

Lecture 15 - B

§7.1, 7.2, 7.3 Systems of Differential Equations

T. Takahashi

Department of Mathematics and Statistics
University of New Mexico



Goals

We will answer these questions:

- How can you use Euler's method to solve a higher order IVP?
- What is the main idea behind the Runge-Kutta methods?
- How do we choose the step size for Runge-Kutta?



System of ODEs



Population Model

Rabbits

Let $R(t)$ = population of rabbits at time t . Then we can let model their population as

$$R' = cR$$

where $c > 0$ is some growth rate constant.

- If R is small, then population increase rate R' is small.
- If R is big, then population increase rate R' is big.
- Idea: more rabbits $\implies$ more breeding



Population Model

Rabbits

Let $R(t)$ = population of rabbits at time t . Then we can let model their population as

$$R' = cR$$

where $c > 0$ is some growth rate constant.

- If R is small, then population increase rate R' is small.
- If R is big, then population increase rate R' is big.
- Idea: more rabbits $\implies$ more breeding

Foxes

Let $F(t)$ = population of foxes at time t and model it as

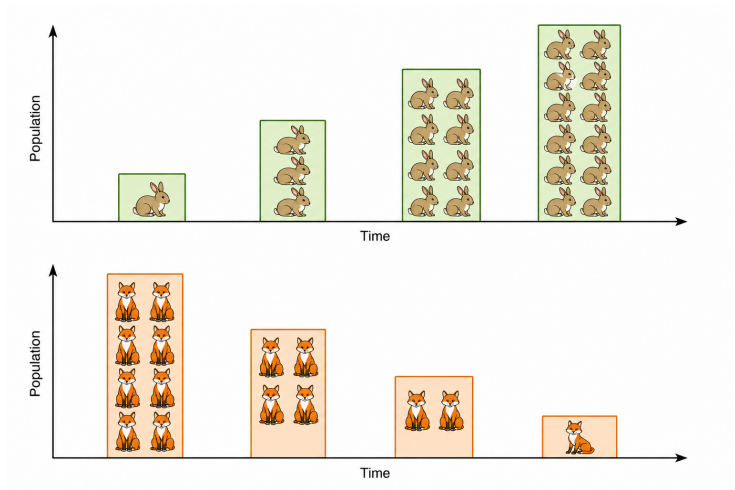
$$F' = -kF$$

where $k > 0$.

- Food is very scarce and they are dying out
- Idea: more foxes $\implies$ less portion of food per fox



Rabbits and Foxes



- Rabbits have infinite amount of grass to graze on
- Foxes have little/no food available

Predator-Prey Model

Question

What if both Foxes and Rabbits lived in the same niche?

- Then Rabbits = Food for Foxes

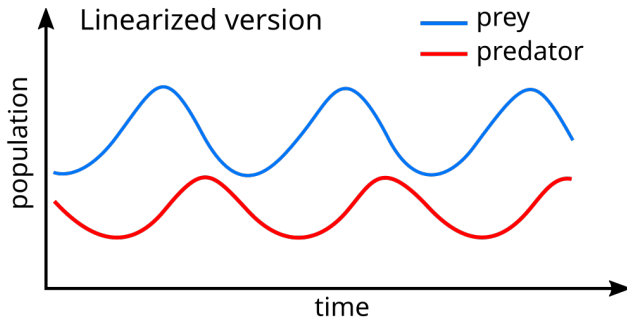
Predator-Prey Model

$$\begin{cases} R' = cR - RF, & \text{Prey growth rate} \\ F' = RF - kF, & \text{Predator growth rate} \end{cases}$$

- ▶ Population of rabbits and foxes influence each other
- ▶ The term RF represents how often they encounter each other
- ▶ Idea: Foxes prey on Rabbits
- ▶ This is a **system of ODEs**



Predator and Prey



Goal

Learn about **systems of ODEs** and how to numerically solve them using Euler's method.



Coupled System

- Suppose you had a **coupled system**

$$\begin{cases} u_1'(t) = f_1(t, u_1, u_2), & u_1(0) = a \\ u_2'(t) = f_2(t, u_1, u_2), & u_2(0) = b \end{cases}$$

- u_1 and u_2 depend on each other so they be solved simultaneously
- Discretize time $t_i = a + i \frac{b-a}{n}$
- We seek discrete solutions of $u_1(t)$ and $u_2(t)$

$$\mathbf{u}_1 = \begin{bmatrix} \mathbf{u}_{1,0} \\ \mathbf{u}_{1,1} \\ \vdots \\ \mathbf{u}_{1,n} \end{bmatrix} \approx \begin{bmatrix} u_1(t_0) \\ u_1(t_0 + h) \\ \vdots \\ u_1(t_0 + nh) \end{bmatrix}, \quad \mathbf{u}_2 = \begin{bmatrix} \mathbf{u}_{2,0} \\ \mathbf{u}_{2,1} \\ \vdots \\ \mathbf{u}_{2,n} \end{bmatrix} \approx \begin{bmatrix} u_2(t_0) \\ u_2(t_0 + h) \\ \vdots \\ u_2(t_0 + nh) \end{bmatrix}.$$



Use Taylor Expansion

- Taylor's Expand both solutions

$$\begin{aligned}u_1(t_i + h) &\approx u_1(t_i) + hu'_1(t_i) \\ &= u_1(t_i) + hf_1(t_i, u_1(t_i), u_2(t_i))\end{aligned}$$

$$\begin{aligned}u_2(t_i + h) &\approx u_2(t_i) + hu'_2(t_i) \\ &= u_2(t_i) + hf_2(t_i, u_1(t_i), u_2(t_i))\end{aligned}$$

- Establish the time stepping rule for both function

$$\mathbf{u}_{1,i+1} = \mathbf{u}_{1,i} + hf_1(t_i, \mathbf{u}_{1,i}, \mathbf{u}_{2,i})$$

- This is **Euler's Method for a coupled system**



Euler's Method for Coupled Systems

- System to solve:

$$\begin{cases} u_1' = f_1(t, u_1, u_2), & u_1(0) = a \\ u_2' = f_2(t, u_1, u_2), & u_2(0) = b \end{cases}$$

- Euler's method for systems

$$\begin{cases} \mathbf{u}_{1,i+1} = \mathbf{u}_{1,i} + h \cdot f_1(t_i, \mathbf{u}_{1,i}, \mathbf{u}_{2,i}), \\ \mathbf{u}_{2,i+1} = \mathbf{u}_{2,i} + h \cdot f_2(t_i, \mathbf{u}_{1,i}, \mathbf{u}_{2,i}). \end{cases}$$

Key Idea

At the core, we are still using the same simple time-stepping idea. The only difference is that we have two variables changing over time.



Euler Example for a System

Part 1

Example

Apply Euler's method with $h = 0.5$ to the IVP

$$\begin{cases} u_1' = u_2, \\ u_2' = -u_1, \\ u_1(0) = 1, \quad u_2(0) = 0 \end{cases}$$

over the time interval $t \in [0, 1]$.

- Step 1: step from $t = 0$ to $t = 0.5$

$$u_1(0.5) \approx u_1(0.0) + h \cdot f_1(u_1(0.0), u_2(0.0)) = 1 + 0.5 * (0) = 1,$$

$$u_2(0.5) \approx u_2(0.0) + h \cdot f_2(u_1(0.0), u_2(0.0)) = 0 + 0.5 * (-1) = -0.5$$



Euler Example for a System

Part 2

- Step 1:

$$\begin{aligned}u_1(0.5) &\approx 1, \\ u_2(0.5) &\approx -0.5\end{aligned}$$

- Step 2: formula step from $t = 0.5$ to $t = 1.0$

$$\begin{aligned}u_1(1.0) &\approx u_1(0.5) + h \cdot f_1(u_1(0.5), u_2(0.5)) \\ u_2(1.0) &\approx u_2(0.5) + h \cdot f_2(u_1(0.5), u_2(0.5))\end{aligned}$$

- Substitute and evaluate

$$\begin{aligned}u_1(1.0) &\approx 1 + 0.5 \cdot (-0.5) = 0.75 \\ u_2(1.0) &\approx -0.5 + 0.5 \cdot (-1) = -1.0\end{aligned}$$



Rabbits and Foxes

Model

Let $x(t)$ = population of rabbits and $y(t)$ = population of foxes (in hundreds at t months).

$$\begin{cases} x' = 4x - 2xy \\ y' = -3y + xy \end{cases}$$

There are initially 300 rabbits and 400 foxes.

- This is a system of the form

$$\begin{cases} x' = f_1(x, y) \\ y' = f_2(x, y) \end{cases}$$

so we can use *Euler's method* to solve this equation



Applying Euler's Method

- Discretize time $\{t_0, t_1, \dots, t_n\}$
- Create two discrete vectors for rabbit and fox populations

$$\mathbf{x} = \begin{bmatrix} 3 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 4 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- Then iterate for i from 0 to $n - 1$

$$x_{i+1} = x_i + h(4x_i - 2x_iy_i)$$

$$y_{i+1} = y_i + h(-3y_i + x_iy_i)$$

- There are two Ways to visualize the solution



Two Ways to Visualize the Solution

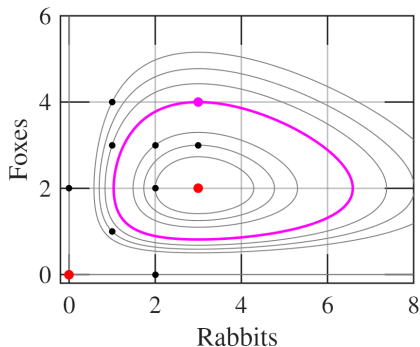


Figure: `plot(x,y)`

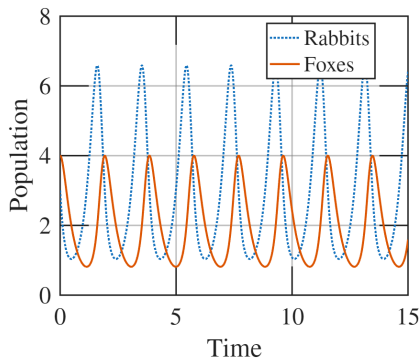


Figure: `plot(t,x,t,y)`

Interpretation

There's an endless self-balancing cycle between rabbits and foxes

General Euler's Method



General System

- In general, a first-order system of m ODEs has the form

$$\begin{cases} u'_1 = f_1(t, u_1, u_2, \dots, u_m) \\ u'_2 = f_2(t, u_1, u_2, \dots, u_m) \\ \vdots \\ u'_m = f_m(t, u_1, u_2, \dots, u_m) \end{cases} \quad t \in (a, b]$$

- Each variable has its own initial condition:

$$\begin{cases} u_1(a) = u_{1a} \\ u_2(a) = u_{2a} \\ \vdots \\ u_m(a) = u_{ma} \end{cases}$$



Compact Notation

- It may be cumbersome to write

$$\begin{cases} u'_1 = f_1(t, u_1, u_2, \dots, u_m) \\ \vdots \\ u'_m = f_m(t, u_1, u_2, \dots, u_m) \end{cases} \quad t \in (a, b]$$

- So we introduce a vector function $\mathbf{u}(t)$ to be the mapping $\mathbf{u} : \mathbb{R} \rightarrow \mathbb{R}^m$

$$\mathbf{u}(t) = [u_1(t) \quad \cdots \quad u_m(t)]^T$$

- Let

$$\mathbf{F}(t, \mathbf{u}) = [f_1(t, \mathbf{u}) \quad \cdots \quad f_m(t, \mathbf{u})]^T$$

Compact Notation for a System

We may define $\mathbf{F}(t, \mathbf{u})$ to be the compact form for f_i for $i = 1$ to m so that we may write the ODE as

$$\underbrace{\mathbf{u}'(t)}_{\text{vector of } u\text{'s}} = \underbrace{\mathbf{F}(t, \mathbf{u})}_{\text{vector of } f\text{'s}}$$

Euler's Method for Systems

Euler's Method

To solve the m dimensional first-order IVP

$$\mathbf{u}'(t) = \mathbf{F}(t, \mathbf{u}), \quad \mathbf{u}(t_0) = \mathbf{u}_0,$$

apply Euler's method component-wise:

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

where $t_i = t_0 + ih$ and h is the step size.

- The compact Euler's method is shorthand for

$$\begin{cases} u_{1,i+1} = u_{1,i} + h \cdot f_1(t_i, u_{1,i}, \dots, u_{m,i}), \\ u_{2,i+1} = u_{2,i} + h \cdot f_2(t_i, u_{1,i}, \dots, u_{m,i}), \\ \vdots \\ u_{m,i+1} = u_{m,i} + h \cdot f_m(t_i, u_{1,i}, \dots, u_{m,i}). \end{cases}$$



Time-Stepping for Any Number of Variables

Euler with One Unknown

$$\text{new} = \text{old} + h * f(\text{old})$$

Euler with Two Unknowns

$$\text{new1} = \text{old1} + h * f_1(\text{old1}, \text{old2})$$

$$\text{new2} = \text{old2} + h * f_2(\text{old1}, \text{old2})$$

Euler with m Unknowns

$$\text{new} = \text{old} + h * \mathbf{F}(\text{old})$$

new and **old** are vectors of length m

Higher Order ODEs



The Genius of Euler

Question

Why does the great Euler get credit for such a simple algorithm?

$$u_{i+1} = u_i + hf(u_i)$$



The Genius of Euler

Question

Why does the great Euler get credit for such a simple algorithm?

$$u_{i+1} = u_i + hf(u_i)$$



His contribution comes from solving *higher order* ODEs.



Higher-Order ODEs

- The **order** refers to the highest derivative.

$$\underbrace{u'(t)}_{\text{1st order}} + 2u = 6$$
$$\underbrace{2u''(t)}_{\text{2nd order}} - u'(t) + 7u(t) = 0.$$

- Euler's method require the equation to be first-order ODEs.
- We may introduce new variables to change the ODE

Goal

Rewrite the second-order ODE as a first-order ODE.



Introducing New Variables

- Consider the 2nd-order ODE:

$$u''(t) + au'(t) + btu(t) = c.$$

- Introduce *new* variables y_1 and y_2 :

$$\begin{cases} y_1 = u(t), & \text{(the original function),} \\ y_2 = u'(t), & \text{(the first derivative or the original).} \end{cases}$$

- We may take the derivative of $y_1(t)$ and $y_2(t)$

$$\begin{cases} y_1' = u'(t) \\ y_2' = u''(t). \end{cases}$$



Rewriting as a First-Order System

- Using the definitions $y_1 = u$ and $y_2 = u'$, compute derivatives:

$$\begin{cases} y_1' = u' = y_2, \\ y_2' = u''. \end{cases}$$

- Substitute u'' from the original ODE:

$$u'' = -au' - btu + c \equiv y_2'.$$



Rewriting as a First-Order System

- Using the definitions $y_1 = u$ and $y_2 = u'$, compute derivatives:

$$\begin{cases} y_1' = u' = y_2, \\ y_2' = u''. \end{cases}$$

- Substitute u'' from the original ODE:

$$u'' = -au' - btu + c \equiv y_2'.$$

Result

Rewrite in terms of y_1 and y_2 :

$$\begin{cases} y_1' = y_2, \\ y_2' = -ay_2 - by_1 + c \end{cases} \Leftrightarrow \begin{cases} y_1' = f_1(t, y_1, y_2), \\ y_2' = f_2(t, y_1, y_2) \end{cases}$$

Key Idea

2nd Order DEs are the same as a 1st order system.

Euler's Idea

Original ODE

$$\begin{cases} u'' - 2u' + 8u = 0, & t \in (0, T] \\ u(0) = A, u'(0) = B \end{cases}$$

New Equivalent System

$$\begin{cases} y_1' = y_2, \\ y_2' = -8y_1 + 2y_2 \\ y_1(0) = A, y_2(0) = B \end{cases}$$

Euler's Idea:

- We can't solve the original ODE
- Rewrite the ODE as a first-order system
- Now we *can* solve it
- The solution to the original problem is $y_1(t) = u(t)$



Reduction of Order

- This technique works for a general n^{th} order ODE

$$a_n u^{(n)} + a_{n-1} u^{(n-1)} + \cdots + a_1 u = f$$

- After doing a change of variables $y_i = u^{(i-1)}$ we get an ODE system

$$\begin{cases} y_1' &= f_1(y_1, \dots, y_n) \\ &\vdots \\ y_n' &= f_n(y_1, \dots, y_n) \end{cases}$$

- After solving with Euler's method, we get $u(t) = y_1(t) \approx \mathbf{y}_1$

Method: Reduction of Order

- **Given:** n^{th} order ODE, $n > 1$
- **Output:** Equivalent system of 1st order ODEs
- **Idea:** By introducing a new set of variables, the equation becomes solvable via Euler's method

Example

Example

Rewrite the IVP as a system of first order ODEs including the initial conditions

$$\begin{cases} u''' = (u'')^2 - uu' + \sin t, \\ u(0) = 1, u'(0) = 0, u''(0) = 2 \end{cases}$$

- Introduce new variables $y_1 = u, y_2 = u', y_3 = u''$
- Take derivatives of y_1, y_2 , and y_3
- Write out the ODE system only in terms of y_1, y_2 , and y_3

$$\begin{cases} y_1' = y_2, & y_1(0) = 1 \\ y_2' = y_3, & y_2(0) = 0 \\ y_3' = y_3^2 - y_1 y_2 + \sin t, & y_3(0) = 2 \end{cases}$$



Alternate Derivation to Euler's Method



Generalizing Euler's Method

Goal

Generalize Euler's Method as a part of a larger class of algorithms

Why?

Convergence rate for Euler's method is only $O(h)$ which is quite slow

How?

- We will reframe Euler's method as doing a crude quadrature
- We will replace the crude quadrature with more advanced approximations
- This is how **Runge-Kutta methods** are derived



Alternate Derivation of Euler's Method

Part 1

- Consider the ODE:

$$\frac{du}{dt} = f(t, u(t)).$$

- Integrating both sides over one step gives:

$$\int_{t_i}^{t_{i+1}} \frac{du}{dt} dt = \int_{t_i}^{t_{i+1}} f(t, u(t)) dt.$$

- Left side simplifies by the Fundamental Theorem of Calculus:

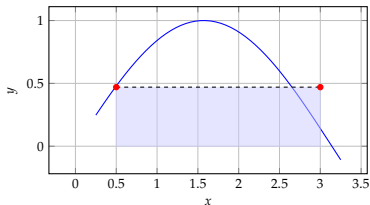
$$u(t_{i+1}) - u(t_i) = \int_{t_i}^{t_{i+1}} f(t, u(t)) dt.$$

- Or rather

$$u(t_{i+1}) = u(t_i) + \int_{t_i}^{t_{i+1}} f(t, u(t)) dt.$$



What if we did something very simple?



- The left-end Riemann sum works like so

$$\int_a^b g(t) dt \approx (b-a)g(a).$$

- What if we applied this simple idea to

$$u(t_{i+1}) = u(t_i) + \int_{t_i}^{t_{i+1}} f(t, u(t)) dt?$$



Alternate Derivation of Euler's Method

Part 2

- The left-point Riemann sum says

$$\begin{aligned}u(t_{i+1}) &= u(t_i) + \int_{t_i}^{t_{i+1}} f(t, u(t)) dt \\&\approx u(t_i) + (t_{i+1} - t_i) f(t_i, u_i) \\&= u(t_i) + h f(t_i, u_i)\end{aligned}$$

where $h = t_{i+1} - t_i$.

- We obtain Euler's method:

$$u_{i+1} = u_i + h f(t_i, u_i).$$

Key Idea

Euler's method was obtained by doing a very simple quadrature.

Question

What if used even better quadratures?

Runge-Kutta Time Stepping

- **Runge-Kutta** methods are derived by approximating the integral in

$$u(t_{i+1}) = u(t_i) + \underbrace{\int_{t_i}^{t_{i+1}} f(t, u(t)) dt}_{\text{approximate}}$$

and developing a time-stepping algorithm

- Euler's method is a Runge-Kutta method using the left-endpoint Riemann sum approximation

Method: Runge-Kutta

- **Given:** Initial Value Problem, time discretization h
- **Output:** Discrete solution $\mathbf{u}$
- **Idea:** Approximate $\int_{t_i}^{t_{i+1}} f(t, u(t)) dt$ with a quadrature



Summary



- Euler can be applied to solve a simple first order ODE

$$u_{i+1} = u_i + hf(u_i)$$

- Euler can also be applied to a coupled ODE

$$x_{i+1} = x_i + hf(x_i, y_i)$$

$$y_{i+1} = y_i + hf(x_i, y_i)$$

- It can generally be applied to an entire system of ODEs

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



Reduction of Order

- The true power of Euler's method is that you can reduce a 2nd order equation

$$x'' + 4x' + 2x = 1$$

- Into a system of first order ODEs

$$u_1' = u_2$$

$$u_2' = -2u_1 - 4u_2 + 1$$

- Then you can solve it using Euler's

$$u_{1,i+1} = u_{1,i} + h u_{2,i}$$

$$u_{2,i+1} = u_{2,i} + h(-2u_{1,i} - 4u_{2,i} + 1)$$



Euler's Method Derivation

- One way to derive Euler's method is with a Taylor expansion

$$u(t_i + h) \approx u(t_i) + hu'(t_i)$$

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

- Another way is to use a very simple quadrature

$$u(t_i + h) = u(t_i) + \int_{t_i}^{t_{i+1}} f(t, u(t)) dt$$

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



- Euler's method is a byproduct of a very crude quadrature

$$u(t_i + h) = u(t_i) + \underbrace{\int_{t_i}^{t_{i+1}} f(t, u(t)) dt}_{\text{Riemann approximation}}$$
$$u(t_i + h) \approx u(t_i) + hf(t_i, u(t_i))$$

Question

What happens when we do a more sophisticated quadrature?



- What time stepping methods are better than Euler's method?
- What potential problems arise when the time step h is not small enough?
- What does a modern ODE solver do?
- How do you make numerical simulations with ODE solvers?

