How is ODE Software Used and What are the Implications for Software Developers

W.H. Enright

Department of Computer Science

University of Toronto

Toronto, Canada M5S 3G4

enright@cs.utoronto.ca

ANODE 2001

Auckland, January 2001

In the last decade there has been a significant change in the way scientific computing is carried out. Those of us developing ODE software must be aware of this change if our software is is ever going to be widely used and have an impact. Two environments or areas where our software is now widely used are:

- Large Scale Scientific Computation: Characterized by simulations generating massive amounts of data and involving expensive runs on High Performance distributed systems (often located at remote sites). Applications require data compression/ interactive data viewing (or data mining)/ remote visualization/distributed collaboration of experts.

- Problem Solving Environments (PSEs): Often where students are first exposed to scientific computing and where scientists and engineers formulate and solve their problems.

In this talk we will focus on the latter environment but many of our observations also apply to the former environment.

Outline of Presentation:

1. What is a PSE ?

2. A typical application/problem 'solved' in a PSE.

3. Implications for software developers.

4. Two complementary approaches for delivering such software.

5. Implications for developers of traditional HP libraries.

6. Additional tools/packages needed for PSEs.

II) What is a PSE ?

- Focus is on visualizing approximate solution. This generally involves graphical representation of a 'subset' of the results. (eg. Phase plane or standard plots.)

- Stringent accuracy not likely, but off-mesh accuracy is required.

- Interface and documentation must be simple and structured in a hierarchichal fashion. (Hide information unless necessary to define the problem.)

- Ease of use as important as efficiency.

$\boxed{\text{II) A typical PSE problem/application:}}$

A predator-prey relationship can be modeled by the well-known IVP:

$$y_1' = y_1 - 0.1y_1y_2 + 0.02x$$

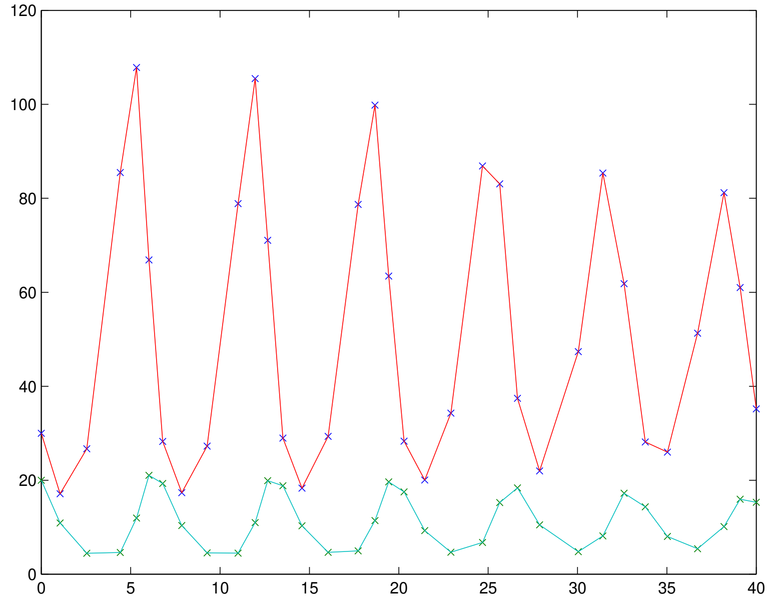
$$y_2' = -y_2 + 0.02y_1y_2 + 0.008x$$

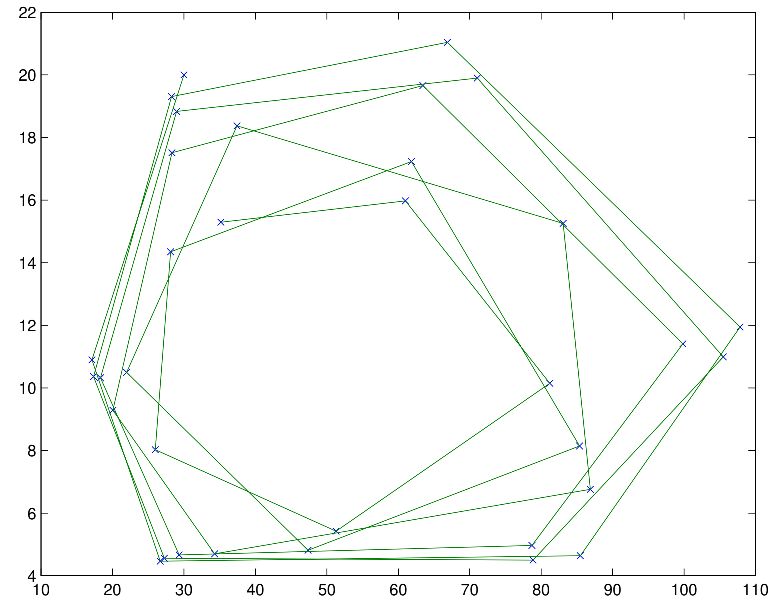
with

$$y_1(0) = 30, \;\; y_2(0) = 20,$$

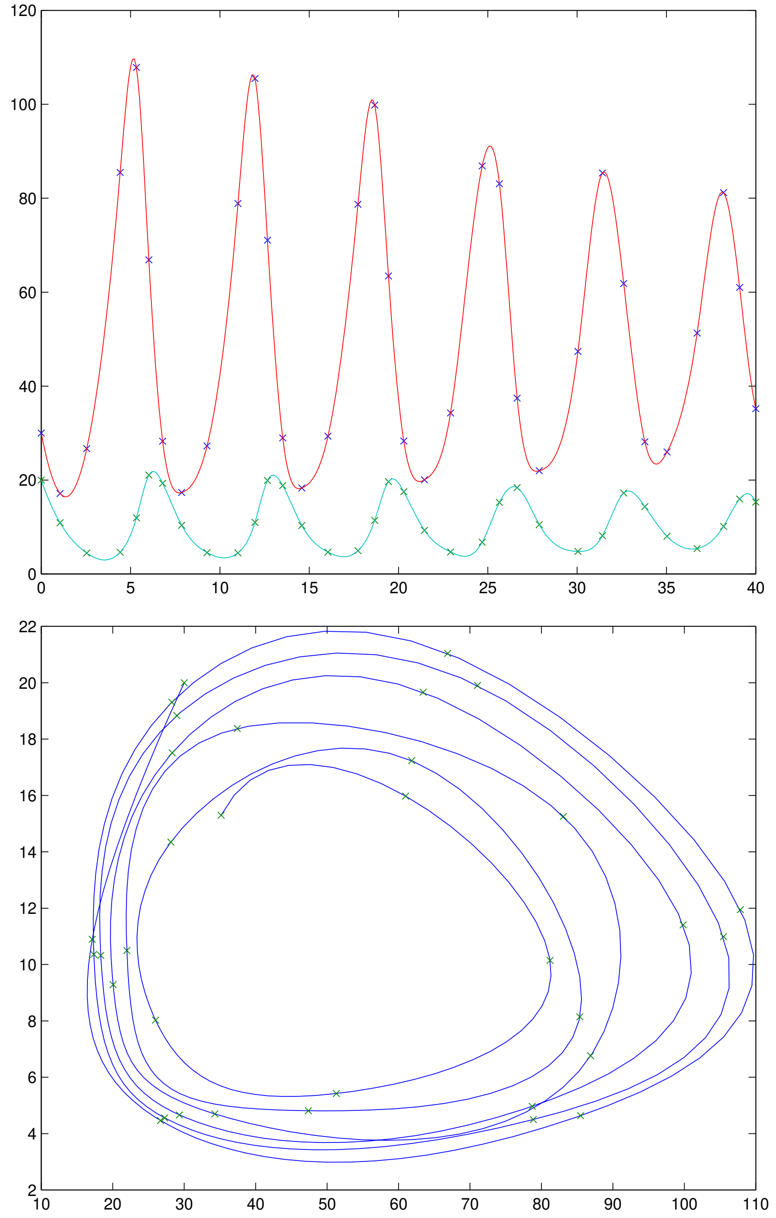
where $y_1(x)$ represents the 'prey' population at time x and $y_2(x)$ represents the 'predator' population at time x. The solution can then be visualized as a standard x/y solution plot or by a 'phase plane' plot. Consider the use of a standard ODE software package such as an 8th order CRK method.

A) Visualization Using Discrete Solution:

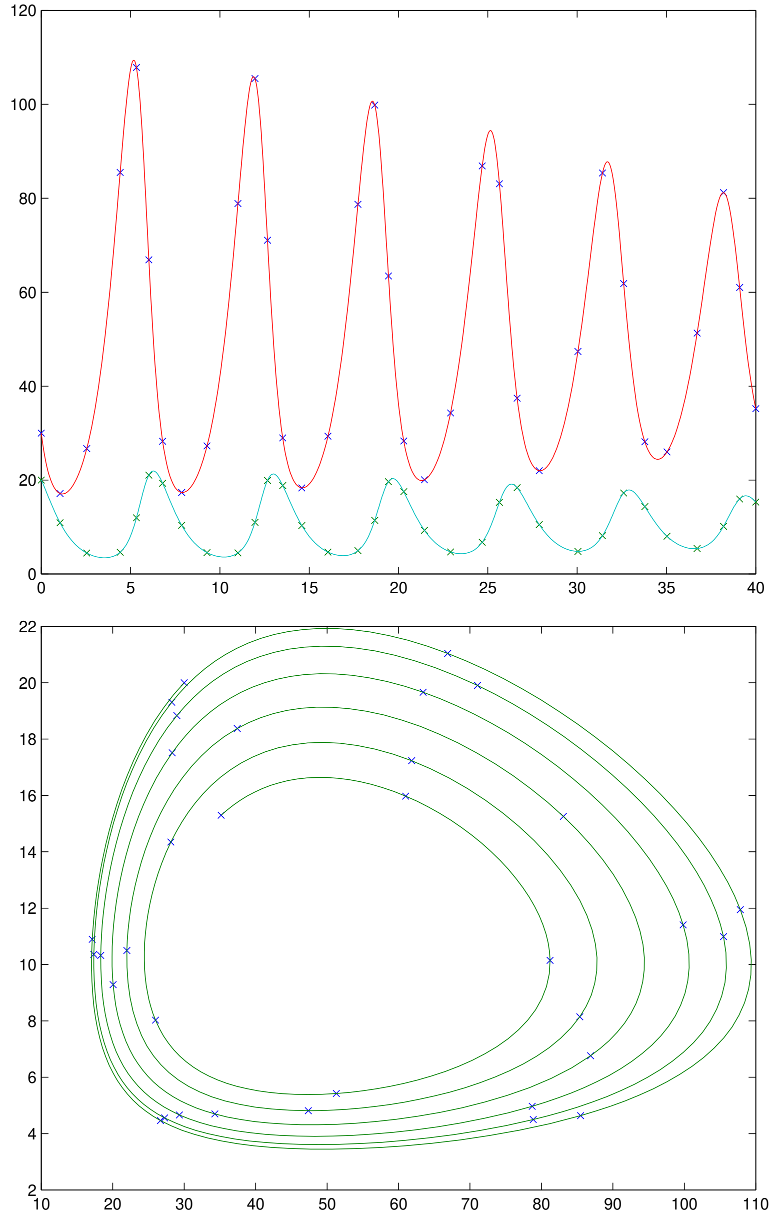




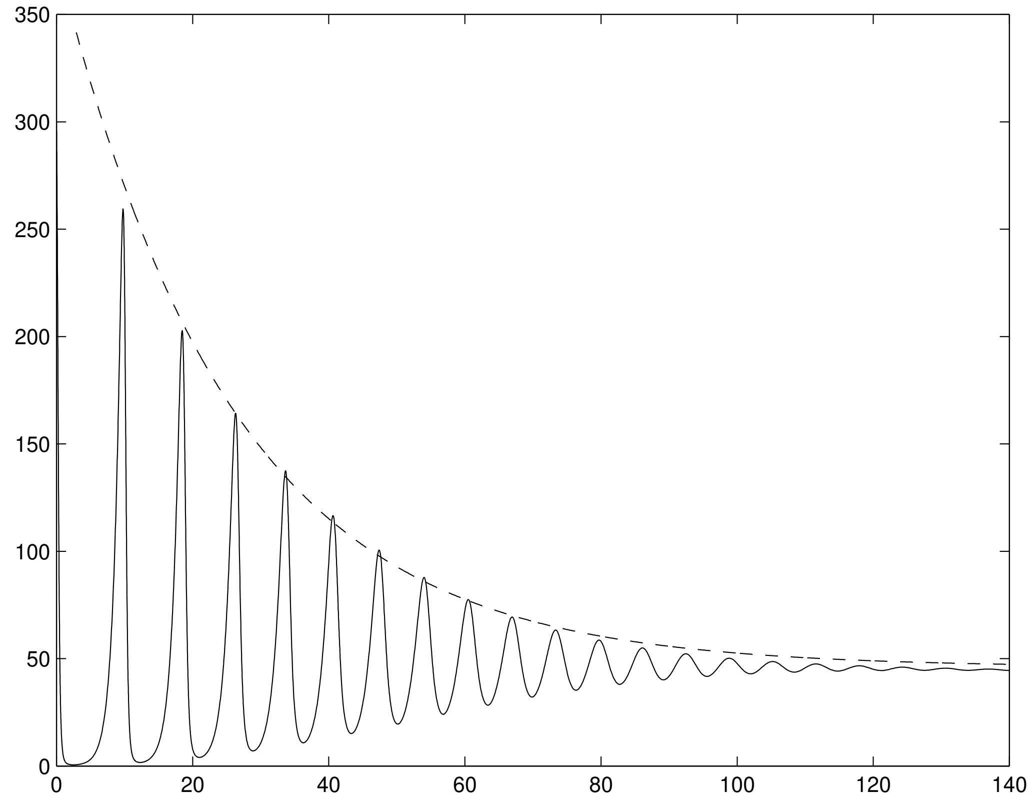
B) Visualization Using Splines:



C) Visualization Using Continuous Extension:



A biologist may be interested in whether the solutions to this equation are 'almost periodic' (in the sense that the difference between successive maximum is constant) and whether the local maximums approach a steady state exponentially.



III) Implications for developers of ODE software:

- Robustness as important as reliability. (Its OK to fail early – just do it gracefully.)

- Adopt a standard interpretation of error control, and a standard program interface (calling sequence).

- Options and additional parameters (if required) are specified in the same way for all methods. This includes mesh-constraints, initial guess of solution, specification of partial derivatives, and accuracy specification.

- Adopt a standard representation of the approximate solution. (eg. a vector of piecewise polynomials.)

IV) Two approaches for delivering such software:

1. "Wrapping" existing HP program libraries and using environment- specific APIs. (Recent IFIP WG2.5 Working Conference 8 on Software Architecture for Scientific Computing Applications, Ottawa,October 2000.)

2. Development of low-overhead, low-order, new methods completely within the PSE. (Recent sequence of improvements to ODE section of MATLAB: [1], [2],[3] .)

There is a need for both approaches as they complement each other. The latter is ideal for rapid prototyping and numerical experiments while the former is often required for final "production" runs and large scale HP applications.

$\boxed{\text{V) Implications for developers of HP libraries:}}$

- Stepsize and error control strategies must be effective for both coarse and fine meshes. That is, the strategies should be asymptotically 'correct' and otherwise 'plausible'.

- Early detection of potential numerical difficulties is important. (Examples of such difficulties include the lack of convergence of an iteration; a discontinuity in a derivative; and an error in the specification of analytic partial derivatives). Action taken may involve the use of alternate strategies or a return with warning message/flag.

VI) Additional tools/packages needed in a PSE:

A PSE provides tools that make it easy to:

1. Formulate a mathematical model of the problem.

2. Approximate the solution of the mathematical problem.

3. Visualize the approximate solution.

4. Validate (or verify) an approximate solution.

5. Validate (or verify) a mathematical model.

If the underlying numerical method provides a continuous approximation to the solution (over the domain of interest) generic tools for 4. can be developed.

Examples of such tools that would be useful and easy to provide are:

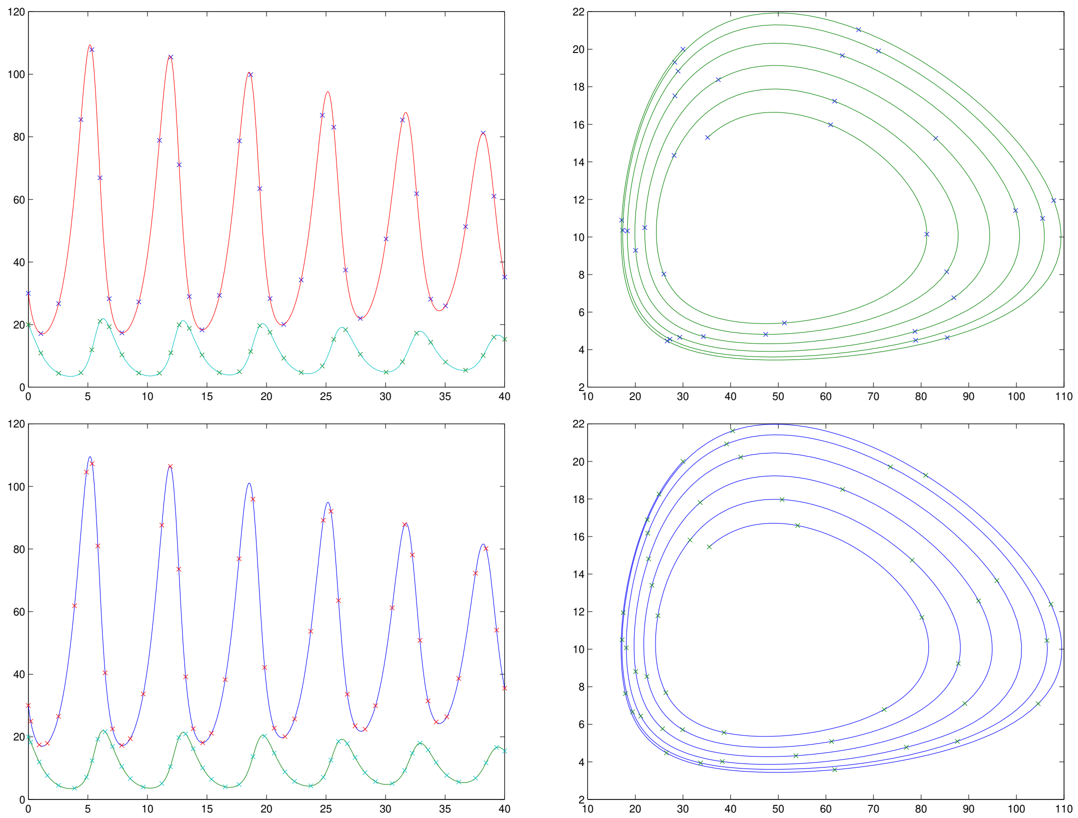
- Generate guaranteed error bounds using an (expensive) validated ODE solver.

- Solve the same ODE with an unrelated method or a more stringent error tolerance and check the difference between the two measured over a selection of predetermined (or random) sample points.

- Sample the defect (or residual) over a selection of predetermined (or random) sample points.

- Estimate the global error by integrating a 'nearby' problem where the exact solution is known. (Using the perturbed ODE, $z' = f(t, z) + \delta(t)$.)

Consider approximate solutions to the predator/prey problem with the 8th order CRK method and the built-in MATLAB method, ode45. Visualizing using the piecewise polynomial representation of the corresponding approximate solutions we have:



$\boxed{\text{Implications and Conclusions:}}$
Numerical ODE software, if it to relevant and used in scientific computation must satisfy:.

- It must provide accurate off-mesh approximations. (The numerical solution, $u(t)$, should be a piecewise approximation.)

- There should be a standard interface:

  - Minimal calling sequence with common set of 'options' and 'defaults'.

  - uniform interpretation of the error control and the 'expectations' of the accuracy.

  - Standard set of warnings and error flags.

- A generic environment independent representation of the solution and the problem should be available.