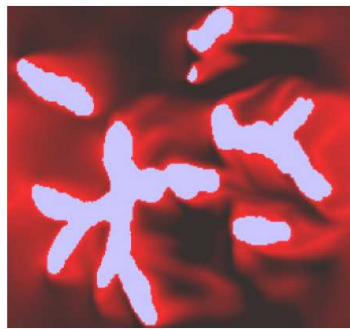


IMEX/FINITE VOLUME FOR CHEMICAL REACTIONS IN FLUID FLOW



Student: VU VAN THIEU¹
Supervisors: Joke Blom² and Jaap Kaandorp¹

¹ *Universiteit van Amsterdam*

² *Centrum voor Wiskunde en Informatica*



UNIVERSITEIT VAN AMSTERDAM

2005

Master thesis

IMEX/FINITE VOLUME
FOR CHEMICAL REACTIONS IN FLUID FLOW

Student: VU VAN THIEU

Faculty of Science,
University of Amsterdam,
The Netherlands.

Amsterdam, 2005

ACKNOWLEDGMENT

I am thankful to the organizers of the Master program in Computer Science, University of Amsterdam, for giving me a chance to study and live in The Netherlands.

I would like to thank to the people working in the International office and the secretaries of the Faculty, who have helped me for all documental matters.

My thankfulness sends to the lecturers for transmitting me interesting and worthful knowledge. Gradually, I have passed initial difficulties of having different background, and go confidently. I have now more insight in the mathematic and computer science world. I can stand firm on my own research.

I am grateful to my daily supervisors. You have been devoted to supervising me on the thesis. I learn much from you, not only in science but also the method of study. Working with you, I am more mature in research.

I thank you all for giving me a very good feeling about the country and the people.

Last but not least, I would like to express my deep gratitude to my family members, who always rely on me and support me continually on my study and personal life.

Amsterdam, 2005

Vu Van Thieu

CONTENTS

1. <i>Introduction</i>	7
2. <i>The mathematical model</i>	13
2.1 The model of reacting chemical species	13
2.2 The model of fluid flow	14
3. <i>Numerical method</i>	16
3.1 Spatial discretization	16
3.1.1 The Navier-Stokes equation	17
3.1.2 The reacting chemical species equation	24
3.1.3 The boundary conditions	24
3.2 Time integration	26
3.2.1 Original explicit Runge Kutta method	26
3.2.2 The explicit Runge Kutta Chebyshev method	28
3.2.3 The implicit-explicit Runge Kutta Chebyshev method	29
3.2.4 Damping parameter selection and stability region	30
3.3 Variable step size control and stage selection	30
3.3.1 Local error estimation	31
3.3.2 Critical step sizes for advection-diffusion equations	31
3.3.3 Stage selection and step size adjustment	32
4. <i>Implementation</i>	34
4.1 Implementation of the spatial discretization	34
4.2 Implementation of the time integration	38
4.3 The main function	43
5. <i>Numerical experiments</i>	45
5.1 Spatial discretization tests	45
5.1.1 Test for discretization of diffusion term	45
5.1.2 Test for discretization of advection term	46
5.2 Time integration tests	48
5.2.1 Problem 1: Burgers equation	48

5.2.2 Problem 2: Molenkamp equation	52
6. <i>Discussion and future work</i>	56
6.1 Discussion	56
6.2 Future work	57
<i>Appendix</i>	61
<i>A. Appendix A</i>	62
<i>B. Appendix B</i>	63
<i>C. Appendix C</i>	64
<i>D. Appendix D</i>	65
<i>E. Appendix E</i>	67

1. INTRODUCTION

Advection-diffusion-reaction equations arise from many fields of science and engineering such as in environmental modeling, in mathematical biology, and in chemistry.

In environmental study, mathematical equations of advection-diffusion-reaction type originate from many natural phenomenon, such as models of weather forecast, of atmospheric air, and of water flow. For instance, studying model of groundwater (water below the ground surface) we determine amount of water in the soil as well as its change in space and in time. Groundwater flow is described by a mathematical model based on the law of conservation of mass and energy and the Darcy's law. The result in Figure 1.1 is obtained from simulation the groundwater in Luxemburg area ([17]). Another example of water-flow simulation is given in [4] (see the figure on the cover of this thesis).

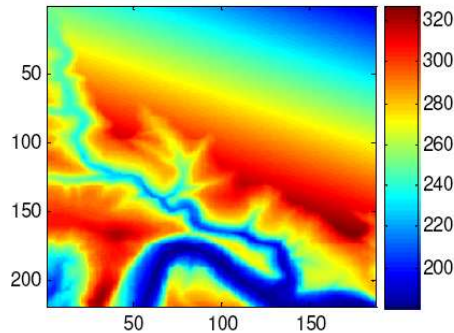


Fig. 1.1: Groundwater in Luxemburg area. The units in two axes are in kilometer. The height of groundwater (the color-bar) is measured in meter. Groundwater depends on the rain input and the discharged water. Rainfall and clay surface are real measured.

In mathematical biology, advection-diffusion-reaction equations are used to model processes such as the bacterial growth, tumor growth and related biochemical phenomena. The growth of a bacteria colony follows the policy: bacteria eat nutrient and grow; nutrient transports from high-concentration area to low-concentration area. Figure 1.2 is an example showing the growth of bacteria in a square nutrient dish ([9]). The model of tumor growth has been studied

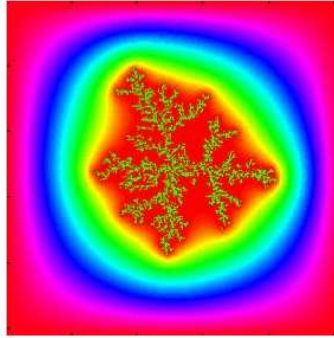


Fig. 1.2: Growth of bacteria colony in a square nutrient dish. Initially, a bacteria particle is set at center of the nutrient dish. Nutrient concentration decreases from edges to center. Colony of bacteria tends to the high nutrient concentration area. At boundaries, nutrient concentration is supposed to be constant.

in [19]. The simulation is based on a model coupling the invasion of the glioblastoma and its mechanical interaction with the invaded structures. This model uses a reaction-diffusion equation for the tumor expansion characterization and the usual continuum mechanics laws for the brain parenchyma behavior. Figure 1.3 shows the simulation result.

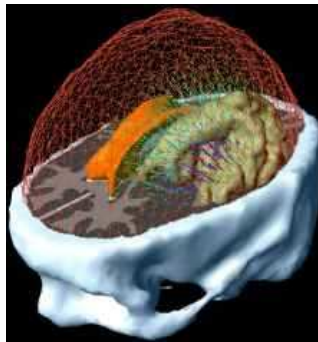


Fig. 1.3: Growth of tumors obtained by simulation.

In this thesis, we study an application in chemistry. That is the evolution in time of reacting chemical species in a fluid, which is also presented in term of advection-diffusion-reaction equations.

A system of advection-diffusion-reaction equations describes the evolution in time of chemical species that react with each other and that are transported in space through an advective flow field and through diffusion.

The evolution in time of reacting chemical species in a fluid is mathematically formulated as

$$c_t + \nabla \cdot (\mathbf{u}c) = \nabla \cdot (K \nabla c) + R(c), \quad (1.1)$$

where c is a vector of chemical species of length N , \mathbf{u} denotes the velocity field of the fluid and K is a diagonal matrix of diffusion coefficients. $\nabla = (\partial/\partial x_1, \partial/\partial x_2, \dots, \partial/\partial x_m)^T$ is the so-called gradient operator, m denotes the domain dimension and x_i ($i = 1, 2, \dots, m$) is the i th coordinate variable. $\nabla \cdot (\mathbf{u}c)$ is the divergence of $\mathbf{u}c$, i.e. the dot-product of the vector operator ∇ and the vector field $\mathbf{u}c$, definition for each component of c

$$\nabla \cdot (\mathbf{u}c) = \sum_{i=1}^m \frac{\partial (\mathbf{u}c)_i}{\partial x_i},$$

where $(\mathbf{u}c)_i$ is the i th component of $\mathbf{u}c$. Here, the advection part $\nabla \cdot (\mathbf{u}c)$ and the diffusion part $\nabla \cdot (K \nabla c)$ describe the transport of the species, while the reaction part $R(c)$ is a function that describes chemical reactions.

In equation (1.1), the velocity field \mathbf{u} of the fluid is computed via the incompressible Navier-Stokes equation

$$\begin{aligned} (\rho \mathbf{u})_t + \nabla \cdot (\rho \mathbf{u} \mathbf{u}^T) &= \mathcal{F}, \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \quad (1.2)$$

where ρ is the density of the fluid and \mathcal{F} consists of the acting forces on the fluid.

In general, it is very difficult to find exact solution of PDEs. Thus, the popular method is to use numerical techniques. This thesis aims to develop a numerical method for solving advection-diffusion-reaction equations. These equations arise from modeling and simulating phenomenon in biology, environment, and chemistry, etc. Although the method is presented specifically for problems (1.1) and (1.2) in a three-dimensional rectangular domain, it is also applicable for other problems with complex domains.

There exist nowadays many numerical techniques to solve advection-diffusion-reaction equations, for instance, the *Direct Space-Time* (DST) discretization and the *Method of Line* (MOL) approach ([3, 7]). With the MOL approach, spatial discretization and time integration are treated separately, whereas in the DST discretization, space and time are discretized simultaneously. The advantage of the MOL method is that it is simple and flexible: for the spatial discretization it is easy to combine various schemes for advection, diffusion and reaction terms, while in time integration we are free to choose a suitable method. With its popularity, we choose the MOL approach as our numerical technique to solve the problems (1.1) and (1.2). In this approach, we firstly need to discretize all the spatial operators (advection, diffusion and reaction terms). This results in a system of ordinary differential equations (*ODEs*) which is still continuous in time. Then, this ODE system will be integrated in time.

There are many ways to discretize the PDEs in space, for example, the finite element method (e.g. [1]), the finite difference method (e.g. [12]), and the finite volume method. The advantage of the finite volume method over the finite difference method is that it does not require a structured mesh (although a structured mesh can also be used). The finite volume method combines the advantages of the finite element method for geometric flexibility and the finite difference method for simple discrete computation. Moreover, in the finite volume method the conservation of flux quantities is guaranteed. Therefore, we choose the *finite volume* method for the spatial discretization.

Based on the finite volume approach, the diffusion terms are approximated by the most popular scheme, that is, the second-order finite difference. For the advection terms, many discretization schemes have been proposed: the first-order upwind discretization, the second-order central difference, the third-order upwind-biased scheme, and even higher order schemes. Although the first-order upwind scheme guarantees the smoothness of (i.e. no oscillations superimposed on) the solution, being first-order, the scheme has a too low accuracy in actual applications. The second-order central scheme is too dispersive (compared with other methods), which might result in unwanted oscillatory and even negative solutions. High-order schemes require too many grid points in order to approximate the derivatives of the solution. This results in a complicated and demanding computation. Therefore, as a compromise, the third-order upwind-biased scheme is used, in order to have high accuracy, less oscillations, and an acceptable computational cost.

After discretizing the PDEs (1.1) and (1.2) in space, we obtain a system of ODEs

$$\frac{d\omega(t)}{dt} = F(t, \omega(t)), \quad t > 0, \quad \omega(0) = \omega_0, \quad (1.3)$$

where $\omega(t)$ is the spatially-discretized solution and $F(t, \omega(t))$ is the corresponding right-hand side. The vector $\omega(t)$ is of length $n = (N + 3) \times \#\text{cells}$. Usually, n is very large which results in a huge-dimensional ODE system. Often, this system is *nonlinear* (i.e, F is a nonlinear function of ω) and *stiff* (i.e, the Jacobian matrix $\partial F/\partial\omega$ has a wide range of eigenvalues).

Due to the stiffness of the ODE system, an implicit time integrator might be necessary (otherwise, one can encounter instability) (see e.g. [2, 3] for more details in numerical treatments of stiff systems). Accordingly, an implicit relation has to be solved. This is a drawback when applied to such a huge-dimensional and nonlinear system, because an expensive computation is required. Explicit time integrators are cheap and simple since only straightforward computations are required. In addition, an explicit time integrator will simplify the parallelization of the code significantly. However, the size of time step using an explicit method may be too restrictive to guarantee that all the eigenvalues of the stiff system lie inside the (usually small) stability region of the method. Therefore, explicit time integrators are suitable only for non-stiff to moderately stiff terms. Combining the advantages of these two integrators into a

so-called *implicit-explicit* (IMEX) method is more promising. In our problem, the stiff reaction terms are treated implicitly, whereas the non-stiff advection and moderately stiff diffusion terms are treated explicitly.

Treating the advection terms explicitly offers a further refinement in their spatial discretization, that is, the combination with the *limiting technique*. The motivation is that, in the smooth regions of the solution, it retains high accuracy of high-order methods, while near extrema where high-order discretizations give rise to oscillations, the limiter switches to the first-order upwind scheme to prevent solution from oscillations (see e.g. [6, 5, 3]). The spatial discretization is then monotonous (i.e., no oscillations superimposed on the smooth solution). In spite of the nonlinearity that the limiter brings (see Chapter 3), this technique does not complicate the overall process significantly, because the time integration is computed explicitly.

In contrast to the advection operators whose eigenvalues are complex, the eigenvalues of the discretized diffusion operators are negative and real (because its Jacobian matrix is symmetric). In our application, this also holds for the eigenvalues obtained from the reaction terms (chemical part). This characteristic of the diffusion and reaction operators motivates the use of *Runge-Kutta-Chebyshev* (RKC) method. Originally, RKC automatically extends the stability region along the negative real axis by adding more stages to the method. With a variable timestep strategy the method can automatically find an optimal balance between accuracy and stability for problems with diffusion and mildly stiff reactions. Later on, the method has been extended to include also advection operators by enlarging the stability region along the imaginary axis and finding the optimal number of stages balancing the influence of the advection and the diffusion/reaction operators.

As stated above, the stability region of the explicit RKC method increases with the stage number s . For problems with very stiff reaction terms, the stability region of the method has to be very large. Consequently, we need a large number of stages, which results in an expensive computation. Therefore, it is more suitable to treat stiff reaction terms implicitly. The non-stiff advection and moderately-stiff diffusion terms are treated explicitly. This implicit-explicit combination leads to the IMEX-RKC approach. This approach can be seen as “operator splitting within the method”, which in contrast to the standard operator splitting techniques ([3]), does not give rise to a splitting error in time ([16]). Moreover, it was shown in [16] that the IMEX-RKC method is unconditionally stable for the implicitly-treated operator and the stability of the method is determined by its explicitly-treated operator. In this thesis, we will consider both the explicit RKC method, applied to advection-diffusion equations, and the IMEX-RKC method, applied to problems with stiff reaction terms.

In summary, our numerical technique is shown in Figure 1.4.

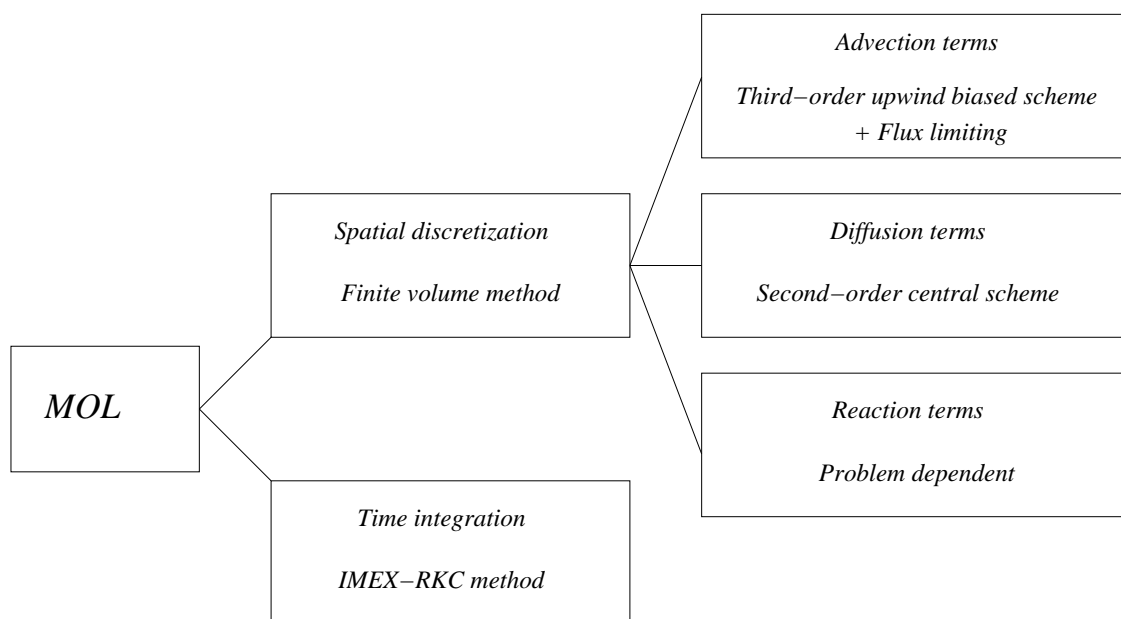


Fig. 1.4: Numerical technique for solving advection-diffusion-reaction equations: IMEX-RKC method for time integration; in spatial discretization, the second-order central scheme for diffusion terms and the third-order upwind-biased scheme combined with limiting technique for advection terms.

The structure of the thesis is organized as follows: first, we describe in Chapter 2 the mathematical problem, i.e. equations (1.1) and the Navier-Stokes equations (1.2). Then, the detailed numerical method used to solve these problems is presented in Chapter 3. In Chapter 4, the implementation of numerical method is introduced. The numerical experiments are given in Chapter 5. Finally, Chapter 6 is reserved for conclusion and discussion.

2. THE MATHEMATICAL MODEL

2.1 The model of reacting chemical species

The equation for mass conservation of a chemical species c is given by (see Appendix A)

$$c_t + \nabla \cdot (uc) = R(c), \quad (2.1)$$

where the subscript t denotes the differentiation in time, u is the velocity field, and R denotes source and sink terms.

To account for turbulent fluctuations in a flow, the Reynolds decomposition of the flow variable is used. That is, a flow variable q is decomposed into a mean and a fluctuating component

$$q = \bar{q} + \hat{q}, \quad (2.2)$$

where \bar{q} is the time-averaged mean of q , which is given by

$$\bar{q} = \frac{1}{\Delta t} \int_t^{t+\Delta t} q d\tau, \quad (2.3)$$

and \hat{q} is the fluctuation in time of the quantity, which is defined such that its time average equals to zero, i.e., $\overline{\hat{q}} = 0$.

Applying the Reynolds decomposition to the variables in Eq. (2.1) yields

$$(\bar{c} + \hat{c})_t + \nabla \cdot ((\bar{u} + \hat{u})(\bar{c} + \hat{c})) = R(\bar{c} + \hat{c}). \quad (2.4)$$

If we take the time average of this equation and neglect the high-order terms in source and sink terms we get

$$\bar{c}_t + \nabla \cdot (\overline{uc}) + \nabla \cdot (\overline{\hat{u}\hat{c}}) = R(\bar{c}). \quad (2.5)$$

In general, the divergence of the turbulent fluxes is parameterized in the form of gradient diffusion ([8])

$$\nabla \cdot (\overline{\hat{u}\hat{c}}) \approx -\nabla \cdot (\overline{K}\nabla\bar{c}), \quad (2.6)$$

where K is the eddy diffusivity tensor. For convenience the bars over the variables are omitted, Eq. (2.5) then reads

$$c_t + \nabla \cdot (uc) = \nabla \cdot (K\nabla c) + R(c), \quad (2.7)$$

yielding the model of reacting chemical species.

The possible boundary conditions for problem (2.7) are of the form

$$\alpha c + \beta n \cdot (K \nabla c) = \gamma. \quad (2.8)$$

2.2 The model of fluid flow

The velocity field \mathbf{u} appearing in Eq. (2.7) is computed from the incompressible *Navier-Stokes equations* which are based on the conservation law of momentum

$$\begin{aligned} (\rho \mathbf{u})_t + \nabla \cdot (\rho \mathbf{u} \mathbf{u}^T) &= \mathcal{F}, \\ \nabla \cdot \mathbf{u} &= 0. \end{aligned} \quad (2.9)$$

Here ρ and \mathcal{F} denote the density of the fluid and the acting forces on the fluid, respectively. By neglecting the Coriolis and centrifugal force, the acting forces include pressure gradient p , friction δ , and gravitation g , i.e.,

$$\mathcal{F} = -\nabla p + \nabla \cdot \delta - \rho g. \quad (2.10)$$

The viscous stress tensor for an incompressible Newtonian fluid is given by ([8])

$$\delta = \mu ((\nabla \mathbf{u}) + (\nabla \mathbf{u})^T), \quad (2.11)$$

where μ is the dynamic viscosity of the fluid.

Combining equations (2.9)–(2.11), the velocity field \mathbf{u} is computed from the following system

$$(\rho \mathbf{u})_t + \nabla \cdot (\rho \mathbf{u} \mathbf{u}^T) = -\nabla p + \nabla \cdot (\mu ((\nabla \mathbf{u}) + (\nabla \mathbf{u})^T)) - \rho g. \quad (2.12)$$

To simplify system (2.12), we denote the suitable reference length, reference velocity, and reference pressure by L_0 , U_0 , and p_0 , respectively, and introduce the dimensionless variables

$$\tilde{\mathbf{x}} = \frac{\mathbf{x}}{L_0}, \quad \tilde{t} = t \frac{U_0}{L_0}, \quad \tilde{\mathbf{u}} = \frac{\mathbf{u}}{U_0}, \quad \tilde{\rho} = 1, \quad \tilde{p} = \frac{p - p_0}{\rho U_0^2}, \quad \text{and} \quad \tilde{\nabla} = L_0 \nabla. \quad (2.13)$$

Substituting these notations into Eq. (2.12) and omit the tildes over the variables, system (2.12) then reads

$$\mathbf{u}_t + \nabla \cdot (\mathbf{u} \mathbf{u}^T) = -\nabla p + \frac{1}{Re} (\nabla \cdot ((\nabla \mathbf{u}) + (\nabla \mathbf{u})^T)) + \frac{1}{Fr^2}. \quad (2.14)$$

Here, $Re = \frac{\rho U_0 L_0}{\mu}$ is called the *Reynolds* number which denotes the ratio of inertial and viscous forces in the flow ([21]), and $Fr = \frac{U_0}{\sqrt{gL_0}}$ is called the *Froude* number which denotes the ratio

of the reference velocity over the velocity of particular gravity wave ([20]).

Eq. (2.14) is written for an arbitrary-dimensional domain, we will consider it in the three-dimensional Cartesian domain. Accordingly, we denote the coordinate variables by x , y and z , i.e. $\mathbf{x} = (x, y, z)$. Let u , v and w be respectively the x -, y - and z -component of the velocity field, i.e. $\mathbf{u} = (u, v, w)^T$. Then, system (2.14) is rewritten as follows (Appendix B)

$$\begin{aligned} u_t + (u^2)_x + (uv)_y + (uw)_z &= -p_x + \frac{1}{Re} (2u_{xx} + (u_y + v_x)_y + (u_z + w_x)_z), \\ v_t + (vu)_x + (v^2)_y + (vw)_z &= -p_y + \frac{1}{Re} ((v_x + u_y)_x + 2v_{yy} + (v_z + w_y)_z), \\ w_t + (wu)_x + (wv)_y + (w^2)_z &= -p_z + \frac{1}{Re} ((w_x + u_z)_x + (w_y + v_z)_y + 2w_{zz}) + \frac{1}{Fr^2}. \end{aligned} \quad (2.15)$$

Here, the subscripts denote the differentiation with respect to the corresponding variables. We note that the constant $\frac{1}{Fr^2}$ contributes only to the last equation, because gravitation g is non-zero in the vertical direction only. System (2.15) is said to be in *conservation form*.

Applying the product rule for differentiation states in Eq. (2.15) and using the divergence-free property of the velocity field $\nabla \cdot \mathbf{u} = 0$, we obtain the corresponding *advection form* (see appendix C)

$$\begin{aligned} u_t + uu_x + vu_y + wu_z &= -p_x + \frac{1}{Re} (u_{xx} + u_{yy} + u_{zz}), \\ v_t + uv_x + vv_y + wv_z &= -p_y + \frac{1}{Re} (v_{xx} + v_{yy} + v_{zz}), \\ w_t + uw_x + vw_y + ww_z &= -p_z + \frac{1}{Re} (w_{xx} + w_{yy} + w_{zz}) + \frac{1}{Fr^2}. \end{aligned} \quad (2.16)$$

Possible boundary conditions for *Navier-Stokes* equation include

- No-slip: $\mathbf{u} = 0$ (at fluid-solid surfaces);
- In-/Outflow: $\mathbf{u} = \mathbf{u}_{\partial\Omega}$;
- Outflow/Pressure : $(-pI + \delta)n = -p_{\partial\Omega}n$;
- Slip/Symmetry: $n \cdot \mathbf{u} = 0, t \cdot (-pI + \delta)n = 0$;
- Normal flow/Pressure: $t \cdot \mathbf{u} = 0, n \cdot (-pI + \delta)n = -p_{\partial\Omega}$;
- Neutral: $(-pI + \delta)n = 0$.

Remark 2.1. For very smooth solutions, it does not matter much whether the conservative or advective form is used, but for non-smooth solutions it is essential to discretize the conservation form in order to get correct propagations of steep gradients ([3, 13]).

3. NUMERICAL METHOD

In order to find numerical solutions for equations (2.7) and (2.15) or (2.16) we use the so-called *Method of Line* (MOL) approach, in which spatial discretization and time integration are considered separately. That is, we firstly discretize the PDEs in space, which gives rise to a large time-dependent system of ODEs, then integrate the ODE system in time to find the fully discrete numerical solution. This method is simple and flexible: for the spatial discretization, it is easy to combine various schemes for advection and diffusion with reaction terms, while for the time integration, we are free to choose a suitable integrator.

3.1 Spatial discretization

There are many ways to discretize the PDEs in space, for example, the finite element method (e.g. [1]), the finite difference method (e.g. [12]), and the finite volume method. Here, we use the finite volume method. The advantage of the finite volume method over the finite difference method is that it does not require a structured mesh (although a structured mesh can also be used). The finite volume method combines the advantages of the finite element method for geometric flexibility and the finite difference method for simple discrete computation. The finite volume method splits up the domain into cells (volumes) and the values of variables are calculated averaged over the cell. In this way, we obtain conservation of the transported quantities since all contributions of the fluxes along the interior cell faces cancel.

For simplicity, we consider a three-dimensional domain $\Omega = [0, 1]^3$ and attach to it a uniform grid

$$\begin{aligned}x_i &= (i - \frac{1}{2})\Delta x, & i &= 1, 2, \dots, nx, \\y_j &= (j - \frac{1}{2})\Delta y, & j &= 1, 2, \dots, ny, \\z_k &= (k - \frac{1}{2})\Delta z, & k &= 1, 2, \dots, nz,\end{aligned}\tag{3.1}$$

where nx , ny and nz are respectively number of grid points in x -, y - and z -directions; $\Delta x = 1/nx$, $\Delta y = 1/ny$ and $\Delta z = 1/nz$ denote the corresponding mesh sizes. The spatial domain is partitioned into cells whose centers coincide with the grid points. This discretization scheme is known as the *cell centered scheme* (see [3]). Using this grid, we need to approximate all the spatial operators in the system (2.15) or (2.16) as well as in the system (2.7). The actual spatial discretization of these systems are presented in the following sub-sections 3.1.1 and 3.1.2.

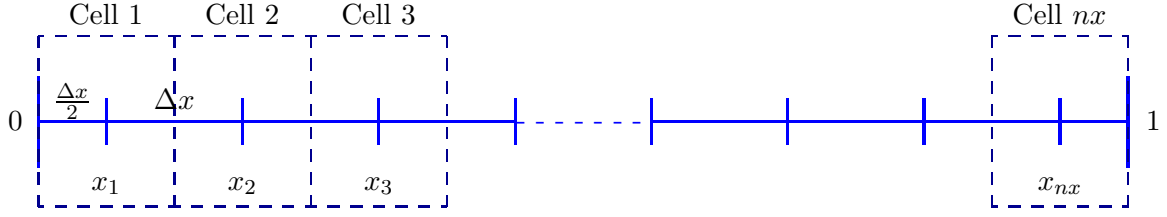


Fig. 3.1: Structure of cells in 1D domain

3.1.1 The Navier-Stokes equation

a) Conservation form

For simplicity in the implementation, we only present here the discretization of the first equation in (2.15)

$$u_t + (u^2)_x + (uv)_y + (uw)_z = -p_x + \frac{1}{Re} (2u_{xx} + (u_y + v_x)_y + (u_z + w_x)_z), \quad (3.2)$$

The other equations can be discretized in a similar way.

In the flux form, equation (3.2) is written as

$$u_t + \left(u^2 - \frac{2}{Re}u_x\right)_x + \left(uv - \frac{1}{Re}(u_y + v_x)\right)_y + \left(uw - \frac{1}{Re}(u_z + w_x)\right)_z = -p_x, \quad (3.3)$$

Let

$$f_1 = u^2 - \frac{2}{Re}u_x, \quad f_2 = uv - \frac{1}{Re}(u_y + v_x), \quad f_3 = uw - \frac{1}{Re}(u_z + w_x)$$

be the fluxes in x -, y - and z -directions, respectively, the equation (3.3) yields

$$u_t + \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y} + \frac{\partial f_3}{\partial z} = -p_x, \quad (3.4)$$

At each grid point (x_i, y_j, z_k) , the term p_x is directly calculated, whereas the derivatives of the fluxes, i.e. $\frac{\partial f_1}{\partial x}$, $\frac{\partial f_2}{\partial y}$ and $\frac{\partial f_3}{\partial z}$, are approximated by the difference of the fluxes at the cell boundaries.

In particular, in the x -direction, $\frac{\partial f_1}{\partial x}$ at (x_i, y_j, z_k) is approximated by

$$\frac{\partial f_1}{\partial x}(x_i, y_j, z_k) \approx \frac{f_{1_{i+1/2,j,k}} - f_{1_{i-1/2,j,k}}}{\Delta x}, \quad (3.5)$$

where $f_{1_{i\pm 1/2,j,k}}$ approximates the flux f_1 at the cell boundaries $(x_i \pm \frac{1}{2}\Delta x, y_j, z_k)$. That is,

$$f_{1_{i\pm 1/2,j,k}} = u_{i\pm 1/2,j,k}^2 - \frac{2}{Re} u_{x_{i\pm 1/2,j,k}}, \quad (3.6)$$

where $u_{i\pm 1/2,j,k}^2$ and $u_{x_{i\pm 1/2,j,k}}$ respectively denote the approximation of u^2 and u_x at the points $(x_i \pm \frac{1}{2}\Delta x, y_j, z_k)$ and need to be calculated. Detailed technique to calculate these terms is individually presented in the two paragraphs below.

The diffusion term

The term $u_{x_{i+1/2,j,k}}$ is approximated by using neighboring averaged cell concentrations, i.e.

$$u_{x_{i+1/2,j,k}} = \frac{u_{i+1,j,k} - u_{i,j,k}}{\Delta x}, \quad (3.7)$$

where $u_{i,j,k}$ denotes the approximation of u at (x_i, y_j, z_k) . As a result, we obtain (see (3.5), (3.6))

$$u_{xx}(x_i, y_j, z_k) \approx \frac{u_{x_{i+1/2,j,k}} - u_{x_{i-1/2,j,k}}}{\Delta x} = \frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{(\Delta x)^2}, \quad (3.8)$$

which is known as the *second-order central discretization scheme*. The second-order accuracy of the scheme is easily verified by inserting exact solution values into the scheme, and using Taylor expansion around (x_i, y_j, z_k) . Suppose that the solution is sufficiently differentiable, we have

$$u(x_{i\pm 1}) = u(x_i) \pm \Delta x u_x(x_i) + \frac{1}{2}(\Delta x)^2 u_{xx}(x_i) \pm \frac{1}{6}(\Delta x)^3 u_{xxx}(x_i) + \frac{1}{24}(\Delta x)^4 u_{xxxx}(x_i) + \mathcal{O}((\Delta x)^5) \quad (3.9)$$

where y_j and z_k have been removed for the simplicity of the expressions. Thus,

$$\frac{1}{(\Delta x)^2} (u(x_{i+1}) - 2u(x_i) + u(x_{i-1})) = u_{xx}(x_i) + \mathcal{O}((\Delta x)^2). \quad (3.10)$$

Comparing the relations (3.8) and (3.10), we obtain the truncation error $\mathcal{O}((\Delta x)^2)$, showing second-order accuracy of the scheme. In spite of its simplicity, the discretization scheme (3.7, 3.8) is popular and successfully used in practice.

The advection term

Let $\nu = u^2$. Consider the approximation $\nu_{i+1/2,j,k}$ by

$$\nu_{i+1/2,j,k} = \frac{\nu_{i,j,k} + \nu_{i+1,j,k}}{2}, \quad (3.11)$$

giving (see (3.5), (3.6))

$$\nu_x(x_i, y_j, z_k) \approx \frac{\nu_{i+1/2,j,k} - \nu_{i-1/2,j,k}}{\Delta x} = \frac{1}{2\Delta x} (\nu_{i+1,j,k} - \nu_{i-1,j,k}), \quad (3.12)$$

which is known as the *second-order central discretization*. Similar to (3.9), the Taylor expansion of ν around (x_i, y_j, z_k) gives ¹

$$\frac{1}{2\Delta x} (\nu(x_{i+1}) - \nu(x_{i-1})) = \nu_x(x_i) + \frac{1}{6}(\Delta x)^2 \nu_{xxx}(x_i) + \mathcal{O}((\Delta x)^4), \quad (3.13)$$

indeed, showing second-order accuracy of the scheme (compare (3.12) and (3.13)).

Although the second-order central discretization is often successfully used for the diffusion terms, this type of approximation, however, is a drawback when applied to the advection terms. This is because the second-order central scheme is too dispersive (due to phase error, see [3, p. 58]) compared with other methods. This may give rise to oscillations and negative solution components as shown in Fig. 3.2b. There, the maxima is faraway from $x = 0.5$. The solution oscillates largely (about 40%) and contains very negative components. The negative solution is of course unrealistic for our physical problem.

To overcome this fact, one can use the *first-order upwind scheme*. The word ‘upwind’ implies that the approximation of ν at the mid-point $(x_i + \frac{1}{2}\Delta x, y_j, z_k)$ is calculated using upstream information. The constant in front of ν in the expression of f_1 is +1, the upstream points lie at the left of the point $(x_i + \frac{1}{2}\Delta x, y_j, z_k)$. Thus, we take

$$\nu_{i+1/2,j,k} = \nu_{i,j,k} \quad (3.14)$$

leading to

$$\nu_x(x_i, y_j, z_k) \approx \frac{1}{\Delta x} (\nu_{i,j,k} - \nu_{i-1,j,k}). \quad (3.15)$$

Again, similar to (3.9), the Taylor expansion of ν around (x_i, y_j, z_k) yields

$$\frac{1}{\Delta x} (\nu(x_i) - \nu(x_{i-1})) = \nu_x(x_i) - \frac{1}{2}\Delta x \nu_{xx}(x_i) + \mathcal{O}((\Delta x)^2),$$

confirming the first-order accuracy of the method.

The advantage of the first-order upwind approximation is the monotonicity which guarantees the absence of oscillations. As we can see from Fig. 3.2a, the solution is very smooth, no oscillation occurs. However, comparing the maxima of the solution with the exact one, it is easily seen that the first-order upwind method is too inaccurate. This is due to the first-order

¹ Again, y_j and z_k have been removed for the simplicity of the expressions.

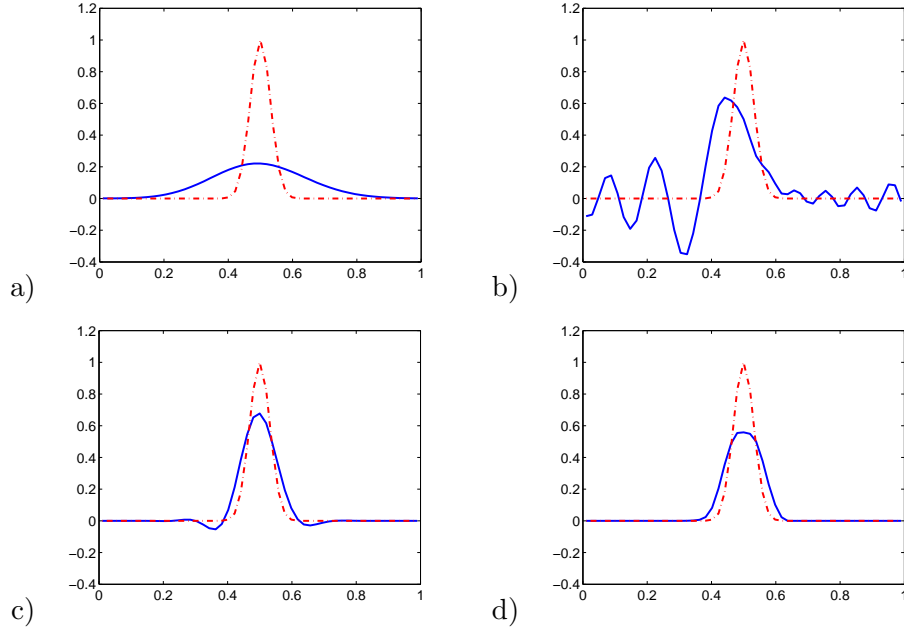


Fig. 3.2: One-dimensional advection test: $u_t + u_x = 0$, $u(x, 0) = (\sin(\pi x))^{100}$, $u(0, t) = u(1, t)$, $h = 1/50$, $t = 1$. The first-order upwind scheme (upper-left), the second-order central scheme (upper-right), and the third-order upwind-biased scheme without limiter (lower-left) and with limiter (lower-right). Dash-dotted lines: exact solution. Solid lines: numerical solutions (These pictures are taken from [3, pages 54, 60 and 218]).

accuracy of the method and the large amount of artificial diffusion of the scheme (see [3, p. 55]).

In order to get a better accuracy, higher order discretization schemes should be used. However, the higher order the scheme is, the more complicated implementation is needed, because a large stencil (number of neighbor grid points) has to be taken into account. Therefore, as a compromise, the *third-order upwind-biased scheme* is used. That is,

$$\nu_{i+1/2,j,k} = \frac{1}{6} (-\nu_{i-1,j,k} + 5\nu_{i,j,k} + 2\nu_{i+1,j,k}), \quad (3.16)$$

giving (see (3.5), (3.6))

$$\nu_{x_{i,j,k}} = \frac{1}{6\Delta x} (\nu_{i-2,j,k} - 6\nu_{i-1,j,k} + 3\nu_{i,j,k} + 2\nu_{i+1,j,k}). \quad (3.17)$$

By inserting the exact solution values into the scheme and using following Taylor expansions

$$\begin{aligned}\nu(x_{i\pm 1}) &= \nu(x_i) \pm \Delta x \nu_x(x_i) + \frac{1}{2}(\Delta x)^2 \nu_{xx}(x_i) \pm \frac{1}{6}(\Delta x)^3 \nu_{xxx}(x_i) \\ &\quad + \frac{1}{24}(\Delta x)^4 \nu_{xxxx}(x_i) + \mathcal{O}((\Delta x)^5), \\ \nu(x_{i-2}) &= \nu(x_i) - 2\Delta x \nu_x(x_i) + \frac{4}{2}(\Delta x)^2 \nu_{xx}(x_i) - \frac{8}{6}(\Delta x)^3 \nu_{xxx}(x_i) \\ &\quad + \frac{16}{24}(\Delta x)^4 \nu_{xxxx}(x_i) + \mathcal{O}((\Delta x)^5),\end{aligned}\tag{3.18}$$

we obtain

$$\frac{1}{6\Delta x}(\nu(x_{i-2}) - 6\nu(x_{i-1}) + 3\nu(x_i) + 2\nu(x_{i+1})) = \nu_x(x_i) + \frac{1}{12}(\Delta x)^3 \nu_{xxxx}(x_i) + \mathcal{O}((\Delta x)^5),\tag{3.19}$$

which verifies the third-order accuracy of the method (compare (3.17) and (3.19)).

Comparing the expressions (3.16) and (3.14), we see that the third-order flux can be written as the first-order flux plus a correction, that is,

$$\nu_{i+1/2,j,k} = \frac{1}{6}(-\nu_{i-1,j,k} + 5\nu_{i,j,k} + 2\nu_{i+1,j,k}) = \nu_{i,j,k} + \left(\frac{1}{3} + \frac{1}{6}\theta_i\right)(\nu_{i+1,j,k} - \nu_{i,j,k}),\tag{3.20}$$

where

$$\theta_i = \frac{\nu_{i,j,k} - \nu_{i-1,j,k}}{\nu_{i+1,j,k} - \nu_{i,j,k}}.\tag{3.21}$$

With the correction $(\frac{1}{3} + \frac{1}{6}\theta_i)(\nu_{i+1,j,k} - \nu_{i,j,k})$, the accuracy of the first-order flux has been raised (being third-order). In spite of its high accuracy, the third-order upwind-biased scheme however does not guarantee the absence of a small oscillations. From Fig. 3.2c we see that the third-order upwind-biased scheme indeed gives some oscillations, but rather small, compared with the oscillations of the second-order central scheme. Moreover, the solution is good in phase with the exact solution. The maximal value is raised up about three times compared with the one obtained from the first-order upwind scheme. The negative under-shoot implies that the correction is sometime too large and therefore its value needs to be limited. This leads to the so-called *limiting technique* (see [6, 5, 3]). That is, $\nu_{i+1/2,j,k}$ is approximated by the relation

$$\nu_{i+1/2,j,k} = \nu_{i,j,k} + \psi(\theta_i)(\nu_{i+1,j,k} - \nu_{i,j,k}),\tag{3.22}$$

where $\psi(\theta)$ is the *limiter* function. The purpose of the limiter function is that in the smooth regions of the solution ($\theta_i \approx 1$) the high accuracy and less dispersive properties of the third-order upwind-biased scheme, i.e. $\psi(\theta) = \frac{1}{3} + \frac{1}{6}\theta$, are maintained, while near the extremum, where oscillations can appear, the limiter switches to the first-order upwind scheme, i.e. $\psi(\theta) = 0$, to maintain the monotonicity of the solution. Thus, the limiter function is chosen by Koren [5] as (see Figure 3.3)

$$\psi(\theta) = \max(0, \min(1, \frac{1}{3} + \frac{1}{6}\theta, \theta)).\tag{3.23}$$

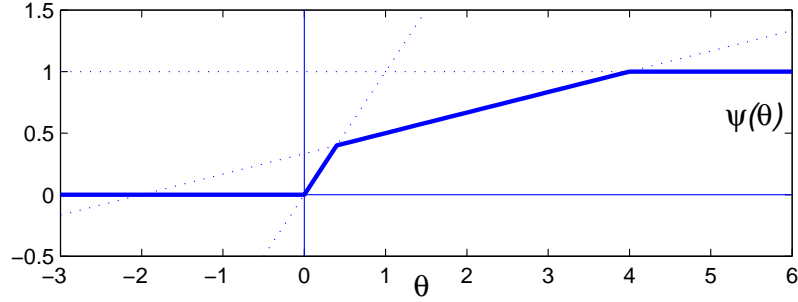


Fig. 3.3: The limiter function. For $\theta \leq 0$, $\psi(\theta) = 0$ and the limiter switches to the first-order upwind method. With $\frac{2}{5} \leq \theta \leq 4$ the third-order is retained. For other values of θ , the conditions $0 \leq \psi(\theta) \leq \theta$ and $0 \leq \frac{\psi(\theta)}{\theta} \leq 1$ are imposed to guarantee positivity of the scheme ([3, p. 216]).

From Fig. 3.2d we see that, using the limiting technique, the solution is neither oscillatory nor negative. With respect to accuracy, the scheme is more accurate than the first-order upwind scheme (comparing Figs. 3.2a and 3.2d), but less accurate than the third-order upwind-biased scheme (comparing Figs. 3.2c and 3.2d). This is because the limiter sometime switches to the first-order upwind scheme. However, in overall, the method is still of satisfactory because the diffusion term is discretized by the second-order method.

We have finished discretizing the derivative of the flux in the x -direction. For other directions, a similar technique can be used. For example, in the y direction, $\partial f_2 / \partial y$ at (x_i, y_j, z_k) is approximated by

$$\frac{\partial f_2}{\partial y}(x_i, y_j, z_k) \approx \frac{f_{2,i,j+1/2,k} - f_{2,i,j-1/2,k}}{\Delta y}, \quad (3.24)$$

where

$$f_{2,i,j+1/2,k} = (uv)_{i,j+1/2,k} - \frac{1}{Re}(u_{y_{i,j+1/2,k}} + v_{x_{i,j+1/2,k}}). \quad (3.25)$$

The approximations of $(uv)_{i,j+1/2,k}$ and $u_{y_{i,j+1/2,k}}$ are obtained in a similar way in (3.22) and (3.7) by setting $\nu = uv$ and interchanging the roles of x and y ; and i and j . For the term $v_{x_{i,j+1/2,k}}$, we use the following second-order approximation scheme

$$v_{x_{i,j+1/2,k}} = \frac{(v_{i+1,j+1,k} + v_{i+1,j,k}) - (v_{i-1,j+1,k} + v_{i-1,j,k})}{4\Delta x}, \quad (3.26)$$

which induces (see (3.24) and (3.25))

$$v_{xy_{i,j,k}} = \frac{(v_{i+1,j+1,k} - v_{i-1,j+1,k}) - (v_{i+1,j-1,k} - v_{i-1,j-1,k})}{4\Delta x \Delta y}. \quad (3.27)$$

Using Taylor expansions, we have ²

$$\begin{aligned}
v(x_{i+1}, y_{j+1}) - v(x_{i-1}, y_{j+1}) &= 2\Delta x v_x(x_i, y_{j+1}) + \frac{2}{6}(\Delta x)^3 v_{xxx}(x_i, y_{j+1}) + \mathcal{O}((\Delta x)^5), \\
v(x_{i+1}, y_{j-1}) - v(x_{i-1}, y_{j-1}) &= 2\Delta x v_x(x_i, y_{j-1}) + \frac{2}{6}(\Delta x)^3 v_{xxx}(x_i, y_{j-1}) + \mathcal{O}((\Delta x)^5), \\
v_x(x_i, y_{j+1}) - v_x(x_i, y_{j-1}) &= 2\Delta y v_{xy}(x_i, y_j) + \frac{2}{6}(\Delta y)^3 v_{xyyy}(x_i, y_j) + \mathcal{O}((\Delta y)^5), \\
v_{xxx}(x_i, y_{j+1}) - v_{xxx}(x_i, y_{j-1}) &= 2\Delta y v_{xxxxy}(x_i, y_j) + \frac{2}{6}(\Delta y)^3 v_{xxxxyy}(x_i, y_j) + \mathcal{O}((\Delta y)^5).
\end{aligned} \tag{3.28}$$

Therefore, we have

$$\begin{aligned}
&\frac{1}{4\Delta x \Delta y} [v(x_{i+1}, y_{j+1}) - v(x_{i-1}, y_{j+1})] - [v(x_{i+1}, y_{j-1}) - v(x_{i-1}, y_{j-1})] = \\
&v_{xy}(x_i, y_j) + \frac{1}{6}(\Delta y)^2 v_{xyyy}(x_i, y_j) + \frac{1}{6}(\Delta x)^2 v_{xxxxy}(x_i, y_j) + \mathcal{O}((\Delta x)^4, (\Delta y)^4, (\Delta x)^2 \cdot (\Delta y)^2),
\end{aligned} \tag{3.29}$$

which verifies the second-order accuracy of the scheme (3.26, 3.27).

b) Advection form

Rewrite the first equation in the system (2.16) as

$$u_t + f_1 + f_2 + f_3 = -p_x, \tag{3.30}$$

where

$$f_1 = uu_x - \frac{1}{Re}u_{xx}, \quad f_2 = vu_y - \frac{1}{Re}u_{yy}, \quad f_3 = wu_z - \frac{1}{Re}u_{zz}.$$

We present here the discretization of f_1 . The discretizations of f_2 and f_3 , can be done in similar manner.

At a particular grid point (x_i, y_j, z_k) , f_1 is approximated by

$$f_1(x_i, y_j, z_k) \approx u_{i,j,k}u_{x_{i,j,k}} - \frac{1}{Re}u_{xx_{i,j,k}}. \tag{3.31}$$

The advection term, $u_{x_{i,j,k}}$ is computed as

$$u_{x_{i,j,k}} = \frac{u_{i+1/2,j,k} - u_{i-1/2,j,k}}{\Delta x}$$

In (3.31), the factor $u_{i,j,k}$ represents for the advection coefficient which is constant over the cell (i,j,k) and its sign can vary from cell to cell. As a result, the upstream information may

² Here, z_k has been removed for simple expression.

change from cell to cell. Depending on the sign of $u_{i,j,k}$, the terms $u_{i\pm 1/2,j,k}$ can be evaluated as follows ([3])

$$u_{i-1/2,j,k} = \begin{cases} u_{i-1,j,k} + \psi(\theta_{i-1})(u_{i,j,k} - u_{i-1,j,k}) & \text{if } u_{i,j,k} \geq 0, \\ u_{i,j,k} + \psi(\frac{1}{\theta_i})(u_{i-1,j,k} - u_{i,j,k}) & \text{if } u_{i,j,k} < 0, \end{cases} \quad (3.32)$$

$$u_{i+1/2,j,k} = \begin{cases} u_{i,j,k} + \psi(\theta_i)(u_{i+1,j,k} - u_{i,j,k}) & \text{if } u_{i,j,k} \geq 0, \\ u_{i+1,j,k} + \psi(\frac{1}{\theta_{i+1}})(u_{i,j,k} - u_{i+1,j,k}) & \text{if } u_{i,j,k} < 0, \end{cases} \quad (3.33)$$

where

$$\theta_i = \frac{u_{i,j,k} - u_{i-1,j,k}}{u_{i+1,j,k} - u_{i,j,k}}.$$

The diffusion term, $u_{xx_{i,j,k}}$ is calculated by the second-order central scheme (3.8).

3.1.2 The reacting chemical species equation

Similar to (2.15), in the three-dimensional Cartesian domain, the system (2.7) can be rewritten as

$$c_t + (uc - K_1 c_x)_x + (vc - K_2 c_y)_y + (wc - K_3 c_z)_z = R(c), \quad (3.34)$$

for all components of the vector of chemical species. Here, K_1 , K_2 and K_3 are the diagonal elements of K ; u , v and w are the components of the velocity field that have been found in the preceding section. In the flux form, this system is as follows

$$c_t + \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y} + \frac{\partial f_3}{\partial z} = R(c), \quad (3.35)$$

where $f_1 = uc - K_1 c_x$, $f_2 = vc - K_2 c_y$, and $f_3 = wc - K_3 c_z$. The spatial operators in this equation are discretized by the same technique that has been proposed in Section 3.1.1, with a small remark regarding to the sign of the velocity field. For instance, in the x -direction ([3], p. 83)

$$c_{i+\frac{1}{2},j,k} = \begin{cases} c_{i,j,k} + \psi(\theta_{i,j,k})(c_{i+1,j,k} - c_{i,j,k}) & \text{if } u_{i+1/2,j,k} \geq 0, \\ c_{i+1,j,k} + \psi(\frac{1}{\theta_{i+1,j,k}})(c_{i,j,k} - c_{i+1,j,k}) & \text{if } u_{i+1/2,j,k} < 0, \end{cases} \quad (3.36)$$

where $\theta_{i,j,k} = \frac{c_{i,j,k} - c_{i-1,j,k}}{c_{i+1,j,k} - c_{i,j,k}}$.

3.1.3 The boundary conditions

a) The reacting chemical species equation

The boundary conditions for reacting chemical species equation Eq. (2.7) are given in Eq. (2.8). With $(\alpha \neq 0, \beta = 0)$, $(\alpha = 0, \beta \neq 0)$, and $(\alpha \neq 0, \beta \neq 0)$ we have Dirichlet, Neumann and Robin boundary conditions, respectively.

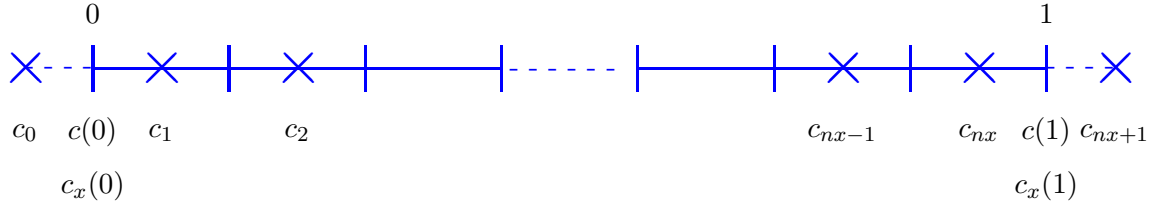


Fig. 3.4: Boundary conditions. With Dirichlet boundary condition, $c(0)$ and $c(1)$ are known; with Neumann boundary condition, $c_x(0)$ and $c_x(1)$ are given.

For the Dirichlet boundary condition, the concentration c at the boundaries are given and can be used in the discretization schemes as exact values. The solution at extra points can be approximated via the second-order central approximation scheme, that is,

$$\begin{aligned} c(0) &= \frac{1}{2}(c_0 + c_1), & \text{i.e. } c_0 &= 2c(0) - c_1, \\ c(1) &= \frac{1}{2}(c_{nx} + c_{nx+1}), & \text{i.e. } c_{nx+1} &= 2c(1) - c_{nx}, \end{aligned} \quad (3.37)$$

where $c(0)$ and $c(1)$ are given at $x = 0$ and $x = 1$; c_0 and c_{nx+1} are the approximated solution at the extra points $x = -\Delta x/2$ and $x = 1 + \Delta x/2$, respectively.

For the Neumann boundary condition, the first derivatives of the solution at the boundaries are known. The extra points can therefore be determined through the second-order finite difference approximation, e.g.

$$\begin{aligned} c_x(0) &= \frac{1}{\Delta x}(c_1 - c_0), & \text{i.e. } c_0 &= c_1 - \Delta x c_x(0), \\ c_x(1) &= \frac{1}{\Delta x}(c_{nx+1} - c_{nx}), & \text{i.e. } c_{nx+1} &= c_{nx} + \Delta x c_x(1), \end{aligned} \quad (3.38)$$

For the Robin boundary condition, we assume the fluxes at the boundaries are given and are used directly in the spatial discretization.

b) The Navier-stokes equation

Possible boundary conditions for *Navier-stokes* equation are given in section 2.2. The extrapolation is applied in the same manner as in reacting chemical species equation. For the first and the second types, the velocity at boundary is known, so that we can determine the extra points as in case of Dirichlet boundary condition. For the other types, the first derivative of the velocity at boundary is known, therefore the extra points can be calculated as in case of Neumann boundary condition.

3.2 Time integration

After discretizing in space, we get an ODE system which is still continuous in time and need to be integrated in time. That is,

$$\frac{dw(t)}{dt} = F(t, w(t)), \quad (3.39)$$

where vector $w(t)$ contains all the solution components (the velocity u , v and w in (2.15) and the concentration c in (3.34)) at time t and F is the corresponding right-hand side function coming from discretization of all the spatial operators. This ODE system is stiff (large spectrum of eigenvalues) coming mainly from the stiff reaction terms.

For the stiff terms, implicit time integration is of utmost importance due to stability requirement. However, implicit methods are expensive because we need to solve a nonlinear system. Explicit schemes are cheaper since the solution is found by explicit computation. In addition, an explicit method will simplify the parallelization of the code. However, explicit time integration methods were only suitable for non-stiff terms, because of the limited stability region. In this thesis we use the recently developed IMEX-RKC scheme ([15]) which is a combination between a stabilized explicit scheme used for non-stiff to moderately stiff part and implicit scheme used for stiff part. In this respect, we split F into two terms: F_E contains information from spatial discretization of the advection and diffusion terms, and F_I from spatial discretization of the reaction terms, then system (3.39) reads

$$\frac{dw(t)}{dt} = F_E(t, w(t)) + F_I(t, w(t)). \quad (3.40)$$

The principal goal when constructing explicit Runge-Kutta methods was to achieve the highest order possible with a given number of stages s . Stabilized methods like RKC are different in that only a few stages are used to achieve a usually low order whereas additional stages are exploited to increase the region of absolute stability, depending on the particular application ([15, 16]). Originally the RKC method was intended for semi-discrete parabolic PDE problems. Correspondingly, the original method is stable on a strip containing a long segment of the negative real axis. The wider the strip, the greater the applicability of the method, but the most important characteristic of the formula is the length of the segment, the real stability boundary, which increases quadratically with s .

3.2.1 Original explicit Runge Kutta method

The simplest way for integrating a system of ODEs is forward Euler method

$$w_{n+1} = w_n + \tau F(t_n, w_n), \quad (3.41)$$

where w_n denotes the numerical approximation to the exact solution $w(t)$ at $t = t_n$ and τ is the step size in the current step from t_n to t_{n+1} , i.e., $\tau = t_{n+1} - t_n$. One of the disadvantages of forward Euler integration is that it is too inaccurate. In this method, the next point (w_{n+1}) is calculated using only information from current point (w_n), and accuracy order is one.

In order to increase the accuracy order of the method, one idea is that we use some intermediate points inside the interval from t_n to t_{n+1} to calculate the next point. The Runge-Kutta method is based on this idea. The number of intermediate points is called the number of stages s .

In equation, an s -stage explicit Runge-Kutta method has following form

$$\begin{aligned} W_i &= w_n + \tau \sum_{j=1}^{s-1} a_{ij} F(t_n + c_j \tau, W_j), \quad i = 1, \dots, s, \quad j = 1, \dots, s, \\ w_{n+1} &= w_n + \tau \sum_{i=1}^s b_i F(t_n + c_i \tau, W_i). \end{aligned} \quad (3.42)$$

The parameters a_{ij} , b_i and c_i are normally given in Butcher array as

Tab. 3.1: Butcher array

c_i	a_{ij}
	b_j

If we assume that the explicit Runge-Kutta method is of order q , and if we apply (3.42) to the scalar test equation $w'(t) = \lambda w(t)$ we obtain $w_{n+1} = P(z)w_n$, where $z = \tau\lambda$ and $P(z)$ is stability function

$$P(z) = 1 + z + \frac{z^2}{2} + \dots + \frac{z^q}{q!} + \mathcal{O}(z^{q+1}), \quad (3.43)$$

The stability region of explicit Runge Kutta method given by $P(z) < 1$.

Although original explicit Runge-Kutta method has highest order of accuracy with a given number of stages s , the stability region of this method is very small. It was found that the stability region of Runge-Kutta integration can be extended along the negative real axis if we build the stability function $P(z)$ based on the Chebyshev polynomial. This combination returns a method which so-called Runge-Kutta Chebyshev (RKC).

3.2.2 The explicit Runge Kutta Chebyshev method

Discretizing PDEs such as (2.15) and (3.34) in 3D results in a huge system of ODE, the size of this system grows quickly proportional to the number of grid points. In order to get high accuracy in space, a large number of grid points is needed which requires a large computation time. Because of the restriction in computation time, the modest accuracy order is usually used in solving these PDEs, this leads to modest accuracy required in time integration. Therefore we choose the second-order RKC method for our time integrator.

The RKC scheme of order 2 was introduced in ([15]). Let w_n and τ_n be the numerical solution and step size at current time t_n , respectively. The second-order RKC method with s stages has the following form

$$\begin{aligned} W_0 &= w_n, \\ W_1 &= W_0 + \tilde{\mu}_1 \tau F_0, \\ W_j &= (1 - \mu_j - \nu_j)W_0 + \mu_j W_{j-1} + \nu_j W_{j-2} + \tilde{\mu}_j \tau F_{j-1} + \tilde{\gamma}_j \tau F_0, \\ w_{n+1} &= W_s, \end{aligned} \tag{3.44}$$

where $j = 2, \dots, s$ and F_k denotes $F(t_n + c_k \tau, W_k)$ and the coefficients are given by

$$\begin{aligned} \tilde{\mu}_1 &= b_1 \omega_1, \\ \mu_j &= \frac{2b_j \omega_0}{b_{j-1}}, \quad \nu_j = \frac{-b_j}{b_{j-2}}, \quad \tilde{\mu}_j = \frac{2b_j \omega_1}{b_{j-1}}, \quad \tilde{\gamma}_j = -a_{j-1} \tilde{\mu}_j, \quad j = 2, \dots, s. \end{aligned} \tag{3.45}$$

The parameters a_j, b_j, c_j and ω_0, ω_1 are calculated as follows.

Applying (3.44) to the scalar test equation $w'(t) = \lambda w(t)$, we get at each stage a relation $W_j = P_j(z)W_0$ with $z = \tau\lambda$ and $P_j(z)$ is a polynomial of degree j in z with $P_s(z)$ as stability function. Formula (3.44) is derived from a particular set of function $P_j(z)$ ($0 \leq j \leq s$) satisfying three design criteria:

- Nearly optimal step-by-step stability of $P_s(z)$ for parabolic problems;
- Internal stability, i.e., controlled round-off accumulation in a single step for s large;
- second-order consistency of $P_j(c_j z)$ with respect to $e^{c_j z}$ for $j = 2, \dots, s$.

The stability functions P_j are chosen as

$$P_j(z) = a_j + b_j T_j(\omega_0 + \omega_1 z), \tag{3.46}$$

where $T_j(x)$ is the first kind Chebyshev polynomials satisfying a three terms recursion.

$$T_j(x) = 2x T_{j-1}(x) - T_{j-2}(x), \quad j = 2, 3, \dots, s, \tag{3.47}$$

with $T_0(x) = 1$, $T_1(x) = x$.

The coefficients a_j and b_j are given by

$$a_j = 1 - b_j T_j(\omega_0), \quad (3.48)$$

where

$$b_0 = b_2, \quad b_1 = \frac{1}{\omega_0}, \quad b_j = \frac{T_j''(\omega_0)}{\left(T_j'(\omega_0)\right)^2}, \quad j = 2, \dots, s, \quad (3.49)$$

with

$$\omega_0 = 1 + \frac{\varepsilon}{s^2}, \quad \omega_1 = \frac{T_s'(\omega_0)}{T_s''(\omega_0)}. \quad (3.50)$$

Here $\varepsilon \geq 0$ is free and is called a damping parameter as $\varepsilon > 0$ gives values of the stability function $P_s(z)$ strictly less than one in the interior of the real stability interval. The stability boundary of RKC method, which is denoted by $\beta(s)$, increases quadratically with s as

$$\beta(s) \approx \frac{2}{3}(s^2 - 1)\left(1 - \frac{2}{15}\varepsilon\right). \quad (3.51)$$

Abscissa c_j are obtained by expanding $P_j(z)$ for $z \rightarrow 0$. We then have $c_j = b_j \omega_1 T_j'(\omega_0)$ and thus

$$c_0 = 0, \quad c_1 = c_2, \quad c_j = \frac{T_s'(\omega_0)}{T_s''(\omega_0)} \frac{T_j''(\omega_0)}{T_j'(\omega_0)}, \quad c_s = 1. \quad (3.52)$$

3.2.3 The implicit-explicit Runge Kutta Chebyshev method

For problems with stiff reaction terms, using explicit RKC scheme requires a large number of stages, which result in an expensive computation time. The use of RKC in an IMEX context is more efficient. The IMEX-RKC method (given in ([15]) can be obtained by modifying explicit RKC method. (for more details, see appendix D).

$$\begin{aligned} W_0 &= w_n, \\ W_1 &= W_0 + \tilde{\mu}_1 \tau F_{E,0} + \tilde{\mu}_1 \tau F_{I,1}, \\ W_j &= (1 - \mu_j - \nu_j)W_0 + \mu_j W_{j-1} + \nu_j W_{j-2} + \tilde{\mu}_j \tau F_{E,j-1} + \tilde{\gamma}_j \tau F_{E,0} \\ &\quad + [\tilde{\gamma}_j - (1 - \mu_j - \nu_j)\tilde{\mu}_1] \tau F_{I,0} - \nu_j \tilde{\mu}_1 \tau F_{I,j-2} + \tilde{\mu}_1 \tau F_{I,j}, \\ w_{n+1} &= W_s, \end{aligned} \quad (3.53)$$

where $j = 2, \dots, s$, $F_{E,k} = F_E(t_n + c_k \tau, W_k)$ and $F_{I,k} = F_I(t_n + c_k \tau, W_k)$.

All parameters are the same as in the explicit RKC method. The difference is that at each stage we have to solve an implicit system $W_j = W^* + \tilde{\mu}_1 \tau F_{I,j}$ where W^* is known. Because

the stiff operator F_I , which has real and negative eigenvalues, is treated implicitly, thus this part is unconditionally stable. The stability region of the method is determined only by the explicitly treated operator F_E .

To solve these implicit ODEs systems, we use modified Newton iteration and standard LU-decomposition.

3.2.4 Damping parameter selection and stability region

The stability region of RKC method depends on the damping parameter ε . With a small value ε , this region is a narrow strip along the negative real axis, which is suitable for the eigenvalues of the diffusion and reaction terms. When ε increases, the strip becomes wider and shorter. This region fits for imaginary part of eigenvalues coming from advection terms. It was found in ([16]) that with $\varepsilon = 10$, scheme (3.53) is suitable for both advection and diffusion terms. With $\varepsilon = 10$, the stability boundary of RKC method is calculated as (see [16])

$$\beta(s) = \begin{cases} 2, & s = 2, \\ (s^2 - 1) \left(0.340 + 0.189 \left(\frac{2}{s-1} \right)^{1.3} \right), & s \geq 3. \end{cases} \quad (3.54)$$

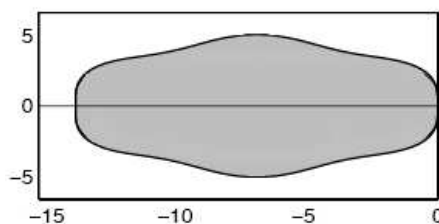


Fig. 3.5: Stability region of RKC method with $\varepsilon = 10$ and $s = 6$. (this picture is copied from [16])

3.3 Variable step size control and stage selection

Constant step size is often used to simplify analysis of integration methods. However, in modern practice, ODE problems are usually integrated with variable step sizes ([10, 3]). This approach leads to small (large) step sizes in regions of rapid (slow) variation of the solution, resulting in efficient computations in term of CPU versus accuracy. The time step has to obey stability requirement of the method and is chosen such that the estimated error is smaller than the tolerance given by user. For a given time step, the minimum number of stages

is determined such that the stability condition is maintained. The whole step-size control and stage-selection strategy based on error estimation and stability analysis which will be presented in the following sub-sections.

3.3.1 Local error estimation

The local error estimation is performed at every step try. At a step, the step size is chosen such that the estimated error is smaller than the tolerance given by user. Depending on the solution value, the tolerance is taken as

$$tol_{n+1} = atol + rtol |w_{n+1}|, \quad (3.55)$$

where $rtol$ and $atol$ are the given relative- and absolute-tolerance, respectively. For the IMEX-RKC method, the estimated error at time t_{n+1} is evaluated by (see Appendix E)

$$Est_{n+1} = \frac{1}{2}\tau_n \left(F(t_{n+1}, w_{n+1}) - F(t_n, w_n) \right) + \frac{3}{s^2 - 1} \tau_n \left(F_I(t_{n+1}, w_{n+1}) - F_I(t_n, w_n) \right), \quad (3.56)$$

where τ_n is the step size at time t_n .

We define the weighted norm of the error as

$$\| Est_{n+1} \| = \frac{\| Est_{n+1} \|_2}{\| tol_{n+1} \|_2}. \quad (3.57)$$

When the estimated error is smaller than the tolerance, i.e. $\| Est_{n+1} \| \leq 1$, the step is accepted. The next step size is predicted using information gathered from the current and preceding step. The current step is rejected if the estimated error is larger than the tolerance, i.e. $\| Est_{n+1} \| > 1$, and a smaller step size is retried. In order to reduce the number of rejected steps, in both cases, the new step size τ^* is taken to be a fraction of the largest possible value (see [11]). That is,

$$\tau^* = \min \left(10, \max(0.1, fac) \right) \tau_n, \quad (3.58)$$

where

$$fac = 0.8 \left(\frac{\| Est_{n-1} \|^{1/2} \tau_{n-1}}{\| Est_{n+1} \|^{1/2} \tau_n} \right) \frac{1}{\| Est_{n+1} \|^{1/2}}. \quad (3.59)$$

3.3.2 Critical step sizes for advection-diffusion equations

Although we use variable step sizes, its maximal value is always restricted by the stability condition of the integration method. That is, the step size is such that all eigenvalues obtained from von Neumann stability analysis lie inside the stability region of the time integration method ([3, 16]). For example, for the advection-diffusion problem, the step size has to satisfy both the following conditions (see [16]).

1. Stability condition for diffusion terms:

$$\tau \leq \frac{1}{2K \sum_{k=1}^3 h_k^{-2} (2 + (1 - \kappa)P_k)} \beta(s), \quad (3.60)$$

where K is diffusion coefficient; $h_k, k = 1, 2, 3$, are grid sizes in x -, y -, and z -direction, respectively; κ is a constant depending on the advection spatial discretization method (e.g. $\kappa = 1/3$ for the third-order upwind-biased scheme); u_k is the k th component of the velocity field; and $P_k = |u_k| h_k / K$ is a mesh Péclet number.

2. Stability condition for advection terms:

$$\tau \leq q_1 \left(\frac{4K\alpha^4(s)}{\beta(s)} \right)^{1/3} / \sum_{k=1}^3 \left(\frac{u_k^4}{h_k^2} \right)^{1/3}, \quad (3.61)$$

where the parameter q_1 depends on the choice of κ (e.g. $q_1 = 0.635$ if $\kappa = 1/3$). The ratio $\alpha^4(s)/\beta(s)$ is calculated as follows

$$\frac{\alpha^4(s)}{\beta(s)} = \begin{cases} 2, & s = 2, \\ 4(6 - s) + 6.15(s - 4), & s = 4, 6, \\ 6.15(10 - s)/2 + 15.5(s - 6)/4, & s = 8, 10, \\ 15.5, & s = 12, 14, \dots \end{cases} \quad (3.62)$$

Remark 3.2. For a nonlinear advection-diffusion equation with $u = u(c)$ and $K = K(c)$, the maximal value of u_k is inserted into both inequalities (3.60) and (3.61). For the diffusion coefficient K , the maximal and minimal value is inserted into (3.60) and (3.61), respectively.

3.3.3 Stage selection and step size adjustment

The stability region of RKC method increases quadratically with s , however a large number of stages s requires an expensive computation, especially in solving implicit relations as IMEX-RKC. Therefore, minimizing the s is necessary in this method.

The step size τ^* obtained from condition (3.58) has to satisfy both inequalities (3.60) and (3.61). Let s_d and s_a respectively be the number of stages for which condition (3.60) and condition (3.61) are satisfied for $\tau = \tau^*$. We have to find a minimal s_d and a minimal s_a such that $s_d \geq s_a$. This adjustment underlies the fact that the best strategy for advection is to minimize s and thus with respect to stability it makes no sense to spend more evaluations on advection than required by diffusion ([16]).

Let ψ_1 and ψ_2 contain the parameters of semi-discrete system

$$\psi_1 = \frac{1}{2K \sum_{k=1}^3 h_k^{-2} (2 + (1 - \kappa) P_k)}, \quad \psi_2 = \frac{4Kq_1^3}{\left(\sum_{k=1}^3 (u_k^4/h_k^2)\right)^{1/3}}. \quad (3.63)$$

The stage selection procedure (given in [16]) is as follows

1. If $\tau^* \leq 2\psi_1$ we put $s = 2$, $\tau = \min(\tau^*, (2\psi_2)^{1/3})$ and are done.
2. Put $\tau = \min(\tau^*, (15.5\psi_2)^{1/3})$. If $\tau \leq 2\psi_1$ we put $s = 2$ and are done.
3. Determine $s_d \geq 4$ such that $\tau \leq \beta(s_d)\psi_1$ to satisfy (3.60).
4. Determine $s_a \geq 4$ such that $\tau \leq \left(\frac{\alpha^4(s_a)}{\beta(s_a)}\psi_2\right)^{1/3}$ to satisfy (3.61).
5. If $s_a \leq s_d$ we put $s = s_d$ and are done. Otherwise let $\tau := 0.8\tau$ and repeat steps 3, 4, and 5.

Remark 3.3. For the pure advection problem, conditions (3.60) and (3.61) are replaced by following CFL condition (see [16])

$$\sum_{k=1}^3 \frac{\tau |u_k|}{h_k} \leq \nu(s), \quad (3.64)$$

where

$$\nu(s) = \begin{cases} 0.87 \frac{4-s}{2} + 1.40 \frac{s-2}{2}, & 2 \leq s \leq 4, \\ 1.40 \frac{9-s}{5} + 1.70 \frac{s-4}{5}, & 4 \leq s \leq 9, \\ 1.70, & s \geq 9. \end{cases} \quad (3.65)$$

The new step size is then adjusted by

$$\tau := \min \left(\tau, \nu(s) / \sum_{k=1}^3 \frac{|u_k|}{h_k} \right). \quad (3.66)$$

4. IMPLEMENTATION

In chapter 3 we have proposed the numerical methods for solving reacting chemical species model (2.7) as well as the model of fluid flow (2.15) and (2.16). This chapter presents the implementation of numerical method for solving advection-diffusion-reaction equation.

$$c_t + (uc - K_1 c_x)_x + (vc - K_2 c_y)_y + (wc - K_3 c_z)_z = R(c), \quad (4.1)$$

where K_1 and u , K_2 and v , K_3 and w are the diffusion coefficients and velocity fields in the x -, y -, and z - direction, respectively; c is the vector of chemical species and $R(c)$ is function that describes chemical reactions.

For other applications, small changes are needed in the spatial discretization, while the time integration procedure is still maintained.

The algorithm for solving equation (4.1) is shown in Figure 4.1.

4.1 Implementation of the spatial discretization

The main functions in spatial discretization procedure including the discretization of advection terms, diffusion terms and reaction terms; then F_E and F_I are calculated

In the following pseudo code, we denote the s -th species by \mathbf{s} ; \mathbf{dx} is the grid size in the x -direction; $\text{id}(\mathbf{i}, \mathbf{j}, \mathbf{k}) = i + (j - 1)nx + (k - 1)nxny$ is the identity of the grid point; $\text{ngp} = nxnynz$ is the number of grid points; and neqn denotes number of equations of the ODE system. The Dirichlet boundary condition given by exact solution $\text{exact}(\mathbf{s}, \mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{t})$.

Advection terms

The advection terms can be discretized by the first-order upwind, the second-order central or the third-order upwind biased combined with the limiting technique. In implementation we use the same code with the choice of `limiter` function is equal to 0 and 0.5 for the first-order and the second-order schemes, respectively. For the third-order method, the `limiter` function as follows (see (3.23))

```
real Limiter(real theta){
  real psi;
  psi = max(0.0, min(1.0, min(1.0/3.0 + 1.0*theta/6.0, theta)));
  return psi;
}
```

where the argument `theta` is θ_i given in (3.21)

```
real Theta(real term1, real term2, real term3){
    real epsilon,temp;
    epsilon = ((term3-term2)>0) ? 1.0e-15 : -1.0e-15; /*epsilon used to prevent division by zero*/
    temp = (term2-term1)/(epsilon+term3-term2);
    return temp;
}
```

Having this limiter, we now implement the discretization of the advection term.

```
void Advec(int s, int dir, real *c, real *f, real t) {
    real Value; /* Approximated solution at the center of the cell. */
    real FluxOld, FluxNew; /* The left- and right- fluxes of the current grid point.*/
    real prev, curr, next, last; /* The grid points needed to approximate the flux. */
    int i,j,k;

    switch (dir){
    case XDIR: /* Discretization in the x-direction */
        for(k=1; k<=nz; k++){
            for(j=1; j<=ny; j++){
                /* Calculate the flux at x=0 */
                FluxOld = u(0.0,y[j],z[k])*exact(s,0.0,y[j],z[k],t);
                /* Calculate the flux at next points */
                for(i=1; i<=nx-1; i++){
                    curr = c[id(i, j,k)];
                    next = c[id(i+1,j,k)];
                    if (u(x[i]+dx/2,y[j],z[k])>=0.0){
                        /* If the velocity is positive, the flux uses information from left side.*/
                        prev=(i==1) ? (2.0*exact(s,0.0,y[j],z[k],t)-c[id(1,j,k)]) : c[id(i-1,j,k)];
                        Value = curr + Limiter(Theta(prev,curr,next))*(next-curr);
                    }else{
                        /* If the velocity is negative, the flux uses information from right side.*/
                        last=(i==nx-1) ? (2.0*exact(s,1.0,y[j],z[k],t)-c[id(nx,j,k)]) : c[id(i+2,j,k)];
                        Value = next + Limiter(Theta(last, next, curr))*(curr-next);
                    }
                    FluxNew = u(x[i]+dx/2,y[j],z[k])*Value;
                    /* The advection term in x-direction */
                    f[id(i,j,k)] = (FluxNew - FluxOld)/dx;
                    /* Update information*/
                    FluxOld = FluxNew;
                }
                /* Calculate the flux at x=1 */
                FluxNew = u(1.0,y[j],z[k])*exact(s,1.0,y[j],z[k],t);
                f[id(nx,j,k)] = (FluxNew - FluxOld)/dx;
            }
        }
        break;
    case YDIR:
        SIMILAR IMPLEMENTATION AS IN THE XDIR-PART.
    }
```

```

break;
case ZDIR:
    SIMILAR IMPLEMENTATION AS IN THE XDIR-PART.
break;
default;;
}
return ;
}

```

Diffusion terms

Next, we implement the diffusion discretization using the second-order central scheme.

```

void Diffu(int s, int dir, real *c, real *f, real t) {
    int i,j,k;
    switch (dir){
    case XDIR: /* Discretization in the x-direction */
        for(k=1; k<=nz; k++){
            for(j=1; j<=ny; j++){
                for(i=1; i<=nx; i++){
                    if (i==1){ /* For the left-boundary point */
                        f[id(i,j,k)] = K1*(c[id(i+1,j,k)]-3.0*c[id(i,j,k)]
                            + 2.0*exact(s,0.0,y[j],z[k],t))/(dx*dx);
                    }else if (i==nx){ /* For the right-boundary point */
                        f[id(i,j,k)] = K1*(2.0*exact(s,1.0,y[j],z[k],t)
                            - 3.0*c[id(i,j,k)]+c[id(i-1,j,k)])/(dx*dx);
                    }else{ /* For internal points */
                        f[id(i,j,k)] = K1*(c[id(i+1,j,k)]-2.0*c[id(i,j,k)]
                            + c[id(i-1,j,k)])/(dx*dx);
                    }
                }
            }
        }
        break;
    case YDIR:
        SIMILAR IMPLEMENTATION AS IN THE XDIR-PART.
        break;
    case ZDIR:
        SIMILAR IMPLEMENTATION AS IN THE XDIR-PART.
        break;
    default;;
    }
    return ;
}

```

Reaction terms

For the reaction terms, the expression depends on particular form of the R -function.

```
void React(int s, real *c, real *f, real t) {
    int i,j,k;
    for(i=1; i<=nx; i++){
        for(j=1; j<=ny; j++){
            for(k=1; k<=nz; k++){
                f[id(i,j,k)] = R(s,x[i],y[j],z[k],t);
            }
        }
    }
    return ;
}
```

The explicitly-treated operator

The advection and diffusion terms are treated explicitly. Hence, they appear in the F_E -function.

```
void FE(real *c, real *f, real t) {
    int i;

    nf0e++ ; /* Calculate number of FE evaluations */

    for (i=1; i<=neqn; i++) f[i] = 0.0;

    /* FE for the first-species */

    Advec(1,XDIR,c,tmp1,t);
    Advec(1,YDIR,c,tmp2,t);
    Advec(1,ZDIR,c,tmp3,t);
    Diffu(1,XDIR,c,tmp4,t);
    Diffu(1,YDIR,c,tmp5,t);
    Diffu(1,ZDIR,c,tmp6,t);
    for (i=1; i<=ngp; i++)
        f[i] = -(tmp1[i] + tmp2[i] + tmp3[i]) + tmp4[i] + tmp5[i] + tmp6[i]; ;

    FE FOR OTHER SPECIES IS CALCULATED IN THE SAME WAY.

    return;
}
```

The implicitly-treated operator

As stated before, the reaction term is treated implicitly. Thus, it will appear in the F_I -function, which is implemented below.

```

void FI(real *c, real *f, real t){
  int i;

  nfile++ ; /* Calculate number of FI evaluations */

  for (i=1; i<=neqn; i++) f[i] = 0.0;

  React(1,c,tmp1,t); /* Reaction term for the first-species */
  for (i=1; i<=ngp; i++) f[i] = tmp1[i];

  FOR OTHER SPECIES, THE f-VALUES ARE CALCULATED IN THE SAME MANNER.

  return;
}

```

4.2 Implementation of the time integration

The spatial discretization part results in an ODE system (3.40). Now we describe the implementation of time integration for solving this system using explicit RKC or IMEX-RKC methods. In either cases, the code contains following main steps: Find the number of stages s ; Compute solution at new time step by explicit RKC or IMEX-RKC methods; Check (Tol) error condition and adjust step size based on error estimation.

In the following pseudo code we denote the number of stages by `stage`; `hold` is step size at time t_n ; `hcurr` is step size at time t_{n+1} ; `Estm-old` and `Estm-curr` are weighted norm of estimated error at time t_n and time t_{n+1} , respectively. The structure `W` contains all information of the solution. We assume that all coefficients of RKC are given.

Stages selection

```

int Findstages(Solution *W){
  /* This function find the stages number based on criterion 3.3.3 and adjust
     step size obtained from estimate error procedure to the new step size. */
  real tmp, Betas, Phi1, Phi2;
  int i, sd = 0, sa = 1, stage;

  /* Step 0 */
  COMPUTE Phi1 AND Phi2 ACCORDING TO FORMULA (3.63)
  /* Step 1 */
  if (hcurr <= (2.0*Phi1)){
    stage = 2;
    hcurr = min(hcurr, RPowerR((2.0*Phi2), 1.0/3.0)); /* RPowerR(a,b) = a^b */
  }else{
    /* Step 2 */
    hcurr = min(hcurr, RPowerR((15.5*Phi2), 1.0/3.0));
    if (hcurr <= (2.0*Phi1)){

```

```

    stage = 2;
  }else{
    while(sa>sd){
      /* Step 3 */
      tmp = hcurr/Phi1;
      i = 4;
      Betas = (i*i-1.0)*(0.340+0.189*PowerR(2.0/(i-1.0),1.3));
      while (Betas < tmp){
        i = i+2; /* Because we have assumed s even*/
        Betas = (i*i-1.0)*(0.340+0.189*PowerR(2.0/(i-1.0),1.3));
      }
      sd = i;
      /* Step 4 */
      tmp = hcurr*hcurr*hcurr/Phi2;
      if ( tmp <= (4.0*(6-4)+6.15*(4-4))){
        sa = 4;
      }else if (( (4.0*(6-4)+6.15*(4-4))<tmp) & (tmp<=(4.0*(6-6)+6.15*(6-4)))){
        sa = 6;
      }else if (( (4.0*(6-6)+6.15*(6-4))<tmp) & (tmp<=(6.15*(10-8)/2.0+15.5*(8-6)/4.0))){
        sa = 8;
      }else if ((6.15*(10-8)/2.0+15.5*(8-6.0)/4.0) < tmp){
        sa = 10;
      }
      /* Step 5 */
      hcurr = 0.8*hcurr;
    }
    stage = sd;
  }
}
return stage;
}

```

The explicit RKC method

```

void EXPL_RKC(Solution *W){
  /* This function solve ODEs system using explicit RKC method: Formula (3.45) */
  int i,j,s;

  /* STAGE 0 */
  for (i=1; i<=neqn; i++) temp[i][0] = w[i]; /*W_{0}*/
  /* STAGE 1 */
  FE(w, tmp1, t);
  FI(w, tmp2, t);
  for (i=1; i<=neqn; i++) {
    Fix[i] = tmp1[i] + tmp2[i]; /*F_{0}*/
    temp[i][1] = w[i] + muy_t[1]*hcurr*Fix[i]; /*W_{1}*/
  }
  /* STAGE J */
  for (j=2;j<=stage;j++){

```

```

/*Compute F_{j-1}*/
for (i=1; i<=neqn; i++) tempv1[i] = temp[i][j-1];
FE(tempv1, tmp1, t+c[j-1]*hcurr);
FI(tempv1, tmp2, t+c[j-1]*hcurr);
for (i=1; i<=neqn; i++) tempv2[i] = tmp1[i] + tmp2[i]; /*F_{j-1}*/
/*Compute W_{j}*/
for (i=1; i<=neqn; i++)
    temp[i][j] = (1.0-muy[j]-nuy[j])*temp[i][0]
                + muy[j]*temp[i][j-1] + nuy[j]*temp[i][j-2]
                + muy_t[j]*hcurr*tempv2[i] + gam_t[j]*hcurr*Fix[i]; /*W_{j}*/
}
/* STAGE S */
for (i=1; i<=neqn; i++) wPred[i] = temp[i][stage];

return;
}

```

The IMEX-RKC method

```

void IMEX_RKC(Solution *W){
/* This function solve ODEs system using IMEX-RKC method: Formula (3.53)*/
real error=1.0, Tol=0.0;
int i,j,s;

/* STAGE 0 */
for (i=1; i<=neqn; i++) temp[i][0] = w[i]; /*W_{0}*/

/* STAGE 1 */
FE(w, tmp1, t);
FI(w, tmp2, t+c[1]*hcurr);
for (i=1; i<=neqn; i++) {
    temp1[i][0] = tmp1[i]; /*FE_{0}*/
    temp2[i][0] = tmp2[i]; /*FI_{1}*/
}

/*Compute the fixed part*/
for (i=1; i<=neqn; i++) Fix[i] = w[i] + muy_t[1] * hcurr * tmp1[i];

/*Predict solution at time = t+c[1]*hcurr using forward Euler*/
for (i=1; i<=neqn; i++) wPred[i] = w[i] + c[1] * hcurr * (tmp1[i]+tmp2[i]);

/*Solve implicit system using modified Newton method and LU-decomposition*/
while(error >= Tol){
    /*Find the RHS of Newton iteration and save results in vector tempv1*/
    NewtonRhs(W, wPred, Fix, muy_t[1]*hcurr, tempv1, t+c[1]*hcurr);

    /*LU-Decomposition matrix: (I-muy_t[1]*hcurr*JACOBI)*/
    Decomposition(W, muy_t[1]*hcurr);
}
}

```

```

/*Solve system and save results in vector tempv2*/
AMF(W, tempv1, tempv2);

/* Update solution*/
for (i=1; i<=neqn; i++) wPred[i] = wPred[i] + tempv2[i];

error = L2Norm(tempv2, 1, neqn);
Tol = Tolerance(atol, rtol, wPred, 1, neqn);
}
for (i=1; i<=neqn; i++) temp[i][1] = wPred[i]; /*W_{1}*/

/* STAGE J */
for (j=2; j<=stage; j++){
/*Compute FE_{j-1}*/
for (i=1; i<=neqn; i++) tempv1[i] = temp[i][j-1];
FE(tempv1, tmp1, t+c[j-1]*hcurr);
for (i=1; i<=neqn; i++) temp1[i][j-1] = tmp1[i]; /*FE_{j-1}*/

/*Compute FI_{j-2}*/
for (i=1; i<=neqn; i++) tempv1[i] = temp[i][j-2];
FI(tempv1, tmp1, t+c[j-2]*hcurr);
for (i=1; i<=neqn; i++) temp2[i][j-2] = tmp1[i]; /*FI_{j-2}*/

/*Compute the fixed part*/
for (i=1; i<=neqn; i++)
Fix[i] = (1.0-muy[j]-nuy[j])*temp[i][0] + muy[j]*temp[i][j-1]+nuy[j]*temp[i][j-2]
+ muy_t[j]*hcurr*temp1[i][j-1] + gam_t[j]*hcurr*temp1[i][0]
+ (gam_t[j]-(1.0-muy[j]-nuy[j])*muy_t[1])*hcurr*temp2[i][0]
- nuy[j]*muy_t[1]*hcurr*temp2[i][j-2];

/*Predict solution at time = t+c[j]*hcurr using forward Euler*/
FE(w, tmp1, t);
FI(w, tmp2, t+c[j]*hcurr);
for (i=1; i<=neqn; i++) wPred[i] = w[i] + c[j] * hcurr * (tmp1[i]+tmp2[i]);

/*Solve implicit system using modified Newton method and LU-decomposition*/
error = 1.0; Tol = 0.0;
while(error >= Tol){
NewtonRhs(W, wPred, Fix, muy_t[1]*hcurr, tempv1, t+c[j]*hcurr);
Decomposition(W, muy_t[1]*hcurr);
AMF(W, tempv1, tempv2);
for (i=1; i<=neqn; i++) wPred[i] = wPred[i] + tempv2[i];
error = L2Norm(tempv2, 1, neqn);
Tol = Tolerance(atol, rtol, wPred, 1, neqn);
}
for (i=1; i<=neqn; i++) temp[i][j] = wPred[i]; /*W_{j}*/
} /* End j */
return;
}

```

Error estimation

```

int Error_Estimate(Solution *W)
{
    /* This function estimate local error using equation (3.56). Base on this estimated
       error and Tol, we determine the new step size using equation (3.58) and (3.59) */

    real Tol, fac, Error_est, Estm_curr;
    int i, Converged;

    /*-----Compute Tolerance: Tol=max(Tol(w_{n}),Tol(w_{n+1}))-----*/
    Tol = max(Tolerance(atol, rtol, w, 1, neqn), Tolerance(atol, rtol, wPred, 1, neqn));

    /*-----Compute Estm_{n+1} using equation (3.56)-----*/
    FE(wPred, tmp1, t+hcurr);
    FI(wPred, tmp2, t+hcurr);
    for (i=1; i<=neqn; i++) tmp3[i] = tmp1[i] + tmp2[i];
    FE(w, tmp1, t);
    FI(w, tmp2, t);
    for (i=1; i<=neqn; i++) tmp4[i] = tmp1[i] + tmp2[i];
    for (i=1; i<=neqn; i++) tmp5[i] = tmp3[i] - tmp4[i]; /*(F(t_{n+1},w_{n+1})-F(t_{n},w_{n}))*/

    FI(wPred, tmp1, t+hcurr);
    FI(w, tmp2, t);
    for (i=1; i<=neqn; i++) tmp6[i] = tmp1[i] - tmp2[i]; /*(FI(t_{n+1},w_{n+1})-FI(t_{n},w_{n}))*/

    for (i=1; i<=neqn; i++) tmp7[i] = 0.5*hcurr*tmp5[i]+(3.0/(stage*stage-1.0))*hcurr*tmp6[i];

    Error_est = L2Norm(tmp7,1,neqn); /*Estm_{n+1}*/

    /*-----Determine new step size-----*/
    Estm_curr = Error_est/Tol; /*Weighted norm of error*/
    fac = 0.8*sqrt(Estm_old)*hcurr/(hold*Estm_curr);
    hold = hcurr; /*Update step size*/
    hcurr = min(10.0, max(0.1, fac)) * hcurr; /*New step size*/

    if (Estm_curr<=1.0){
        Converged = 1;
        accept = accept+1;
    }
    else{
        Converged = 0;
        reject = reject+1;
    }

    Estm_old = Estm_curr; /*Update weighted norm */

    return Converged;
}

```

Time integration procedure

```

void Integration(Solution *W)
{
    int  flag = 0, i;

    hcurr = Starting step size;
    Estm_old = 1.0; /*Initial value for weighted norm of estimated error*/

    while (t <= t_END) {
        while (flag < 1){ /* If current step rejected*/

            stage = Findstages(W);

            FIND ALL THE COEFFICIENTS OF RKC;

            if (IMEX){
                IMEX_RKC(W);
            }else{
                EXPL_RKC(W);
            }
            flag = Error_Estimate(W);
        }
        /*-----Update solution-----*/
        for (i=1; i<=neqn; i++) w[i] = wPred[i];

        t = t + hcurr;
        nst++ ; /* Counts number of steps */
    }
    return;
}

```

4.3 The main function

The main function

```

int main(int argc, char** argv)
{
    INPUT PARAMETERS

    SPATIAL DISCRETIZATION

    TIME INTEGRATION

    DISPLAY AND SAVE RESULTS

    return;
}

```

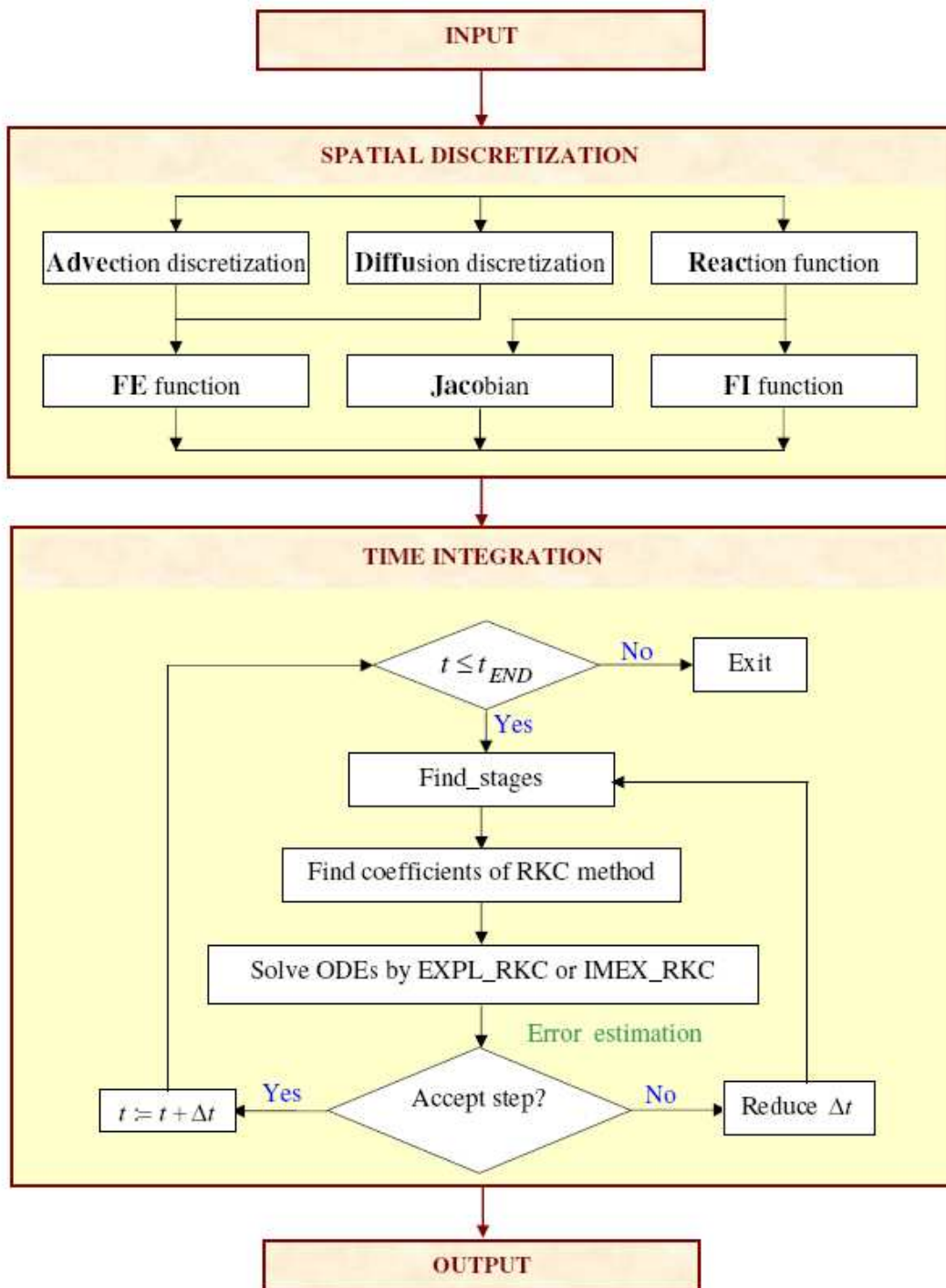


Fig. 4.1: Implementation for solving advection-diffusion-reaction equations.

5. NUMERICAL EXPERIMENTS

5.1 Spatial discretization tests

In this section, we will illustrate the performance of the spatial discretization methods discussed in Section 3.1: the second-order finite difference scheme for diffusion terms; the first-order upwind scheme, the second-order central scheme, and the limited third-order upwind-biased scheme for advection terms. The experiments are to study the order of accuracy and behaviour of numerical solution produced by each method.

For simplicity, we consider the reacting chemical species equation in 1D domain and time interval $[0,1]$, i.e.,

$$c_t + uc_x = Kc_{xx} + R(c), \quad x \in [0, 1], \quad (5.1)$$

whose exact solution is $c(x, t) = c(x, 0) = (\sin(\pi x))^{100}$ and $R(c)$ is given by

$$R(c) = u 100\pi (\sin(\pi x))^{99} \cos(\pi x) - K 100\pi^2 \left(99 (\sin(\pi x))^{98} (\cos(\pi x))^2 - (\sin(\pi x))^{100} \right). \quad (5.2)$$

We will use the Backward Euler method with a very small time step such that the error of the full PDEs is dominated by the spatial discretization error.

5.1.1 Test for discretization of diffusion term

Firstly, we study the spatial discretization of diffusion term. Setting $u = 0$ and $K = 1$, Eq. (5.1) is then in term of diffusion-reaction equation. The reaction term is given exactly by (5.2). The diffusion term is discretized by the second-order central scheme on a uniform spatial grid with grid sizes $1/10$, $1/20$, $1/40$, $1/80$, and $1/160$.

Table 5.1 shows the L_∞ -error¹ of diffusion-reaction equation with time step $\tau = 10^{-6}$. We observe that when the grid size decreases twice, the error reduces about four times. This implies the second-order accuracy of the method. Figure 5.1 shows the numerical solution on grid size $1/100$ and step size $\tau = 10^{-3}$. We see that numerical solution is smooth and accurate.

¹ L_∞ norm of a vector $x = (x_1, x_2, \dots, x_n)$ is denoted by $\|x\|_\infty = \max |x_i|, i = 1, \dots, n$.

Tab. 5.1: L_∞ -error of the second-order central diffusion discretization scheme: The errors decrease by second-order with number of grid points. Step size $\tau = 10^{-6}$.

Grid size	L_∞ -error	order
1/10	1.13e-02	
1/20	2.88e-03	1.96
1/40	7.24e-04	1.99
1/80	1.83e-05	1.98
1/160	4.73e-05	1.93

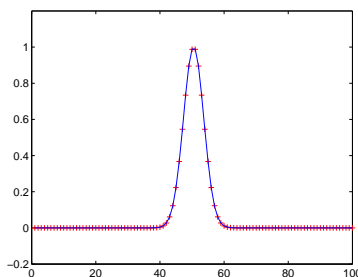


Fig. 5.1: Diffusion test, using the second-order central scheme with grid size 1/100. Step size $\tau = 10^{-3}$. Numerical solution (solid line) and exact solution (plus-marked line).

5.1.2 Test for discretization of advection term

To study the spatial discretization of the advection term, we choose $u = 1$ and $K = 0$. Then, Eq. (5.1) becomes the advection-reaction equation. Again, the reaction term is given exactly by (5.2). The advection term is discretized by the first-order upwind, the second-order central, and the limited third-order upwind-biased schemes. The step size and grid sizes are chosen as given in the preceding subsection.

L_∞ -errors obtained by these schemes with step size $\tau = 10^{-6}$ are listed in Table 5.2. From this table we see that when the grid size decreases twice, the errors in the second and third columns decrease about two and four times, respectively. This shows the first- and the second-order accuracy of the spatial discretization methods. From the last column we see that the order of accuracy of the limited third-order upwind-biased is slightly larger than one. This is because the limiter switches to the first-order upwind scheme, due to the steep profile of the solution.

Figure 5.2 shows results for advection-reaction equation on grid size 1/100 and step size

Tab. 5.2: L_∞ -error of the first-order upwind, the second-order central, and the limited third-order upwind-biased advection discretization schemes. Time step $\tau = 10^{-6}$.

Grid size	1 st -order upwind		2 nd -order central		3 rd -order upwind-biased + limiter	
	L_∞ -error	order	L_∞ -error	order	L_∞ -error	order
1/10	2.31.e-04		5.10.e-05		4.84.e-04	
1/20	1.16.e-04	1.0	1.30.e-05	1.96	2.38.e-04	1.0
1/40	5.66.e-05	1.0	3.28.e-06	1.98	1.15.e-04	1.2
1/80	2.77.e-05	1.0	8.33.e-07	1.97	5.40.e-05	1.1
1/160	1.35.e-05	1.0	2.15.e-07	1.94	2.36.e-05	1.1

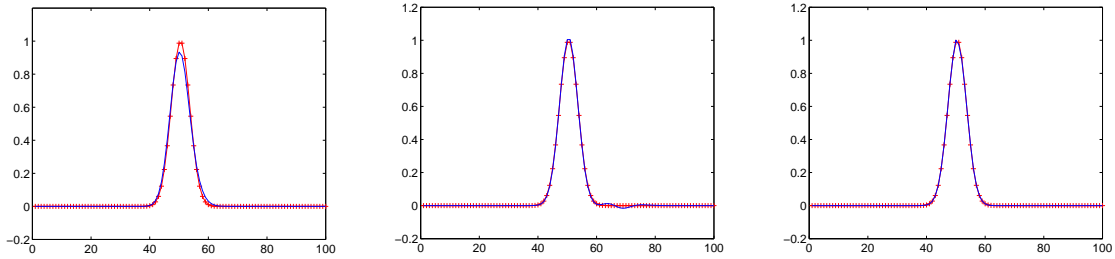


Fig. 5.2: Advection test, using the first-order upwind scheme (left), the second-order central scheme (middle), and the limited third-order upwind-biased scheme (right) with grid size 1/100, step size $\tau = 10^{-3}$. Plus-marked line: exact solution. Solid lines: numerical solutions.

$\tau = 10^{-3}$. With the first-order upwind scheme, the numerical solution is monotone but inaccurate, solution at the peak is diffusive. With the second-order central scheme, we obtain a more accurate solution. However, oscillations occur at the right side and solution contains negative values. The third-order upwind-biased scheme combined with the limiting technique gives a good result. The solution is accurate, non-oscillatory and positive.

In summary, we have shown that the chosen second-order central diffusion discretization scheme and the limited third-order upwind-biased advection discretization scheme are indeed of satisfactory.

5.2 Time integration tests

In this section, we study two test examples for the time integration. In subsection 5.2.1, a nonlinear Burgers equation of advection-diffusion type is solved. The Molenkamp equation, which is of advection-reaction type with stiff reaction terms, is studied in subsection 5.2.2.

For the spatial discretization, we use methods that have been successfully tested in Section 5.1. That is, the second-order central method for diffusion terms and third-order upwind-biased scheme combined with limiting technique for advection terms.

For the time integration, the IMEX-RKC method will be applied for the Molenkamp equation which contains the stiff reaction terms. For the Burgers equation, the reaction is absent, both advection and diffusion terms are treated explicitly. Therefore we apply the explicit RKC method.

The solver uses damping parameter $\varepsilon = 10$ and stability boundary (3.54). We use the variable step size control as described in Section 3.3. For the Burgers equation the stages number is selected by criterion given in subsection 3.3.3. For the Molenkamp equation in which the diffusion term is absent and the stiff reaction terms are solved implicitly, we use the CFL condition (3.64) to determine the step size and stages number.

5.2.1 Problem 1: Burgers equation

In this section we consider the advection form of Navier-Stokes equation discussed in Section 2.2. Assuming that both pressure gradient p and gravitation g are omitted, equation (2.16) then reads

$$\begin{aligned} u_t + uu_x + vu_y + wu_z &= \varepsilon \Delta u, \\ v_t + uv_x + vv_y + wv_z &= \varepsilon \Delta v, \\ w_t + uw_x + vw_y + ww_z &= \varepsilon \Delta w. \end{aligned} \quad (5.3)$$

Here u , v and w represent the velocity field in x -, y - and z - direction, respectively, and ε is *Reynolds* number. This system is the so-called Burgers equation.

We find the exact solutions for (5.3) such that it describes the moving of a wave front skew through a unit cube.

Using the Cole-Hopf transformation (see [18])

$$u = -2\varepsilon \frac{\varphi_x}{\varphi}, \quad v = -2\varepsilon \frac{\varphi_y}{\varphi}, \quad w = -2\varepsilon \frac{\varphi_z}{\varphi}, \quad (5.4)$$

then (5.3) reads

$$\varphi_t = \varepsilon \Delta \varphi. \quad (5.5)$$

An exact solution representing a single shock is given by Whitham ([18]) as follows

$$\varphi = f_1 + f_2, \quad (5.6)$$

where

$$f_i = \exp\left(-\frac{c_{i1}x + c_{i2}y + c_{i3}z}{2\varepsilon} + \frac{(c_{i1}^2 + c_{i2}^2 + c_{i3}^2)t}{4\varepsilon}\right), \quad i = 1, 2. \quad (5.7)$$

The center of the shock is where $f_1 = f_2$, and we choose this center at a plane skew through the cube

$$x - y - z = -\frac{3}{4}t. \quad (5.8)$$

This is obtained when $c_{jj} = 1$ and $c_{ij} = 1/2$ giving as exact solution for (5.3)

$$u = 1 - \frac{1}{2\left(1 + \exp\left(\frac{-x+y+z-\frac{3}{4}t}{4\varepsilon}\right)\right)}, \quad (5.9)$$

$$v = w = \frac{3}{2} - u.$$

Substituting v and w from (5.9) into (5.3) we get the scalar PDE

$$u_t + uu_x + \left(\frac{3}{2} - u\right)(u_y + u_z) = \varepsilon\Delta u. \quad (5.10)$$

In one dimension the Burgers equation has following exact solution

$$u(x, t) = 1 - 0.9\frac{r_1}{r_1 + r_2 + r_3} - 0.5\frac{r_2}{r_1 + r_2 + r_3}, \quad (5.11)$$

where

$$r_1 = \exp\left(\frac{0.5 - x}{20\varepsilon} - \frac{99t}{400\varepsilon}\right), \quad r_2 = \exp\left(\frac{0.5 - x}{4\varepsilon} - \frac{3t}{16\varepsilon}\right), \quad r_3 = \exp\left(\frac{3/8 - x}{2\varepsilon}\right). \quad (5.12)$$

We solve problem (5.3) in time interval $[0,1]$. The domain is the unit cube. The uniform spatial grid with grid size $1/20$ is used. The initial value and Dirichlet boundary condition are given by the exact solution (5.9). The advection terms are discretized by the third-order upwind-biased combined with limiter and the second-order central approximation method is used for discretizing diffusion terms. The diffusion coefficients $\varepsilon = 0.1$ and $\varepsilon = 0.01$ are tested. The values of tolerance (Tol) ranges from 10^{-1} to 10^{-4} . The results are listed in Tables 5.3 and 5.4.

The structure of Tables 5.3 and 5.4 is as follows: The first column contains the relative tolerances (absolute tolerance is set to 10^{-6}). The second column contains the total number of the accepted plus the rejected steps. The third column shows the number of F -evaluations (F-eval). The fourth column indicates the L_2 -errors² of the full PDEs equation. s_{max} denotes the maximal number of stages. The last column contains the computation time.

² L_2 norm of a vector $x = (x_1, x_2, \dots, x_n)$ is $\|x\|_2 = \sqrt{\frac{1}{n} \sum_{i=1}^n |x_i|^2}$.

Tab. 5.3: The performance of explicit RKC for 3D-Burgers equation with $\varepsilon = 0.1$. Grid size: 1/20.

Tol	Steps (Rejects)	F-evals	s_{max}	L_2 error	Cost(s)
10^{-1}	17 (0)	198	10	6.94×10^{-2}	6
10^{-2}	34 (0)	376	8	3.78×10^{-2}	12
10^{-3}	97 (0)	776	4	3.38×10^{-2}	23
10^{-4}	212 (0)	1684	4	2.94×10^{-2}	49

Table 5.3 shows the results for Burgers equation with $\varepsilon = 0.1$ and different values of tolerance. From this table we see that the error estimation and the stages selection procedures are successfully used. The code always satisfies with the chosen step size and the stages number. No step rejection occurs.

With the large values of tolerance ($\text{Tol} = 10^{-1}, 10^{-2}$), the estimated errors easily satisfy the tolerance conditions. This allows a large time step, resulting in a small number of steps. Accordingly, the integration is cheap (small number of F-evaluations and CPU time). However, with such the large step size, we have to increase the stages number in order to guarantee the stability condition. Thus, the stages numbers are large. In contrast to these cases, with smaller values of tolerance ($\text{Tol} = 10^{-3}, 10^{-4}$), the step sizes obtained from error estimation procedure are smaller. As a results, the number of steps, the number of F-evaluations and CPU time increase, while the stages numbers decrease.

We note that the errors do not change much when the tolerances decrease from 10^{-2} to lower values. This is because the errors are mainly determined by the spatial discretization. More precisely, for these tolerances, the step size are smaller than the grid size (the grid size is 1/20, while the time step is 1/34, 1/97, and 1/212 for $\text{Tol}=10^{-2}, 10^{-3}$, and 10^{-4} , respectively).

Tab. 5.4: The performance of explicit RKC for 3D-Burgers equation with $\varepsilon = 0.01$. Grid size: 1/20.

Tol	Steps (Rejects)	F-evals	s_{max}	L_2 error	Cost(s)
10^{-1}	22 (0)	166	4	3.17×10^{-1}	5
10^{-2}	49 (0)	382	4	2.97×10^{-1}	11
10^{-3}	80 (0)	480	2	2.82×10^{-2}	13
10^{-4}	227 (0)	1362	2	3.37×10^{-3}	38

Table 5.4 shows the results for Burgers equation with $\varepsilon = 0.01$. From this table we again see that the larger the tolerance, the smaller the number of steps and F-evaluations, and the larger the number of stages. The errors are also mainly contributed by the spatial discretization.

Comparing the results in Tables 5.3 and 5.4, it turns out that the larger the ε -value, the larger the stages number and the more expensive the computation. This is due to the fact that the larger the ε the more stiff the diffusion term. This term is treated explicitly. Hence, the larger number of stages is needed, implying the more expensive computation. This result is in accordance with the stability condition discussed in Section 3.3, where both stability conditions for advection and diffusion terms depend on the diffusion coefficient (see (3.60) and (3.61)).

Figure 5.3 shows the solution of 3D Burgers equation at time $t = 1$. The solution describes the moving of a wave in the unit cube, the value varies from 0.5 to 1. With a large epsilon

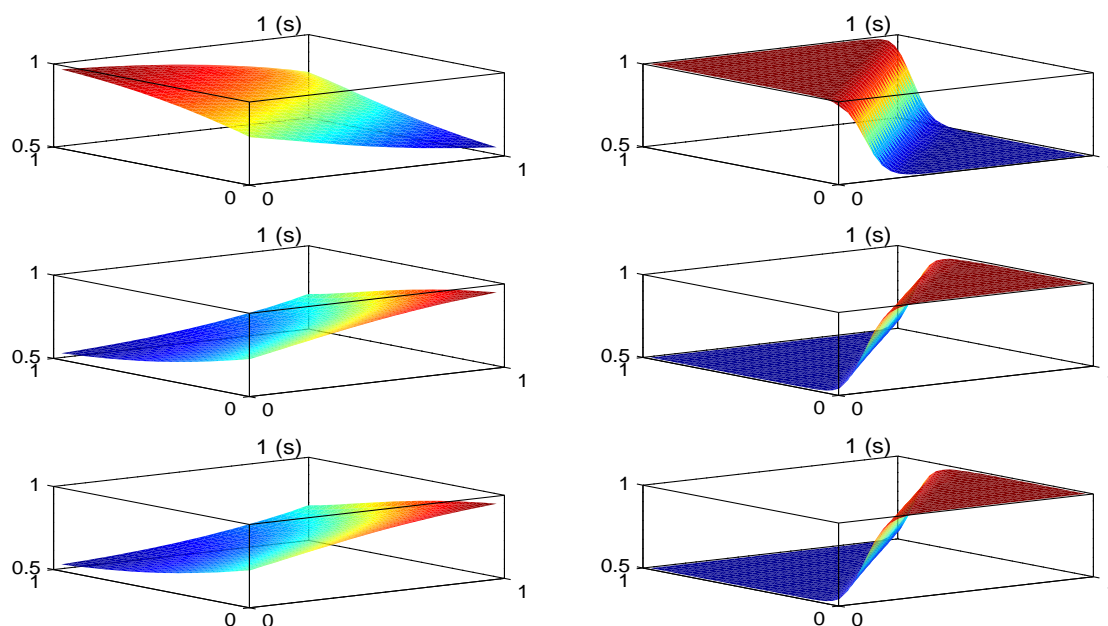


Fig. 5.3: The Burgers equation in three dimensions with $\varepsilon = 0.1$ (left) and $\varepsilon = 0.01$ (right). Numerical solution at time $t = 1$. (Grid size: $1/30$).

value ($\varepsilon = 0.1$), the solution is nearly a skew plane moving slowly from right to left for u and opposite direction for v and w . With a small epsilon value ($\varepsilon = 0.01$), the solution includes

two planes at values 0.5 and 1.0, and a skew plane moving quickly from right to left for u and opposite direction for v and w . At time $t > 2$ the skew plane moves out of the unit cube, the solution u becomes a plane at value 1.0, and the solutions v and w become a plane at value 0.5.

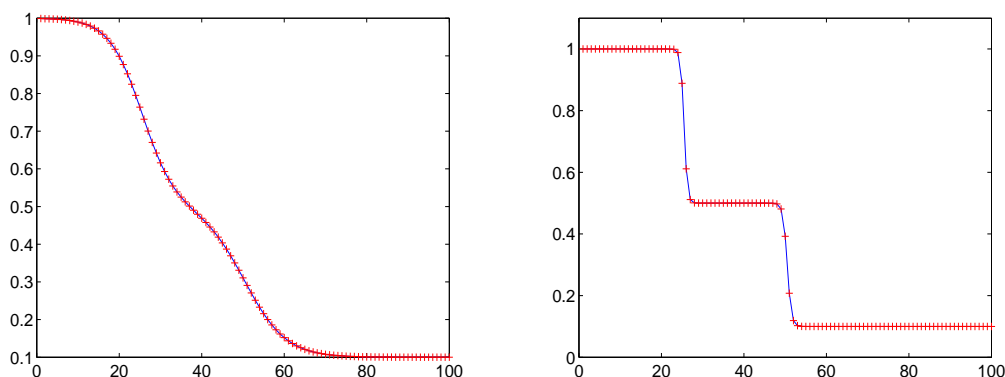


Fig. 5.4: The Burgers equation in one dimension with $\varepsilon = 0.01$ (left) and $\varepsilon = 0.001$ (right). The numerical solutions are indicated by solid lines. The exact solutions are plus-marked lines.

Figure 5.4 shows the solution of 1D Burgers equation with $\varepsilon = 0.01$ (left) and $\varepsilon = 0.001$ (right) at time $t = 1$. We see that the numerical solutions are accurate and smooth.

5.2.2 Problem 2: Molenkamp equation

As an example of advection-reaction equation, in this section we solve the Molenkamp equation

$$c_t + u c_x + v c_y + w c_z = f(c), \quad (5.13)$$

where u, v and w are components of the velocity field in x -, y - and z -directions, respectively and the reaction term consists of two components

$$\begin{aligned} f_1 &= -k_2 c_1 c_2 + k_1 c_2^2, \\ f_2 &= +k_2 c_1 c_2 - k_1 c_2^2. \end{aligned} \quad (5.14)$$

If we set the sum of c_1 and c_2 equal to constant d , then the solution of (5.13) with zero velocity ($u = v = w = 0$) is given by

$$\begin{aligned} c_1(t) &= \frac{d}{k_1 + k_2} \frac{k_1(1 - \alpha) + (k_1 + k_2) \alpha e^{-dk_2 t}}{1 - \alpha + \alpha e^{-dk_2 t}}, \\ c_2(t) &= d - c_1(t), \end{aligned} \quad (5.15)$$

with

$$\alpha = \frac{(k_1 + k_2) c_1(0) - d k_1}{d k_2}. \quad (5.16)$$

We select $k_1 = 1000, k_2 = 1$ and choose $c_1(0) = 0$ and $c_2(0) = d$ as initial values, so that $\alpha = -k_1/k_2$.

For the advection part we determine an exact solution such that it represents a rotating sphere with the highest solution in its center, along an ellipse on a skew plane $y = z$ through the unit cube. This rotation is obtained by choosing the velocities

$$\begin{aligned} u &= 2\pi\sqrt{2} \left(\frac{y+z}{2} - \frac{1}{2} \right), \\ v &= -\pi\sqrt{2} \left(x - \frac{1}{2} \right), \\ w &= v. \end{aligned} \quad (5.17)$$

The exact solution for (5.13) with velocities as (5.17) and $f(c) = 0$ is

$$c(t) = \exp \left(-80 \left[(x - r(t))^2 + (y - s(t))^2 + (z - s(t))^2 \right] \right), \quad (5.18)$$

where

$$r(t) = \frac{2 + \sin(2\pi t)}{4}, \quad s(t) = \frac{4 + \sqrt{2} \cos(2\pi t)}{8}. \quad (5.19)$$

Combining (5.15) and (5.18), we obtain the exact solution for (5.13) as follows

$$\begin{aligned} c_1(t) &= d(x, y, z, t) \frac{1 - e^{-d(x,y,z,t)t}}{\frac{1001}{1000} - e^{-d(x,y,z,t)t}}, \\ c_2(t) &= d(x, y, z, t) \frac{10^{-3}}{\frac{1001}{1000} - e^{-d(x,y,z,t)t}}, \end{aligned} \quad (5.20)$$

where

$$d(x, y, z, t) = \exp \left(-80 \left[(x - r(t))^2 + (y - s(t))^2 + (z - s(t))^2 \right] \right). \quad (5.21)$$

We solve Molenkamp equation (5.13) in the time interval $[0,1]$. The domain and grid size are the same as in Section 5.2.1. The initial value and Dirichlet boundary condition are given by the exact solution (5.20). The advection terms are discretized by the third-order upwind-biased combined with limiting technique. The stiff reaction terms are treated implicitly. To control the step size and determine the stage number, the CFL condition (3.64) is used. Tolerance (Tol) ranges from 10^{-1} to 10^{-3} . The results are shown in Table 5.5.

From this table we see that, similar as shown in Tables 5.3 and 5.4, the error estimation and

Tab. 5.5: The performance of IMEX-RKC for Molenkamp equation in 3D. Grid size: 1/20.

Tol	Steps (Rejects)	F-evals	s_{max}	L_2 error	Cost(s)
10^{-2}	205 (0)	2048	2	2.59×10^{-2}	43
10^{-3}	642 (0)	6419	2	2.52×10^{-2}	135

stages selection procedure were successfully used for $Tol \leq 10^{-2}$.

The number of stages for this problem is always equal to 2. This means that the RKC method becomes the trapezoidal rule. This is reasonable because the explicit trapezoidal rule method is most suitable for advection problem discretized by the third-order upwind-biased scheme.

Similar to the results in Tables 5.3 and 5.4, the errors in Table 5.5 do not change much when Tol decreases. This is because with $Tol \leq 10^{-2}$ the step sizes ($\tau \leq 1/205$) are small compared with the grid size (1/20). Hence, the errors are mainly contributed by spatial discretization.

The left panel of Figure 5.5 shows the initial values (at $t = 0$) at which $c_1(0) = 0$ and $c_2(0) = d(x, y, z, 0)$. In the short time interval $[0, 0.01]$ the solution $c_1(t)$ increases rapidly from 0 to $d(x, y, z, t)/2$. In contrast to $c_1(t)$, the solution $c_2(t)$ decreases quickly from $d(x, y, z, t)$ to $d(x, y, z, t)/2$ (see the right panel in Figure 5.5). Beyond this time interval, both solutions $c_1(t)$ and $c_2(t)$ change in time mainly by advective transport. That is, both solutions turn around the center of the plane and turn back to the starting position at time $t = 1, 2, 3, \dots$ (see Figure 5.6). The amplitudes of c_1 and c_2 are approaching the values $d(x, y, z, t)$ and 0, respectively.

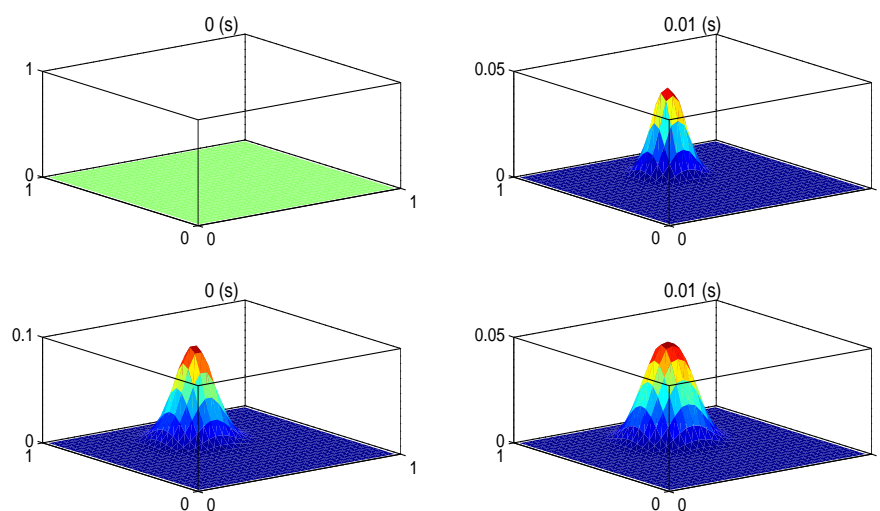


Fig. 5.5: Three-dimensional Molenkamp equation. Initial values at $t = 0$ (left) and solution at $t = 0.01$ (right). Both components c_1 (first row) and c_2 (second row) rapidly approach the value $d(x, y, z, t)/2$ after a short time interval $[0, 0.01]$. Grid size: $1/30$.

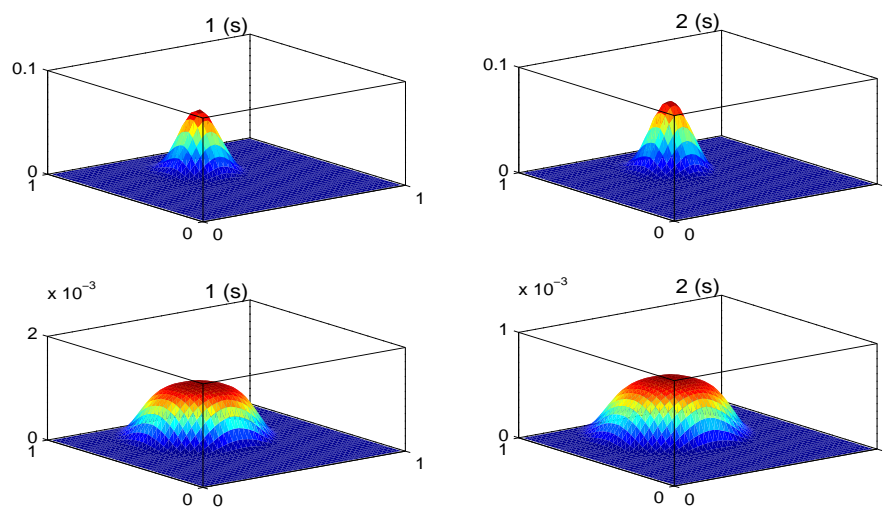


Fig. 5.6: Three-dimensional Molenkamp equation. Solution at time $t = 1$ (left) and $t = 2$ (right). Both components c_1 (first row) and c_2 (second row) turn around the center of the plane. Positions of the solutions at time $t = 1, 2, 3, \dots$ are the same. Grid size: $1/30$.

6. DISCUSSION AND FUTURE WORK

6.1 Discussion

In this thesis, we have developed a numerical method to solve advection-diffusion-reaction equations. We use the finite volume method for spatial discretization and the IMEX-RKC method for time integration.

In spatial discretization, the diffusion terms are discretized by the most popular method, that is, the second-order central scheme. For the advection terms, we use an accurate and positive discretization, that is, the third-order upwind-biased scheme combined with the limiting technique.

Then, the semi-discrete ODE system resulting from spatial discretization is integrated in time. The IMEX-RKC is chosen as our time integrator. Originally, RKC automatically extends the stability region along the negative real axis by adding more stages to the method. With a variable timestep strategy the method can automatically find an optimal balance between accuracy and stability for problems with diffusion and mildly stiff reactions. Later on, the method has been extended to include also advection operators by enlarging the stability region along the imaginary axis and finding the optimal number of stages balancing the influence of the advection and the diffusion/reaction operators ([16]). A second extension has been the use of the IMEX (implicit-explicit) approach ([15]). This can be seen as “operator splitting within the method” which, in contrast to the standard operator splitting techniques like e.g. Strang splitting, does not give rise to a splitting error in time. In the IMEX-RKC method, highly stiff operators are handled implicitly and non-stiff or mildly stiff operators explicitly. In this thesis the spatial operators - advection and diffusion - are treated explicitly and the chemistry implicitly. This implies that the only linear systems to be solved are block-diagonal containing the chemical components within a grid cell, thus simplifying a parallel implementation significantly.

To have more insight in the performance of the proposed method, several numerical experiments are presented in Section 5.1, i.e. for spatial discretization, and in Section 5.2, i.e. for time integration.

For the spatial discretization, the results turned out that using the second-order central scheme

for the diffusion terms and the limited third-order upwind-biased scheme for the advection terms is a good choice. The solutions are indeed positive and accurate as demonstrated in Figures 5.1 and 5.2, and in Tables 5.1 and 5.2.

For the time integration, two test examples are studied: the Burgers and the Molenkamp equations. The Burgers equation does not contain reaction terms. Hence, the explicit RKC method is used. Tables 5.3 and 5.4 show that for different values of tolerance and diffusion coefficient ε , the error estimation and stages selection procedure were successfully used. The code always satisfied with the chosen step size and the stages number. No step rejection occurs. The error and computational cost are within a satisfactory range (for example, with grid size $1/20$, $\text{Tol} = 10^{-2}$ and $\varepsilon = 0.1$, the L_2 error is $3.38 \cdot 10^{-2}$ and the cost is $23(s)$, see Table 5.3). Comparing the Burgers equation with two different values of diffusion coefficient ε , it turns out that the larger the ε , the more expensive the computation. This is due to the fact that, the larger ε the larger stiffness of the diffusion term. This term is treated explicitly. Hence, the larger number of stages is needed. This implies more expensive computation.

The Molenkamp equation contains stiff reaction terms, therefore we use the IMEX-RKC method. It was shown that the IMEX-RKC method is unconditional stable for the implicitly-treated operator, and its stability is determined by that of the explicit counterpart. In this problem, the diffusion term is not included, the explicit part consists of the advection terms only. With the given step size, the stages selection is based on stability of the advection terms. As it was shown in Table 5.5, for different tolerance values, the code always integrates with two stages ($s = 2$). This implies that the RKC method has switched to the trapezoidal rule method. This is reasonable since the explicit trapezoidal rule method is the most suitable for advection problem discretized by the third-order upwind-biased scheme. Similar to the Burgers test example, the code integrates without any failures. The resulting error and computational cost are within a satisfactory range (for instance, with grid size $1/20$ and $\text{Tol} = 10^{-2}$, the L_2 error is $2.59 \cdot 10^{-2}$ and the cost is $43(s)$, see Table 5.5).

6.2 Future work

This thesis is a part of a larger project which aims to develop a 3D parallel code using a finite volume discretization in space and a recently proposed RKC method in time, then, apply this code to solve the *incompressible Navier-Stokes* equations for the flow coupled with a system of advection-diffusion-reaction equations that describe the reactions of chemical species dissolved in the water. One of the applications for which the code is meant is flow and reactions around corals and sponges which implies highly complex geometries.

At the current stage, we have developed the finite volume combined with IMEX-RKC method to solve advection-diffusion-reaction equations coming from model of reacting chemical species

in fluid flow. For other applications of this type, similar technique can be obtained. The domain that we considered is the three-dimensional rectangular. Therefore, needed extensions in the future are not only the geometry and the parallelization, but also making the flow incompressible, i.e. solving $\nabla \cdot u = 0$, and apply to the model in a realistic case study.

BIBLIOGRAPHY

- [1] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, 2nd ed., Texts in Applied Mathematics 15, Springer-Verlag, 2002.
- [2] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*, 2nd ed., Springer Series in Computational Mathematics 14, Springer-Verlag, Berlin, 1996.
- [3] W. Hundsdorfer and J. G. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, Springer Series in Computational Mathematics 33, Springer-Verlag, 2003.
- [4] J. A. Kaandorp, E. A. Koopman, P. M. A. Sloot, R. P. M. Bak, M. J. A. Vermeij, and L. E. H. Lampmann, *Simulation and analysis of flow patterns around the scleractinian coral *Madracis mirabilis* (Duchassaing and Michelotti)*, Philos Trans R Soc Lond B Biol Sci. 358; pp. 1551-1557, 2003 .
- [5] B. Koren, *A robust upwind discretization for advection, diffusion and source terms*. In: *Numerical Methods for Advection-Diffusion Problems*. Eds. C.B Vreugdenhil, B. Koren, Notes on Numerical Fluid Mechanis, Vol. 45, Viegweg, Braunschweig, pp. 117–138, 1993.
- [6] B. van Leer, *Towards the ultimate conservative difference scheme II. Monotonicity and conservation combined in a second-order scheme*, J. Comput. Phys. 14, pp. 361-370, 1974.
- [7] N. K. Madsen, *The method of lines for the numerical solution of partial differential equations*, Proceeding of the SIGNUM meeting on software for partial differential equations, pp. 5-7, 1975.
- [8] W. C. Muller, D. Carati, *Dynamic gradient-diffusion subgrid models for incompressible magnetohydrodynamic turbulence*, Physics of plasmas, Volume 9, Number 3, March 2002.
- [9] D. Q. A. Nguyen, V. T. Vu, Z. Wang, and Y. Zhang, *Modeling diffusion limited aggregation using distributed domputing*, Report on the Master course “Profile project” supervised by J. Kaandorp, Amsterdam University, 2004.

-
- [10] L. F. Shampine, *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall, New York, 1994.
- [11] L. F. Shampine, B. P. Sommeijer, J. G. Verwer, *IRKC: an IMEX solver for stiff diffusion-reaction PDEs*, CWI Report, MAS-E0513, ISSN 1386-3703, 2005.
- [12] J. W. Thomas, *Numerical Partial Differential Equations: Finite Difference Methods*, Texts in Applied Mathematics 22, Springer-Verlag, 1995.
- [13] A. E. P. Veldman, *Computational Fluid Dynamics*, Lecture Notes, University of Groningen, The Netherlands, 2001.
- [14] J. G. Verwer, W. Hundsdorfer and J. G. Blom, *Numerical Time Integration for Air Pollution Models*, Surveys on Mathematics for Industry 10, pp. 107–174, 2002.
- [15] J. G. Verwer, B. P. Sommeijer, *An implicit-explicit Runge-Kutta-Chebyshev scheme for diffusion-reaction equations*, SIAM J. Sci. Comput. 25, pp. 1824–1835, 2004.
- [16] J. G. Verwer, B. P. Sommeijer, W. Hundsdorfer, *RKC time-stepping for advection-diffusion-reaction problems*, J. Comput. Phys. 201, pp. 61–79, 2004.
- [17] V. T. Vu, *Groundwater model for study-area Luxemburg*, Report on the Master course “Geo modeling” supervised by W. Bouten, Amsterdam University, 2005.
- [18] G. B. Whitham, *Linear and Nonlinear Waves*, Wiley-Interscience, New York, 1974.
- [19] <http://www-sop.inria.fr/epidaure/personnel/Olivier.Clatz>.
- [20] <http://www.bsu.edu/web/jkshim/locomotion/locomotion.htm>
- [21] <http://en.wikipedia.org/wiki/Reynolds-number>.

APPENDIX

A. APPENDIX A

The physical conservation law expressing conservation of some quantity over a domain Ω with boundary Γ reads (see [13])

$$\frac{\partial}{\partial t} \int_{\Omega} c \, d\Omega = - \int_{\Gamma} F(c) \cdot n \, d\Gamma + \int_{\Omega} R \, d\Omega.$$

The integral over Γ describes the net outflow of the considered quantity (given by the flux function $F(c)$, e.g. $F(c) = uc$ in the model of reacting chemical species), whereas R represents a source term. When the flux function is differentiable, Gauss' divergence theorem can be applied and we obtain

$$\int_{\Omega} \frac{\partial c}{\partial t} \, d\Omega + \int_{\Omega} \nabla \cdot F(c) \, d\Omega = \int_{\Omega} R \, d\Omega.$$

Since this holds for arbitrary small volumes Ω the conservation law can be written as a PDE in divergence form

$$\frac{\partial c}{\partial t} + \nabla \cdot F(c) = R.$$

B. APPENDIX B

The three-dimensional presentations of the terms in Eq. (2.14) read

$$\begin{aligned}
 \mathbf{u}_t &= (u_t, v_t, w_t)^T, \\
 \mathbf{u}\mathbf{u}^T &= \begin{pmatrix} u \\ v \\ w \end{pmatrix} (u, v, w) = \begin{pmatrix} uu & uv & uw \\ vu & vv & vw \\ wu & wv & ww \end{pmatrix}, \\
 \nabla \cdot (\mathbf{u}\mathbf{u}^T) &= \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \cdot \begin{pmatrix} uu & uv & uw \\ vu & vv & vw \\ wu & wv & ww \end{pmatrix} = \begin{pmatrix} (u^2)_x + (uv)_y + (uw)_z \\ (vu)_x + (v^2)_y + (vw)_z \\ (wu)_x + (wv)_y + (w^2)_z \end{pmatrix}, \\
 \nabla p &= (p_x, p_y, p_z)^T, \\
 \nabla \mathbf{u} + (\nabla \mathbf{u})^T &= \begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix} + \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix} = \begin{pmatrix} 2u_x & u_y + v_x & u_z + w_x \\ v_x + u_y & 2v_y & v_z + w_y \\ w_x + u_z & w_y + v_z & 2w_z \end{pmatrix}, \\
 \nabla \cdot (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) &= \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \cdot \begin{pmatrix} 2u_x & u_y + v_x & u_z + w_x \\ v_x + u_y & 2v_y & v_z + w_y \\ w_x + u_z & w_y + v_z & 2w_z \end{pmatrix} \\
 &= \begin{pmatrix} 2u_{xx} + (u_y + v_x)_y + (u_z + w_x)_z \\ (v_x + u_y)_x + 2v_{yy} + (v_z + w_y)_z \\ (w_x + u_z)_x + (w_y + v_z)_y + 2w_{zz} \end{pmatrix}.
 \end{aligned}$$

From Eq. (2.14) and the above relations, we arrive at Eq. (2.15).

C. APPENDIX C

The divergence-free property of \mathbf{u} reads $\nabla \cdot \mathbf{u} = u_x + v_y + w_z = 0$.

Using the product rule

$$(f(x)g(x))_x = (f(x))_x g(x) + f(x)(g(x))_x$$

for the terms in the first equation of (2.15), we have

$$\begin{aligned} u_t + (u^2)_x + (uv)_y + (uw)_z &= u_t + 2uu_x + u_yv + uv_y + u_zw + uw_z \\ &= u_t + uu_x + vu_y + wu_z + u(u_x + v_y + w_z) \\ &= u_t + uu_x + vu_y + wu_z, \\ 2u_{xx} + (u_y + v_x)_y + (u_z + w_x)_z &= 2u_{xx} + u_{yy} + v_{yx} + u_{zz} + w_{zx} \\ &= u_{xx} + u_{yy} + u_{zz} + (u_x + v_y + w_z)_x \\ &= u_{xx} + u_{yy} + u_{zz}. \end{aligned}$$

The first equation of (2.16) is then easily followed. The two remained equations can be verified in the same way.

D. APPENDIX D

The IMEX-RKC scheme obtained by modifying the explicit RKC (3.44). For the first stage, apply the IMEX Euler scheme we have

$$W_1 = W_0 + \tilde{\mu}_1 \tau F_{E,0} + \tilde{\mu}_1 \tau F_{I,1}, \quad (\text{D.1})$$

where $F_{E,0} = F_E(t_n + c_0 \tau, W_0)$ and $F_{I,1} = F_E(t_n + c_1 \tau, W_1)$. The other stages can be obtained in similar manner. To maintain the recursive nature properties derived from the first kind Chebyshev polynomials, we take the function $P_j(z_E, z_I)$, $j = 0, 1, \dots, s$ to be of the form

$$P_j(z_E, z_I) = a_j + b_j T_j \left(\frac{\omega_0 + \omega_1 z_E}{1 - \frac{\omega_1}{\omega_0} z_I} \right), \quad (\text{D.2})$$

where $z_E = \tau \lambda_E$ and $z_I = \tau \lambda_I$, λ_E and λ_I are the eigenvalues of Jacobians $F'_E(t, w(t))$ and $F'_I(t, w(t))$, respectively; a_j and b_j are the same as in (3.48) and (3.49).

From (D.2) we have

$$T_j \left(\frac{\omega_0 + \omega_1 z_E}{1 - \frac{\omega_1}{\omega_0} z_I} \right) = \frac{-a_j}{b_j} + \frac{P_j(z_E, z_I)}{b_j}. \quad (\text{D.3})$$

Substituting (3.47) into (D.2) we have

$$P_j(z_E, z_I) = a_j + b_j \left(2 \frac{\omega_0 + \omega_1 z_E}{1 - \frac{\omega_1}{\omega_0} z_I} T_{j-1} \left(\frac{\omega_0 + \omega_1 z_E}{1 - \frac{\omega_1}{\omega_0} z_I} \right) - T_{j-2} \left(\frac{\omega_0 + \omega_1 z_E}{1 - \frac{\omega_1}{\omega_0} z_I} \right) \right). \quad (\text{D.4})$$

From (D.3) and (D.4) we have

$$P_j(z_E, z_I) = a_j + b_j \left(2 \frac{\omega_0 + \omega_1 z_E}{1 - \frac{\omega_1}{\omega_0} z_I} \left(\frac{-a_{j-1}}{b_{j-1}} + \frac{P_{j-1}(z_E, z_I)}{b_{j-1}} \right) - \left(\frac{-a_{j-2}}{b_{j-2}} + \frac{P_{j-2}(z_E, z_I)}{b_{j-2}} \right) \right), \quad (\text{D.5})$$

or

$$\begin{aligned}
P_j(z_E, z_I) \left(1 - \frac{\omega_1}{\omega_0} z_I\right) &= a_j \left(1 - \frac{\omega_1}{\omega_0} z_I\right) + 2 \frac{b_j}{b_{j-1}} P_{j-1}(z_E, z_I) (\omega_0 + \omega_1 z_E) \\
&\quad - 2 \frac{b_j}{b_{j-1}} a_{j-1} (\omega_0 + \omega_1 z_E) + \frac{b_j}{b_{j-2}} a_{j-2} \left(1 - \frac{\omega_1}{\omega_0} z_I\right) \\
&\quad - \frac{b_j}{b_{j-2}} P_{j-2}(z_E, z_I) \left(1 - \frac{\omega_1}{\omega_0} z_I\right).
\end{aligned} \tag{D.6}$$

From relation (D.6), by substituting $P_j = \frac{W_j}{W_0}$, $\frac{\omega_1}{\omega_0} = b_1 \omega_1 = \tilde{\mu}_1$ and using the coefficient in (3.45) we then have

$$\begin{aligned}
W_j = \tilde{\mu}_1 \tau F_{R,j} &= (a_j - \mu_j a_{j-1} - \nu_j a_{j-2}) W_0 + \mu_j W_{j-1} + \nu_j W_{j-2} + \\
&\quad \tilde{\mu}_j \tau F_{D,j-1} + \tilde{\gamma}_j \tau F_{D,0} - \nu_j \tilde{\mu}_1 \tau F_{R,j-2} - \tilde{\mu}_1 (a_j - \nu_j a_{j-2}) \tau F_{R,0},
\end{aligned} \tag{D.7}$$

where $F_{R,j} = F_R(t_n + c_j \tau, W_j)$ and $F_{D,j} = F_D(t_n + c_j \tau, W_j)$.

Finally, using $a_j - \mu_j a_{j-1} - \nu_j a_{j-2} = 1 - \nu_j - \mu_j$, we get the IMEX RKC formula as follows

$$\begin{aligned}
W_0 &= w_n, \\
W_1 &= W_0 + \tilde{\mu}_1 \tau F_{E,0} + \tilde{\mu}_1 \tau F_{I,1}, \\
W_j &= (1 - \mu_j - \nu_j) W_0 + \mu_j W_{j-1} + \nu_j W_{j-2} + \tilde{\mu}_j \tau F_{E,j-1} + \tilde{\gamma}_j \tau F_{E,0} \\
&\quad + [\tilde{\gamma}_j - (1 - \mu_j - \nu_j) \tilde{\mu}_1] \tau F_{I,0} - \nu_j \tilde{\mu}_1 \tau F_{I,j-2} + \tilde{\mu}_1 \tau F_{I,j}, \quad j = 2, \dots, s, \\
W_{n+1} &= W_s,
\end{aligned} \tag{D.8}$$

where $F_{E,k} = F_E(t_n + c_k \tau, W_k)$ and $F_{I,k} = F_I(t_n + c_k \tau, W_k)$.

E. APPENDIX E

The stability function of IMEX-RKC given in ([15]) as

$$R_s(z_E, z_I) = 1 + \tilde{z} + \frac{1}{2}\tilde{z}^2 + \frac{1}{10}\tilde{z}^3 + \dots, \quad (\text{E.1})$$

where

$$\tilde{z} = \frac{z}{1 - \omega_1 z_I}, \quad \omega_1 = \frac{3}{s^2 - 1} \quad z = z_E + z_I, \quad z_E = \tau_n \lambda_E, \quad z_I = \tau_n \lambda_I. \quad (\text{E.2})$$

Here s is number of stages, τ_n is step size at time t_n , λ_E and λ_I are eigenvalues of explicit and implicit part, respectively.

Expanding (E.1) for small z , we have

$$R_s(z_E, z_I) = 1 + z + \frac{1}{2}z^2 + \omega_1 z z_I + \mathcal{O}(z^3, z^2 z_I, z z_I^2). \quad (\text{E.3})$$

Consider equation $w'(t) = F(t, w(t)) = F_E(t, w(t)) + F_I(t, w(t))$ with smooth solution $w(t)$. Let $w_n = w(t_n)$ and apply (E.3) we have

$$w_{n+1} = w(t_n) + \tau_n w'(t_n) + \frac{1}{2} \tau_n^2 w''(t_n) + \omega_1 \tau_n^2 F_I'(t_n, w(t_n)) F(t_n, w(t_n)) + \omega_1 \tau_n^2 \frac{\partial F_I(t_n, w(t_n))}{\partial t} + \mathcal{O}(\tau_n^3). \quad (\text{E.4})$$

The local truncation error is of the form

$$Est_{n+1} = \frac{1}{2} \tau_n^2 w''(t_n) + \omega_1 \tau_n^2 F_I'(t_n, w(t_n)) F(t_n, w(t_n)) + \omega_1 \tau_n^2 \frac{\partial F_I(t_n, w(t_n))}{\partial t}. \quad (\text{E.5})$$

It is easy to prove that

$$w''(t_n) = \frac{w'(t_{n+1}) - w'(t_n)}{\tau_n} = \frac{F(t_{n+1}, w_{n+1}) - F(t_n, w_n)}{\tau_n}, \quad (\text{E.6})$$

and

$$F_I'(t_n, w(t_n)) F(t_n, w(t_n)) + \frac{\partial F_I(t_n, w(t_n))}{\partial t} = \frac{dF_I(t_n, w(t_n))}{dt} = \frac{F_I(t_{n+1}, w_{n+1}) - F_I(t_n, w_n)}{\tau_n}. \quad (\text{E.7})$$

Substitute (E.6) and (E.7) to (E.5) and use $\omega_1 = \frac{3}{s^2 - 1}$ we obtain the local error estimation for IMEX-RKC method

$$Est_{n+1} = \frac{1}{2} \tau_n (F(t_{n+1}, w_{n+1}) - F(t_n, w_n)) + \frac{3}{s^2 - 1} \tau_n (F_I(t_{n+1}, w_{n+1}) - F_I(t_n, w_n)). \quad (\text{E.8})$$