Computer Algebra in Simulation Code Development

Richard Liska
Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague
Břehová 7, 115 19 Prague 1, Czechoslovakia
tjerl@cspuni12.bitnet, tjerl@aci.cvut.cs

Abstract

The majority of large-scale scientific computation problems fall into the area of numerical simulations of real world systems. Such systems are usually described by a mathematical model in the form of a system of partial differential equations (PDEs) which are solvable only numerically. The productivity of particular numerical PDEs solver development can be increased by using special program tools. Such tools are usually implemented in a computer algebra system. The FIDE package includes several tools performing such tasks as PDEs discretization, analysis of difference schemes and generation of numerical code for difference schemes solving. Capability of using the package for a real hard simulation problem is illustrated. Possibilities to enlarge the abilities of such tools and to incorporate artificial intelligence methods in the presented methodology are discussed.

Introduction

The greatest part of scientific computation is covered by simulation problems. In the simulation problem we have a real world system and want to know how this system will behave under certain conditions. The system is usually described by the set of equations, the so called mathematical model, in which characteristic quantities of the system appear. The mathematical model of a real world problem cannot be usually solved directly analytically and has to be solved numerically. We will restrict only to numerical simulation problems. The majority of large-scale numerical simulation problems is described by the mathematical model in the form of a system of partial differential equations (PDEs) or integro-differential equations. So we will further restrict to such mathematical models.

There exist several general numerical PDEs solvers. However the experience shows that such solvers are not capable to solve the real world problems. Even more any new real world PDEs problem requires developing of its own numerical solver. The construction of general PDEs solvers suitable for broad classes of equations is not possible for real world problems. Thus the development of simulation code for the specific real problem has a crucial importance for scientific computation.

It is necessary to stress out that our aim is not to construct a black box PDEs solvers development system which should result in limited class of solvable problems. Rather we want to design tools which can help and assist a user during the solver development. The user is expected to understand and control the development process. The control remains in user's hands while a computer performs mechanical tasks necessary during the development. The other interesting approach can be a "grey box" PDEs solvers development system which can be closed for beginners in PDEs solving and open for experts in PDEs solving. Such system working in the closed (black box) mode could be controlled by the decision strategy implemented by artificial intelligence methods and containing a large amount of knowledge about numerical PDEs solving.

Very simply the numerical simulation can be divided into four tasks:

1. designing the mathematical model of a real world problem
2. developing a numerical code solving the mathematical model
3. numerical testing of the code
4. actual calculation and results post-processing

These tasks are not performed purely sequentially. Finding an inappropriate behavior of a task one usually returns to one of the previous tasks and modifies it. We will consider here the second task, which on the base of a mathematical model (here PDEs) develops a numerical algorithm solving the model and implements the algorithm in a numerical program. The questions of grid generation and grid adjusting belonging to this task do not fall into the scope of this paper. During the algorithm and code development a considerable amount of analytical calculations has to be done. This analytical calculations can be performed by a computer algebra system (CAS). The paper concentrates mainly on this aspect of using computer algebra during designing an algorithm and a numerical code for PDEs solving. This methodology increases the productivity of the development process. The productivity boost is more significant if a mathematical model or the geometry of a problem is complicated.

From the artificial intelligence point of view CASs are knowledge systems intercepting a large amount of mathematical knowledge. When the knowledge of numerical PDEs solving is added the resulted program tools can save time and effort of a developer.

The paper concentrates on description of the FIDE package implemented in the REDUCE (Hearn 1991) computer algebra system to assist users during development of numerical code for solving PDEs. The numerical programming language FORTRAN is used in the methodology presented. The package uses the finite difference method for PDEs solving. The FIDE package consists of the following modules that have been built, tested, and documented:

EXPRES for transforming PDES into any orthogonal coordinate system.

IIMET for discretization of PDEs by the integro-interpolation method.

GRIDOP for discretization of PDEs by the support-operators method.

APPROX for determining the truncation error of difference schemes.

CHARPOL for calculating the amplification matrix and characteristic polynomial of difference schemes, which are needed in the Fourier stability analysis.

HURWP for locating polynomial roots necessary in verifying the von Neumann stability condition.

LINBAND for generating the block of FORTRAN code, which solves a system of linear algebraic equations with band matrix appearing frequently in difference schemes.

The FIDE package has been included into the REDUCE network library available on <reduce-netlib@rand.org> where the full code, documentation and test files can be found. For more detailed description of the package see (Liska & Drska 1990).

Discretization

Discretization is the transformation of PDEs into a system of algebraic difference equations, so called difference scheme. The discretization can become a tedious task especially in cases of complicated PDEs, complicated geometry and higher dimensions. For example in 2D problem on rectangle one has to treat at least 9 difference schemes, one inside, 4 on the different boundary lines and 4 on the boundary corners. In 3D the number of different difference schemes on cube becomes even 27. The amount of analytical transformations involved in such tasks is really large and well suited for treating by CAS.

Integro-Interpolation Method

The FIDE package includes the IIMET module which performs the discretization of PDEs on orthogonal grids by integro-interpolation method. This method was chosen because it is quite universal, it can be applied to nonlinear PDEs, and in linear cases it produces conservative difference schemes. In every coordinate the second grid with grid points halfway between the original ones is defined. The basic principle of the integro-interpolation method is in integrating the given PDEs over the whole cell of a grid. If direct integration is not possible, some kind of interpolation is used. Every quantity appearing in a PDEs can be discretized in every coordinate on the original grid or on the second one. Every equation of the system can be integrated over the cell of the original grid or of the second one. The decision on which grids the individual quantities (looked for functions of coordinates) are to be defined and over which grid cells the individual equations are to be integrated is made so as to minimize the number of interpolations used during integrating the whole PDEs. Such a decision is really not easy, especially for large systems.

The IIMET module for discretization by the integro-interpolation method accomplishes this decision by either heuristically driven, or full search and from input PDEs the module produces the system of difference equations. The discretization of individual basic types of terms appearing in PDEs can be defined, and the discretization can be controlled in several manners, so that almost any discretization, according to the user's wish, can be achieved by this module. Input PDEs can also include vector or tensor equations, in which case these equations are converted into scalar ones by the EXPRES module before the discretization begins.

Support-Operators Method

The support-operators method (Samarski et al. 1981) is a special method for PDEs discretization on non-orthogonal meshes. According to this method the difference scheme for a system of differential equations, which are written in the terms of first order invariant differential operators, is built so that the difference analogues of chosen integral relations between the operators hold. One of these relations for the first order invariant differential operators div, grad is

$$\int_V u \operatorname{div} \vec{W} \, \mathrm{d}\, V + \int_V (\vec{W}, \operatorname{grad} u) \, \mathrm{d}\, V = \oint_{\partial V} u(\vec{W}, \vec{n}) \, \mathrm{d}\, S. \tag{1}$$

Considering a problem with zero boundary condition on the surface ∂V we can from (1) derive that grad $= -\mathrm{div}^*$ where the star denotes the adjoint operator with respect to the functional scalar product $(f, g) = \int_V fg \, \mathrm{d}\, V$. Having a suitable defined difference operator DIV and the difference approximation of the integral relation (1) the support-operators method construct a difference operator GRAD namely from the operator DIV and the approximation.

The method can be automated on logically rectangular meshes. The formalization of the method, which is necessary for algorithms development, is based on the concept of grid function spaces. The scalar or vector grid function is a function defined in the grid nodes (or cells) and has unique symbolic values (depending on multi-index which corresponds to the node) on logically rectangular subsets of the grid. Grid operators are linear operators acting from one grid function space to another. The grid operator has again unique values on logically rectangular subsets of the grid. On every grid function space A two scalar products are defined. The natural scalar product $(u, v)_A$ is a difference approximation of the functional scalar product. The formal scalar product $[u, v]_A$ is the sum of products of the grid functions values over all nodes (or cells) in the grid. The natural

$O^\star$ and formal $O^\otimes$ adjoint operators to the grid operator O which works from the space A of its arguments to the space V of its values are defined by

$$(u, Ov)_V = (O^\star u, v)_A, \; [u, Ov]_V = [O^\otimes u, v]_A, \; \forall u, v.$$

The grid operator P_A corresponding to the natural scalar product on the grid function space A is defined by

$$(u, v)_A = [u, P_A v]_A.$$

The natural adjoint grid operator $O^\star$, necessary for the construction of difference operators which approximate the differential ones, can then be expressed as

$$O^\star = P_V^{-1}(P_A O)^\otimes.$$

The non-zero boundary conditions can also be incorporated into this formalization.

The support-operator method has been implemented in the GRIDOP module. For more detailed description of the used algorithms see (Liska & Shashkov 1991; Liska, Shashkov , & Solovjov 1992)

Analysis of Difference Scheme

Having obtained the difference scheme one needs to investigate its properties which are very important for numerical behaving of code derived from the difference scheme and which also describe the relation between the solution of difference scheme and PDEs. The following subsections are devoted to investigating the properties of difference schemes.

Truncation error

Truncation error determines the order of accuracy of the difference scheme in powers of grid steps. Truncation error analysis is performed by expanding all discrete function values in the difference scheme into the Taylor series and collecting terms as polynomial in grid steps Δ. For the truncation error analysis it is necessary to distinguish on which type of grid we are working. We restrict again to logically rectangular grids. Orthogonal grids can be uniform with constant grid step Δx or non-uniform with grid step Δx_i. For non-orthogonal grids we have grid steps e.g. $\Delta x_{ij}, \Delta y_{ij}$ and we are using substitution

$$\Delta x_{ij} = \delta \alpha_{ij}$$
$$\Delta y_{ij} = \delta \beta_{ij}$$

where δ is small and α_{ij}, β_{ij} are parameters. Then the truncation error is determined in power of δ. We have to distinguish adaptive grids and other special cases too. The truncation error analysis is accomplished by the APPROX module.

Stability

Stability is one of the most important properties of difference scheme. If the difference scheme is unstable the rounding errors are amplified during numerical solving of the scheme which results in increasing artificial oscillations in the numerical solution.

The Fourier method is probably the most common and simplest method for local stability analysis. The method after Fourier expansions in non-time coordinates calculates the amplification matrix of the difference scheme for transition from one time layer to another one. The von Neumann stability condition then states that all eigenvalues of the amplification matrix, i.e. roots of its characteristic polynomial, must lie in the unit circle in the complex plane for all Fourier components. Calculation of the amplification matrix and its characteristic polynomial is performed by the module CHARPOL.

By conformal mapping the problem of locating roots of polynomial in the unit circle can be transformed to locating roots of polynomial in left half plane (with negative real parts), i.e. to the Routh-Hurwitz problem (Ganzha & Vorozhtsov 1987). The Routh-Hurwitz problem can be solved by several methods one of which is Hurwitz method utilizing the Lienard-Chipart stability criterion (for other method see (Ganzha & Liska 1989)). As a result, from the characteristic polynomial of the amplification matrix of the difference scheme one obtains several inequalities connected by logical operators AND and OR, which have to hold for the scheme to be stable. The inequalities have to hold for all Fourier components in all non-time coordinates. This method is implemented in the HURWP module and results in the system of generally quantified inequalities. For quantifier elimination from these inequalities the cylindrical algebraic decomposition (CAD) algorithm can be used (Steinberg 1991). The first tests using the SACLIB implementation of partial CAD algorithm (Collins & Hong 1991; Hong 1992) for stability analysis by this method has been done.

Code Generation

After discretization and the analysis of the properties of the difference scheme the next task is to develop the numerical code for solving the system of algebraic equations which formed the difference scheme. The system of algebraic equations has special form. Its matrix (or Jacobian in non-linear case) is usually the band matrix or block band matrix for which special solution algorithms exists. Sometimes, e.g. in altered direction implicit (ADI) method, the system is constructed so that it has some special features allowing to use fast techniques for the solution. For band matrices the fast band matrix solvers exists for other systems the most promising solvers are precondition conjugate gradient solvers or algebraic multigrid solvers. Anytime the best is to use the verified routines from numerical libraries.

Code generation is the process during which CAS creates a text file containing a source numerical program. Having the difference scheme in forms of formulas in CAS we can analyze the scheme extracting the matrix (or Jacobian) of the system of algebraic equations and generate the code which prepares the input parameters for the solver routine, calls the routine and extract the output parameters to our data structures.

The LINBAND module has been designed for automatic generation of FORTRAN code for solving system of linear algebraic equations with band matrices which appear quite often in difference schemes. The program is based on GENTRAN (Gates 1986) REDUCE pack-

age for code generation and translation. The program performs all the above tasks and can be used in cooperation with numerical libraries ESSL (IBM), LINPACK, and NAG, for which it prepares the band matrices in arrays in special economical forms used by these libraries and it uses their linear solvers for band matrix systems. The tridiagonal systems are taken as a special case using more effective tridiagonal routines. The use of band matrices can be extended from 1D difference schemes to more dimensions by using ADI method which splits the difference scheme into several substeps in each of which one direction remain implicit. We need to solve a set of linear band systems in each time substep. The code generation for difference schemes solving is described in more detail in (Liska 1992).

Application

As an application of the FIDE package to a real world problem we present here the numerical solution of the Fokker-Planck equation (FPE). Via the distribution function $f(t,\mathbf{z},\mathbf{v})$ on the position and velocity space, the FPE describes a system of electrically charged particles interacting through Coulomb forces. We use it for a very detailed description of electrons in plasmas heated by high intensity laser beams. The method of solution is taken from (Epperlein, Rickard, & Bell 1988).

The distribution function is replaced by its diffusive approximation $f(t,\mathbf{z},\mathbf{v}) = f_0(t,\mathbf{z},\ v) + \mathbf{v}/v \cdot \mathbf{f}_1(t,\mathbf{z},v)$, for whose components f_0, f_1 the FPE in one dimension can be written as

$$\begin{aligned} \frac{\partial f_0}{\partial t} + \frac{v}{3}\frac{\partial f_1}{\partial z} - \frac{a}{3v^2}\frac{\partial}{\partial v}\left(v^2 f_1\right) &= \frac{Y_{ee}}{v^2}\frac{\partial}{\partial v}\left(K^0 f_0 + \right. \\ &\qquad \left. K^1 \frac{\partial f_0}{\partial v}\right), \qquad (2) \\ v\frac{\partial f_0}{\partial z} - a\frac{\partial f_0}{\partial v} &= -\frac{n_i Y_{ei}}{v^3} f_1 , \end{aligned}$$

where

$$\begin{aligned} K^0 &= 4\pi \int_0^v f_0(u)\, u^2\, du , \\ K^1 &= \frac{4\pi}{3}\left(\frac{1}{v}\int_0^v f_0(u)\, u^4\, du + v^2 \int_v^\infty f_0(u)\, u\, du\right) \\ &\quad + \frac{n_i}{6}\frac{Y_{ei}}{Y_{ee}}\left(\frac{eE_0}{m_e}\right)^2 \frac{1}{v(\omega^2 + (n_i Y_{ei}/v^3)^2)} \end{aligned}$$

and $e, m_e, \omega, a(t,z), Y_{ee}(t,z), Y_{ei}(t,z), n_i(t,z),\ E_0(t,z)$ are physical constants or quantities, which need not be defined here.

Equations (2) have been discretized by the IIMET and/or GRIDOP module by using the altered direction implicit method. In the first step of the ADI method from the time layer n to $n+1$ the spatial derivatives are taken on the implicit layer $n+1$ and the velocity derivatives on the explicit time layer n. In the interior grid point the following scheme is obtained

$$\begin{aligned} &\frac{f_{jk}^{0,n+1} - f_{jk}^{0n}}{h^t} + \frac{v_j}{3}\frac{f_{j,k+1/2}^{1,n+1} - f_{j,k-1/2}^{1,n+1}}{h_k^z} - \\ &\frac{a_k}{3v_j^2}\frac{f_{j+1/2,k}^{1n} v_{j+1/2}^2 - f_{j-1/2,k}^{1n} v_{j-1/2}^2}{h_j^v} = \\ &\frac{Y_k^{ee}}{v_j^2}\left(\frac{K_{j+1/2,k}^0 f_{j+1/2,k}^{0n} - K_{j-1/2,k}^0 f_{j-1/2,k}^{0n}}{h_j^v} + \right. \\ &\frac{1}{h_j^v}\left(K_{j+1/2,k}^1 \frac{f_{j+1,k}^{0n} - f_{jk}^{0n}}{1/2\left(h_{j+1}^v + h_j^v\right)} - \right. \qquad (3) \\ &\left.\left. K_{j-1/2,k}^1 \frac{f_{jk}^{0n} - f_{j-1,k}^{0n}}{1/2\left(h_{j-1}^v + h_j^v\right)}\right)\right) \\ &v_j \frac{f_{j,k+1}^{0,n+1} - f_{jk}^{0,n+1}}{1/2\left(h_{k+1}^z + h_k^z\right)} - \\ &a_{k+1/2}\frac{f_{j+1,k+1/2}^{0n} - f_{j,k+1/2}^{0n}}{1/2\left(h_{j+1}^v + h_j^v\right)} = -\frac{Y_k^{ei} n_{k+1/2}^i}{v_j^3} f_{j,k+1/2}^{1,n+1} \end{aligned}$$

where h^t, h_k^z and h_j^v are grid steps in time, space and velocity coordinates. In the second step of the ADI method from the time layer $n+1$ to $n+2$ the spatial derivatives are taken on the explicit time layer $n+1$ and the velocity derivatives on the implicit time layer $n+2$. Discretization on velocity borders and numerical evaluation of integrals in terms K^0, K^1 is performed in a special way ensuring conservation of the number of particles and particles energy. By discretizing equations (2) also on all boundaries and boundary crossings we obtain the total number of 18 difference schemes similar to (3) for both steps of the ADI method.

The scheme has been analyzed in the interior point. The APPROX module determines the order of its approximation $O(h^t + (h^z)^2 + (h^v)^2)$ in both steps of the ADI method. After reordering, all the 18 difference schemes make up several systems of linear algebraic equations with band matrices. For their solution, blocks of FORTRAN code have been generated by the LINBAND module. Also some other parts of the numerical code, as e.g. integral evaluation, have been generated by GENTRAN. Up to 2000 lines of FORTRAN code, what is more than 2/3 of the whole numerical program for FPE solving, has been generated automatically.

The developed numerical code for solving Fokker-Planck equation has been used for simulation of ultrashort laser pulses interaction with plasma. Several variants of the code in plane and cylindrical geometries with additional terms in equations has been prepared. New physical results has been obtained with the code (Drska, Limpouch, & Liska 1991; Drska, Limpouch, & Liska 1992). The experience with the code development proves the advantages of presented methodology which are summarized in the conclusion.

Other Possibilities

In this section we will consider other possibilities of applying computer algebra in the process of developing numerical algorithms and codes for solving PDEs. We concentrate here again on the finite difference method for PDEs solving. Of course variational methods as the finite element method or other methods can be used too.

For discretization of PDEs also other methods are

suitable for implementation in CAS. As an example the boundary fitted coordinates method (Steinberg & Roache 1991) can be mentioned. This method is based on transforming a non-orthogonal logically rectangular mesh to a rectangular one on which the discretization is performed.

More research should be done on the quantifier elimination from inequalities which can be applied in stability analysis as shown above. Present software for the elimination is extremely time consuming or needs very qualified, careful and time consuming human support. But our experience in this area is still insufficient.

For the stability analysis also other methods could be employed. Important is the stability on boundaries because quite often boundaries are sources of instabilities, for an example of CAS application in this area see (Mazepa 1989). Powerful stability analysis method is the energy method. Although this method probably cannot be fully automatized, tools can be built to assist user who performs stability analysis by this method. Also other important properties of difference schemes as symmetry or conservation could be checked by CAS tools.

For the code generation step the code generation facilities for other types of difference schemes (general linear schemes and non-linear schemes) should be designed. As is stated already above the best approach is to use verified numerical routines from numerical libraries in the generated code for actual solving of the schemes. To any solver routine its precise formal description can be associated (Dewar & Richardson 1990; Cook 90). The formal description contains the information about what task the routine performs, meaning of the routine parameters and the way in which data should be stored in the parameters (e.g. storing the band matrix in an array). The general tool for code generation for solving difference schemes can be constructed which according to the formal description of the routines can choose the most suitable routine and generate the code using this routine. This approach has the advantage that new solver routines can be included without changing the tool. In artificial intelligence terminology this formalism can be described as reasoning with frame knowledge representation. Using CAS means that during the reasoning we can use analytical conditions and formulas analysis.

The generated code should have the minimal number of arithmetic operations (code optimization problem), see e.g. (Hulzen et al. 1989) and also should minimize the floating-point errors appearing during the calculation, see e.g. (Molenkamp, Goldman, & Hulzen 1991).

Knowledge of PDEs Code Development

Knowledge about numerical solving of PDEs is spread in a large amount of books, articles, programs and experts. The task of extracting and organizing this knowledge is really immense. According to different phases of the development process one can distinguish the special knowledge concerning

grid generation – how the grid should be build on given region for given problem; how the grid should be changed during calculation.

numerical algorithms – which algorithm is suitable for given problem; how the algorithm should be changed if it has unsuitable properties.

algorithm analysis – methods of investigating properties of numerical algorithms as truncation error, stability and conservation.

code generation – how utilize existing numerical libraries; how generate code for new architecture machines.

Considering organization and utilization of this knowledge the following questions have to be investigated

aim – identify which aims the knowledge should address.

knowledge formalization – how the knowledge should be represented; which types of knowledge can appear.

knowledge acquisition – locate knowledge sources; which methods of knowledge acquisition should be used.

inference methods – which reasoning methods should be applied to the knowledge; combining reasoning and procedural methods.

knowledge system – identify which capabilities should the perspective knowledge system posses to be suitable for use (numeric, symbolic and algebraic computation; access to numerical libraries; graphical facilities; knowledge representation; reasoning facilities)

Let us mention the relation between procedural and declarative knowledge in CAS and in the presented methodology of PDEs solvers development. In CAS the majority of knowledge is implemented as a procedural one. The same is true in the package FIDE which supports the developments of PDEs solvers. The possibility to use declarative knowledge for the description of the numerical algebraic equations systems solvers is discussed in the end of the previous section. It is very likely that a more advanced and capable system for supporting the development of PDEs solvers will need a clever combination of procedural and declarative knowledge processing. Similarly the combination of different types of knowledge representation and reasoning mechanisms will be necessary.

Conclusion

Using of computer algebra systems in the process of numerical simulation codes development brings several advantages

- increased speed of development
- decreased probability of bugs
- rapid and accurate modifications of existing programs

The first two advantages are related to the fact that using this methodology the developer is working with higher constructs than are arrays of numbers and loops. She or he can more concentrate on the solved problem while mechanical tasks are performed by a computer. The last advantage expresses the experience that the

problem or method of its solution can be quite easily modified, e.g. by adding some previously neglected term to PDEs or choosing different difference scheme on a boundary. It seems that for high dimensions and complicated geometries the simulation code development without the assisting program tools outlined here is almost impossible.

Acknowledgments

This work has been partially supported within the framework of the IBM Academic Initiative in Czechoslovakia and by the Czech Technical University grant No. 906.

References

[Collins and Hong 1991] Collins, G. E. and Hong, H. 1991. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symb. Comp.* 12(3):299–328.

[Cook 1990] Cook, G. O. Jr. 1990. ALPAL: a program to generate physics simulation codes from natural descriptions. *Int. J. Mod. Phys. C* 1(1):1–51.

[Dewar and Richardson 1990] Dewar:90 Dewar, M. C. and Richardson, M. G. 1990. Reconciling symbolic and numeric computation in a practical setting. In Miola, A., editor, *Design and Implementation of Symbolic Computation Systems, DISCO'90*, Berlin. Springer-Verlag. 195–204. LNCS 429.

[Drska *et al.*1991] Drska, L.; Limpouch, J.; and Liska, R. 1991. Simulation study of laser-matter interaction in subpicosecond range. In Tsintsadze, N. L., editor, *10-th European Summer School on Plasma Physics, Tbilisi, USSR, 1990*, Singapore. World Scientific. 435–440.

[Drska *et al.*1992] Drska, L.; Limpouch, J.; and Liska, R. 1992. Fokker-planck simulations of ultrashort-pulse laser-plasma interactions. *Laser and Particle Beams.* (In press).

[Epperlein *et al.*1988] Epperlein, E. M.; Rickard, G. J.; and Bell, A. R. 1988. A code for the solution of the Vlasov-Fokker-Planck equation in 1-D or 2-D. *Comp. Phys. Comm.* 52:7–13.

[Ganzha and Liska 1989] Ganzha, V. and Liska, R. 1989. Application of the REDUCE computer algebra system to stability analysis of difference schemes. In Kaltofen, E. and Watt, S. M., editors, *Computers and Mathematics '89, MIT*, New York. Springer-Verlag. 119–129.

[Ganzha and Vorozhtsov 1987] Ganzha, V. G. and Vorozhtsov, E. V. 1987. The stability analysis of difference schemes by numerical solution of the generalized Routh-Hurwitz problem. *Comp. Phys. Comm.* 43:209–216.

[Gates 1986] Gates, B. L. 1986. A numerical code generation facility for REDUCE. In Char, B. W., editor, *SYMSAC '86*, Waterloo. ACM Press. 94–99.

[Hearn 1991] Hearn, A. C. 1991. REDUCE user's manual, version 3.4. Technical Report, CP 78 (Rev. 7/91), The RAND Corporation, Santa Monica.

[Hong 1992] Hong, H. 1992. Simple solution formula construction in cylindrical algebraic decomposition based quantifier elimination. In Wang, P. S., editor, *ISSAC'92, International Symposium on Symbolic and Algebraic Computation, Berkeley, California.* ACM Press. 177–188.

[van Hulzen *et al.*1989] Hulzen, J. A. van; Hulshof, B. J. A.; Gates, B. L.; and Heerwaarden, M. C. van 1989. A code optimization package for REDUCE. In Gonnet, G., editor, *ISSAC'89, International Symposium on Symbolic and Algebraic Computation, Portland*, New York. ACM. 163–170.

[Liska and Drska 1990] Liska, R. and Drska, L. 1990. FIDE: A REDUCE package for automation of FInite difference method for solving pDE:. In Watanabe, S. and Nagata, M., editors, *ISSAC '90, Proceedings of the International Symposium on Symbolic and Algebraic Computation, Tokyo*, New York. ACM Press, Addison Wesley. 169–176.

[Liska and Shashkov 1991] Liska, R. and Shashkov, M. Yu. 1991. Algorithms for difference schemes construction on non-orthogonal logically rectangular meshes. In Watt, S. M., editor, *ISSAC'91, International Symposium on Symbolic and Algebraic Computation, Bonn*, New York. ACM Press. 419–426.

[Liska *et al.*1992a] Liska, R.; Shashkov, M. Yu.; and Solovjov, A.V. 1992. Support-operators method for PDE discretization: Symbolic algorithms and realization. In Richter, G., editor, *IMACS PDE-7, 7th IMACS International Conference on Computer Methods for Partial Differential Equations, New Brunswick*, New Brunswick. IMACS. (In press).

[Liska 1992] Liska, R. 1992. Numerical code generation for finite difference schemes solving. In Brezinski, C. and Kulisch, U., editors, *Computational and Applied Mathematics I - Algorithms and Theory*, Amsterdam. Elsevier. Proceedings of the 13th IMACS World Congress, Dublin, Ireland, 1991 and IMACS Conference on Modelling and Control of Technological Systems, Lille, France, 1991 (In press).

[Mazepa 1989] Mazepa, N. E. 1989. Vycislenije spektrov raznostnych krajevych zadac s primeneniem SAV. Technical Report P11-89-382, SUJV, Dubna.

[Molenkamp *et al.*1991] Molenkamp, J. H. J.; Goldman, V. V.; and Hulzen, J. A. van 1991. An improved approach to automatic error cumulation control. In Watt, S. M., editor, *ISSAC'91, International Symposium on Symbolic and Algebraic Computation, Bonn*, New York. ACM Press. 414–418.

[Samarskii *et al.*1981] Samarskii, A. A.; Tishkin, V. F.; Favorskii, A. P.; and Shashkov, M. Yu. 1981. Operational finite-difference schemes. *Differential equations* 17:863–885.

[Steinberg 1991] Steinberg, S. 1991. private communication.

[Steinberg and Roache 1991] Steinberg, S. and Roache, P. J. 1991. Discretizing symmetric operators in general coordinates. Technical report, University of New Mexico, Albuquerque.