Continuous Simulation with Ordinary Differential Equations
Seminar - Modeling and Simulation

Julian Busch
9busch@informatik.uni-hamburg.de

University of Hamburg
Department of Informatics
Scientific Computing

December 3, 2012
Outline

Introduction

Continuous Simulation

Ordinary Differential Equations
  Differential Equations
  Numerical Methods

Application: N-body Simulation

Conclusion
Introduction

- Changes in quantities can accurately be described with derivatives and be related to other quantities via equations
- Differential equations are a natural way to describe dynamically changing systems
- They arise in many different contexts such as
  - Physics & astronomy (celestial mechanics)
  - Geology (weather modeling)
  - Chemistry (reaction rates)
  - Biology, social sciences, economics, ...
- We will focus on the numerical simulation of systems modeled with Ordinary differential equations
Recap:

- **Continuous System**: input, output and state variables are defined over a range of time
- **Discrete Systems**: input, output and state variables are defined for $t_0, t_1, t_2, \ldots$
- **Lumped Models**: only one independent variable (time)
- **Distributed Model**: more than one independent variable
Continuous Simulation

- Ordinary Differential Equations model Continuous and Lumped Systems
- Simulating a system means to solve its mathematical model and predict its behavior in different situations
- Many important differential equations cannot be solved exactly
  - Numerical methods are employed
  - The model is discretized
Example

Falling Ball:

- A ball is falling from a height of 100 m.
- When does it reach the ground?
- Assumption: only gravitational force is acting
- Model:

\[ v'(t) = -9.8 \]

\[ x'(t) = v(t) \]

Initial conditions: \( x(0) = 100, \ v(0) = 0 \)
Example

Solution:

\[ x(t) = 100 - 0.5(9.8)t^2 \]
\[ v(t) = -9.8t \]

System can be simulated: vertical position and speed of the ball is given for any time \( t \)

The ball reaches the ground at \( t = 4.5175 \) s with a velocity \( v = -44.2719 \) m/s
Definition
An ordinary differential equation (ODE) is an equation containing a function of one independent variable and its derivatives.

- Ordinary: no partial derivatives
- Examples:
  
  \[ y'(t) = y(t), \]
  
  \[ x''(t) = \frac{F(t, x(t))}{m} \] (Newton’s Second Law)
Ordinary Differential Equations

- General form:

\[ F \left( t, y, y', \ldots, y^{(n-1)} \right) = y^{(n)} \]

where \( y = y(t) \) and \( y^{(n)} \) denotes the \( n \)th derivative of \( y \)

- \( t \) is called \textit{independent variable} or \textit{time variable}

- \( n \) is the \textit{order} of the equation

- A \textit{system} of coupled differential equations is specified, if \( y \) is a vector of functions and \( F \) is a vector valued function
Order Reduction

- Most solvers expect first order equations
- Higher order equations can be reduced to an equivalent first order system by introducing new functions for the derivatives:

\[
\begin{pmatrix}
y'_1 \\
y'_2 \\
\vdots \\
y'_{n-1} \\
y'_n
\end{pmatrix} =
\begin{pmatrix}
y_2 \\
y_3 \\
\vdots \\
y_n
\end{pmatrix} +
\begin{pmatrix}
F(t, y_1, \ldots, y_n)
\end{pmatrix}
\]
Some classes of ODEs have special properties

- **Autonomous:** $F$ does not depend on $t$, also named Time-invariant system
- **Linear:** $F$ can be written as a linear combination of the derivatives of $y$

\[
y^{(n)} = \sum_{i=0}^{n-1} a_i(t)y^{(i)} + r(t)\]

- **Homogeneous:** $r(t) = 0$
- **Inhomogeneous:** $r(t) \neq 0$

Determining stability of solutions is relatively easy
A solution is a function $u$ that is $n$ times differentiable and satisfies

$$F \left( t, u, u', \ldots, u^{(n-1)} \right) = u^{(n)}$$

It can be defined on all of $\mathbb{R}$ (global solution) or only on a maximal time interval (maximal solution)

Equations without additional conditions have a general solution that contains a number of independent constants

- ODE describes the slope of a solution, not the actual values
The constants can be set to specific values to fulfill additional conditions, yielding a *particular solution*

- **Initial value problem:** specifies a particular solution by a given initial value \( y(t_0) = y_0 \)

- **Boundary value problem:** conditions for more than one point are given, typically specified for the endpoints of some time-interval

- We will focus on Initial value problems
Some but not all ODEs have solutions that can be written in exact and closed form.

Several techniques for solving exist, for example:
- Direct integration
- Separation of variables
- Laplace transform
- Specialized methods for classes of equations
Example

- Consider the equation \( y' = y \)
- If \( y \neq 0 \) on its whole domain, we can write it as \( \frac{y'}{y} = 1 \) so that

\[
\int \frac{y'}{y} \, dt = \int 1 \, dt = t + C
\]

- Since an antiderivative of \( \frac{y'}{y} \) is \( \ln|y| \), we get

\[
\ln|y| = t + C \quad \text{or} \quad y = \pm e^C \cdot e^t
\]

- Solutions have the form \( y = Ae^t \) (\( A \neq 0 \))
- \( y = 0 \) is a solution for \( A = 0 \)
Example

- A given initial value \( y(0) = y_0 \) specifies the particular solution \( y = y_0 e^t \)
Initial Value Problems

- Given: the initial state $y_0$ of a system at time $t_0$ and an ODE that determines its evolution
- Want: a function $y(t)$ that describes the state of the system as a function of time
- Most numerical methods expect first order equations as an input:
  \[ y' = F(t, y), \quad y(t_0) = y_0 \]
- The IVP has a unique solution, provided $F$ is sufficiently smooth (continuous in $t$ and Lipschitz-continuous in $y$)
Numerical Methods

- Many important problems cannot be solved analytically
- Goal: predict future values of $y$ by simulating the system's behavior
- Calculate a sequence of approximations $y_1, y_2, y_3, \ldots$ for $y$ at consecutive points in time $t_0 + h, t_0 + 2h, t_0 + 3h, \ldots$
- $h$ is called *step size*
Euler’s Method

- Probably the most simple and popular method
- Trivial case of several more general techniques
- Consider the Taylor expansion of $y$ around $t_n$:

$$y(t_n + h) = y(t_n) + hy'(t_n) + \frac{1}{2}h^2y''(t_n) + O(h^3)$$

- Euler’s method uses the first two terms as an approximation for $y(t_{n+1})$:

$$y_{n+1} = y_n + hF(t_n, y_n)$$
Euler’s Method

Geometrical description:

- Start at initial value
- Take small steps along the tangent lines through the previous approximations
Stability of Solutions

- Roughly speaking, the stability for an ODE reflects the sensitivity of its solution to perturbations.
- If the solutions are stable, they converge with time so that perturbations are damped out.
- If the solutions are unstable, they diverge with time so that perturbations will grow.
- When stepping from one approximation to the next, we land on a different solution from what we started from.
- The stability of the solutions has an influence on whether the incurred error grows or decreases with time.
Stiffness

- ODEs for which stable solutions converge too rapidly are called *stiff*
- Some methods are inefficient for stiff equations
- Example: Euler’s Method applied to the IVP

\[ y' = -2.3y, \quad y(0) = 1 \]

with step-sizes \( h = 1 \) and \( h = 0.7 \)
Stiffness

- Numerical solution for $h = 1$ oscillates and grows without bound, stiffness forces very small step-sizes

Analysis

Numerical Analysis offers several concepts to evaluate the quality of numerical methods:

- **Global truncation error**: difference between computed and true solution passing through initial value:
  \[ e_n = y_n - y(t_n) \]

- **Local truncation error**: error made in one step of the method
  \[ l_n = y_n - u_{n-1}(t_n) \]

  where \( u_{k-1} \) is the solution passing through the previous approximation

- Want: small global error, but can only control local error
Analysis

- **Order:** method has order $p$, if

$$l_n = O(h^{p+1})$$

(How much does the local error decrease with the step-size?)

- **Stability:** method is stable if it produces stable solutions, so that errors are not magnified
  - Stability depends on the stability of the ODE being solved, the method itself and the step-size
  - A method with low stability can produce high global errors despite a high order
  - Several different definitions exist
Analysis of Euler’s Method

- Euler’s Method has Order 1 (compare to Taylor series)
- It can produce unstable solutions (as seen before)
- Not effective for stiff equations
Backward Euler Method

- Use $F(t_{n+1}, y_{n+1})$ instead of $F(t_n, y_n)$:

  $$y_{n+1} = y_n + hF(t_{n+1}, y_{n+1})$$

- Need to solve an algebraic equation
- Fixed-point iteration or Newton’s method often used
- Starting guess for iteration can be obtained from explicit method or previous solution
- Same order as Euler’s Method
- But: better stability and effective for stiff equations
Generalizations

- Exploit values available upon reaching $t_n$: $y_n, y_{n-1}, \ldots$ and $F(t_n, y_n), F(t_{n-1}, y_{n-1}), \ldots$

- **Explicit methods** use information at time $t_n$ for the solution at time $t_{n+1}$
  - e.g. Euler Method

- **Implicit methods** use information at time $t_{n+1}$
  - Needs to evaluate $F$ with argument $y_{n+1}$ before its value is known
  - Generally more stable than comparable explicit methods
  - e.g. Backward Euler Method

- Important classes of methods: Backward Differentiation Formulas, Adams Methods, Runge-Kutta Methods
Trapezoidal Rule

- Implicit second order method
- Combines Euler & Backward Euler:

\[ y_{n+1} = y_n + \frac{h}{2} \left( F(t_n, y_n) + F(t_{n+1}, y_{n+1}) \right) \]

- Starting guess for \( y_{n+1} \) can be provided by an explicit method
- Correct it with the implicit formula, either repeatedly (fixed point iteration) or a fixed number of times
- The two methods are called a predictor-corrector pair
Heun’s Method

- Heun’s Method results from predicting with Euler’s Method and correcting once with the Trapezoidal rule:

\[ p_{n+1} = y_n + hF(t_n, y_n) \]

\[ y_{n+1} = y_n + \frac{h}{2} \left( F(t_n, y_n) + F(t_{n+1}, p_{n+1}) \right) \]

- Performing a single correction amounts to an explicit method
- Heun’s method has order 2
Example: Euler vs. Trapezoidal rule

Consider the IVP

\[ y' = -15y, \quad y(0) = 1 \]

with the exact solution \( y(t) = e^{-15t} \)

![Graph comparing Euler and Trapezoidal methods for solving the given IVP](http://en.wikipedia.org/wiki/File:StiffEquationNumericalSolvers.svg)
Adaptive step size

- It is appropriate to use a different size for each step to keep the local error below some tolerance level.
- Sometimes the step-size can be increased to save computation time.
- Sometimes it has to be reduced to ensure accuracy and stability.
- The Runge-Kutta-Fehlberg method RK45 achieves this by producing 5th- and 4th-order estimates.
  - The difference provides an estimate for the local error.
  - The step-size is then adapted to the error estimate.
  - ode45 in matlab/octave.
The gravitational N-body problem:

- Predict the motion of $N$ gravitationally interacting particles
- Applications range from systems of few bodies to solar systems and even systems of galactic and cosmological scale
- We will model our solar system, considering the Sun, the eight inner and outer planets and the dwarf-planet Pluto ($N = 10$)
Combining Newton’s Law of Universal Gravitation and the Second Law of Motion yields the following second order system:

\[ a_i = \frac{d^2 r_i}{d t^2} = G \cdot \sum_{j \neq i} \frac{m_j (r_j - r_i)}{\| r_j - r_i \|^3} \quad (i = 1, \ldots, N) \]

(consisting of \(3N\) equations, one per body and coordinate)

Initial values for the positions \(r_i\) and velocities \(v_i\) can be obtained from http://ssd.jpl.nasa.gov/?horizons
Before the presented numerical methods can be applied, the system must be reduced to a first order system.

We do so by introducing the first derivative of position, the velocity, to get the following first order system:

\[
\begin{align*}
\frac{dr_i}{dt} &= v_i \quad (i = 1, \ldots, N) \\
\frac{dv_i}{dt} &= a_i = G \cdot \sum_{j \neq i} \frac{m_j (p_j - p_i)}{\|p_j - p_i\|^3} \quad (i = 1, \ldots, N)
\end{align*}
\]

(consisting of now 6N equations)
For simplicity, we will use Euler’s method to solve the equations.

Starting with the initial values, a series of approximations for the position and velocity of all bodies is calculated.

The values after a time-step of length $h$ are given by:

$$r_i(t_{n+1}) = r_i(t_n) + hv_i(t_n) \quad \text{and} \quad v_i(t_{n+1}) = v_i(t_n) + ha_i(t_n)$$
Simulation

- The obtained values can be used to plot the bodies’ trajectories:
Conclusion

- Many different numerical methods for solving ODEs exist
- Higher order methods provide better accuracy but are more expensive
- When dealing with stiff ODEs, implicit methods should be employed (BDF, Adams-Moulton, ...)
- Most solvers use variable step-sizes
- The choice of an appropriate method for a given problem is crucial
Samir Al-Amer.  
Se207: Modeling and simulation.  

John C. Butcher.  
*Numerical Methods for Ordinary Differential Equations.*  

Florin Diacu.  
The solution of the n-body problem.  
Dimitry Gorinevsky.
Ee392m: Control engineering methods for industry.

Michael T. Heath.

Piet Hut and Michele Trenti.
N-body simulations (gravitational).
