17 Mathcad[1] Worksheets for Partial Differential Algebraic Equations

Address:	Prof. Dr. K. Schittkowski Department of Mathematics University of Bayreuth D - 95440 Bayreuth
Phone:	+921 553278 (office) +921 32887 (home)
Fax:	+921 35557
E-mail:	klaus.schittkowski@uni-bayreuth.de
Web:	http://www.klaus-schittkowski.de
Date:	April 8, 2004

Abstract

The purpose of the paper is to introduce a set of Mathcad worksheets containing partial differential algebraic equations (PDAEs). They can be used to become familiar with Mathcad implementation of PDAEs and with the behavior of dynamical systems in general. The problems are taken from a collection of test examples for data fitting in dynamical systems, see Schittkowski [24]. The report contains a summary of 17 partial differential algebraic equations that have been transferred to Mathcad and a detailed example. All worksheets can be downloaded from the home page of the author[2]. A particular advantage of executing these problems from Mathcad is the possibility to plot corresponding solutions very easily.

[1] ©2003 Mathsoft Engineering & Education Inc.

[2] http://www.klaus-schittkowski.de

1 Introduction

Partial differential algebraic equations (PDAEs) are based on the same model structure as one-dimensional, time-dependent partial differential equations. The only difference is that additional algebraic equations are permitted as in the case of DAEs. Typical examples are higher order partial differential equations, for example

$$u_t = f(u, u_{xxxx}, x, t) \ ,$$

transformed into a second-order equation by introducing an additional variable w,

$$\begin{aligned} u_t &= f(u, w_{xx}, x, t) \ , \\ 0 &= w - u_{xx} \ . \end{aligned}$$

Also, distributed parameter systems of the form

$$\begin{aligned} u_t &= f_1(u, v, x, t) \ , \\ v_x &= f_2(u, v, x, t) \end{aligned}$$

with initial values $u(x, 0) = u_0(x)$, $v(0, t) = v_0(t)$ are transformed into the PDAEs

$$\begin{aligned} u_t &= f_1(u, v, x, t) \ , \\ 0 &= v_x - f_2(u, v, x, t) \end{aligned}$$

or

$$\begin{aligned} v_x &= f_2(u, v, x, t) \ , \\ 0 &= u_x - f_1(u, v, x, t) \ , \end{aligned}$$

respectively.

Thus, we proceed again from a spatial integration interval $[x_L, x_R]$, a time interval $[0, T]$, and consider the general system

$$\begin{aligned} u_{dt} &= F_d(u, u_x, u_{xx}, x, t) \ , \\ 0 &= F_a(u, u_x, u_{xx}, x, t) \ , \end{aligned} \tag{1}$$

where $x \in I\!R$ is the spatial variable with $x_L \leq x \leq x_R$, and $t \in I\!R$ the time variable with $0 < t \leq T$. The n_p state variables u are divided into n_d differential variables and n_a algebraic variables, $u = (u_d, u_a)^T$, where the number of algebraic variables is identical to the number of algebraic equations defined by the vector-valued function F_a and $n_p = n_d + n_a$.

Initial values are given for all variables,

$$u(x,0) = u_0(x) \quad , \tag{2}$$

where we assume again that the integration is to be performed over a time interval of the form $[0,T]$ with a sufficiently large bound T. $u_0(x)$ is a given function defined on the integration area $[x_L, x_R]$.

As in case of time-dependent differential equations only, classical boundary conditions are either of Dirichlet type

$$\begin{aligned} u(x_L,t) &= u^L(t) \quad , \\ u(x_R,t) &= u^R(t) \quad , \end{aligned} \tag{3}$$

with given functions $u^L(t)$ and $u^R(t)$ defined on the whole time interval $(0,T]$, or of Neumann type

$$\begin{aligned} u_x(x_L,t) &= \hat{u}^L(t) \quad , \\ u_x(x_R,t) &= \hat{u}^R(t) \quad , \end{aligned} \tag{4}$$

where derivative or flux functions $\hat{u}^L(t)$ and $\hat{u}^L(t)$ are known for all $t \in (0,T]$. We have the option to fix either the function values of the solution or its spatial derivatives, to mix both conditions, or to omit one or another boundary condition completely.

The number and order of boundary conditions depends on the model structure, for example whether F depends on u_{xx} or not, in order to get a uniquely defined solution. As a rule of thumb, the total number of scalar boundary conditions should coincide with the number of all partial differentiations, where the highest order of spatial differentiation is counted for each state variable. Boundary and initial conditions are called inconsistent, if they do not fit continuously at connecting points, for instance if $u^L(0) \neq u_0(x_L)$ in case of a left Dirichlet boundary condition.

However, we must treat initial values with more care, since the situation is a bit more complex than in the case of ordinary differential algebraic equations. We have to guarantee that they satisfy the n_a algebraic equations

$$0 = F_a(u(x,0), u_x(x,0), u_{xx}(x,0), x, 0) \tag{5}$$

for all x. But if the right-hand side depends also on some components of u_x or u_{xx}, respectively, it is in general not possible to eliminate the algebraic variables and to compute consistent initial values *by hand*.

There are some attempts to define the index of partial differential algebraic equations in a way similar to ordinary differential equations. However, an exact definition for general nonlinear equations is somewhat cumbersome, see for example Martinson and Barton [13]. On the other hand, indices for linear or semilinear systems are discussed in several papers, see Campbell and Marszalek [2] or Lucht and Strehmel [12].

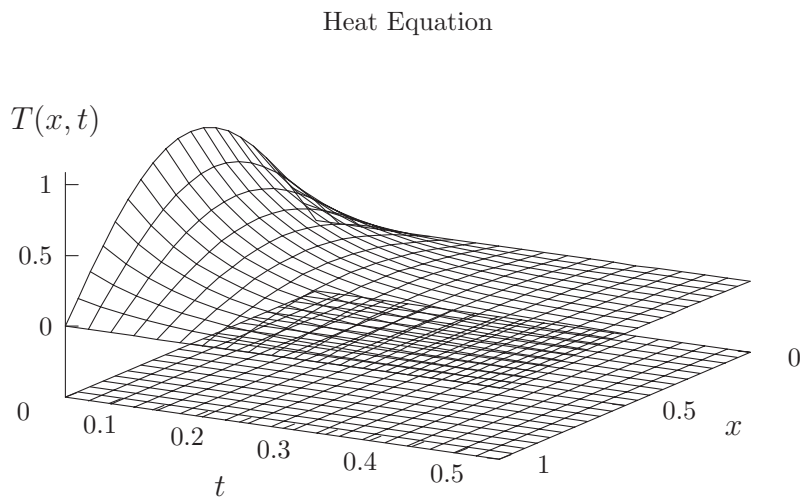

Figure 1: Solution of Heat Equation

Example 1.1 *We consider again the heat equation also discussed in Schittkowski [30], now formulated in the form of a system of two first-order differential algebraic equations*

$$\begin{aligned} T_t &= D\, S_x \ , \\ 0 &= T_x - S \end{aligned} \tag{6}$$

with a thermal diffusion constant $D > 0$. The spatial variable x varies from 0 to L, and for the time variable we require $t > 0$. Initial heat distribution is $T(x, 0) = \sin(x\pi/L)$ for all $x \in (0, L)$, and Dirichlet boundary conditions $T(0, t) = T(L, t) = 0$ are given for all $t \geq 0$. We know that

$$T(x, t) = e^{-t\pi^2/L^2} \sin(x\pi/L)$$

is a solution of the parabolic PDE. If we set $L = 1$ and $D = 1$, we get the heat distribution shown in Figure 1.

The report is one out of a series of Mathcad test problem collections by which numerical routines are tested and the implementation of optimization problems and dynamical systems is outlined, i.e.,

1. nonlinear programming [26],
2. data fitting [27],
3. ordinary differential equations [28],

4. differential algebraic equations [29],

5. partial differential equations [30],

Section 2 contains a brief outline of the method of lines by which a PDAE is transformed into a system of ordinary differential algebraic equations (DAEs). Subsequently, an implicit integration routine is used to integrate these equations. This basic idea is implemented in the routine *Pdesolve* in Mathcad, which is used for all test cases. A simple example is shown in Section 3 to illustrate the numerical solution of a partial differential equation. A list of the Mathcad worksheet files and some further details about problem structure, background, and source is given in Section 4.

2 The Method of Lines

The underlying idea is to transform the partial differential equations into a system of ordinary differential equations by discretizing the model functions with respect to the spatial variable x. This approach is known as the *method of lines*, see for example Schiesser [19] or Schittkowski [24]. The method of lines is also briefly outlined in a related report about Mathcad worksheets for partial differential equations, see Schittkowski [30].

Algebraic equations of the form

$$0 = F_a(u, u_x, u_{xx}, x, t) \tag{7}$$

are discretized in the same way as the time-dependent ones, leading to a system of $n_a n$ algebraic equations of the form

$$0 = F_a(u_i(t), d_i^1(u, t, h), d_i^2(u, t, h), x_i, t) \quad , \tag{8}$$

$i = 1, \ldots, n$, where $d_i^1(u, t, h)$ and $d_i^2(u, t, h)$ represent suitable approximations of first and second spatial derivatives.

In the discretized problem, we are also able to distinguish explicitly between the differential and algebraic variables, $u_i(t) = (u_i^d(t), u_i^a(t))$, where $u_i^d(x, t)$ denotes the n_d differential variables and $u_i^a(t)$ the n_a algebraic ones, $i = 1, \ldots, n$. The result is a system of $nn_p = n(n_d + n_a)$ ordinary differential algebraic equations, that we can solve by any available DAE solver, for example the implicit Radau-type method of Hairer and Wanner [5].

But without knowing all details of these spatial approximations, it is more or less impossible to provide the DAE solver with consistent initial values. Thus, they must be computed in advance by Newton's method. Consequently, we allow only index-1-systems

of the discretized DAE unless it is guaranteed that consistent initial values are available. More precisely, we have to solve a system of $n_a n$ equations

$$0 = F_a(u_i^0, d_i^1(u_0, 0, h), d_i^2(u_0, 0, h), x_i, 0) \ , \tag{9}$$

where $u_0 = (u_1^0, \ldots, u_n^0)^T$ and $u_i^0 = (u_i^{d0}, u_i^{a0})$ for $i = 1, \ldots, n$ with given initial values u_i^{d0} for the differential variables. The unknown algebraic variables u_i^{a0} are computed by Newton's method starting from given initial values, more precisely, by the code NLPQL of Schittkowski [21].

Example 2.1 *We consider the heat equation, which is also discussed in Example 6. A very simple approximation by symmetric differences of the equations*

$$\begin{aligned} T_t &= D\, S_x \ , \\ 0 &= T_x - S \end{aligned}$$

with $n = 5$ lines leads to a system of ten differential algebraic equations with index 1,

$$\begin{aligned}
\dot{T}_1(t) &= \frac{D}{2h}(-S_3(t) + 4S_2(t) - 3S_1(t)) \ , & 0 &= \frac{1}{2h}(-T_3(t) + 4T_2(t)) - S_1(t) \ , \\
\dot{T}_2(t) &= \frac{D}{2h}(S_3(t) - S_1(t)) \ , & 0 &= \frac{1}{2h}T_3(t) - S_2(t) \ , \\
\dot{T}_3(t) &= \frac{D}{2h}(S_4(t) - S_2(t)) \ , & 0 &= \frac{1}{2h}(T_4(t) - T_2(t)) - S_3(t) \ , \\
\dot{T}_4(t) &= \frac{D}{2h}(S_5(t) - S_3(t)) \ , & 0 &= \frac{1}{2h}(-T_3(t)) - S_4(t) \ , \\
\dot{T}_5(t) &= \frac{D}{2h}(3S_5(t) - 4S_4(t) + S_3(t)) \ , & 0 &= \frac{1}{2h}(-4T_4(t) + T_3(t)) - S_5(t) \ .
\end{aligned}$$

Dirichlet boundary conditions define $T_1(t) = 0$ and $T_5(t) = 0$. Thus, we can omit the differential equations for $\dot{T}_1(t)$ and $\dot{T}_5(t)$, and the algebraic equations are somewhat simplified. Initial values are $T_i(0) = \sin(x_i\pi)$ for $i = 2, \ldots, 4$. Consistent initial values for $S_i(t)$ are easily computed from the last five equations

$$\begin{aligned}
S_1(0) &= \frac{1}{2h}(-T_3(0) + 4T_2(0)) &&= \frac{1}{2h}(-\sin(x_3\pi) + 4\sin(x_2\pi)) \ , \\
S_2(0) &= \frac{1}{2h}T_3(0) &&= \frac{1}{2h}\sin(x_3\pi) \ , \\
S_3(0) &= \frac{1}{2h}(T_4(0) - T_2(0)) &&= \frac{1}{2h}(\sin(x_4\pi) - \sin(x_2\pi)) \ , \\
S_4(0) &= \frac{1}{2h}(-T_3(0)) &&= -\frac{1}{2h}\sin(x_3\pi) \ , \\
S_5(0) &= \frac{1}{2h}(-4T_4(0) + T_3(0)) &&= \frac{1}{2h}(-4\sin(x_4\pi) + \sin(x_3\pi)) \ .
\end{aligned}$$

The above example, however, could have been formulated also as a time-dependent PDE. In general, we do not make any assumptions about the number of differential or algebraic equations in (1), i.e., the dimensions of F_d or F_a, respectively. It is possible that the second set of equations vanishes completely leading to a system of time-dependent partial differential equations, $n_a = 0$, or that the differential equations in (1) do not appear at all, $n_d = 0$. In the latter case, we get a system of algebraic equations, where one or another coefficient may depend on the time variable. Thus, the resulting problem is not necessarily a boundary value problem (BVP) and the solution can depend on the time as shown by the subsequent fourth-order partial differential equation.

Example 2.2 *Assume that the displacement of a beam is described by a fourth-order partial differential equation without time-derivatives. After introducing an auxiliary variable v, we get a second-order system of equations*

$$\begin{aligned} 0 &= E\, v_{xx} + w - l(t) \ , \\ 0 &= w_{xx} - v \end{aligned} \tag{10}$$

with an elasticity parameter $E > 0$ and an external time-dependent load $l(t) = 1 + t + t^2$, see Timoshenko and Goodier [31]. The spatial variable x varies from 0 to 3, and the time variable t between 0 and 2. Boundary values are $w(0) = w(3) = 0$, $w_x(0) = w_x(3) = 0$ to force the beam being fixed at both end points. The resulting displacement distribution is shown in Figure 2.

3 A Mathcad Worksheet Example

Mathcad (`http://www.mathcad.com`) is an interactive GUI with a large number of built-in mathematical functions. Special commands allow to solve systems of one-dimensional partial differential equations. The subsequent lines describe the usage of *Pdesolve* for solving PDAEs, see also the Mathcad documentation

The Pdesolve function is used within solve blocks, allowing for natural notation, and are the easiest to use and interpret.

Pdesolve(u, x, xrange, t, trange, [xpts], [tpts])) returns a function or vector of functions of x and t that solve a one-dimensional nonlinear PDAE or a system of PDAEs, see (1). Values are interpolated from a matrix of solution points calculated using the numerical method of lines.

Arguments:

- u is the explicit vector of function names (with no variable names included) precisely as they appear within the solve block. This argument degenerates to a scalar in the case of a single PDAE.

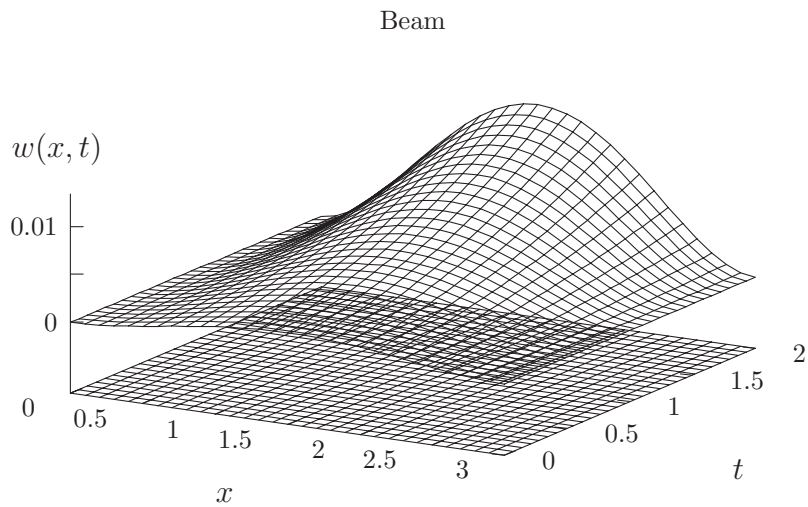

Figure 2: Solution of a Fourth-Order Partial Differential Equation

- x is the spatial variable.
- *xrange* is a 2-element column vector containing the boundary values for x. Both values must be real.
- t is the time variable.
- *trange* is a 2-element column vector containing the boundary values for t. Both values must be real.
- *xpts* (optional) is the integer number of spatial discretization points.
- *tpts* (optional) is the integer number of temporal discretization points.

A *solve block* refers to a group of steps involved when solving a system of differential equations. Needed are initial the key word *Given*, a set of equations, and the solving function *Pdesolve*. Collectively, these steps are known as a solve block.

To give an impression how a test problem is implemented, we consider problem HEAT, see Example 6 or Schittkowski [30] for an alternative Mathcad implementation. Subsequently, the Mathcad implementation is listed, see Figure 3.

HEAT_A

Partial Differential Algebraic Equation

Description: Heat equation, formulated with algebraic equation

Constants: $D := 1$ $\quad a := 1$

Differential Equations: Given

$$u_t(x,t) = D \cdot v_x(x,t)$$

$$0 = v(x,t) - u_x(x,t)$$

Initial Values: $u(x,0) = a \cdot \sin(\pi \cdot x)$ $\quad v(x,0) = 0.0$

Boundary Values: $u(0,t) = 0$ $\quad u(1,t) = 0$

Integration: $\begin{pmatrix} u \\ v \end{pmatrix} := \mathrm{Pdesolve}\left[\begin{pmatrix} u \\ v \end{pmatrix}, x, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, t, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, 50, 50\right]$

Plots: $U := \mathrm{CreateMesh}(u, 0, 1, 0, 1, 50, 50)$

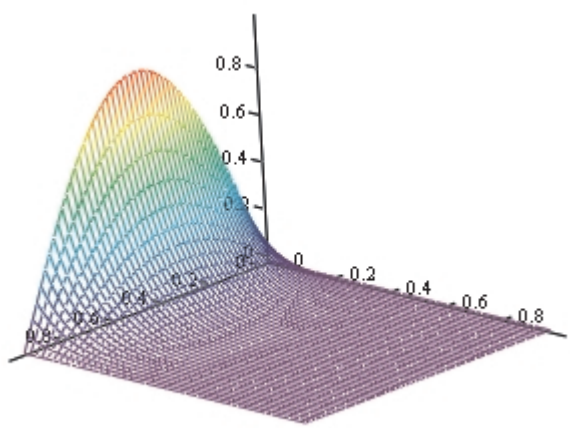

Figure 3: Mathcad Implementation: Heat Equation

4 List of All Test Problems

The subsequent table contains a list of all test problems together with the number of differential variables or equations n_d, the number of algebraic variables or equations, n_a, a brief description of the practical or mathematical background, and some references. The differential equations have first been implemented in the modelling language PCOMP, see Dobmann et al. [3] or Schittkowski [23, 24, 25]. The transformation into Mathcad worksheets follows a unified format based on the PCOMP equations. Thus, the implementations do not exploit all possible features of Mathcad to get the most elegant and compact description. All mcd-files can be downloaded from the home page of the author[3].

Partial Differential Algebraic Equations

name	n_d	n_a	*background*	*ref*
2ND_DIR1	0	1	Second order Dirichlet problem	[10]
2ND_DIR2	0	1	Second order inhomogeneous Dirichlet problem	[10]
ACCRET_F	2	1	Thermal equilibrium curves in Keplerian accretion disks	[17]
BEAM1	0	2	Curved beam	[8], [31]
BVP_TRIV	0	1	Boundary value problem with known solution	[9]
CAPILL	0	1	Capillar filled with water under electric charge	
CNT_CUR2	2	1	Counter-current separation of fluid phase concentrations with phase equilibrium	[15]
CTFLOW	2	2	Two incompressible counter-current flows of binary liquid mixture with semi-permeable wall	[14]
ELA_TUBE	1	1	Waves propagating in a liquid-filled elastic tube (Korteweg-de Vries-Burgers equation)	[7]
ELDYN_A	2	2	Electrodynamic application with algebraic equations	[1]
EW_WAVE	1	1	Wave propagation in media with nonlinear steepening and dispersion	[6]
HEAT_A	1	1	Heat equation, formulated with algebraic equation	
KDVE	1	1	Shallow water flow, balancing front sharpening and dispersion to produce solitons	[20]
PAR_SINA	1	1	Parabolic PDE with inhomogeneous sinus-term (with algebraic equation)	[18], [16]

(continued)

[3] http://www.klaus-schittkowski.de

name	n_d	n_a	*background*	*ref*
PDAE4	1	1	Simple fourth order PDAE with exact solution	
PLASMA	2	2	Space-time movement of ions and electrons	[11]
UNI_BEAM	2	3	Thin uniform cantilevered beam	[4]

Acknowledgement: The author would like to thank Angela Busse, Irina Hübner, Dominik Stadelmaier for for preparing the Mathcad worksheets.

References

[1] Blom J.G., Zegeling P.A. (1994): *Algorithm 731: A moving grid interface for systems of one-dimensional time-dependent partial differential equations*, ACM Transactions on Mathematical Software, Vol. 20, No. 2, 194-214

[2] Campbell S.L., Marszalek W. (1996): *The index of an infinite dimensional implicit system*, Mathematical Modelling of Systems, Vol. 1, No. 1, 1-25

[3] Dobmann M., Liepelt M., Schittkowski K. (1995): *Algorithm 746: PCOMP: A Fortran code for automatic differentiation,* ACM Transactions on Mathematical Software, Vol. 21, No. 3, 233-266

[4] Goodson R.E., Polis M.P. (1978): *Identification of parameters in distributed systems*, in: Distributed Parameter Systems, W.H. Ray, D.G. Lainiotis eds., Marcel Dekker, New York, Basel, 47-134

[5] Hairer E., Wanner G. (1991): *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems,* Springer Series Computational Mathematics, Vol. 14, Springer, Berlin

[6] Hamdi S., Gottlieb J.J., Hanson J.S. (2001): *Numerical solutions of the equal width wave equation using an adaptive method of lines*, in: Adaptive Methods of Lines, A. Vande Wouwer, Ph. Saucec Ph., W. Schiesser eds., Chapman and Hall/CRC, Boca Raton

[7] Johnson R.S. (1970): *A nonlinear equation incorporating damping and dispersion*, Journal of Fluid Dynamics, Vol. 42, 49-60

[8] van Kan J.J.I.M., Segal A. (1995): *Numerik partieller Differentialgleichungen für Ingenieure,* Teubner

[9] Langtangen H.P. (1999): *Computational Partial Differential Equations*, Lecture Notes in Computational Science and Engineering, Vol. 2, Springer, Berlin, Heidelberg

[10] Lewis R.M., Patera A.T., Peraire J. (2000): *A posteriori finite element bounds for sensitivity derivatives of partial-differential-equation outputs*, Finite Elements in Design, Vol. 34, 271-290

[11] Lucht W., Debrabant K. (1996): *Models of quasi-linear PDAEs with convection*, Report, Dept. of Mathematics and Computer Science, University of Halle, Germany

[12] Lucht W., Strehmel K. (1998): *Discretization based indices for semilinear partial differential algebraic equations*, Applied Numerical Mathematics, Vol. 28, 371-386

[13] Martinson W.S., Barton P.I. (1996): *A differentiation index for partial differential equations*, SIAM Journal on Scientific Computing, Vol. 21, No. 6, 2295-2315

[14] Molander M. (1990): *Computer aided modelling of distributed parameter process*, Technical Report No. 193, School of Electrical and Computer Engineering, Chalmers University of Technology, Göteborg, Sweden

[15] Pfeiffer B.-M., Marquardt W. (1996): *Symbolic semi-discretization of partial differential equation systems*, Mathematics and Computers in Simulation, Vol. 42, 617-628

[16] Pfleiderer J., Reiter J. (1991): *Biplicit numerical integration of partial differential equations with the transversal method of lines*, Report No. 279, DFG SPP *Anwendungsbezogene Optimierung und Steuerung*, Technical University, Dept. of Mathematics, Munich

[17] Pin-Gao Gu, E.T. Vishniac, J.K. Cannizo (2000): *Thermal equilibrium curves and turbulent mixing in Keplerian accretion disks*, The Astrophysical Journal, Vol. 534, 380-397

[18] Rektorys K. (1982): *The Method of Discretization in Time and Partial Differential Equations*, Reidel, Dordrecht

[19] Schiesser W.E. (1991): *The Numerical Method of Lines,* Academic Press, New York, London

[20] Schiesser W.E. (1994): *Method of lines solution of the Korteweg-de Vries equation*, Computers in Mathematics and Applications, Vol. 28, No. 10-12, 147-154

[21] Schittkowski K. (1985/86): *NLPQL: A Fortran subroutine solving constrained nonlinear programming problems,* Annals of Operations Research, Vol. 5, 485-500

[22] Schittkowski K. (1997): *Parameter estimation in one-dimensional time dependent partial differential equations*, Optimization Methods and Software, Vol. 7, No. 3-4, 165-210

[23] Schittkowski K. (2001): *EASY-FIT: A software system for data fitting in dynamic systems*, Structural and Multidisciplinary Optimization, Vol. 23, No. 2, 153-169

[24] Schittkowski K. (2002): *Numerical Data Fitting in Dynamical Systems - A Practical Introduction with Applications and Software*, Kluwer Academic Publishers

[25] Schittkowski K. (2004): *PCOMP: A modeling language for nonlinear programs with automatic differentiation*, in: *Modeling Languages in Mathematical Optimization*, J. Kallrath ed., Kluwer, Norwell, MA, 349-367

[26] Schittkowski K. (2004): *110 Mathcad worksheets for nonlinear programming*, Report, Dept. of Computer Science, University of Bayreuth, Germany

[27] Schittkowski K. (2004): *178 Mathcad worksheets for data fitting*, Report, Dept. of Computer Science, University of Bayreuth, Germany

[28] Schittkowski K. (2004): *295 Mathcad worksheets for differential equations*, Report, Dept. of Computer Science, University of Bayreuth, Germany

[29] Schittkowski K. (2004): *28 Mathcad worksheets for differential algebraic equations*, Report, Dept. of Computer Science, University of Bayreuth, Germany

[30] Schittkowski K. (2004): *131 Mathcad worksheets for partial differential equations*, Report, Dept. of Computer Science, University of Bayreuth, Germany

[31] Timoshenko S., Goodier J.N. (1970): *Theory of Elasticity*, McGraw Hill, New York