

Lecture 15 - A

§7.1, 7.2, 7.3 Euler's Method

T. Takahashi

Department of Mathematics and Statistics
University of New Mexico



We will answer these questions:

- What are differential equations?
- In what way does a computer "solve" a differential equation?
- What is the difference between local and global error for ODE solvers?



Motivating Differential Equations



Types of Equations to Solve

An Algebraic Equation

Solve for $x \in \mathbb{R}$ in:

$$x^2 - 2x - 8 = 0.$$

An Ordinary Differential Equation

Solve for $u(t) \in C^2(\mathbb{R})$ in:

$$\frac{d^2u}{dt^2} - 2\frac{du}{dt} - 8 = 0.$$

The unknown u is a function which depends on time.

A Partial Differential Equation

Solve for $u(x, t) \in C^2(\mathbb{R} \times \mathbb{R})$ in:

$$\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = 0.$$

What is a Differential Equation?

Definition: ODE

An **ordinary differential equation** is an equation that relates a function $u(t)$ to its derivatives:

$$F(t, u, u', u'', \dots) = 0.$$

Examples

- First-order: $u' + u = 0$
- Second-order: $u'' - 3u' + u = 0$
- Nonlinear: $u' - u^2 - t = 0$

Why study DEs?

- Used to model systems that change over time or space
- Physics, biology, engineering, economics

Example: Snowball Effect

Model

Problem

Suppose a snowball with an initial radius $r = 1$ is rolled down from the top of a snowy mountain. Find an equation for the radius of the snowball at time t .

- This is a difficult problem
- The goal is to derive $r(t)$ = the radius of the snowball at time t
- We need to make simplifying assumptions

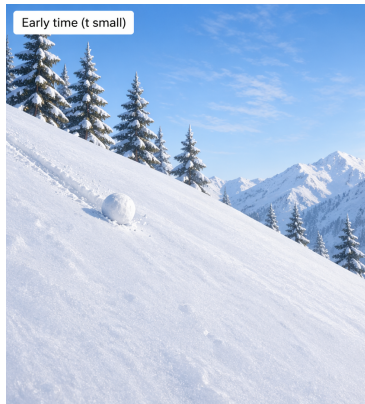
Question

How should the radius change over time? What influences the growth rate?
How do differential equations come into play?



Example: Snowball Effect

Assumptions



- Assume that mass increases at a rate proportional to its current size
- smaller snowball $\implies$ slower growth
- bigger snowball $\implies$ faster growth



Example: Snowball Effect

Model

Model for the Snowball

Let $r(t)$ be the radius at time t

$$\underbrace{\frac{dr}{dt}}_{\text{growth speed}} = k \underbrace{r}_{\text{current size}}$$

Constant $k > 0$ is a proportionality constant.

- If r is small, r' is small
- If r is large, r' is large

Definition: Model

A **mathematical model** is a differential equation whose solution describes a quantity of interest.

Example: Snowball Effect

Solution

- We let $t = 0$ be the time when the snowball begins to roll
- Since the initial radius is 1, we write $r(0) = 1$
- The full problem is to solve for the function $r(t)$ in

$$\begin{cases} \frac{dr}{dt} = kr, & t > 0 \\ r(0) = 1 \end{cases}$$

Solution

The exact solution is

$$r(t) = e^{kt}$$

The size of the snowball grows exponentially with rate k .

- *The focus of this class is to not solve DEs by hand, but numerically.*



Main Idea

1. Difficult Problem

Find the equation for the size of the snowball.

2. Simplifying Assumption

Snowball's size increases proportional to its current size.

3. Differential Equation

$$r' = kr$$

4. Solution

Radius of the snowball as a function of time

$$r(t) = e^{kt}$$



Initial Value Problems



- The model for the size of the snowball is

$$\begin{cases} r' = kr, & t > 0 \\ r(0) = 1 \end{cases}$$

- The equation $r' = kr$ is the **differential equation**
- The equation $r(0) = 1$ is called the **initial condition**
- A differential equation together with an initial condition is called an *initial value problem*



Initial Value Problems (IVPs)

Definition: IVP

An **initial value problem** is the task to solve the time-dependent function $u(t)$ given a differential equation and an initial condition.

$$\begin{cases} F(t, u, u', \dots) = 0, & t > 0 \\ u(0), u'(0), \dots \text{given}, & t = 0 \end{cases}$$

Examples:

1 First order IVP

$$\begin{cases} u' = f(t, u(t)), & t \in (0, T] \\ u(0) = u_0 \end{cases}$$

2 Second order IVP

$$\begin{cases} mu''(t) + cu'(t) + ku(t) = f(t), & t \in (0, T] \\ u(0) = A, u'(0) = B \end{cases}$$



Second Order Example

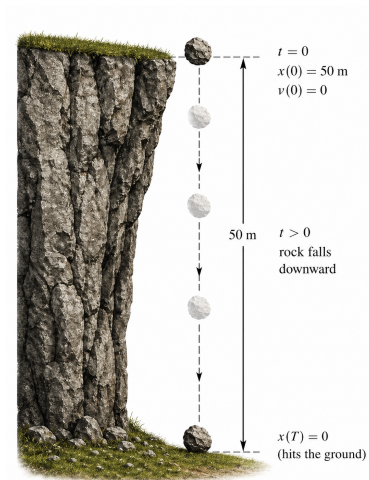
Problem

Suppose a rock with mass m is dropped from a cliff that is 50 meters high. What is the elevation of the rock as a function of time after its release?

Assumption

- No wind resistance
- Rock is a sphere
- Use Newton's Law

$$F = ma = -mg$$



Second Order Example

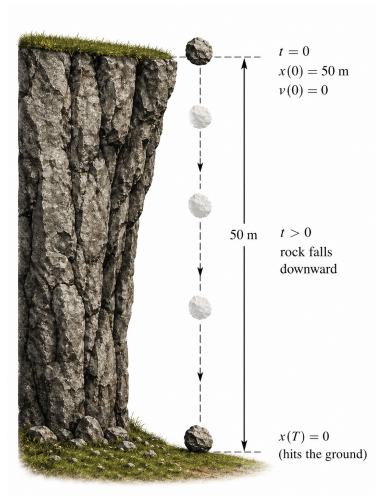
- Let $x(t)$ = elevation of the rock (up is positive).

Model

$$\begin{cases} x'' = -g, & t \in (0, T] \\ x(0) = 50, x'(0) = 0 \end{cases}$$

Solution

$$x(t) = 50 - \frac{1}{2}gt^2$$



Verifying a Solution

Question

If you were given the solution $u(t)$ to an IVP, how do you mathematically verify that it truly is a solution?

Steps:

- First, substitute $u(t)$ into the left-hand side (LHS) of the DE
- Next, substitute $u(t)$ into the right-hand side (RHS) of the DE
- Show that $\text{LHS} = \text{RHS}$
- Verify that the initial condition holds



Verifying a Solution 1

Example

Show that $r(t) = e^{kt}$ solves the IVP

$$\begin{cases} \frac{dr}{dt} = kr & t > 0 \\ r(0) = 1 \end{cases}$$

- LHS is

$$\text{LHS} = \frac{dr}{dt} = (e^{kt})' = ke^{kt}$$

- The RHS is

$$\text{RHS} = kr = k(e^{kt}) = \text{LHS}$$

- Then verify the initial condition $r(0) = e^{k0} = e^0 = 1$.



Verifying a Solution 2

Example

Show that $x(t) = 50 - \frac{1}{2}gt^2$ solves the IVP

$$\begin{cases} x'' = -g, & t \in (0, T] \\ x(0) = 50, \quad x'(0) = 0 \end{cases}$$

- LHS is

$$\text{LHS} = x'' = (50 - \frac{1}{2}gt^2)'' = -g$$

- The RHS is

$$\text{RHS} = -g = \text{LHS}$$

- Then verify the initial conditions $x(0) = 50$ and $x'(0) = 0$.



First-Order IVP

First-Order IVP

Given the derivative of a function $u'(t)$ for time $t \in (a, b]$ and an initial condition, find the antiderivative $u(t)$.

$$\begin{cases} u' = f(t, u(t)), & t \in (a, b] \\ u(a) = u_a \end{cases}$$

Theorem

If $f = f(t, u)$ and $\partial f / \partial u$ are continuous in the rectangle $\{|t - a| < \alpha, |u - u_a| < \beta\}$, then the first-order IVP has a unique continuous solution in some interval $|t - t_0| < \epsilon$.



Time-Stepping Algorithms



Solvable vs. Unsolvable

Solvable

- $x' = xt$
- $x'' + 4x' + 2x = 0$
- $t^2x'' + tx' + x = 0$

Unsolvable - No Closed-form Solution

- $x' = \cos(xt)$
- $x'' + x^2x' + x = 0$
- $\sqrt{x''} + x^2x' + \tan x = 0$

Only a small class of DEs can be solved by hand.



Numerical Solution of IVPs

The need for approximations

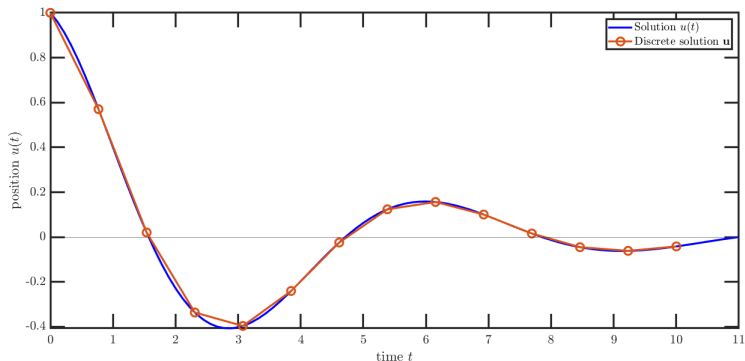
- Many important ODEs have no closed-form solution
- We can only solve a few types of ODEs by hand

Question

What does it mean to solve for the solution $u(t)$ on the computer?



Numerical Solution of IVPs



- Computers don't "solve" the differential equations
- They find a discrete approximation $\mathbf{u}$ of the continuous $u(t)$

Key Idea

We replace a continuous function with values at discrete time steps.

The Problem

First-Order Initial Value Problem

Consider the task of solving for the unknown function $u(t)$ in:

$$\begin{cases} u' = f(t, u(t)), & t \in (a, b] \\ u(a) = u_a \end{cases}$$

Solution

The solution to the IVP is a continuous function $u(t)$ defined on $t \in [a, b]$.

Objective

Since we cannot find $u(t)$ in general, our goal is to find a *discrete approximation* to u .



The Discrete Solution

- Discretize $[a, b]$ into equi-spaced points:

$$a = t_0 < t_1 < t_2 < \cdots < t_{n-1} < t_n = b$$

with $h = t_i - t_{i-1} = (b - a)/n$

- Our discrete numerical solution is

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_n \end{bmatrix} \approx \begin{bmatrix} u(t_0) \\ u(t_0 + h) \\ \vdots \\ u(t_0 + nh) \end{bmatrix}$$

- Initial condition is given: $\mathbf{u}_0 = u(t_0)$
- We want an algorithm to produce

$$\mathbf{u}_i \approx u(t_0 + ih)$$

for $i = 1, \dots, n$



Time-Stepping Algorithms

Goal

Construct the vector $\mathbf{u}$ whose elements are $\mathbf{u}_i \approx u(t_0 + ih)$.

Question

If you know $\mathbf{u}_0$, how would you approximate the solution at the next time step $\mathbf{u}_1$?

Main Idea: Use recursion

- Suppose you have an update formula

$$\mathbf{u}_{i+1} = T(\mathbf{u}_i)$$

where $T()$ is the update function

- We can recursively find all values of $\mathbf{u}_i$
- This is the main idea of **Time-Stepping algorithms**

Time-Stepping Algorithms

Goal

Construct the vector $\mathbf{u}$ whose elements are $\mathbf{u}_i \approx u(t_0 + ih)$.

Assume you know:

- Initial condition $\mathbf{u}_0$
- Update function T satisfying

$$\mathbf{u}_{i+1} \approx T(\mathbf{u}_i)$$

Do a time-stepping loop:

$$\mathbf{u}_1 \approx T(\mathbf{u}_0)$$

$$\mathbf{u}_2 \approx T(\mathbf{u}_1)$$

$$\vdots$$

$$\mathbf{u}_n \approx T(\mathbf{u}_{n-1})$$



Time Stepping Psuedocode

- In general the update function T may depend on u and t ,

$$T = T(u, t)$$

- Start by initializing an empty vector for the solution
- Create a discretization of time
- View time-stepping as an iterative method

```
1 initialize vector u
2 create time vector t
3 u(1) = u_0 % given data
4 for i = 1 : n - 1
5     u(i+1) = T(u(i), t(i))
```

Listing 1: Simple Time-stepping



Euler's Method



Use Taylor Expansion

Part 1

Goal

Come up with formula for $\mathbf{u}_{i+1}$ which depends on $\mathbf{u}_i$.

- Taylor's Expansion of $u(t_i + h)$

$$\begin{aligned}u(t_i + h) &= u(t_i) + hu'(t_i) + (h^2/2)u''(c), \quad t < c < t + h \\ &\approx u(t_i) + hu'(t_i)\end{aligned}$$

- Recall that we are trying to solve the first order IVP

$$\begin{cases} u' = f(t, u(t)), & t \in (a, b] \\ u(a) = u_a \end{cases}$$

- We can substitute $u'(t_i) = f(t_i, u(t_i))$



Use Taylor Expansion

Part 2

Goal

Come up with formula for $\mathbf{u}_{i+1}$ which depends on $\mathbf{u}_i$.

- Taylor's Expansion of $u(t_i + h)$ with f

$$\begin{aligned}u(t_i + h) &= u(t_i) + hu'(t_i) + (h^2/2)u''(c), \quad t < c < t + h \\&\approx u(t_i) + hu'(t_i) \\&= u(t_i) + hf(t_i, u(t_i))\end{aligned}$$

- Now make the discrete approximation $\mathbf{u}_i = u(t_i)$

$$\mathbf{u}_{i+1} = \mathbf{u}_i + hf(t_i, \mathbf{u}_i)$$

- This is called **Euler's Method**.



Method: Euler's Method

- **Given:** ODE $u' = f(t, u)$, initial condition u_0
- **Output:** Discrete approximation to $u(t)$
- **Idea:** Use the approximation $u(t + h) \approx u(t) + hf(t, u(t))$

- Euler's method is given by the recurrence equation

$$\mathbf{u}_{i+1} = \mathbf{u}_i + hf(t_i, \mathbf{u}_i)$$

- Very simple code

```
1 u(1) = u_a
2 for i = 1 : n
3     u(i+1) = u(i) + h * f(t(i), u(i))
```



Euler Example

(Not to be done by hand)

Example

Apply Euler's method with $h = 0.2$ to the IVP

$$\begin{cases} u' = tu + t^3, & t \in [0, 1] \\ u(0) = 1 \end{cases}$$

• $h = (b - a)/n \implies n = 5$

i	t_i	$\mathbf{u}_i$	$f(t_i, \mathbf{u}_i) = t_i \mathbf{u}_i + t_i^3$
0	0.0	1.0	0.0
1	0.2	1.0	0.2080
2	0.4	1.0416	0.4806
3	0.6	1.1377	0.89865
4	0.8	1.3175	1.5660
5	1.0	1.6306	

- The column u_i is the numerical solution

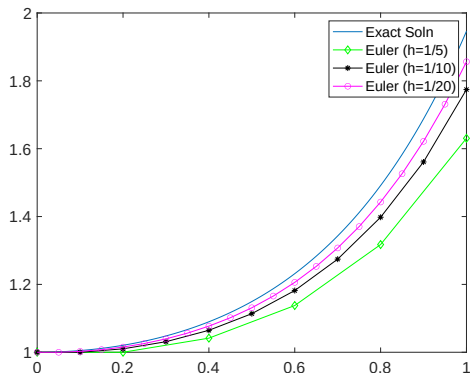


Euler Example

- Exact solution is

$$u(t) = 3e^{t^2/2} - t^2 - 2$$

- Euler's method with $h = 0.2, 0.1, 0.05$



- As h decreases, approximation *converges* to the exact solution



Error Analysis



Review: Truncation Error

Recall the Forward Difference Scheme

$$u'(t) \approx \frac{u(t+h) - u(t)}{h}$$

- Recall how **forward difference scheme** was derived

$$u(t+h) = u(t) + hu'(t) + (h^2/2)u''(c)$$

$$u(t+h) - u(t) = hu'(t) + (h^2/2)u''(c)$$

$$\underbrace{\frac{u(t+h) - u(t)}{h}}_{\text{approximate}} = \underbrace{u'(t)}_{\text{true}} + \underbrace{(h/2)u''(c)}_{\text{truncation term}}$$

- Truncation Error for Forward Difference:

$$E(h) = -\frac{h}{2}u''(c)$$

- Conclusion: the truncation error is $\mathcal{O}(h)$ so the accuracy is $\mathcal{O}(h)$



Truncation Error for Euler's Method

- Approximation of $u(t + h)$

$$u(t + h) = u(t) + hf(t, u(t))$$

- True value

$$u(t + h) = u(t) + hu'(t) + (h^2/2)u''(c)$$

- Truncation = True - Approx.

$$\begin{aligned} E_T(h) &= \left(u(t) + hu'(t) + (h^2/2)u''(c) \right) - (u(t) + hf(t, u(t))) \\ &= hu'(t) - hf(t, u(t)) + (h^2/2)u''(c) \\ &= \frac{h^2}{2}u''(c) \end{aligned}$$



Truncation Error for Euler's Method

Local Truncation Error

The Euler's method approximation for $u(t+h)$ has a local truncation error

$$E(h) = \frac{h^2}{2} u''(c) = \mathcal{O}(h^2)$$

Question

Does this mean Euler's method for IVPs converges quadratically?



Numerical Experiment

- Let's solve

$$\begin{cases} u' = tu + t^3, & t \in [0, 1] \\ u(0) = 1 \end{cases}$$

for $h = 1/10, 1/20, 1/40, 1/80, 1/160, 1/320, 1/640, 1/1280$.

- Let error be $E = |u(1) - \mathbf{u}_n|$ (error at final time $t = 1$)
- Hypothesis: Error $E = \mathcal{O}(h^p)$
- We can find p two ways
 - Do a log-log plot of error vs. h
 - Calculate

$$p \approx \frac{\log(e_k/e_{k-1})}{\log(h_k/h_{k-1})}$$

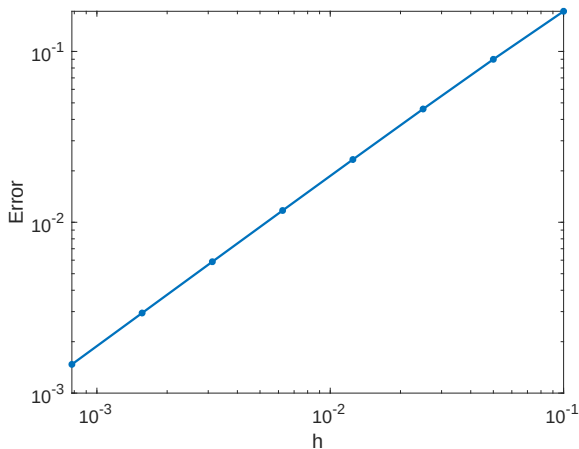
- Run `forward_euler_script.m`



Numerical Experiment

Results

- Result:



$\Rightarrow$ p appears to be 1?



Numerical Experiment

Results

- Calculate p for successive values of h

$$p \approx \frac{\log(e_k/e_{k-1})}{\log(h_k/h_{k-1})}$$

Experiment k	e_k	e_{k-1}	h_k	h_{k-1}	p
0	0.1718	-	1/10	-	-
1	0.08991	0.1718	1/20	1/10	0.93
2	0.04603	0.08991	1/40	1/20	0.96
3	0.02329	0.04603	1/80	1/40	0.98
4	.01172	0.02329	1/160	1/80	0.99

- $p \approx 1$

Question

If the truncation error is $\mathcal{O}(h^2)$, then why is the numerical accuracy $\mathcal{O}(h)$?

Global Truncation Error

- While it is true that the **local error**, $|u(t+h) - u_{i+1}|$ is

$$E_L = |u(t+h) - u_{i+1}| = O(h^2)$$

- The **global error** is the error at final time:

$$E_G = |u(t_n) - u_n|$$

- The errors accumulates from using Euler's method n times

$$O(nh^2) = O((b-a)h) = O(h)$$

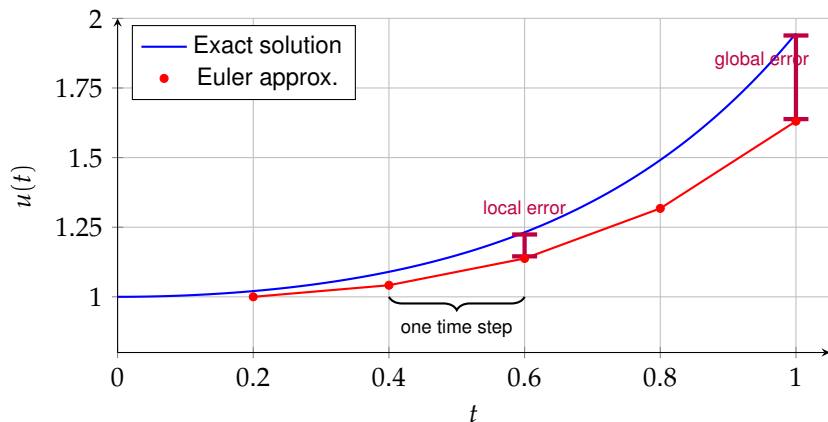
global accuracy since $nh = (b-a)$.

Global Truncation Error

The Euler's method approximation for solving an IVP has a global truncation error

$$E(h) = \mathcal{O}(h)$$

Visualizing Local vs Global Error



Key Idea

- The **local error** accounts for one time step
- The **global error** accounts for all accumulated error

Summary



Differential Equations

- Differential equations describe how quantities change
- Real-world problems $\rightarrow$ models $\rightarrow$ differential equations
- Most differential equations cannot be solved exactly

Key Idea

We use numerical methods to approximate solutions.



What Does a Computer Do?

- We cannot compute the full function $u(t)$
- Instead, we compute values at discrete times:

$$t_0, t_1, t_2, \dots, t_n$$

- This gives a vector:

$$\mathbf{u}_i \approx u(t_i)$$

Key Idea

Continuous problem $\rightarrow$ discrete approximation



Euler's Method

- Based on Taylor expansion:

$$u(t+h) \approx u(t) + hf(t, u)$$

- Update rule:

$$\mathbf{u}_{i+1} = \mathbf{u}_i + hf(t_i, \mathbf{u}_i)$$

Key Idea

Euler's method follows the slope (tangent line) step-by-step.



Error in Euler's Method

- Local truncation error: (Error from one time step)

$$\mathcal{O}(h^2)$$

- Global error: (Accumulated error)

$$\mathcal{O}(h)$$

Key Idea

Errors accumulate over many steps.

