Overview

1. Projects
2. Ordinary differential equations
   - The Euler method
3. Higher-order ODEs
4. Stability
5. Summary

Next quiz 24/26 March!

Projects

Assessment criteria

1. Ability to write code as indicated in the project description
2. Ability to formulate the problem and select appropriate methods (where this is not completely specified in the brief)
3. Interpreting and discussing results, putting them into context and suggesting avenues for further research. This includes the ability to identify wrong or suspicious results (which may result from bugs in your code).
4. Clarity of presentation in report (including figures and tables), and readability of code
5. Quality of report

Projects

Tips for working in groups

1. Meet up early to agree division of labour
2. Let the supervisor know if someone is not contributing or is not allowing the others to contribute
3. Do not let conflicts fester — it is better to break up a dysfunctional group
4. It is best if all contribute to both coding and report writing

Ordinary differential equations

Consider again the integral $I = \int_a^b f(x)dx$.

Ordinary differential equations

Consider again the integral $I = \int_a^b f(x)dx$. Introduce the function

$$I(x) = \int_a^x f(\xi)d\xi \quad \Longrightarrow \; I(a) = 0, \; I(b) = I \, .$$

Ordinary differential equations

Consider again the integral $I = \int_a^b f(x)dx$. Introduce the function

$$I(x) = \int_a^x f(\xi)d\xi \quad \Longrightarrow \; I(a) = 0, \; I(b) = I \, .$$

This function obeys the ODE

$$\frac{dI}{dx} = f(x) \, , \quad I(a) = 0$$

Our integral has been transformed into an ODE inital-value problem!

Ordinary differential equations

Consider again the integral $I = \int_a^b f(x)dx$. Introduce the function

$$I(x) = \int_a^x f(\xi)d\xi \quad \Longrightarrow \; I(a) = 0, \; I(b) = I \, .$$

This function obeys the ODE

$$\frac{dI}{dx} = f(x) \, , \quad I(a) = 0$$

Our integral has been transformed into an ODE inital-value problem!

Main advantage

Powerful methods exist for solving ODEs

- we will acquaint ourselves with these methods
- they apply to general, first-order or higher-order odes

$$\frac{dy}{dx} = f(x, y) \, , \qquad \frac{d^2y}{dx^2} + b(x, y)\frac{dy}{dx} + c(x, y) = 0 \, , \quad \text{etc}$$

The Euler method

Let us try to solve the equation

$$\frac{dy}{dx} = f(x) \qquad \text{starting from} \quad y(a) = c \quad x \in [a, b]$$

The Euler method

Let us try to solve the equation

$$\frac{dy}{dx} = f(x) \qquad \text{starting from} \quad y(a) = c \quad x \in [a, b]$$

We divide the interval into N segments with length $\varepsilon = (b - a)/N$
Number the points

$$x_0 = a, x_1 = a + \varepsilon, \ldots, x_N = b$$

We want to compute the values

$$y_n = y(x_n) = y(x_0 + n\varepsilon) \quad \forall n = 1, \ldots, N$$

We know $y(x_0) = y_0 = c$.

The Euler method

We know $y(x_0) = y_0 = c$.
We can calculate $v_0 = y'(x_0) = f(x_0)$

The Euler method

We know $y(x_0) = y_0 = c$.
We can calculate $v_0 = y'(x_0) = f(x_0)$

Take a small step ε:

$$y_1 = y(x_1) = y(x_0 + \varepsilon) = y(x_0) + \varepsilon y'(x_0) = y_0 + \varepsilon v_0$$
$$v_1 = y'(x_1) = f(x_1)$$

Iterate this to get $y_2, y_3, \ldots, y_N$

The Euler method

We know $y(x_0) = y_0 = c$.
We can calculate $v_0 = y'(x_0) = f(x_0)$

Take a small step ε:

$$y_1 = y(x_1) = y(x_0 + \varepsilon) = y(x_0) + \varepsilon y'(x_0) = y_0 + \varepsilon v_0$$
$$v_1 = y'(x_1) = f(x_1)$$

Iterate this to get $y_2, y_3, \ldots, y_N$

Algorithm

Set $x_0 = a, y_0 = c$, choose stepsize ε or $N = (b - a)/\varepsilon$
Calculate $v_0 = f(x_0, y_0)$.
Then, for each $n = 1, \ldots, N$ do

1. $x_n = x_{n-1} + \varepsilon$
2. $y_n = y_{n-1} + \varepsilon v_{n-1}$
3. $v_n = f(x_n, y_n)$

Error estimate

Taylor expand as before:

$$\begin{aligned} y^{\text{true}}(x_n) &= y(x_{n-1}) + \varepsilon y'(x_{n-1}) + \frac{\varepsilon^2}{2} y''(x_{n-1}) + \cdots \\ &= y_n^E + \mathcal{O}(\varepsilon^2) \end{aligned} \tag{1}$$

Error estimate

Taylor expand as before:

$$\begin{aligned} y^{\text{true}}(x_n) &= y(x_{n-1}) + \varepsilon y'(x_{n-1}) + \frac{\varepsilon^2}{2} y''(x_{n-1}) + \cdots \\ &= y_n^E + \mathcal{O}(\varepsilon^2) \end{aligned} \tag{1}$$

Each step induces an error of $\mathcal{O}(\varepsilon^2)$.
The number of steps is $N = (b-a)/\varepsilon$ [or $n = (x_n - a)/\varepsilon$]

Real error

$$y^{\text{true}} - y^E \sim N \cdot \mathcal{O}(\varepsilon^2) \sim \mathcal{O}(\varepsilon)$$

Error estimate

Taylor expand as before:

$$\begin{aligned} y^{\text{true}}(x_n) &= y(x_{n-1}) + \varepsilon y'(x_{n-1}) + \frac{\varepsilon^2}{2} y''(x_{n-1}) + \cdots \\ &= y_n^E + \mathcal{O}(\varepsilon^2) \end{aligned} \tag{1}$$

Each step induces an error of $\mathcal{O}(\varepsilon^2)$.
The number of steps is $N = (b-a)/\varepsilon$ [or $n = (x_n - a)/\varepsilon$]

Real error

$$y^{\text{true}} - y^E \sim N \cdot \mathcal{O}(\varepsilon^2) \sim \mathcal{O}(\varepsilon)$$

This analysis holds in exactly the same way for a general ODE

$$\frac{dy}{dx} = f(x, y)$$

The Euler method is accurate up to $\mathcal{O}(\varepsilon)$

Higher order ODEs

Generic second order ODE:

$$\frac{d^2y}{dx^2} + f(x,y)\frac{dy}{dx} + g(x,y) = 0$$

How would we solve this numerically?

Higher order ODEs

Generic second order ODE:

$$\frac{d^2y}{dx^2} + f(x,y)\frac{dy}{dx} + g(x,y) = 0$$

How would we solve this numerically?

I. Try discretising both derivatives

Use symmetric derivatives for both:

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{\varepsilon^2} + f(x_n, y_n)\frac{y_{n+1} - y_{n-1}}{2\varepsilon} + g(x_n, y_n) = 0$$

$$\implies \quad y_{n+1} = \frac{2y_n - \left(1 - \frac{\varepsilon}{2}f(x_n, y_n)\right)y_{n-1} - g(x_n, y_n)}{1 + \frac{\varepsilon}{2}f(x_n, y_n)}$$

Higher order ODEs

Generic second order ODE:

$$\frac{d^2y}{dx^2} + f(x,y)\frac{dy}{dx} + g(x,y) = 0$$

How would we solve this numerically?

I. Try discretising both derivatives

Use symmetric derivatives for both:

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{\varepsilon^2} + f(x_n, y_n)\frac{y_{n+1} - y_{n-1}}{2\varepsilon} + g(x_n, y_n) = 0$$

$$\implies \quad y_{n+1} = \frac{2y_n - \left(1 - \frac{\varepsilon}{2}f(x_n, y_n)\right)y_{n-1} - g(x_n, y_n)}{1 + \frac{\varepsilon}{2}f(x_n, y_n)}$$

- Need to know both y_n and y_{n-1} to compute y_{n+1}
- Can get y_1 from y_0, y_0' using forward derivative, then iterate

Higher order ODEs

Generic second order ODE:

$$\frac{d^2y}{dx^2} + f(x,y)\frac{dy}{dx} + g(x,y) = 0$$

How would we solve this numerically?

I. Try discretising both derivatives

Use symmetric derivatives for both:

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{\varepsilon^2} + f(x_n, y_n)\frac{y_{n+1} - y_{n-1}}{2\varepsilon} + g(x_n, y_n) = 0$$

$$\implies \quad y_{n+1} = \frac{2y_n - \left(1 - \frac{\varepsilon}{2}f(x_n, y_n)\right)y_{n-1} - g(x_n, y_n)}{1 + \frac{\varepsilon}{2}f(x_n, y_n)}$$

Error in derivatives: $\mathcal{O}(\varepsilon^2)$ Error in each step: $\varepsilon^2\mathcal{O}(\varepsilon^2) = \mathcal{O}(\varepsilon^4)$

Higher order ODEs

Generic second order ODE:

$$\frac{d^2y}{dx^2} + f(x,y)\frac{dy}{dx} + g(x,y) = 0$$

How would we solve this numerically?

I. Try discretising both derivatives

Use symmetric derivatives for both:

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{\varepsilon^2} + f(x_n, y_n)\frac{y_{n+1} - y_{n-1}}{2\varepsilon} + g(x_n, y_n) = 0$$

$$\implies \quad y_{n+1} = \frac{2y_n - \left(1 - \frac{\varepsilon}{2}f(x_n, y_n)\right)y_{n-1} - g(x_n, y_n)}{1 + \frac{\varepsilon}{2}f(x_n, y_n)}$$

Error in derivatives: $\mathcal{O}(\varepsilon^2)$ Error in each step: $\varepsilon^2\mathcal{O}(\varepsilon^2) = \mathcal{O}(\varepsilon^4)$

Total error: $\mathcal{O}(\varepsilon^3)$

Higher order ODEs

Generic second order ODE:

$$\frac{d^2y}{dx^2} + f(x,y)\frac{dy}{dx} + g(x,y) = 0$$

II. Make it into two first order ODEs

Define 'new' function $z(x) = y'(x)$

$$\frac{dy}{dx} = z\,, \qquad \frac{dz}{dx} = -f(x,y)z - g(x,y)$$

Higher order ODEs

Generic second order ODE:

$$\frac{d^2y}{dx^2} + f(x,y)\frac{dy}{dx} + g(x,y) = 0$$

II. Make it into two first order ODEs

Define 'new' function $z(x) = y'(x)$

$$\frac{dy}{dx} = z\,, \qquad \frac{dz}{dx} = -f(x,y)z - g(x,y)$$

This system can be solved using the Euler method:

$$y_{n+1} = y_n + \varepsilon z_n$$
$$z_{n+1} = z_n - \varepsilon\big[f(x_n,y_n)z_n + g(x_n,y_n)\big]$$

Higher order ODEs

Generic second order ODE:

$$\frac{d^2y}{dx^2} + f(x,y)\frac{dy}{dx} + g(x,y) = 0$$

II. Make it into two first order ODEs

Define 'new' function $z(x) = y'(x)$

$$\frac{dy}{dx} = z\,, \qquad \frac{dz}{dx} = -f(x,y)z - g(x,y)$$

This system can be solved using the Euler method:

$$y_{n+1} = y_n + \varepsilon z_n$$
$$z_{n+1} = z_n - \varepsilon\big[f(x_n,y_n)z_n + g(x_n,y_n)\big]$$

Error in each step: $\mathcal{O}(\varepsilon^2)$
Total error: $\mathcal{O}(\varepsilon)$

Molecular dynamics

The classical equations of motion for a collection of interacting particles (eg, molecules or planets) will be

- coupled second-order ODEs for positions $x_i(t)$ (Lagrange), or
- coupled first-order ODEs for positions $x_i(t)$ and momenta $p_i(t)$ (Hamilton)

Solving such coupled equations numerically is molecular dynamics.

Molecular dynamics

The classical equations of motion for a collection of interacting particles (eg, molecules or planets) will be

- coupled second-order ODEs for positions $x_i(t)$ (Lagrange), or
- coupled first-order ODEs for positions $x_i(t)$ and momenta $p_i(t)$ (Hamilton)

Solving such coupled equations numerically is molecular dynamics.

Reducing everything to first-order ODEs corresponds to the 'Hamiltonian' approach. This has several advantages, first and foremost:

You can use the same methods for all equations

Molecular dynamics

The classical equations of motion for a collection of interacting particles (eg, molecules or planets) will be

- coupled second-order ODEs for positions $x_i(t)$ (Lagrange), or
- coupled first-order ODEs for positions $x_i(t)$ and momenta $p_i(t)$ (Hamilton)

Solving such coupled equations numerically is molecular dynamics.

Reducing everything to first-order ODEs corresponds to the 'Hamiltonian' approach. This has several advantages, first and foremost:

You can use the same methods for all equations

Black-box solvers (`ode45` etc) require as 'input' a set of first-order equations with initial values.

You may still need physical or mathematical insight to choose the most appropriate method!

Stability 1: 1st order ODEs

To understand stability, let us look at a simple differential equation:

$$\frac{dy}{dx} = -\gamma y = f(x, y)$$

The Euler method

Start with $x_0, y_0, v_0 = f(x_0, y_0)$.

1. $x_{n+1} = x_n + \varepsilon$
2. $y_{n+1} = y_n + \varepsilon f(x_n, y_n)$

Stability 1: 1st order ODEs

To understand stability, let us look at a simple differential equation:

$$\frac{dy}{dx} = -\gamma y = f(x, y)$$

The Euler method

Start with $x_0, y_0, v_0 = f(x_0, y_0)$.

1. $x_{n+1} = x_n + \varepsilon$
2. $y_{n+1} = y_n + \varepsilon f(x_n, y_n)$

Applying the Euler method to our equation, we find

$$y_{n+1} = y_n - \varepsilon\gamma y_n = (1 - \varepsilon\gamma)y_n$$

Stability 1: 1st order ODEs

To understand stability, let us look at a simple differential equation:

$$\frac{dy}{dx} = -\gamma y = f(x, y)$$

The Euler method

Start with $x_0, y_0, v_0 = f(x_0, y_0)$.

1. $x_{n+1} = x_n + \varepsilon$
2. $y_{n+1} = y_n + \varepsilon f(x_n, y_n)$

Applying the Euler method to our equation, we find

$$y_{n+1} = y_n - \varepsilon\gamma y_n = (1 - \varepsilon\gamma)y_n$$

We know that the solution is an exponential decay.
But if $\varepsilon > 2/\gamma$ we will instead get an increase!

Instability

The Euler method is unstable for $\varepsilon > 2/\gamma$

Stability 2: Hamiltonian equations

Look at a simple second order ODE again: $y''(x) = f(x, y)$

Stability 2: Hamiltonian equations

Look at a simple second order ODE again: $y''(x) = f(x, y)$

Make it into two first order equations as usual

$$\frac{dy}{dx} = z(x) \qquad \frac{dz}{dx} = f(x, y)$$

The Euler method can be written

1. $z_{n+1} = z_n + \varepsilon f(x_n, y_n)$
2. $y_{n+1} = y_n + \varepsilon z_n$

Stability 2: Hamiltonian equations

Look at a simple second order ODE again: $y''(x) = f(x, y)$

Make it into two first order equations as usual

$$\frac{dy}{dx} = z(x) \qquad \frac{dz}{dx} = f(x, y)$$

The Euler method can be written

1. $z_{n+1} = z_n + \varepsilon f(x_n, y_n)$
2. $y_{n+1} = y_n + \varepsilon z_n$

We have already calculated z_{n+1} by the time we update y.
Why not use it instead?

Stability 2: Hamiltonian equations

Look at a simple second order ODE again: $y''(x) = f(x, y)$

Make it into two first order equations as usual

$$\frac{dy}{dx} = z(x) \qquad \frac{dz}{dx} = f(x, y)$$

The Euler method can be written

1. $z_{n+1} = z_n + \varepsilon f(x_n, y_n)$
2. $y_{n+1} = y_n + \varepsilon z_n$

We have already calculated z_{n+1} by the time we update y.
Why not use it instead?

Euler–Cromer

1. $z_{n+1} = z_n + \varepsilon f(x_n, y_n)$
2. $y_{n+1} = y_n + \varepsilon z_{n+1}$

Stability 2: Hamiltonian equations

Look at a simple second order ODE again: $y''(x) = f(x, y)$

Make it into two first order equations as usual

$$\frac{dy}{dx} = z(x) \qquad \frac{dz}{dx} = f(x, y)$$

The Euler method can be written

1. $z_{n+1} = z_n + \varepsilon f(x_n, y_n)$
2. $y_{n+1} = y_n + \varepsilon z_n$

We have already calculated z_{n+1} by the time we update y.
Why not use it instead?

Euler–Cromer

1. $z_{n+1} = z_n + \varepsilon f(x_n, y_n)$
2. $y_{n+1} = y_n + \varepsilon z_{n+1}$

Alternatively: update y first, then z

Euler vs Euler–Cromer

Assertion: Euler–Cromer is better than Euler and midpoint. Why?

Euler vs Euler–Cromer

Assertion: Euler–Cromer is better than Euler and midpoint. Why?

The error is $\mathcal{O}(\varepsilon)$, the same as Euler, and worse than midpoint.
That cannot be the answer.

Answer

Euler–Cromer is stable.
Euler and midpoint are not!

Errors in Euler–Cromer do not grow exponentially!

Euler vs Euler–Cromer

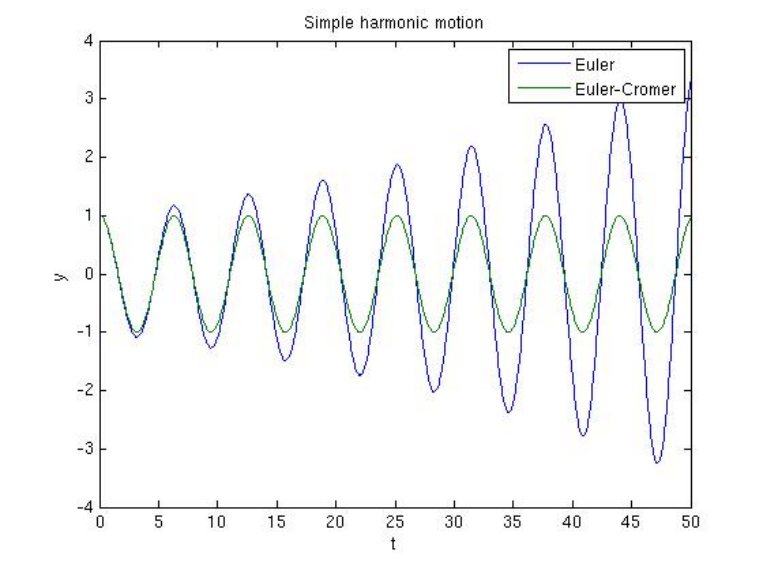
Assertion: Euler–Cromer is better than Euler and midpoint. Why?

The error is $\mathcal{O}(\varepsilon)$, the same as Euler, and worse than midpoint.
That cannot be the answer.

Answer

Euler–Cromer is stable.
Euler and midpoint are not!

Errors in Euler–Cromer do not grow exponentially!



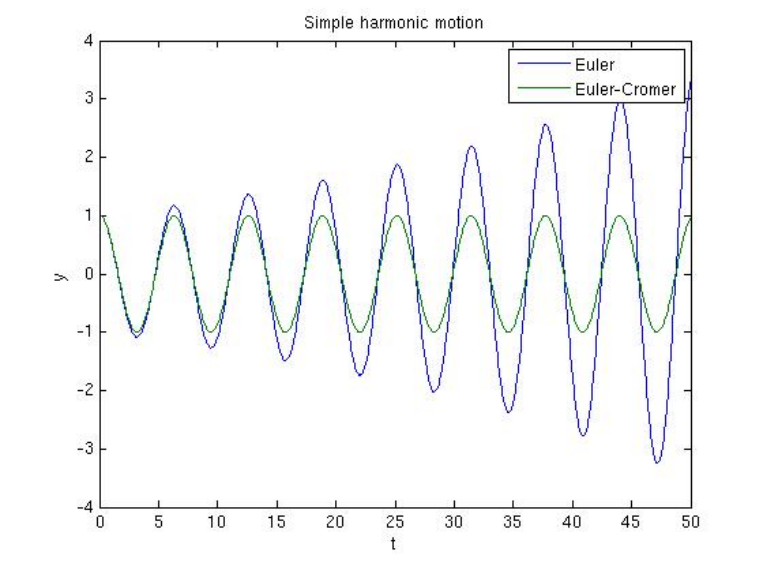
Euler vs Euler–Cromer

Assertion: Euler–Cromer is better than Euler and midpoint. Why?

The error is $\mathcal{O}(\varepsilon)$, the same as Euler, and worse than midpoint.
That cannot be the answer.

Answer

Euler–Cromer is stable.
Euler and midpoint are not!



Errors in Euler–Cromer do not grow exponentially!

Euler–Cromer is stable because it is a symplectic integrator.
It conserves the 'Hamiltonian'

Harmonic oscillator

$$\ddot{q} + \omega^2 q = 0 \qquad H = \frac{1}{2} m \dot{q}^2 + \frac{1}{2} m \omega^2 q^2$$

Set $\omega = 1, m = 2$ and set $\dot{q} = v$

Let us calculate how the Hamiltonian evolves with Euler and Euler–Cromer

Harmonic oscillator

$$\ddot{q} + \omega^2 q = 0 \qquad H = \frac{1}{2} m \dot{q}^2 + \frac{1}{2} m \omega^2 q^2$$

Set $\omega = 1, m = 2$ and set $\dot{q} = v$

Let us calculate how the Hamiltonian evolves with Euler and Euler–Cromer

Euler

$$\begin{aligned} v_{n+1} &= v_n - \varepsilon q_n, \qquad q_{n+1} = q_n + \varepsilon v_n \\ H_{n+1} &= v_{n+1}^2 + q_{n+1}^2 = (v_n - \varepsilon q_n)^2 + (q_n + \varepsilon v_n)^2 \\ &= v_n^2 + \varepsilon^2 q_n^2 - 2\varepsilon v_n q_n + q_n^2 + \varepsilon^2 v_n^2 + 2\varepsilon v_n q_n \\ &= (1 + \varepsilon^2) H_n > H_n \end{aligned}$$

Harmonic oscillator

$$\ddot{q} + \omega^2 q = 0 \qquad H = \frac{1}{2} m \dot{q}^2 + \frac{1}{2} m \omega^2 q^2$$

Set $\omega = 1, m = 2$ and set $\dot{q} = v$

Let us calculate how the Hamiltonian evolves with Euler and Euler–Cromer

Euler

$$\begin{aligned} v_{n+1} &= v_n - \varepsilon q_n, \qquad q_{n+1} = q_n + \varepsilon v_n \\ H_{n+1} &= v_{n+1}^2 + q_{n+1}^2 = (v_n - \varepsilon q_n)^2 + (q_n + \varepsilon v_n)^2 \\ &= v_n^2 + \varepsilon^2 q_n^2 - 2\varepsilon v_n q_n + q_n^2 + \varepsilon^2 v_n^2 + 2\varepsilon v_n q_n \\ &= (1 + \varepsilon^2) H_n > H_n \end{aligned}$$

The energy is steadily increasing!

Harmonic oscillator

$$\ddot{q} + q = 0 \qquad H = \dot{q}^2 + q^2$$

Euler

$$v_{n+1} = v_n - \varepsilon q_n, \qquad q_{n+1} = q_n + \varepsilon v_n$$

$$H_{n+1} = (1 + \varepsilon^2) H_n > H_n$$

Harmonic oscillator

$$\ddot{q} + q = 0 \qquad H = \dot{q}^2 + q^2$$

Euler

$$v_{n+1} = v_n - \varepsilon q_n, \qquad q_{n+1} = q_n + \varepsilon v_n$$
$$H_{n+1} = (1 + \varepsilon^2) H_n > H_n$$

Euler–Cromer

$$v_{n+1} = v_n - \varepsilon q_n, \qquad q_{n+1} = q_n + \varepsilon v_{n+1} = (1 - \varepsilon^2) q_n + \varepsilon v_n$$
$$\begin{aligned} H_{n+1} &= v_{n+1}^2 + q_{n+1}^2 \\ &= H_n + \varepsilon^2 (v_n^2 - q_n^2) - 2\varepsilon^3 v_n q_n + \varepsilon^4 q_n^2 \end{aligned}$$

Harmonic oscillator

$$\ddot{q} + q = 0 \qquad H = \dot{q}^2 + q^2$$

Euler

$$v_{n+1} = v_n - \varepsilon q_n, \qquad q_{n+1} = q_n + \varepsilon v_n$$
$$H_{n+1} = (1 + \varepsilon^2) H_n > H_n$$

Euler–Cromer

$$v_{n+1} = v_n - \varepsilon q_n, \qquad q_{n+1} = q_n + \varepsilon v_{n+1} = (1 - \varepsilon^2) q_n + \varepsilon v_n$$
$$\begin{aligned} H_{n+1} &= v_{n+1}^2 + q_{n+1}^2 \\ &= H_n + \varepsilon^2 (v_n^2 - q_n^2) - 2\varepsilon^3 v_n q_n + \varepsilon^4 q_n^2 \end{aligned}$$

For the exact solution, the ε^2 and ε^3 term will average to 0 over a complete oscillation.

Summary

- Euler method: simple iterative procedure, accurate to $\mathcal{O}(\varepsilon)$
- Higher-order ODEs can be reduced to coupled first-order ODEs
- Instability in Euler method:
  - 1st order ODE: too large stepsize $\longrightarrow$ runaway solutions
  - Hamiltonian systems: H increases with every step
  - Euler–Cromer cures instability in hamiltonian systems