Solving ODEs and PDEs in MATLAB

Sören Boettcher

16.02.2009

Introduction

- Quick introduction to MATLAB syntax
- ODE in the form of Initial Value Problems (IVP)
  - what equations can MATLAB handle
  - how to code into MATLAB
  - how to choose the right MATLAB solver
  - how to get the solver to do what you want
  - how to see the result(s)
  - several examples
- Boundary Value Problems (BVP)
- Delay Differential Equations (DDE)
- Partial Differential Equations (PDE)
- NOT todays topic: numerical methods, ODE, BVP, DDE, PDE or MATLAB

Problem

- DEs are functions of one or several variables that relate the values of the function itself and of its derivatives of various orders
- An ODE is a DE in which the unknown function is a function of a single independent variable

$$y' = f(t, y) \tag{1}$$

- In many cases, a solution exists, but the ODE may not necessarily be directly solvable. Instead, the solution may be numerically approximated using computers
- There are many numerical methods in use to solve (**??**), but one has to use the right solver in order to obtain good solutions

The MATLAB ODE Solvers

Explicit methods for nonstiff problems:

- `ode45` - Runge-Kutta pair of Dormand-Prince
- `ode23` - Runge-Kutta pair of Bogacki-Shampine
- `ode113` - Adams predictor-corrector pairs of orders 1 to 13
- `ode15i` - BDF

Implicit methods for stiff problems:

- `ode23s` - Runge-Kutta pair of Rosenbrock
- `ode23t` - Trapezoidal rule
- `ode23tb` - TR-BDF2
- `ode15s` - NDF of orders 1 to 5

All these methods have built-in local error estimate to control the step size; codes are found in the `/toolbox/matlab/funfun` folder

Basic usage for MATLAB's solvers

- Apply a solver:
  `[t,y] = solver(@odefun, time_interval, y0, options)`
- `odefun` - a function handle that evaluates the right side of the differential equations.
- `time_interval` - a vector specifying the interval of integration.
  `[t0,tf]` - initial and final value
  `[t0,t1,...,tn]` - evaluation of the method at certain points
- `y0` - a vector of initial conditions.
- `options` - structure of optional parameters that change the default integration properties.

Approach

- Consider the IVP:

$$y'' + y' = 0,\ y(0) = 2,\ y'(0) = 0$$

- Rewrite the problem as a system of first-order ODEs:

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -y_1 \end{aligned}$$

- Code the system of first-order ODEs:

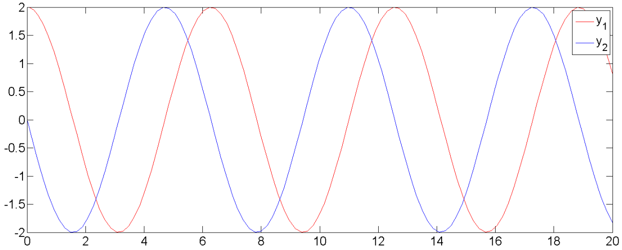
```
function dy_dt = odefun(t,y)
dy_dt = [y(2); -y(1)];
```

- Apply a solver to the problem:

```
[t,y] = ode45(@odefun, [0,20], [2,0]);
```

- The algorithm selects a certain partition of the time interval and returns the value at each point of the partition.

Solution of harmonic oscillation



- View the solver output:

```
plot(t, y(:,1),'r',t,y(:,2),'b')
title('Solution of van der Pol Equation);
xlabel('time t'); ylabel('solution y');
legend('y_1','y_2')
```

Options

- Several options are available for MATLAB solvers.
- The `odeset` function lets you adjust the integration parameters of the following ODE solvers.
- Save options in `opts`
  `opts=odeset('name1','value1','name2','value2',...)`
- Expand `opts`
  `opts=odeset(old_opts,'name','value')`
- If no options are specified, the default values are used.

The ODESET Options

name	meaning	default value
`RelTol`	relative error tolerance	10^{-3}
`AbsTol`	absolute error tolerance	10^{-6}
`Refine`	output refinement factor	1 (4)
`MaxStep`	upper bound on step size	
`Stats`	display computational cost statistics	`off`

The estimated error in each integration step satiesfies

$$e_k \leq \max\{RelTol \cdot y_k, AbsTol\}$$

whereas y_k the approximation of $y(x_k)$ at step k

Example of efficiency differences

- Van der Pol oscillator as a system of first-order ODEs:

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= \mu(1 - y_1^2 y_2 - y_1) \end{aligned}$$

- as a function with $\mu = 1000$:

```
function dy_dt = vdp(t,y,mu)
dy_dt = [y(2); mu*(1-y(1).^2).*y(2)-y(1)];
```

- Apply a solver (takes 123 seconds)

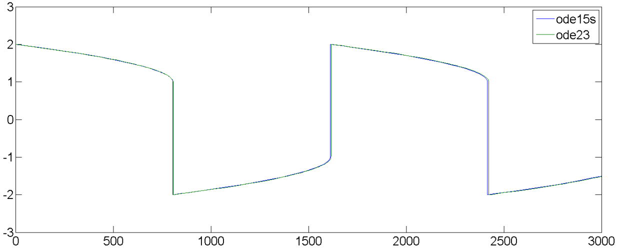
```
[t,y]=ode23(@(t,y)vdp(t,y,1000),[0,3000],[2,0]);
```

- Different solver (takes 56 milliseconds)

```
[t,y]=ode15s(@(t,y)vdp(t,y,1000),[0,3000],[2,0]);
```

Example of efficiency differences

Although the above function is stiff for large μ, `ode23` has almost achieved the same result as `ode15s`



Example for false solver

- Simple ODE:

$$y' = \sin(1000t),\ y(0) = 1.2$$

- Analytic solution:

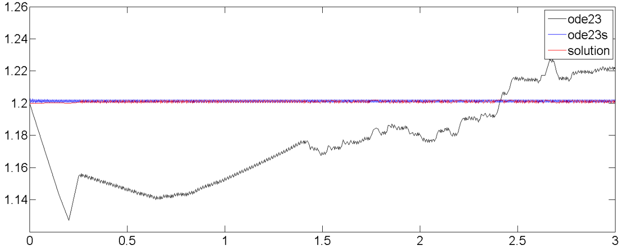
$$y(t) = \frac{-\cos(1000t) + 1201}{1000}$$

- 2 different solvers, one for stiff ODEs:

```
[t,y]=ode23(@(t,y)sin(1000*t),[0,3],1.2);
[t,y]=ode23s(@(t,y)sin(1000*t),[0,3],1.2);
```

Example for false solver

`ode23` is totally wrong, `ode23s` makes it well



Solving BVPs with MATLAB

- BVPs can have multiple solutions and one purpose of the initial guess is to indicate which solution you want. The 2nd order DE

$$y'' + |y| = 0$$

has exactly two solutions that satisfy the boundary conditions

$$y(0) = 0,\ y(4) = -2$$

- DE for boundary value

```
function dy_dx = bvpex(x,y)
dy_dx = [y(2); -abs(y(1))];
```

- Evaluate residual of boundary condition

```
function res = bc(ya,yb)
res = [ ya(1); yb(1) + 2];
```

- Apply a solver:

```
solinit = bvpinit(linspace(0,4,5),[-1 0]);
sol = bvp4c(@bvpex,@bc,solinit);
```

Solving DDEs with MATLAB

- A DDE is a DE in which the derivative of the unknown function at a certain time is given in terms of the values of the function at previous times.
- Consider the problem

$$y'(t) = \frac{2y(t-2)}{1+y(t-2)^{9.65}} - y(t), \;\; t \in [0, 100], \;\; y(t) = 0.5 \text{ for } t < 0$$

- Code the function:

```
function dy_dt = ddes(t,y,z)
dy_dt = 2*z/(1+z^9.65)-y;
```

- Apply a solver:

```
sol=dde23(@ddes,2,0.5,[0,100])
plot(sol.x,sol.y)
```

Numerical Solution of PDEs with MATLAB

A PDE is a DE in which the unknown function is a function of multiple independent variables and their partial derivatives.

	solver	nonlinear	system
1D	`pdepe`	✓	✓
2D	`pdenonlin` (`elliptic`)	✓	×
	`parabolic`	×	×
	`hyperbolic`	×	×
3D	×	×	×

Specifying an IVBP

- pdepe solves PDEs of the form

$$\mu(x, t, u, u_x)u_t = x^{-m}(x^m f(x, t, u, u_x))_x + s(x, t, u, u_x)$$

- $x \in (a, b),\ a > 0,\ t \in [t_0, t_f],\ m = 0, 1, 2,\ \mu \geq 0$
- The problem has an initial condition of the form

$$u(x, t_0) = \Phi(x),\ x \in [a, b]$$

- The boundary conditions are

$$\begin{aligned} p(a, t, u(a, t)) + q(a, t)f(a, t, u(a, t), u_x(a, t)) &= 0,\ t \geq t_0 \\ p(b, t, u(b, t)) + q(b, t)f(b, t, u(b, t), u_x(b, t)) &= 0,\ t \geq t_0 \end{aligned}$$

Example

- Consider the PDE

$$\pi^2 u_t = u_{xx},\ x \in (0,1),\ t \in (0,2]$$

- with boundary conditions

$$u(0,t) = 0,\ u_x(1,t) = -\pi \exp(-t)$$

- and initial conditions

$$u(x,0) = \sin(\pi x)$$

- The exact solution for this problem is

$$u(x,t) = \exp(-t)\sin(\pi x)$$

Example

- The specification of the problem for solution by pdepe is

$$
\begin{aligned}
& m = 0,\ a = 0,\ b = 1,\ t_0 = 0,\ t_f = 2, \\
& \mu(x,t,u,u_x) = \pi^2,\ f(x,t,u,u_x) = u_x,\ s(x,t,u,u_x) = 0, \\
& p(a,t,u(a,t)) = u(a,t),\ q(a,t) = 0, \\
& p(b,t,u(b,t)) = \pi \exp(-t),\ q(b,t) = 1, \\
& \Phi(x) = \sin(\pi x)
\end{aligned}
$$

Solving an IBVP

- The syntax of the MATLAB PDE solver is
  `sol=pdepe(m,pdefun,icfun,bcfun,xmesh,tspan)`
- `pdefun` is a function handle that computes μ, f and s
  `[mu,f,s]=pdefun(x,t,u,ux)`
- `icfun` is a function handle that computes Φ
  `phi=icfun(x)`
- `bcfun` is a function handle that computes the BC
  `[pa,qa,pb,qb]=bcfun(a,ua,b,ub,t)`
- `xmesh` is a vector of points in $[a, b]$ where the solution is approximated
- `tspan` is a vector of time values where the solution is approximated
- `sol` is a 3D array where `sol(i,j,1)` is the solution at `tspan(i)` and `xmesh(j)`

References

- Books
  - Coombes et al.; Differential Equations with MATLAB
  - Cooper; Introduction to PDEs with MATLAB
  - Fansett; Applied Numerical Analysis using MATLAB
  - Moler; Numerical Computing with MATLAB
  - Shampine et al.; Solving ODEs with MATLAB
  - Stanoyevitch; Introduction to ODEs and PDEs using MATLAB
- Papers
  - Shampine, Reichelt; The MATLAB ODE Suite
  - Shampine et al.; Solving BVPs for ODEs in MATLAB with `bvp4c`
- MATLAB Help

Exercises

- $$y' + 2y = \sin(t) + \exp(-5t),\ y(0) = 0$$
- $$y' + y\cot(t) = 5\exp(\cos(t)),\ y(0) = 0$$
- $$y'' - 4y' + 5y = \exp(-2t)\tan(t),\ y(0) = 0,\ y'(0) = 0$$
- $$2y'' + y = 2\tan(t),\ y(0) = 0,\ y'(0) = 1$$

Thank you for your attention.