underline{\mathbf{Q}}})^T (\underline{\underline{\mathbf{S}}} \cdot \underline{\underline{\mathbf{Q}}})$ define an upper bound regarding the number of non-zero pattern of the factor $\underline{\underline{\mathbf{U}}}$ for the unsymmetric LU factorization - $\underline{\underline{\mathbf{P}}} \cdot \underline{\underline{\mathbf{S}}} \cdot \underline{\underline{\mathbf{Q}}} = \underline{\underline{\mathbf{L}}} \cdot \underline{\underline{\mathbf{U}}}$ - without knowing $\underline{\underline{\mathbf{P}}}$ which ensures a good choice for $\underline{\underline{\mathbf{Q}}}$. COLAMD is also responsible for the computation of the column elimination tree, the post-ordering of this tree, determination of an upper bound of non-zeros in the LU factors and has a different threshold for identifying dense rows and columns. In order to reduce fill-in, the column pre-ordering might be modified by reshuffling of columns within a single super-column node. The algorithm uses threshold partial pivoting with no preference given to the diagonal entry. If the condition $|a_{ij}| = 0.1 \cdot \max |a_{*j}|$ is fulfilled, an entry a_{ij} within a given pivot column j is chosen and the sparsest row of these is the pivot row. Following the strategy, the factorization of $\underline{\underline{\mathbf{A}}}$ breaks down into the factorization of a sequence of dense rectangular frontal matrices, whereby each frontal matrix is stored as node in the supernodal column elimination tree. Analogue to the factorization of the whole matrix by a unifrontal method, each chain of frontal matrices is factorized in a single working array. A frontal matrix can be understood as a Gaussian elimination of one or more columns. The frontal matrix for the elimination of a specific column of $\underline{\underline{\mathbf{A}}}$ is selected within the pre-analysis phase.

1.2.3 PARDISO 4.1

The third library of investigated SLSE solvers is called PARDISO [16, 17, 18], a pre-compiled, high-performance, robust and memory-efficient software. It is applicable for SLSE with large symmetric or non-symmetric coefficient matrices. To achieve better sequential and parallel performance, the solver relies on Level-3 BLAS update, and pipelining parallelism is exploited within left- and right-looking Level-3 BLAS supernode techniques. The supernode pivoting allows one to balance numerical stability and scalability during the factorization process.
In case of non-symmetric coefficient matrices the solver determines a permutation $\underline{\underline{\mathbf{P_{MPS}}}}$ and scaling matrices $\underline{\underline{\mathbf{D_r}}}$ and $\underline{\underline{\mathbf{D_c}}}$ in order to place large entries on the diagonal [18]. Before the calculation of the numerical factorization, a fill-in reducing permutation $\underline{\underline{\mathbf{P}}}$, based on the matrix $\underline{\underline{\mathbf{P_{MPS}}}} \cdot \underline{\underline{\mathbf{A}}} + (\underline{\underline{\mathbf{P_{MPS}}}} \cdot \underline{\underline{\mathbf{A}}})^T$, is computed. The parallel numerical factorization is made up of the above-defined permutation and scaling matrices,

$$\begin{aligned} \underline{\underline{\mathbf{Q}}} \cdot \underline{\underline{\mathbf{L}}} \cdot \underline{\underline{\mathbf{U}}} \cdot \underline{\underline{\mathbf{R}}} &= \underline{\underline{\mathbf{A_2}}}, \\ \underline{\underline{\mathbf{A_2}}} &= \underline{\underline{\mathbf{P}}} \cdot \underline{\underline{\mathbf{P_{MPS}}}} \cdot \underline{\underline{\mathbf{D_r}}} \cdot \underline{\underline{\mathbf{A}}} \cdot \underline{\underline{\mathbf{D_c}}} \cdot \underline{\underline{\mathbf{P}}}, \end{aligned}$$

where $\underline{\underline{\mathbf{Q}}}$ and $\underline{\underline{\mathbf{R}}}$ are supernode pivoting matrices. A pivoting pertubation strategy is applied in the case that the supernodes cannot be factorized with that strategy. The magnitude of the potential pivot is compared to a constant threshold $\alpha = \varepsilon \cdot ||\underline{\underline{\mathbf{A_2}}}||$ with the machine precision ε and the ∞-norm of the scaled and permuted matrix $\underline{\underline{\mathbf{A}}}$. To guarantee numerical stability the pivots are kept from getting too small by setting any tiny pivots l_{ii} to $\text{sign}(l_{ii}) \cdot \varepsilon ||\underline{\underline{\mathbf{A_2}}}||$. In practice diagonal elements are rarely modified for a large class of matrices. This pivoting approach generally yields not exact factorizations and an iterative refinement might be considered.

1.3 Benchmarks for the Library Routines

This section presents the results of the benchmarks of the three implemented library routines - SuperLU 4.0, PARDISO 4.1 and SuiteSparse 3.6.0 (see Section 1.2). A lot of effort has been put into the implementation of a handy and easy to use Fortran interface for the library routines. To validate the factorization time of the SLSE solvers, test cases with real and complex asymmetric coefficient matrices, which have different condition numbers $\kappa = (\left\|\underline{\underline{\mathbf{A}}}\right\|_1 \left\|\underline{\underline{\mathbf{A}}}^{-1}\right\|_1)^{-1}$, number of non-zeros and matrix sizes, are performed. For the generation of the test matrices the sprand function of MATLAB [19] and Octave[20], respectively, is exploited. To be able to perform such an extensive investigation, an automatization of the tests becomes inevitable. The speed of Gaussian elimination is determined by test cases which show the CPU time for solving a SLSE with different number of righthand sides (NRHS). Two independent runs of each test case scenario have have been performed so as to avoid systematical errors in the measurement of the CPU time. The statistical error, defined as the standard deviation of the mean [21,p. 815], is indicated by the error bars in the subsequent figures. Furthermore, the accuracy of a SLSE solver, defined as the relative precision

$\frac{\max\left(\left\|\underline{\mathbf{A}}\cdot\underline{\mathbf{x}}-\underline{\mathbf{b}}\right\|\right)}{\max(\|\underline{\mathbf{b}}\|)}$, is determined for the same set of coefficient matrices as in the case of factorization time. The benchmarks have been conducted on a Linux workstation with four CPUs (2593MHz) and a total memory of 33087116KB.

1.3.1 Test Cases for Real Asymmetric Coefficient Matrices

In this subsection the three library routines are tested against real asymmetric coefficient matrices with varying condition number, number of non-zeros and matrix size. The subsequent figures highlight the differences with regard to performance and accuracy. The lines between the measured data points are guiding lines for the eye and possess no further meaning.

The CPU times, required to solve SLSEs with coefficient matrices of size of $10^4 \times 10^4$, depending on number of non-zeros and condition numbers are shown in Figure 1.1. The highest influence on the CPU time is due to the factorization time. Thus, the factorization times of Sequential SuperLU, SuiteSparse with or without iterative refinement and PARDISO with four threads are compared. Furthermore, the upper axis indicating the condition number has no influence on the scaling of the horizontal axis. Comparing the performance of PARDISO with different number of threads, the best results are obtained with four threads under the constraints of the used test system. A more detailed view on the performance of PARDISO with different number of threads is given in Figure 1.2. The SuiteSparse solver routines with and without iterative refinement share the same factorization procedure. For this reason, the curves of SuiteSparse with and without iterative refinement fully overlap and cannot be distinguished. In the regime of number of non-zeros below 39502 the factorization times of all solver routines lie within an order of magnitude, whereby PARDISO reaches slightly worse results. The coefficient matrix with 39502 has the best condition number of this set of test cases. In this instance the three solver routines produce nearly the same result. For coefficient matrices with a larger number of nonzeros SuperLU reaches factorization times of an order of magnitude worse than PARDISO and SuiteSparse, whereby SuiteSparse yields slightly better results than PARDISO.

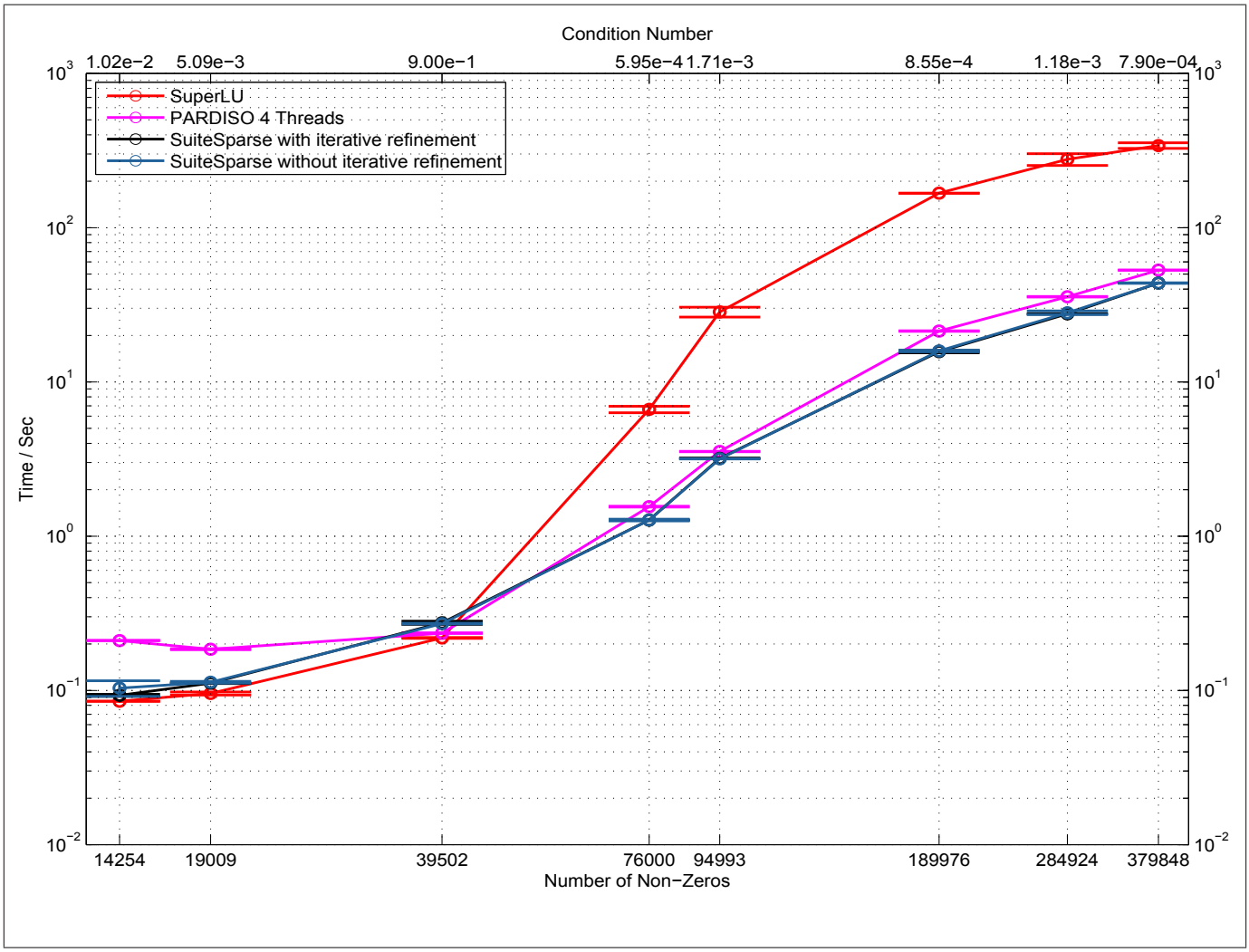

Figure 1.1: Performance of SLSE solvers for different coefficient matrices of size $10^4 \times 10^4$ with different number of non-zeros and condition number.

In Figure 1.2 one can observe that the performance of PARDISO increases with the number of threads, which is in good agreement with the supposed behaviour. The scaling factor between the different curves is approximately two to three. A number of four threads constitutes the limit of performance of the used testing system which possesses four CPUs.

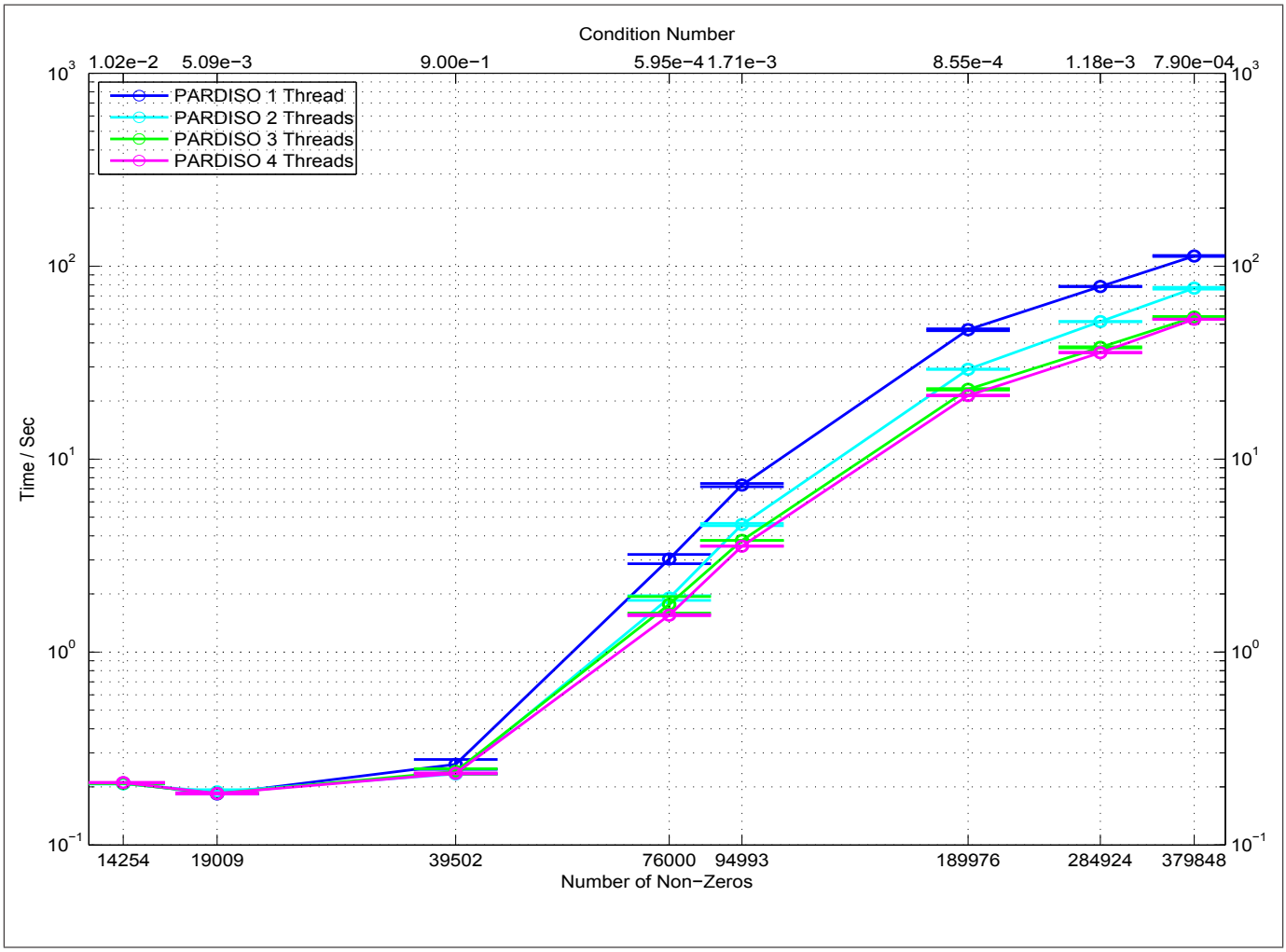

Figure 1.2: Performance of PARDISO solver routine with varying number of threads for different coefficient matrices of size $10^4 \times 10^4$ with different number of non-zeros and condition number.

Figure 1.3 illustrates the speed of SLSE solvers for a given coefficient matrix, the largest of the above-described factorization test case, and different NRHS. The resulting CPU time in the limit of a NRHS of one is equal to the factorization time. A remarkable fact is that the SuperLU shows the flattest slope in comparison to the other solver routines. Although the PARDISO solver routine has a factorization time of an order of magnitude better than SuperLU, it intersects the red SuperLU curve at a NRHS of 1000. The SuiteSparse without iterative refinement has a flatter slope than the version with iterative refinement and reaches the best absolute CPU times for different NRHS up to a value of 1000.

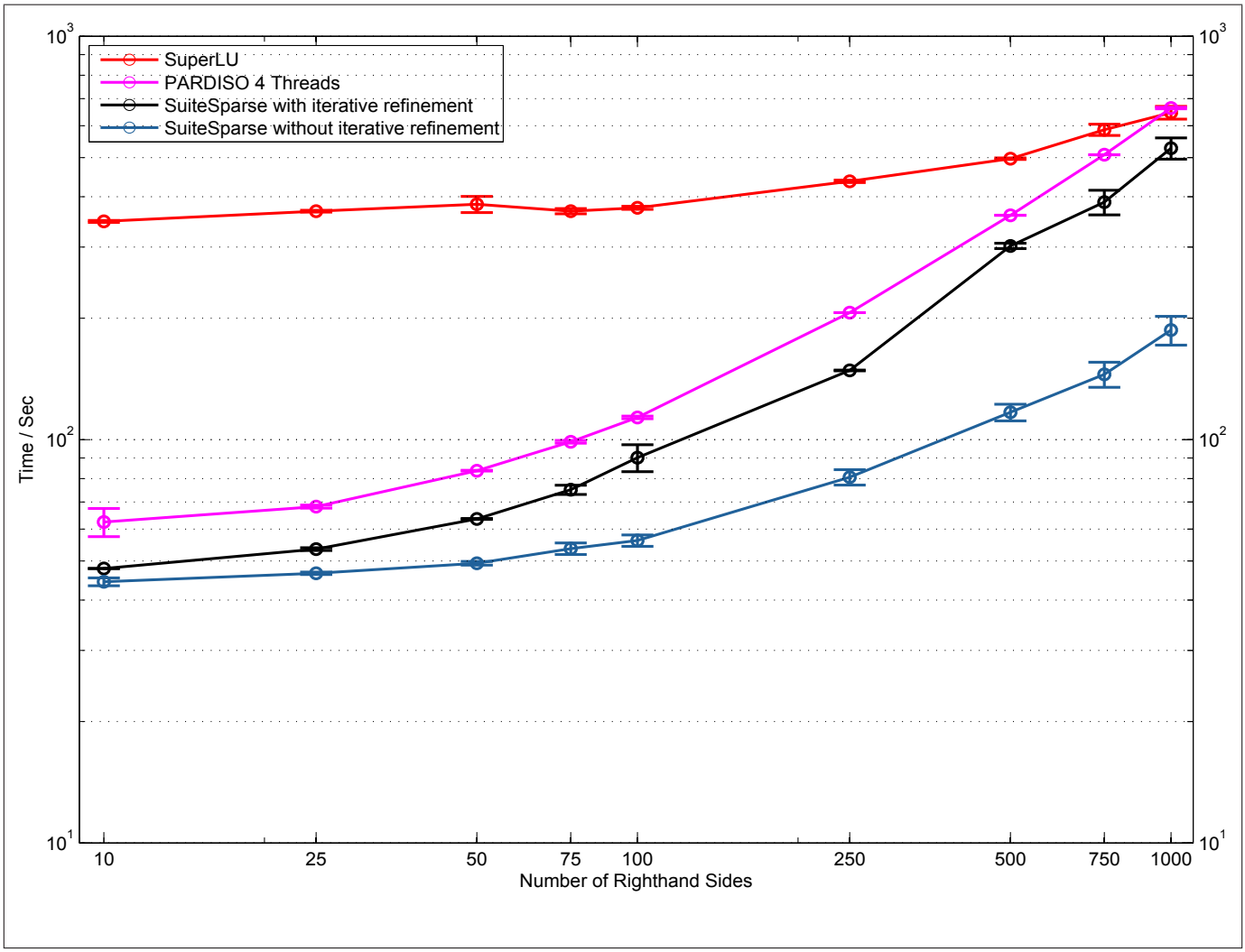

Figure 1.3: Performance of SLSE solvers for SLSEs with different number of righthand sides and a coefficient matrix of size $10^4 \times 10^4$ with 379848 non-zero entries and a condition number of 7.90e-4.

The accuracies of the solver routines for the same set of coefficient matrices as in the test case for the factorization time are shown in Figure 1.4. The curves and data points of SuperLU, PARDISO, SuiteSparse with and without iterative refinement partially overlap. PARDISO and SuiteSparse with iterative refinement display similar accuracy, wherby PARDISO has a bit better results. SuperLU and SuiteSparse without iterative refinement yield a slightly worse accuracy than the other two solver routines, but the produced values are still within one order of magnitude.

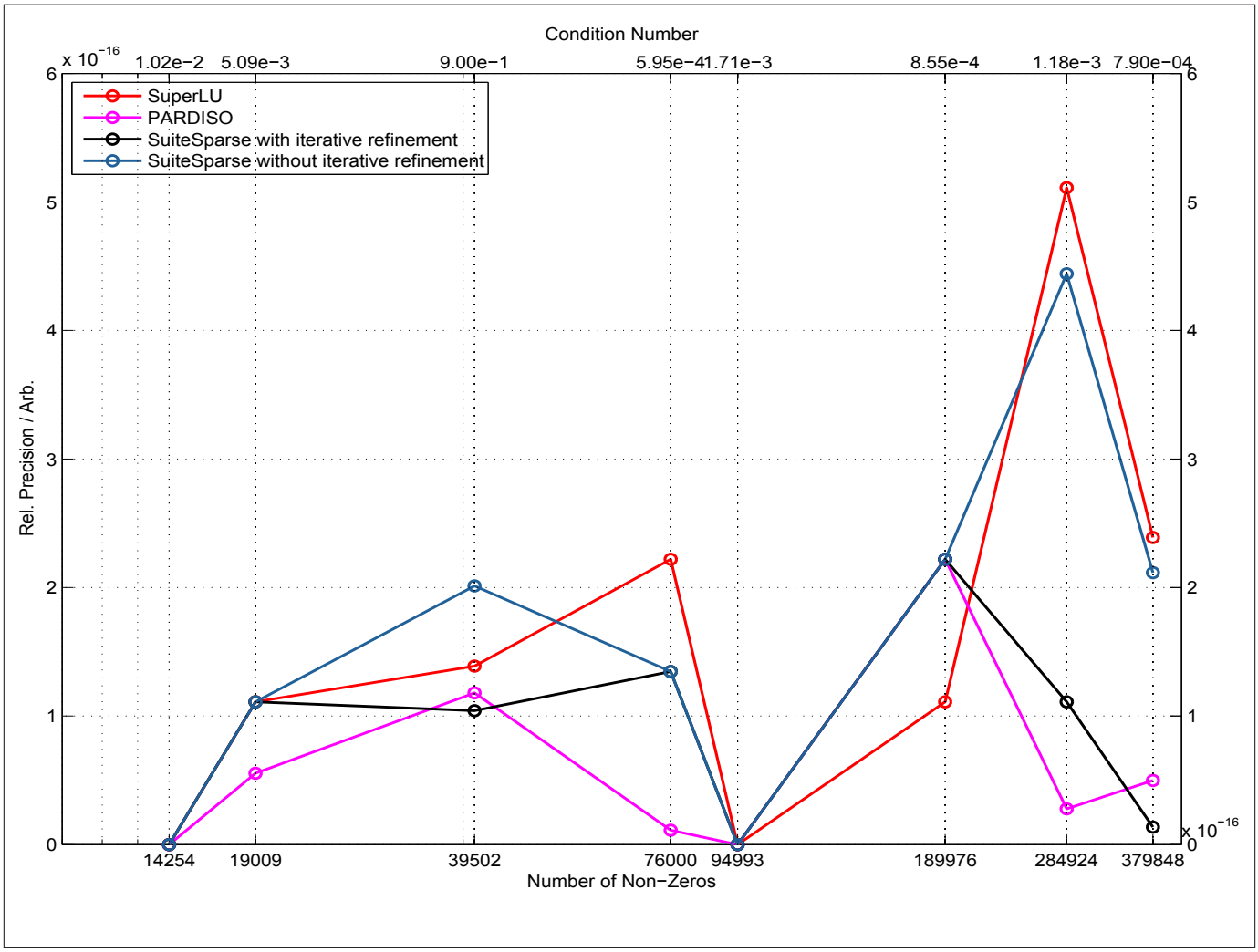

Figure 1.4: Accuracy of SLSE solvers for different coefficient matrices of size $10^4 \times 10^4$ with different number of non-zeros and condition number.

The factorization time for coefficient matrices with slightly increasing number of non-zeros and very varying condition number is apparent in Figure 1.5. In case of SuperLU the last data point is missing, because the corresponding test cases have been aborted after two hours. Contrary to SuperLU, PARDISO manages all test cases, but the factorization times differ by more than one order of magnitude. The curves and data points of SuiteSparse with and without iterative refinement fully overlap. The outstanding performance of SuiteSparse is evident from this figure.

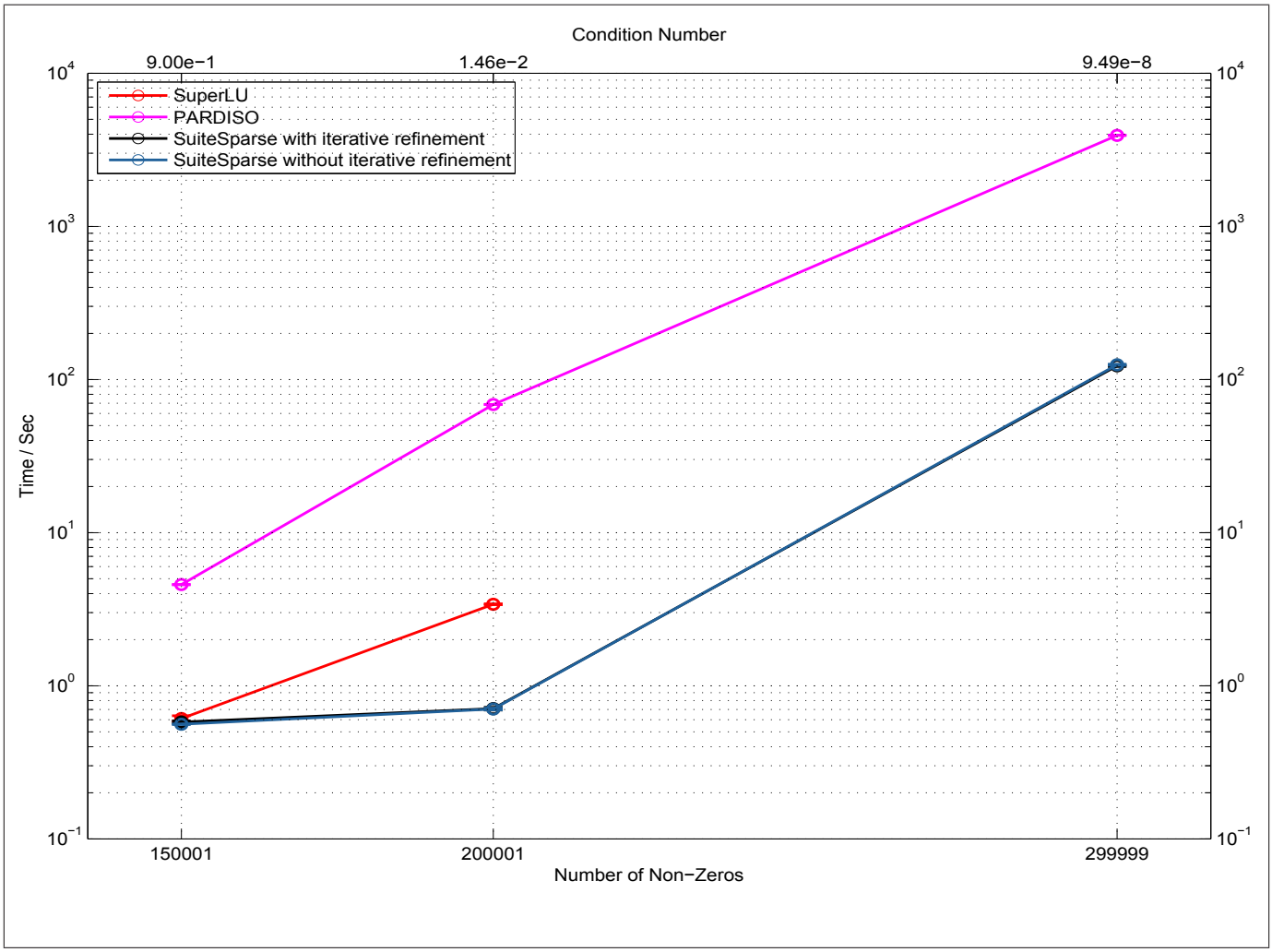

Figure 1.5: Performance of SLSE solvers for different coefficient matrices of size $10^5 \times 10^5$ with different number of non-zeros and bad condition number.

In Figure 1.6 the accuracies corresponding to the Figure 1.5 are shown. An image detail of Figure 1.6 showing the relative precision on a shorter length scale is illustrated in Figure 1.7. Neglecting SuperLU due to the missing data point, one notices the similar accuracy obtained by PARDISO and SuiteSparse with iterative refinement. The accuracy of the SuiteSparse version without iterative refinement is three orders of magnitude worse.

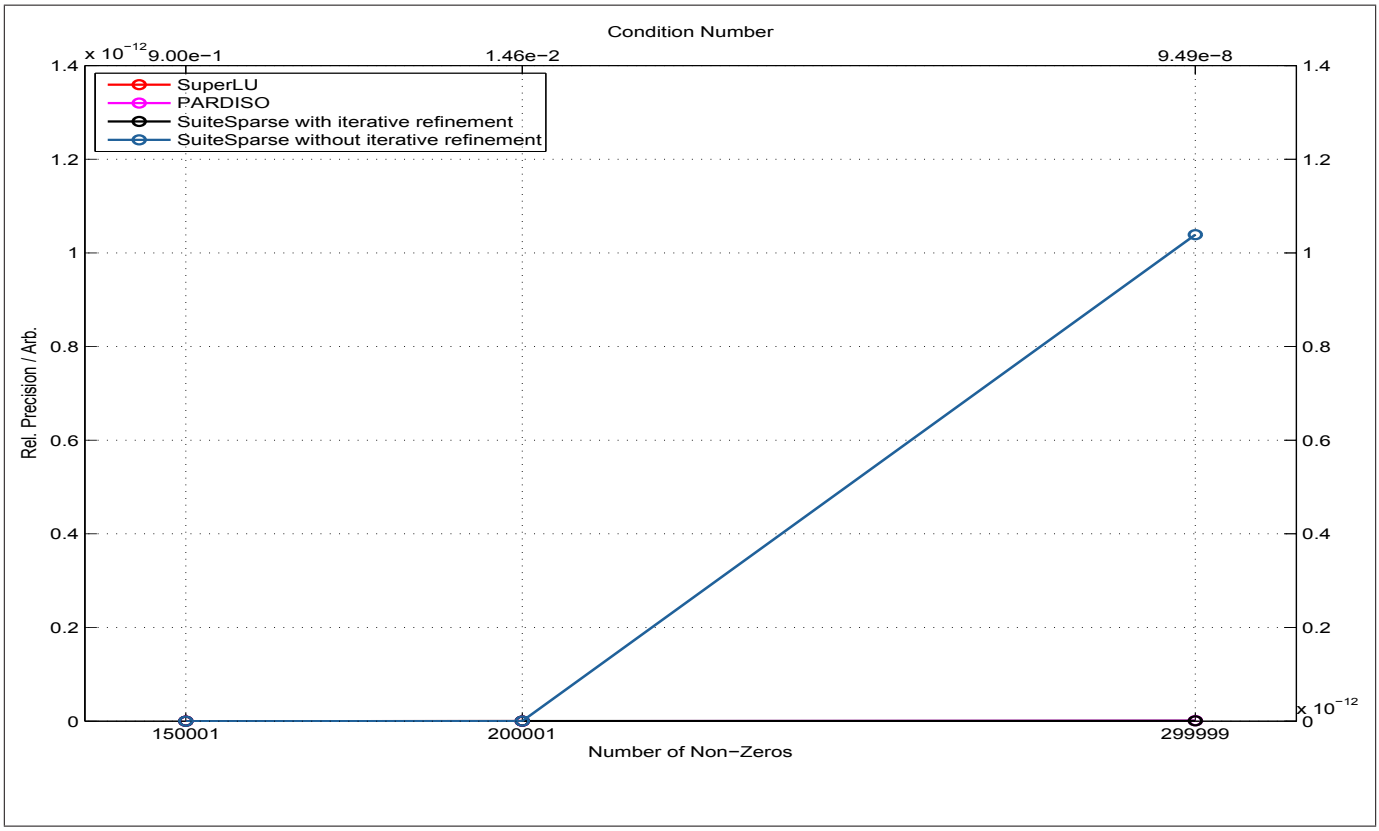

Figure 1.6: Accuracy of SLSE solvers for different coefficient matrices of size $10^5 \times 10^5$ with different number of non-zeros and bad condition number.

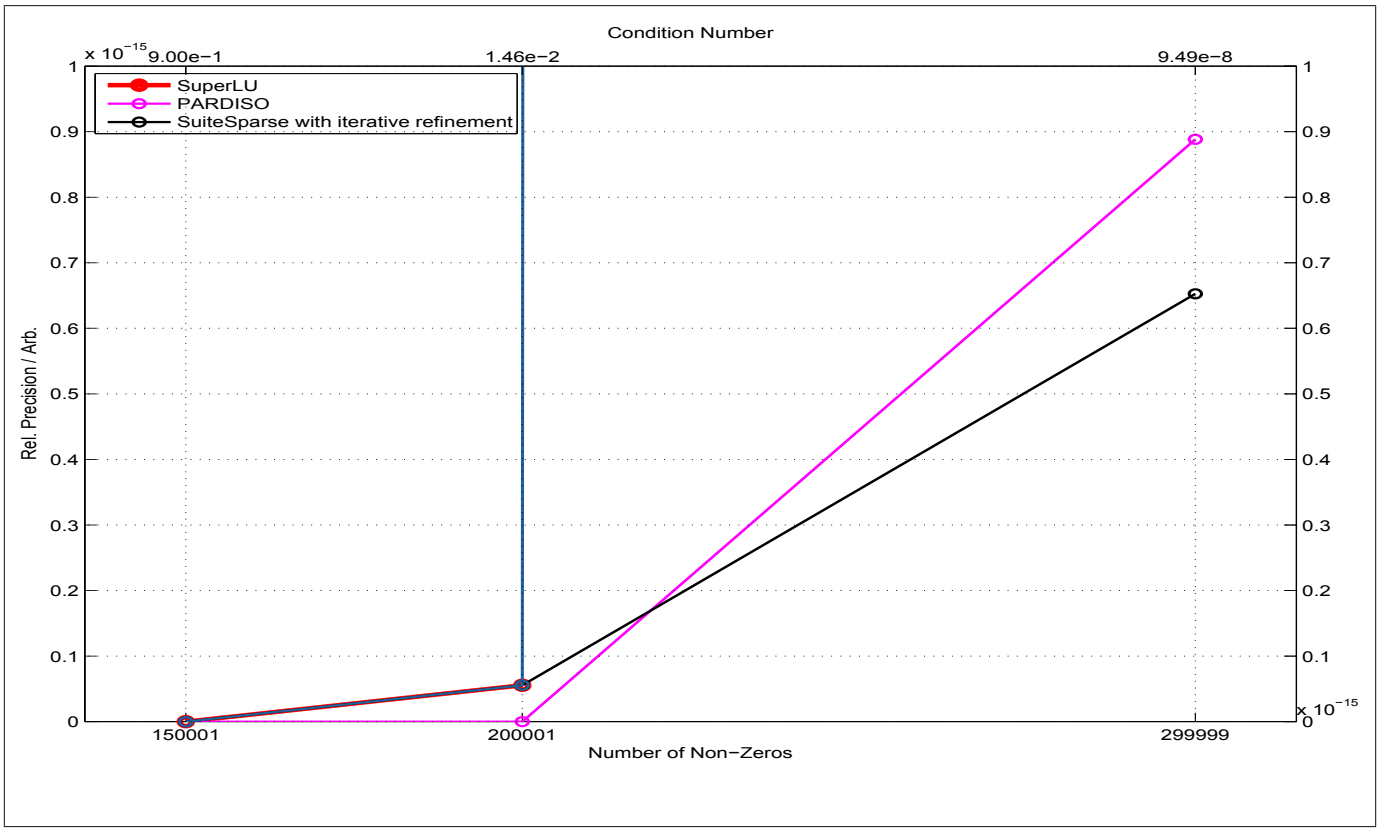

Figure 1.7: Image detail showing the accuracy of SLSE solvers for different coefficient matrices of size $10^5 \times 10^5$ with different number of non-zeros and bad condition number.

1.3.2 Test Cases for Complex Asymmetric Coefficient Matrices

In this subsection the performance and the accuracy of the three library routines are tested with respect to complex asymmetric coefficient matrices with varying condition number, number of non-zeros and matrix size. These differences with regard to performance and accuracy are presented in the subsequent figures. The lines between the measured data points are guiding lines to the eye.
The CPU time, required to solve SLSEs with a complex coefficient matrices of size of $10^4 \times 10^4$, dependent on number of non-zeros and condition number is shown in Figure 1.8. the factorization time has again the highest influence on the CPU time. Thus, the factorization time of SuperLU, PARDISO with four threads and SuiteSparse with iterative refinement can be compared. Furthermore, the upper axis indicating the condition number has no influence on the scaling of the horizontal axis. Comparing the performance of PARDISO with different number of threads, the best results are obtained with four threads under the constraints of the used test system. A more detailed view on the performance of PARDISO with different number of threads is given in Figure 1.9. The curves of the SuiteSparse solver routines with and without iterative refinement overlap in the figure, because they share the same factorization procedure. It is apparent from the figure that the plotted curves exhibit a similar shape which allows a ranking of solver routines. The SuiteSparse indicates the best performance for this set of test cases. PARDISO claims the second rank with a slightly worse performance than SuiteSparse and SuperLU yields a factorization speed which is a factor 10 slower.

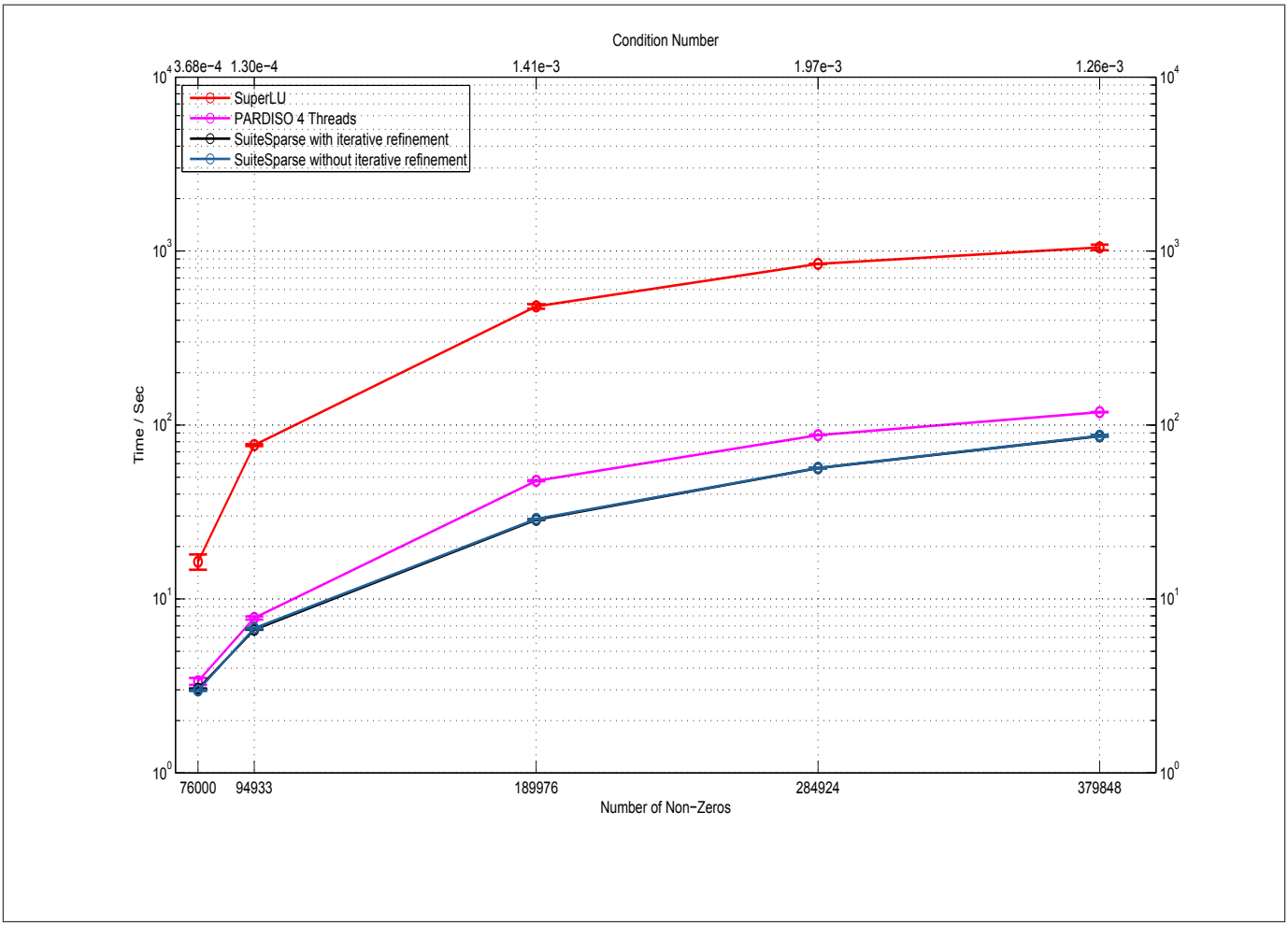

Figure 1.8: Performance of SLSE solvers for different coefficient matrices of size $10^4 \times 10^4$ with different number of non-zeros and condition number.

In Figure 1.9 one can observe the supposed dependence of CPU time on the used number of threads, namely that the performance of PARDISO increases with the number of threads. The scaling factor between the different curves is again approximately two to three. A number of four threads constitutes again the limit of performance of the used testing system which possesses four CPUs.

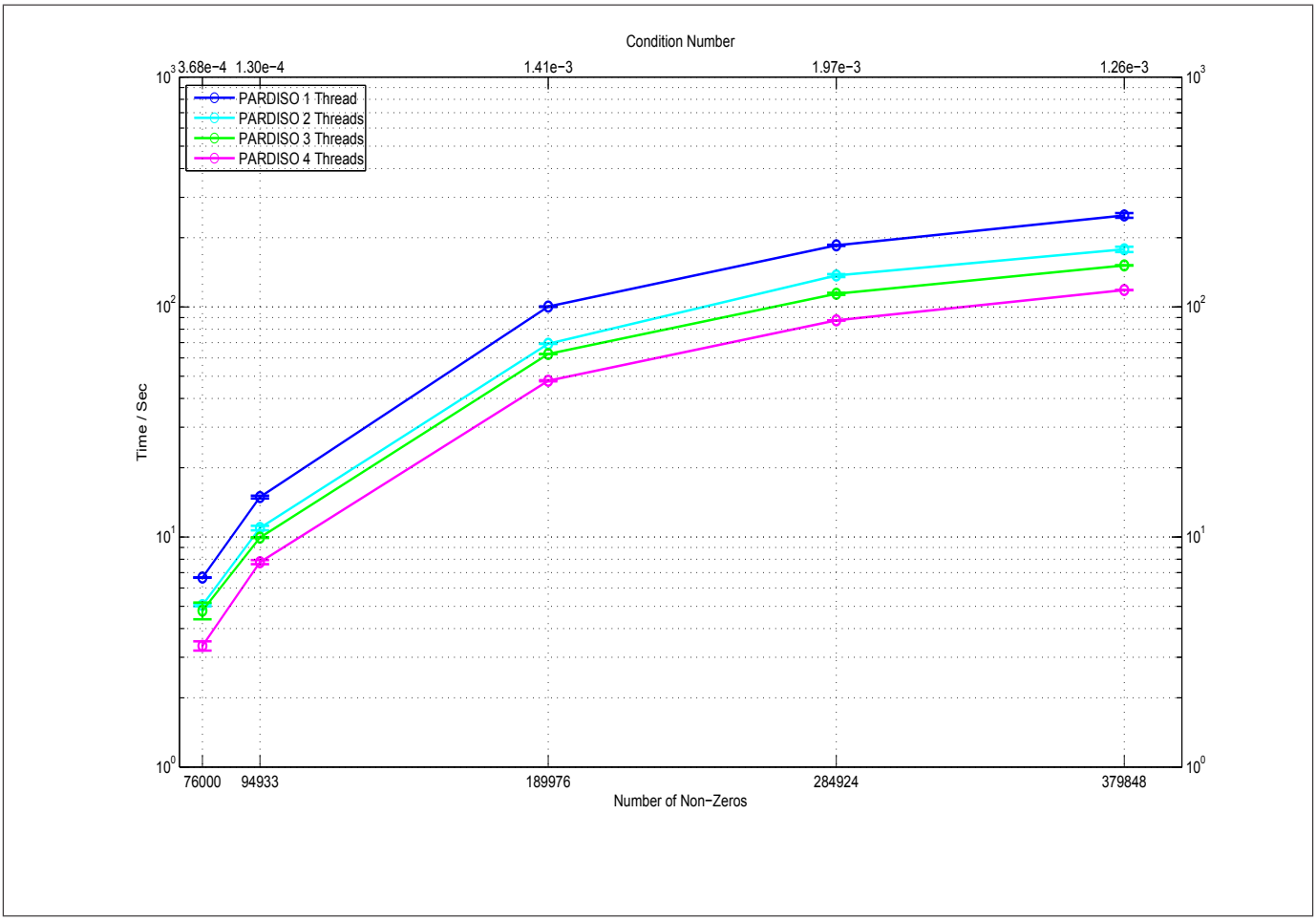

Figure 1.9: Performance of PARDISO solver routine with varying number of threads for different coefficient matrices of size $10^4 \times 10^4$ with different number of non-zeros and condition number.

Figure 1.10 illustrates the speed of SLSE solvers for a given complex coefficient matrix, the largest of the above-described factorization test case, and different NRHS. The resulting CPU time in the limit of a NRHS of one reflects the factorization time. SuperLU shows again the flattest slope in comparison to the other solver routines. Although the PARDISO solver routine has factorization time of an order of magnitude better than SuperLU, it nearly intersects the red SuperLU curve at a NRHS of 1000. The SuiteSparse without iterative refinement has once more a flatter slope than the version with iterative refinement and reaches the best absolute CPU times for different NRHS up to a value of 1000.

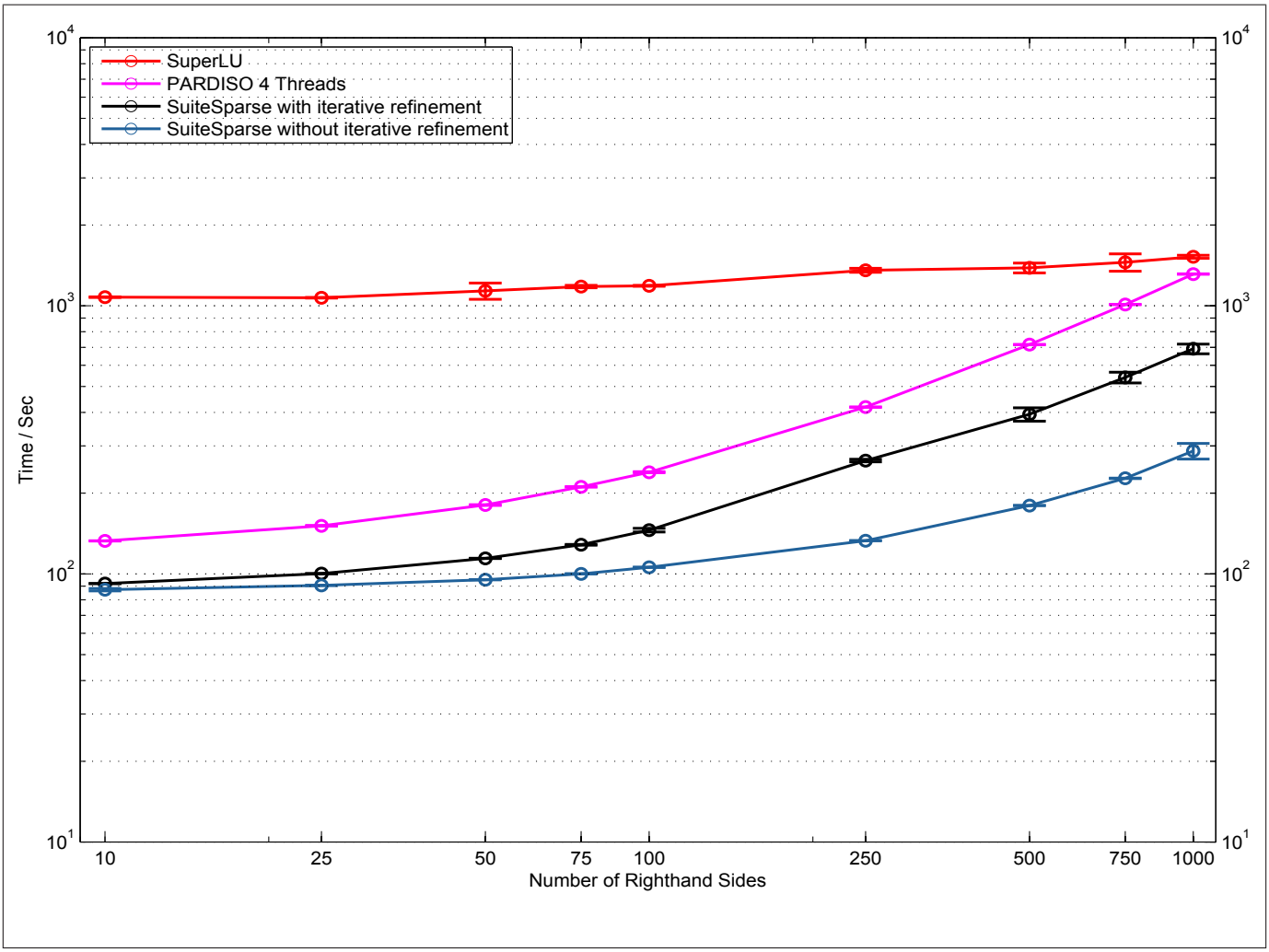

Figure 1.10: Performance of SLSE solvers for SLSEs with different number of righthand sides and a coefficient matrix of size $10^4 \times 10^4$ with 379848 non-zero entries and a condition number of 1.26e-3.

The accuracies of the solver routines for the same set of coefficient matrices as in the test case for the factorization time are shown in Figure 1.11. An image detail of Figure 1.11 showing the relative precision on a shorter length scale is illustrated in Figure 1.12. PARDISO and SuiteSparse with iterative refinement display again similar accuracy, wherby PARDISO has a bit better results. SuperLU and SuiteSparse without iterative refinement yield an accuracy which is three orders of magnitude worse than the accuracy of the other two solver routines.

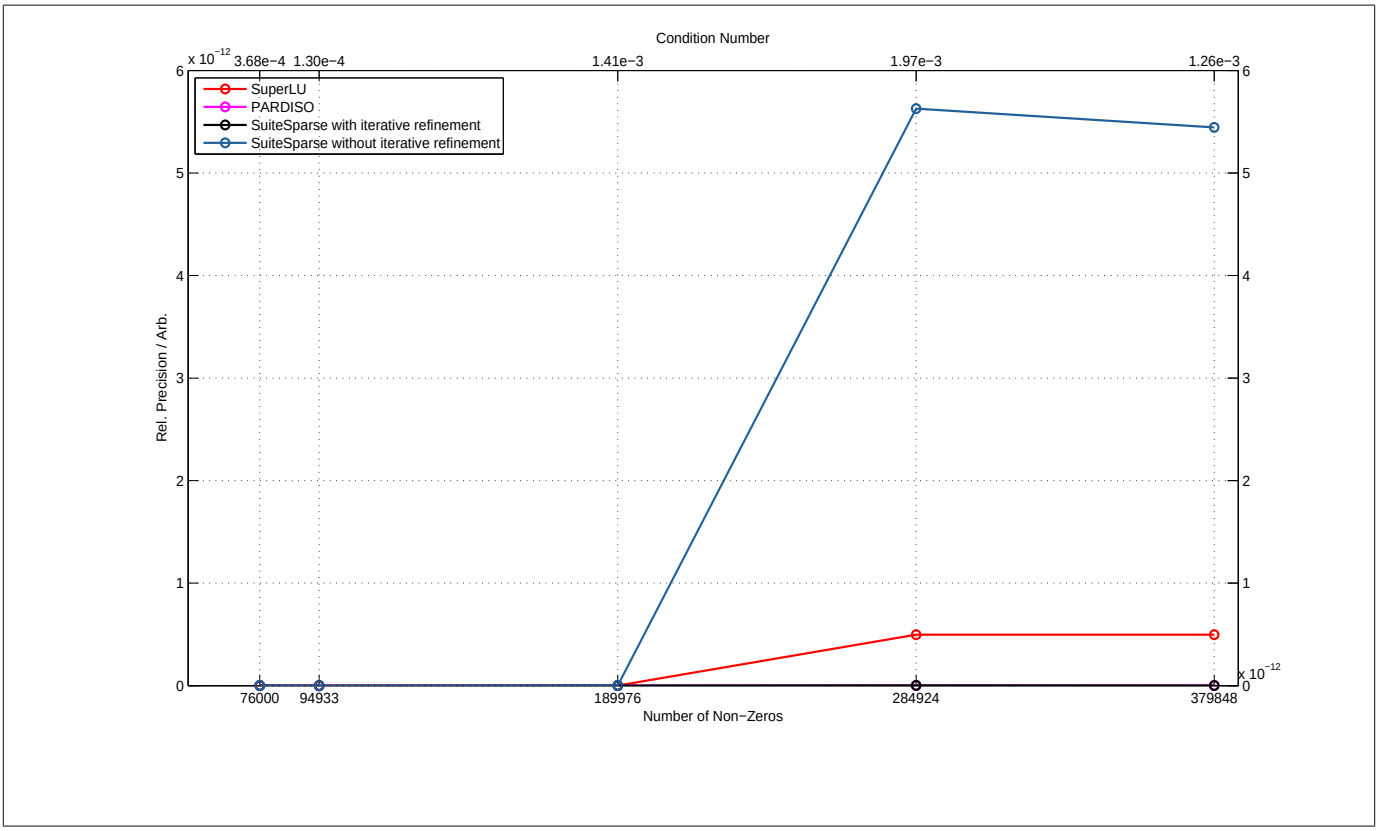

Figure 1.11: Accuracy of SLSE solvers for different coefficient matrices of size $10^4 \times 10^4$ with different number of non-zeros and condition number.

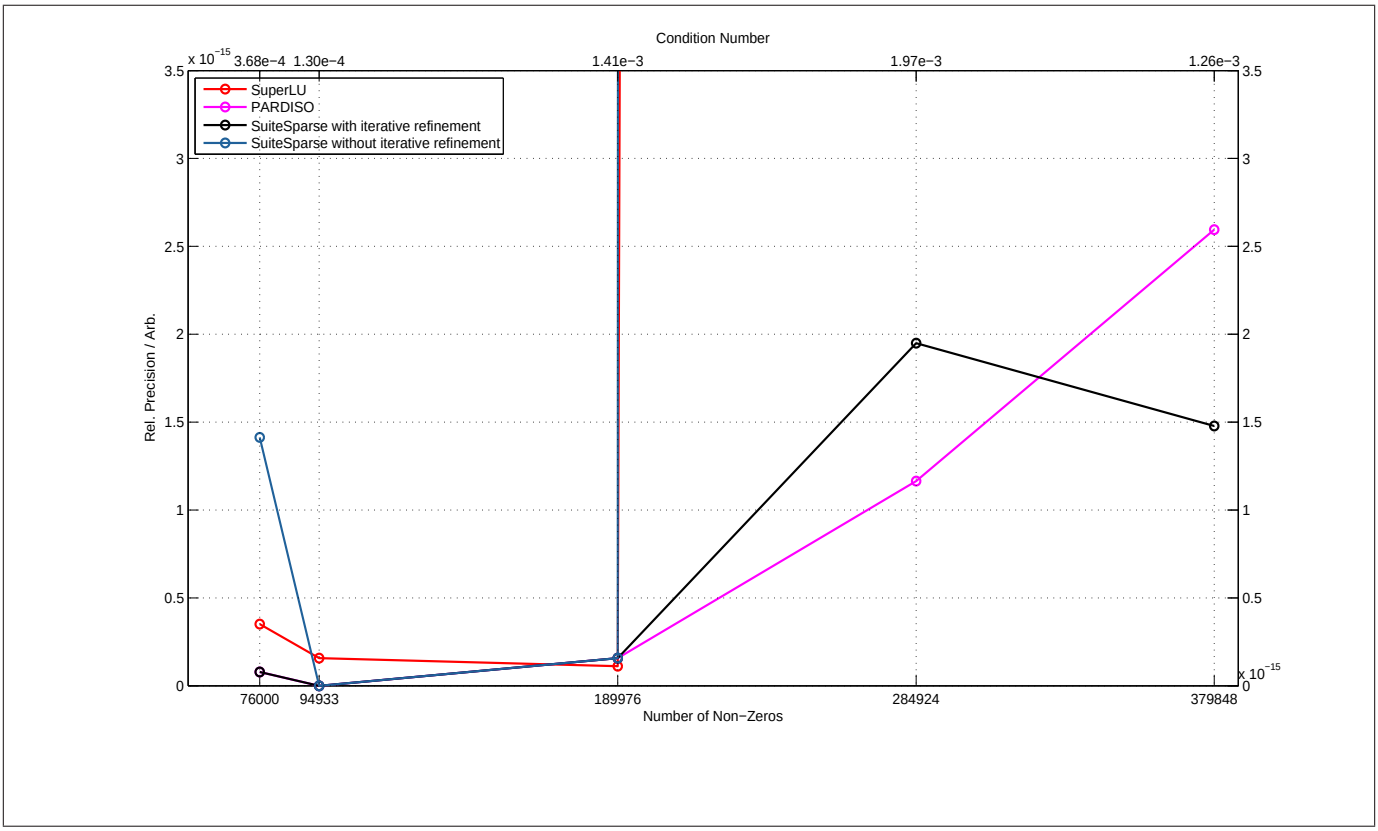

Figure 1.12: Image detail showing the accuracy of SLSE solvers for different coefficient matrices of size $10^4 \times 10^4$ with different number of non-zeros and condition number.

1.3.3 Conclusion

Surprisingly, the parallelized solver routines of SuperLU and PARDISO do not yield the best performance results for the studied asymmetric coefficient matrices. The results for the multi-threaded version of SuperLU have been omitted, since a notable number of test cases could not be performed due to unsettled issues. In addition, no further releases of the multi-threaded SuperLU are available - the latest version is from 2007 - and one might give preference to the distributed version of SuperLU which is much more complex to implement. Investigations on large sparse, symmetric linear systems of equations, as discussed in the article of Gould et al. [22], exhibit a different behaviour. In this case the PARDISO solver routine demonstrates an overwhelming performance. Concerning asymmetric coefficient matrices, it seems that the results from earlier performed benchmarks [16] are outdated because the comparison involves a SuiteSparse version with UMFPACK 3. In this thesis, the for the moment latest versions - UMFPACK 5.5.0, SuperLU 4.0 and Pardiso 4.1 - are compared with each other. Recapitulating, the best factorization time is obtained using SuiteSparse which becomes important when calculating steady-state solutions, see Section 2.3.3. Furthermore, the SuiteSparse without iterative refinement yields the best performance for processing multiple righthand sides up to a NRHS of 1000; despite, one has to mention that the best scalability is reached by SuperLU. Although turning off the iterative refinement worsens the accuracy by up to three orders of magnitude, the SuiteSparse without iterative refinement is used in all further computations since a relative precision of 10^{-12} is more than sufficient for each of our prospective applications. Depending on the individual demands, one might come to other conclusions.

2 A Conservative Finite Difference Scheme for General Diffusion Equations in one Dimension

Many interesting physical problems in the field of fluid dynamics can be formulated as conservation laws. This thesis investigates transport processes in fusion plasmas which are mathematically described by a general diffusion equation. In order to be able to apply a numerical scheme, one has to rewrite the one dimensional (1D) general diffusion equation in a conservative form, as done in Section 2.1. A good introduction into this topic is provided in the books of Versteeg and Malalasekera [23] and Ferziger and Peric [24]. The fundamental concepts behind the conservative finite difference scheme (CFDS) developed within this chapter are analogue to the well-known and wide-spread FVM in computational fluid dynamics. Discrepancies between classical FVM and the described scheme arise from the new approach for the interpolation of the numerical fluxes at the cell boundaries, see Section 2.2, which is a crucial part dominating the performance of the code. Solving the time-dependent general diffusion equation requires, besides a suitable time integration procedure, the knowledge of consistency, stability and convergence of a numerical scheme. The concepts and derived relations are presented in Section 2.3. Before the developed scheme can be applied to test cases, Section 2.4 outlines the implementation of boundary conditions in the CFDS. Section 2.5 finally shows results about the stability of the CFDS that is used to solve a general diffusion equation which generates a Gaussian profile as a solution. In Chapter 3, the presented concepts are extended to 2D meshes which can be refined adaptively.

2.1 Conservative Formulation of a one dimensional General Diffusion Equation

A general form of a diffusion equation in 1D is given by

$$\frac{\partial f(\eta,t)}{\partial t} = \frac{\partial}{\partial \eta}\left(D(\eta)\frac{\partial f(\eta,t)}{\partial \eta} - v(\eta) f(\eta,t)\right) + q(\eta), \tag{2.1}$$

where $D(\eta)$ describes the diffusivity and accordingly $v(\eta)$ the velocity of convection or advection. The last term $q(\eta)$ stands for sources or drains in the system. As described in LeVeque [25,pp. 15-46], for a further numerical treatment the domain of solution is subdivided into N cells. Then Equation (2.1) is integrated over the i-th cell, $\eta \in [\eta_{i-1}, \eta_i]$. With the cell-volume, $F_i(t) = \int_{\eta_{i-1}}^{\eta_i} \mathrm{d}\eta\ f(\eta,t)$, and interchange

of differentiation and integration Equation (2.1) becomes

$$\frac{\partial F_i(t)}{\partial t} = \Gamma(\eta_{i-1},t) - \Gamma(\eta_i,t) + Q_i \tag{2.2}$$

with

$$\Gamma(\eta,t) = -D(\eta)\frac{\partial f(\eta,t)}{\partial \eta} + v(\eta) f(\eta,t) \tag{2.3}$$

and

$$Q_i = \int_{\eta_{i-1}}^{\eta_i} \mathrm{d}\eta \; q(\eta).$$

Thus, the value of cell-volume $F_i(t)$ can only change by numerical fluxes $\Gamma(\eta,t)$ at the boundaries η_{i-1} and η_i or by a source or drain Q_i. Integrating Equation (2.1) over the total domain of solution leads to a conservation law,

$$\begin{aligned} \frac{\partial F(t)}{\partial t} &= \underbrace{\Gamma(\eta_0,t) - \Gamma(\eta_\mathrm{N},t)}_{\to 0} + \underbrace{Q}_{\to 0} \\ &= 0 \end{aligned} \tag{2.4}$$

with

$$F(t) = \int_{\eta_0}^{\eta_\mathrm{N}} \mathrm{d}\eta \; f(\eta,t)$$

and

$$Q(t) = \int_{\eta_0}^{\eta_\mathrm{N}} \mathrm{d}\eta \; q(\eta,t).$$

The relation (2.4) is valid if all sources balance the drains and the numerical fluxes vanish at the boundaries of the domain of solution, i.e. a closed system.

2.2 Polynomial Reconstruction of the Numerical Fluxes at the Cell Boundaries in one Dimension

In order to evaluate the numerical fluxes at the cell boundaries, the function $f(\eta,t)$ and its derivative must be known at the boundaries. In the conservative formulation, described in Section 2.1, instead of the function f the cell volumes $F_i(t)$ are calculated. To reconstruct the function f from the cell volumes $F_i(t)$

[26], we expand f at boundary η_i into a Taylor series up to the order n,

$$\begin{aligned} {}_if(\eta,t) &= f^{(0)}(\eta_i,t)+f^{(1)}(\eta_i,t)(\eta-\eta_i)+\frac{f^{(2)}(\eta_i,t)}{2}(\eta-\eta_i)^2+\frac{f^{(3)}(\eta_i,t)}{6}(\eta-\eta_i)^3+\cdots \\ &= {}_ia_0(t)+{}_ia_1(t)(\eta-\eta_i)+{}_ia_2(t)(\eta-\eta_i)^2+{}_ia_3(t)(\eta-\eta_i)^3+\cdots \\ &= \sum_{j=0}^{n} {}_ia_j(t)\,(\eta-\eta_i)^j. \end{aligned} \tag{2.5}$$

The left subscripts, as in Equation (2.5), are used to highlight the expansion at a certain boundary and the elements ${}_ia_j(t)$ abbreviate the corresponding polynomial coefficients of the Taylor series. Integration of Equation (2.5) over the k-th cell, $\eta \in [\eta_{k-1},\eta_k]$, and subsequent interchange of integration und summation yields an expansion for the cell volume $F_k(t)$ at boundary η_i,

$$\begin{aligned} {}_iF_k(t) &= \int_{\eta_{k-1}}^{\eta_k} \mathrm{d}\eta \sum_{j=0}^{n} {}_ia_j(t)\,(\eta-\eta_i)^j \\ &= \sum_{j=0}^{n} {}_ia_j(t)\;{}_iA_{kj} \end{aligned} \tag{2.6}$$

with

$${}_iA_{kj} = \frac{(\eta_k-\eta_i)^{j+1}-(\eta_{k-1}-\eta_i)^{j+1}}{j+1},$$

whereby the matrix ${}_iA_{kj}$ weights the polynomial coefficients and is only dependent on the generated computational grid. Based on Equation (2.6), one demands that the cell volumes ${}_iF_k(t)$ within a defined stencil are approximated by a polynomial of order n that is expanded at boundary η_i. In this context a stencil is a subset of points of the computational grid which is used to reconstruct the function at a particular boundary. This allows us to set up a system of linear equations which is used to determine the unknown polynomial coefficients at a certain boundary. The number of equations, required to obtain a well-defined set of equations, complies with the selected order n of the Taylor expansion. One has to choose between a variety of stencils which strongly differ in the condition number of the created linear system of equations. A severe restriction, ensuring a good condition number, is the demand for a symmetric stencil around the boundary. Recapitulating, the value of function $f(\eta,t)$ and its derivatives can be reconstructed at the boundaries including a particular truncation error. In the developed numerical scheme the polynomial coefficents ${}_ia_j(t)$ are never calculated explicitly; instead of them, one computes the inverse of ${}_iA_{ij}$, ${}_iB_{jk}$, which is independent of time and can be used to determine the ${}_ia_j(t)$ from the cell volumes,

$${}_ia_j(t) = \sum_{\substack{k \in \text{ symmetric stencil} \\ \text{of boundary } \eta_i}} {}_iB_{jk}\;{}_iF_k(t) \tag{2.7}$$

with

$$\delta_{ik} = \sum_{j=0}^{n} {}_iA_{ij}\;{}_iB_{jk}, \tag{2.8}$$

whereby δ_{ik} denotes the *Kronecker delta*. Since the inverse ${}_iB_{jk}$ links the polynomial coefficents with the cell volumes ${}_iF_k(t)$, it is possible to formulate a scheme solving Equation (2.1) which only makes use of the ${}_iF_k(t)$ within the stencil. For instance an appropriate third order approximation of function $f(\eta,t)$ at boundary η_i, which requires a four-cell stencil, is depicted in Figure 2.1.

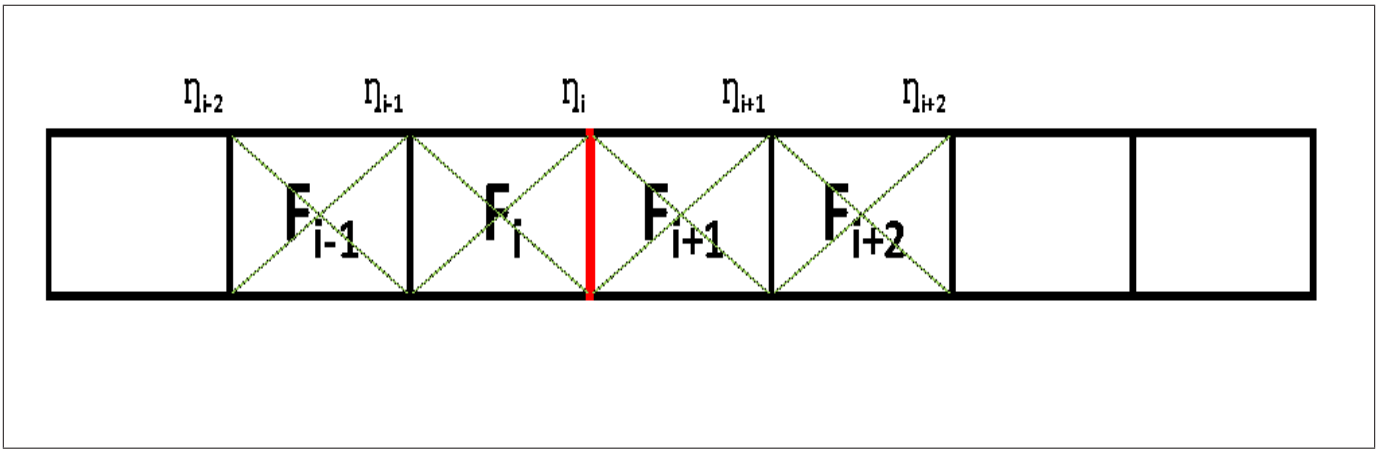

Figure 2.1: Four-cell stencil for third order approximation of function $f(\eta,t)$ at boundary η_i.

Before proceeding with the derivation of the numerical flux function $\Gamma(\eta,t)$ with polynomial reconstruction of function values, one has to discuss the need for a labeling of the cell volumes ${}_iF_k(t)$ with a left subscript. Independent of the expansion at a certain boundary, Equation (2.6) shall always yield the same value of cell volume up to a known truncation error. This is an intuitive demand for the polynomial reconstruction because it makes no sense that a specific cell volume is approximated by an expansion at two or more different boundaries with unequal values. Therefore, we are able to omit the left subscript distinguishing the cell volumes. Bearing this in mind, the polynomial coefficients for the above-depicted stencil (see Figure 2.1) can be calculated with the subsequent formula,

$$ {}_ia_j(t) \;=\; \sum_{k=i-1}^{i+2} {}_iB_{jk}\, F_k(t). \tag{2.9} $$

Using expressions (2.7) and (2.8), one is able to evaluate the numerical flux function $\Gamma(\eta,t)$ (2.3) at boundary η_i,

$$ \Gamma(\eta_i,t) \;=\; -D(\eta_i)\,\frac{\partial f(\eta,t)}{\partial \eta}\bigg|_{\eta_i} + v(\eta_i)\,f(\eta_i,t), $$

and to express the upcomming values of the function $f(\eta,t)$ and its first derivative in terms of cell volumes

$F_{\tilde{k}}(t)$ which belong to a chosen stencil,

$$\begin{aligned} f(\eta_i,t) &= {}_ia_0(t) \\ &= \sum_{\substack{k \in \text{ symmetric stencil} \\ \text{of boundary } \eta_i}} {}_iB_{0k}\, F_k(t) \end{aligned}$$

and

$$\frac{\partial f(\eta,t)}{\partial \eta} = \sum_{j=1}^{n} {}_ia_j(t)\; j\; (\eta-\eta_i)^{j-1}$$

with

$$\begin{aligned} \left.\frac{\partial f(\eta,t)}{\partial \eta}\right|_{\eta_i} &= {}_ia_1(t) \\ &= \sum_{\substack{k \in \text{ symmetric stencil} \\ \text{of boundary } \eta_i}} {}_iB_{1k}\, F_k(t). \end{aligned}$$

For the computational treatment it is advantageous to subsume all contributions from the numerical flux differences in a matrix $\underline{\underline{\mathbf{M}}}$ which allows us to rewirte Equation (2.2) in the form of a vector $\underline{\mathbf{F}}(t)$ containing the elements $F_i(t)$ and a vector $\underline{\mathbf{Q}}$ containing the elements Q_i,

$$\frac{\partial \underline{\mathbf{F}}(t)}{\partial t} = \underline{\underline{\mathbf{M}}} \cdot \underline{\mathbf{F}}(t) + \underline{\mathbf{Q}}. \tag{2.10}$$

The i-th row of the matrix $\underline{\underline{\mathbf{M}}}$ can be reconstructed from the numerical flux differences $I_i(t)$ of the i-th cell,

$$\begin{aligned} I_i(t) &= \Gamma(\eta_{i-1},t) - \Gamma(\eta_i,t) \\ &= \sum_{\substack{k \in \text{ symmetric stencil} \\ \text{of boundary } \eta_{i-1}}} \left(-D(\eta_{i-1})\; {}_{i-1}B_{1k} + v(\eta_{i-1})\; {}_{i-1}B_{0k}\right)\; F_k(t) \; - \\ &\quad - \sum_{\substack{k' \in \text{ symmetric stencil} \\ \text{of boundary } \eta_i}} \left(-D(\eta_i)\; {}_iB_{1k'} + v(\eta_i)\; {}_iB_{0k'}\right)\; F_{k'}(t), \end{aligned}$$

whereby indices k and k' of the corresponding stencil determine the column indices of the entries of $\underline{\underline{\mathbf{M}}}$. In order to illustrate the proposed procedure for the generation of $\underline{\underline{\mathbf{M}}}$, one considers once again the stencil in Figure 2.1. The needed values of function $f(\eta,t)$,

$$f(\eta_i,t) = \sum_{k=i-1}^{i+2} {}_iB_{0k}\; {}_iF_k(t),$$

and the values of its derivative,

$$\left.\frac{\partial f(\eta,t)}{\partial \eta}\right|_{\eta_i} = \sum_{k=i-1}^{i+2} {}_iB_{1k}\ {}_iF_k(t),$$

are computed according to Equation (2.9). Hence, one can assemble the numerical flux differences $I_i(t)$ of the i-th cell,

$$\begin{aligned} I_i(t) = & \sum_{k=i-2}^{i+1} \left(-D(\eta_{i-1})\ {}_{i-1}B_{1k} + \nu(\eta_{i-1})\ {}_{i-1}B_{0k}\right)\ F_k(t) - \\ & - \sum_{k=i-1}^{i+2} \left(-D(\eta_i)\ {}_iB_{1k} + \nu(\eta_i)\ {}_iB_{0k}\right)\ F_k(t). \end{aligned} \tag{2.11}$$

Looking at the column indices which are determined by the indices of summation in Equation (2.11), it is apparent that the matrix $\underline{\underline{\mathbf{M}}}$ will be sparse and exhibit a band structure. The band structure of $\underline{\underline{\mathbf{M}}}$ only exists if the computational grid is regular. Especially in 2D, adaptive computational grids, considered in the developed code, are in general irregular and produce a very asymmetric matrix $\underline{\underline{\mathbf{M}}}$ without band structure. The sparsity of the matrix is maintained. Thus, the computational solution of the generated linear systems of equations relies on the SLSE solvers which have been tested in Chapter 1.

2.3 Time Integration of the one dimensional General Diffusion Equation

After the spatial discretization by means of the developed CFDS, confer Section 2.1 and Section 2.2, the general diffusion equation transforms into a time-dependent ordinary differential equation (2.10). This thesis treats only time-independent computational grids, diffusivities and velocities of advection or convection. Therefore, the matrix $\underline{\underline{\mathbf{M}}}$ in Equation (2.10) is also time-independent which simplifies the subsequent calculations. A further simplification is the assumption that the sources or drains do not depend on time. Since the relevant and interesting informations of the investigated systems are determined by a state of equilibrium, implicit time integration schemes become more appropriate because they enable larger time steps Δt. The book of Hairer et al. [27,pp. 27-50] presents a number of time integration schemes and describes the respective advantages and disadvantages.
The simplest implicit time-integration scheme is a backward Euler method [27,pp. 28-30]. Applied to Equation (2.10), one obtains a time-discretized equation

$$\underline{\mathbf{F}}^{\mathbf{n+1}} = \underline{\mathbf{F}}^{\mathbf{n}} + \Delta t \cdot \left(\underline{\underline{\mathbf{M}}} \cdot \underline{\mathbf{F}}^{\mathbf{n+1}} + \underline{\mathbf{Q}}\right), \tag{2.12}$$

where the superscripts refer to the respective time level. Grouping terms of the same time level leads to

$$\left(\underline{\underline{\mathbb{1}}} - \Delta t \cdot \underline{\underline{\mathbf{M}}}\right) \underline{\mathbf{F}}^{\mathbf{n+1}} = \underline{\mathbf{F}}^{\mathbf{n}} + \Delta t \cdot \underline{\mathbf{Q}}$$

which can be solved for $\underline{\mathbf{F}}^{\mathbf{n+1}}$,

$$\underline{\mathbf{F}}^{\mathbf{n+1}} = \left(\underline{\underline{\mathbb{1}}} - \Delta t \cdot \underline{\underline{\mathbf{M}}}\right)^{-1} \cdot \left(\underline{\mathbf{F}}^{\mathbf{n}} + \Delta t \cdot \underline{\mathbf{Q}}\right).$$

Another widely used technique for numerical time integration is the implicit trapezoidal rule which is also called *Crank-Nicolson* method. This method is based on an approximation of the time integration by means of a trapezoidal rule as implied by the name. Analogue to Equation (2.12), one obtains again a time-discretized equation

$$\underline{\mathbf{F}}^{\mathbf{n+1}} = \underline{\mathbf{F}}^{\mathbf{n}} + \frac{\Delta t}{2} \cdot \left(\underline{\underline{\mathbf{M}}} \cdot \underline{\mathbf{F}}^{\mathbf{n+1}}\right) + \frac{\Delta t}{2} \cdot \left(\underline{\underline{\mathbf{M}}} \cdot \underline{\mathbf{F}}^{\mathbf{n}}\right) + \Delta t \cdot \underline{\mathbf{Q}},$$

which can be solved for a consecutive time step $\underline{\mathbf{F}}^{\mathbf{n+1}}$,

$$\underline{\mathbf{F}}^{\mathbf{n+1}} = \left(\underline{\underline{\mathbb{1}}} - \frac{\Delta t}{2} \cdot \underline{\underline{\mathbf{M}}}\right)^{-1} \cdot \left(\left(\underline{\underline{\mathbb{1}}} + \frac{\Delta t}{2} \cdot \underline{\underline{\mathbf{M}}}\right) \underline{\mathbf{F}}^{\mathbf{n}} + \Delta t \cdot \underline{\mathbf{Q}}\right).$$

The proceeding subsections cover the definitions of consistency, stability and convergence of numerical schemes, see Section 2.3.1, and provide a method for the determination of stability conditions, see Section 2.3.2. In order to illustrate the terms and conditions an exemplary analysis of a primitive finite difference scheme is performed. The derived results can be transferred to the original problem. Finally, Section 2.3.3 shows the computation of a steady-state solution without a time-integration, which is applicable in many instances.

2.3.1 Considerations about Consistency, Stability and Convergence of the Numerical Scheme

Further investigations of numerical schemes require definitions of terms like consistency, stability and convergence [28,pp. 270-281]. This allows one to formulate conditions for acceptable approximations to the differential problem and to predict stability limits and different behavior of numerical schemes. Knowing the stability enables us to determine quantitatively the accuracy of the numerical result.
The consistency condition associates the discretized equation with the differential equation and restricts the structure of numerical schemes. In particular, consistency implies that a numerical scheme approaches the differential equation in the limit of a spatial discretization and temporal discretization that both tend to zero, $\Delta x \rightarrow 0$ and $\Delta t \rightarrow 0$. Consistency analysis also yields practical information about the accuracy and the truncation error of a numerical scheme. An exemplary consistency analysis on the linear advection equation is reviewed in Hirsch [28,p. 276]. The first step consists in developing the solution function of the

discretized scheme u_j^m in a Taylor series around the value u_i^n,

$$\begin{aligned} u_i^{n+1} &= u_i^n + \Delta t\,(u_t)_i^n + \frac{\Delta t^2}{2}(u_{tt})_i^n \\ u_{i+1}^n &= u_i^n + \Delta x\,(u_x)_i^n + \frac{\Delta x^2}{2}(u_{xx})_i^n + \frac{\Delta x^3}{6}(u_{xxx})_i^n \\ u_{i-1}^n &= u_i^n - \Delta x\,(u_x)_i^n + \frac{\Delta x^2}{2}(u_{xx})_i^n - \frac{\Delta x^3}{6}(u_{xxx})_i^n, \end{aligned} \tag{2.13}$$

and substituting this back in the numerical scheme. In eq. (2.13) x and t subscripts denote partial derivatives and i or j subscripts and n or m superscripts, respectively, of the solution function u specify the position in space or time level. In the case of a linear advection equation which is solved by means of a central, second-order in space and forward, first-order in time finite difference scheme, one obtains the relation,

$$\frac{u_i^{n+1}-u_i^n}{\Delta t} + a\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} - (u_t + a u_x)_i^n = +\frac{\Delta t}{2}(u_{tt})_i^n + \frac{\Delta x^2}{6}a\,(u_{xxx})_i^n + \mathscr{O}\left(\Delta t^2, \Delta x^4\right). \tag{2.14}$$

From the consistency equation (2.14) it is obvious that the right-hand side is zero if Δt and Δx tend to zero; in turn, that means consistency of the considered finite difference scheme. Furthermore, the accuracy of the scheme is confirmed as first order in time and second order in space. Based on the Equation (2.14), the truncation error and its implications can be derived whereby one has to choose between two similar approaches. The first assumes that u_i^n represents the exact solution of the discretized equation. If the exact solution of the discretized equation is available, it is plugged in the consistency equation,

$$(\bar{u}_t + a\bar{u}_x)_i^n = -\frac{\Delta t}{2}(u_{tt})_i^n - \frac{\Delta x^2}{6}a\,(u_{xxx})_i^n + \mathscr{O}\left(\Delta t^2, \Delta x^4\right). \tag{2.15}$$

The bar in Equation (2.15) indicates the exact numerical solution which fulfills an equivalent differential equation for finite values of Δt and Δx. In the computational treatment the limit of Δt and Δx to zero is never realized and one always gets to an Equation (2.15). The right-hand side of this equivalent differential equation is defined as the truncation error ε_T. It is more convenient to have an expression for the truncation error without partial time derivatives. This can be realized using Equation (2.15) and *Young's theorem*, whereby the left-hand side of (2.15) is neglected as a correction of an order of Δt and Δx^2. The modified formula for the truncation error,

$$\varepsilon_T = -\frac{\Delta t}{2}a^2\,(u_{xx})_i^n - \frac{\Delta x^2}{6}a\,(u_{xxx})_i^n + \mathscr{O}\left(\Delta t^2, \Delta x^2\right),$$

yields a physical explanation for the instability of the scheme. Consider the right-hand side of the modified equivalent differential equation,

$$u_t + a u_x = -\frac{\Delta t}{2}a^2\,(u_{xx})_i^n + \mathscr{O}\left(\Delta t^2, \Delta x^2\right).$$

It corresponds to a negative viscosity term which amplifies oscillations and strong gradients. For this reason the scheme is instable. In summary, the determination of the truncation error is a starting point for stability analysis and yields the accuracy of the scheme. A vanishing truncation error provides also another criterion for the consistency of a numerical scheme. The second approach for the determination of the truncation error starts with the exact solution of the differential equation and produces analogue results.
The stability criterion relates the numerical solution to the exact solution of the discretized equation. A stable numerical scheme should not allow errors ε to grow indefinitely, that is, to be amplified without bounds, as we progress from one time step to another [28,p. 278]. This is mathematically expressed by

$$\lim_{n\to\infty} |\varepsilon_i^n| \;\leq\; K \quad \text{at fixed } \Delta t,$$

where K is a number independent of n and ε_i^n is defined as the difference between the computed solution u_i^n and the exact solution $\bar{u}_i^n$,

$$\varepsilon_i^n \;\equiv\; u_i^n - \bar{u}_i^n. \tag{2.16}$$

The above-defined criterion makes no point about the error at an intermediate time step which could be arbitrarily large; hence, a more general definition has to be introduced. According to *Richtmyer and Lax* [28,p. 278], any component of the initial solution should not be amplified without bound in a stable numerical scheme. This general treatment requires to formulate the numerical scheme in a matrix or operator form,

$$\underline{\mathbf{U}}^{\mathbf{n+1}} \;=\; \underline{\underline{\mathbf{C}}} \cdot \underline{\mathbf{U}}^{\mathbf{n}}, \tag{2.17}$$

whereby the matrix $\underline{\underline{\mathbf{C}}}$ depends on the time step Δt and mesh size Δx and $\underline{\mathbf{U}}$ denotes a vector containing the u_i at a given time level. Considering the above-defined matrix $\underline{\underline{\mathbf{C}}}$, an amplification without bound is prevented by the condition

$$\left\| \left(\underline{\underline{\mathbf{C}}} \right)^n \right\| \;=\; K \quad \text{for} \quad \begin{array}{l} 0 < \Delta t < \tau \\ 0 \leq n\Delta t \leq T \end{array} \tag{2.18}$$

and for all n, whereby τ and T are fixed numbers and the norm is unspecified for the moment.
Knowing consistency and stability signifies that the investigated scheme is convergent. This context is described by the fundamental *Equivalence Theorem of Lax* [28,p. 281] which states that for a well-posed initial value problem and a consistent discretization scheme, stability is the necessary and sufficient condition for convergence. In this sense, convergence means that the numerical solution approaches the exact solution of the differential equation at any point and time for $\Delta x \to 0$ and $\Delta t \to 0$. A mathematical description of convergence based on a matrix or form of the numerical scheme (2.17) is given by *Richtmyer*

and Morton[28,p. 296],

$$\lim_{\substack{\Delta t \to 0 \\ n \to \infty}} \left\| \left[\underline{\underline{\mathbf{C}}}(\Delta t) \right]^n \underline{\mathbf{U}}^{\mathbf{0}} - \underline{\bar{\mathbf{U}}}(t) \right\|,$$

whereby the bar indicates the exact solution of the numerical scheme.

2.3.2 Von Neumann Method for Stability Analysis and Courant-Friedrichs-Lewy Condition

The investigation of the stability of a numerical scheme, confer Section 2.3.1, is widely studied in literature [28, 29, 25, 30], but it is still a demanding topic and mostly limited to linear problems. Although this limitation is used, initial and boundary conditions complicate the investigations.
A wide-spread technique is the *Von Neumann method* for stability analysis [28,pp. 283-338], which was introduced by Von Neumann during World War II. This method is based on a Fourier decomposition of the solution u_i^n and of the errors ε_i^n defined by Equation (2.16),respectively,

$$\begin{aligned} \varepsilon_i^n &= \sum_{j=-\mathrm{N}}^{\mathrm{N}} E_j^n \, e^{I \cdot k_j \cdot i \cdot \Delta x} \\ &= \sum_{j=-\mathrm{N}}^{\mathrm{N}} E_j^n \, e^{I \cdot i \cdot j \cdot \frac{\pi}{\mathrm{N}}} \\ &\text{with} \\ k_j &= j \, \frac{\pi}{L} \\ &= j \, \frac{\pi}{N \Delta x}, \end{aligned} \tag{2.19}$$

where N is the number of subdivisions Δx of the domain of solution with length L, k_j is the wavenumber and E_j^n is the amplitude of the j-th harmonic and I denotes the imaginary unit. As a discretization scheme for the solution exists, this scheme must also hold for the errors. From the resulting equation, one can determine the absolute value of the ratio $|G|$ of temporally sequential amplitude factors. If $|G|$ is less than or equal one for any harmonic,

$$\begin{aligned} |G| &\equiv \left| \frac{E^{n+1}}{E^n} \right| \\ &\leq 1, \end{aligned} \tag{2.20}$$

the considered discretization scheme fulfills the stability conditon (2.18). By definition, the above-applied Fourier decomposition is only possible in the case of an infinite domain or periodic boundary conditions on a finite domain. In fact, most of the studied physical problems involve boundary conditions and non-linearities; despite the made restrictions, it yet can be useful to perform a Von Neumann analysis. In the case of a non-linear differential equation with eventually non-constant coefficients a local Von Neumann analysis of the linearized problem yields at least a necessary, though not sufficient, condition for stability. To

illustrate the procedure, one can consider again the example of a linear advection equation which is solved by means of a central, second-order in space and forward, first-order in time finite difference scheme, confer Section 2.3.1. Plugging Equation (2.16) in the discretization scheme for the u_i^n results in an equation,

$$\frac{\bar{u}_i^{n+1} - \bar{u}_i^n}{\Delta t} + \frac{\varepsilon_i^{n+1} - \varepsilon_i^n}{\Delta t} = -\frac{a}{2\Delta x}\left(\bar{u}_{i+1}^n - \bar{u}_{i-1}^n\right) - \frac{a}{2\Delta x}\left(\varepsilon_{i+1}^n - \varepsilon_{i-1}^n\right), \tag{2.21}$$

where the $\bar{u}_i^n$ eventually cancel out because they satisfy exactly the scheme as demanded. Finally Equation (2.21) reduces to

$$\frac{\varepsilon_i^{n+1} - \varepsilon_i^n}{\Delta t} = -\frac{a}{2\Delta x}\left(\varepsilon_{i+1}^n - \varepsilon_{i-1}^n\right).$$

The difference scheme for the errors above is the starting point for the Von Neumann stability analysis. Plugging Equation (2.19) in (2.21) and some algebraic manipulations allow us to determine an expression for the quantity $|G|$,

$$\begin{aligned} |G|^2 &= 1 + \left(\frac{a\Delta t}{\Delta x}\right)^2 \sin^2\phi \\ &\text{with} \\ \phi &\equiv k_j \cdot \Delta x, \end{aligned}$$

which apparently does not satisfy Equation (2.20) for none of the phase angles ϕ. In such a case the applied numerical scheme is called unconditonally unstable. If Equation (2.20) is fulfilled for a defined ratio of Δt and Δx the scheme is called conditionally stable and the condition is referred to as *Courant-Friedrichs-Lewy* (CFL) condition [28,p. 287]. In the case that Equation (2.20) is always fulfilled, one denotes these schemes as unconditonally stable.

As mentioned, a detailed analysis of stability can be very demanding. The investigated general diffusion equation (2.1), used to model transport in fusion plasmas, can have complicated, non-constant diffusivities and velocities of advection or convection and naturally boundary conditions are involved. Furthermore, the developed discretization scheme, confer Section 2.1 and Section 2.2, generally uses an adaptive computational mesh which makes the anaylsis difficult. Even in the instance of an equidistant mesh, it is hard to perform a full Von Neumann stability analysis. During the computational implementation some practical approximations for CFL conditions have proven successful. In the book of Hirsch [28,pp. 331-335] a CFL condition for multidimensional space-centered, convection-diffusion equations, applicable to a wider range of problems, can be found and the book of Cockburn et al. [30,pp. 384-423] shows further appropriate conditions.

2.3.3 Determination of the Steady-State Solution

Considering again the transport problem in fusion plasmas, one has to figure the interesting quantities which should be determined. In most of the cases an exact time-development of these quantities is not important, instead the steady-state solution contains mainly the relevant information. If the basic differential equation is explicitly time-dependent such as in Equation (2.10), the determination of the steady-state becomes easy. Since a steady-state solution does not change in time, partial time derivatives can be neglected. In the case of the regarded differential equation (2.10) one obtains subsequent linear system of equations,

$$\underline{\underline{\mathbf{M}}} \cdot \underline{\mathbf{F}} \;=\; -\underline{\mathbf{Q}}. \tag{2.22}$$

For a simultaneously vanishing source term Equation (2.22) yields the trivial solution $\underline{\mathbf{F}} = \underline{\mathbf{0}}$.

2.4 Implementation of Boundary Conditions in the Conservative Finite Difference Scheme for General Diffusion Equations in one Dimension

So far boundary conditions are not incorporated in the CFDS which is developed in Section 2.1 and Section 2.2. In general, one has to distinguish between *Dirichlet boundary conditions*, *Neumann boundary conditions* and mixed types [21,p. 691]. This thesis discusses only the implementation of Dirichlet boundary conditions that are used to describe walls of finite temperature as in the case of the investigated model for the transport in fusion plasmas. For the computational implementation two different approaches mainly exist which have respective pros and cons. A wide-spread technique uses ghost cells, confer Blazek [29,pp. 267-297] or Versteeg and Malalasekera [23,pp. 192-209], outside the domain of solution to model the chosen boundary condition. Another possibility is the modification of the stencil or of the interpolation function at boundary of the domain. This approach can be found in the books of Ferziger and Peric [24,pp. 81-89] or Versteeg and Malalasekera [23,pp. 86-98].
The ghost cell method extends the domain of solution by adding additional cells $F_{\tilde{i}}$ with specific values which describe the boundary condition. For instance a temperature sink can be modeled by ghost cells with zero temperature. In Figure 2.2 F_0 denotes the ghost cell that is associated with boundary η_0.

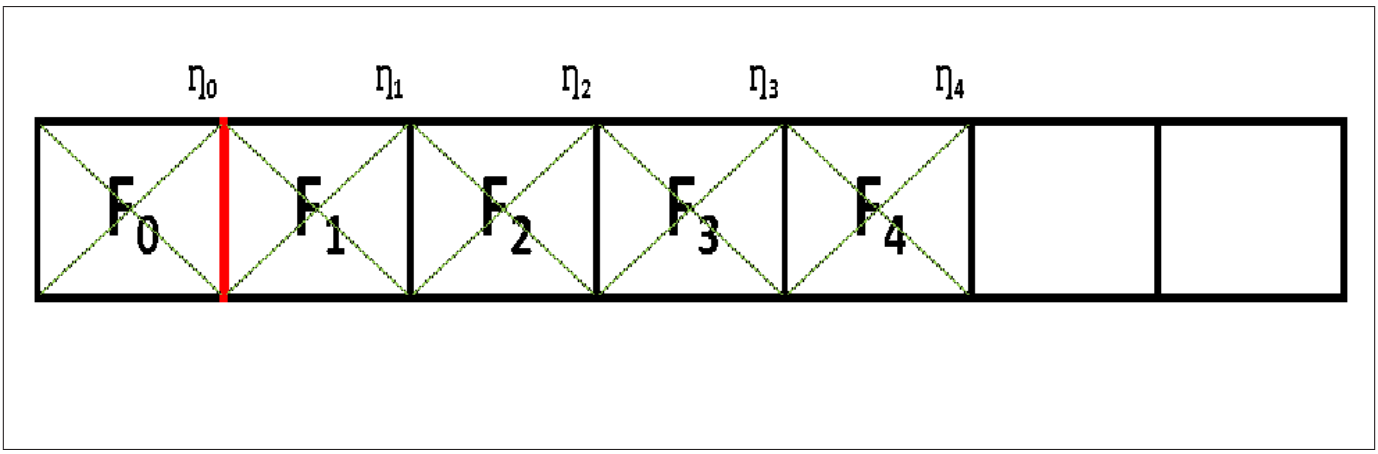

Figure 2.2: Evaluation of function $f(\eta,t)$ at boundary η_0 according to the specified boundary condition.

A major advantage of this technique is a simple implementation without modification of the stencil or interpolation function; on the other hand, ghost cells cannot exactly set the value of a function or of its derivative at the boundaries.
The second method makes use of a modification of the interpolation function at the edges of the domain, e.g. boundary η_0 in Figure 2.2. To illustrate this procedure we consider a Dirichlet boundary condition where the function $f(\eta,t)$ vanishes at edges of the domain. Following Equation (2.5), the function f can be expanded into a Taylor series at boundary η_0,

$$ {}_0f(\eta,t) = {}_0a_0(t) + {}_0a_1(t)(\eta-\eta_0) + {}_0a_2(t)(\eta-\eta_0)^2 + {}_0a_3(t)(\eta-\eta_0)^3 + \cdots, $$

whereby the polynomial coefficient ${}_0a_0(t)$ has to vanish due to boundary condition,

$$ {}_0f(\eta_0,t) = 0. $$

For the determination of numerical flux function $\Gamma(\eta,t)$ at boundary η_0 (2.3),

$$ \begin{aligned} \Gamma(\eta_0,t) &= -D(\eta_0)\left.\frac{\partial f(\eta,t)}{\partial \eta}\right|_{\eta_0} + v(\eta_0) f(\eta_0,t) \\ &= -D(\eta_0)\ {}_0a_1(t) + v(\eta_0)\ {}_0a_0(t) \end{aligned} $$

one also has to compute the derivative of the function f at boundary η_0. An approximation of first order to the derivative is given by subsequent finite difference,

$$ \left.\frac{\partial f(\eta,t)}{\partial \eta}\right|_{\eta_0} = \frac{\overbrace{{}_0f(\eta_0,t)}^{\to 0} - {}_1f(\eta_1,t)}{\Delta\eta}. \tag{2.23} $$

From Equation (2.23) one eventually obtains a relation for the sought polynomial coefficient ${}_0a_1(t)$,

$$ {}_0a_1(t) = -\frac{{}_1a_0(t)}{\Delta\eta}. $$

Better approximations to the derivative are often required and can be realized using finite differences with a truncation error of higher order. In the conducted tests in Chapter 4 the primitive approximation is always used and no disturbing influences on the solution are noticed.

2.5 Test Cases for the one dimensional Conservative Finite Difference Scheme

The subsequent test cases study a general diffusion equation (2.1) with a constant diffusion coefficient $D(\eta)=1$ and velocity of advection $v(\eta)=-2\eta$,

$$\frac{\partial f(\eta,t)}{\partial t} = \frac{\partial}{\partial \eta}\left(\frac{\partial f(\eta,t)}{\partial \eta}+2\eta f(\eta,t)\right),$$

which is solved by a Gaussian curve with a full width half maximum (FWHM) of $\sqrt{\frac{1}{2}}$,

$$f(\eta,t) = e^{-\eta^2}.$$

Since there are no sources in the considered system, the steady-state solution will be in all test cases $f(\eta,t\to\infty)=0$ because particles get lost due to round-off errors and numerical instabilities. As an initial profile we choose a rectangular whereby the included area must be equal to the area of the Gaussian $A_{\text{Gaussian}}=\sqrt{\pi}$. This demand results from the conservativity of the applied numerical scheme. In the subsequent test cases the length of the rectangular is 2 and consequently the height has to be $\sqrt{\frac{\pi}{4}}$. Furthermore, the chosen polynomial reconstruction is of fourth order and an estimate for the CFL condition is calculated by following expression [30,p. 415],

$$\Delta t \leq c_{\text{CFL}}\cdot\left(\max_i\left(\frac{\|v(\eta_i)\|}{\Delta\eta}\right)+2\,\frac{D}{(\Delta\eta)^2}\right)^{-1},$$

whereby a spatial discretization of $\Delta\eta=0.08$ is chosen. Figure 2.3 shows the time evolution of the Gaussian curve and the relative precision using a Crank-Nicolson time integrator. Under the considered CFL condition $c_{\text{CFL}}=5$ the relative error grows only linearily and the CFDS proves to be conservative.

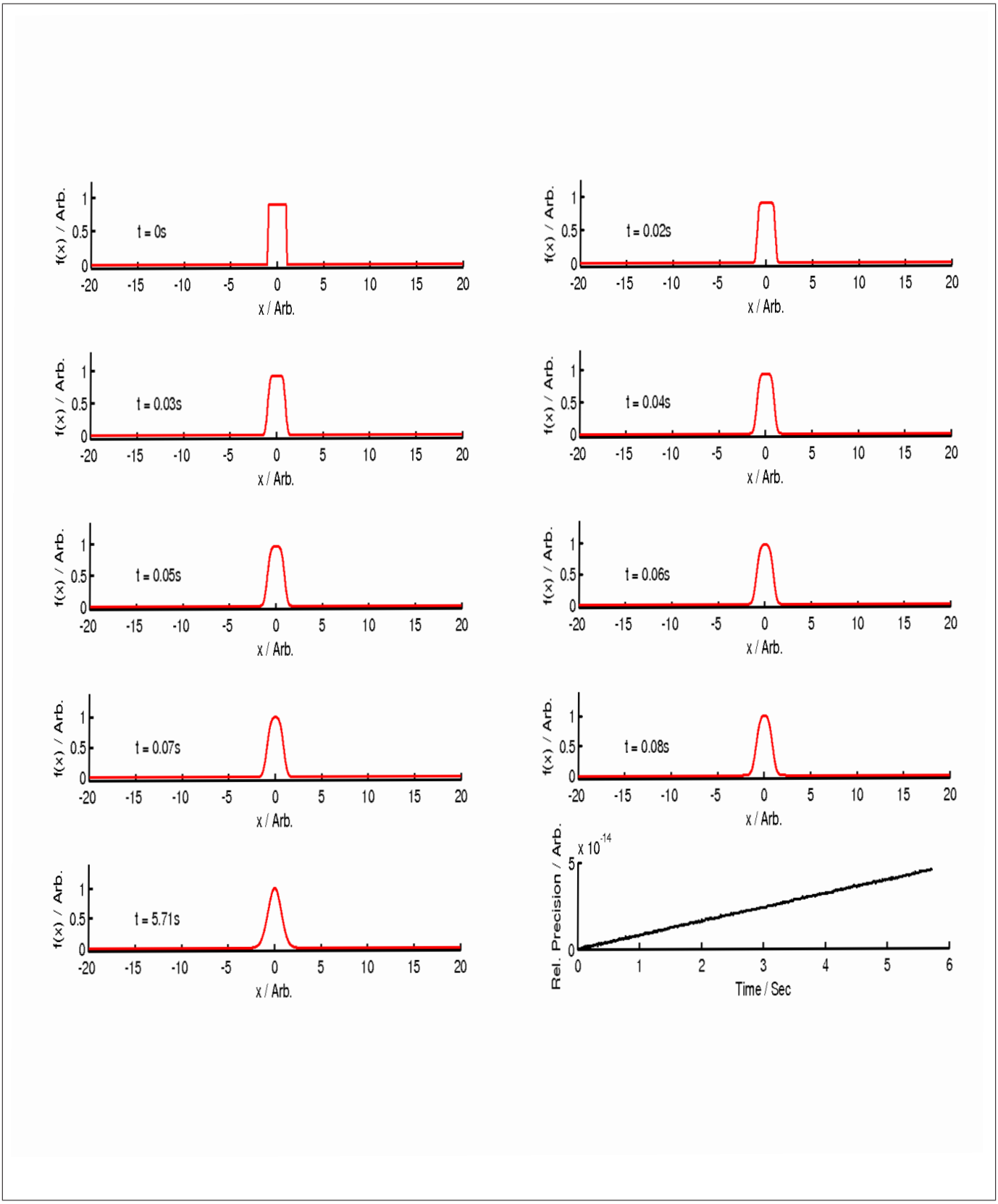

Figure 2.3: Time evolution of the Gaussian curve and the relative precision using a Crank-Nicolson time integrator and a $c_{\mathrm{CFL}} = 5$ from 0s to 5.71s.

The time evolution of the Gaussian curve and the corresponding relative precision for another CFL condition is depicted in Figure 2.4 whereby the same time integration procedure is used. In this case an insufficient CFL condition $c_{\mathrm{CFL}} = 25$ is tested. As the relative error grows exponentially, the CFDS is not

conservative.

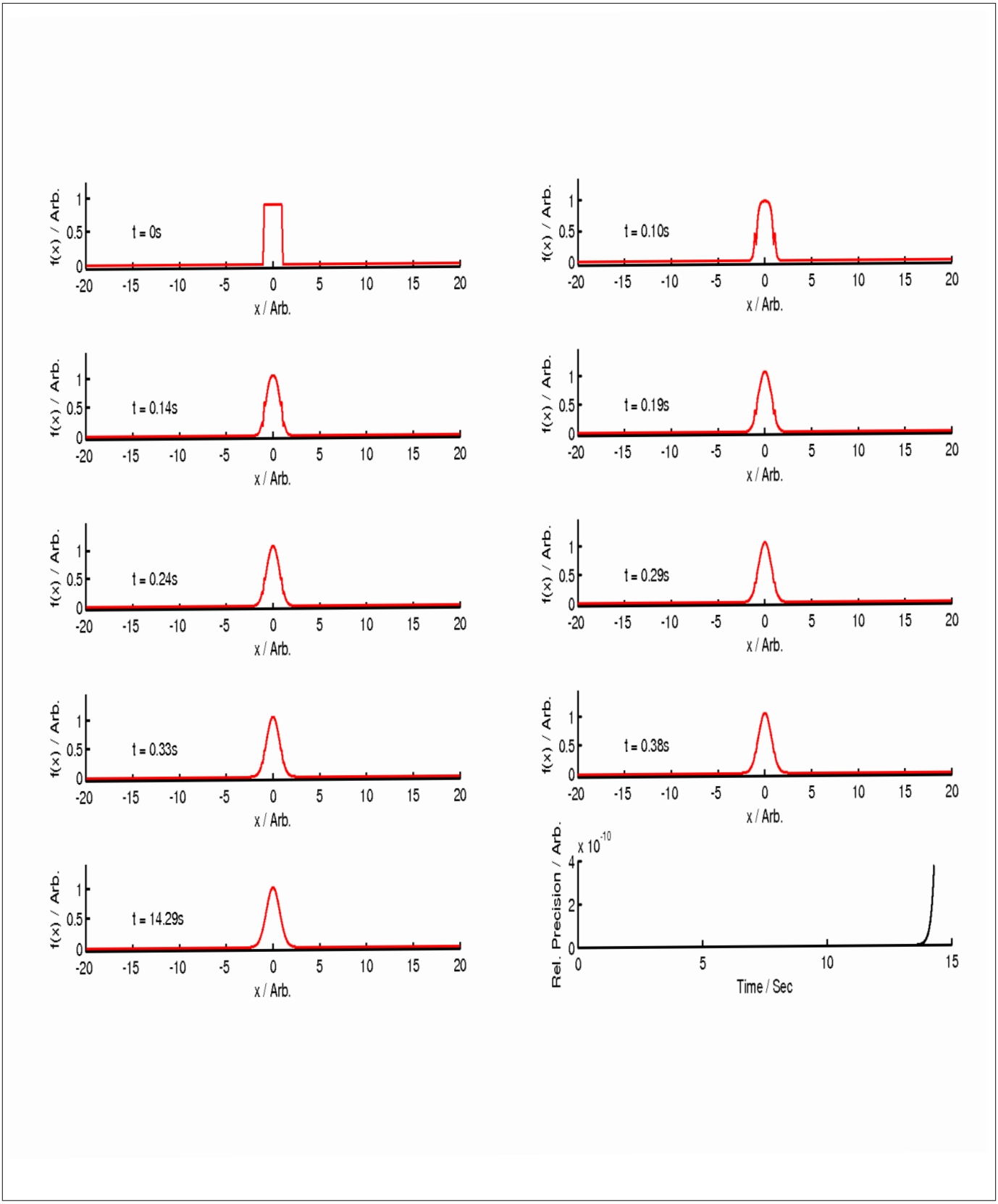

Figure 2.4: Time evolution of the Gaussian curve and the relative precision using a Crank-Nicolson time integrator and a $c_{\mathrm{CFL}} = 25$ from 0s to 14.29s.

As shown in Figure 2.5 an optimal CFL condition $c_{\mathrm{CFL}} = 10$ leads to a fully stable Crank-Nicolson time integration and the relative error oscillates around a constant value for times larger than 8s. The total cell

volume is conserved for all times. This behavior for an optimal c_{CFL} is common for both considered time integrators.

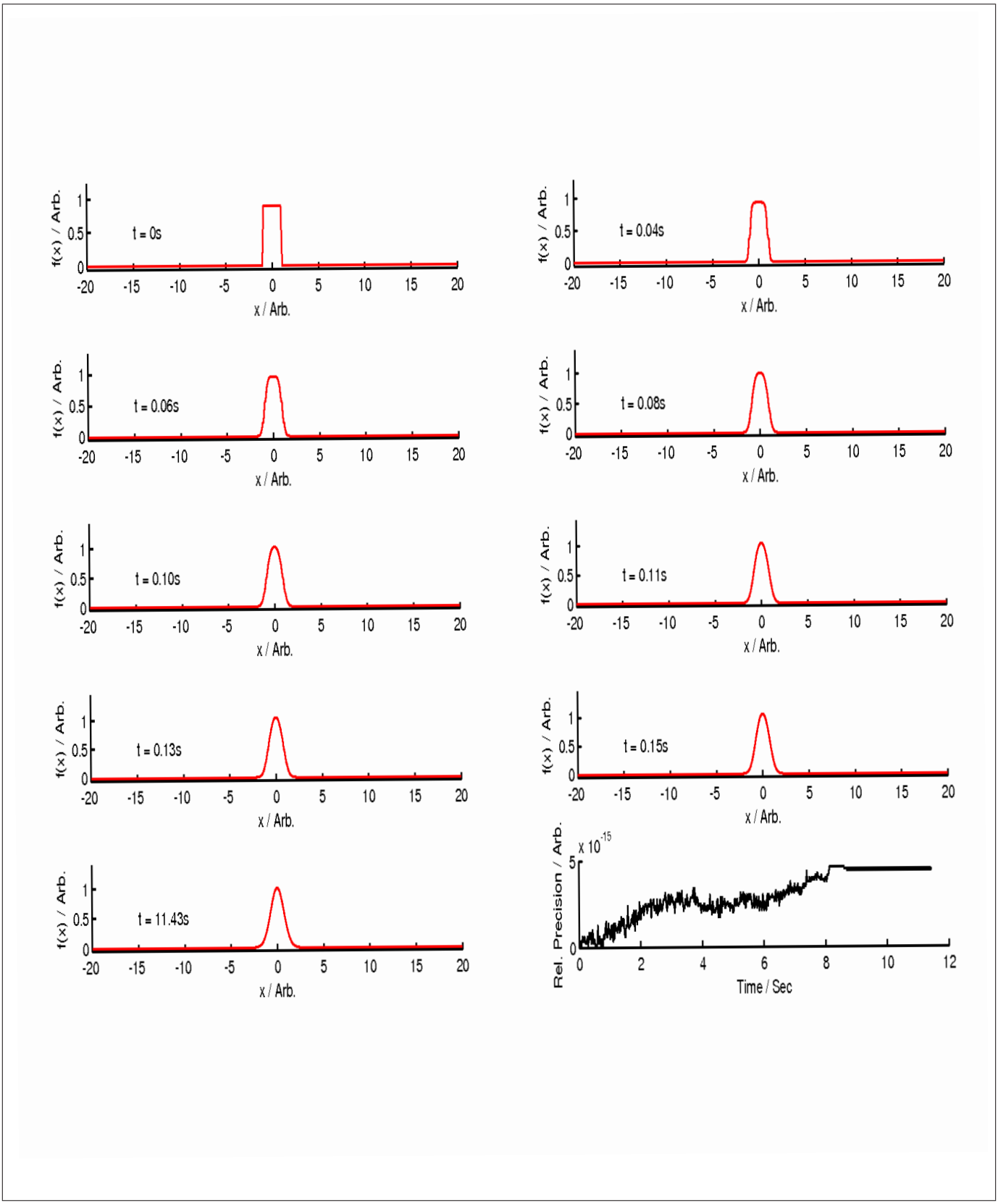

Figure 2.5: Time evolution of the Gaussian curve and the relative precision using a Crank-Nicolson time integrator and a $c_{\mathrm{CFL}} = 10$ from 0s to 11.43s.

In Figure 2.6 a backward Euler time integration of the same test case is performed and the CFL condition

is again $c_{\mathrm{CFL}} = 5$. The time evolution of the Gaussian curve is analogue and the relative precision grows also linearily, but not so fast as in the test case for a Crank-Nicolson time integrator.

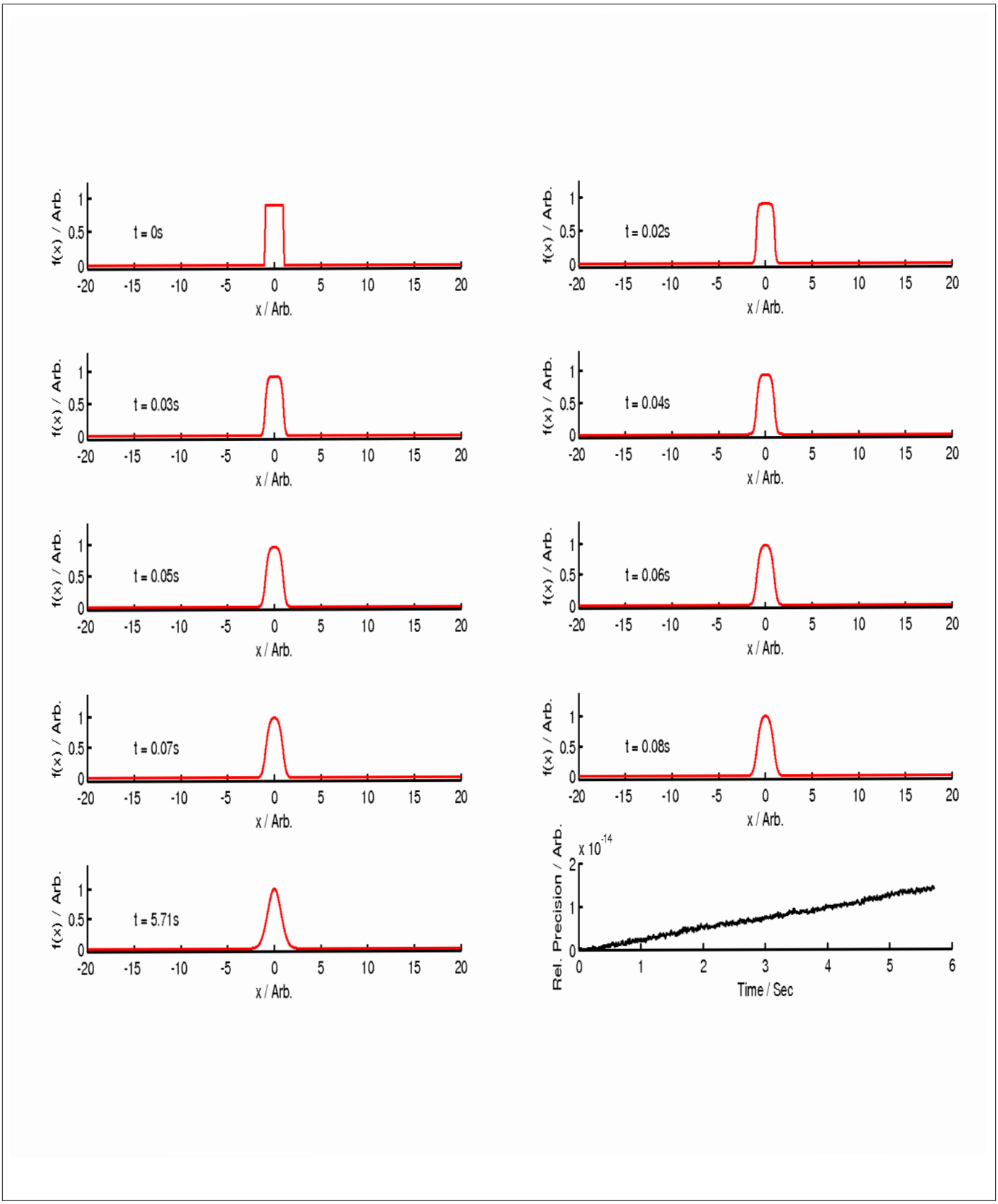

Figure 2.6: Time evolution of the Gaussian curve and the relative precision using a backward Euler time integrator and a $c_{\mathrm{CFL}} = 5$ from 0s to 5.71s.

Figure 2.7 shows the time evolution of the Gaussian curve by means of a backward Euler and the corre-

sponding relative precision for an insufficient CFL condition c_{CFL}. In comparison to the insufficient CFL condition for the Crank-Nicolson time integrator, the value of c_{CFL} is by far smaller and instabilites occur much earlier. As the relative error grows also exponentially, the CFDS is not conservative.

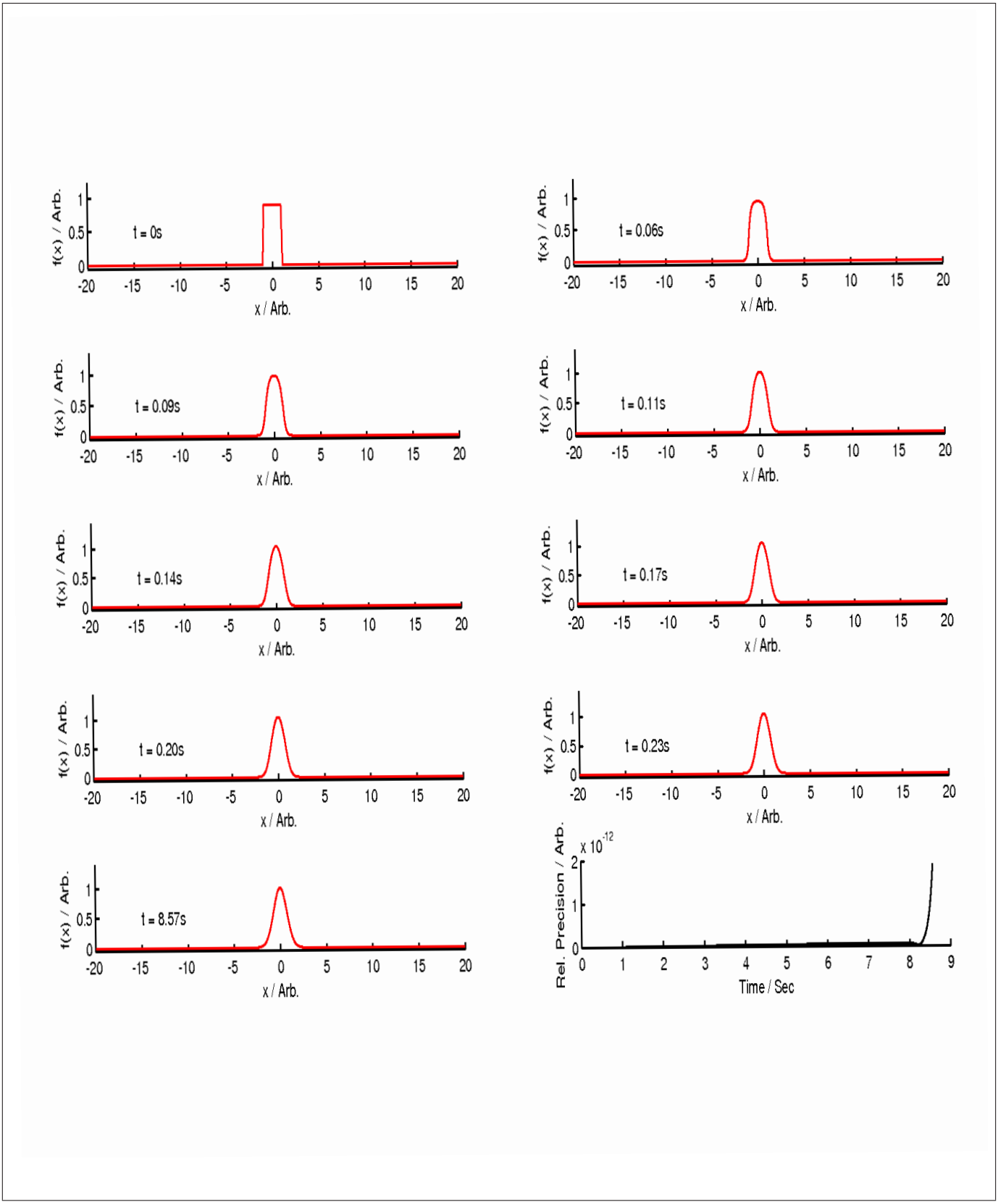

Figure 2.7: Time evolution of the Gaussian curve and the relative precision using a backward Euler time integrator and a $c_{\mathrm{CFL}} = 15$ from 0s to 8.57s.

As mentioned an optimal CFL condition $c_{\mathrm{CFL}} = 10$ leads to a fully stable backward Euler time integration which is illustrated in Figure 2.8. The only difference between both time integration procedures is that the relative error does not oscillate and is constant for times larger than 8s. The total cell volume is conserved as well for all times.

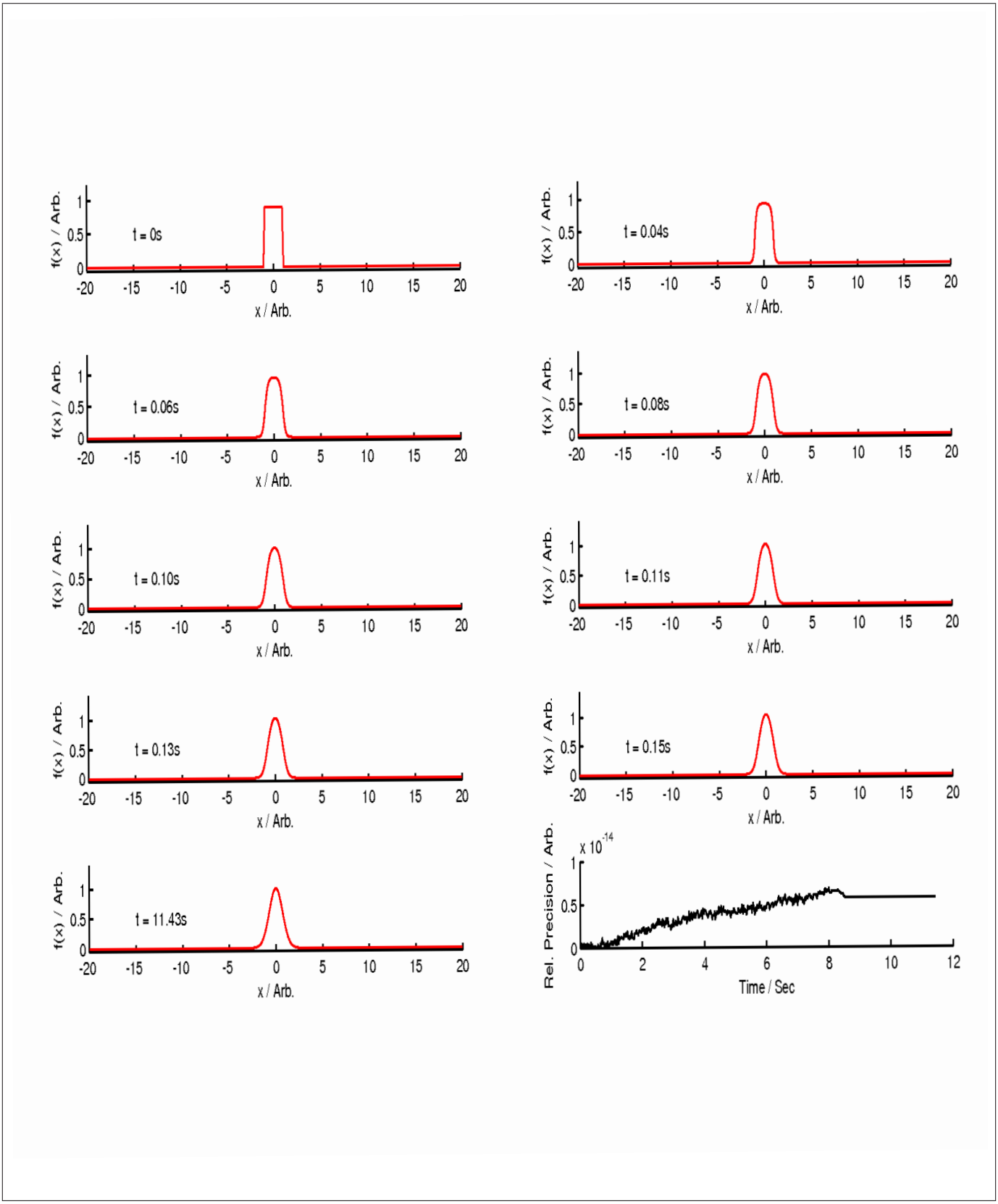

Figure 2.8: Time evolution of the Gaussian curve and the relative precision using a backward Euler time integrator and a $c_{\mathrm{CFL}} = 10$ from 0s to 11.43s

3 Extension of the Conservative Finite Difference Scheme to two dimensional General Diffusion Equations

The basic concepts behind the developed CFDS are derived within Chapter 2 and can be easily extended to 2D, see Section 3.1. Difficulties in context with a multi-dimensional treatment of the general diffusion equation especially rise from the limited computational ressources. One simply realizes that the number of cells N for certain level of mesh refinement grows with the power of the dimension D. As a further consequence, the resulting linear systems of equations are of the order $\mathrm{N}^{\mathrm{D}} \times \mathrm{N}^{\mathrm{D}}$ which implies high memory requirements. Another interesting quantity that allows us to make predictions about the computational feasibility of the problem is the number of non-zeros of the coefficient matrix. Beside the dependence on the level of mesh refinement and dimensionality of the problem, the number of non-zeros is mainly determined by the order of polynomial interpolation of numerical flux functions at the boundaries, see Section 3.2. Thus, one has to make a compromise between the desired mesh refinement for the problem and a sufficient order of polynomial interpolation of the numerical flux functions. In Section 3.3 the relations for boundary conditions in 2D are outlined. Section 3.4 eventually presents the results of a test case that is analogue to Section 2.5. Once again the stability of the CFDS, used to solve a 2D general diffusion equation which generates a Gaussian profile as a solution, is studied. In Chapter 4 the 2D CFDS is applied to heat transport problems in magnetized fusion plasmas.

3.1 Conservative Formulation of a two dimensional General Diffusion Equation

A general form of a diffusion equation in 2D is given by the subsequent expression,

$$\frac{\partial f(x,y,t)}{\partial t} = \underline{\boldsymbol{\nabla}} \cdot (D(x,y)\, \underline{\boldsymbol{\nabla}}\, f(x,y,t) - \underline{\mathbf{v}}(x,y)\, f(x,y,t)) + q(x,y), \tag{3.1}$$

whereby $D(x,y)$ describes the diffusivity and accordingly $\underline{\mathbf{v}}(x,y)$ the velocity of convection or advection. The last term $q(x,y)$ stands for sources or drains in the system. A further numerical treatment requires a discretization of the domain of solution. Figure 3.1 shows an equidistant regular 2D mesh which is split up into N cells with N_x or N_y subdivisions in x- and y-direction, respectively. The axes in Figure 3.1 are labeled in relation to a highlighted cell F_k. In order to be able to apply the 1D computational treatment to the 2D case, a linear index k, going down the columns consecutively in an ascending order, is assigned to each cell. In the case of a regular mesh one can calculate the indices $i(k)$ linked with the x-coordinate of the left boundary and the indices $j(k)$ linked with the y-coordinate of the upper boundary from the linear

index k,

$$\begin{aligned} i(k) &= \lceil \frac{k}{\mathrm{N}_y} \rceil \\ j(k) &= ((k-1) \mod \mathrm{N}_y) + 1, \end{aligned}$$

whereby $\lceil \cdot \rceil$ denotes the ceiling function. Naturally, the information about the boundaries must be provided by adaptive mesh infrastructure in the instance of irregular meshes.
For a conservative treatment one again integrates Equation (3.1) over the area of the k-th cell as in the 1D case, confer Section 2.1.

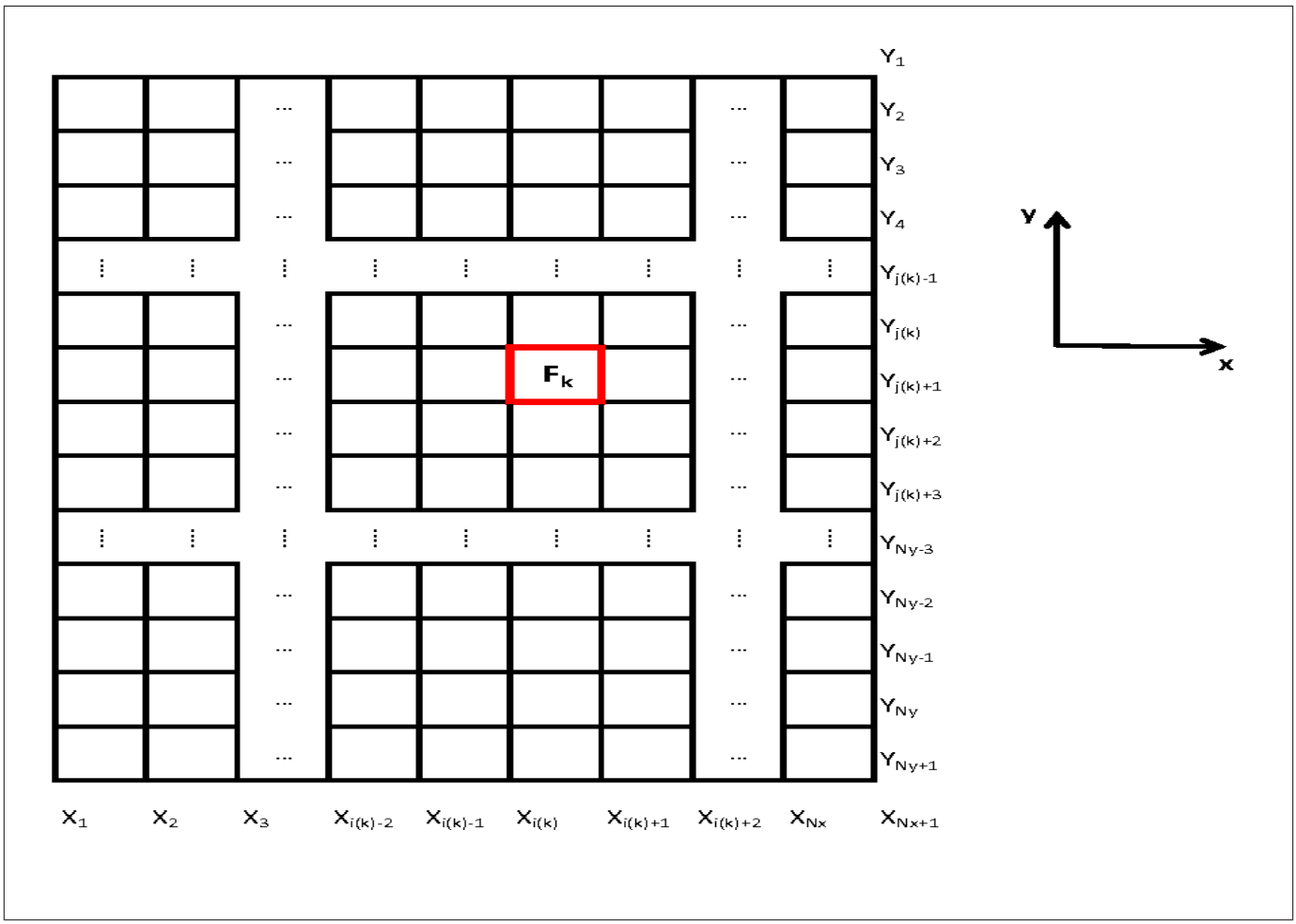

Figure 3.1: Computational grid in 2D with labeling of the axes in relation to the highlighted cell F_k.

Defining the cell-volume $F_k(t) = \int_{A_k} \mathrm{d}A \; f(x,y,t)$ and applying *Gauss's law* [21,p. 687], Equation (3.1)

becomes an ordinary differential equation,

$$\frac{\mathrm{d}F_k(t)}{\mathrm{d}t} = \int\limits_{\partial A_k} \mathrm{d}s\ \underline{\mathbf{n}} \cdot \underline{\mathbf{\Gamma}}(x,y,t) + Q_k \tag{3.2}$$

with

$$\underline{\mathbf{\Gamma}}(x,y,t) = D(x,y)\ \underline{\nabla}\ f(x,y,t) - \underline{\mathbf{v}}(x,y)\ f(x,y,t) \tag{3.3}$$

and

$$Q_k = \int\limits_{A_k} \mathrm{d}A\ q(x,y).$$

As a consequence, the value of cell-volume F_k can only change by numerical fluxes $\underline{\mathbf{\Gamma}}(x,y,t)$ through the boundaries or by a source or a drain Q_k. Considering the quadrilateral grid in Figure 3.1, the outward-pointing unit normal vector to the boundary ∂A_k is

$$\underline{\mathbf{n}} = \frac{1}{\sqrt{\mathrm{d}x^2 + \mathrm{d}y^2}} \begin{pmatrix} \mathrm{d}y \\ -\mathrm{d}x \end{pmatrix}$$

and the line element is

$$\mathrm{d}s = \sqrt{\mathrm{d}x^2 + \mathrm{d}y^2}.$$

Hence, Equation 3.2 can be evaluated in the case of a quadrilateral grid and one obtains

$$\begin{aligned} \frac{\mathrm{d}F_k(t)}{\mathrm{d}t} &= \int\limits_{\partial A_k} (\Gamma_x(x,y,t)\ \mathrm{d}y - \Gamma_y(x,y,t)\ \mathrm{d}x) + Q_k \\ &= I_k(t) + Q_k, \end{aligned}$$

whereby $I_k(t)$ denotes the numerical flux difference for the k-th cell. In order to compute the numerical flux difference $I_k(t)$, one has to specify the path of integration ∂A_k which moves in a counter clockwise manner

around F_k,

$$
\begin{aligned}
I_k(t) &= \int_{\partial A_k} \left(\Gamma_x(x,y,t)\,\mathrm{d}y - \Gamma_y(x,y,t)\,\mathrm{d}x\right) \\
&= \int_{y_{j(k)+1}}^{y_{j(k)}} \mathrm{d}y \left[D(x_{i(k)+1},y)\ \frac{\partial f(x,y,t)}{\partial x}\bigg|_{x_{i(k)+1}} - v_x(x_{i(k)+1},y)\ f(x_{i(k)+1},y,t) - \right. \\
&\quad \left. - D(x_{i(k)},y)\ \frac{\partial f(x,y,t)}{\partial x}\bigg|_{x_{i(k)}} + v_x(x_{i(k)},y)\ f(x_{i(k)},y,t) \right] - \\
&\quad - \int_{x_{i(k)}}^{x_{i(k)+1}} \mathrm{d}x \left[D(x,y_{j(k)+1})\ \frac{\partial f(x,y,t)}{\partial y}\bigg|_{y_{j(k)+1}} - v_y(x,y_{j(k)+1})\ f(x,y_{j(k)+1},t) - \right. \\
&\quad \left. - D(x,y_{j(k)})\ \frac{\partial f(x,y,t)}{\partial x}\bigg|_{y_{j(k)}} + v_y(x,y_{j(k)})\ f(x,y_{j(k)},t) \right].
\end{aligned} \tag{3.4}
$$

A mathematically closed formulation requires that the values of $f(x,y,t)$ and its derivatives at the boundaries in Equation (3.4) must be reconstructed from the cell volumes. For this reconstruction a polynomial interpolation of the function at the boundaries is considered again, confer Section 3.2. The integrals appearing in Equation (3.4) can be evaluated either analytically or by means of numerical integration procedures. In the developed code a *Gauss-Legendre quadrature* routine [2,pp. 140-155] is exploited for the integration which allows us to handle complex analytical expressions for the diffusivity or velocity of advection.
If the numerical fluxes vanish at the boundaries of the domain of solution and all sources balance the drains, the total cell volume is conserved as described in the 1D case, confer Equation (2.4).

3.2 Polynomial Reconstruction of the Numerical Fluxes at the Cell Boundaries in two Dimensions

Analogue to the 1D case, the polynomial reconstruction of the numerical fluxes in 2D is based on a Taylor expansion of the function f at the boundaries. Due to the higher dimensionality one has to treat four boundaries. In the following calculations the polynomial interpolation of the right and left boundary, respectively, of a cell is described extensively. The polynomial interpolation of the upper or lower boundary can be done accordingly by rotating the stencils and corresponding polynomial expansions.

From the Taylor expansion of $f(x,y,t)$ up to order m in x-direction and order n in y-direction,

$$
\begin{aligned}
{}_{i(k),j(k)+1/2}f(x,y,t) &= \sum_{l=1}^{m*n} {}_{i(k),j(k)+1/2}a_l(t)\ (x-x_{i(k)})^{p_x(l)}\ (y-y_{j(k)+1/2})^{p_y(l)} \qquad (3.5)\\
&\text{with}\\
y_{j(k)+1/2} &= \frac{y_{j(k)}+y_{j(k)+1}}{2}\\
&\text{and}\\
p_x &= (\underbrace{0,1,2,\ldots,m-1}_{n\times},\ldots\ldots,0,1,2,\ldots,m-1)\\
p_y &= (\underbrace{\underbrace{0,0,0,\ldots,0}_{m\times},\underbrace{1,1,1,\ldots,1}_{m\times}}_{n\times},\ldots\ldots,n-1,n-1,n-1,\ldots,n-1),
\end{aligned}
$$

whereby the left subscripts again denote the expansion at a certain boundary and the elements ${}_{i(k),j(k)+1/2}a_l(t)$ abbreviate the corresponding polynomial coefficients of the Taylor series, one can derive again a relation for the cell volumes $F_{\tilde{k}}(t)$,

$$
\begin{aligned}
F_{\tilde{k}}(t) &= \int_{x_{i(\tilde{k})}}^{x_{i(\tilde{k})+1}} \mathrm{d}x \int_{y_{j(\tilde{k})+1}}^{y_{j(\tilde{k})}} \mathrm{d}y \sum_{l=1}^{m*n} {}_{i(k),j(k)+1/2}a_l(t)\ (x-x_{i(k)})^{p_x(l)}\ (y-y_{j(k)+1/2})^{p_y(l)}\\
&= \sum_{l=1}^{m*n} {}_{i(k),j(k)+1/2}a_l(t)\ {}_{i(k),j(k)+1/2}A_{\tilde{k}l} \qquad (3.6)\\
&\text{with}\\
{}_{i(k),j(k)+1/2}A_{\tilde{k}l} &= \frac{\left(x_{i(\tilde{k})+1}-x_{i(k)}\right)^{p_x(l)+1}-\left(x_{i(\tilde{k})}-x_{i(k)}\right)^{p_x(l)+1}}{p_x(l)+1}\cdot\\
&\quad\cdot\frac{\left(y_{j(\tilde{k})}-y_{j(k)+1/2}\right)^{p_y(l)+1}-\left(y_{j(\tilde{k})+1}-y_{j(k)+1/2}\right)^{p_y(l)+1}}{p_y(l)+1}.
\end{aligned}
$$

In Equation (3.5) a compact notation for the powers in the Taylor expansion is introduced. Along with the linear indexing of the cell volumes, this allows us to map the multi-dimensional problem to a formal 1D problem for which a matrix formalism and computational routines are available. Using the polynomial approximation (3.6), the cell volumes $F_{\tilde{k}}(t)$ within a defined stencil should be approached up to a known truncation error. This demand results in a system of linear equations for the unknown polynomial coefficients. A well-posed set of equations is obtained by stencils that are symmetric with respect to the boundary and that involve m cells in x-direction and n cells in the y-direction. As one can imagine, this demand is hard to realize in the instance of adaptive meshes. For this purpose more sophisticated criteria must be considered. Since the same computational treatment, the polynomial coefficents ${}_{i(k),j(k)+1/2}a_l(t)$ are again not calculated explicitly; instead of them, one computes the inverse of ${}_{i(k),j(k)+1/2}A_{ij}$, ${}_{i(k),j(k)+1/2}B_{jk}$, which

is independent of time and can be used to determine the ${}_{i(k),j(k)+1/2}a_l(t)$) from the cell volumes,

$$
\begin{aligned}
{}_{i(k),j(k)+1/2}a_l(t) \quad &= \sum_{\substack{\tilde{k} \in \text{ symmetric stencil} \\ \text{of boundary } (x_{i(k)},y_{j(k)+1/2})}} {}_{i(k),j(k)+1/2}B_{l\tilde{k}}\, F_{\tilde{k}}(t) \\
&\text{with} \\
\delta_{k\tilde{k}} \quad &= \sum_{l=1}^{m*n} {}_{i(k),j(k)+1/2}A_{kl}\ {}_{i(k),j(k)+1/2}B_{l\tilde{k}}.
\end{aligned}
$$

The inverse ${}_{i(k),j(k)+1/2}B_{jk}$ links the polynomial coefficents with the cell volumes $F_{\tilde{k}}(t)$; thus, it is possible to formulate a scheme solving Equation (3.1) which only makes use of the $F_{\tilde{k}}(t)$ within the stencil as in the 1D case. For instance an appropriate fourth order in x-direction and third order in y-direction approximation of function $f(x,y,t)$ at the left boundary $(x_{i(k)},y_{j(k)+1/2})$, which requires a twelve-cell stencil, is depicted in Figure 3.2.

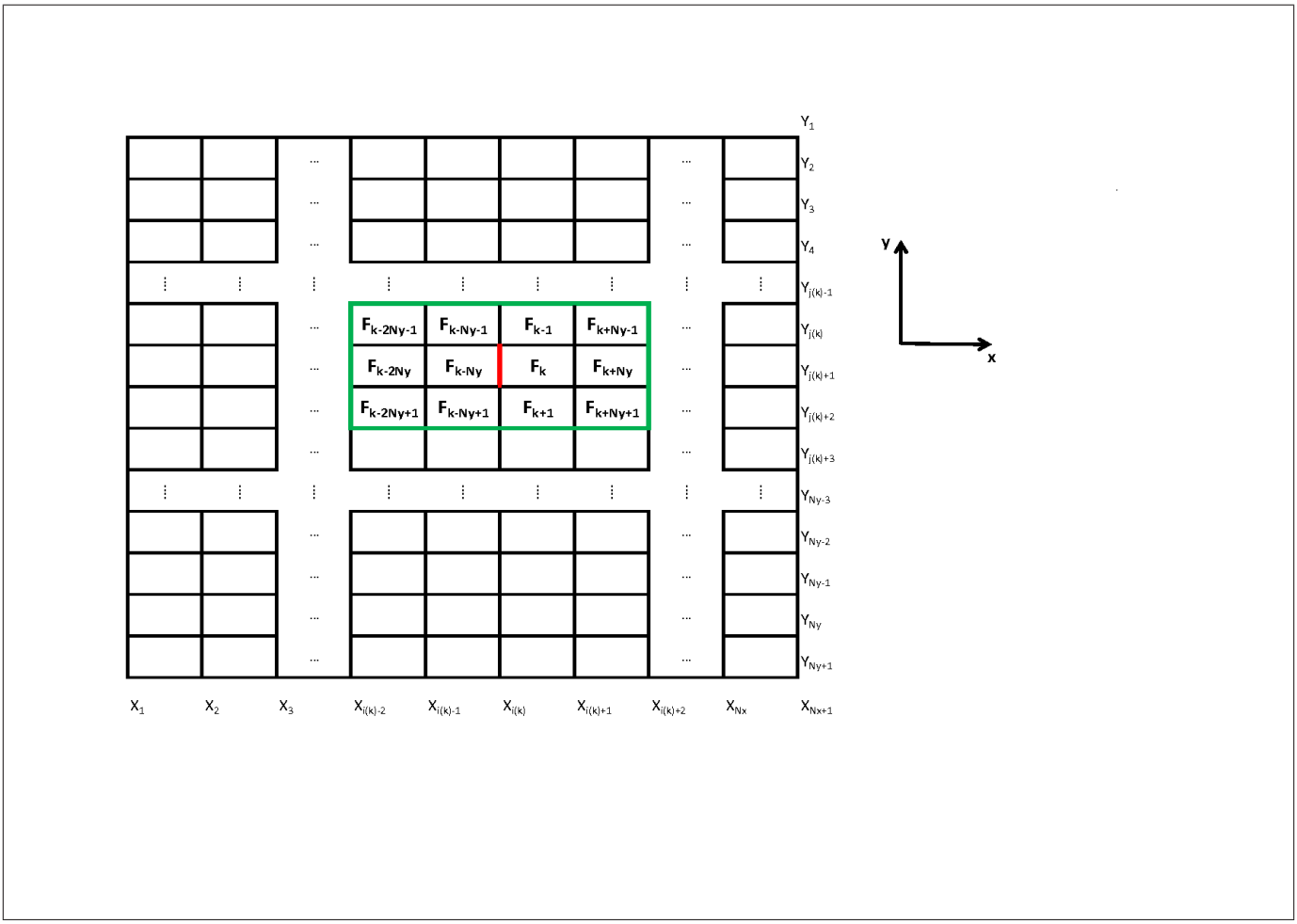

Figure 3.2: Twelve-cell stencil for fourth order in x-direction and third order in y-direction approximation of function $f(x,y,t)$ at the left boundary of cell $F_{\tilde{k}}(t)$.

Using the interpolation formula for the polynomial coefficients ${}_{i(k),j(k)+1/2}a_l(t)$, one can evaluate the nu-

merical flux difference $I_{\tilde{k}}$ given by Equation (3.4),

$$I_k(t) \quad = \quad \left[I'_k(t) - I''_k(t)\right] - \left[I'''_k(t) - I''''_k(t)\right]$$

with

$$I'_k(t) = \int_{y_{j(k)+1}}^{y_{j(k)}} \mathrm{d}y \left[D(x_{i(k)+1},y) \sum_{\substack{l=1 \\ p_x(l)\stackrel{!}{=}1}}^{m*n} \left(y - y_{j(k)+1/2}\right)^{p_y(l)} \sum_{\substack{\tilde{k} \in \text{ symmetric stencil} \\ \text{of boundary } (x_{i(k)+1},y_{j(k)+1/2})}} {}_{i(k)+1,j(k)+1/2}B_{l\tilde{k}}\, F_{\tilde{k}} - \right.$$
$$\left. -v_x(x_{i(k)+1},y) \sum_{\substack{l=1 \\ p_x(l)\stackrel{!}{=}0}}^{m*n} \left(y - y_{j(k)+1/2}\right)^{p_y(l)} \sum_{\substack{\tilde{k} \in \text{ symmetric stencil} \\ \text{of boundary } (x_{i(k)+1},y_{j(k)+1/2})}} {}_{i(k)+1,j(k)+1/2}B_{l\tilde{k}}\, F_{\tilde{k}} \right]$$

$$I''_k(t) = \int_{y_{j(k)+1}}^{y_{j(k)}} \mathrm{d}y \left[D(x_{i(k)},y) \sum_{\substack{l=1 \\ p_x(l)\stackrel{!}{=}1}}^{m*n} \left(y - y_{j(k)+1/2}\right)^{p_y(l)} \sum_{\substack{\tilde{k} \in \text{ symmetric stencil} \\ \text{of boundary } (x_{i(k)},y_{j(k)+1/2})}} {}_{i(k),j(k)+1/2}B_{l\tilde{k}}\, F_{\tilde{k}} - \right.$$
$$\left. -v_x(x_{i(k)},y) \sum_{\substack{l=1 \\ p_x(l)\stackrel{!}{=}0}}^{m*n} \left(y - y_{j(k)+1/2}\right)^{p_y(l)} \sum_{\substack{\tilde{k} \in \text{ symmetric stencil} \\ \text{of boundary } (x_{i(k)},y_{j(k)+1/2})}} {}_{i(k),j(k)+1/2}B_{l\tilde{k}}\, F_{\tilde{k}} \right]$$

$$I'''_k(t) = \int_{x_{i(k)}}^{x_{i(k)+1}} \mathrm{d}x \left[D(x,y_{j(k)+1}) \sum_{\substack{l=1 \\ \hat{p}_y(l)\stackrel{!}{=}1}}^{m*n} \left(x - x_{i(k)+1/2}\right)^{\hat{p}_x(l)} \sum_{\substack{\tilde{k} \in \text{ symmetric stencil} \\ \text{of boundary } (x_{i(k)+1/2},y_{j(k)+1})}} {}_{i(k)+1/2,j(k)+1}B_{l\tilde{k}}\, F_{\tilde{k}} - \right.$$
$$\left. -v_y(x,y_{j(k)+1}) \sum_{\substack{l=1 \\ \hat{p}_y(l)\stackrel{!}{=}0}}^{m*n} \left(x - x_{i(k)+1/2}\right)^{\hat{p}_x(l)} \sum_{\substack{\tilde{k} \in \text{ symmetric stencil} \\ \text{of boundary } (x_{i(k)+1/2},y_{j(k)+1})}} {}_{i(k)+1/2,j(k)+1}B_{l\tilde{k}}\, F_{\tilde{k}} \right]$$

$$I''''_k(t) = \int_{x_{i(k)}}^{x_{i(k)+1}} \mathrm{d}x \left[D(x,y_{j(k)}) \sum_{\substack{l=1 \\ \hat{p}_y(l)\stackrel{!}{=}1}}^{m*n} \left(x - x_{i(k)+1/2}\right)^{\hat{p}_x(l)} \sum_{\substack{\tilde{k} \in \text{ symmetric stencil} \\ \text{of boundary } (x_{i(k)+1/2},y_{j(k)})}} {}_{i(k)+1/2,j(k)}B_{l\tilde{k}}\, F_{\tilde{k}} - \right.$$
$$\left. -v_y(x,y_{j(k)}) \sum_{\substack{l=1 \\ \hat{p}_y(l)\stackrel{!}{=}0}}^{m*n} \left(x - x_{i(k)+1/2}\right)^{\hat{p}_x(l)} \sum_{\substack{\tilde{k} \in \text{ symmetric stencil} \\ \text{of boundary } (x_{i(k)+1/2},y_{j(k)})}} {}_{i(k)+1/2,j(k)}B_{l\tilde{k}}\, F_{\tilde{k}} \right],$$

whereby the hat denotes the powers of the polynomial for the rotated stencil. All contributions arising from

the numerical flux differences $I_k(t)$ are again subsumed in a matrix $\underline{\underline{\mathbf{M}}}$, as described in Section 2.2. The resulting ordinary differential equaton (2.10), which has simple dependence on time, can be solved using suitable time integration procedures, confer Section 2.3.

3.3 Implementation of Boundary Conditions in the Conservative Finite Difference Scheme for General Diffusion Equations in two Dimensions

The discussion in this section will be restricted to the implementation of Dirichlet boundary conditions that are used to describe walls of finite temperature as in the case of the investigated model for the transport in fusion plasmas. As mentioned in Section 2.4, for the computational implementation two different approaches exist which have respective pros and cons. Either one makes use of ghost cells or the interpolation function is modified at boundary of the domain. The ghost cell method can be readily extended to 2D and needs no further discussion. For the modification of the interpolation function at boundary of the domain, one has to adapt the derived formulas.
To illustrate this adaption of the formulas, we consider the stencil in Figure 3.2 and a Dirichlet boundary condition where the function $f(x,y,t)$ vanishes at the left edge of the domain. Following Equation (3.5), the function f can be expanded into a Taylor series at the boundary $(x_1, y_{j(k)+1/2})$,

$$ {}_{1,j(k)+1/2}f(x,y,t) \;=\; \sum_{l=1}^{m*n} {}_{1,j(k)+1/2}a_l(t)\;(x-x_{i(k)})^{p_x(l)}\;(y-y_{j(k)+1/2})^{p_y(l)}, $$

whereby the polynomial coefficients ${}_{1,j(k)+1/2}a_l(t)$ have to vanish for $p_x(l) \stackrel{!}{=} 0$ due to boundary condition,

$$ {}_{1,j(k)+1/2}f(x_1, y_{j(k)+1/2}, t) \;=\; 0. $$

Using the modified interpolation function, it is possible to evaluate the sought numerical flux in x-direction $\Gamma_x(x,y,t)$ (3.3) at the left edge of the domain,

$$ \begin{aligned} \Gamma_x(x_1, y_{j(k)+1/2}, t) &= D(x_1, y_{j(k)+1/2})\;\left.\frac{\partial f(x,y,t)}{\partial x}\right|_{(x_1, y_{j(k)+1/2})} + v_x(x_1, y_{j(k)+1/2})\; f(x_1, y_{j(k)+1/2}, t) \\ &= D(x_1, y_{j(k)+1/2}) \sum_{\substack{l=1 \\ p_x(l)\stackrel{!}{=}1}}^{m*n} \left(y-y_{j(k)+1/2}\right)^{p_y(l)}\; {}_{1,j(k)+1/2}a_l(t) - \\ &\quad - v_x(x_1, y_{j(k)+1/2}) \sum_{\substack{l=1 \\ p_x(l)\stackrel{!}{=}0}}^{m*n} \left(y-y_{j(k)+1/2}\right)^{p_y(l)}\; \underbrace{{}_{1,j(k)+1/2}a_l(t)}_{\to 0}\,. \end{aligned} $$

As in the 1D case, the partial derivative $\left.\frac{\partial f(x,y,t)}{\partial x}\right|_{(x_1,y_{j(k)+1/2})}$ can be approximated by a finite difference of first order accuracy,

$$\left.\frac{\partial f(x,y,t)}{\partial x}\right|_{(x_1,y_{j(k)+1/2})} = \frac{\overbrace{{}_{1,j(k)+1/2}f(x_1,y_{j(k)+1/2},t)}^{\to 0} - {}_{2,j(k)+1/2}f(x_2,y_{j(k)+1/2},t)}{\Delta x}. \tag{3.7}$$

From Equation (3.7) one eventually obtains a relation for the sought polynomial coefficient ${}_{1,j(k)+1/2}a_l(t))$,

$${}_{1,j(k)+1/2}a_l(t) = -\frac{{}_{2,j(k)+1/2}a_{l'}(t)}{\Delta x} \quad \text{for} \quad \begin{cases} p_x(l) \stackrel{!}{=} 1 \\ p_x(l') \stackrel{!}{=} 0 \end{cases}.$$

Better approximations to the derivative use finite differences with truncation errors of higher order.

3.4 Test Cases for the two dimensional Conservative Finite Difference Scheme

As in Section 2.5, the subsequent test cases study a 2D general diffusion equation (3.1) with a constant diffusion coefficient $D(x,y)=1$ and velocity of advection $\underline{\mathbf{v}}(x,y)=\begin{pmatrix} -2x \\ -2y \end{pmatrix}$,

$$\frac{\partial f(x,y,t)}{\partial t} = \underline{\boldsymbol{\nabla}}\cdot\left(\underline{\boldsymbol{\nabla}}\, f(x,y,t) - \begin{pmatrix} -2x \\ -2y \end{pmatrix} f(x,y,t)\right),$$

which is solved by a 2D Gaussian curve with a FWHM of $\sqrt{\frac{1}{2}}$ in each direction,

$$f(x,y,t) = e^{-(x^2+y^2)}.$$

Since there are no sources in the considered system, the steady-state solution will be in all test cases $f(x,y,t\to\infty)=0$ because particles get lost due to round-off errors and numerical instabilities. As an initial profile we choose a cylinder whereby the included area must be equal to the area of the 2D Gaussian $A_{\text{Gaussian2D}}=\pi$. This demand results from the conservativity of the applied numerical scheme. In the subsequent test cases the radius of the cylinder is 1 and consequently the height has to be 1. Furthermore, the chosen polynomial reconstruction is of fourth order in x-direction and third order in y-direction, respectively. An estimate for the CFL condition is calculated by following expression [30,p. 415],

$$\Delta t \leq c_{\text{CFL}}\cdot\left(\max_{i,j}\left(\frac{\left\|v_x(x_i,y_j)\right\|}{\Delta x}+\frac{\left\|v_y(x_i,y_j)\right\|}{\Delta y}\right)+2D\left(\frac{1}{(\Delta x)^2}+\frac{1}{(\Delta y)^2}\right)\right)^{-1},$$

whereby a spatial discretization of $\Delta x = 0.08$ and $\Delta y = 0.08$ is chosen.

Figure 3.3 shows the time evolution of the 2D Gaussian curve and the relative precision using a Crank-Nicolson time integrator. Under the considered CFL condition $c_{\mathrm{CFL}} = 10$ the relative precision grows only linearily and the CFDS proves to be conservative.

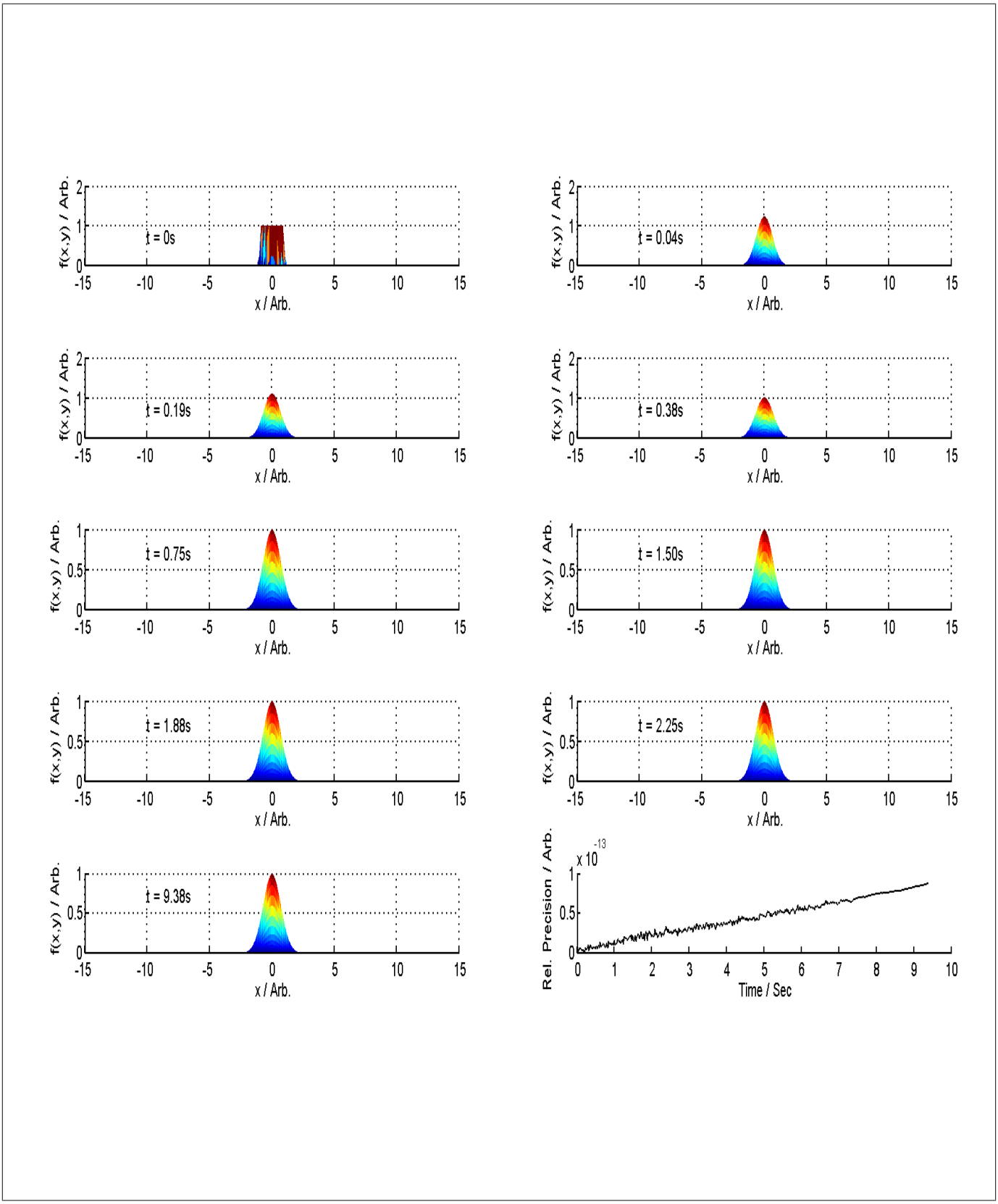

Figure 3.3: Time evolution of the 2D Gaussian curve and the relative precision using a Crank-Nicolson time integrator and a $c_{\mathrm{CFL}} = 10$. Note that the first four plots have a different scale on y-axis.

The time evolution of the Gaussian curve and the corresponding relative precision for an insufficient CFL condition is depicted in Figure 3.4 whereby the same time integration procedure is used. As the relative error grows exponentially, the CFDS is not conservative.

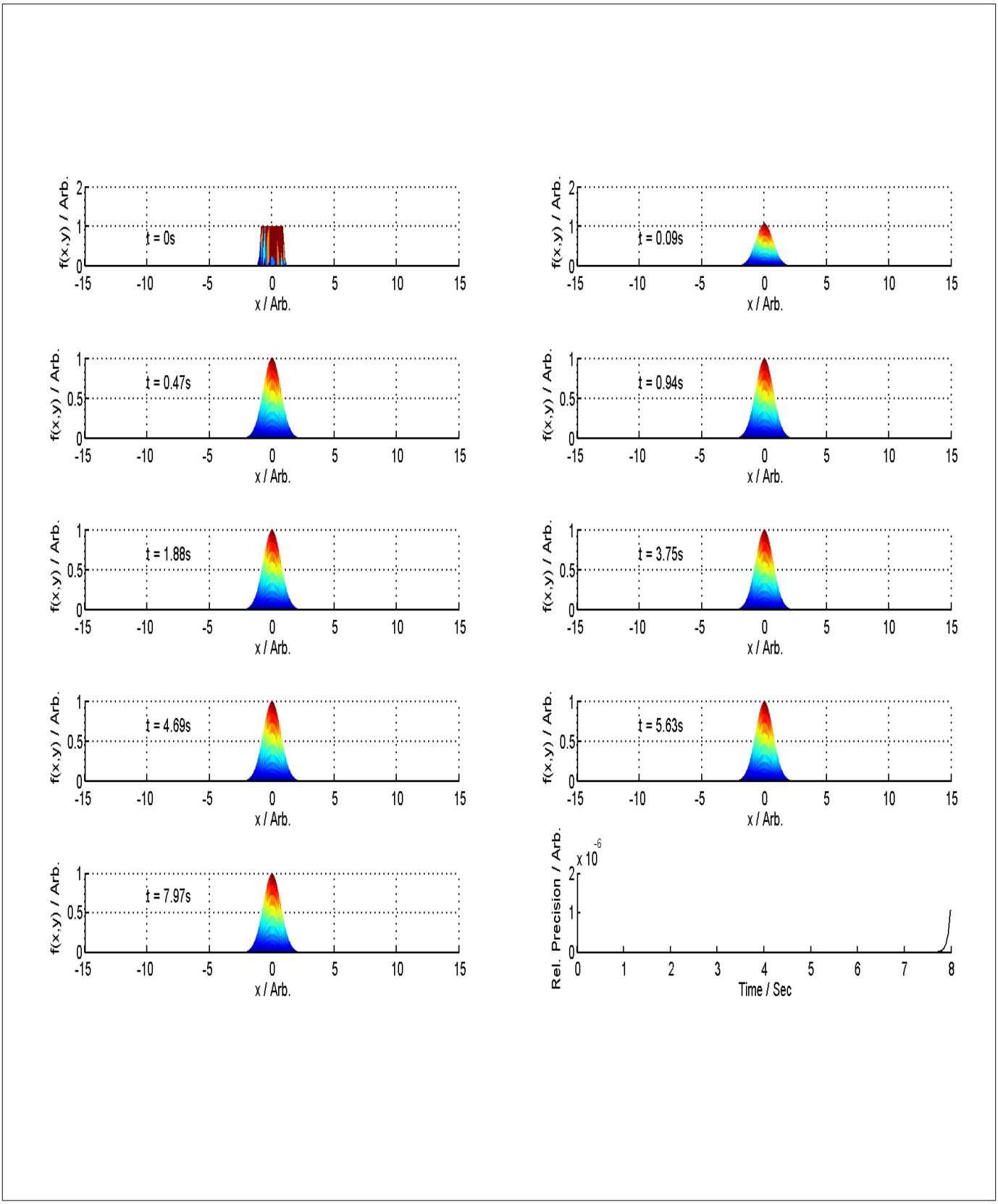

Figure 3.4: Time evolution of the 2D Gaussian curve and the relative precision using a Crank-Nicolson time integrator and a $c_{\mathrm{CFL}} = 25$. Note that the first two plots have a different scale on y-axis.

In Figure 3.5 a backward Euler time integration of the same test case is performed and the CFL condition is again $c_{\mathrm{CFL}} = 10$. The time evolution of the Gaussian curve is analogue and the relative precision decreases also linearily.

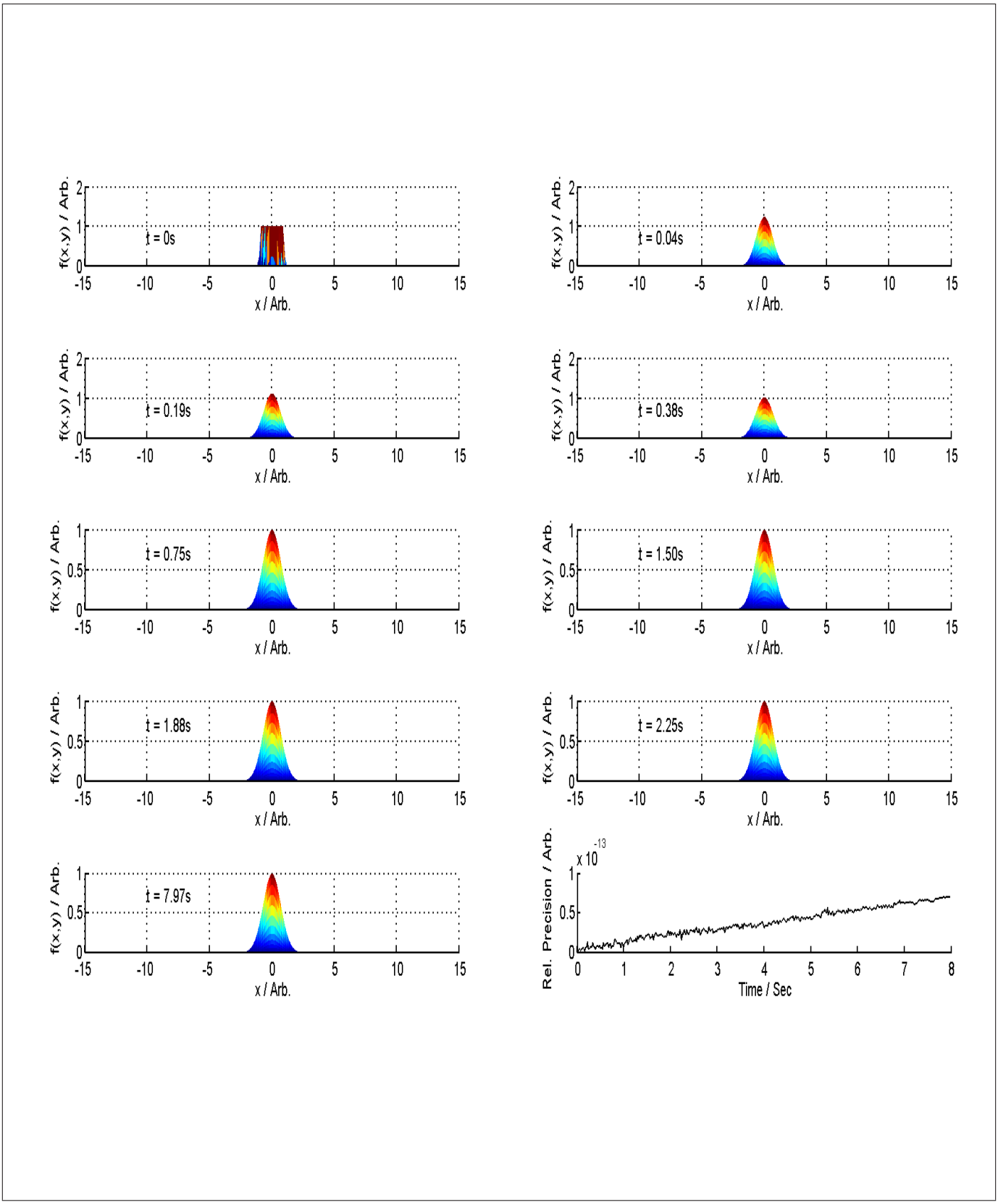

Figure 3.5: Time evolution of the 2D Gaussian curve and the relative precision using a backward Euler time integrator and a $c_{\mathrm{CFL}} = 10$. Note that the first two plots have a different scale on y-axis.

Figure 3.6 shows the time evolution of the Gaussian curve by means of a backward Euler and the corresponding relative precision for an insufficient CFL condition c_{CFL}. In comparison to the insufficient CFL condition for the Crank-Nicolson time integrator, the value of c_{CFL} is by far smaller and instabilites occur much earlier. As the relative error grows also exponentially, the CFDS is not conservative.

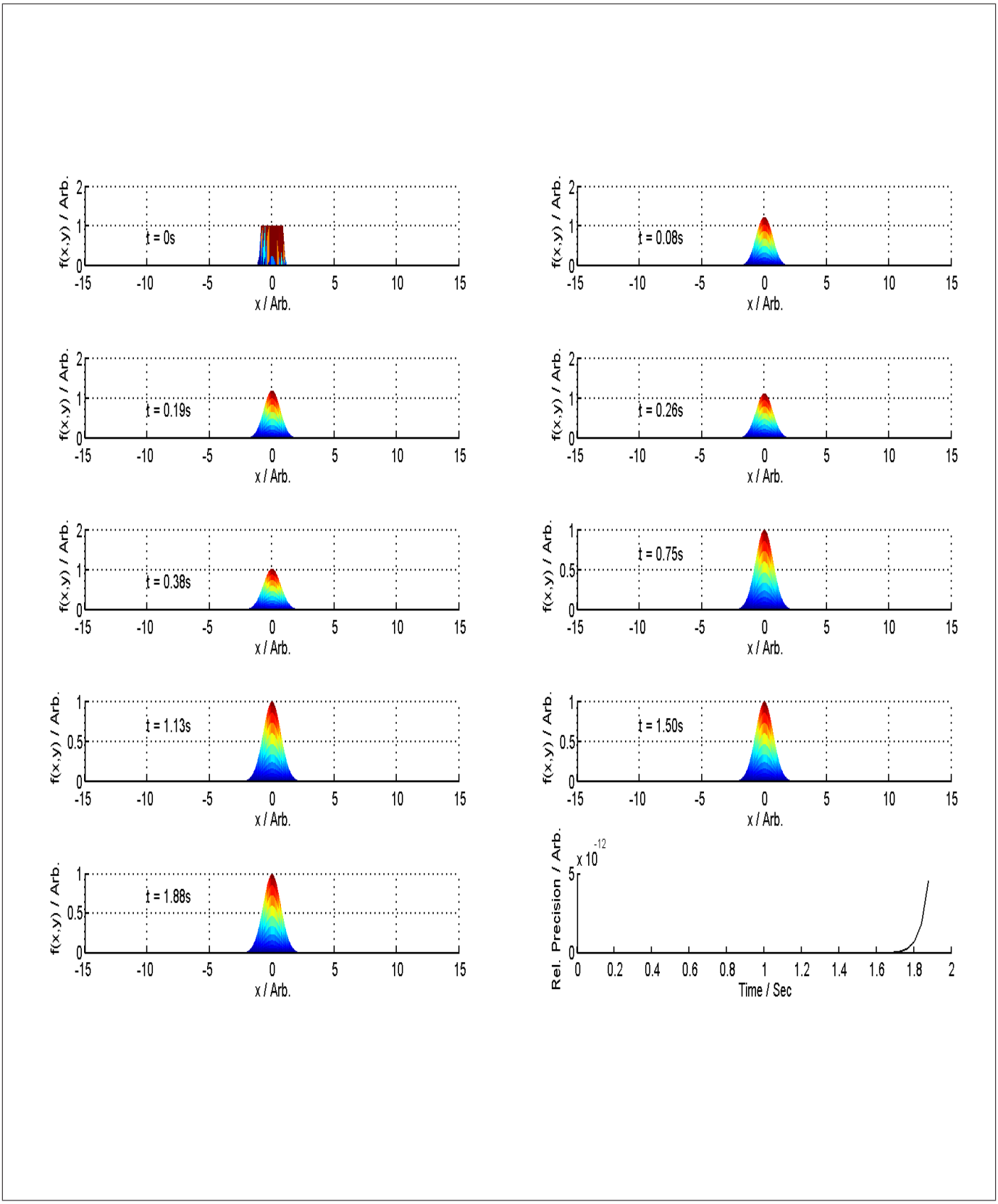

Figure 3.6: Time evolution of the 2D Gaussian curve and the relative precision using a backward Euler time integrator and a $c_{\mathrm{CFL}} = 20$. Note that the first five plots have a different scale on y-axis.

In contrast to the 1D test case, an optimal value for c_{CFL} could not be found for both time integrators. More challenging test cases concerning heat transport in magnetized fusion plasmas are performed in Chapter 4. Especially the occurring high anisotropies, which should be resolved with a high order scheme, are a benchmark for the chosen mesh refinement and the order of the polynomial reconstruction.

4 Heat Transport in Magnetized Fusion Plasmas

The so far performed test scenarios for the 2D CFDS (confer Section 3.4) can be used to validate the operability, but are certainly not suited for a benchmark under realistic conditions. A widely studied problem, which is often considered as a benchmark for PDE solvers, is the heat transport in magnetized plasmas [31, 32, 33, 34]. A mathematical description of the heat transport in magnetized plasmas is given in Section 4.1. In Section 4.2 an analytical solution for an equilibrium configuration in a Tokamak is compared to the numerical results of the CFDS. A disussion of a divertor model for the Tokamak is given in Section 4.3. Finally, Section 4.4 concludes this chapter with an outlook on future improvements of the developed CFDS.

4.1 General Diffusion Equation for Modelling Heat Transport in Magnetized Fusion Plasmas

A macroscopic description of the heat transport in magnetized fusion plasmas is given by a general diffusion equation [31],

$$\frac{\partial f}{\partial t} = \underline{\nabla} \cdot \underline{\Gamma} + q \tag{4.1}$$

with

$$\underline{\Gamma} = \overleftrightarrow{\mathbf{D}} \cdot \underline{\nabla} f - \underline{\mathbf{v}} f, \tag{4.2}$$

whereby q is a heat source and $\underline{\Gamma}$ corresponds to the numerical flux function that in turn depends on the velocity of advection or convection $\underline{\mathbf{v}}$ and on the second-order diffusivity tensor $\overleftrightarrow{\mathbf{D}}$. This second-order diffusivity tensor $\overleftrightarrow{\mathbf{D}}$ models a local anisotropic diffusion with different coefficients for parallel ($D_{\parallel}$) and perpendicular heat transport ($D_{\perp}$) [35],

$$\overleftrightarrow{\mathbf{D}} = D_{\perp} \left(\overleftrightarrow{\mathbb{1}} - \underline{\hat{\mathbf{h}}}\, \underline{\hat{\mathbf{h}}} \right) + D_{\parallel}\, \underline{\hat{\mathbf{h}}}\, \underline{\hat{\mathbf{h}}}. \tag{4.3}$$

In Equation (4.3) $\underline{\hat{\mathbf{h}}}\, \underline{\hat{\mathbf{h}}}$ denotes the dyadic product of the unit vectors in the magnetic field direction $\underline{\mathbf{B}}$,

$$\underline{\hat{\mathbf{h}}} \equiv \frac{\underline{\mathbf{B}}}{\|\underline{\mathbf{B}}\|}.$$

Since the geometries of fusion research devices, e.g. a Tokamak, can be very complicated, one is interested in having a coordinate-system independent formulation of Equation (4.1). According to the book

of D'haeseleer et al. [36], a co- and contravariant notation including the *Einstein summation convention* is introduced. Let $\underline{\mathbf{R}}(u^1,u^2,u^3)$ denote an invertible transform from Cartesian coordinates x,y,z to any curvilinear coordinate system u^1,u^2,u^3,

$$\underline{\mathbf{R}}(u^1,u^2,u^3): \quad \begin{array}{rcl} x &=& x(u^1,u^2,u^3) \\ y &=& y(u^1,u^2,u^3) \\ z &=& z(u^1,u^2,u^3) \end{array} . \tag{4.4}$$

Using this transformation, the divergence occurring in the righthand side of Equation (4.1) becomes

$$\underline{\boldsymbol{\nabla}} \cdot \underline{\boldsymbol{\Gamma}} \;=\; \frac{1}{\sqrt{g}} \frac{\partial}{\partial u_i} \left(\sqrt{g}\, \Gamma^i \right), \tag{4.5}$$

whereby $\sqrt{g}$ is the determinant of the covariant metric tensor g_{ij}, that in turn is defined as the dot product of the tangent basis vectors $\underline{\mathbf{e}}_\mathbf{i}$ and $\underline{\mathbf{e}}_\mathbf{j}$,

$$\begin{aligned} g_{ij} &\equiv \underline{\mathbf{e}}_\mathbf{i} \cdot \underline{\mathbf{e}}_\mathbf{j} \\ &= \frac{\partial \underline{\mathbf{R}}}{\partial u^i} \cdot \frac{\partial \underline{\mathbf{R}}}{\partial u^j} \\ &= \frac{\partial x}{\partial u^i}\frac{\partial x}{\partial u^j} + \frac{\partial y}{\partial u^i}\frac{\partial y}{\partial u^j} + \frac{\partial z}{\partial u^i}\frac{\partial z}{\partial u^j}. \end{aligned}$$

The numerical flux function $\underline{\boldsymbol{\Gamma}}$ (4.2) in curvilinear coordinates is obtained by the transformation of the gradient and of the second-order diffusivity tensor $\overleftrightarrow{\mathbf{D}}$,

$$\begin{aligned} \underline{\boldsymbol{\Gamma}} &= D^{ij}\, \underline{\mathbf{e}}_\mathbf{i}\, \underbrace{\underline{\mathbf{e}}_\mathbf{j}\, \underline{\mathbf{e}}^\mathbf{m}}_{\rightarrow\, \delta^m_j} \frac{\partial f}{\partial u^m} - v^i\, \underline{\mathbf{e}}_\mathbf{i}\, f \\ &= \Gamma^i\, \underline{\mathbf{e}}_\mathbf{i}. \end{aligned} \tag{4.6}$$

Using (4.5) and (4.6), one can rewrite the heat transport equation (4.1) into a general form which is valid for any curvilinear coordinate system,

$$\frac{\partial f}{\partial t} \;=\; \frac{1}{\sqrt{g}} \frac{\partial}{\partial u_i} \sqrt{g} \left(D^{ij} \frac{\partial f}{\partial u^j} - v^i\, f \right) + q. \tag{4.7}$$

Comparing Equation (4.7) to the conservative form of the 2D general diffusion equation (3.1), one recognizes that Gauss's law cannot be applied directly. A conservative formulation of Equation (4.7) is realized

by introducing a new variable, $\tilde{f} = \sqrt{g}\, f$,

$$\begin{aligned}
\frac{\partial \tilde{f}}{\partial t} &= \frac{\partial}{\partial u_i}\left(D^{ij}\,\frac{\partial \tilde{f}}{\partial u^j} - v_c^i\,\tilde{f}\right) + \tilde{q} \\
&\text{with} \\
v_c^i &= v^i + \left(D^{ij}\,\frac{1}{\sqrt{g}}\,\frac{\partial}{\partial u_i}\,\sqrt{g}\right) \\
&\text{and} \\
\tilde{q} &= \sqrt{g}\, q.
\end{aligned} \tag{4.8}$$

Furthermore, a coordinate transformation of the diffusivity tensor D^{ij} (4.3) is necessary for a generalized treatment of the heat transport as in Equation (4.7). The dyadic product occurring in the righthand side of Equation (4.3) needs no further treatment, since it transforms like the components of the vector $\underline{\hat{\mathbf{h}}}$. For this reason one has to consider only the unit tensor $\overleftrightarrow{\mathbb{1}}$. In Cartesian coordinates $\overleftrightarrow{\mathbb{1}}$ is equivalent to the Kronecker delta,

$$\overleftrightarrow{\mathbb{1}}_{\text{cart}} \equiv \delta^{ij}\,\underline{\mathbf{e_i}}\,\underline{\mathbf{e_j}}. \tag{4.9}$$

Since a tensor is independent of the choice of a particular coordinate system, the identity,

$$\mathbb{1}'^{\,ij}\,\underline{\mathbf{e}}'_{\mathbf{i}}\,\underline{\mathbf{e}}'_{\mathbf{j}} = \mathbb{1}^{lm}\,\underline{\mathbf{e_l}}\,\underline{\mathbf{e_m}}, \tag{4.10}$$

must be fulfilled. Plugging the expression for the unit tensor in Cartesian coordinates (4.9) into the identity (4.10) yields the following transformation for the unit tensor,

$$\begin{aligned}
\mathbb{1}'^{\,ij} &= \mathbb{1}^{lm}_{\text{cart}}\,\frac{\partial u'^{\,i}}{\partial R^l}\,\frac{\partial u'^{\,j}}{\partial R^m} \\
&= \underbrace{\delta^{lm}\,\frac{\partial u'^{\,i}}{\partial R^l}\,\frac{\partial u'^{\,j}}{\partial R^m}}_{\stackrel{!}{=} g^{ij}} \\
&= g^{ij}.
\end{aligned} \tag{4.11}$$

The components of the diffusivity tensor occurring in Equation (4.7) are

$$D^{ij} = D_\perp\left(g^{ij} - \hat{h}^i\,\hat{h}^j\right) + D_\parallel\,\hat{h}^i\,\hat{h}^j, \tag{4.12}$$

if Equation (4.11) is considered. In the next sections the heat transport for two magnetic field configurations in a Tokamak is studied. The considered $\underline{\mathbf{B}}$-fields are based upon an analytical model which makes use of the cylindrical symmetry of the Tokamak [37]. A model of an equilibrium configuration is presented in Section 4.2. In this instance an analytical solution for the heat transport is available which is used to

benchmark the numerical results regarding the resolution of high anisotropies. In the Section 4.3 the heat transport for a divertor configuration is illustrated.

4.2 Equilibrium Configuration in a Tokamak

Since the magnetized fusion plasma has a temperature of up to $10^8 K$, a magnetic confinement is absolutely essential for fusion research devices. The simplest concept for a closed field configuration is a Tokamak which generates a toroidal confinement [38, 39]. In order to make the discussion of the magnetic field easier, one introduces cylindrical coordinates R, ϕ, Z due to the rotational symmetry of a Tokamak,

$$\begin{aligned} x &= R\,\cos\phi \\ y &= R\,\sin\phi \\ z &= Z. \end{aligned} \tag{4.13}$$

For a Tokamak the total magnetic field can be split in a toroidal and a poloidal part,

$$\underline{\mathbf{B}} = \underline{\mathbf{B}}_{\text{tor}} + \underline{\mathbf{B}}_{\text{pol}}.$$

The toroidal magnetic field $\underline{\mathbf{B}}_{\text{tor}}$,

$$\begin{aligned} \underline{\mathbf{B}}_{\text{tor}} &= B_\phi\ \underline{\mathbf{e}}^\phi \\ &\text{with} \\ B_\phi &= \text{const.}\,, \end{aligned} \tag{4.14}$$

is achieved by external field coils. A toroidal current, flowing in the plasma, produces according to *Ampere's law* [40,p. 180] the superimposed poloidal magnetic field $\underline{\mathbf{B}}_{\text{pol}}$,

$$\begin{aligned} \underline{\mathbf{B}}_{\text{pol}} &= \underline{\boldsymbol{\nabla}} \times \underline{\mathbf{A}}_{\text{pol}} \\ &\text{with} \\ \underline{\mathbf{A}}_{\text{pol}} &= A_\phi\ \underline{\mathbf{e}}^\phi. \end{aligned} \tag{4.15}$$

In Equation (4.15) $\underline{\mathbf{A}}_{\text{pol}}$ denotes the corresponding vector potential. To obtain a model for a stable heat transport, one chooses a poloidal vector potential [37],

$$\begin{aligned} A_\phi &= A_{\phi,0}\, w(\underbrace{(R-R_0)^2+(Z-Z_0)^2}_{\equiv r^2}) \\ &\text{with} \\ w(r^2) &= 1 + r^2 \\ &\text{and} \\ A_{\phi,0} &= \text{const.}\,, \end{aligned} \tag{4.16}$$

whereby R_0, Z_0 denotes the center of the tube. Additionally, the ratio of the absolute values of the poloidal and toroidal magnetic field must be proportional to

$$\frac{\left\|\underline{\mathbf{B}}_{\text{pol}}\right\|}{\left\|\underline{\mathbf{B}}_{\text{tor}}\right\|} \propto \frac{r_{\text{L}}}{R_0\, q_{\text{factor}}}, \tag{4.17}$$

In Equation (4.17) r_{L} is the radius of the tube and q_{factor} indicates the safety factor [38,p. 53]. The values for the constants appearing in Equation (4.14) to (4.17) are chosen in such a way that a fusion research device like ITER [38,pp. 511-515] is modelled,

$$\begin{aligned} Z_0 &= 0 \\ R_0 &= 5 \\ r_{\text{L}} &= 2 \\ B_\phi &= 2 \\ q_{\text{factor}} &= 3. \end{aligned}$$

Hence, one can evaluate the components of the diffusivity tensor (4.12) in cylindrical coordinates (4.13),

$$\begin{aligned} D^{RR} &= D_\perp + \left(D_\parallel - D_\perp\right) \frac{4\,A_{\phi,0}^2\,(Z-Z_0)^2}{B_\phi^2 + 4\,A_{\phi,0}^2\,r^2} \\ D^{RZ} &= -\left(D_\parallel - D_\perp\right) \frac{4\,A_{\phi,0}^2\,(R-R_0)\,(Z-Z_0)}{B_\phi^2 + 4\,A_{\phi,0}^2\,r^2} \\ D^{ZR} &= D^{RZ} \\ D^{ZZ} &= D_\perp + \left(D_\parallel - D_\perp\right) \frac{4\,A_{\phi,0}^2\,(R-R_0)^2}{B_\phi^2 + 4\,A_{\phi,0}^2\,r^2}, \end{aligned}$$

whereby the ϕ-elements vanish due to the rotational symmetry of a Tokamak. Another criterion for a stable transport is that the heat source q in Equation (4.7) [37], that balances the heat transport perpendicular to

the magnetic field lines, is proportional to an exponential function of the vector potential,

$$\begin{aligned} q &\propto e^{-\frac{A_\phi}{\psi\sigma}} \\ &= e^{-\frac{A_{\phi,0}\, r^2}{\psi\sigma}} \\ &\text{with} \\ \psi_\sigma &\propto r_\mathrm{L}. \end{aligned}$$

Since all quantities are determined in cylindrical coordinates, one uses this representation also for the conservative heat transport equation (4.8),

$$\begin{aligned} \frac{\partial \tilde{f}}{\partial t} &= \frac{\partial}{\partial R}\left(D^{RR}\,\frac{\partial \tilde{f}}{\partial R} + D^{RZ}\,\frac{\partial \tilde{f}}{\partial Z} - \frac{D^{RR}}{R}\,\tilde{f}\right) + \\ &\quad + \frac{\partial}{\partial Z}\left(D^{ZR}\,\frac{\partial \tilde{f}}{\partial R} + D^{ZZ}\,\frac{\partial \tilde{f}}{\partial Z} - \frac{D^{ZR}}{R}\,\tilde{f}\right) + \tilde{q}, \end{aligned} \tag{4.18}$$

whereby the velocity of advection $\underline{\mathbf{v}}$ is set to zero and all derivatives in ϕ-direction vanish because of the rotational symmetry. For the considered heat transport model one is able to derive an analytical solution of Equation (4.18) [37],

$$f(r) = -\frac{\psi_\sigma}{2\, D_\perp\, A_{\phi,0}} \int_r^{r_\mathrm{L}} \mathrm{d}r'\, \frac{e^{-\frac{r'^{\,2}\, A_{\phi,0}}{\psi_\sigma}} - 1}{r'}. \tag{4.19}$$

In Figure 4.1 the numerical results for the heat transport equation (4.18), that are obtained by a 2D CFDS using a fourth order reconstruction of the numerical flux function, are compared to the analytical solution (4.19). To allow a comparison of the numerical results to the analytical solution, the domain of solution is cut along the R-axis and the ratio of parallel transport to perpendicular transport is set to 10^6. As one recognizes easily, the numerical results for mesh sizes of up to 901×901 do not reproduce the behavior of the analytical solution at the boundaries; instead, a smearing out is observed. In contrast, the core of the Tokamak is well approximated for all mesh sizes. The small gap between the amplitudes of the analytical solution and of the numerical solutions, respectively, arises from a finite mesh size.

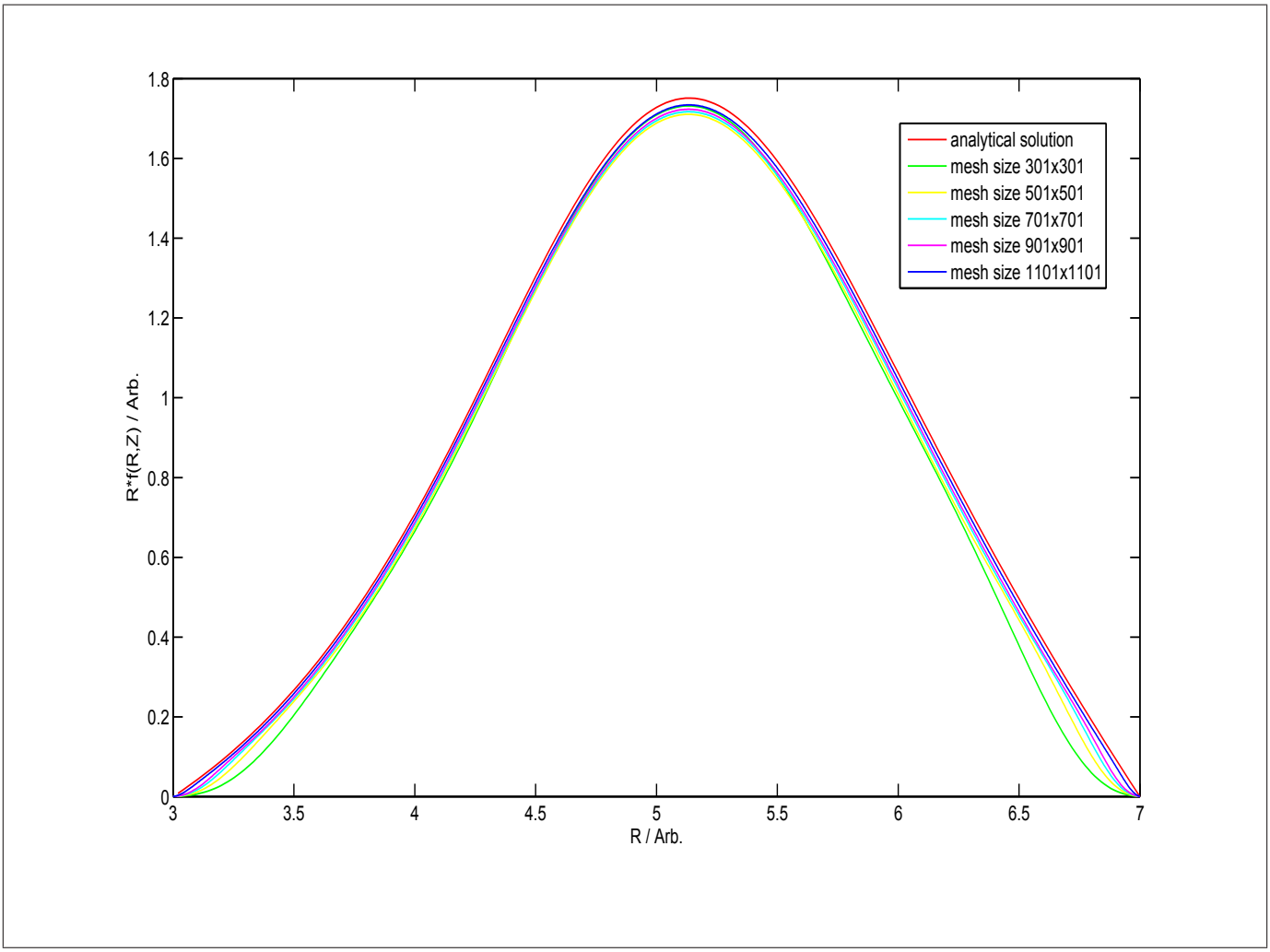

Figure 4.1: Cross section along the R-axis of an analytical solution for an equilibrium configuration in a Tokamak for $D_{\perp} = 1$ and $D_{||} = 10^6$, respectively, and comparison to numerical results using a polynomial reconstruction of fourth order in each direction and varying mesh sizes.

The surface (left) and contour plot (right) for a mesh size of 1101×1101 and for the same set of parameters as in Figure 4.1 are depicted in Figure 4.2. As predicted by the analytical solution, the computed solution has a rotational symmetry which results in concentric circles in the contour plot.

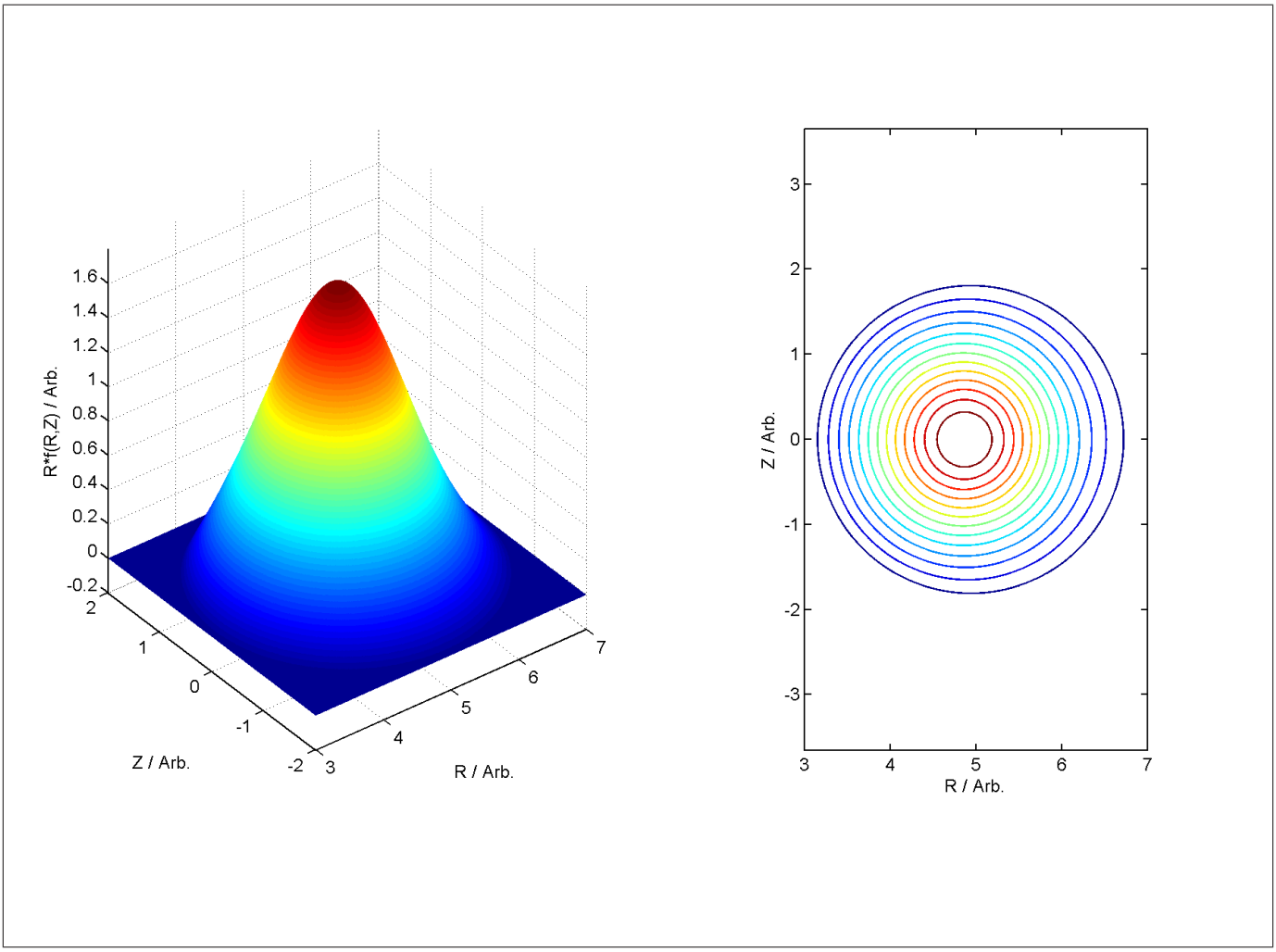

Figure 4.2: Surface (left) and contour plot (right) of an equilibrium configuration in a Tokamak for $D_{\perp} = 1$ and $D_{\parallel} = 10^6$, respectively, using a polynomial reconstruction of fourth order in each direction and a mesh size of 1101×1101.

In Figure 4.3 the growth of absolute error in dependence on the mesh size is shown, whereby a polynomial reconstruction of fourth order in each direction is used. The other parameters are chosen as in Figure 4.1. In this context, the absolute error is defined as the absolute value of the height difference at $R = 5, Z = 0$ between the numerical solution for a mesh size of 1101×1101 and the numerical solution for all other tested mesh sizes. From the slope between the last two data points, which is about three, one can estimate the order of the polynomial reconstruction. This difference indicates that the numerical solutions are not converged for the considered mesh sizes. Hence, a higher mesh refinement must be used.

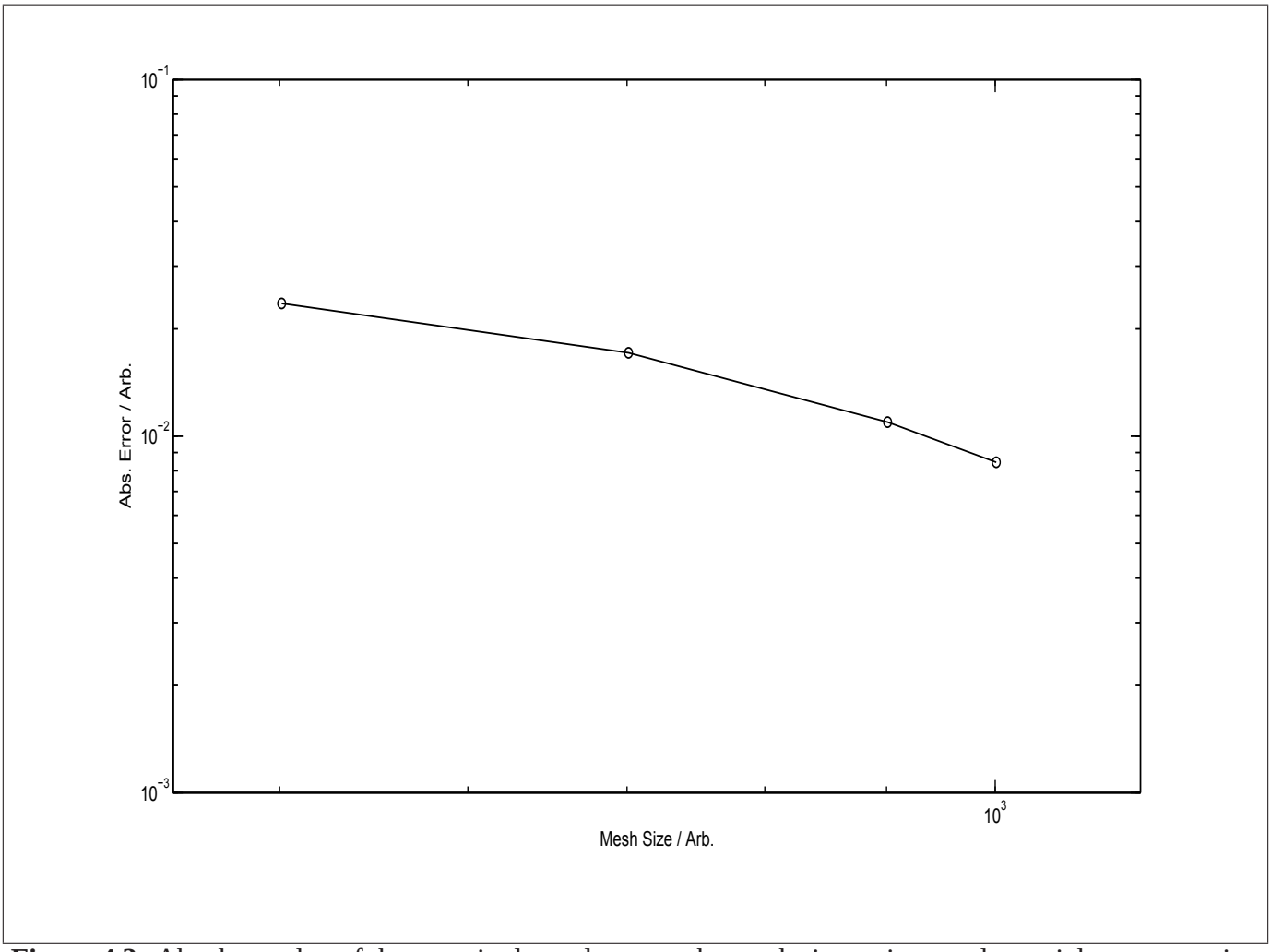

Figure 4.3: Absolute value of the error in dependence on the mesh size, using a polynomial reconstruction of fourth order in each direction. The transport coefficients of the equilibrium configuration are set to $D_\perp = 1$ and $D_\parallel = 10^6$, respectively.

The comparison of the analytical solution (4.19) with the numerical results for the heat transport equation (4.18), that are obtained by a 2D CFDS using a fifth order reconstruction of the numerical flux function, is shown in Figure 4.4. As in Figure 4.1 the domain of solution is cut along the R-axis and the ratio of parallel transport to perpendicular transport is set to 10^6. In contrast to the numerical solutions obtained by a fourth order reconstruction of the numerical flux function, the results for mesh sizes of up to 901×901 are able to reproduce the behavior of the analytical solution at the boundaries. Analoque to Figure 4.1, a small gap between the amplitudes of the analytical solution and of the numerical solutions, respectively, appears which arises from a finite mesh size.

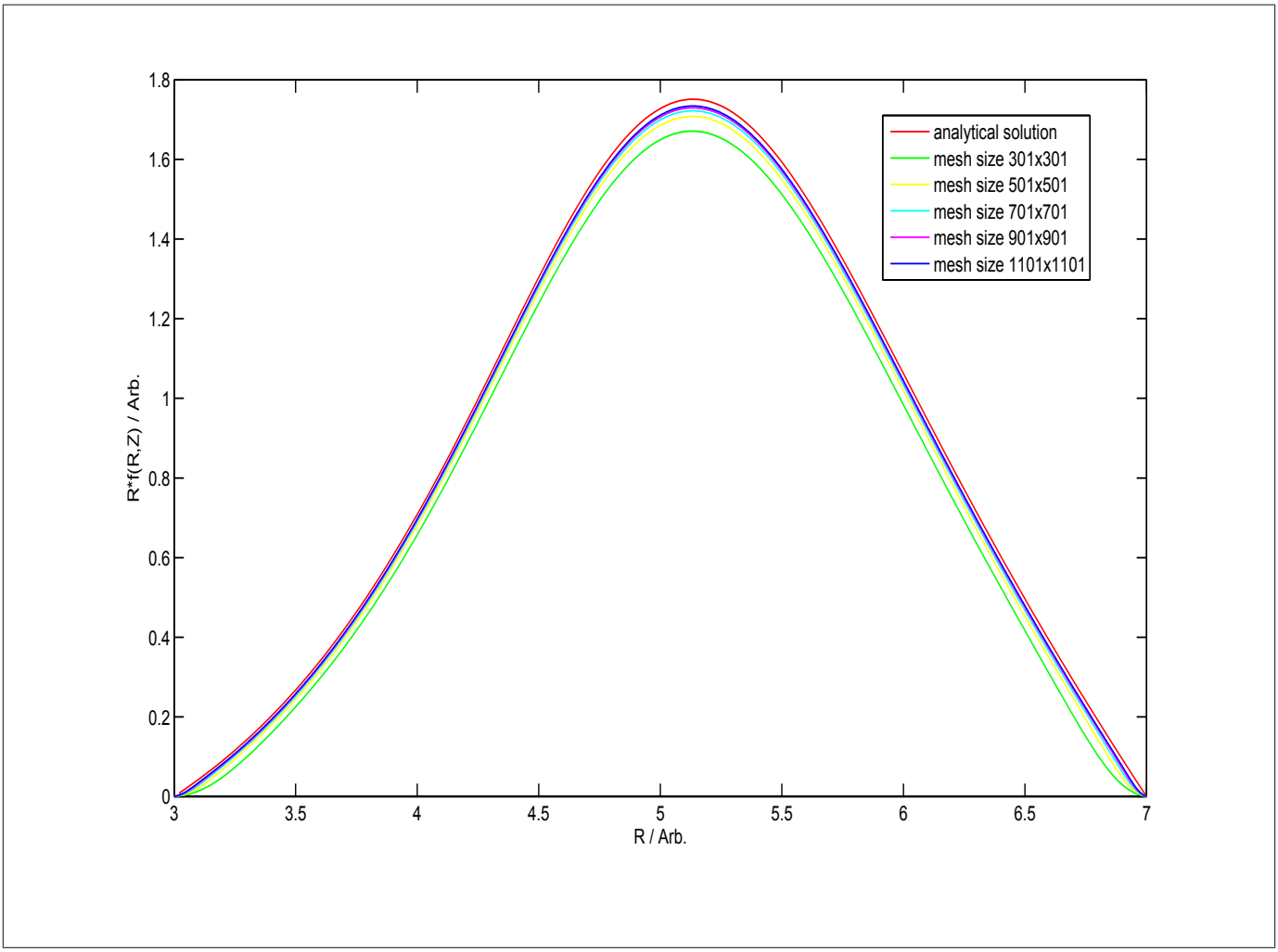

Figure 4.4: Cross section along the R-axis of an analytical solution for an equilibrium configuration in a Tokamak for $D_{\perp} = 1$ and $D_{\parallel} = 10^6$, respectively, and comparison to numerical results using a polynomial reconstruction of fifth order in each direction and varying mesh sizes.

The surface (left) and contour plot (right) for a mesh size of 1101×1101 are illustrated in Figure 4.5, using the same set of parameters as in Figure 4.4. Again, concentric circles appear in the contour plot which is due to the rotational symmetry.

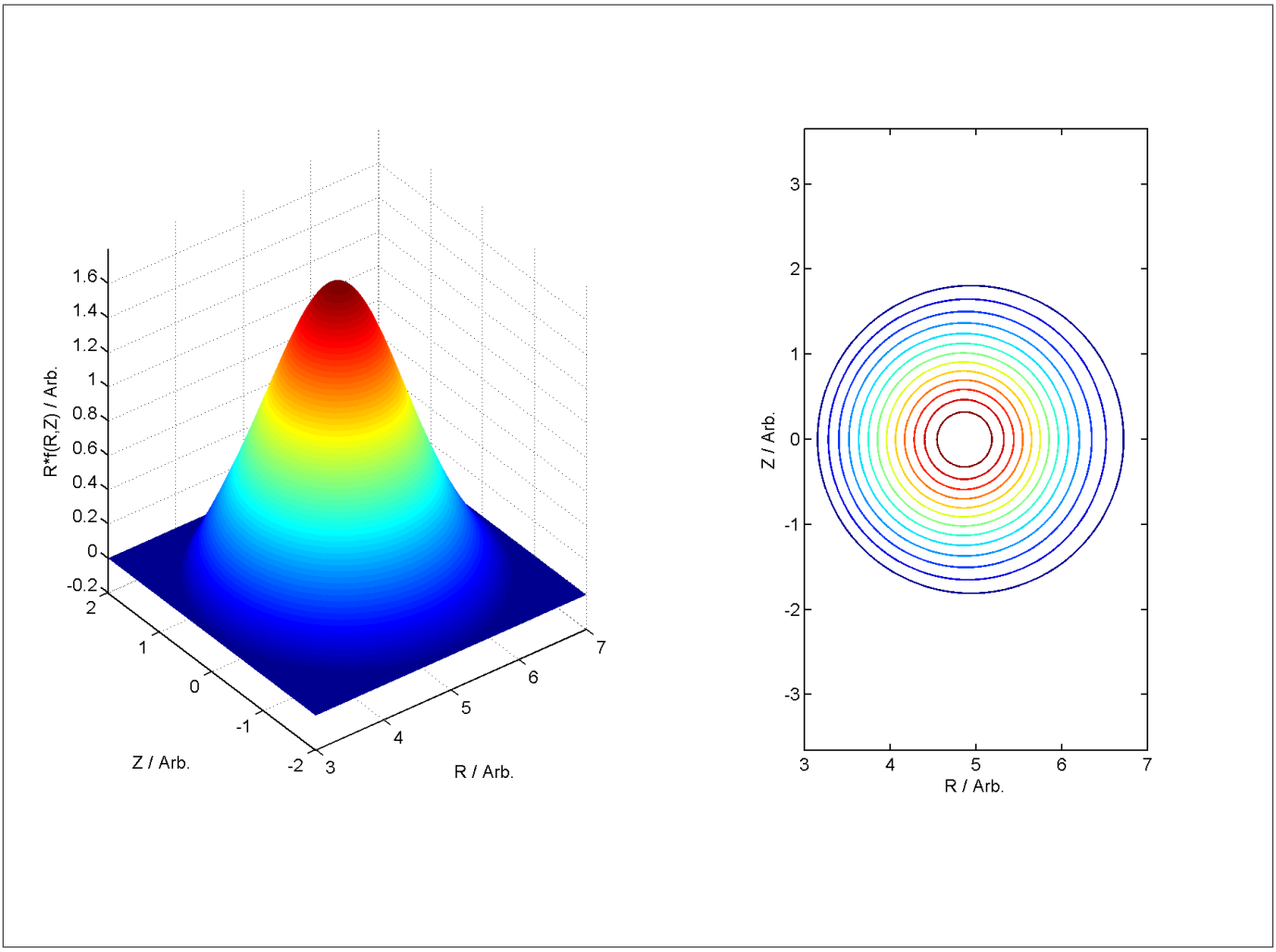

Figure 4.5: Surface (left) and contour plot (right) of an equilibrium configuration in a Tokamak for $D_{\perp} = 1$ and $D_{\parallel} = 10^6$, respectively, using a polynomial reconstruction of fifth order in each direction and a mesh size of 1101×1101.

In Figure 4.6 the absolute error in dependence on the mesh size is shown again, whereby a polynomial reconstruction of fifth order in each direction is used. The other parameters are chosen as in Figure 4.4. From the slope between the last three data points, which is about five, it is possible to make an estimate for the order of the polynomial reconstruction. This value is in good agreement with the order of the polynomial reconstruction of the numerical fluxes.

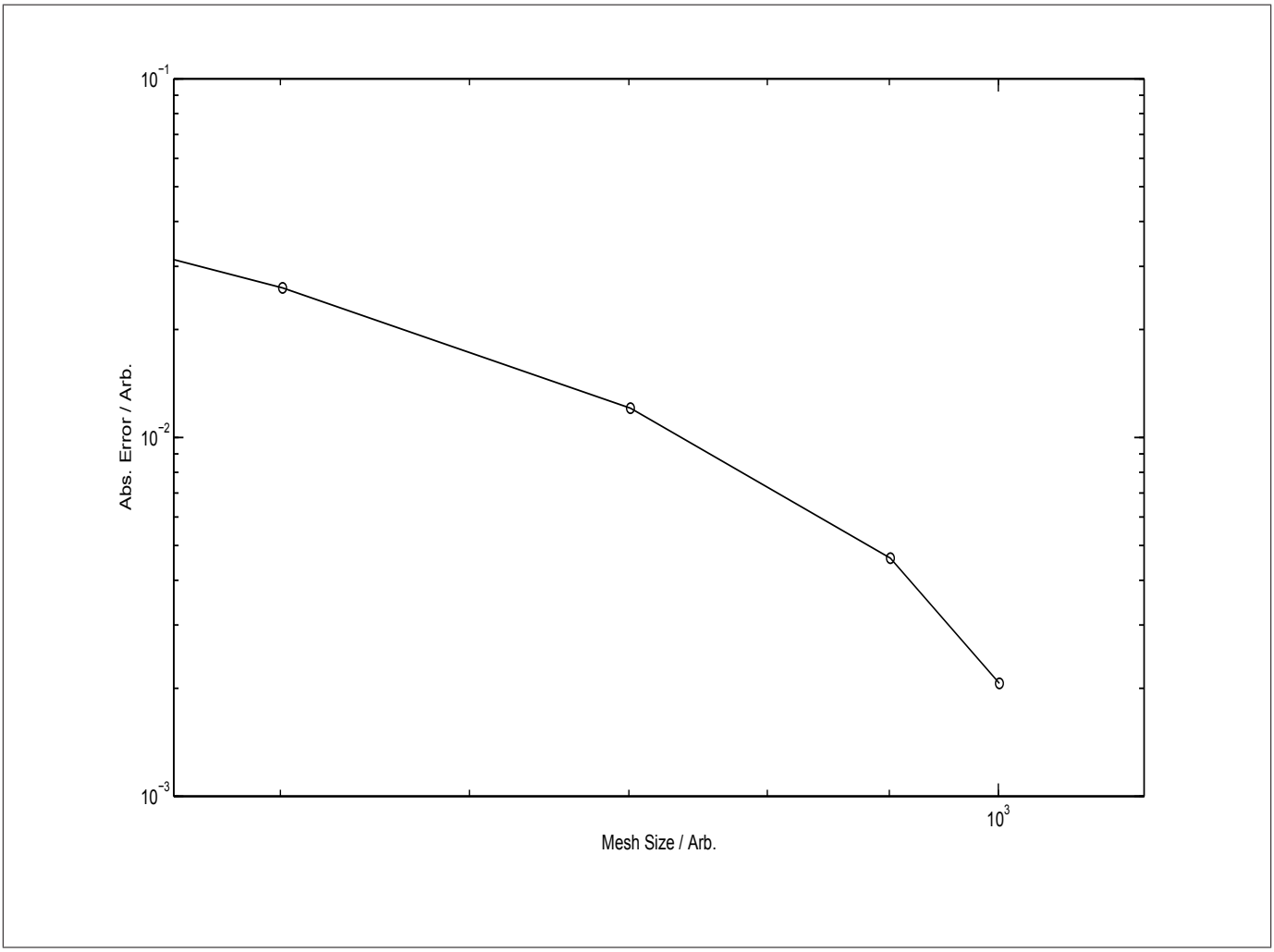

Figure 4.6: Absolute value of the error in dependence on the mesh size using a polynomial reconstruction of fifth order in each direction in the case of an equilibrium configuration in a Tokamak for $D_{\perp} = 1$ and $D_{\parallel} = 10^6$, respectively.

In Figure 4.7 the numerical results for the heat transport equation (4.18), that are obtained by a 2D CFDS using a fifth order reconstruction of the numerical flux function, are compared to the analytical solution (4.19). For the comparison the domain of solution is again cut along the *R*-axis and the ratio of parallel transport to perpendicular transport is set to 10^8. In this case, the numerical results for mesh sizes of up to 1101×1101 reproduce neither the behavior of the analytical solution at the boundaries nor the core of the Tokamak. To get a better agreement with the analytical solution, one has to make use of a higher mesh refinement or of a polynomial reconstruction of higher order. This brute force strategy is limited due to the vast computational costs; instead, it is better to use meshes with an adaptive refinement corresponding to a test function, whereby the test function is a guess for the true solution. A computational infrastructure for the adaptive meshes is under way.

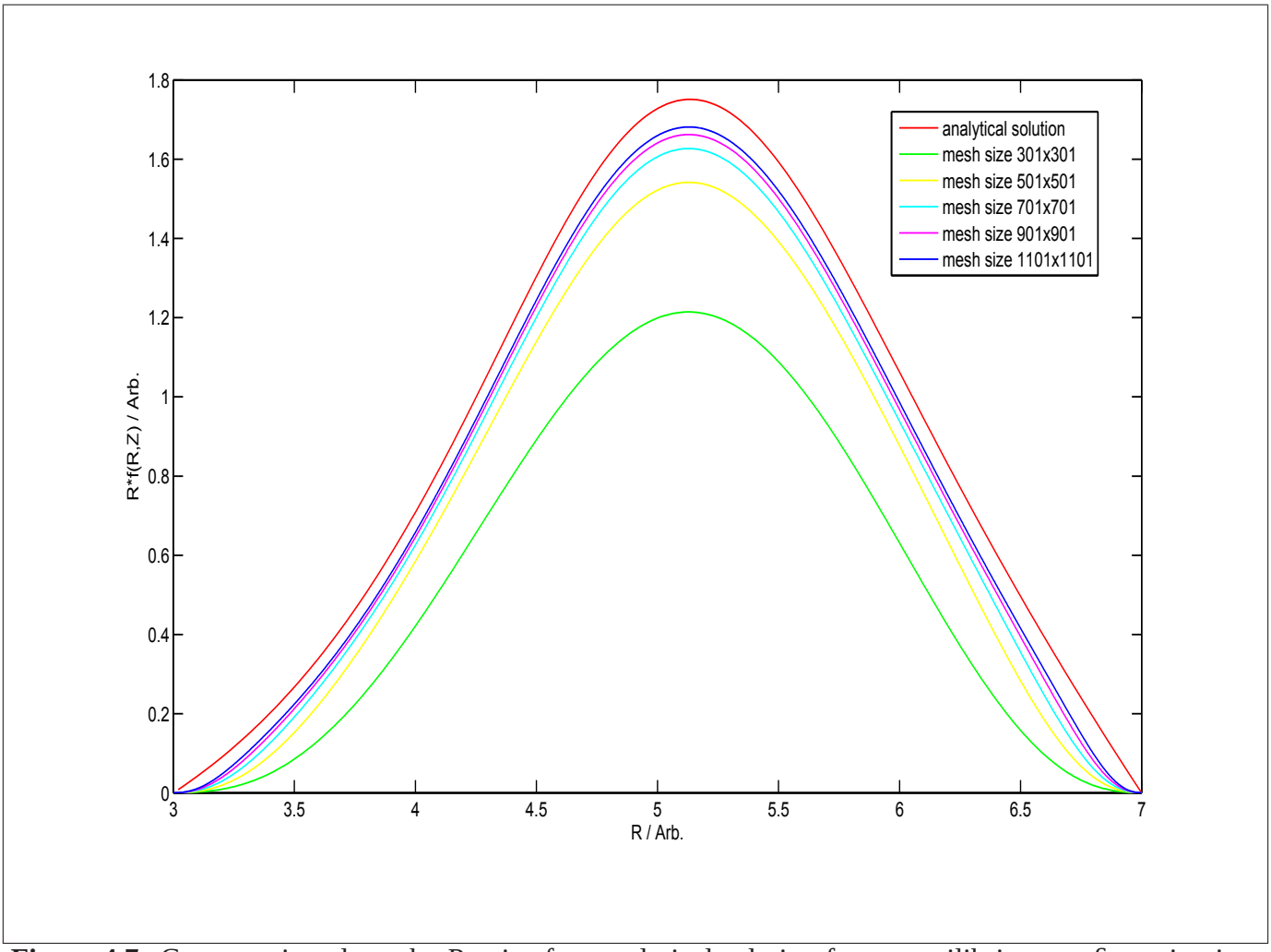

Figure 4.7: Cross section along the R-axis of an analytical solution for an equilibrium configuration in a Tokamak for $D_{\perp} = 1$ and $D_{||} = 10^8$, respectively, and comparison to numerical results using a polynomial reconstruction of fifth order in each direction and varying mesh sizes.

The surface (left) and contour plot (right) for a mesh size of 1101×1101 and for the same set of parameters as in Figure 4.7 are shown in Figure 4.8. Although the height of the numerical results does not converge to the height of the analytical solution, the surface plot has as predicted a rotational symmetry, which in turn yields the concentric circles in the contour plot.

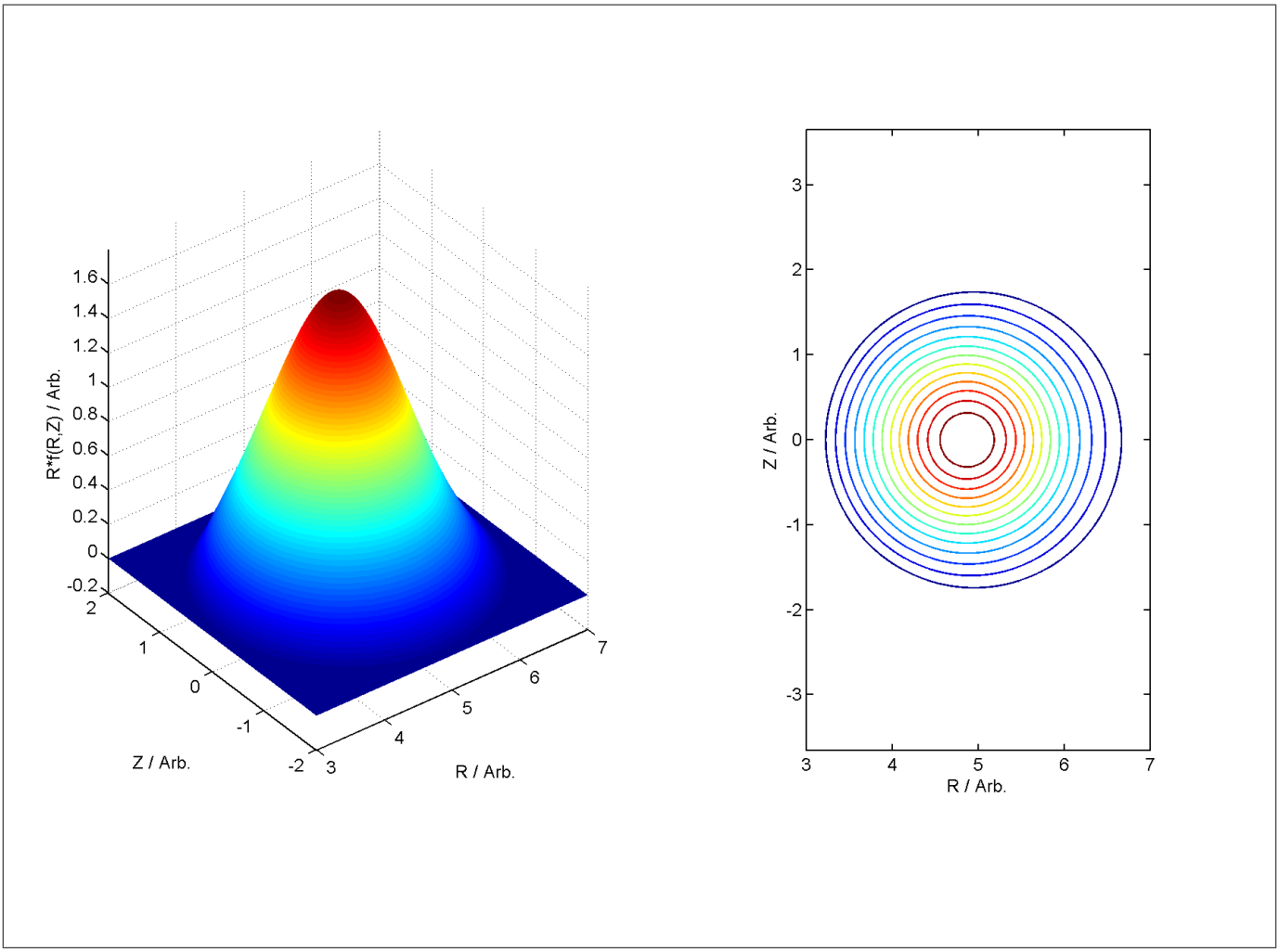

Figure 4.8: Surface (left) and contour plot (right) of an equilibrium configuration in a Tokamak for $D_{\perp} = 1$ and $D_{\parallel} = 10^8$, respectively, using a polynomial reconstruction of fifth order in each direction and a mesh size of 1101×1101.

In Figure 4.9 the absolute error in dependence on the mesh size is shown again, whereby a polynomial reconstruction of fifth order in each direction is used. All other parameters are chosen as in Figure 4.7. The slope between the last three data points is approximately five. This value is in a good agreement with the order of the polynomial reconstruction of the numerical fluxes.

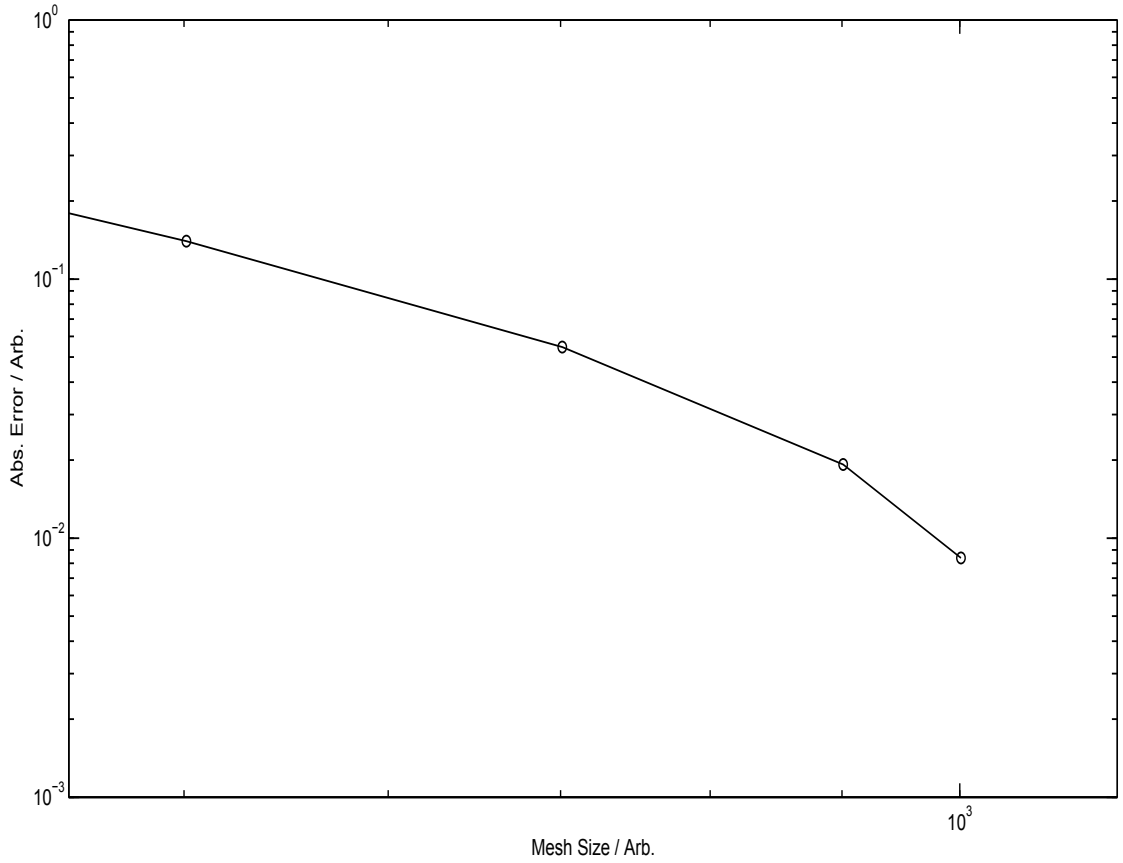

Figure 4.9: Absolute value of the error in dependence on the mesh size using a polynomial reconstruction of fifth order in each direction in the case of an equilibrium configuration in a Tokamak for $D_\perp = 1$ and $D_\parallel = 10^8$, respectively.

A divertor model for a Tokamak is studied in the next section. Contrary to the model investigated within this section no analytical solution exists. For this reason it is only possible to make a qualitative comparison between the numerical result and the assumed solution.

4.3 Divertor Model for the Tokamak

A divertor in a Tokamak is a magnetic field configuration which is produced by additional external coils in order to divert the outermost magnetic field lines out of the main plasma into a separate chamber where they intersect a material divertor target [39,pp.331-360]. This divertor configuration is used in fusion devices to exhaust the burnt fuel. In contrast to the limiter concept, in which the material target is in contact with main plasma, the recycling of neutrals and the production of impurities is spatially separated from the core plasma. Practically, poloidal divertors are chosen because the poloidal field is easier to divert than the larger toroidal field. The poloidal divertor configuration uses coils carrying current in the same direction as the plasma current for the purpose of the formation of null- or X-point, as depicted in Figure 4.10. A X-Point

marks the spatial position where the net poloidal field, that is generated by the toroidal plasma current and divertor coils, vanishes. The last closed flux surface, also called separatrix, is produced by the magnetic field lines passing through the X-Point. As desired, the separatrix devides the plasma into a core region, where the particles are confined, and into the scrape-off layer. Particles crossing this scrape-off layer move along the magnetic fields lines to the divertor region where they are exhausted.
So as to adapt the analytical model described in Section 4.2 to an divertor configuration, a new poloidal field is introduced [37],

$$\begin{aligned} A_\phi &= A_{\phi,0}\,\log\left(1 + \frac{(R-R_0)^2+(Z-Z_0)^2}{r_0^2}\right) + \\ &\quad + A_{\phi,1}\,\log\left((R_0-R_1)^2+(Z-Z_1)^2\right) \\ &\text{with} \\ A_{\phi,1} &= 0.7\,A_{\phi,0}\,. \end{aligned} \tag{4.20}$$

In the calculations following values of the constants are chosen,

$$\begin{aligned} r_\mathrm{L} &= 2 \\ r_0 &= 0.2 \cdot r_\mathrm{L} \\ Z_0 &= 0 \\ Z_0 &= -3 \cdot r_0 \\ R_0 &= 5 \\ B_\phi &= 2 \\ q_\mathrm{factor} &= 3. \end{aligned} \tag{4.21}$$

The value for the constant $A_{\phi,0}$ is calculated according to the relation given by Equation 4.17. Based on the above-described poloidal magnetic field, one is able to compute the required components for the diffusivity tensor (4.12) appearing in Equation 4.18. In Figure 4.10 a surface plot of the modelled divertor configuration using a polynomial reconstruction of fifth order in each direction is shown. For the numerical result a mesh size of 501×501 is used and the ratio of parallel transport to perpendicular transport is set to 10^2. On a larger length scale (left) only the behavior of the hot core plasma is observed. As expected, on shorter length scales (right) one recognizes the typical X-Point including the separatrix.

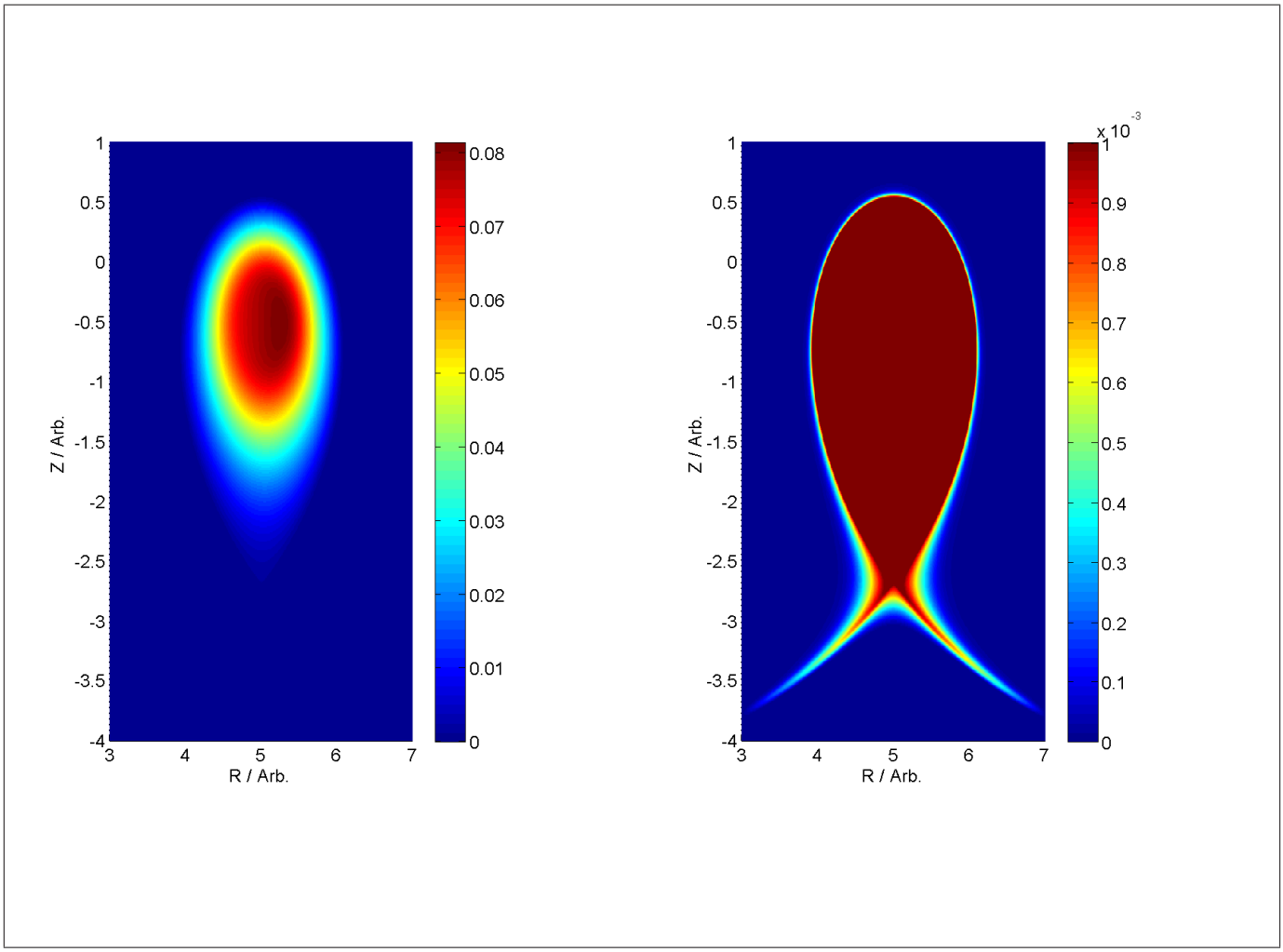

Figure 4.10: Surface plots showing a divertor configuration in a Tokamak on a larger length scale (right) and on a shorter length scale (left). The numerical result is obtained by a polynomial reconstruction of fifth order in each direction and a mesh size of 501×501, whereby the transport coefficients are set to $D_{\perp} = 1$ and $D_{\parallel} = 10^2$, respectively.

4.4 Conclusion and Outlook

We studied the heat transport in magnetized plasma which is a benchmark for the resolution of a numerical scheme regarding the high anisotropies of the transport coefficients. In realistic models for Tokamaks the ratio of parallel transport ($D_{\parallel}$) to perpendicular transport ($D_{\perp}$) varies between 10^8 and 10^{12} [31, 34]. The developed CFDS using an equidistant regular mesh is able to resolve ratios of $D_{\parallel}$ to $D_{\perp}$ of up to 10^9. Considering the simple analytical model (Section 4.2), one recognizes the massive influence of anistropy on the resolution of the numerical scheme. While a fifth order scheme is able to reproduce an anistropy of 10^6, it fails in the case of an anisotropy of 10^8. The divertor configuration (Section 4.3) is used to test, if more complex $\underline{\mathbf{B}}$-field configurations can be resolved. Using a fifth order scheme, a qualitatively correct result is obtained In order to exceed this limit an adaptive mesh has been implemented, which is a part of ongoing research. The so far obtained results using adaptive meshes show very promising prospects.

References

[1] Reginald P. Tewarson. *Sparse Matrices*. Mathematics in Science and Engineering. Academic Press, 1973. ISBN 9780126856507.

[2] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in FORTRAN: The Art of Scientific Computing*. Cambridge University Press, 1992. ISBN 052143064X.

[3] David Poole. *Linear Algebra: A Modern Introduction (2nd ed.)*. Cengage Learning, 2006. ISBN 0534998453.

[4] Gene H. Golub and Charles F. van Van Loan. *Matrix Computations (3rd Edition)*. The Johns Hopkins University Press, 1996. ISBN 0-8018-5413-X.

[5] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3): 720–755, 1999.

[6] James W. Demmel, John Gilbert, and Xiaoye S. Li. SuperLU Users' Guide. Technical report, University of California at Berkeley, Berkeley, CA, USA, 1999. URL `http://crd-legacy.lbl.gov/~xiaoye/SuperLU/`.

[7] Timothy A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30:165–195, June 2004. ISSN 0098-3500.

[8] Timothy A. Davis. Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30:196–199, June 2004. ISSN 0098-3500.

[9] Timothy A. Davis and Iain S. Duff. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. *ACM Trans. Math. Softw.*, 25:1–20, March 1999. ISSN 0098-3500.

[10] Timothy A. Davis. UMFPACK Version 5.5.0 User Guide. Technical report, Dept. of Computer and Information Science and Engineering, Univ. of Florida, Gainesville, FL, 2009. URL `http://www.cise.ufl.edu/research/sparse/SuiteSparse/`.

[11] Zahari Zlatev. *Computational Methods for General Sparse Matrices*. Mathematics and Its Applications. Kluwer Academic Publishers, 1991. ISBN 0-7923-1154-X.

[12] Timothy A. Davis, John R. Gilbert, Stefan I. Larimore, and Esmond G. Ng. A column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30:353–376, September 2004. ISSN 0098-3500.

[13] Timothy A. Davis, John R. Gilbert, Stefan I. Larimore, and Esmond G. Ng. Algorithm 836: Colamd, a column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30:377–380, September 2004. ISSN 0098-3500.

[14] Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. Algorithm 837: Amd, an approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30:381–388, September 2004. ISSN 0098-3500.

[15] Patrick R. Amestoy, Timothy a. Davis, and Iain S. Duff. An Approximate Minimum Degree Ordering Algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996. ISSN 08954798.

[16] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems*, 20(3):475–487, April 2004. ISSN 0167739X.

[17] O. Schenk and K. Gärtner. On fast factorization pivoting methods for symmetric indefinite systems. *Elec. Trans. Numer. Anal.*, 23:158–179, 2006.

[18] O. Schenk and K. Gärtner. PARDISO User Guide Version 4.0.0. Technical report, Institute of Computational Science, USI Lugano, Switzerland, 2009. URL `http://www.pardiso-project.org/`.

[19] MATLAB. *version 7.10 (R2010b)*. The MathWorks Inc., Natick, Massachusetts, 2010. URL `http://www.mathworks.de/products/matlab/`.

[20] John W. Eaton, David Bateman, and Søren Hauberg. *GNU Octave Manual Version 3*. Network Theory Limited, 2008. ISBN 0-9546120-6-X. URL `http://www.gnu.org/software/octave/`.

[21] Ilja N. Bronstein, Konstantin A. Semendjajew, Gerhard Musiol, and Heiner Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 2005. ISBN 3-8171-2006-2.

[22] Nicholas I. M. Gould, Jennifer A. Scott, and Yifan Hu. A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations. *ACM Trans. Math. Softw.*, 33, June 2007. ISSN 0098-3500.

[23] H K Versteeg and W Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Longmans, Harlow, 1995. ISBN 0-582-21884-5.

[24] Joel H Ferziger and Milovan Peric. *Computational Methods for Fluid Dynamics; 3rd ed.* Springer, Berlin, 1999. ISBN 3-540-42074-6.

[25] Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002. ISBN 0-521-01087-6.

[26] Sergei Kasilov. A new approach for reconstruction of numerical fluxes in 1D by means of polynomial interpolation. Private Communication, 2011.

[27] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer Series in Computational Mathematics. Springer, 2006. ISBN 3-540-30663-3.

[28] Charles Hirsch. *Numerical Computation of INTERNAL AND EXTERNAL FLOWS: Fundamentals of Numerical Dyiscretization*. John Wiley & Sons, Inc., New York, NY, USA, 1988. ISBN 0-471-91762-1.

[29] J. Blazek. *Computational Fluid Dynamics: Principles and Applications*. Elsevier Academic Press, 2001. ISBN 0-080-43009-0.

[30] B. Cockburn, C. Johnson, C.-W. Shu, and E. Tadmor. *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations: Lectures Given at the 2nd Session of the Centro Internazionale Matematico Estivo (C.I.M.E.) Held in Cetraro, Italy, June 23-28, 1997*. Lecture Notes in Mathematics. Springer, 1998. ISBN 3-540-64977-8.

[31] S. Günter, Q. Yu, J. Krüger, and K. Lackner. Modelling of heat transport in magnetised plasmas using non-aligned coordinates. *J. Comput. Phys.*, 209(1):354–370, October 2005. ISSN 0021-9991.

[32] S. Günter, K. Lackner, and C. Tichmann. Finite element and higher order difference formulations for modelling heat transport in magnetised plasmas. *J. Comput. Phys.*, 226(2):2306–2316, October 2007. ISSN 0021-9991.

[33] S. Günter and K. Lackner. A mixed implicit-explicit finite difference scheme for heat transport in magnetised plasmas. *J. Comput. Phys.*, 228(2):282–293, February 2009. ISSN 0021-9991.

[34] C.R. Sovinec, A.H. Glasser, T.A. Gianakon, D.C. Barnes, R.A. Nebel, S.E. Kruger, D.D. Schnack, S.J. Plimpton, A. Tarditi, and M.S. Chu. Nonlinear magnetohydrodynamics simulation using high-order finite elements. *Journal of Computational Physics*, 195(1):355 – 386, 2004. ISSN 0021-9991.

[35] S. I. Braginskii. Transport processes in a plasma. *Reviews of Plasma Physics*, 1:205–311, 1965.

[36] W. D. D'haeseleer, W. N. G. Hitchon, J. D. Callen, and J. L. Shohet. *Flux Coordinates and Magnetic Field Structure*. Springer-Verlag, Berlin, 1991. ISBN 0-387-52419-3.

[37] Sergei Kasilov. Heat transport in magnetized fusion plasmas. Private Communication, 2011.

[38] Weston M. Stacey. *Fusion Plasma Physics*. Physics Textbook. Wiley-VCH, 2005. ISBN 3-527-40586-0.

[39] Weston M. Stacey. *Fusion: An Introduction to the Physics and Technology of Magnetic Confinement Fusion*. Physics Textbook. Wiley-VCH, 2010. ISBN 978-3-527-40967-9.

[40] John David Jackson. *Classical Electrodynamics Third Edition*. John Wiley & Sons, Inc., 1998. ISBN 0-471-30932-X.