Simulation and High-Performance Computing

Part 1: Introduction to Time-Stepping Methods

Steffen Börm

Christian-Albrechts-Universität zu Kiel

September 28th, 2020

Simulations

Traditional experiments: Laws of nature are derived from observations.

Numerical experiments: Nature approximated by mathematical model.

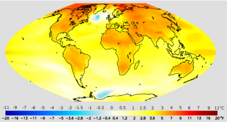
Simulations

Traditional experiments: Laws of nature are derived from observations.

Numerical experiments: Nature approximated by mathematical model.

Advantages

- We can perform experiments on a computer that would be impossible in the real world.



Source: NOAA (via Wikipedia)

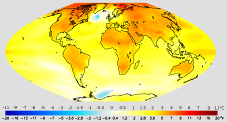
Simulations

Traditional experiments: Laws of nature are derived from observations.

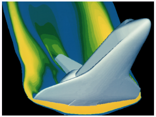
Numerical experiments: Nature approximated by mathematical model.

Advantages

- We can perform experiments on a computer that would be impossible in the real world.
- We can perform experiments on a computer that would be too dangerous or expensive in the real world.



Source: NOAA (via Wikipedia)



Source: NASA (via Wikipedia)

Simulations

Traditional experiments: Laws of nature are derived from observations.

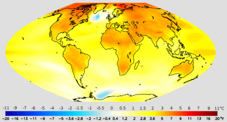
Numerical experiments: Nature approximated by mathematical model.

Advantages

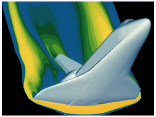
- We can perform experiments on a computer that would be impossible in the real world.
- We can perform experiments on a computer that would be too dangerous or expensive in the real world.

Disadvantages

- If the mathematical model is wrong, so are the results.
- Large-scale simulations require powerful computers.



Source: NOAA (via Wikipedia)



Source: NASA (via Wikipedia)

Source: D. N. Arnold

Overview

First week: Numerical algorithms

1. Time-stepping methods

Overview

First week: Numerical algorithms

1. Time-stepping methods
2. Higher-order time-stepping methods

Overview

First week: Numerical algorithms

1. Time-stepping methods
2. Higher-order time-stepping methods
3. Finite difference methods for PDEs

Overview

First week: Numerical algorithms

1. Time-stepping methods
2. Higher-order time-stepping methods
3. Finite difference methods for PDEs
4. Iterative solvers

Overview

First week: Numerical algorithms

1. Time-stepping methods
2. Higher-order time-stepping methods
3. Finite difference methods for PDEs
4. Iterative solvers
5. Solvers for large systems

Overview

First week: Numerical algorithms

1. Time-stepping methods
2. Higher-order time-stepping methods
3. Finite difference methods for PDEs
4. Iterative solvers
5. Solvers for large systems

Second week: High-performance computing

1. Shared-memory parallelization (OpenMP)

Overview

First week: Numerical algorithms

1. Time-stepping methods
2. Higher-order time-stepping methods
3. Finite difference methods for PDEs
4. Iterative solvers
5. Solvers for large systems

Second week: High-performance computing

1. Shared-memory parallelization (OpenMP)
2. Vectorization

Overview

First week: Numerical algorithms

1. Time-stepping methods
2. Higher-order time-stepping methods
3. Finite difference methods for PDEs
4. Iterative solvers
5. Solvers for large systems

Second week: High-performance computing

1. Shared-memory parallelization (OpenMP)
2. Vectorization
3. GPU computing (CUDA)

Overview

First week: Numerical algorithms

1. Time-stepping methods
2. Higher-order time-stepping methods
3. Finite difference methods for PDEs
4. Iterative solvers
5. Solvers for large systems

Second week: High-performance computing

1. Shared-memory parallelization (OpenMP)
2. Vectorization
3. GPU computing (CUDA)
4. Distributed computing (MPI)

Overview

First week: Numerical algorithms

1. Time-stepping methods
2. Higher-order time-stepping methods
3. Finite difference methods for PDEs
4. Iterative solvers
5. Solvers for large systems

Second week: High-performance computing

1. Shared-memory parallelization (OpenMP)
2. Vectorization
3. GPU computing (CUDA)
4. Distributed computing (MPI)
5. Kiel University's computing center

Example: Mass-spring systems

Newton: Axioms of classical mechanics.

- Each body has a time-dependent position $x(t)$.
- It moves at a velocity $v(t) = x'(t)$.
- The velocity changes in response to forces $f(t) = m\,v'(t)$, where m is the body's mass.

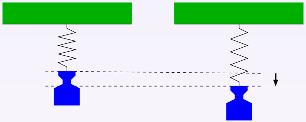
Example: Mass-spring systems

Newton: Axioms of classical mechanics.

- Each body has a time-dependent position $x(t)$.
- It moves at a velocity $v(t) = x'(t)$.
- The velocity changes in response to forces $f(t) = m\,v'(t)$, where m is the body's mass.

Hooke: Force exerted by a spring.

$$f(t) = -c\,x(t)$$

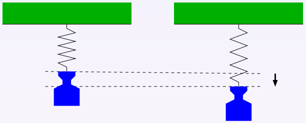


Example: Mass-spring systems

Newton: Axioms of classical mechanics.

- Each body has a time-dependent position $x(t)$.
- It moves at a velocity $v(t) = x'(t)$.
- The velocity changes in response to forces $f(t) = m\,v'(t)$, where m is the body's mass.

Hooke: Force exerted by a spring.



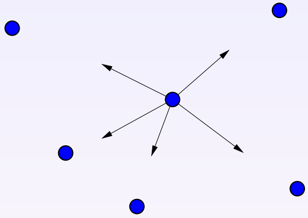
$$f(t) = -c\,x(t)$$

Result: Coupled ordinary differential equations.

$$x'(t) = v(t), \qquad v'(t) = -\frac{c}{m}x(t).$$

Example: Gravity

Generalization: Multiple bodies at positions $x_i(t)$ with velocities $v_i(t)$, masses m_i, and forces $f_i(t)$.

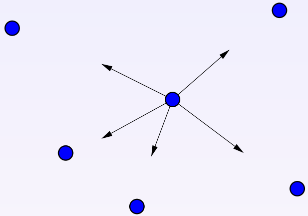


Newton: Gravitational force given by

$$f_i(t) = \kappa\, m_i \sum_{j \neq i} m_j \frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|^3}.$$

Example: Gravity

Generalization: Multiple bodies at positions $x_i(t)$ with velocities $v_i(t)$, masses m_i, and forces $f_i(t)$.



Newton: Gravitational force given by

$$f_i(t) = \kappa\, m_i \sum_{j \neq i} m_j \frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|^3}.$$

Result: Coupled ordinary differential equations.

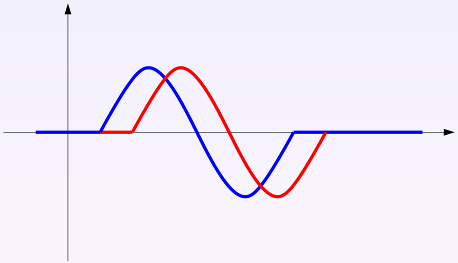
$$x_i'(t) = v_i(t), \qquad v_i'(t) = \kappa \sum_{j \neq i} m_j \frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|^3}.$$

Example: Wave equation

Model: Displacement of point s at time t given by $x(t,s)$, velocity $v(t,s)$.

Elasticity: Stress caused by deformation of the medium.

$$f(t,s) = c_{el} \frac{\partial^2 x}{\partial s^2}(t,s)$$

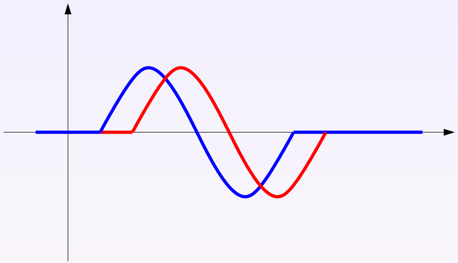


Example: Wave equation

Model: Displacement of point s at time t given by $x(t,s)$, velocity $v(t,s)$.

Elasticity: Stress caused by deformation of the medium.

$$f(t,s) = c_{el} \frac{\partial^2 x}{\partial s^2}(t,s)$$



Result: Coupled partial differential equations.

$$\frac{\partial x}{\partial t}(t,s) = v(t,s), \qquad \frac{\partial v}{\partial t}(t,s) = c \frac{\partial^2 x}{\partial s^2}(t,s).$$

Example: Predator-prey model

Lotka-Volterra model for predator-prey populations.

- $b(t)$ is the "number" of prey at time t.
- $r(t)$ is the "number" of predators.
- α is the reproduction rate of prey.
- β and κ describe predators feeding on prey.
- ω is the starvation rate of predators.

Model: Coupled ordinary differential equations.

$$b'(t) = b(t)\big(\alpha - \beta\, r(t)\big), \qquad r'(t) = r(t)\big(\kappa\, b(t) - \omega\big).$$

Explicit ordinary differential equation

Common form of all these examples: $y'(t) = f(t, y(t))$ with

- state variables collected in a vector $y(t)$ and
- derivative in state z at time t given by a function $f(t, z)$.

Example: Mass-spring system

$$y(t) = \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}, \qquad f(t, z) = \begin{pmatrix} z_2 \\ -\frac{c}{m} z_1 \end{pmatrix}.$$

Example: Predator-prey model

$$y(t) = \begin{pmatrix} b(t) \\ r(t) \end{pmatrix}, \qquad f(t, z) = \begin{pmatrix} z_1(\alpha + \beta z_2) \\ z_2(\kappa z_1 - \omega) \end{pmatrix}.$$

Explicit Euler method

Challenge: Solving $y'(t) = f(t, y(t))$ by hand is usually hard, since the state appears on both sides of the equation.

Idea: Use a sufficiently accurate approximation.

Forward difference quotient: Taylor expansion yields $\eta \in [t, t+\delta]$ with

$$y(t+\delta) = y(t) + \delta\, y'(t) + \frac{\delta^2}{2} y''(\eta),$$
$$\frac{y(t+\delta) - y(t)}{\delta} = y'(t) + \frac{\delta}{2} y''(\eta)$$

Explicit Euler method

Challenge: Solving $y'(t) = f(t, y(t))$ by hand is usually hard, since the state appears on both sides of the equation.

Idea: Use a sufficiently accurate approximation.

Forward difference quotient: Taylor expansion yields $\eta \in [t, t+\delta]$ with

$$y(t+\delta) = y(t) + \delta\, y'(t) + \frac{\delta^2}{2} y''(\eta),$$
$$\frac{y(t+\delta) - y(t)}{\delta} = y'(t) + \frac{\delta}{2} y''(\eta)$$

Explicit Euler: Drop last term, replace $y'(t)$ using the differential equation.

$$y(t+\delta) \approx \tilde{y}(t+\delta) := y(t) + \delta\, f(t, y(t)).$$

Example: Explicit Euler for the mass-spring system

Mass-spring system: We have

$$y(t) = \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}, \qquad f(t,z) = \begin{pmatrix} z_2 \\ -\frac{c}{m} z_1 \end{pmatrix},$$

therefore $y(t+\delta) \approx y(t) + \delta\, f(t, y(t))$ takes the form

$$x(t+\delta) \approx \tilde{x}(t+\delta) := x(t) + \delta\, v(t),$$
$$v(t+\delta) \approx \tilde{v}(t+\delta) := v(t) - \delta \frac{c}{m} x(t).$$

Example: Explicit Euler for the mass-spring system

Mass-spring system: We have

$$y(t) = \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}, \qquad f(t,z) = \begin{pmatrix} z_2 \\ -\frac{c}{m} z_1 \end{pmatrix},$$

therefore $y(t+\delta) \approx y(t) + \delta\, f(t, y(t))$ takes the form

$$x(t+\delta) \approx \tilde{x}(t+\delta) := x(t) + \delta\, v(t),$$
$$v(t+\delta) \approx \tilde{v}(t+\delta) := v(t) - \delta \frac{c}{m} x(t).$$

Implementation in C:

```
dx = v;                    /* Compute derivative */
dv = -c/m * x;
x += delta * dx;           /* Update position */
v += delta * dv;           /* Update velocity */
```

Experiment: Explicit Euler for the mass-spring system

Approach: Start at $t = 0$, perform successive timesteps to reach $t = 20$.

δ	error	ratio
1	1.0_{+3}	
$1/2$	8.2_{+1}	12.2
$1/4$	7.9_{+0}	10.4
$1/8$	1.3_{+0}	6.1
$1/16$	4.0_{-1}	3.3
$1/32$	1.6_{-1}	2.5
$1/64$	7.1_{-2}	2.3
$1/128$	3.4_{-2}	2.1
$1/256$	1.6_{-2}	2.1

First-order convergence: Halving the timestep size, i.e., doubling the number of timesteps, only halves the error.

Central difference quotient

Idea: Replace forward difference quotient by a better approximation.

Taylor expansion yields $\eta_+ \in [t, t+\delta]$ and $\eta_- \in [t-\delta, t]$ with

$$y(t+\delta) = y(t) + \delta y'(t) + \frac{\delta^2}{2} y''(t) + \frac{\delta^3}{6} y'''(\eta_+),$$
$$y(t-\delta) = y(t) - \delta y'(t) + \frac{\delta^2}{2} y''(t) - \frac{\delta^3}{6} y'''(\eta_-).$$

Central difference quotient

Idea: Replace forward difference quotient by a better approximation.

Taylor expansion yields $\eta_+ \in [t, t+\delta]$ and $\eta_- \in [t-\delta, t]$ with

$$y(t+\delta) = y(t) + \delta y'(t) + \frac{\delta^2}{2} y''(t) + \frac{\delta^3}{6} y'''(\eta_+),$$
$$y(t-\delta) = y(t) - \delta y'(t) + \frac{\delta^2}{2} y''(t) - \frac{\delta^3}{6} y'''(\eta_-).$$

Intermediate value theorem gives us $\eta \in [t-\delta, t+\delta]$ with

$$y(t+\delta) - y(t-\delta) = 2\delta\, y'(t) + \frac{\delta^3}{3} y'''(\eta),$$
$$\frac{y(t+\delta) - y(t-\delta)}{2\delta} = y'(t) + \frac{\delta^2}{6} y'''(\eta).$$

Runge's method

First idea: Central difference quotient

$$y(t+\delta) \approx y(t) + \delta\, y'(t+\tfrac{\delta}{2}) = y(t) + \delta\, f(t+\tfrac{\delta}{2}, y(t+\tfrac{\delta}{2})).$$

Runge's method

First idea: Central difference quotient

$$y(t+\delta) \approx y(t) + \delta\, y'(t+\tfrac{\delta}{2}) = y(t) + \delta\, f(t+\tfrac{\delta}{2}, y(t+\tfrac{\delta}{2})).$$

Second idea: Approximate midpoint state $y(t+\frac{\delta}{2})$ using explicit Euler.

$$\tilde{y}(t+\tfrac{\delta}{2}) := y(t) + \tfrac{\delta}{2} f(t, y(t)), \quad \tilde{y}(t+\delta) := y(t) + \delta\, f(t+\tfrac{\delta}{2}, \tilde{y}(t+\tfrac{\delta}{2})).$$

Runge's method

First idea: Central difference quotient

$$y(t+\delta) \approx y(t) + \delta\, y'(t+\tfrac{\delta}{2}) = y(t) + \delta\, f(t+\tfrac{\delta}{2}, y(t+\tfrac{\delta}{2})).$$

Second idea: Approximate midpoint state $y(t+\frac{\delta}{2})$ using explicit Euler.

$$\tilde{y}(t+\tfrac{\delta}{2}) := y(t) + \tfrac{\delta}{2} f(t, y(t)), \quad \tilde{y}(t+\delta) := y(t) + \delta\, f(t+\tfrac{\delta}{2}, \tilde{y}(t+\tfrac{\delta}{2})).$$

Implementation in C for the mass-spring system:

```c
/* Approximate midpoint state */
xm = x + 0.5 * delta * v;
vm = v - 0.5 * delta * c / m * x;

/* Approximate next state */
x += delta * vm;
v -= delta * c / m * xm;
```

Experiment: Runge's method for the mass-spring system

Approach: Start at $t = 0$, perform successive timesteps to reach $t = 20$.

	Euler		Runge	
δ	error	ratio	error	ratio
1	1.0_{+3}		9.6_{+0}	
1/2	8.2_{+1}	12.2	8.7_{-1}	11.0
1/4	7.9_{+0}	10.4	1.9_{-1}	4.6
1/8	1.3_{+0}	6.1	4.6_{-2}	4.1
1/16	4.0_{-1}	3.3	1.2_{-2}	3.8
1/32	1.6_{-1}	2.5	2.9_{-3}	4.1
1/64	7.1_{-2}	2.3	7.4_{-4}	3.9
1/128	3.4_{-2}	2.1	1.9_{-4}	3.9
1/256	1.6_{-2}	2.1	4.6_{-5}	4.1

Second-order convergence: The error in Runge's method behaves like δ^2, i.e., doubling the number of time steps quarters the error.

Predator-prey: Implementation

Lotka-Volterra model:

$$b'(t) = b(t)\big(\alpha - \beta\, r(t)\big), \qquad r'(t) = r(t)\big(\kappa\, b(t) - \omega\big).$$

Explicit Euler in C: Derivatives stored in `db` and `dr`

```
db = b * (alpha - beta * r);
dr = r * (kappa * b - omega);

b += delta * db;
r += delta * dr;
```

Predator-prey: Implementation

Lotka-Volterra model:

$$b'(t) = b(t)\big(\alpha - \beta\, r(t)\big), \qquad r'(t) = r(t)\big(\kappa\, b(t) - \omega\big).$$

Explicit Euler in C: Derivatives stored in `db` and `dr`

```
db = b * (alpha - beta * r);
dr = r * (kappa * b - omega);

b += delta * db;
r += delta * dr;
```

Runge's method in C: Midpoint state stored in `bm`, `rm`

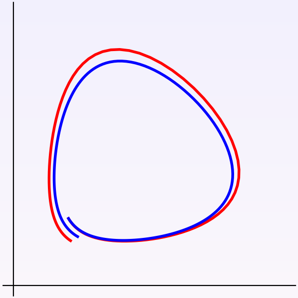
```
bm = b + 0.5 * delta * (alpha - beta * r);
rm = r + 0.5 * delta * (kappa * b - omega);

b += delta * bm * (alpha - beta * rm);
r += delta * rm * (kappa * bm - omega);
```

Predator-prey: Euler vs Runge

Explicit Euler

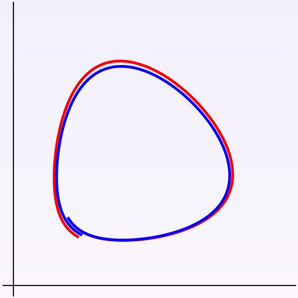
with 100/200 timesteps.



Predator-prey: Euler vs Runge

Explicit Euler

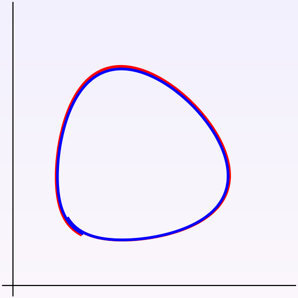
with 200/400 timesteps.



Predator-prey: Euler vs Runge

Explicit Euler

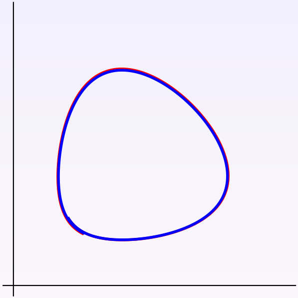
with 400/800 timesteps.



Predator-prey: Euler vs Runge

Explicit Euler

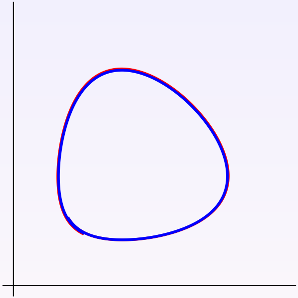
with 800/1600 timesteps.



Predator-prey: Euler vs Runge

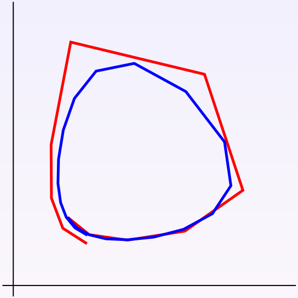
Explicit Euler

with 800/1600 timesteps.



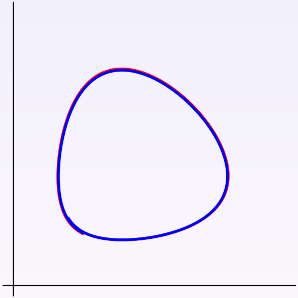
Runge's method

with 10/20 timesteps.

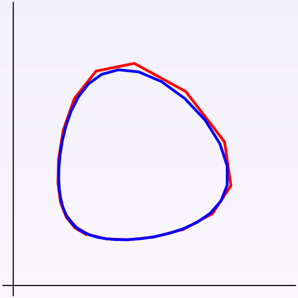


Predator-prey: Euler vs Runge

Explicit Euler
with 800/1600 timesteps.

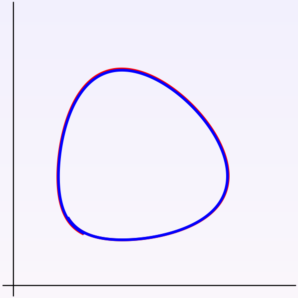


Runge's method
with 20/40 timesteps.

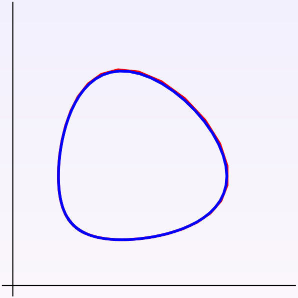


Predator-prey: Euler vs Runge

Explicit Euler
with 800/1600 timesteps.

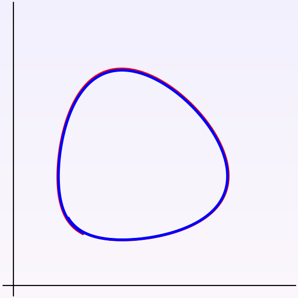


Runge's method
with 40/80 timesteps.

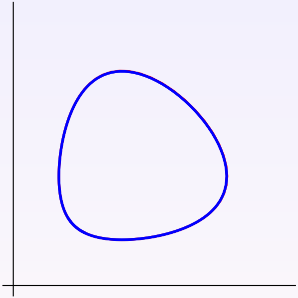


Predator-prey: Euler vs Runge

Explicit Euler
with 800/1600 timesteps.



Runge's method
with 80/160 timesteps.



Gravity: Implementation I

Gravitational acceleration given by

$$f_i = \sum_{j \neq i} m_j \frac{x_j - x_i}{\|x_j - x_i\|^3}.$$

Implementation in C:

```
f0 = 0.0;
f1 = 0.0;
for(j=0; j<n; j++)
  if(j != i) {
    d0 = x0[j] - x0[i];
    d1 = x1[j] - x1[i];
    dist2 = d0 * d0 + d1 * d1;
    alpha = m[j] / (dist2 * sqrt(dist2));
    f0 += alpha * d0;
    f1 += alpha * d1;
  }
```

Gravity: Implementation II

Explicit Euler in C:

```
force(n, x0, x1, m, f0, f1);

for(i=0; i<n; i++) {
  x0[i] += delta * v0[i];
  x1[i] += delta * v1[i];

  v0[i] += delta * f0[i];
  v1[i] += delta * f1[i];
}
```

Gravity: Implementation III

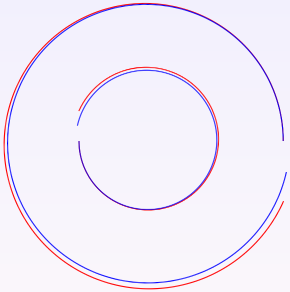
Runge's method in C:

```
force(n, x0, x1, m, f0, f1);
for(i=0; i<n; i++) {
  xm0[i] = x0[i] + 0.5 * delta * v0[i];
  xm1[i] = x1[i] + 0.5 * delta * v1[i];
  vm0[i] = v0[i] + 0.5 * delta * f0[i];
  vm1[i] = v1[i] + 0.5 * delta * f1[i];
}

force(n, xm0, xm1, m, f0, f1);
for(i=0; i<n; i++) {
  x0[i] += delta * vm0[i];
  x1[i] += delta * vm1[i];
  v0[i] += delta * f0[i];
  v1[i] += delta * f1[i];
}
```

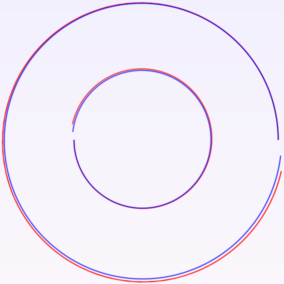
Gravity: Euler vs Runge

Explicit Euler
with 800/1600 timesteps.



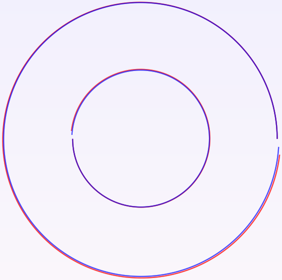
Gravity: Euler vs Runge

Explicit Euler
with 1600/3200 timesteps.



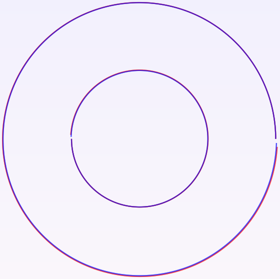
Gravity: Euler vs Runge

Explicit Euler
with 3200/6400 timesteps.



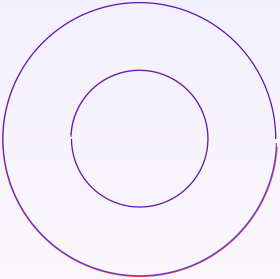
Gravity: Euler vs Runge

Explicit Euler
with 6400/12800 timesteps.

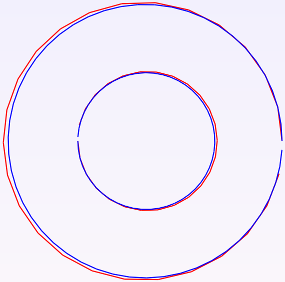


Gravity: Euler vs Runge

Explicit Euler
with 6400/12800 timesteps.

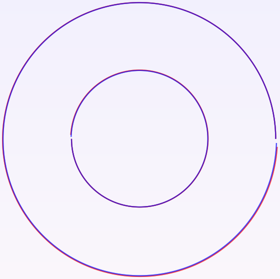


Runge's method
with 25/50 timesteps.

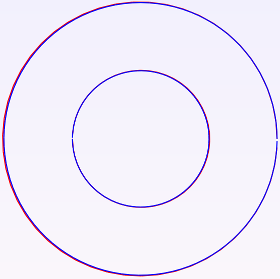


Gravity: Euler vs Runge

Explicit Euler
with 6400/12800 timesteps.

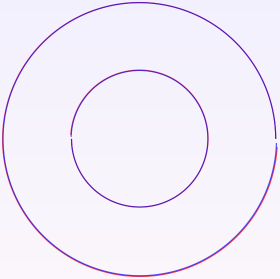


Runge's method
with 50/100 timesteps.

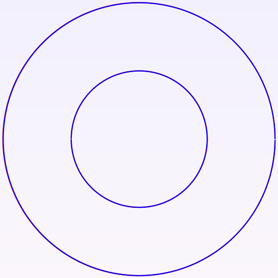


Gravity: Euler vs Runge

Explicit Euler
with 6400/12800 timesteps.



Runge's method
with 100/200 timesteps.



Summary

Examples: Mass-spring system, gravity, predator-prey populations.

Initial value problem: Find y with $y(0) = y_0$ and

$$y'(t) = f(t, y(t)) \qquad \text{for all } t \in \mathbb{R}.$$

Euler's method: Approximate derivative by forward difference quotient.

$$\frac{y(t+\delta) - y(t)}{\delta} \approx y'(t) = f(t, y(t)), \quad \tilde{y}(t+\delta) := y(t) + \delta\, f(t, y(t)).$$

Runge's method: Approximate derivative by central difference quotient.

$$\frac{y(t+\delta) - y(t)}{\delta} \approx y'(t+\tfrac{\delta}{2}) = f(t+\tfrac{\delta}{2}, y(t+\tfrac{\delta}{2})),$$

$$\tilde{y}(t+\tfrac{\delta}{2}) := y(t) + \tfrac{\delta}{2} f(t, y(t)), \quad \tilde{y}(t+\delta) := y(t) + \delta\, f(t+\tfrac{\delta}{2}, \tilde{y}(t+\tfrac{\delta}{2})).$$