Partial Differential Equations in Modelica

Jan Šilar

First Faculty of Medicine
Charles University
Prague, Czech Republic

OpenModelica Workshop 2015

Introduction – PDE

- PDE generalization of ODE
- in PDE (of evelution):
  - unknown – function of time and space coordinates
  - eq. contains $\frac{\partial}{\partial t}$ and $\frac{\partial}{\partial x}$
  - initial and boundary condition
  - problems as e.g. vibrating string/membrane (compare to oscillator – mass on a spring), heat transfer, ...
- Peter Fritzson's Modelica book[1]
- Levon Saldamli PhD thesis [2]
  - extension – some weaknesses
  - not supported in OM or other

We continue this work

Complexity of PDE

- solving PDE is more complicated than ODE
- difficulties grow with every dimension: ODE (0D) $\rightarrow$ 1D PDE $\rightarrow$ 2D PDE $\rightarrow$ 3D PDE $\rightarrow$???
- there is no suitable numerical method for all PDEs
    - common approach – develop method for one particular equation/problem
- no numerical library able to solve vast majority of PDEs (as is *dassl* or *IDA* in ODE)
- Modelica tools solve enormous amount of ODE (DAE) problems (would say almost any) – probably not possible in PDE

Plan for my PhD

- proposed modifications and enhancements to Saldamli's extension

future goals

- add support for particular 1D models, with first derivative
- concern on hyperbolic systems – time dependent, "wave-like" problems, namely conservation laws

$$\bar{w}_t + \bar{F}(\bar{w})_x = 0$$

  - e.g. advection eq, string eq, Euler eq, arterial pulse waves
- other types of PDE: parabolic (time dependent, diffusion problems), elliptic (time independent, potential problems)
  - some of them also solvable some not, no guarantee for correct solution

Extension

example – advection equation

```
model advection "advection equation"
  parameter Real L = 1; // length
  parameter PDEDomains.DomainLineSegment1D omega(l = L);
  field Real u(domain = omega);
  parameter Real c = 1;
initial equation
  u = if omega.x<0.25 then cos(2*3.14*omega.x) else 0;
equation
  der(u) + c*der(u,x) = 0;         //by default in omega.interior
  u = 1                      in omega.left;
  annotation(experiment(GridNodes = 100));
end advection;
```

- domain – `omega`
- field variable – `u`
- partial derivatives – `der(u)` – time, `der(u,x)` – space
- BC with region specifier `in`
- annotation for number of nodes

MOL, implementation

- PDE $\rightarrow$ system of ODEs
- space dimension is discretized
- field variables $\rightarrow$ array
- space derivatives $\rightarrow$ difference

- will implement discretization module in FrontEnd
- resulting ODE system will be written in recent Modelica
- will be solved by rest of the curent compiler and runtime

Discretized (manually) advection eq.

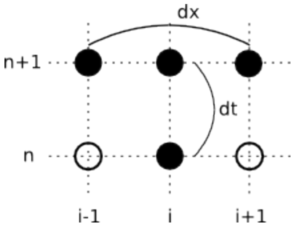
```
model advectionDiscretized
  // u_t + u_x = 0
  parameter Real L = 1;
  constant Integer N = 100;
  parameter Real dx = L / (N - 1);
  parameter Real[N] x = array(i * dx for i in 0:N - 1);
  Real[N] u, u_x;
  parameter Real c = 1;
initial equation
  for i in 1:N loop
    u[i] = if x[i]<0.25 then cos(2*3.14*x[i]) else 0;
  end for;
equation
  //unused array elements, eqs. just for balanced system:
  u_x[1] = 0; u_x[N] = 0;
  for i in 2:N - 1 loop
    //discretization of spatial derivative:
    u_x[i] = (u[i + 1] - u[i - 1]) / (2*dx);
    // the equation:
    der(u[i]) + c*u_x[i] = 0;
  end for;
  u[1] = 1;  //left BC
  u[N] = 2 * u[N - 1] - u[N - 2]; //extrapolation in the last node
  annotation(experiment(Interval = 0.002));
end advectionDiscretized;
```

BTCS difference scheme

combination of particular space difference and time solver $\rightarrow$ concrete methods

in example above – central difference for space derivative

- when using explicit solver (time) – unstable $\Rightarrow$using implicit solver (eg. *radau*1 – implicit Euler)
    - $O(h)$ in time, $O(h^2)$ in space
    - discontinuous solution $\rightarrow$ oscillations
    - problematic parallelization



$$\frac{u_i^{n+1}-u_i^n}{\Delta t} + a\frac{u_{i+1}^{n+1}-u_{i-1}^{n+1}}{2\Delta x} = 0$$

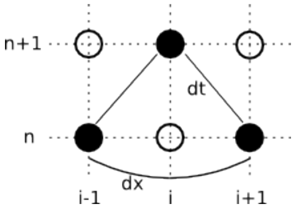
lower space index – by discretization, upper time index – by solver

not any method implementable this way

Lax-Friedrichs (LF)

modified central space difference + Euler time solver

$$\frac{u_i^{n+1} - \frac{1}{2}(u_{i+1}^n + u_{i-1}^n)}{\Delta t} + a\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} = 0$$



- also $O(h)$ in time, $O(h^2)$ in space
- diffusive – doesn't oscillate
- explicit – suitable for parallelization

Time step – cfl condition

hyperbolic problems, stable simulation $\Rightarrow$ the CFL condition

- limits length of dt wrt. dx

$$\lambda \frac{dt}{dx} < c,$$

c .. constant specific to the used method
λ .. speed of waves (of sound) in the system

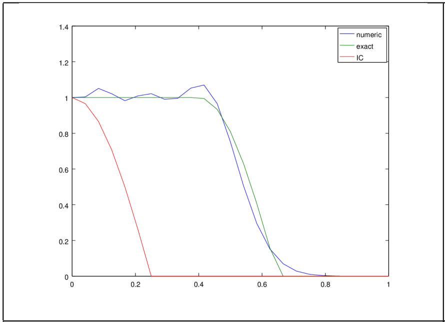
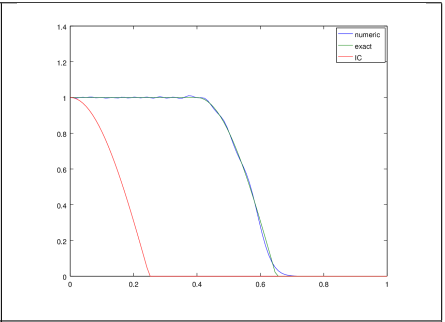
- in conservation laws – maximal eigenvalue of Jacobian

Evaluation of λ must be implemented.
Mechanism to control dt

Tests, advection eq, cos wave, BTCS scheme

$$u_t + a \cdot u_x = 0$$

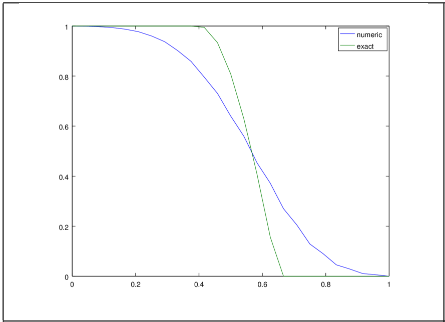
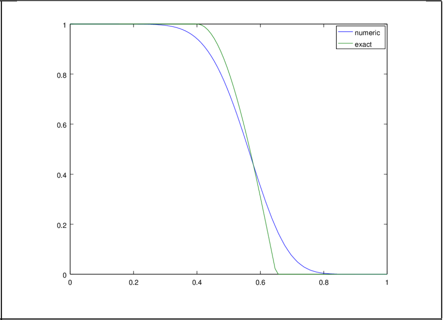
- IC shifts from left to right (a – speed) – known solution

	
N = 25, cfl = 0.2	N = 100, cfl = 0.2

Numerical convergence test:

N	25	50	100	200	400	800	1600
err	0.034875	0.016674	0.0090165	0.0048615	0.0015817	6.2515e-04	2.6352e-04
conv		1.06459	0.88696	0.89117	1.61993	1.33920	1.24629

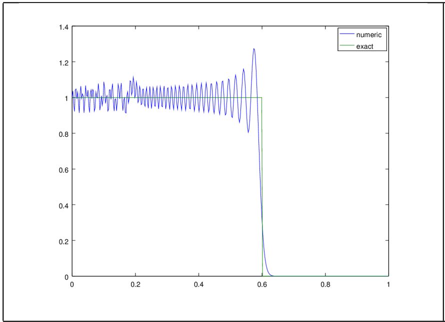
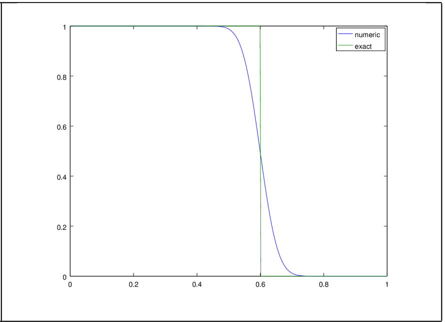
Tests, advection eq, cos wave, LF scheme

	
N = 25, cfl = 0.5	N = 100, cfl = 0.5

Numerical convergence test:

N	25	50	100	200	400	800	1600
err	0.10452	0.074538	0.031533	0.021360	0.0098777	0.0050463	0.0022630
conv		0.48773	1.24111	0.56195	1.11266	0.96895	1.15699

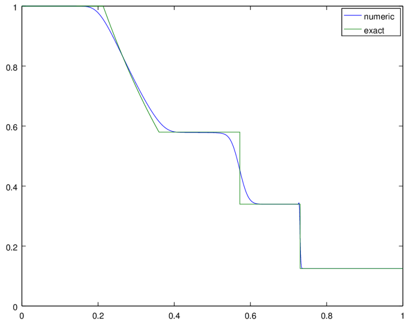
Tests, advection eq, step IC

	
BTCS, N = 400, cfl = 0.2	LF, N = 400, cfl = 0.5

Riemann test

Euler equation of hydrodynamics, ideal gas
IC piecewise constant, one discontinuity

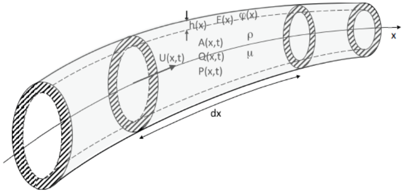
$$\begin{array}{ccc} x \in (0.0, 1.0) & x_0 = 0.3 & T = 0.2 \\ \varrho_l = 1.0 & u_l = 0.75 & p_l = 1.0 \\ \varrho_r = 0.125 & u_r = 0.0 & p_r = 0.1 \end{array}$$



LF, N = 1000, simulation fragile

Arterial pulse wave model

- 1D model of blood flow in elastic artery (all quantities assumed constant on cross section)
- artery described by
  - $A(x,t)$ .. cross section of vessel
  - $U(x,t)$ .. velocity of blood (average over the crossection)
  - $P(x,t)$ .. pressure
- left BC – blood flow from heart (ODE model)
- pulse caused by the systole propagates along the artery



Conclusion

- PDE extension was studied and enhanced
- Suitable numerical methods were proposed and tested
- Discretization module in front end will be implemented

End

Thank you

- Peter Fritzson.
  Principles of Object-Oriented Modeling and Simulation with Modelica 2.1.
  Wiley-IEEE Press, 2004.

- Levon Saldamli.
  A High-Level Language for Modeling with Partial Differential Equations.
  PhD thesis, Department of Computer and Information Science, Linköping University, 2006.