Numerical Methods for Ordinary Differential Equations

Dangxing Chen

Duke Kunshan University

Initial value problem

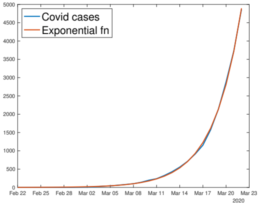
- **Def**: **Initial value problem** (IVP):

$$\begin{cases} y'(t) &= f(t,y), \\ y(0) &= y_0. \end{cases}$$

- **Def**: **Picard integral equation**: $y(t) = y_0 + \int_0^t f(\tau, y)\ d\tau$
- E.g. Exponential grow/decay $y(t) = ce^{\lambda t}$

$$\begin{cases} y'(t) &= \lambda y(t), \\ y(0) &= c. \end{cases}$$

- Real life example: Covid cases



Three-body problem

- A famous scientific fiction by Cixin Liu

- Motions of Moon, Earth, and the Sun
- Generalization: N-body simulation

Three-body problem mathematical description

- Approximate bodies by point particles
- G: gravitational constant
- Mass: m_i
- Position vector of ith particle (x_i, y_i, z_i)
- Distance between each particle
  $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$
- Newtonian equations of motion,

$$m_1 \frac{d^2 x_1}{dt^2} = -\frac{G m_1 m_2 (x_1 - x_2)}{d_{1,2}^3} - \frac{G m_1 m_3 (x_1 - x_3)}{d_{1,3}^3},$$

$$m_1 \frac{d^2 y_1}{dt^2} = -\frac{G m_1 m_2 (y_1 - y_2)}{d_{1,2}^3} - \frac{G m_1 m_3 (y_1 - y_3)}{d_{1,3}^3},$$

$$m_1 \frac{d^2 z_1}{dt^2} = -\frac{G m_1 m_2 (z_1 - z_2)}{d_{1,2}^3} - \frac{G m_1 m_3 (z_1 - z_3)}{d_{1,3}^3},$$

$$\vdots$$

Reformulation

- Velocity vector of ith particle (u_i, v_i, w_i)
- Hamiltonian formalism

$$
\begin{aligned}
\frac{dx_1}{dt} &= u_1, \\
\frac{dy_1}{dt} &= v_1, \\
\frac{dz_1}{dt} &= w_1, \\
\frac{du_1}{dt} &= -\frac{Gm_2(x_1 - x_2)}{d_{1,2}^3} - \frac{Gm_3(x_1 - x_3)}{d_{1,3}^3}, \\
\frac{dv_1}{dt} &= -\frac{Gm_2(y_1 - y_2)}{d_{1,2}^3} - \frac{Gm_3(y_1 - y_3)}{d_{1,3}^3}, \\
\frac{dw_1}{dt} &= -\frac{Gm_2(z_1 - z_2)}{d_{1,2}^3} - \frac{Gm_3(z_1 - z_3)}{d_{1,3}^3}, \\
&\vdots
\end{aligned}
$$

Euler's method

- IVP for $t \in [0, T]$:

$$\begin{cases} y'(t) &= f(t, y), \\ y(0) &= y_0. \end{cases}$$

- Use n equispace grids, $\Delta t = \frac{T}{n}$
- $t_j = \frac{jT}{n}$
- Linear approximation: $y(t_{j+1}) \approx y(t_j) + y'(t_j)\Delta t$
- **Explicit Euler's method**: $y(t_{j+1}) = y(t_j) + f(t_j, y(t_j))\Delta t$
- Analysis:
  - Accuracy
  - Stability
- **Question**: What is the local truncation error of explicit Euler's method?

Euler's method

- IVP for $t \in [0, T]$:

$$\begin{cases} y'(t) & = f(t, y), \\ y(0) & = y_0. \end{cases}$$

- Use n equispace grids, $\Delta t = \frac{T}{n}$
- $t_j = \frac{jT}{n}$
- Linear approximation: $y(t_{j+1}) \approx y(t_j) + y'(t_j)\Delta t$
- **Explicit Euler's method**: $y(t_{j+1}) = y(t_j) + f(t_j, y(t_j))\Delta t$
- Analysis:
  - Accuracy
  - Stability
- **Question**: What is the local truncation error of explicit Euler's method?
- $y(\Delta t) - y_0 - y_0'\Delta t = \mathcal{O}(\Delta t^2)$

Global error

- **Question**: How about global error?
- $|f(t, y_1) - f(t, y_2)| \leq K|y_1 - y_2|$
- $\max_x |y''(x)| \leq L$

Global error

- **Question**: How about global error?
- $|f(t, y_1) - f(t, y_2)| \leq K|y_1 - y_2|$
- $\max_x |y''(x)| \leq L$
- Comparison

$$y_{n+1} = y_n + \Delta t f(t_n, y_n) + \frac{y''(\xi_n)}{2}\Delta t^2,$$

$$\widetilde{y}_{n+1} = \widetilde{y}_n + \Delta t f(t_n, \widetilde{y}_n).$$

- $e_{n+1} = e_n + \Delta t[f(t_n, y_n) - f(t_n, \widetilde{y}_n)] + \frac{y''(\xi_n)}{2}\Delta t^2$
- $|e_{n+1}| \leq |e_n| + \Delta t K|e_n| + \frac{L}{2}\Delta t^2$
- $|e_{n+1}| \leq (1 + \Delta t K)|e_n| + \frac{L}{2}\Delta t^2$
- $|e_{n+1}| \leq \left[1 + (1 + \Delta t K) + \cdots + (1 + \Delta t K)^{n-1}\right] \frac{L}{2}\Delta t^2$
- $|e_{n+1}| \leq \frac{(1+\Delta t K)^n - 1}{2K} L\Delta t \leq \frac{e^{TK} - 1}{2K} L\Delta t$
- **Def**: $|e(\Delta t)| \approx c\Delta t^p$, then it is pth **order** method
- Euler's method is the first order method

Stability

- For stability, we always consider $y(t) = e^{-\lambda t}$, $\lambda > 0$ with

$$y'(t) = -\lambda y(t),$$
$$y(0) = 1.$$

- Despite accuracy, the solution should decay
- Explicit Euler's method: $y_{n+1} = y_n - \Delta t \lambda y_n = (1 - \Delta t \lambda) y_n$
- Need $|1 - \Delta t \lambda| < 1 \Rightarrow \Delta t < \frac{2}{\lambda}$
- **Question**: How to improve the stability?

Stability

- For stability, we always consider $y(t) = e^{-\lambda t}$, $\lambda > 0$ with
$$y'(t) = -\lambda y(t),$$
$$y(0) = 1.$$
- Despite accuracy, the solution should decay
- Explicit Euler's method: $y_{n+1} = y_n - \Delta t \lambda y_n = (1 - \Delta t \lambda) y_n$
- Need $|1 - \Delta t \lambda| < 1 \Rightarrow \Delta t < \frac{2}{\lambda}$
- **Question**: How to improve the stability?
- $y(t) = y_0 + \int_0^t f(\tau, y)\, d\tau$
- **Implicit Euler's method**: $y_{n+1} = y_n + \Delta t f(t_{n+1}, y_{n+1})$
- For this example, $y_{n+1} = y_n - \Delta t \lambda y_{n+1} \Rightarrow y_{n+1} = \frac{y_n}{1+\Delta t \lambda}$
- **Remark**: Implicit method improves the stability
- **Question**: Drawback of implicit methods?

Stability

- For stability, we always consider $y(t) = e^{-\lambda t}$, $\lambda > 0$ with

$$\begin{aligned} y'(t) &= -\lambda y(t), \\ y(0) &= 1. \end{aligned}$$

- Despite accuracy, the solution should decay
- Explicit Euler's method: $y_{n+1} = y_n - \Delta t \lambda y_n = (1 - \Delta t \lambda) y_n$
- Need $|1 - \Delta t \lambda| < 1 \Rightarrow \Delta t < \frac{2}{\lambda}$
- **Question**: How to improve the stability?
- $y(t) = y_0 + \int_0^t f(\tau, y)\, d\tau$
- **Implicit Euler's method**: $y_{n+1} = y_n + \Delta t f(t_{n+1}, y_{n+1})$
- For this example, $y_{n+1} = y_n - \Delta t \lambda y_{n+1} \Rightarrow y_{n+1} = \frac{y_n}{1 + \Delta t \lambda}$
- **Remark**: Implicit method improves the stability
- **Question**: Drawback of implicit methods?
- Computation expensive

Trapezoidal method

- **Question**: How to use the Trapezoidal's rule to improve?

Trapezoidal method

- **Question**: How to use the Trapezoidal's rule to improve?
- $y_{n+1} = y_n + \frac{\Delta t}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1})]$
- **Question**: What is the local truncation error?

Trapezoidal method

- **Question**: How to use the Trapezoidal's rule to improve?
- $y_{n+1} = y_n + \frac{\Delta t}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1})]$
- **Question**: What is the local truncation error?
- Equivalent to analyze
  $y(\Delta t) = y_0 + \frac{\Delta t}{2}[f(0, y(0)) + f(\Delta t, y(\Delta t))]$
- $y(t) = 1$: $1 = 1 + 0$
- $y(t) = t$: $\Delta t = 0 + \frac{\Delta t}{2}[1 + 1]$
- $y(t) = t^2$: $\Delta t^2 = 0 + \frac{\Delta t}{2}[0 + 2\Delta t]$
- $y(t) = t^3$: $\Delta t^3 \neq 0 + \frac{\Delta t}{2}[0 + 3\Delta t^2]$
- $|e(\Delta t)| = \mathcal{O}(\Delta t^3)$
- **Question**: What is the global error?

Trapezoidal method

- **Question**: How to use the Trapezoidal's rule to improve?
- $y_{n+1} = y_n + \frac{\Delta t}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1})]$
- **Question**: What is the local truncation error?
- Equivalent to analyze
  $y(\Delta t) = y_0 + \frac{\Delta t}{2}[f(0, y(0)) + f(\Delta t, y(\Delta t))]$
- $y(t) = 1$: $1 = 1 + 0$
- $y(t) = t$: $\Delta t = 0 + \frac{\Delta t}{2}[1 + 1]$
- $y(t) = t^2$: $\Delta t^2 = 0 + \frac{\Delta t}{2}[0 + 2\Delta t]$
- $y(t) = t^3$: $\Delta t^3 \neq 0 + \frac{\Delta t}{2}[0 + 3\Delta t^2]$
- $|e(\Delta t)| = \mathcal{O}(\Delta t^3)$
- **Question**: What is the global error?
- $\mathcal{O}(\Delta t^2)$
- second order method

Stability analysis

- $y'(t) = -\lambda y(t)$
- $y_{\Delta t} = y_0 - \frac{\lambda \Delta t}{2}(y_0 + y_{\Delta t})$
- $\left(1 + \frac{\lambda \Delta t}{2}\right) y_{\Delta t} = \left(1 - \frac{\lambda \Delta t}{2}\right) y_0$
- $y_{\Delta t} = \frac{2 - \lambda \Delta t}{2 + \lambda \Delta t} y_0$
- Stable

Pseudo-spectral method

- $y(t) = y_0 + \int_0^t f(\tau, y)\ d\tau$
- **Question**: How to obtain arbitrage high-order?

Pseudo-spectral method

- $y(t) = y_0 + \int_0^t f(\tau, y)\ d\tau$
- **Question**: How to obtain arbitrage high-order?
- Suppose $0 \le t_1 < \cdots < t_n \le \Delta t$
- $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \ \mathbf{y}_0 = \begin{bmatrix} y_0 \\ \vdots \\ y_0 \end{bmatrix}, \ \mathbf{f}(\mathbf{y}) = \begin{bmatrix} f(t_1, y_1) \\ \vdots \\ f(t_n, y_n) \end{bmatrix}$
- Solves $\mathbf{y} = \mathbf{y}_0 + \Delta t \mathbf{S} \mathbf{f}(\mathbf{y})$
- Calculate $y_{\Delta t} = y_0 + \Delta t \mathbf{s} \mathbf{f}(\mathbf{y})$
- Advantages: have good math properties

Picard iteration

- **Question**: A simplest way to solve $\mathbf{y} = \mathbf{y}_0 + \Delta t \mathbf{S}\mathbf{f}(\mathbf{y})$?

Picard iteration

- **Question**: A simplest way to solve $\mathbf{y} = \mathbf{y}_0 + \Delta t \mathbf{S} \mathbf{f}(\mathbf{y})$?
- $\mathbf{y}^{[i+1]} = \mathbf{y}_0 + \Delta t \mathbf{S} \mathbf{f}(\mathbf{y}^{[i]})$

Picard iteration

- **Question**: A simplest way to solve $\mathbf{y} = \mathbf{y}_0 + \Delta t \mathbf{S}\mathbf{f}(\mathbf{y})$?
- $\mathbf{y}^{[i+1]} = \mathbf{y}_0 + \Delta t \mathbf{S}\mathbf{f}(\mathbf{y}^{[i]})$
- Error: $\mathbf{e}^{[i]} = \mathbf{y} - \mathbf{y}^{[i]}$
- $$\mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{y}^{[i]}) = \begin{bmatrix} f(y_1) \\ \vdots \\ f(y_n) \end{bmatrix} - \begin{bmatrix} f(y_1^{[i]}) \\ \vdots \\ f(y_n^{[i]}) \end{bmatrix} \approx \begin{bmatrix} f'(y_1)(y_1 - y_1^{[i]}) \\ \vdots \\ f'(y_n)(y_n - y_n^{[i]}) \end{bmatrix}$$
- $$\mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{y}^{[i]}) \approx \begin{bmatrix} f'(y_1) & 0 & \dots & 0 \\ 0 & f'(y_2) & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & f'(y_n) \end{bmatrix} \begin{bmatrix} y_1 - y_1^{[i]} \\ y_2 - y_2^{[i]} \\ \vdots \\ y_n - y_n^{[i]} \end{bmatrix}$$
- $\mathbf{e}^{[i+1]} = \Delta t \mathbf{S}(\mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{y}^{[i]})) \approx \Delta t \mathbf{S}\mathbf{J}(\mathbf{y}^{[i]})\mathbf{e}^{[i]}$
- **Question**: Converge?

Picard iteration

- **Question**: A simplest way to solve $\mathbf{y} = \mathbf{y}_0 + \Delta t \mathbf{S}\mathbf{f}(\mathbf{y})$?
- $\mathbf{y}^{[i+1]} = \mathbf{y}_0 + \Delta t \mathbf{S}\mathbf{f}(\mathbf{y}^{[i]})$
- Error: $\mathbf{e}^{[i]} = \mathbf{y} - \mathbf{y}^{[i]}$
- $\mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{y}^{[i]}) = \begin{bmatrix} f(y_1) \\ \vdots \\ f(y_n) \end{bmatrix} - \begin{bmatrix} f(y_1^{[i]}) \\ \vdots \\ f(y_n^{[i]}) \end{bmatrix} \approx \begin{bmatrix} f'(y_1)(y_1 - y_1^{[i]}) \\ \vdots \\ f'(y_n)(y_n - y_n^{[i]}) \end{bmatrix}$
- $\mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{y}^{[i]}) \approx \begin{bmatrix} f'(y_1) & 0 & \dots & 0 \\ 0 & f'(y_2) & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & f'(y_n) \end{bmatrix} \begin{bmatrix} y_1 - y_1^{[i]} \\ y_2 - y_2^{[i]} \\ \vdots \\ y_n - y_n^{[i]} \end{bmatrix}$
- $\mathbf{e}^{[i+1]} = \Delta t \mathbf{S}(\mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{y}^{[i]})) \approx \Delta t \mathbf{S}\mathbf{J}(\mathbf{y}^{[i]})\mathbf{e}^{[i]}$
- **Question**: Converge?
- $\max |\lambda(\Delta t \mathbf{S}\mathbf{J}(\mathbf{y}^{[i]}))| < 1$

Newton's method

- **Question**: Newton's method to solve $\mathbf{y} = \mathbf{y}_0 + \Delta t \mathbf{S}\mathbf{f}(\mathbf{y})$?

Newton's method

- **Question**: Newton's method to solve $\mathbf{y} = \mathbf{y}_0 + \Delta t \mathbf{S}\mathbf{f}(\mathbf{y})$?
- Solve root-finding for $\mathbf{g}(\mathbf{y}) = \mathbf{y} - \mathbf{y}_0 - \Delta t \mathbf{S}\mathbf{f}(\mathbf{y})$
- $\mathbf{J}_{\mathbf{g}}(\mathbf{y}) = \mathbf{I} - \Delta t \mathbf{S}\mathbf{J}_{\mathbf{f}}(\mathbf{y})$
- $\mathbf{y}^{[i+1]} = \mathbf{y}^{[i]} - \mathbf{J}_{\mathbf{g}}^{-1}(\mathbf{y}^{[i]})\mathbf{g}(\mathbf{y}^{[i]})$
- $\mathbf{y}^{[i+1]} = \mathbf{y}^{[i]} - (\mathbf{I} - \Delta t \mathbf{S}\mathbf{J}_{\mathbf{f}}(\mathbf{y}^{[i]}))^{-1}(\mathbf{y}^{[i]} - \mathbf{y}_0 - \Delta t \mathbf{S}\mathbf{f}(\mathbf{y}^{[i]}))$

Comparison for a linear ODE

- Scalar linear ODE:

$$\begin{cases} y'(t) &= \lambda y(t), \\ y(0) &= 0. \end{cases}$$

- Picard integral: $y(t) = y_0 + \lambda \int_0^t y(\tau)\ d\tau$
- Discretization: $\mathbf{y} = \mathbf{y}_0 + \Delta t \lambda \mathbf{S}\mathbf{y}$
- Solution: $(\mathbf{I} - \Delta t \lambda \mathbf{S})\mathbf{y} = \mathbf{y}_0$
- Newton: Directly solve $\mathbf{y} = (\mathbf{I} - \Delta t \lambda \mathbf{S})^{-1}\mathbf{y}_0$
- Picard iteration: Solve it by Neumann series $\mathbf{y}^{[i+1]} = \mathbf{y}_0 + \Delta t \lambda \mathbf{S}\mathbf{y}^{[i]}$

Motivation

- Picard integral equation: $y_{\Delta t} = y_0 + \int_0^{\Delta t} f(\tau, y)\ d\tau$
- For high-order, $\int_0^{\Delta t} f(\tau, y) d\tau \approx \Delta t \sum_j w_j f(t_j, y_j)$
- **Question**: How to simplify the calculation?

Motivation

- Picard integral equation: $y_{\Delta t} = y_0 + \int_0^{\Delta t} f(\tau, y)\ d\tau$
- For high-order, $\int_0^{\Delta t} f(\tau, y) d\tau \approx \Delta t \sum_j w_j f(t_j, y_j)$
- **Question**: How to simplify the calculation?
- Predict y_j using previous information
- E.g. Trapezoidal method:
  $y_{\Delta t} = y_0 + \frac{\Delta t}{2}[f(0, y_0) + f(\Delta t, y_{\Delta t})]$

Motivation

- Picard integral equation: $y_{\Delta t} = y_0 + \int_0^{\Delta t} f(\tau, y)\ d\tau$
- For high-order, $\int_0^{\Delta t} f(\tau, y) d\tau \approx \Delta t \sum_j w_j f(t_j, y_j)$
- **Question**: How to simplify the calculation?
- Predict y_j using previous information
- E.g. Trapezoidal method:
  $y_{\Delta t} = y_0 + \frac{\Delta t}{2}[f(0, y_0) + f(\Delta t, y_{\Delta t})]$
- Let $\widetilde{y}_{\Delta t} \approx y_0 + \Delta t f(0, y_0)$
- **Heun's method**:
  $y_{\Delta t} = y_0 + \frac{\Delta t}{2}[f(0, y_0) + f(\Delta t, y_0 + \Delta t f(0, y_0))]$
- **Question**: Local truncation error?

Convergence analysis

- Taylor expansion of $y_{\Delta t}$

$$
\begin{aligned}
y_{\Delta t} &= y_0 + y_0'\Delta t + \frac{y_0''}{2}\Delta t^2 + \mathcal{O}(\Delta t^3) \\
&= y_0 + f(0, y_0)\Delta t + \frac{f'(0, y_0)}{2}\Delta t^2 + \mathcal{O}(\Delta t^3) \\
&= y_0 + f(0, y_0)\Delta t + \frac{f_t(0, y_0) + f_y(0, y_0) f(0, y_0)}{2}\Delta t^2 + \mathcal{O}(\Delta t^3)
\end{aligned}
$$

- Taylor expansion of Heun's method

$$
\begin{aligned}
y_{\Delta t} &= y_0 + \frac{\Delta t}{2}\left[f(0, y_0) + f(\Delta t, y_0 + \Delta t f(0, y_0))\right] \\
&= y_0 + \frac{\Delta t}{2}\left[2f(0, y_0) + \Delta t f_t(0, y_0) + \Delta t f(0, y_0) f_y(0, y_0) + \mathcal{O}(\Delta t^2)\right] \\
&= y_0 + f(0, y_0)\Delta t + \frac{f_t(0, y_0) + f_y(0, y_0) f(0, y_0)}{2}\Delta t^2 + \mathcal{O}(\Delta t^3)
\end{aligned}
$$

- Second order method

Example

- Constant ODE:

$$\begin{cases} y'(t) & = \lambda y(t), \\ y(0) & = y_0. \end{cases}$$

- Heun's method:
  $y_{\Delta t} = y_0 + \frac{\Delta t}{2}[f(0, y_0) + f(\Delta t, y_0 + \Delta t f(0, y_0))]$

Example

- Constant ODE:

$$\begin{cases} y'(t) & = \lambda y(t), \\ y(0) & = y_0. \end{cases}$$

- Heun's method:
  $y_{\Delta t} = y_0 + \frac{\Delta t}{2}[f(0, y_0) + f(\Delta t, y_0 + \Delta t f(0, y_0))]$
- $y_{\Delta t} = y_0 + \frac{\Delta t}{2}[\lambda y_0 + \lambda(y_0 + \Delta t \lambda y_0)] = y_0 + \lambda \Delta t y_0 + \frac{\lambda^2 \Delta t^2}{2} y_0$

RK4

- The most common RK: RK4
- $y_{n+1} = y_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
- $k_1 = f(t_n, y_n)$
- $k_2 = f\left(t_n + \frac{\Delta t}{2}, y_n + \Delta t \frac{k_1}{2}\right)$
- $k_3 = f\left(t_n + \frac{\Delta t}{2}, y_n + \Delta t \frac{k_2}{2}\right)$
- $k_4 = f(t_n + \Delta t, y_n + \Delta t k_3)$
- Works very good empirically
- **Question**: Do we need to worry about stability?
- Balance between stability and accuracy
- Advantages: Simple yet effective

Background

- Prey and Predators
- E.g. Rabbit and wolves, fish and sharks ...
- $R(t)$ - number of prey
- $W(t)$ - number of predators
- In the absence of predators,

$$\frac{dR}{dt} = kR, \quad k > 0.$$

- In the absence of prey,

$$\frac{dW}{dt} = -rW, \quad r > 0.$$

- Interaction: RW

- **Predator-prey equations** or **Lotka-Volterra equations**

$$\frac{dR}{dt} = kR - aRW,$$
$$\frac{dW}{dt} = -rW + bRW.$$

- Sample solutions:

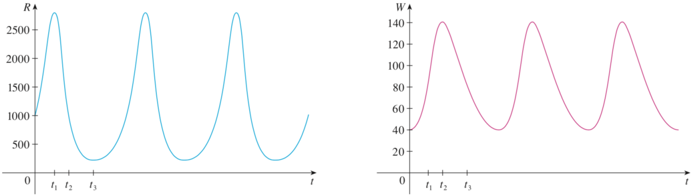
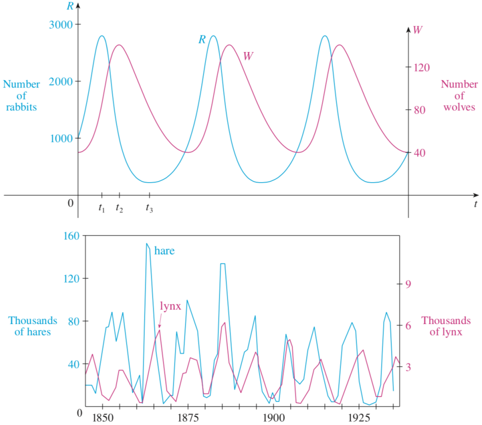


FIGURE 4 Graphs of the rabbit and wolf populations as functions of time

- Hudson's Bay Company: trading in animal furs
- Snowshoe hare and Canada lynx



- **Question**: How to solve this equation?

Explicit methods

- Predator-prey equations:

$$\frac{dR}{dt} = kR - aRW,$$
$$\frac{dW}{dt} = -rW + bRW.$$

- Vector form: $\mathbf{y} = \begin{bmatrix} R \\ W \end{bmatrix}$ and $\mathbf{f}(\mathbf{y}) = \begin{bmatrix} kR - aRW \\ -rW + bRW \end{bmatrix}$

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}).$$

- Explicit Euler's method:

Explicit methods

- Predator-prey equations:

$$\frac{dR}{dt} = kR - aRW,$$
$$\frac{dW}{dt} = -rW + bRW.$$

- Vector form: $\mathbf{y} = \begin{bmatrix} R \\ W \end{bmatrix}$ and $\mathbf{f}(\mathbf{y}) = \begin{bmatrix} kR - aRW \\ -rW + bRW \end{bmatrix}$

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}).$$

- Explicit Euler's method: $\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{f}(\mathbf{y}_n)\Delta t$

$$R_{n+1} = R_n + (kR_n - aR_nW_n)\Delta t,$$
$$W_{n+1} = W_n + (-rW_n + bR_nW_n)\Delta t.$$

Heun's method

- Predator-prey equations

$$\frac{dR}{dt} = kR - aRW,$$
$$\frac{dW}{dt} = -rW + bRW.$$

- Vector form: $\mathbf{y} = \begin{bmatrix} R \\ W \end{bmatrix}$ and $\mathbf{f}(\mathbf{y}) = \begin{bmatrix} kR - aRW \\ -rW + bRW \end{bmatrix}$

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}).$$

- Heun's method:

Heun's method

- Predator-prey equations

$$\frac{dR}{dt} = kR - aRW,$$
$$\frac{dW}{dt} = -rW + bRW.$$

- Vector form: $\mathbf{y} = \begin{bmatrix} R \\ W \end{bmatrix}$ and $\mathbf{f}(\mathbf{y}) = \begin{bmatrix} kR - aRW \\ -rW + bRW \end{bmatrix}$

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}).$$

- Heun's method: $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{2}[\mathbf{f}(\mathbf{y}_n) + \mathbf{f}(\mathbf{y}_n + \Delta t \mathbf{f}(\mathbf{y}_n))]$

RK4

- Predator-prey equations

$$\frac{dR}{dt} = kR - aRW,$$
$$\frac{dW}{dt} = -rW + bRW.$$

- Vector form: $\mathbf{y} = \begin{bmatrix} R \\ W \end{bmatrix}$ and $\mathbf{f}(\mathbf{y}) = \begin{bmatrix} kR - aRW \\ -rW + bRW \end{bmatrix}$

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}).$$

- RK4: $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$
- $\mathbf{k}_1 = \mathbf{f}(\mathbf{y}_n)$
- $\mathbf{k}_2 = \mathbf{f}\left(\mathbf{y}_n + \Delta t \frac{\mathbf{k}_1}{2}\right)$
- $\mathbf{k}_3 = \mathbf{f}\left(\mathbf{y}_n + \Delta t \frac{\mathbf{k}_2}{2}\right)$
- $\mathbf{k}_4 = \mathbf{f}(\mathbf{y}_n + \Delta t \mathbf{k}_3)$

Implicit Euler's method

- Implicit Euler's method:

Implicit Euler's method

- Implicit Euler's method: $\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{f}(\mathbf{y}_{n+1})\Delta t$

$$
\begin{aligned}
R_{n+1} &= R_n + (kR_{n+1} - aR_{n+1}W_{n+1})\Delta t, \\
W_{n+1} &= W_n + (-rW_{n+1} + bR_{n+1}W_{n+1})\Delta t.
\end{aligned}
$$

- **Question**: How to solve it?

Implicit Euler's method

- Implicit Euler's method: $\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{f}(\mathbf{y}_{n+1})\Delta t$

$$
\begin{aligned}
R_{n+1} &= R_n + (kR_{n+1} - aR_{n+1}W_{n+1})\Delta t, \\
W_{n+1} &= W_n + (-rW_{n+1} + bR_{n+1}W_{n+1})\Delta t.
\end{aligned}
$$

- **Question**: How to solve it?
- Root-finding: $\mathbf{y}_{n+1} - \mathbf{y}_n - \mathbf{f}(\mathbf{y}_{n+1})\Delta t = \mathbf{0}$
- Newton's method:
$\mathbf{y}_{n+1}^{[i+1]} = \mathbf{y}_{n+1}^{[i]} - (\mathbf{I} - \mathbf{J}(\mathbf{y}_{n+1}^{[i]})\Delta t)^{-1}(\mathbf{y}_{n+1}^{[i]} - \mathbf{y}_n - \mathbf{f}(\mathbf{y}_{n+1}^{[i]})\Delta t)$

Implicit Euler's method

- Implicit Euler's method: $\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{f}(\mathbf{y}_{n+1})\Delta t$

$$\begin{aligned} R_{n+1} &= R_n + (kR_{n+1} - aR_{n+1}W_{n+1})\Delta t, \\ W_{n+1} &= W_n + (-rW_{n+1} + bR_{n+1}W_{n+1})\Delta t. \end{aligned}$$

- **Question**: How to solve it?
- Root-finding: $\mathbf{y}_{n+1} - \mathbf{y}_n - \mathbf{f}(\mathbf{y}_{n+1})\Delta t = \mathbf{0}$
- Newton's method:
$\mathbf{y}_{n+1}^{[i+1]} = \mathbf{y}_{n+1}^{[i]} - (\mathbf{I} - \mathbf{J}(\mathbf{y}_{n+1}^{[i]})\Delta t)^{-1}(\mathbf{y}_{n+1}^{[i]} - \mathbf{y}_n - \mathbf{f}(\mathbf{y}_{n+1}^{[i]})\Delta t)$
- Jacobian matrix:

$$\mathbf{J}(\mathbf{y}) = \begin{bmatrix} k - aW & -aR \\ bW & -r + bR \end{bmatrix}$$

- **Remark**: More complicated than the explicit Euler method!

Trapezoidal's rule

- Trapezoidal's rule:

Trapezoidal's rule

- Trapezoidal's rule: $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{2}(\mathbf{f}(\mathbf{y}_n) + \mathbf{f}(\mathbf{y}_{n+1}))$

$$R_{n+1} = R_n + \frac{\Delta t}{2}(kR_n - aR_nW_n + kR_{n+1} - aR_{n+1}W_{n+1}),$$
$$W_{n+1} = W_n + \frac{\Delta t}{2}(-rW_n + bR_nW_n - rW_{n+1} + bR_{n+1}W_{n+1}).$$

- **Question**: How to solve it?

Trapezoidal's rule

- Trapezoidal's rule: $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{2}(\mathbf{f}(\mathbf{y}_n) + \mathbf{f}(\mathbf{y}_{n+1}))$

$$R_{n+1} = R_n + \frac{\Delta t}{2}(kR_n - aR_nW_n + kR_{n+1} - aR_{n+1}W_{n+1}),$$
$$W_{n+1} = W_n + \frac{\Delta t}{2}(-rW_n + bR_nW_n - rW_{n+1} + bR_{n+1}W_{n+1}).$$

- **Question**: How to solve it?
- Root-finding: $\mathbf{y}_{n+1} - \mathbf{y}_n - \frac{\Delta t}{2}(\mathbf{f}(\mathbf{y}_n) + \mathbf{f}(\mathbf{y}_{n+1})) = \mathbf{0}$
- Newton's method: $\mathbf{y}_{n+1}^{[i+1]} = \mathbf{y}_{n+1}^{[i]} - \left(\mathbf{I} - \frac{\Delta t}{2}\mathbf{J}(\mathbf{y}_{n+1}^{[i]})\right)^{-1}\left(\mathbf{y}_{n+1}^{[i]} - \mathbf{y}_n - \frac{\Delta t}{2}(\mathbf{f}(\mathbf{y}_n) + \mathbf{f}(\mathbf{y}_{n+1}^{[i]}))\right)$

Trapezoidal's rule

- Trapezoidal's rule: $\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{2}(\mathbf{f}(\mathbf{y}_n) + \mathbf{f}(\mathbf{y}_{n+1}))$

$$R_{n+1} = R_n + \frac{\Delta t}{2}(kR_n - aR_nW_n + kR_{n+1} - aR_{n+1}W_{n+1}),$$
$$W_{n+1} = W_n + \frac{\Delta t}{2}(-rW_n + bR_nW_n - rW_{n+1} + bR_{n+1}W_{n+1}).$$

- **Question**: How to solve it?
- Root-finding: $\mathbf{y}_{n+1} - \mathbf{y}_n - \frac{\Delta t}{2}(\mathbf{f}(\mathbf{y}_n) + \mathbf{f}(\mathbf{y}_{n+1})) = \mathbf{0}$
- Newton's method: $\mathbf{y}_{n+1}^{[i+1]} = \mathbf{y}_{n+1}^{[i]} - \left(\mathbf{I} - \frac{\Delta t}{2}\mathbf{J}(\mathbf{y}_{n+1}^{[i]})\right)^{-1}\left(\mathbf{y}_{n+1}^{[i]} - \mathbf{y}_n - \frac{\Delta t}{2}(\mathbf{f}(\mathbf{y}_n) + \mathbf{f}(\mathbf{y}_{n+1}^{[i]}))\right)$
- Jacobian matrix:

$$\mathbf{J}(\mathbf{y}) = \begin{bmatrix} k - aW & -aR \\ bW & -r + bR \end{bmatrix}$$

High-order using polynomials

- For simplicity, focus on $[0, \Delta t]$ and use n nodes

High-order using polynomials

- For simplicity, focus on $[0, \Delta t]$ and use n nodes
- $\mathbf{r} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix}, \mathbf{w} = \begin{bmatrix} W_1 \\ \vdots \\ W_n \end{bmatrix}$
- $\mathbf{f}_1(\mathbf{r}, \mathbf{w}) = \begin{bmatrix} kR_1 - aR_1W_1 \\ \vdots \\ kR_n - aR_nW_n \end{bmatrix}, \mathbf{f}_2(\mathbf{r}, \mathbf{w}) = \begin{bmatrix} -rW_1 + bR_1W_1 \\ \vdots \\ -rW_n + bR_nW_n \end{bmatrix}$
- Discretization:

$$\begin{bmatrix} \mathbf{r} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_0 \\ \mathbf{w}_0 \end{bmatrix} + \Delta t \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{f}_1(\mathbf{r}, \mathbf{w}) \\ \mathbf{f}_2(\mathbf{r}, \mathbf{w}) \end{bmatrix}.$$

- In general, write $\mathbf{y} = \mathbf{y}_0 + \Delta t \mathbf{S} \otimes \mathbf{F}(\mathbf{y})$
- Solve by Newton's method

N-body simulation

- Position vector (x_i, y_i, z_i) and Velocity vector (u_i, v_i, w_i)
- Hamiltonian formalism

$$\frac{dx_1}{dt} = u_1,$$
$$\frac{dy_1}{dt} = v_1,$$
$$\frac{dz_1}{dt} = w_1,$$
$$\frac{du_1}{dt} = -\frac{Gm_2(x_1 - x_2)}{d_{1,2}^3} - \frac{Gm_3(x_1 - x_3)}{d_{1,3}^3},$$
$$\frac{dv_1}{dt} = -\frac{Gm_2(y_1 - y_2)}{d_{1,2}^3} - \frac{Gm_3(y_1 - y_3)}{d_{1,3}^3},$$
$$\frac{dw_1}{dt} = -\frac{Gm_2(z_1 - z_2)}{d_{1,2}^3} - \frac{Gm_3(z_1 - z_3)}{d_{1,3}^3},$$
$$\vdots$$

Computation comparison

- $\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y})$
- Assume n particles, then we have $6n$ equations
- Fix the same interval $[0, \Delta t]$
- Explicit Euler's method: $\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{f}(\mathbf{y}_n)\Delta t$
- Major work: evaluate $\mathbf{f}(\mathbf{y}_n)$ of size $6n$, roughly $\mathcal{O}(n^2)$
- Implicit Euler's method: $\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{f}(\mathbf{y}_{n+1})\Delta t$
- Major work: solve root-finding of size $6n$, roughly $\mathcal{O}((n)^3)$
- Pseudo-spectral method using m nodes:
  $\mathbf{Y}_{n+1} = \mathbf{Y}_n + \Delta t \mathbf{S} \otimes \mathbf{F}(\mathbf{Y}_n)$
- Major work: solve root-finding of size $6mn$, roughly $\mathcal{O}((mn)^3)$
- **Question**: Why pseudo-spectral?
- Better solution after solved, much better accuracy