UMONS
Université de Mons

Dynamic simulation techniques for systems described by AEs/ODEs/PDEs

Alain Vande Wouwer
Philippe Saucez, William Schiesser, Carlos Vilas, Filip Logist

POLYTECH MONS

Outline

Introduction

Simulation environments

Numerical solution of partial differential equations

- A brief view of ODE integration
- The Method Of Lines (MOL)
- Weighted residual methods
- Nonlinear discretization schemes
- Adaptive gridding (static and dynamic)
- Application examples

Conclusions

Introduction

Why is it useful to develop simulators ?

- to study nonlinear system dynamics
- to optimize operating conditions (simulation + optimization)
- to estimate unknown parameters (simulation+ off-line experimental data + optimization)
- to design software sensors (simulation + on-line experimental data)
- to test control structures
- to design model-based controllers (e.g. NMPC)
- to train operators

Introduction

Many systems from science and engineering are described by mixed sets of:

Ordinary Differential Equations (ODEs) representing lumped parameter systems $\underline{x}_t = \underline{f}(\underline{x}, t)$

Partial Differential Equations (PDEs) representing distributed parameter systems $\underline{x}_t = \underline{f}(\underline{x}, \underline{x}_z, \underline{x}_{zz}, ..., z, t)$

Algebraic Equations (AEs) representing constraints on the system states $0 = \underline{f}(\underline{x}, z, t)$

or PDAEs

Introduction

Many systems are distributed in time and space:

- time-varying temperature profiles in a heat exchanger
- time-varying temperature and concentration profiles in a tubular reactor,
- time-varying car density along a highway,
- time-varying deflection profile of a beam subject to external forces,
- time-varying shape and velocity of a water wave,

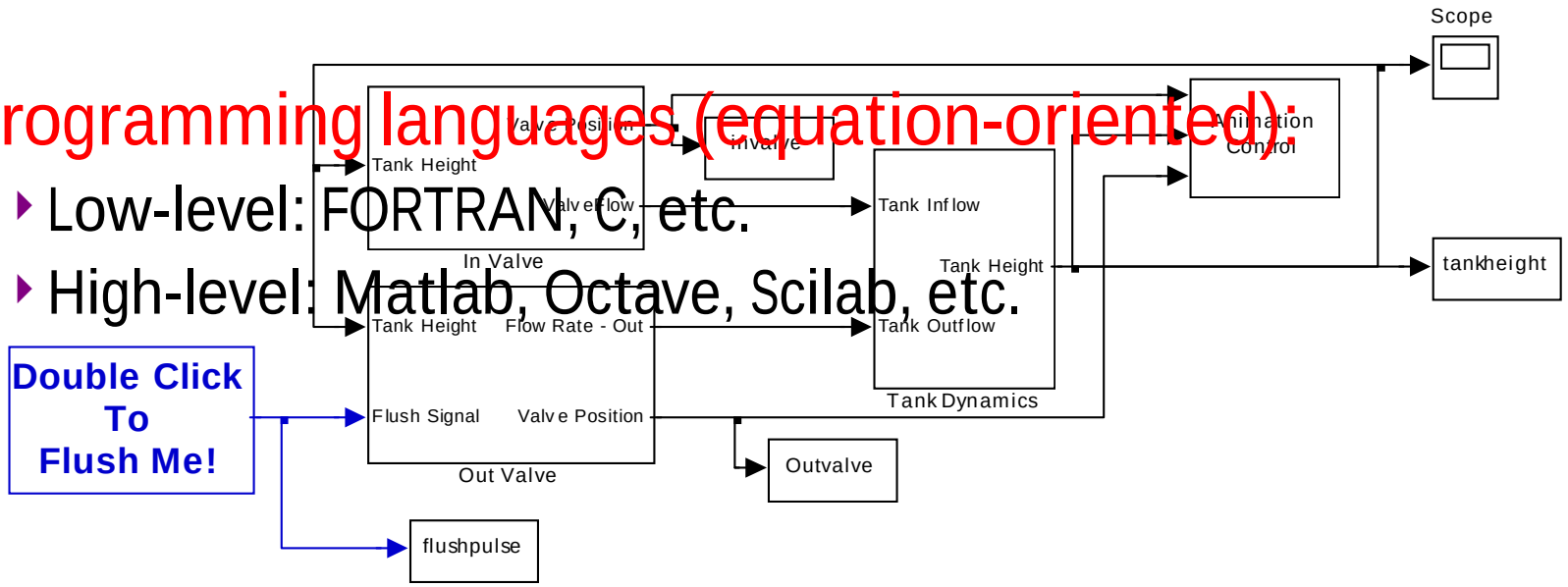
or other independent variable such as “age” or “size” in population models (crystallization, polymerization, grinding, etc.)

Simulation environments

There are two main classes of simulation environments:

Graphical environments (block-oriented):

- Simulink (Matlab)



Programming languages (equation-oriented):

- Low-level: FORTRAN, C, etc.
- High-level: Matlab, Octave, Scilab, etc.

Simulation environments

Graphical environments are easy-to-use, user-friendly, visual and intuitive, but get cumbersome when handling complex systems

Programming environments are more complex (when using low-level languages), but provides more efficient codes

Graphical environments and high-level programming environments provide many useful algorithms, which however appear often as “black-boxes”

- In early developments, these algorithms were sometimes “outdated”, whereas more powerful algorithms were available in public-domain libraries
- Black-box algorithms can be very powerful when used in an informed way (i.e. making them less black-box …)

Simulation environments

Simulation environments can include:

Mathematical libraries:

- Matrix algebra (Blas, LINPACK, LAPACK, etc)
- ODE/DAE solvers (ODEPACK, DASSL, VODE, etc.)
- Optimization algorithms, etc.

Symbolic Manipulation

- Preprocessing tools for generating equations, or even a programming code (based on MAPLE, Mathematica, etc.)

Dedicated environment

- SBT2: Systems Biology Toolbox (Matlab) provides a dedicated environment for the analysis of biological models

A brief view of ODE integration

A ODE system:

$$\underline{x}_t = \underline{f}(\underline{x}, t) \qquad \underline{x}(t_0) = \underline{x}_0$$

Time discretization and integration

$$\underline{x}_{i+1} = \underline{x}_i + \int_{t_i}^{t_{i+1}} f(\underline{x}, t)dt$$

the numerical integration is performed by quadrature, for example

$$\int_{t_i}^{t_{i+1}} f(\underline{x}, t)dt = f(\underline{x}_i, t_i)\Delta t$$

A brief view of ODE integration

which leads to the explicit Euler method:

$$\underline{x}_{i+1} = \underline{x}_i + f(\underline{x}_i, t_i)\Delta t$$

Two central considerations:

- Accuracy $\rightarrow$ error estimation and time-step control
- Stability $\rightarrow$ time step limitation

The explicit Euler method has a limited stability region

$$\left|\lambda \Delta t\right| < 2$$

A brief view of ODE integration

A more advanced method is the 4th-order Runge-Kutta scheme

$$\underline{x}_{i+1} = \underline{x}_i + \left[\frac{25}{216} k_1 + \ldots + \frac{1}{5} k_5 \right]$$

where the several stages are computed as follows:

$$k_1 = f(\underline{x}_i, t_i) \quad \cdots \quad k_5 = f(\underline{x}_i + (\frac{439}{216} k_1 + \ldots + \frac{845}{4104} k_4), t_i + \Delta t)$$

- Accuracy $\rightarrow$ error estimation by comparison with a 5th-order formula
- Stability $\rightarrow$ not much better $\left|\lambda \Delta t\right| < 2.785$

A brief view of ODE integration

Stiffness ?

The smallest eigenvalue λ_{min} determines the problem time scale

The largest eigenvalue λ_{max} determines the maximum time step size Δt_{max}

$$\left|\lambda_{max}\Delta t_{max}\right| < 2 \qquad \text{Euler}$$

$$\left|\lambda_{max}\Delta t_{max}\right| < 2.785 \qquad \text{RKF45}$$

Stiff problems are characterized by $\left|\frac{\lambda_{max}}{\lambda_{min}}\right| > 1000$

Explicit methods are computationally inefficient for stiff problems due to their limited stability regions

A brief view of ODE integration

Implicit methods (a simple example: implicit Euler)

$$\int_{t_i}^{t_{i+1}} f(\underline{x}, t)dt = f(\underline{x}_{i+1}, t_{i+1})\Delta t$$

$$\underline{x}_{i+1} = \underline{x}_i + f(\underline{x}_{i+1}, t_{i+1})$$

Unconditionally stable !

More complicated to implement (Newton's iteration)

Less efficient on non-stiff problems

More advanced methods: implicit RK methods

Backward differentiation formulas (BDF)

$$\underline{x}_{i+1} = \sum_{k=0}^{q-1} \alpha_k \underline{x}_{i-k} + \beta_0 f(\underline{x}_{i+1}, t_{i+1})\Delta t$$

A brief view of ODE integration

A ODE system:

$$\underline{x}_t = \underline{f}(\underline{x}, t) \qquad \underline{x}(t_0) = \underline{x}_0$$

Jacobian matrix

$$J = \left[\partial f_i / \partial x_j \right] = J(\underline{x}, t)$$

The spectrum of eigenvalues of J is determinant for stability considerations

The structure of J (full or dense, banded, sparse), is determinant for the computational efficiency

A brief view of ODE integration

Available ODE/DAE solvers:

ODEPACK

DASSL

VODE

MATLAB ODE suite

SUNDIALS

Etc...

The Method Of Lines (MOL)

A two-step procedure:

1. spatial derivatives are first approximated

2. the resulting system of semi-discrete (discrete in space – continuous in time) equations is integrated in time using available ODE solvers

The popularity of the MOL arises from the large collection of readily available ODE solvers

MOL

$$\underline{x}_t = f(t, z, \underline{x}, \underline{x}_z, \underline{x}_{zz})$$

$$0 = b(t, z = 0, \underline{x}, \underline{x}_z)$$

$$0 = b(t, z = L, \underline{x}, \underline{x}_z)$$

$$\underline{x}(t_0, z) = \underline{x}_0(z)$$

PDE problem

Spatial discretization

$$\underline{\tilde{x}}_t = \tilde{f}(t, \underline{\tilde{x}})$$

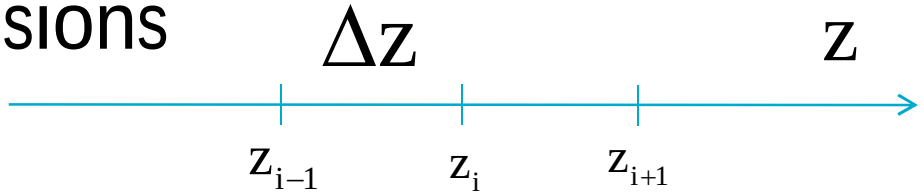
$$0 = \tilde{b}(t, \underline{\tilde{x}})$$

$$\underline{\tilde{x}}(t_0) = \underline{\tilde{x}}_0$$

DAE problem

Finite Differences

Finite difference approximations can be obtained from Taylor series expansions



$$\left.\frac{\partial x}{\partial z}\right|_{z_i} = \frac{x(z_{i+1}) - x(z_{i-1})}{2\Delta z} + O(\Delta z^2)$$

in the spatial domain

$$\left.\frac{\partial x}{\partial z}\right|_{z_1} = \frac{-3x(z_1) + 4x(z_2) - x(z_3)}{2\Delta z} + O(\Delta z^2)$$

at the boundaries

$$\left.\frac{\partial x}{\partial z}\right|_{z_N} = \frac{3x(z_N) - 4x(z_{N-1}) + x(z_{N-2})}{2\Delta z} + O(\Delta z^2)$$

(e.g., three-point centered FDs)

Finite differences

These formulas can be summarized in a compact way using differentiation matrices (easy to implement in Matlab)

$$\underline{\tilde{x}}_z = D_1 \underline{\tilde{x}} = \frac{1}{2\Delta z} \begin{bmatrix} -3 & 4 & -1 & 0 & \dots & \dots & 0 \\ -1 & 0 & 1 & 0 & \dots & \dots & 0 \\ & & & \dots & & & \\ 0 & \dots & -1 & 0 & 1 & \dots & 0 \\ & & & \dots & & & \\ 0 & \dots & \dots & 0 & -1 & 0 & 1 \\ 0 & \dots & \dots & 0 & 1 & -4 & 3 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \dots \\ \tilde{x}_i \\ \dots \\ \tilde{x}_{N-1} \\ \tilde{x}_N \end{bmatrix}$$

(e.g., three-point centered FDs)

Differentiation matrices

Differentiation matrices: a simple and powerful tool within MATLAB

Direct differentiation:

$$\underline{\tilde{x}}_z = \mathbf{D}_1 \underline{\tilde{x}} \qquad \underline{\tilde{x}}_{zz} = \mathbf{D}_2 \underline{\tilde{x}}$$

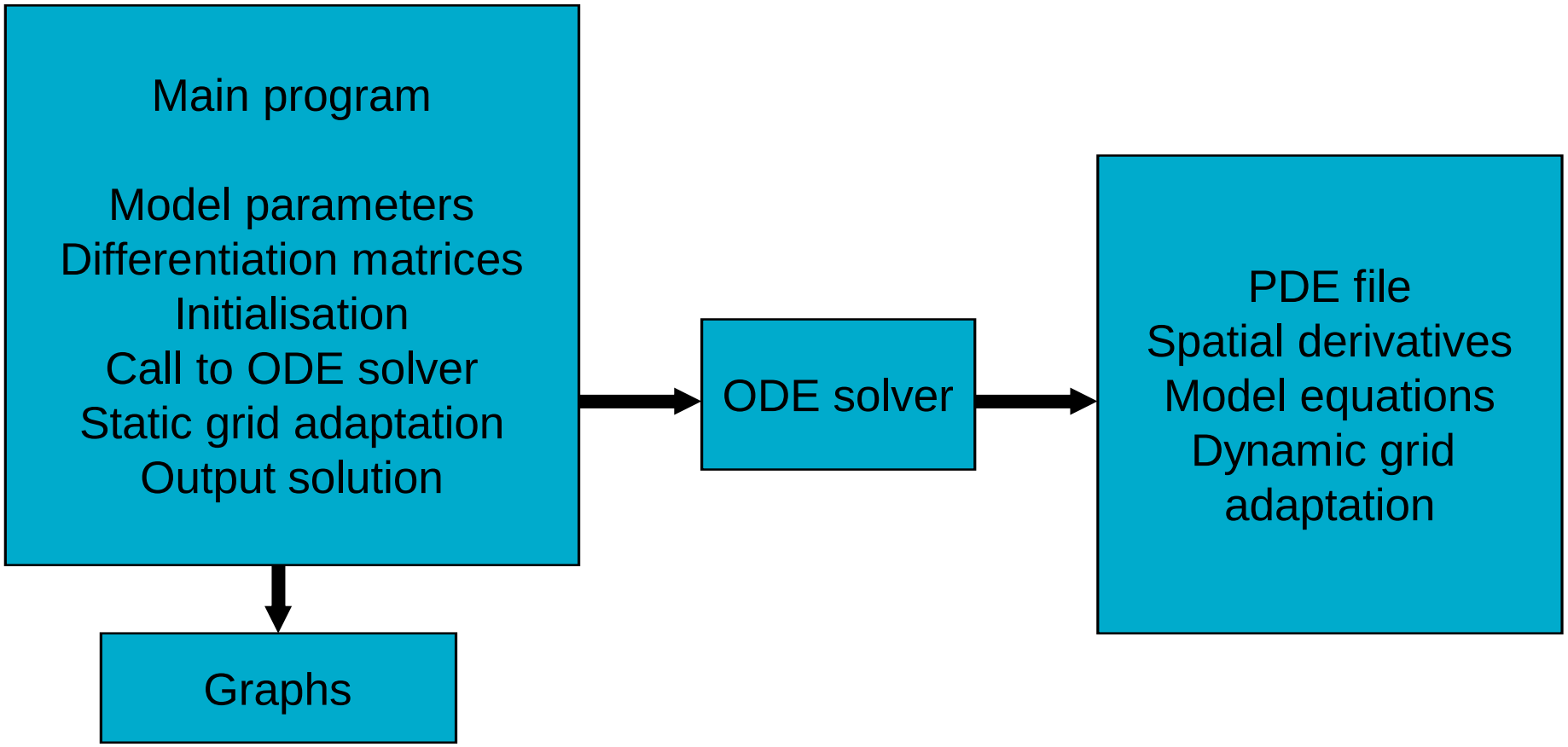
Stagewise differentiation:

$$\underline{\tilde{x}}_z = \mathbf{D}_1 \underline{\tilde{x}} \qquad \underline{\tilde{x}}_{zz} = \mathbf{D}_1 \underline{\tilde{x}}_z$$

$$\underline{\tilde{x}}_{zzzz} = \mathbf{D}_1(\mathbf{D}_1(\mathbf{D}_1(\mathbf{D}_1 \underline{\tilde{x}})))$$

MATMOL: a Matlab MOL Library

Code template (matrix/vector form)



MATMOL: a Matlab MOL Library

Available Methods:

Finite differences

Spectral methods

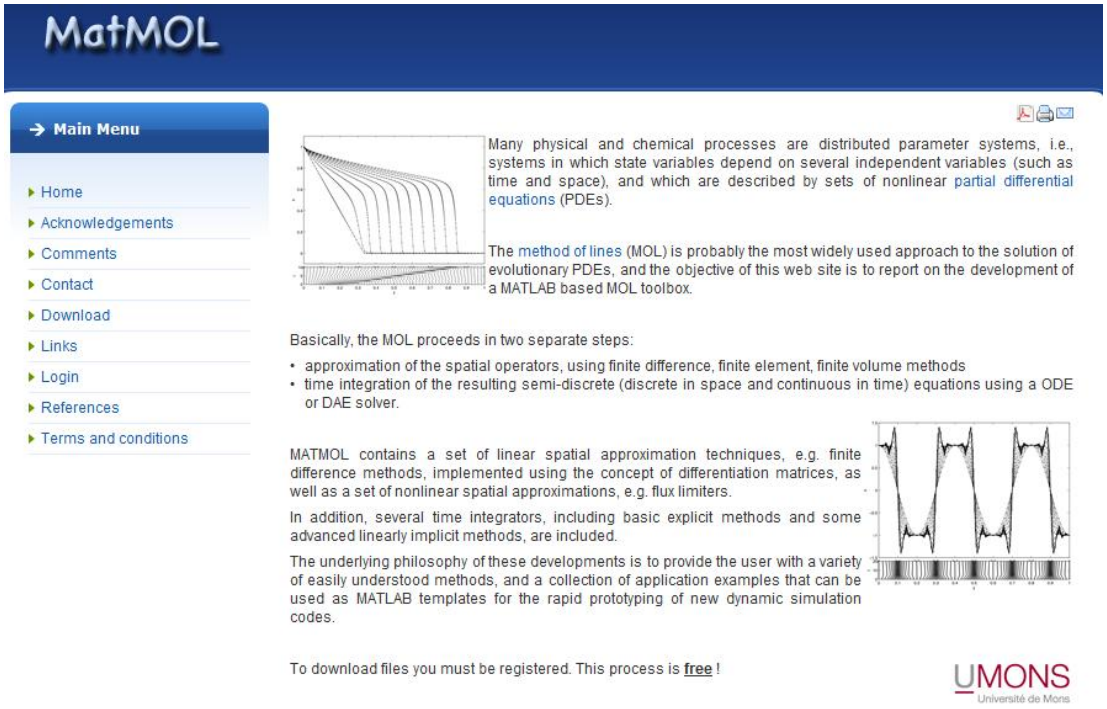
Finite elements

Slope limiters

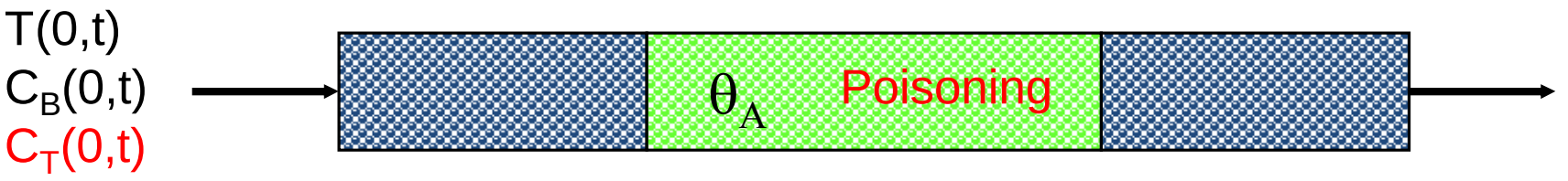
Adaptive gridding (static and dynamic)

Operator splitting

www.matmol.org



A catalytic fixed-bed reactor



$$\frac{\partial c_B}{\partial t} = -v\frac{\partial c_B}{\partial z} + D_B\frac{\partial^2 c_B}{\partial^2 z} + \frac{\rho_c}{\varepsilon} r_B(\theta_A, c_B, T)$$

Benzene hydrogenation

$$\frac{\partial c_T}{\partial t} = -v\frac{\partial c_T}{\partial z} + D_T\frac{\partial^2 c_T}{\partial^2 z} + \frac{\rho_c}{\varepsilon} r_T(\theta_A, c_T, T)$$

Poisoning kinetics of Thiophene

$$\frac{\partial T}{\partial t} = -\varepsilon v\frac{\rho_G c_{pG}}{\overline{\rho c_p}}\frac{\partial T}{\partial z} + \frac{\lambda_{eff}}{\overline{\rho c_p}}\frac{\partial^2 T}{\partial^2 z} + \frac{2\alpha}{R\overline{\rho c_p}}(T_w - T) - \frac{(-\Delta H)}{\overline{\rho c_p}}\rho_c r_B(\theta_A, c_B, T)$$

$$\frac{\partial \theta_A}{\partial t} = r_d(\theta_A, c_T, T)$$

Bed activation

A catalytic fixed-bed reactor

```
%... temporal derivatives
%...
    cB1t = -v*cB1z + DB*cB1zz;
    cT1t = -v*cT1z + DT*cT1zz;
    T1t = -((eps*v*rhog*cpg)/rhocp)*T1z + (leff/rhocp)*T1zz + 2*alpha*(Tw - T1)/(Rr*rhocp);
%...
    cB2t = -v*cB2z + DB*cB2zz - rhoc*rB/eps;
    cT2t = -v*cT2z + DT*cT2zz - rhoc*rT/eps;
    T2t = -((eps*v*rhog*cpg)/rhocp)*T2z + (leff/rhocp)*T2zz + 2*alpha*(Tw - T2)/(Rr*rhocp) +
                                                                    (DH/rhocp)*rhoc*rB;
    thetat = -rd;
%...
    cB3t = -v*cB3z + DB*cB3zz;
    cT3t = -v*cT3z + DT*cT3zz;
    T3t = -((eps*v*rhog*cpg)/rhocp)*T3z + (leff/rhocp)*T3zz + 2*alpha*(Tw - T3)/(Rr*rhocp);
```

A catalytic fixed-bed reactor

Algebraic BCs

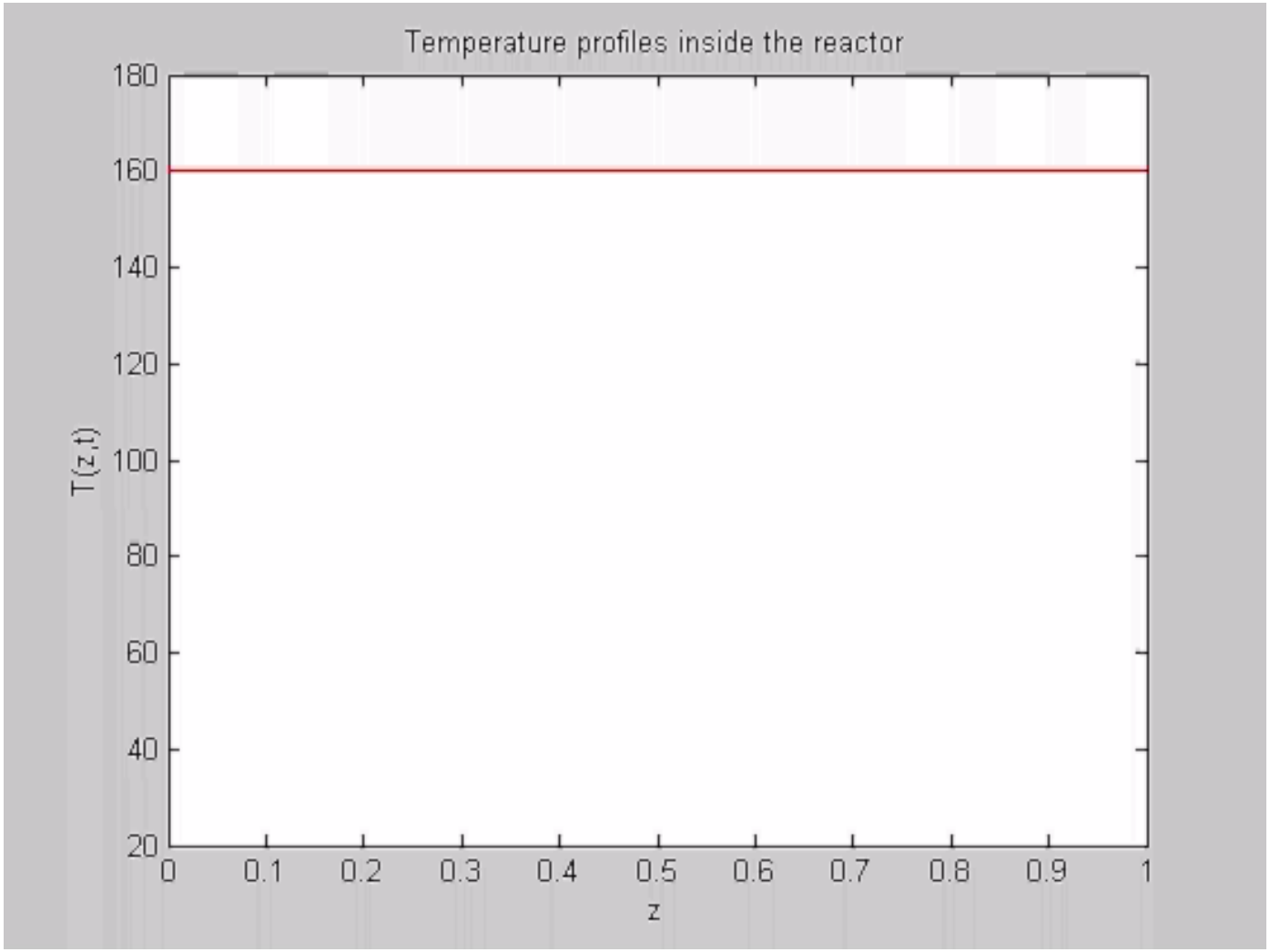
```
%...
%... boundary conditions at z = z01
      cB1t(1) = - (cB1(1)-cBin);
      cT1t(1) = - (cT1(1)-cTin);
       T1t(1) = - (T1(1)-Tin);
%...
%... boundary conditions at z = zL1 = z02
     cB1t(n1) = cB2z(1) - cB1z(n1);
     cT1t(n1) = cT2z(1) - cT1z(n1);
      T1t(n1) = T2z(1) - T1z(n1);
%...
     cB2t(1) = cB1(n1) - cB2(1);
     cT2t(1) = cT1(n1) - cT2(1);
      T2t(1) = T1(n1) - T2(1);
```

```
%...
%... boundary conditions at z = zL2 = z03
      cB2t(n2) = cB3z(1) - cB2z(n2);
      cT2t(n2) = cT3z(1) - cT2z(n2);
       T2t(n2) = T3z(1) - T2z(n2);
%...
      cB3t(1) = cB2(n2) - cB3(1);
      cT3t(1) = cT2(n2) - cT3(1);
      , T3t(1) = T2(n2) - T3(1);
%...
%... boundary conditions at z = zL
      cB3t(n3) = - cB3z(n3);
      cT3t(n3) = - cT3z(n3);
       T3t(n3) = - T3z(n3);
```

A catalytic fixed-bed reactor

5-point biased-upwind FDs for first derivatives

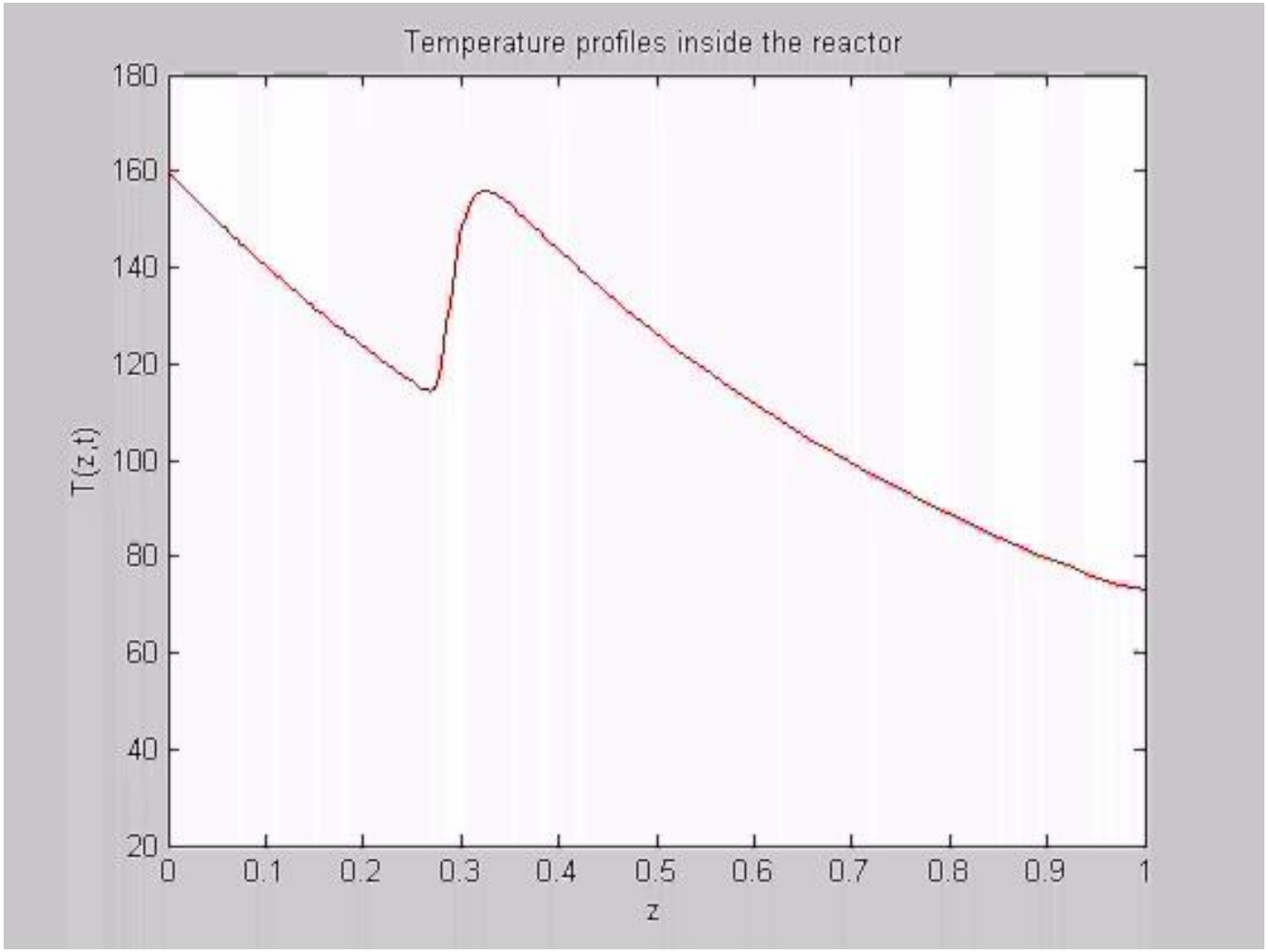
N1 = 71
N2 = 71
N3 = 21



A catalytic fixed-bed reactor

5-point biased-upwind FDs for first derivatives

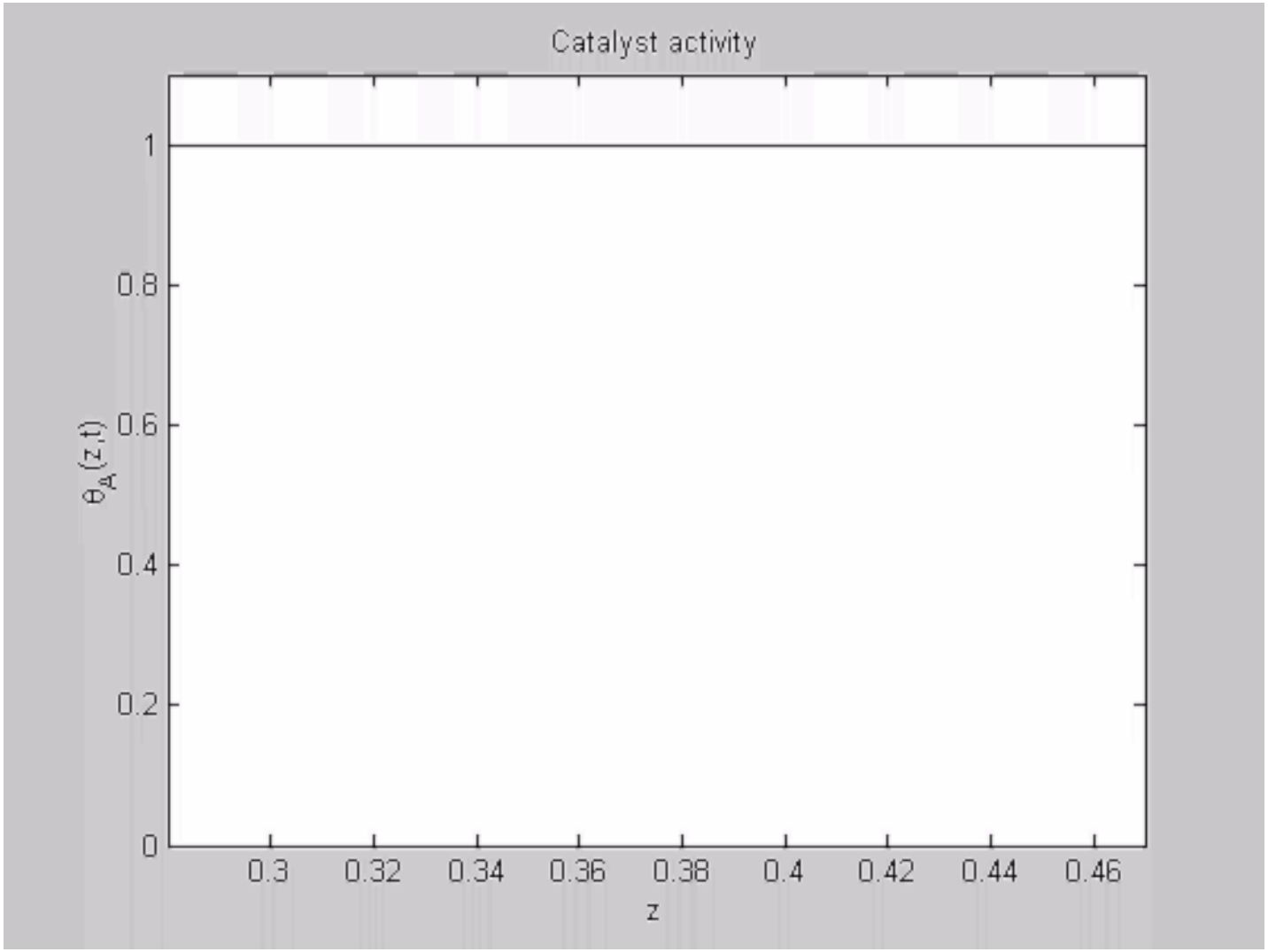
N1 = 71
N2 = 71
N3 = 21



A catalytic fixed-bed reactor

5-point biased-upwind FDs for first derivatives

N1 = 71
N2 = 71
N3 = 21



Weighted residual methods

Weighted residual methods use (usually orthonormal) basis functions to represent the solution:

$$\hat{x}(t,z) \approx \sum_{i=1}^{N} a_i(t)\varphi_i(z)$$

The weighting coefficients are found by solving residual equations, i.e., the PDE residuals are made orthogonal to some (also orthonormal) test functions

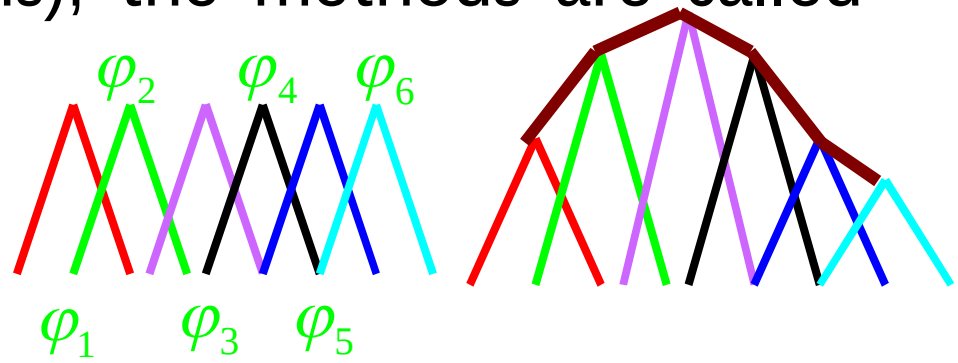
$$R(t,z) = \hat{x}_t(t,z) - f(\hat{x}, \hat{x}_z, \hat{x}_{zz}, z, t)$$

$$\int R(t,z)\eta_i(z)dz = 0 \qquad i = 1, \cdots, N$$

Weighted residual methods

If the basis functions are global (i.e. cover the whole spatial domain), the methods are usually called spectral methods

If the basis functions are local (i.e. have a compact support, e.g. hat functions), the methods are called finite element methods



The finite difference method can be considered as a particular case of finite element method where the basis functions are rectangular pulses

Weighted residual methods

Depending on the choice of the test functions, several particular methods arise:

- $\eta_i(z) = \varphi_i(z) \qquad i = 1, \cdots, N$: Galerkin method
- $\eta_i(z) = \delta(z - z_i) \qquad i = 1, \cdots, N$: Collocation method

The rate of convergence of the collocation method can be accelerated by considering nonuniform distributions, in which more grid points are clustered near the boundaries

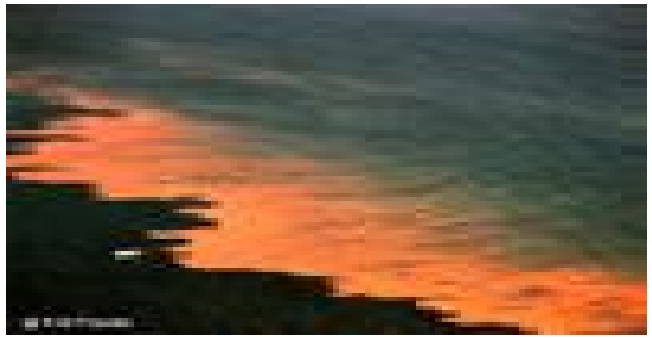
Weighted residual methods

Three main types of basis functions are considered:

- The eigenfunctions of the corresponding linearized, homogeneous, IBVP, when these functions exist and can be computed
- *"Empirical" basis functions (Karhuenen-Loeve or proper orthogonal decomposition – construction of solution « snapshots » and solution of an eigenvalue problem)*
- *Polynomials (*As stated in Weiertrass theorem, any continuous function can be represented in terms of polynomials)

Application example (1)

very compact differentiation matrices, e.g. using spectral collocation



A simple algae bloom model

$$\frac{\partial \rho}{\partial t} = \nu \frac{\partial^2 \rho}{\partial z^2} + r\rho\left(1 - \frac{\rho}{K}\right)$$

$$\rho(0,t) = \rho(L,t) = 0$$

Critical patch size

$$L \approx \pi\sqrt{\frac{\nu}{r}}$$

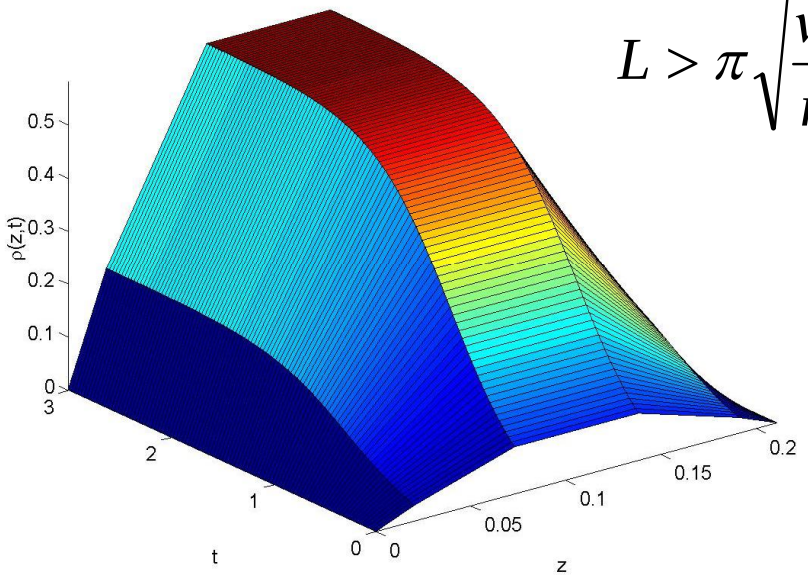
Application example (1)

A simple algae bloom model

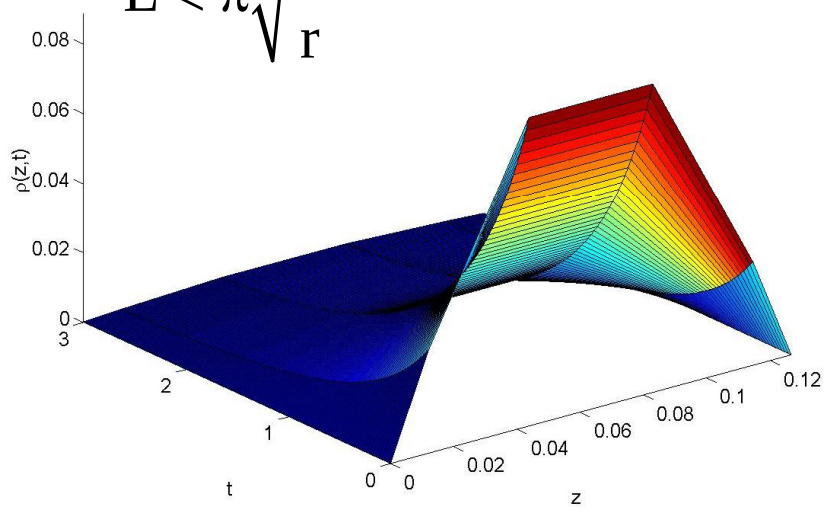
$$\frac{\partial \rho}{\partial t} = \nu \frac{\partial^2 \rho}{\partial z^2} + r\rho\left(1 - \frac{\rho}{K}\right) \qquad \rho(0,t) = \rho(L,t) = 0$$

Chebyshev collocation with only 6 nodes

$$L > \pi\sqrt{\frac{\nu}{r}}$$



$$L < \pi\sqrt{\frac{\nu}{r}}$$



Application example (2)

... and larger ones on more challenging problems,
e.g. an extended equal-width wave equation
[Hamdi, Enright, Schiesser, Gottlieb, 2003]

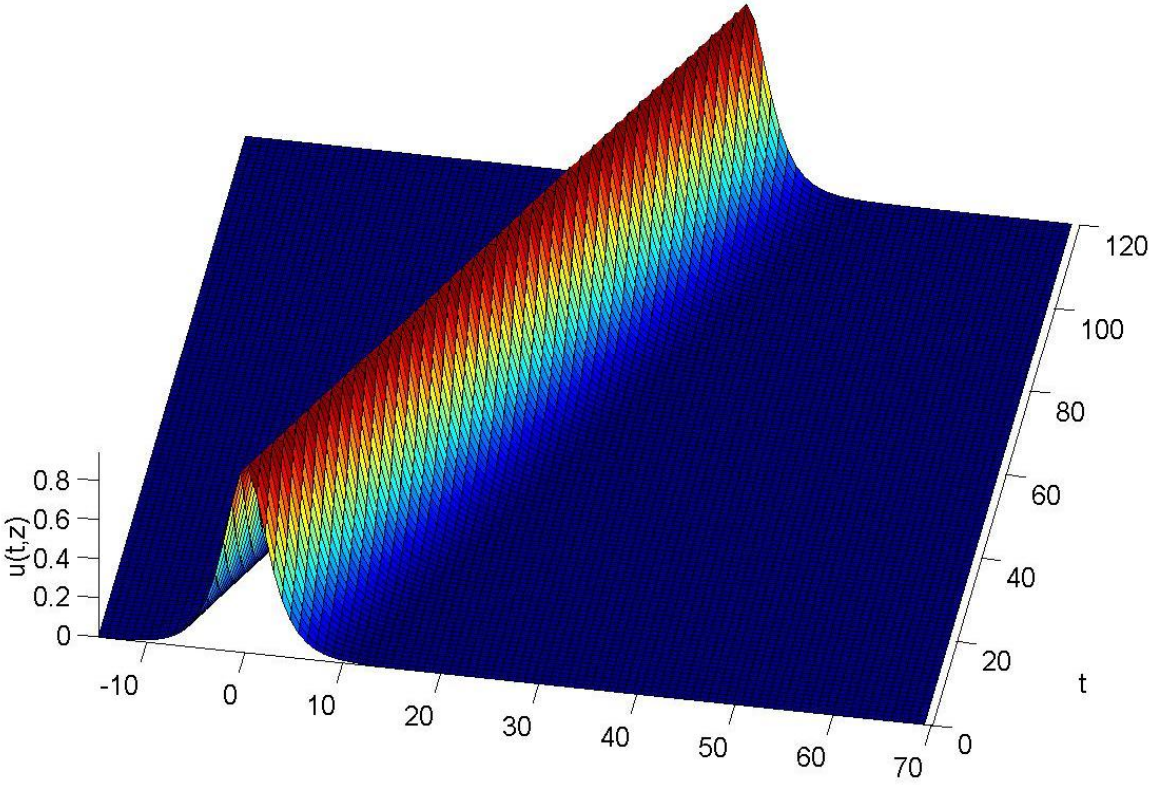
$$u_t + a\,u^p u_z - \mu\,u_{zzt} = 0$$

and an extended equal-width-Burgers equation

$$u_t + a\,u^p u_z - \delta\,u_{zz} - \mu\,u_{zzt} = 0$$

Application Example (2)

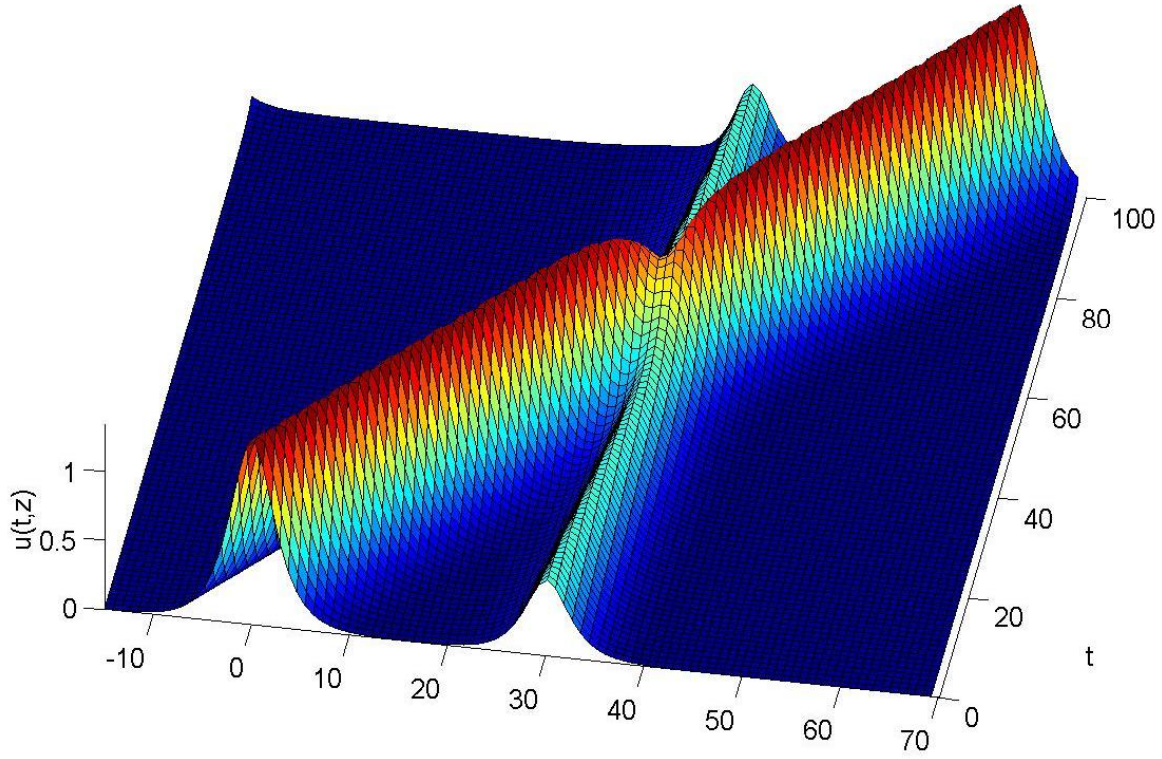
$$u_t + 2u^2 u_z - 4u_{zzt} = 0$$



(c = 0.3 - spectral collocation – 100 nodes)

Application Example (2)

$$u_t + 2u^2 u_z - 4u_{zzt} = 0$$



(c1 = 0.6 – c2 = 0.1 - spectral collocation – 100 nodes)

Application example (2)

$$u_t + 2u^2 u_z - 4u_{zzt} = 0$$

Mass

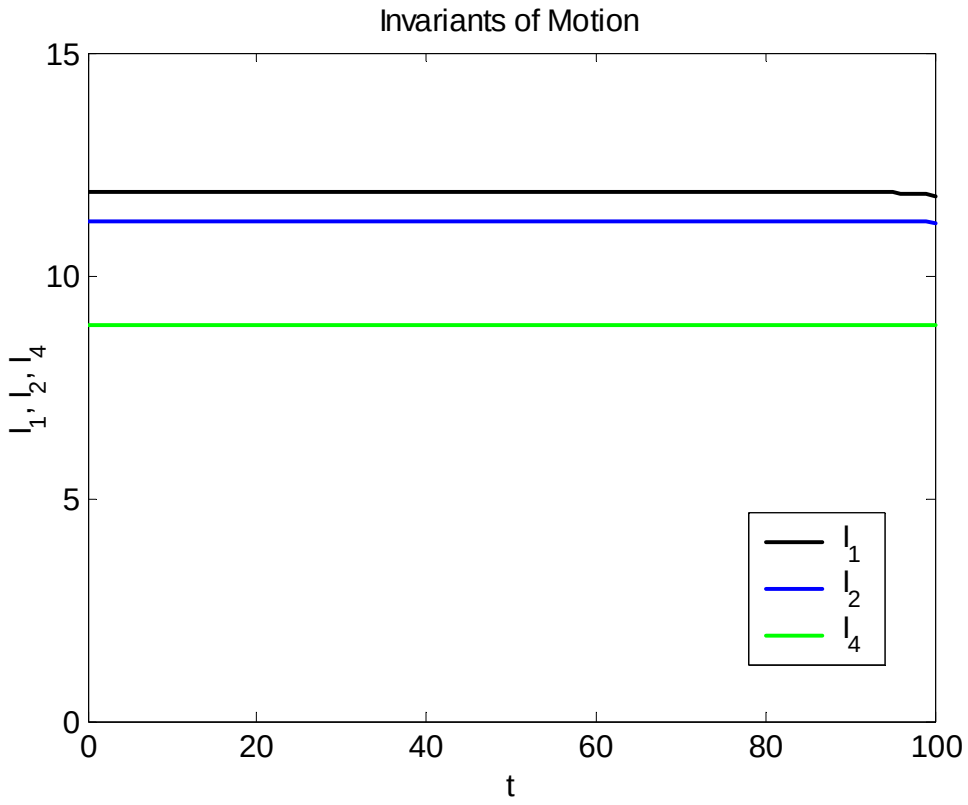
$$I_1 = \int u \, dz$$

Energy

$$I_2 = \int (u^2 + \mu u_z u_z) \, dz$$

« Whitham »

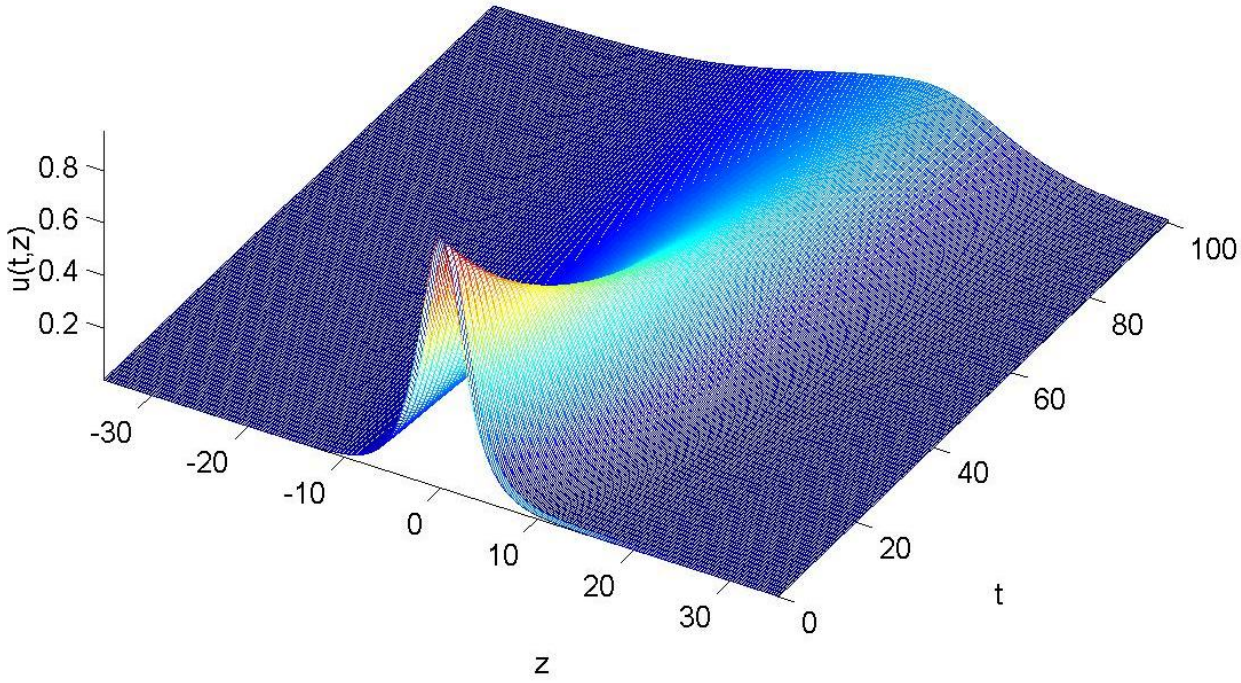
$$I_4 = \int u^4 \, dz$$



(c1 = 0.6 – c2 = 0.1 - spectral collocation – 100 nodes)

Application example (2)

$$u_t + 2u^2 u_z - 0.5u_{zz} + 4u_{zzt} = 0$$

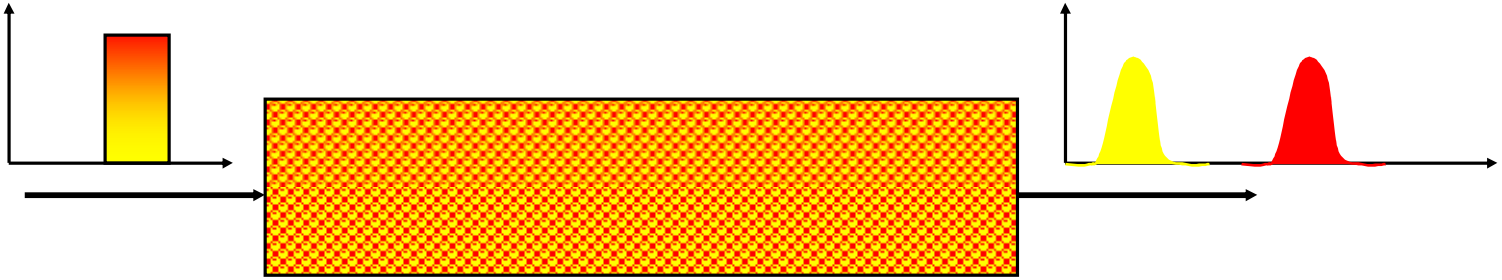


Application example (2)

```
        function xt = f(t,x)
%...
%... set global variables
       global a p mu c;
       global z0 zL z n D1;
%...
%... spatial derivatives
%...
       xz = D1*x;
%...
%... temporal derivatives
%...
       xt = -a*(x.^p).*xz;
%...

      function M = mass(t,x)
%...
%... set global variables
       global a p mu c;
       global z0 zL z n D1;
%...
%... Assemble mass matrix
       M = diag(ones(n,1),0)-mu*D1*D1;
```

Application example (3)



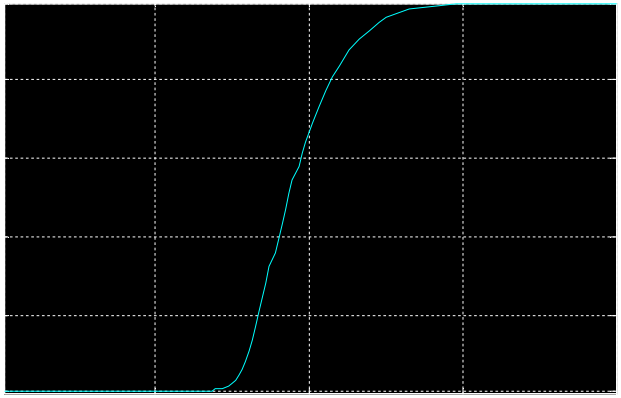
$$\frac{\partial c_b}{\partial t} = -v\frac{\partial c_b}{\partial z} + D_b\frac{\partial^2 c_b}{\partial z^2} - \frac{(1-\varepsilon_b)}{\varepsilon_b}\frac{3k}{R_p}(c_b - c_{p,R_p})$$

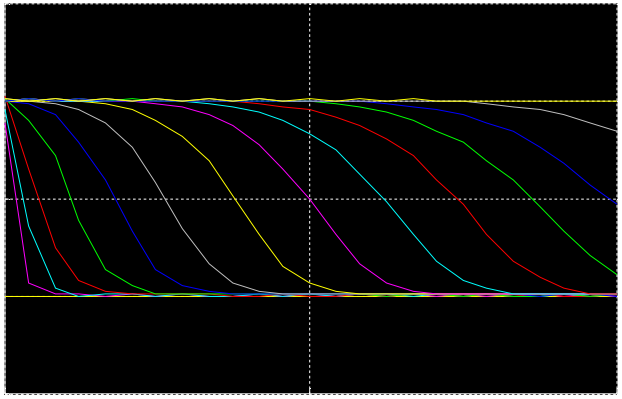
$$\varepsilon_p\frac{\partial c_p}{\partial t} + (1-\varepsilon_p)\frac{\partial c_p^*}{\partial t} = \varepsilon_p D_p\left(\frac{1}{R_p^2}\frac{\partial}{\partial r}\left(\frac{\partial c_p}{\partial r}\right)\right)$$

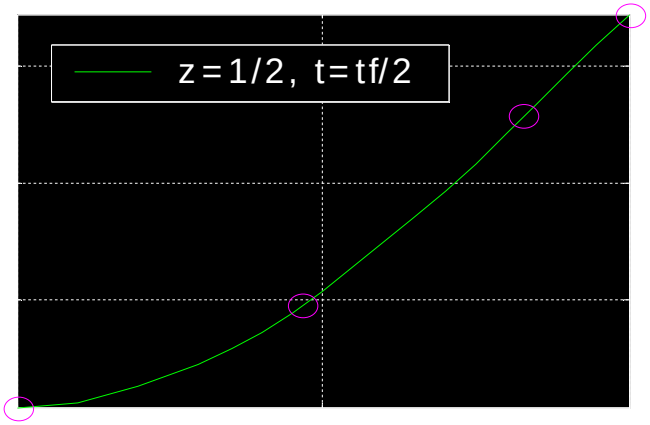
$$c_p^* = \frac{a c_p}{1 + \sum_{i=1}^{n_c} b_i c_{pi}}$$

[Lazo,1999]

Application example (3)







Finite element in the bulk phase (Galerkin method – second-order elements)
Orthogonal collocation in the particles

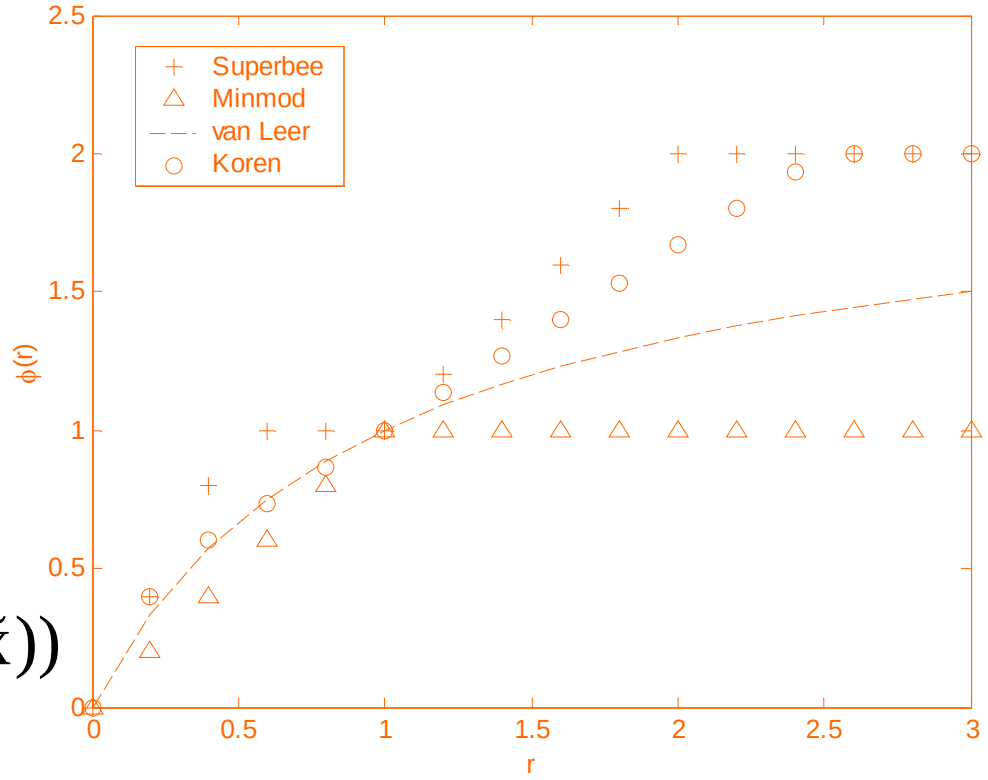
Nonlinear discretization schemes

Of course, other approximation techniques do not cast into the differentiation matrix format, e.g. nonoscillatory schemes such as slope/flux limiters, which are nonlinear

$$\frac{\partial x(t,z)}{\partial t} = \frac{\partial f(x(t,z))}{\partial z}$$

$$r_i = \frac{\tilde{x}_i - \tilde{x}_{i-1}}{\tilde{x}_{i+1} - \tilde{x}_i}$$

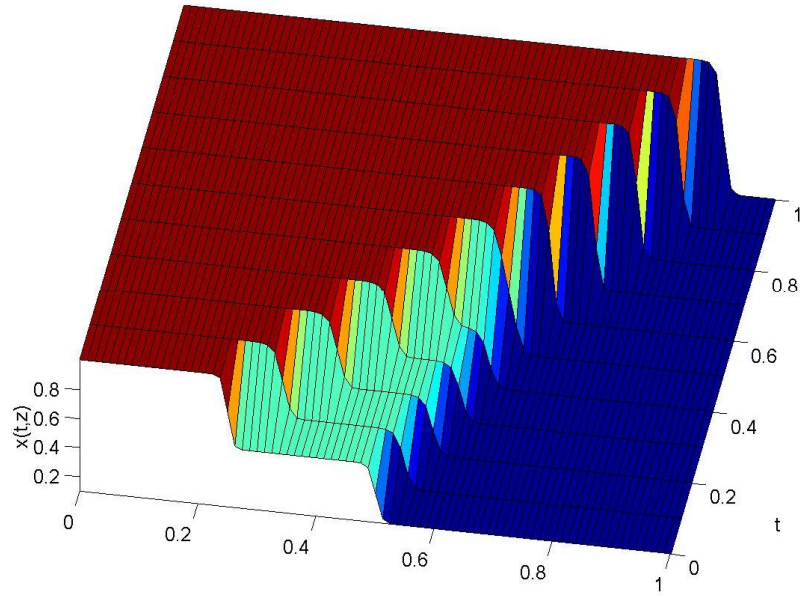
$$\tilde{f}(\tilde{x}) = \tilde{f}_{lo}(\tilde{x}) + \varphi(r)(\tilde{f}_{ho}(\tilde{x}) - \tilde{f}_{lo}(\tilde{x}))$$

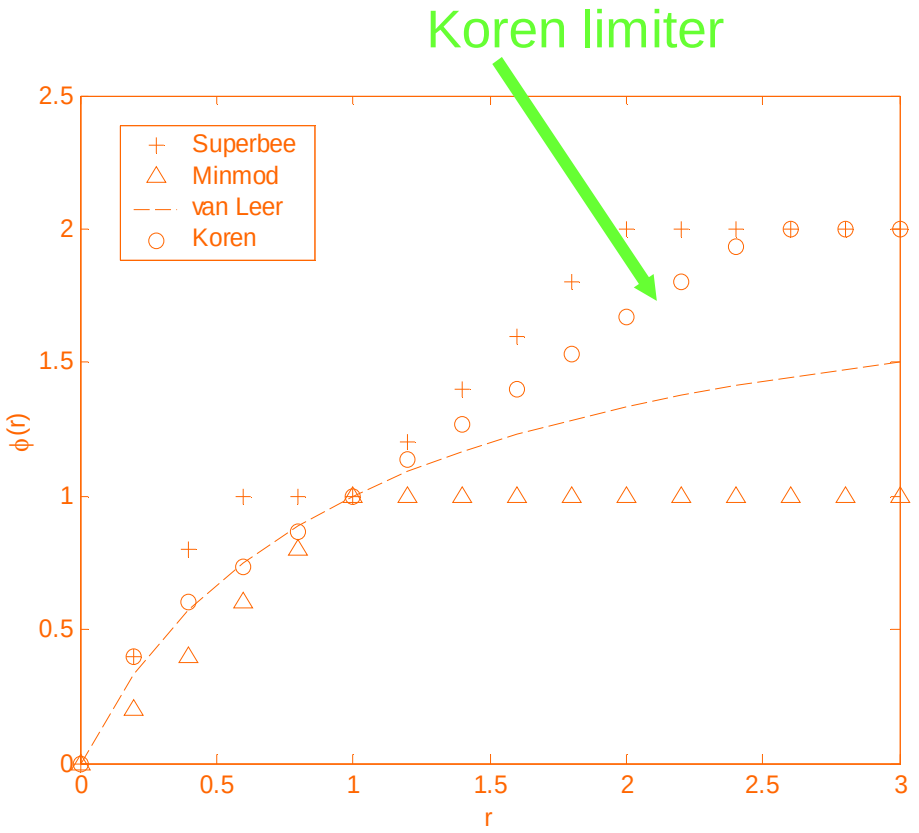


Application example

Burgers equation

$$u_t = -(0.5u^2)_z + \mu u_{zz}$$





Static grid adaptation

Grid Refinement: AGEREG

equidistribution of a given monitor function subject to constraints on the grid regularity

$$m_{i-1}(\underline{x})\,\Delta z_{i-1} = m_i(\underline{x})\,\Delta z_i = c$$

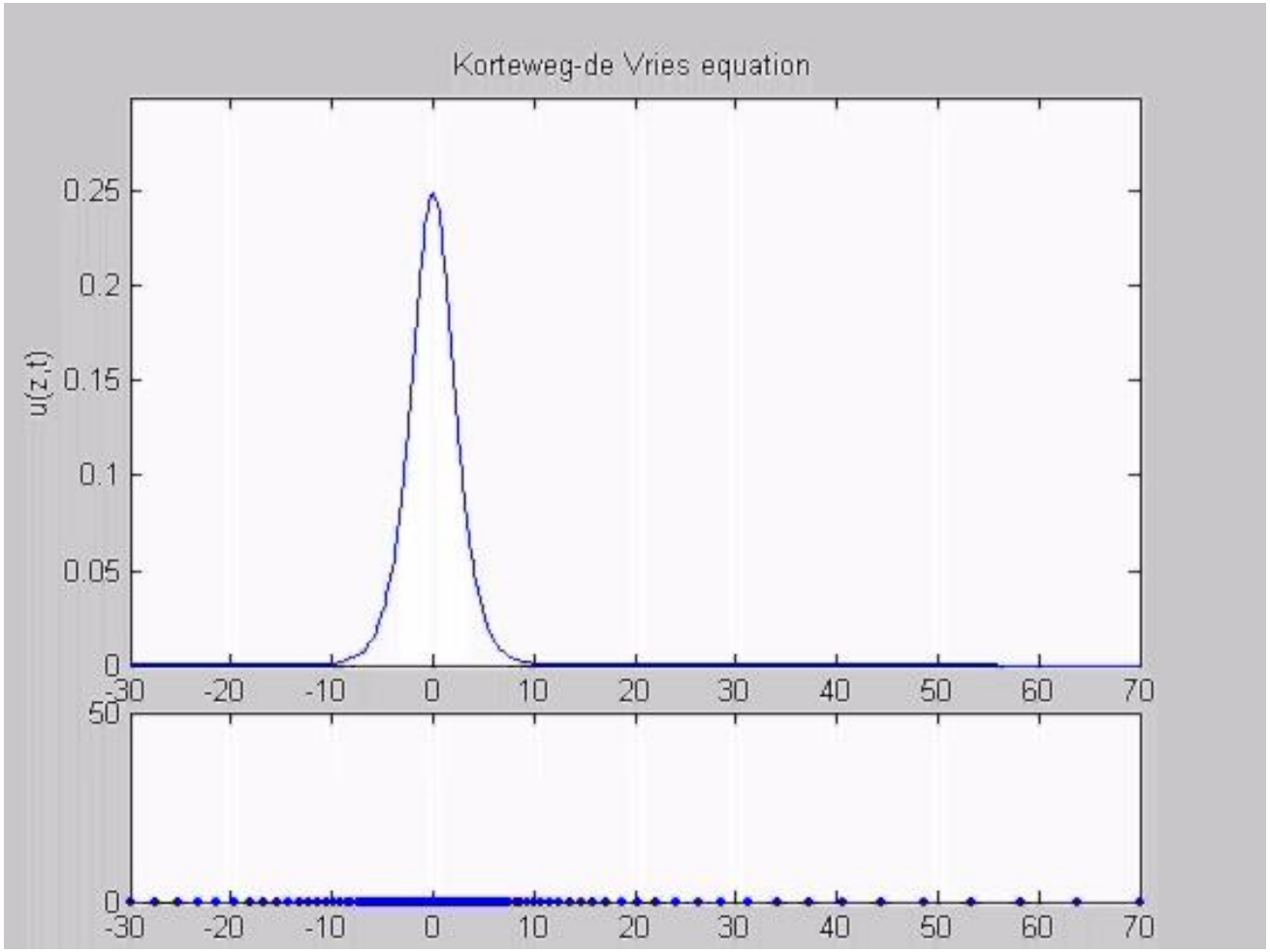
$$\frac{1}{K} \le \frac{\Delta z_i}{\Delta z_{i-1}} \le K$$

An example: Korteweg-de-Vries equation

$$u_t + 6uu_z + u_{zzz} = 0$$

Finite differences and stagewise differentiation
Very small absolute tolerance on ODE15s

Application example



Average number of nodes = 154

Dynamic grid adaptation

Grid Movement: a basic version of MOVGRD [Blom & Zegeling, 1994]

$$\underline{x}_t = \underline{f}(\underline{x}, \underline{x}_z, \underline{x}_{zz}, z, t) \quad \longrightarrow \quad \dot{x} - x_z \dot{z} = \underline{f}(\underline{x}, \underline{x}_z, \underline{x}_{zz}, z, t)$$

Spatial smoothing

$$n_i = 1/\Delta z_i = 1/(z_{i+1} - z_i)$$

$$\tilde{n}_i = n_i - \kappa(\kappa+1)(n_{i+1} - 2n_i + n_{i-1})$$

Temporal smoothing

$$\frac{\tilde{n}_{i-1} + \tau \dot{\tilde{n}}_{i-1}}{M_{i-1}} = \frac{\tilde{n}_i + \tau \dot{\tilde{n}}_i}{M_i}$$

Dynamic grid adaptation

Finite difference approximations

$$\dot{\mathbf{x}} - \mathbf{D}\,\dot{\mathbf{z}} = \mathbf{f}$$

$$\tau\,\mathbf{B}\,\dot{\mathbf{z}} = \mathbf{g}$$

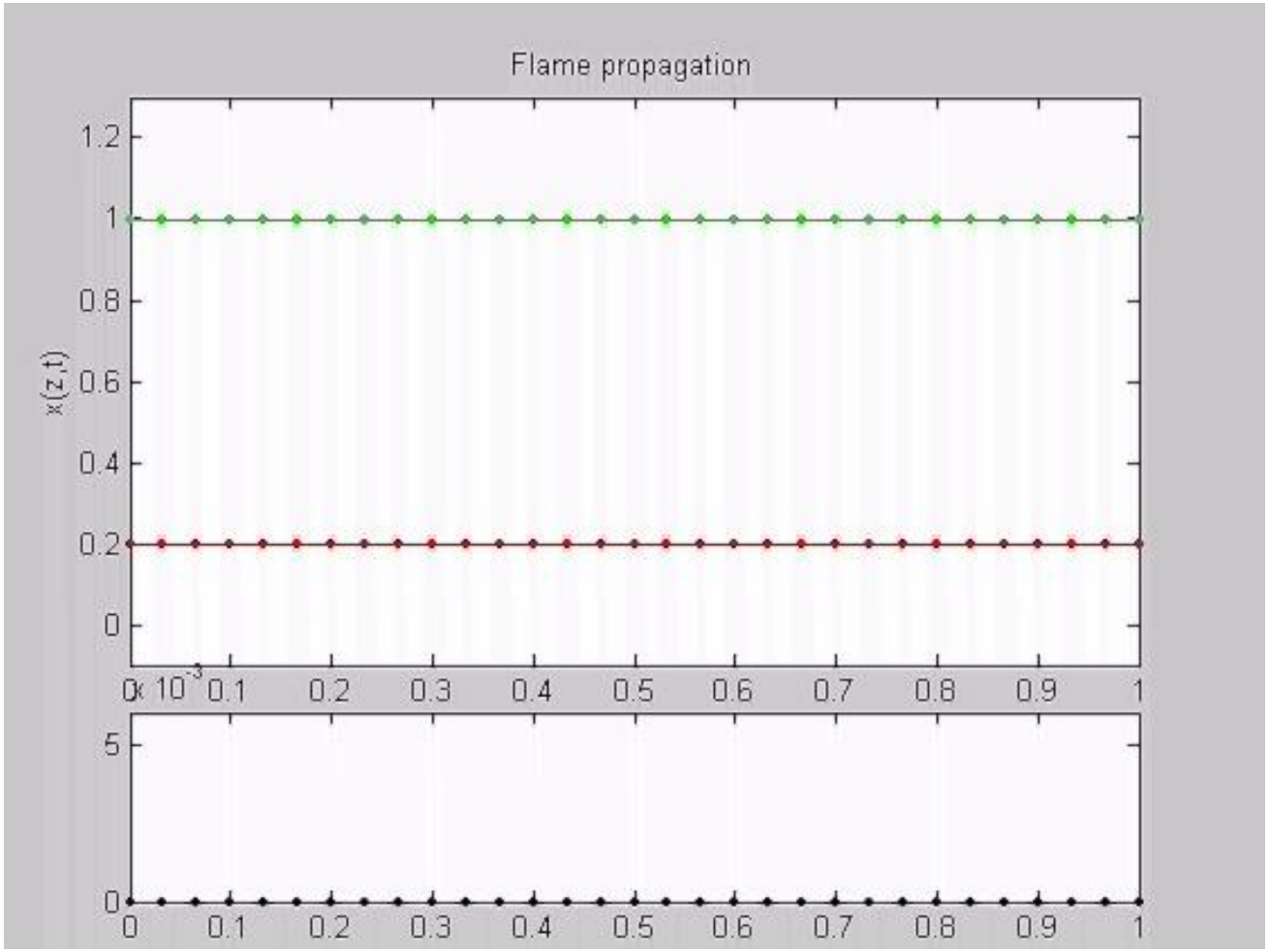
Time integration of the resulting DAEs with ode15s

A simple example: flame propagation

$$\frac{\partial \rho}{\partial t} = \frac{\partial^2 \rho}{\partial z^2} - r\rho, \qquad \rho(z,0) = 1, \qquad r = 3.52 \times 10^6\, e^{-4/T}$$

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial z^2} + r\rho, \qquad T(z,0) = 0.2, \quad T(1,t) = f(t)$$

Application example



(31 moving nodes)

Conclusions

Computational modeling is increasingly used in science (and noteworthy, in emerging fields such as biology or human physiology) and engineering

- commercial software packages
- public–domain libraries
- equation-oriented or block-oriented environments
- numerical algorithms and symbolic manipulations

➢a vast array of readily available methods
➢a gap between research and current practice