

Lecture 15 - C

§7.1, 7.2, 7.3 Runge-Kutta Methods

T. Takahashi

Department of Mathematics and Statistics
University of New Mexico



We will answer these questions:

- What is the main idea behind the Runge-Kutta methods?
- How are the RK2 methods derived?
- How is the RK4 method derived?
- How do the RK2 and RK4 methods compare to Euler's method in terms of accuracy?
- How do we choose the step size for Runge-Kutta methods?



Runge-Kutta Methods



Fundamental Theorem

- The goal is to solve the ODE

$$u' = f(t, u)$$

- We can use the fundamental theorem

$$\int_{t_i}^{t_{i+1}} u' dt = \int_{t_i}^{t_{i+1}} f(t, u) dt$$

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

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

Heart of Runge-Kutta Methods

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

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

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



Euler's Method

- Consider the simple Riemann approximation

$$\int_a^b f(x) dx \approx (b - a)f(a)$$

- Apply this approximation

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

$$u(t_{i+1}) \approx u(t_i) + (t_{i+1} - t_i)f(t_i, u(t_i))$$

$$u_{i+1} \approx u_i + h f(t_i, u_i)$$

- The result is *Euler's method*

Key Idea

Euler's method can be thought of as a low-order Runge-Kutta method.



Question

What happens when you apply better quadratures to

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

- Trapezoid Rule

$$\int_a^b f(x) dx \approx (b-a) \left(\frac{f(a) + f(b)}{2} \right)$$

- Midpoint Rule

$$\int_a^b f(x) dx \approx (b-a) f\left(\frac{a+b}{2}\right)$$

- Simpson's Rule

$$\int_a^b f(x) dx \approx \frac{b-a}{6} \left(f(a) + f\left(\frac{a+b}{2}\right) + f(b) \right)$$



RK2a



Applying Trapezoidal Rule

Goal

Derive a time-stepping method by applying the **Trapezoidal Rule**

$$\int_a^b f(x) dx \approx (b - a) \left(\frac{f(a) + f(b)}{2} \right)$$

in the formula

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

- The approximate integral is

$$\int_{t_i}^{t_{i+1}} f(t, u(t)) dt \approx (t_{i+1} - t_i) \left(\frac{f(t_i, u(t_i)) + f(t_{i+1}, u(t_{i+1}))}{2} \right)$$



Runge-Kutta with Trapezoid

RK2 - Version a

- Let the approximation $u_i \approx u(t_i)$ and let $h = t_{i+1} - t_i$

$$\begin{aligned}u(t_{i+1}) &= u(t_i) + \int_{t_i}^{t_{i+1}} f(t, u(t)) dt \\&\approx u(t_i) + (t_{i+1} - t_i) \left(\frac{f(t_i, u(t_i)) + f(t_{i+1}, u(t_{i+1}))}{2} \right) \\u_{i+1} &\approx u_i + h \left(\frac{f(t_i, u_i) + f(t_{i+1}, u_{i+1})}{2} \right)\end{aligned}$$

- But this is **not** usable

$$\mathbf{u}_{i+1} \approx \mathbf{u}_i + \frac{h}{2} [f(t_i, \mathbf{u}_i) + f(t_i + h, \mathbf{u}_{i+1})]$$

- This is not in time-stepping form

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

because of the $\mathbf{u}_{i+1}$



Runge-Kutta with Trapezoid

RK2 - Version a

- **Problem:** We need a way to approximate $\mathbf{u}_{i+1}$ on the right-hand side

$$\mathbf{u}_{i+1} \approx \mathbf{u}_i + \frac{h}{2}[f(t_i, \mathbf{u}_i) + f(t_i + h, \mathbf{u}_{i+1})]$$

- **Fix:** Use Euler's method on $\mathbf{u}_{i+1}$

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

- Finally, we have a usable time-stepping algorithm

$$\mathbf{u}_{i+1} \approx \mathbf{u}_i + \frac{h}{2}[f(t_i, \mathbf{u}_i) + f(t_i + h, \mathbf{u}_{i+1})]$$

↓

$$\mathbf{u}_{i+1} \approx \mathbf{u}_i + \frac{h}{2}[f(t_i, \mathbf{u}_i) + f(t_i + h, \mathbf{u}_i + hf(t_i, \mathbf{u}_i))]$$



Runge-Kutta with Trapezoid

RK2 - Version a

- This algorithm is often expressed in two "stages"

$$\mathbf{u}_{i+1} \approx \mathbf{u}_i + \frac{h}{2} \underbrace{f(t_i, \mathbf{u}_i)}_{\text{stage 1}} + \underbrace{f(t_i + h, \mathbf{u}_i + hf(t_i, \mathbf{u}_i))}_{\text{stage 2}}$$

RK2a (a.k.a. Heun's method)

$$K_1 = f(t_i, \mathbf{u}_i)$$

$$K_2 = f(t_i + h, \mathbf{u}_i + hK_1)$$

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \frac{h}{2}(K_1 + K_2)$$

- This is version A because there is another two-stage method.



Applying Midpoint Rule

Goal

Derive a time-stepping method by applying the **Midpoint Rule**

$$\int_a^b f(x) dx \approx (b-a)f\left(\frac{a+b}{2}\right)$$

in the formula

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

- The approximate integral is

$$\int_{t_i}^{t_{i+1}} f(t, u(t)) dt \approx (t_{i+1} - t_i) f\left(t_i + \frac{h}{2}, u\left(t_i + \frac{h}{2}\right)\right)$$



Runge-Kutta with Midpoint Rule

RK2 - Version b

- Let's approximate the integral with **midpoint rule**

$$u(t_{i+1}) = u(t_i) + \underbrace{\int_{t_i}^{t_{i+1}} f(t, u(t)) dt}_{\text{approximate}} \approx u(t_i) + (t_{i+1} - t_i) f\left(t_i + \frac{h}{2}, u\left(t_i + \frac{h}{2}\right)\right)$$

- Let $\mathbf{u}_i \approx u(t_i)$ and $h = t_{i+1} - t_i$

$$\mathbf{u}_{i+1} - \mathbf{u}_i \approx h[f(t_i + h/2, \mathbf{u}_{i+1/2})]$$

- We can approximate $\mathbf{u}_{i+1/2}$ with half step of Euler's method

$$\mathbf{u}_{i+1/2} = \mathbf{u}_i + (h/2)f(t_i, \mathbf{u}_i)$$

$$\mathbf{u}_{i+1} = \mathbf{u}_i + h[f(t_i + h/2, \mathbf{u}_i + (h/2)f(t_i, \mathbf{u}_i))]$$



Runge-Kutta with Midpoint Rule

RK2 - Version b

- We get a different two-stage method

$$\mathbf{u}_{i+1} \approx \mathbf{u}_i + h \underbrace{\left[f\left(t_i + h/2, \mathbf{u}_i + \underbrace{(h/2) f(t_i, \mathbf{u}_i)}_{\text{stage 1}}\right) \right]}_{\text{stage 2}}$$

RK2b (a.k.a. Midpoint Method)

$$K_1 = f(t_i, \mathbf{u}_i)$$

$$K_2 = f(t_i + h/2, \mathbf{u}_i + (h/2) K_1)$$

$$\mathbf{u}_{i+1} = \mathbf{u}_i + hK_2$$



Two-Stage Runge-Kutta Methods

RK2a (a.k.a. Heun's method)

$$K_1 = f(t_i, \mathbf{u}_i)$$

$$K_2 = f(t_i + h, \mathbf{u}_i + hK_1)$$

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \frac{h}{2}(K_1 + K_2)$$

RK2b (a.k.a. midpoint method)

$$K_1 = f(t_i, \mathbf{u}_i)$$

$$K_2 = f(t_i + h/2, \mathbf{u}_i + (h/2) K_1)$$

$$\mathbf{u}_{i+1} = \mathbf{u}_i + hK_2$$



RK2 Methods

- Both RK2a and RK2b have
 - *local accuracy* $\mathcal{O}(h^3)$
 - *global accuracy* $\mathcal{O}(h^2)$
- They have accuracy that is one order better than Euler's method

Method: RK2

- **Given:** First order system $\mathbf{u}' = \mathbf{F}(t, \mathbf{u})$, $\mathbf{u}(0) = \mathbf{u}_0$, step size h
- **Output:** Discrete approximation to $\mathbf{u}$
- **Idea:** They have a higher accuracy than Euler's because a better quadrature is used in the derivation.

Key Idea

Better quadrature $\implies$ better time-stepping method



RK4



Simpson's Rule

- Simpson's Rule

$$\int_a^b f(x) dx \approx \frac{b-a}{6} \left(f(a) + f\left(\frac{a+b}{2}\right) + f(b) \right)$$

- Applying it to $\int_{t_i}^{t_{i+1}} f(t, u(t))$ yields

$$\frac{t_{i+1} - t_i}{6} \left[f(t_i, u_i) + f\left(t_i + \frac{h}{2}, u\left(t_i + \frac{h}{2}\right)\right) + f(t_i + h, u(t_i + h)) \right]$$

- Utilize slope of $u(t)$ at three locations

- t_i
- $t_i + h/2$
- t_{i+1}



Four-Stage Runge-Kutta Method

- Simpson's quadrature is broken into four stages

RK4

$$K_1 = f(t_i, \mathbf{u}_i)$$

$$K_2 = f(t_i + h/2, \mathbf{u}_i + (h/2) K_1)$$

$$K_3 = f(t_i + h/2, \mathbf{u}_i + (h/2) K_2)$$

$$K_4 = f(t_i + h, \mathbf{u}_i + hK_3)$$

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

- Global accuracy is $O(h^4)$



Accuracy

- Euler: $\mathcal{O}(h)$
- RK2: $\mathcal{O}(h^2)$
- RK4: $\mathcal{O}(h^4)$

Key Idea

RK4 achieves high order of accuracy with respect to time step h .



Method: Runge-Kutta 4

- **Given:** First order system $\mathbf{u}' = \mathbf{F}(t, \mathbf{u})$, $\mathbf{u}(0) = \mathbf{u}_0$, step size h
 - **Output:** Discrete approximation to $\mathbf{u}$
 - **Idea:** It is derived from Simpson's rule which has accuracy $\mathcal{O}(h^4)$
-
- Most popular time-stepping algorithm
 - Matlab's `ode45()` is a professional implementation of RK4 with *adaptive step size*
 - *Fourth order accurate!*



Using ode45

Problem

$$\begin{cases} u' = -2u, & t \in (0, 5] \\ u(0) = 1 \end{cases}$$

```
1 f = @(t,u) -2*u;  
2 [t,u] = ode45(f, [0 5], 1);  
3 plot(t,u)
```

- First, create a function handle $f=@(t,u)$ for the RHS of the ODE
- Call `ode45` with a specified time interval and IC
- The function chooses the step size h dynamically



Stability



How to choose h ?

- Suppose you were tasked with solving an important problem

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

- We pick the proper time scale for the problem:
 - Weather: days or weeks
 - Aerodynamics: seconds
 - Chemical reactions: microseconds

Important Questions

- Can I trust that $\mathbf{u}$ is correct for important applications?
- Will my $\mathbf{u}$ retain key dynamics of the differential equation?
- Does $\mathbf{u}$ become less accurate in the long term?



What is Stability?

Definition: Stability

A numerical method is **stable** if small errors introduced at any step do not grow uncontrollably as computations proceed.

Key Features of Stable time-stepping methods:

- Does not propagate error
- Does not diverge from the true solution for sufficiently small h
- Does not fail for test problem $u' = \lambda u$



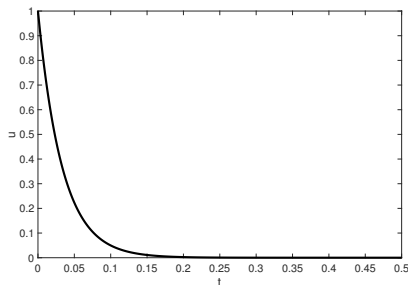
Test Problem for Time-stepping Methods

- Consider this initial value problem

Test Problem

$$\begin{aligned}u'(t) &= \lambda u(t), & \lambda \in \mathbb{C}, \quad t \in [0, T] \\ u(0) &= u_0\end{aligned}$$

- Example: $u_0 = 1$, $\lambda = -30$, $T = 0.5$
- Exact solution is: $u(t) = e^{-30t}$

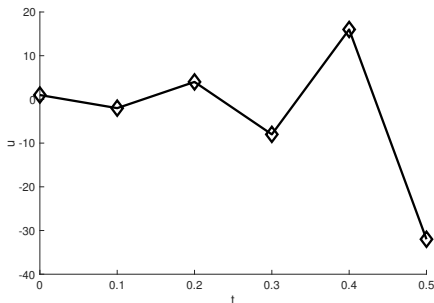


Euler's method approximation

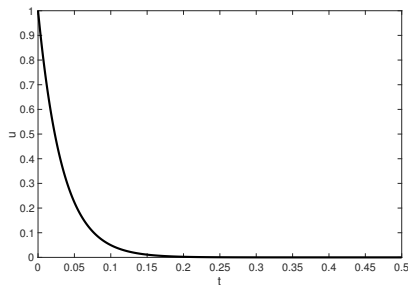
- Euler's method for the model problem:

$$u_{i+1} = u_i + h(\lambda u_i) = (1 + h\lambda)u_i, \quad i = 0, 1, 2, \dots$$

- Solution for choose $\lambda = -30, u_0 = 1, h = 0.1$



Euler solution

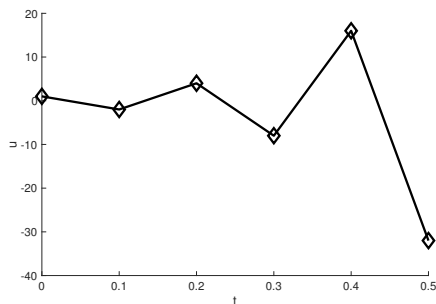


True solution

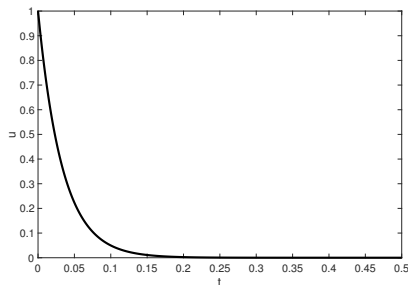
- This is **numerical instability**



What went wrong?



Euler solution



True solution

What went wrong?

- Didn't choose h small enough
- Error propagated
- Numerical solution is growing (!?)



Question

Why did the numerical solution grow?

- Euler's method is $u_{i+1} = u_i + hf(u_i)$ for $u' = \lambda u$ yields

$$\begin{aligned}u_{i+1} &= u_i + h\lambda u_i \\&= u_i(1 + h\lambda) \\ \implies |u_{i+1}| &= |1 + h\lambda||u_i|\end{aligned}$$

- Or more intuitively,

$$\underbrace{|u_{i+1}|}_{\text{new}} = \underbrace{|1 + h\lambda|}_{\text{amplification factor}} \underbrace{|u_i|}_{\text{old}}$$

- For stability we need

$$|1 + h\lambda| < 1$$



Stability Region

- Suppose $u_{i+1} = u_i + T(u_i)$ is a time-stepping algorithm
- We define our **test problem** to be

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

Definition: Stability Region

The **region of absolute stability** is defined to be a region for λh such that the time stepping algorithm produces a stable solution for any u_0 .

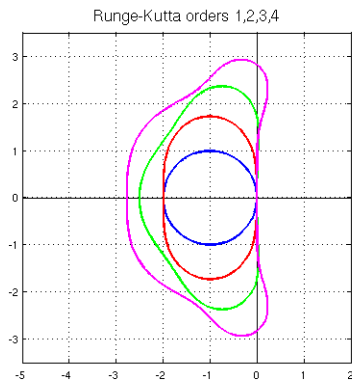
Key Idea

Stability region is an innate property of time-stepping methods.

Bigger region $\implies$ Better method



Visualizing Stability Regions



- Stability regions are visualized in the complex plane
- The small blue circle is the stability region for Euler's method
- The largest magenta region is for RK4



Choosing a stable h

Example

What value of h would give a stable solution for Euler's method for the IVP

$$\begin{aligned}u'(t) &= -30u(t), & t \in [0, 0.5] \\ u(0) &= 1\end{aligned}$$

- Euler's method yields

$$\begin{aligned}u_{i+1} &= u_i - 30hu_i \\ &= u_i(1 - 30h) \\ \implies |u_{i+1}| &= |1 - 30h||u_i|\end{aligned}$$

- For stability we need $|1 - 30h| < 1$ or

$$\begin{aligned}-1 &< 1 - 30h < 1 \\ -2 &< -30h < 0 \\ 1/15 &> h > 0\end{aligned}$$

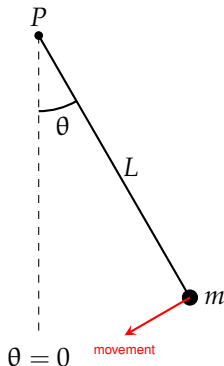
- We need $h < 1/15$



The Pendulum



Physical Setup



- Pendulum of length L is attached to a bob of mass m
- Pendulum may pivot about point P , i.e. it can oscillate
- Motion is described by the angular displacement $\theta(t)$
- The neutral hanging position is $\theta = 0$

Objective

Variables:

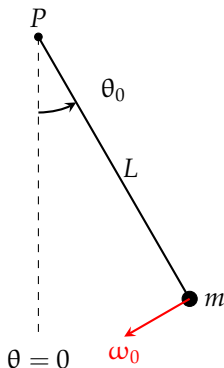
- t = time
- $\theta(t)$ = angular displacement of the pendulum at time t
- L = length of the rod
- m = mass of the weight
- $\omega = \theta' =$ angular speed

Goal

Derive a differential equation describing the motion of the pendulum using the variables above.



Initial Conditions



- Initial position $\theta(0) = \theta_0$
- Initial angular velocity $\theta'(0) = \omega_0$

We pull the bob to the right, and then give it a tap in the left direction

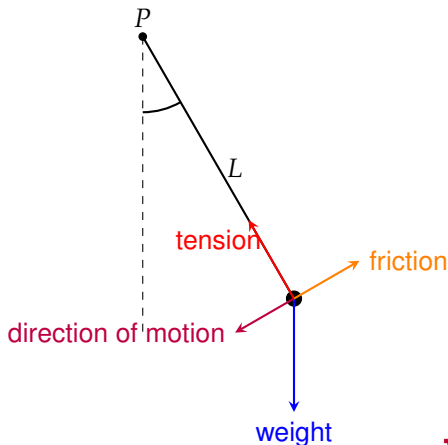


Free Body Diagram

Three forces

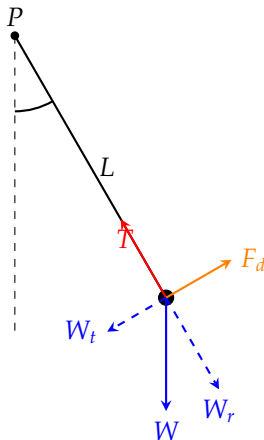
- Tension
- Friction
- Weight

Since the bob is attached to a rigid rod, the forces cancel out to point in the *tangential direction*



Tangential vs. Radial

- The weight force W can be expressed as
 $W = W_t + W_r$
- Tension T is canceled out by W_r
- Tangential component of the weight
 $W_t = -mg \sin \theta$
- Damping force
 $F_d = -c \theta', c \geq 0$



Derivation

Step 1. Coordinates and Kinematics

Take two derivatives of the Arc length formula $s = L\theta$:

$$s'' = L\theta'' = a.$$

Step 2. Newton's Law

Newton's Second Law says the tangential forces satisfy

$$\sum F = ma = mL\theta''$$

Step 3. Sum of Forces The two forces are tangential component of weight and friction

$$\sum F = W_t + F_d = -mg \sin \theta - c \theta'$$

Step 4. Putting it all together

$$-mg \sin \theta - c \theta' = mL \theta''$$

$$mL \theta'' + c \theta' + mg \sin \theta = 0$$



Types of Initial Conditions

Equation for the Pendulum

$$mL \theta'' + c \theta' + mg \sin \theta = 0$$

we must specify two initial conditions.

Common choices of initial conditions:

- **Released from rest:**

$$\theta(0) = \theta_0, \quad \theta'(0) = 0.$$

- **Given an initial push:**

$$\theta(0) = \theta_0, \quad \theta'(0) = \omega_0.$$

- **Starting at the bottom with a kick:**

$$\theta(0) = 0, \quad \theta'(0) = \omega_0.$$

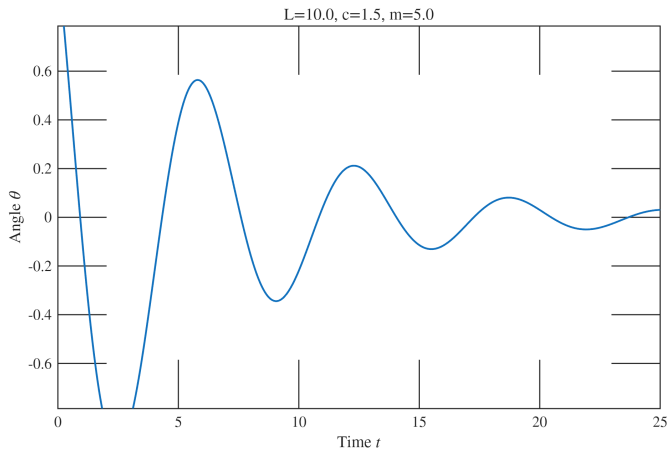
- **Starting near the top (unstable):**

$$\theta(0) \approx \pi, \quad \theta'(0) \approx 0.$$



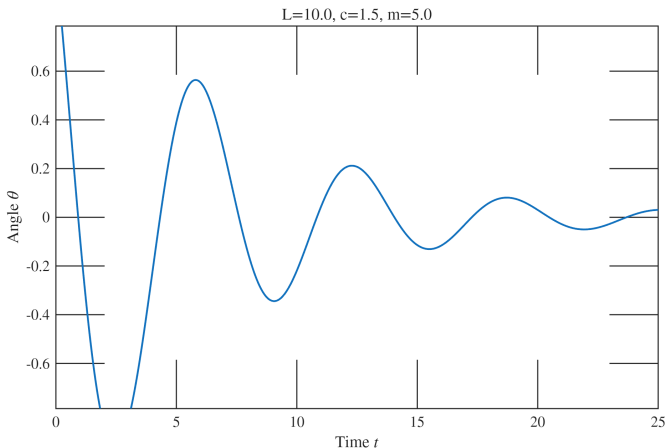
Visualizing the Solution

- We can solve $mL\theta'' + c\theta' - mg\sin\theta = 0$ with Euler's method and plot



Visualizing the Solution

- We can solve $mL\theta'' + c\theta' - mg\sin\theta = 0$ with Euler's method and plot



- But that's boring



Making Animations

- An **animation** is made by plotting the pendulum in increments of time Δt and writing it as a frame of a video
- First, solve $mL \theta'' + c \theta' + mg \sin \theta = 0$
- Plot pendulum at $t = 0$, save it.
- Plot pendulum at $t = t_1$, save it.
- Keep going up to $t = t_n$.

```
1 [t,u] = solve_pendulum()
2 for i = 1 : n
3     plot(state of pendulum at time  $t_i$ )
4     imwrite(plot_to_gif)
5 end
```

