NO CLASS NEXT WEEK
April 6th
Primer on Differential Equations

deterministic differential equations first
stochastic differential equations after
Basic Calculus

Derivatives

Integrals
Derivatives

\[
\frac{df(x)}{dx}\bigg|_{x=a} = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}
\]

\(h = 4\)
Derivatives

\[ \frac{df(x)}{dx} \bigg|_{x=a} = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h} \]
Derivatives

\[
\frac{df(x)}{dx} \bigg|_{x=a} = \lim_{h \to 0} \frac{f(a + h) - f(a)}{h}
\]
Derivatives

\[
\frac{df(x)}{dx} \bigg|_{x=a} = \lim_{h \to 0} \frac{f(a + h) - f(a)}{h}
\]

\( h = 1 \)
Derivatives

\[
\frac{df(x)}{dx} \bigg|_{x=a} = \lim_{h \to 0} \frac{f(a + h) - f(a)}{h}
\]
Integrals

\[ f(x) \]
Integrals
Simple differential equations

\[ \frac{dx(t)}{dt} = I \]

where \( I \) is a constant

REMEMBER: the derivative of a function is another function that specifies the rate of change of that function with respect to one (or more) of it’s variables.
Simple differential equations

\[
\frac{dx(t)}{dt} = I \quad \text{where } I \text{ is a constant}
\]

REMEMBER: the derivative of a function is another function that specifies the rate of change of that function with respect to one (or more) of its variables.

In beginning Calculus, you’re given a function and asked to compute its derivative.

Here, you’re just working backwards, you’re given the derivative and asked to find out what function it came from.
Simple differential equations

\[ \frac{dx(t)}{dt} = I \]

where \( I \) is a constant

there is some unknown function \( x(t) \) and all we know
is that the slope of \( x(t) \), which is \( dx/dt \), is equal to a constant
everywhere where the function is defined

without knowing any Calculus, you know that this has to be the equation for a line with slope \( I \), but you don’t have enough information to know the intercept.
\[ \frac{d x(t)}{d t} = I \]
for every value of $t$ and $x(t)$, the slope of $x(t)$ is equal to $I$
for every value of $t$ and $x(t)$, the slope of $x(t)$ is equal to $I$
Simple differential equations

\[
\frac{dx}{dt} = I \quad \text{where } I \text{ is a constant}
\]
Simple differential equations

\[
\frac{dx}{dt} = I \\
\frac{dx(t)}{dt} = I
\]

where \( I \) is a constant

the “unknown” is a function \( x(t) \)
Simple differential equations

\[ \frac{dx}{dt} = I \quad \text{where } I \text{ is a constant} \]

\[ \frac{dx(t)}{dt} = I \quad \text{the “unknown” is a function } x(t) \]

\[ dx = I \, dt \]
Simple differential equations

\[ \frac{dx}{dt} = I \]

where \( I \) is a constant

\[ \frac{dx(t)}{dt} = I \]

the “unknown” is a function \( x(t) \)

\[ dx = I \, dt \]

an integral is often characterized as an “antiderivative”

\[ \int dx = \int I \, dt \]
Simple differential equations

\[
\frac{dx}{dt} = I
\]

where \( I \) is a constant

\[
\frac{dx(t)}{dt} = I
\]

the “unknown” is a function \( x(t) \)

\[
dx = I \, dt
\]

an integral is often characterized as an “antiderivative”

\[
\int dx = \int I \, dt
\]

\[
x = It + C
\]
Simple differential equations

\[ \frac{dx}{dt} = I \]

where \( I \) is a constant

\[ \frac{dx(t)}{dt} = I \]

the “unknown” is a function \( x(t) \)

\[ dx = I \, dt \]

\[ \int dx = \int I \, dt \]

an integral is often characterized as an “antiderivative”

\[ x = It + C \]

\[ x(t) = It + C \]

where \( C \) is a constant
Simple differential equations

\[ x(t) = It + C \]

how do you figure out \( C \)?
Simple differential equations

\[ x(t) = It + C \]

how do you figure out \( C \)?

\[ x(0) = 2 \]

initial conditions
Simple differential equations

\[ x(t) = It + C \]  \hspace{1cm} \text{how do you figure out } C? \\
\[ x(0) = 2 \]  \hspace{1cm} \text{initial conditions} \\
\[ x(0) = I \times 0 + C = 2 \]
Simple differential equations

\[ x(t) = It + C \quad \text{how do you figure out } C? \]
\[ x(0) = 2 \quad \text{initial conditions} \]

\[ x(0) = I \times 0 + C = 2 \]
\[ C = 2 \]
Simple differential equations

\[ x(t) = It + C \]  
how do you figure out \( C \)?

\[ x(0) = 2 \]  
initial conditions

\[ x(0) = I \times 0 + C = 2 \]

\[ C = 2 \]

\[ x(t) = It + 2 \]  
solution
for every value of $t$, the slope of $x(t)$ is equal to $I$
Simple differential equations

\[ \frac{dx}{dt} = -\frac{1}{\tau} x \]

\( \tau \) is just a number (a parameter)

\( x \) is a function, \( x(t) \)
Simple differential equations

\[ \frac{dx}{dt} = -\frac{1}{\tau} x \]

\( \tau \) is just a number (a parameter)

\( x \) is a function, \( x(t) \)

we’re looking for a function \( x(t) \) whose slope at each point in time \( t \) is a function of its current value \( x \)
for every value of $t$, the slope of $x(t)$ is a function of $x(t)$
for every value of $t$, the slope of $x(t)$ is a function of $x(t)$.
Simple differential equations

\[ \frac{dx}{dt} = -\frac{1}{\tau} x \]

\( \tau \) is just a number (a parameter)

\( x \) is a function, \( x(t) \)

we’re looking for a function \( x(t) \) whose slope at each point in time \( t \) is a function of its current value \( x \)

you might suspect \( x(t) \) is somehow related to an exponential with base \( e \) if you recall that

\[ \frac{de^t}{dt} = e^t \]
Simple differential equations

\[ \frac{dx}{dt} = -\frac{1}{\tau} x \]

\( \tau \) is the time constant
Simple differential equations

\[ \frac{dx}{dt} = -\frac{1}{\tau} x \]

try \( x(t) = A \exp(bt) \)

\( \tau \) is the time constant
Simple differential equations

\[ \frac{dx}{dt} = -\frac{1}{\tau} x \]

\[ \text{try } x(t) = A \exp(bt) \]

\[ \frac{dA \exp(bt)}{dt} = -\frac{1}{\tau} A \exp(bt) \]

\[ Ab \exp(bt) = -\frac{1}{\tau} A \exp(bt) \]

\[ b = -\frac{1}{\tau} \]

\( \tau \) is the time constant

one method of solving
Simple differential equations

\[ x(t) = A \exp(-t/\tau) \]  
what is \( A \)?
Simple differential equations

\[ x(t) = A \exp(-t/\tau) \]

what is \( A \)?

\[ x(0) = x_0 \]

initial conditions
Simple differential equations

\[ x(t) = A \exp(-t / \tau) \]  \hspace{1cm} \text{what is } A?

\[ x(0) = x_0 \]  \hspace{1cm} \text{initial conditions}

\[ x_0 = A \exp(-0 / \tau) \]

\[ A = x_0 \]

\[ x(t) = x_0 \exp(-t / \tau) \]  \hspace{1cm} \text{solution}
Plotting the solution

\[ x(t) = x_0 \exp(-t / \tau) \]

\[ x_0 = 1 \]
Simple, commonly used, model of a neural integrator

\[ \frac{dx}{dt} = \frac{1}{\tau} (I - kx) \]

- \( k \) is a constant (leakage),
- \( I \) is the input
- \( x(t) \) is the firing rate at time \( t \)
- \( I \) is its input
- inhibits itself by a factor \( kx \)

The same equation is used in integrate-and-fire model of neurons, in which case \( I \) is the synaptic input and \( x(t) \) is the voltage at time \( t \).
Simple, commonly used, model of a neural integrator

\[
\frac{dx}{dt} = \frac{1}{\tau} (I - kx)
\]

- \(k\) is a constant (leakage),
- \(I\) is the input
- \(x(t)\) is the firing rate at time \(t\)
- \(I\) is its input
- inhibits itself by a factor \(kx\)

where does this (or any) differential equation come from?

in essence the differential equation is the mathematical instantiation of the theory of the process under investigation

generating theory from a core understanding of the behavioral, biological, chemical, physical processes
Simple, commonly used, model of a neural integrator

\[
\frac{dx}{dt} = \frac{1}{\tau} (I - kx)
\]

- \( k \) is a constant (leakage), \( I \) is the input
- \( x(t) \) is the firing rate at time \( t \)
- \( I \) is its input
- inhibits itself by a factor \( kx \)

the model is specified as a differential equation because

1) we understand the physical or biological processes giving rise to change; physical, electrical, chemical “forces” cause things to change
2) compared to the equation for \( x(t) \), the differential equation is general, for any possible \( I \) ... you want theories to generalize
3) the differential equation explains \textit{WHY} the neurons fires as \( x(t) \)
Simple, commonly used, model of a neural integrator

\[
\frac{dx}{dt} = \frac{1}{\tau}(I - kx) \quad k \text{ is a constant (leakage)}
\]

\[I \text{ is the input}\]

\[x(0) = 0 \quad k = 1\]

\[x(t) = I(1 - \exp(-t / \tau))\]
Simple, commonly used, model of a neural integrator

\[
\frac{dx}{dt} = \frac{1}{\tau} (I - kx)
\]

\[
x(t) = I (1 - \exp(-t/\tau))
\]
Simple, commonly used, model of a neural integrator

\[
\frac{dx}{dt} = \frac{1}{\tau} (I(t) - kx) \quad \text{input is a function } I(t)
\]
Simple, commonly used, model of a neural integrator

\[
\frac{dx}{dt} = \frac{1}{\tau} (I(t) - kx) \quad \text{input is a function } I(t)
\]

\[x(0) = x_0 \quad k = 1\]

\[x(t) = x_0 \exp(-t / \tau) + \frac{1}{\tau} \int_0^t \exp\left(-\frac{(t - t')}{\tau}\right) I(t') dt'\]

convolution
Numerical vs. simulation methods for solving differential equations

analytic solutions to differential equations are preferred
- they’re exact, not approximations
- they’re often far faster to calculate than any simulation
- but they’re hard, and sometimes impossible to produce
analytic solutions to differential equations are preferred
- they’re exact, not approximations
- they’re often far faster to calculate than any simulation
- but they’re hard, and sometimes impossible to produce

some advantages of numerical and simulation approaches
- easier, faster turnaround, especially with fast computers
Analytic solutions to differential equations are preferred
- they’re exact, not approximations
- they’re often far faster to calculate than any simulation
- but they’re hard, and sometimes impossible to produce

Some advantages of numerical and simulation approaches
- easier, faster turnaround, especially with fast computers
- let’s you easily play “what if” games with variations of a model
Numerical vs. simulation methods for solving differential equations

analytic solutions to differential equations are preferred
- they’re exact, not approximations
- they’re often far faster to calculate than any simulation
- but they’re hard, and sometimes impossible to produce

some advantages of numerical and simulation approaches
- easier, faster turnaround, especially with fast computers
- let’s you easily play “what if” games with variations of a model
- permit solutions to nonlinear and stochastic equations
  impossible to solve analytically
Numerical vs. simulation methods for solving differential equations

analytic solutions to differential equations are preferred
- they’re exact, not approximations
- they’re often far faster to calculate than any simulation
- but they’re hard, and sometimes impossible to produce

some advantages of numerical and simulation approaches
- easier, faster turnaround, especially with fast computers
- let’s you easily play “what if” games with variations of a model
- permit solutions to nonlinear and stochastic equations impossible to solve analytically
- help “democratize” model development and testing for scientists with insufficient mathematical expertise
Numerical vs. simulation methods for solving differential equations

Analytic solutions to differential equations are preferred
- they’re exact, not approximations
- they’re often far faster to calculate than any simulation
- but they’re hard, and sometimes impossible to produce

Some advantages of numerical and simulation approaches
- easier, faster turnaround, especially with fast computers
- let’s you easily play “what if” games with variations of a model
- permit solutions to nonlinear and stochastic equations impossible to solve analytically
- help “democratize” model development and testing for scientists with insufficient mathematical expertise
- “democracy” can be dangerous, and people need to be responsible, which means understanding the techniques
# Built-in Matlab functions for solving ODEs

<table>
<thead>
<tr>
<th>Solver</th>
<th>Problem Type</th>
<th>Order of Accuracy</th>
<th>When to Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>ode45</td>
<td>Nonstiff</td>
<td>Medium</td>
<td>Most of the time. This should be the first solver you try.</td>
</tr>
<tr>
<td>ode23</td>
<td>Nonstiff</td>
<td>Low</td>
<td>For problems with crude error tolerances or for solving moderately stiff problems.</td>
</tr>
<tr>
<td>ode113</td>
<td>Nonstiff</td>
<td>Low to high</td>
<td>For problems with stringent error tolerances or for solving computationally intensive problems.</td>
</tr>
<tr>
<td>ode15s</td>
<td>Stiff</td>
<td>Low to medium</td>
<td>If ode45 is slow because the problem is stiff.</td>
</tr>
<tr>
<td>ode23s</td>
<td>Stiff</td>
<td>Low</td>
<td>If using crude error tolerances to solve stiff systems and the mass matrix is constant.</td>
</tr>
<tr>
<td>ode23t</td>
<td>Moderately Stiff</td>
<td>Low</td>
<td>For moderately stiff problems if you need a solution without numerical damping.</td>
</tr>
<tr>
<td>ode23tb</td>
<td>Stiff</td>
<td>Low</td>
<td>If using crude error tolerances to solve stiff systems.</td>
</tr>
</tbody>
</table>
Built-in Matlab functions for solving ODEs

<table>
<thead>
<tr>
<th>Solver</th>
<th>Problem Type</th>
<th>Order of Accuracy</th>
<th>When to Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>ode45</td>
<td>Nonstiff</td>
<td>Medium</td>
<td>Most of the time. This should be the first solver you try.</td>
</tr>
<tr>
<td>ode23</td>
<td>Nonstiff</td>
<td>Low</td>
<td>For problems with crude error tolerances or for solving moderately stiff problems.</td>
</tr>
<tr>
<td>ode113</td>
<td>Nonstiff</td>
<td>Low to high</td>
<td>For problems with stringent error tolerances or for solving computationally intensive problems.</td>
</tr>
<tr>
<td>ode15s</td>
<td>Stiff</td>
<td>Low to medium</td>
<td>If ode45 is slow because the problem is stiff.</td>
</tr>
<tr>
<td>ode23s</td>
<td>Stiff</td>
<td>Low</td>
<td>If using crude error tolerances to solve stiff systems and the mass matrix is constant.</td>
</tr>
<tr>
<td>ode23t</td>
<td>Moderately Stiff</td>
<td>Low</td>
<td>For moderately stiff problems if you need a solution without numerical damping.</td>
</tr>
<tr>
<td>ode23tb</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

“stiff” comes from mass-spring systems in physics and refers to the potential instability of the dynamic system to choices of step size (which we will get to soon)
Built-in Matlab functions for solving ODEs

\[
\frac{dx}{dt} = \frac{1}{\tau} (I(t) - kx)
\]

\[I(t) = \sin(t)\]

function \(dx = \text{myfun}(t,x)\)

\[
\begin{align*}
\text{tau} &= 1; \\
\text{k} &= .5; \\
\text{dx} &= (1/\text{tau})*(\sin(t) - \text{k}\times x);
\end{align*}
\]

end

t = 0:.01:20;
x0 = -5;
[T,X] = ode45(@myfun, t, x0);
Matlab
Built-in Matlab functions for solving ODEs

ode45() implements numeric methods for simulating solutions to ordinary differential equations (ODEs)

these built-in routines work for some kinds of ODEs but may not work for other more complex forms, so these must be simulated by hand (well, by your code)
\[ \frac{dx}{dt} = \frac{1}{\tau} (I - kx) \]
\[ \frac{dx}{dt} = \frac{1}{\tau} (I - kx) \]
\[ \frac{dx}{dt} = F(x,t) \]  
make it completely generic
\[
\frac{dx}{dt} = \frac{1}{\tau} (I - kx)
\]

where \( F(x,t) \) can be any linear or nonlinear function …

\[
\frac{dx}{dt} = F(x,t)
\]

make it completely generic

\[
\frac{dx}{dt} = xt
\]

where \( F(x,t) \) can be any linear or nonlinear function …

\[
\frac{dx}{dt} = \sin(x) + \cos(t)
\]

\[
\frac{dx}{dt} = t \exp(-x / t^x)
\]
\[
\begin{align*}
\frac{dx}{dt} &= \frac{1}{\tau} (I - kx) \\
\frac{dx}{dt} &= F(x, t)
\end{align*}
\]

make it completely generic

it could also be defined by other data or by a simulation …
Euler’s Method
Simulating solutions to ordinary differential equations

Euler’s Method

\[
\frac{dx}{dt} = F(x, t)
\]
Simulating solutions to ordinary differential equations

**Euler’s Method**

\[ \frac{dx}{dt} \approx \frac{\Delta x}{\Delta t} = F(x, t) \]

\[ x(t + \Delta t) = x(t) + \Delta t F(x, t) \]
Simulating solutions to ordinary differential equations

Euler’s Method

\[
\frac{dx}{dt} \approx \frac{\Delta x}{\Delta t}
\]
Simulating solutions to ordinary differential equations

Euler’s Method

\[
\frac{dx}{dt} \approx \frac{\Delta x}{\Delta t} = F(x, t)
\]
Simulating solutions to ordinary differential equations

Euler’s Method

\[
\frac{dx}{dt} \approx \frac{\Delta x}{\Delta t} = F(x, t)
\]

\[
\Delta x = \Delta t \cdot F(x, t)
\]
Simulating solutions to ordinary differential equations

Euler’s Method

\[
\frac{dx}{dt} \approx \frac{\Delta x}{\Delta t} = F(x, t)
\]

\[
\Delta x = \Delta t \cdot F(x, t)
\]

\[
x(t + \Delta t) = x(t) + \Delta x
\]
Simulating solutions to ordinary differential equations

Euler’s Method

\[ \frac{dx}{dt} \approx \frac{\Delta x}{\Delta t} = F(x, t) \]

\[ \Delta x = \Delta t \cdot F(x, t) \]

\[ x(t + \Delta t) = x(t) + \Delta x \]

\[ x(t + \Delta t) = x(t) + \Delta t \cdot F(x, t) \]
\[
\frac{dx}{dt} = F(x, t)
\]
The diagram illustrates the relationship between $x(t)$ and $t$ with a focus on a specific interval $\Delta t$. The instantaneous rate of change of $x(t)$ with respect to $t$ is shown at $t_N$ and $t_{N+1}$, represented by $\frac{dx}{dt}|_{x_N,t_N}$. The points $x_N$ and $t_N$ are marked on the graph, indicating the position and time at which the derivative is calculated. The interval $\Delta t$ is the time difference between $t_N$ and $t_{N+1}$. The dashed line connects these points, indicating the change in $x(t)$ over $\Delta t$. The graph shows the progression of $x(t)$ from $t_N$ to $t_{N+1}$.
\[ \frac{dx}{dt} \bigg|_{x_N, t_N} \]

\[ \Delta t \]

\[ \Delta x \]
\[ \Delta x = \Delta t \frac{dx}{dt} \bigg|_{x_N, t_N} \]
\[ x(t) \]

\[ x_N \]

\[ x_{N+1} \]

\[ t_N \]

\[ t_{N+1} \]

\[ t_{N+2} \]

\[ \Delta t \]
\[
\frac{dx}{dt}_{x_{N+1}, t_{N+1}}
\]
\[ x(t) \]

\[ \Delta x = \Delta t \left. \frac{dx}{dt} \right|_{x_{N+1}, t_{N+1}} \]

\[ \Delta x = \Delta t \left. \frac{dx}{dt} \right|_{x_{N+1}, t_{N+1}} \]

\[ x_{N+1} \]

\[ x_N \]

\[ t_N \]

\[ t_{N+1} \]

\[ t_{N+2} \]

\[ \Delta t \]

\[ \Delta x \]
\[ x(t) \]
\[ x_{N+1} = x_N + \frac{dx}{dt}_{|x_{N+1},t_{N+1}} \Delta t \]

\[ \Delta x = \Delta t \left( \frac{dx}{dt}_{|x_{N+1},t_{N+1}} \right) \]

Graph showing the transition from \( x_N \) to \( x_{N+1} \) along the time axis \( t_N \) to \( t_{N+1} \).

Diagram with points \( x_N, x_{N+1}, x_{N+2} \) and time intervals \( t_N, t_{N+1}, t_{N+2} \).
The diagram illustrates the function $x(t)$ over time with points at $t_N$, $t_{N+1}$, and $t_{N+2}$. The values of the function at these points are $x_N$, $x_{N+1}$, and $x_{N+2}$, respectively.
Simulating solutions to ordinary differential equations

Euler’s Method

\[ \frac{dx}{dt} = \frac{1}{\tau} (I - x) \]
Simulating solutions to ordinary differential equations

**Euler’s Method**

\[
\frac{dx}{dt} = \frac{1}{\tau} (I - x)
\]

\[
x(t + \Delta t) = x(t) + \Delta t \left[ \frac{1}{\tau} (I - x(t)) \right]
\]
Simulating solutions to ordinary differential equations

Euler’s Method

\[
\frac{dx}{dt} = \frac{1}{\tau} (I - x)
\]

\[
x(t + \Delta t) = x(t) + \Delta t \left[ \frac{1}{\tau} (I - x(t)) \right]
\]

\[
x(0) = x_0
\]
Simulating solutions to ordinary differential equations

Euler’s Method

\[ \frac{dx}{dt} = \frac{1}{\tau} (I - x) \]

\[ x(t + \Delta t) = x(t) + \Delta t \left( \frac{1}{\tau} (I - x(t)) \right) \]

\[ x(0) = x_0 \]

\[ x(0 + \Delta t) = x_0 + \Delta t \left( \frac{1}{\tau} (I - x_0) \right) \]
Simulating solutions to ordinary differential equations

**Euler’s Method**

\[
\frac{dx}{dt} = \frac{1}{\tau} (I - x)
\]

\[
x(t + \Delta t) = x(t) + \Delta t \left[ \frac{1}{\tau} (I - x(t)) \right]
\]

\[
x(0) = x_0 \quad \text{this is literally an initial condition}
\]

\[
x(0 + \Delta t) = x_0 + \Delta t \left[ \frac{1}{\tau} (I - x_0) \right]
\]
Simulating solutions to ordinary differential equations

Euler’s Method

x euler = zeros(1, length(teuler));
idx = 1;
x euler(idx) = x0;
idx = idx + 1;
for i = 2:length(teuler)
    x euler(idx) = x euler(idx-1) + dt * myfun2(teuler(idx-1), x euler(idx-1));
    idx = idx + 1;
end
Simulating solutions to ordinary differential equations

Euler’s Method

how big should $\Delta t$ be?
small $\Delta t$ slow but accurate
large $\Delta t$ fast but potentially inaccurate

large $\Delta t$

small $\Delta t$
Simulating solutions to ordinary differential equations

illustrate this example with Euler’s method

$$\frac{dx}{dt} = x$$
Simulating solutions to ordinary differential equations

illustrate this example with Euler’s method

\[
\frac{dx}{dt} = x
\]

\[
x(t) = e^x
\]
Simulating solutions to ordinary differential equations

illustrate this example with Euler’s method

\[
\frac{dx}{dt} = x \\
\]

\[x(t) = e^x\]

\[x(t + \Delta t) = x(t) + \Delta t F(x,t)\]

\[x(t + \Delta t) = x(t) + \Delta t x\]
Simulating solutions to ordinary differential equations

illustrate this example with Euler’s method

Why?
Runge-Kutta
Simulating solutions to ordinary differential equations

Taylor Series Expansion

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2 x(t_N)}{dt^2} + \cdots \]
Simulating solutions to ordinary differential equations

Taylor Series Expansion

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2 x(t_N)}{dt^2} + \ldots \]

\[ \frac{dx}{dt} = F(x, t) \]

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t F(x(t_N), t_N) + \frac{(\Delta t)^2}{2} \frac{dF(x(t_N), t_N)}{dt} + \ldots \]

- Euler’s Method
- additional derivatives
- “higher order methods”
Simulating solutions to ordinary differential equations

Taylor Series Expansion

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2 x(t_N)}{dt^2} + \cdots \]

\[ \frac{dx}{dt} = F(x,t) \]

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t F(x(t_N), t_N) + \frac{(\Delta t)^2}{2} \frac{dF(x(t_N), t_N)}{dt} + \cdots \]

\[ x(t_N + \Delta t) = \sum_{i=0}^{\infty} \frac{(\Delta t)^i}{i!} \frac{dF^i(x(t_N), t_N)}{dt^i} \]
Simulating solutions to ordinary differential equations

**Taylor Series Expansion**

\[
x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2 x(t_N)}{dt^2} + \ldots
\]

evaluating \( x(t) = e^t \) at \( t=0 \)

by summing the first \( n \) terms
Simulating solutions to ordinary differential equations

Taylor Series Expansion

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2 x(t_N)}{dt^2} + \ldots \]

evaluating \( x(t) = e^t \) at \( t=0 \)

by summing the first \( n \) terms
Simulating solutions to ordinary differential equations

Taylor Series Expansion

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2 x(t_N)}{dt^2} + \cdots \]

evaluating \( x(t) = e^t \) at \( t=0 \)

by summing the first \( n \) terms
Simulating solutions to ordinary differential equations

Taylor Series Expansion

\[
x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2x(t_N)}{dt^2} + \cdots
\]

evaluating \(x(t) = e^t\) at \(t=0\) by summing the first \(n\) terms
Simulating solutions to ordinary differential equations

Taylor Series Expansion

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2x(t_N)}{dt^2} + \ldots \]

evaluating \( x(t) = e^t \) at \( t = 0 \)

by summing the first \( n \) terms
Simulating solutions to ordinary differential equations

Taylor Series Expansion

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2x(t_N)}{dt^2} + \ldots \]

evaluating \( x(t) = e^t \) at \( t = 0 \)

by summing the first \( n \) terms
Simulating solutions to ordinary differential equations

Taylor Series Expansion

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2x(t_N)}{dt^2} + \cdots \]

evaluating \( x(t) = e^t \) at \( t = 0 \)

by summing the first \( n \) terms
Simulating solutions to ordinary differential equations

Taylor Series Expansion

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2x(t_N)}{dt^2} + \cdots \]

evaluating \( x(t) = e^t \) at \( t = 0 \)
by summing the first \( n \) terms
Simulating solutions to ordinary differential equations

**Taylor Series Expansion**

\[
x(t_N + \Delta t) \approx x(t_N) + \Delta t \frac{dx(t_N)}{dt} + \frac{(\Delta t)^2}{2} \frac{d^2 x(t_N)}{dt^2} + \cdots
\]

evaluating \(x(t) = e^t\) at \(t = 0\)

by summing the first \(n\) terms
Simulating solutions to ordinary differential equations

Runge-Kutta

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t F(x(t_N), t_N) + \frac{(\Delta t)^2}{2} \frac{dF(x(t_N), t_N)}{dt} + \cdots \]

this requires taking the derivative of \( F(x,t) \)

including 3\(^{rd}\) or 4\(^{th}\) or 5\(^{th}\) derivatives means even more work

at times, analytic expression of derivatives may be difficult or impossible
Simulating solutions to ordinary differential equations

“2nd order” Runge-Kutta

\[ x(t_{N + \Delta t}) \approx x(t_N) + \Delta t F(x(t_{N}), t_N) + \frac{(\Delta t)^2}{2} \frac{dF(x(t_N), t_N)}{dt} + \ldots \]

\[ x(t_{N + \Delta t}) \approx x(t_N) + K_2 \]

\[ K_1 = \Delta t F(x(t_N), t_N) \]

\[ K_2 = \Delta t F(x(t_N) + \frac{1}{2} K_1, t_N + \frac{1}{2} \Delta t) \]
Simulating solutions to ordinary differential equations

"2nd order" Runge-Kutta

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t F(x(t_N), t_N) + \frac{(\Delta t)^2}{2} \frac{dF(x(t_N), t_N)}{dt} + \ldots \]

\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F(x(t_N), t_N, t_N + \frac{1}{2} \Delta t) \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[x(t_N), t_N\right] + \frac{1}{2} \Delta t F\left[x(t_N), t_N\right], t_N + \frac{1}{2} \Delta t \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[x(t_N) + \frac{1}{2}\Delta t F(x(t_N),t_N),t_N + \frac{1}{2} \Delta t\right] \]
If we stopped here, it would be Euler’s Method

\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F \left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F \left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F \left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left(x(t_N), t_N\right) + \frac{1}{2} \Delta t F\left[x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t\right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
This is also called the "mid-point method"

\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t\right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F \left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t \left[ x(t_N) + \frac{\Delta t}{2} F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F \left[ x(t_N) + \frac{1}{2} \Delta t F(x(t_N), t_N), t_N + \frac{1}{2} \Delta t \right] \]
\[ x_{N+1} = x(t_N + \Delta t) \approx x(t_N) + \Delta t F\left[ x(t_N) + \frac{\Delta t}{2} F(x(t_N), t_N), \ t_N + \frac{1}{2} \Delta t \right] \]
Simulating solutions to ordinary differential equations

“2\textsuperscript{nd} order” Runge-Kutta

Euler’s Method

2\textsuperscript{nd} Order Runge-Kutta
YouTube Videos with further explanation of equations for Runge Kutta

Runge Kutta 2nd Order Method
http://www.youtube.com/watch?v=OnpwDGumKiw

Runge Kutta 2nd Order Method: Formulas
http://www.youtube.com/watch?v=ihEfPp5WRcE&feature=relmfu

Runge Kutta 2nd Order Method: Derivation (Part 1)
http://www.youtube.com/watch?v=X-_qCcYDbuY&feature=edu&list=PL9C549A03F84233ED

Runge Kutta 2nd Order Method: Derivation (Part 2)
http://www.youtube.com/watch?v=hhgG8KL_pCk&feature=autoplay&list=EC9C549A03F84233ED&playnext=1

Runge Kutta 4th Order Method
http://www.youtube.com/watch?v=hGN54bkE8Ac&feature=edu&list=PL8A2E3E2A5106FFA1
Simulating solutions to ordinary differential equations

“4th order” Runge-Kutta

\[
x(t_N + \Delta t) \approx x(t_N) + \frac{1}{6} (K_1 + 2K_2 + 2K_3 + K_4)
\]

\[
K_1 = \Delta t \ F(x(t_N), t_N)
\]

\[
K_2 = \Delta t \ F(x(t_N) + \frac{1}{2} K_1, t_N + \frac{1}{2} \Delta t)
\]

\[
K_3 = \Delta t \ F(x(t_N) + \frac{1}{2} K_2, t_N + \frac{1}{2} \Delta t)
\]

\[
K_4 = \Delta t \ F(x(t_N) + K_3, t_N + \Delta t)
\]

approximates Taylor expansion to the 4th derivative without needing to take derivatives
Simulating solutions to ordinary differential equations

“4th order” Runge-Kutta

Euler’s Method

2^{nd} Order Runge-Kutta

4^{th} Order Runge-Kutta
Simulating solutions to ordinary differential equations

“4th order” Runge-Kutta

Euler’s Method  2nd Order Runge-Kutta  4th Order Runge-Kutta

How do you decide the step size for any of these methods:
- trial and error, especially for systems repeatedly simulated
- also adaptive step size techniques
- relative to some other metric, like conservation of energy
Adapting Step Size

How do you adjust the step size $\Delta t$?

A very small step size means accurate simulations that can take an extraordinarily long time to complete.

A large step size means fast simulations that have poor accuracy.
Adapting Step Size

You can pick a global step size $\Delta t$ that "works" – e.g., keep cutting the step size in half until the simulation converges for a range of starting points.
Adapting Step Size

Compare RKs of different orders.

e.g., at each step, if a 4\textsuperscript{th} and 5\textsuperscript{th} order RK give basically the same $x(t_{N}+\Delta t)$ for a given step size $\Delta t$ then $\Delta t$ is small enough, otherwise cut $\Delta t$ in half and try again; and sometimes you may want to try to double the size of $\Delta t$ so the simulation runs faster.

\texttt{ode45()} does this
Adapting Step Size

Compare an RK with step size $\Delta t$ to two RKs, each with steps $\Delta t/2$

Illustrating using Euler's Method
Adapting Step Size

Compare an RK with step size $\Delta t$ to two RKs, each with steps $\Delta t/2$
Adapting Step Size

Compare an RK with step size $\Delta t$ to two RKs, each with steps $\Delta t/2$
Adapting Step Size

Compare an RK with step size $\Delta t$ to two RKs, each with steps $\Delta t/2$
Adapting Step Size

Compare an RK with step size $\Delta t$ to two RKs, each with steps $\Delta t/2$.

Adjust $\Delta t$ until this is "small enough" (tolerance).

$e_N$ e.g., Runge-Kutta-Fehlberg Method (RKF45)
Adapting Step Size

Compare an RK with step size $\Delta t$ to two RKs, each with steps $\Delta t/2$ and if they are “too close” you can increase $\Delta t$

e.g., Runge-Kutta-Fehlberg Method (RKF45)
Systems of Differential Equations
Ordinary differential equation

\[
\frac{dx}{dt} = \frac{1}{\tau} (I - kx)
\]
Systems of ordinary differential equations

\[ I_1 \rightarrow x_1 \rightarrow k \]

\[ \beta_{12} \]

\[ \beta_{21} \]

\[ I_2 \rightarrow x_2 \rightarrow k \]
Systems of ordinary differential equations

\[ \frac{dx_1}{dt} = \frac{1}{\tau}(I_1 - kx_1 - \beta_{21}x_2) \]

\[ \frac{dx_2}{dt} = \frac{1}{\tau}(I_2 - kx_2 - \beta_{12}x_1) \]
Systems of ordinary differential equations

\[
\begin{align*}
\frac{dx_1}{dt} &= \frac{1}{1} (1.2 - x_1 - x_2) \\
\frac{dx_2}{dt} &= \frac{1}{1} (1.0 - x_2 - x_1)
\end{align*}
\]
Systems of ordinary differential equations

\[ x(t) \]

Graph with nodes labeled as \( x_1 \) and \( x_2 \) connected by edges labeled with 1.2, 1.0, and 1.
The same equation can exhibit qualitatively different behavior depending on the values of the parameters
The same equation can exhibit qualitatively different behavior depending on the values of the parameters

\[
\frac{dx_1}{dt} = \frac{1}{\tau} (I_1 - kx_1 - \beta_{21}x_2)
\]

\[
\frac{dx_2}{dt} = \frac{1}{\tau} (I_2 - kx_2 - \beta_{12}x_1)
\]
The same equation can exhibit qualitatively different behavior depending on the values of the parameters

\[
\frac{dx_1}{dt} = \frac{1}{1}(1.2 - x_1 - x_2)
\]

\[
\frac{dx_2}{dt} = \frac{1}{1}(1.0 - x_2 - x_1)
\]
The same equation can exhibit qualitatively different behavior depending on the values of the parameters.
The same equation can exhibit qualitatively different behavior depending on the values of the parameters.

\[
\frac{dx_1}{dt} = \frac{1}{1} \left( 5.0 - 2x_1 - 20x_2 \right)
\]

\[
\frac{dx_2}{dt} = \frac{1}{1} \left( 1.0 - 2x_2 + 4x_1 \right)
\]
The same equation can exhibit qualitatively different behavior depending on the values of the parameters.

\[ x(t) \]

\[ x_1(t), \quad x_2(t) \]
The same equation can exhibit qualitatively different behavior depending on the values of the parameters

\[
\frac{dx_1}{dt} = \frac{1}{1}(1.0 + x_1 - 2x_2)
\]

\[
\frac{dx_2}{dt} = \frac{1}{1}(1.0 - x_2 + 5x_1)
\]
The same equation can exhibit qualitatively different behavior depending on the values of the parameters.
The same equation can exhibit qualitatively different behavior depending on the values of the parameters.
The same equation can exhibit qualitatively different behavior depending on the values of the parameters.

Note: I've extended the time axis here.
Hodgkin-Huxley model of action potentials

\[
\frac{dV}{dt} = \frac{1}{C} \left(-I_{Na} - I_K - I_{leak} + I_{input}\right)
\]
Hodgkin-Huxley model of action potentials

\[
\frac{dV}{dt} = \frac{1}{C} \left( -I_{Na} - I_K - I_{\text{leak}} + I_{\text{input}} \right)
\]

\[
\frac{dV}{dt} = \frac{1}{C} \left( -g_{Na} m^3 h(V - E_{Na}) - g_K n^4 (V - E_K) - g_{\text{leak}} (V - E_{\text{leak}}) + I_{\text{input}} \right)
\]
Hodgkin-Huxley model of action potentials

\[
\frac{dV}{dt} = \frac{1}{C} \left( -I_{Na} - I_K - I_{leak} + I_{input} \right)
\]

\[
\frac{dV}{dt} = \frac{1}{C} \left( -g_{Na} m^3 h (V - E_{Na}) - g_K n^4 (V - E_K) - g_{leak} (V - E_{leak}) + I_{input} \right)
\]

conductance (g)  voltage  Ohm’s Law

1 / resistance (R)

\[
V = IR
\]

\[
\frac{V}{R} = I
\]

\[
gV = I
\]
Hodgkin-Huxley model of action potentials

\[
\frac{dV}{dt} = \frac{1}{C} \left( -I_{Na} - I_K - I_{leak} + I_{input} \right)
\]

\[
\frac{dV}{dt} = \frac{1}{C} \left( -g_{Na} m^3 h (V - E_{Na}) - g_K n^4 (V - E_K) - g_{leak} (V - E_{leak}) + I_{input} \right)
\]

\[
\frac{dm}{dt} = \frac{1}{\tau_m(V)} (-m + M(V))
\]

\[
\frac{dh}{dt} = \frac{1}{\tau_h(V)} (-h + H(V))
\]

\[
\frac{dn}{dt} = \frac{1}{\tau_n(V)} (-n + N(V))
\]

measurements of time-dependent conductance and formulation of the differential equation won Hodgkin and Huxley the 1963 Nobel Prize in physiology
Hodgkin-Huxley model of action potentials

$I_{\text{input}} = 10 \, \text{uA/cm}^2$
Hodgkin-Huxley model of action potentials

$I_{\text{input}} = 6 \text{ uA/cm}^2$
Hodgkin-Huxley model of action potentials

\[ I_{\text{input}} = 1 \text{ uA/cm}^2 \]
Hodgkin-Huxley model of action potentials

\[ I_{\text{input}} = 2.2292 \, \text{uA/cm}^2 \]
Hodgkin-Huxley model of action potentials

\[ I_{\text{input}} = 2.2291 \text{ uA/cm}^2 \]
Another Example

Lotka-Volterra equations of population dynamics (Ecology)

**predator**
- wolves

\[ w = \text{# wolves per acre} \]

**prey**
- rabbit

\[ r = \text{# rabbits per acre} \]
Another Example

Lotka-Volterra equations of **population dynamics** (Ecology)

- **predator**: wolves
- **prey**: rabbit

\[
\begin{align*}
\frac{dw}{dt} &
\\
\frac{dr}{dt} &
\end{align*}
\]

**simplifying assumptions:**
- wolves are the only threat to rabbits as prey
- rabbits are the wolves only source of food
Another Example

Lotka-Volterra equations of population dynamics (Ecology)

\[ \frac{dr}{dt} = \alpha r - \beta rw \]
\[ \frac{dw}{dt} = -\gamma w + \delta rw \]

r is the number of rabbits in an area
\[ \frac{dr}{dt} \] is the rate of change of the rabbit population

w is the number of wolves in an area
\[ \frac{dw}{dt} \] is the rate of change of the wolf population
Another Example

Lotka-Volterra equations of population dynamics (Ecology)

\[ \frac{dr}{dt} = \alpha r - \beta rw \]

\[ \frac{dw}{dt} = -\gamma w + \delta rw \]

because of the nonlinearities

an analytic solution is hard

w is the number of wolves in an area

dw/dt is the rate of change of the wolf population

r is the number of rabbits in an area

dr/dt is the rate of change of the rabbit population
Another Example

Lotka-Volterra equations of population dynamics (Ecology)

rabbit population increases with the current number of rabbit

\[
\frac{dr}{dt} = \alpha r - \beta rw \\
\frac{dw}{dt} = -\gamma w + \delta rw
\]
Another Example

Lotka-Volterra equations of population dynamics (Ecology)

\[
\frac{dr}{dt} = \alpha r - \beta rw \\
\frac{dw}{dt} = -\gamma w + \delta rw
\]

wolf population decreases with the number of wolves
Another Example

Lotka-Volterra equations of population dynamics (Ecology)

\[
\begin{align*}
\frac{dr}{dt} &= \alpha r \quad \text{rabbits increase exponentially} \\
\frac{dw}{dt} &= -\gamma w \quad \text{rabbits decrease exponentially}
\end{align*}
\]

without any interactions between rabbits and wolves

(to be a realistic model, you will need to include the source of food for rabbits too)
Another Example

Lotka-Volterra equations of population dynamics (Ecology)

rabbit population decreases the more likely encounters with wolves

\[
\frac{dr}{dt} = \alpha r - \beta rw
\]

\[
\frac{dw}{dt} = -\gamma w + \delta rw
\]
Another Example

Lotka-Volterra equations of population dynamics (Ecology)

rabbit population decreases the more likely encounters with wolves

\[
\frac{dr}{dt} = \alpha r - \beta rw \\
\frac{dw}{dt} = -\gamma w + \delta rw
\]
Another Example

Lotka-Volterra equations of population dynamics (Ecology)

\[
\frac{dr}{dt} = \alpha r - \beta rw
\]

\[
\frac{dw}{dt} = -\gamma w + \delta rw
\]

wolf population increases the more likely encounters with rabbits
Another Example

Lotka-Volterra equations of population dynamics (Ecology)

\[
\frac{dr}{dt} = \alpha r - \beta rw
\]

\[
\frac{dw}{dt} = -\gamma w + \delta rw
\]

- \( \alpha \) is the growth rate in absence of wolves.
- \( \beta \) is the death rate due to predation.
- \( \gamma \) is the death rate in absence of rabbits.
- \( \delta \) is the efficiency of “turning” rabbits into wolves.
Another Example

\( r(t) \) and \( w(t) \)
An alternative way of displaying the dynamics

plot the number of rabbits \((r)\) against the number of wolves \((w)\)

*with time implicit*

\[
\begin{align*}
\frac{dr}{dt} &= \alpha r - \beta rw \\
\frac{dw}{dt} &= -\gamma w + \delta rw
\end{align*}
\]
An alternative way of displaying the dynamics

plot the number of rabbits \( (r) \) against the number of wolves \( (w) \)

*with time implicit*
An alternative way of displaying the dynamics

plot the number of rabbits (r) against the number of wolves (w) \textit{with time implicit}

\begin{align*}
  w(t) & \quad r(t) \\
  r(t) & \quad w(t) \\
  t & = 2
\end{align*}
An alternative way of displaying the dynamics

plot the number of rabbits \((r)\) against the number of wolves \((w)\)

*with time implicit*
An alternative way of displaying the dynamics

plot the number of rabbits (r) against the number of wolves (w) with time implicit
An alternative way of displaying the dynamics

plot the number of rabbits (r) against the number of wolves (w) 

*with time implicit*

\[ t = 5 \]
An alternative way of displaying the dynamics

plot the number of rabbits ($r$) against the number of wolves ($w$) with time implicit

$t = 6$
An alternative way of displaying the dynamics

plot the number of rabbits \( r \) against the number of wolves \( w \) with time implicit

\[ t = 7 \]
An alternative way of displaying the dynamics

plot the number of rabbits \(r\) against the number of wolves \(w\) with time implicit
Another Example

compare Runge-Kutta and Euler’s method ...
Another Example

even with a small dt, Euler’s method misses
Characterizing the qualitative behavior of dynamic systems

\[
\frac{dx_1}{dt} = 5 - 2x_1 - 20x_2 \\
\frac{dx_2}{dt} = 1 - 2x_2 + 4x_1
\]
Characterizing the qualitative behavior of dynamic systems

Phase Space / Phase Plot / Phase Diagram

\[
\frac{dx_1}{dt} = 5 - 2x_1 - 20x_2 \\
\frac{dx_2}{dt} = 1 - 2x_2 + 4x_1
\]
Characterizing the qualitative behavior of dynamic systems

Phase Space / Phase Plot / Phase Diagram

\[
\frac{dx_1}{dt} = 5 - 2x_1 - 20x_2
\]

\[
\frac{dx_2}{dt} = 1 - 2x_2 + 4x_1
\]
Characterizing the qualitative behavior of dynamic systems

Phase Space / Phase Plot / Phase Diagram

\[
\frac{dx_1}{dt} = 5 - 2x_1 - 20x_2
\]

\[
\frac{dx_2}{dt} = 1 - 2x_2 + 4x_1
\]

different starting positions
Characterizing the qualitative behavior of dynamic systems

**Attractors**
equilibrium within a local region is a point
start points in the local region move to that point

**Repellers**
the point represents an equilibrium
but any starting point any epsilon distance away moves away
Characterizing the qualitative behavior of dynamic systems

**Attractors**
equilibrium within a local region is a point
start points in the local region move to that point

**Repellers**
the point represents an equilibrium
but any starting point any epsilon distance away moves away

**Cycles**

**Limit Cycles**

**Chaotic Attractors**
Characterizing the qualitative behavior of dynamic systems

\[
\frac{dr}{dt} = \alpha r - \beta rw
\]

\[
\frac{dw}{dt} = -\gamma w + \delta rw
\]
Characterizing the qualitative behavior of dynamic systems

\[ \frac{dr}{dt} = 2r - rw \]

\[ \frac{dw}{dt} = -w + \frac{1}{2}rw \]
Characterizing the qualitative behavior of dynamic systems

\[
\frac{dr}{dt} = 2r - rw \\
\frac{dw}{dt} = -w + \frac{1}{2} rw
\]
Characterizing the qualitative behavior of dynamic systems

\[
\begin{align*}
\frac{dr}{dt} & = 2r - rw \\
\frac{dw}{dt} & = -w + \frac{1}{2}rw
\end{align*}
\]
Characterizing the qualitative behavior of dynamic systems

\[
\frac{dV}{dt} = \frac{1}{C} \left( -g_{Na} m^3 h (V - E_{Na}) - g_K n^4 (V - E_K) - g_{leak} (V - E_{leak}) + I_{input} \right)
\]
Characterizing the qualitative behavior of dynamic systems

\[
\frac{dV}{dt} = \frac{1}{C} \left( -g_{Na} m^3 h (V - E_{Na}) - g_K n^4 (V - E_K) - g_{\text{leak}} (V - E_{\text{leak}}) + I_{\text{input}} \right)
\]

Limit Cycle
Cycles vs. Limit Cycles

Cycle

Limit Cycle

- $x_0 = [6, 2]$
- $x_0 = [3, 5]$
- $x_0 = [8, 2]$
Cycles vs. Limit Cycles

no noise
Cycles vs. Limit Cycles

add some “noise”
Another way to view behavior of a dynamical system

vector field plots, aka quiver plots
let you glimpse the global behavior of the system
Another way to view the behavior of a dynamical system is through vector field plots, aka quiver plots, which let you glimpse the global behavior of the system.
Another way to view behavior of a dynamical system

vector field plots, aka quiver plots
let you glimpse the global behavior of the system
Another way to view behavior of a dynamical system

vector field plots, aka quiver plots
let you glimpse the global behavior of the system
another example of a limit cycle

\[
\frac{dx}{dt} = y \\
\frac{dy}{dt} = a(1 - x^2)y - bx
\]
Homework 12
Simulate solution to systems of differential equations
Another example: a neural short-term memory model

\[
\begin{align*}
\frac{dx_1}{dt} &= -x_1 + f(x_2) \\
\frac{dx_2}{dt} &= -x_2 + f(x_1)
\end{align*}
\]

let’s assume that \( x_1 \) and \( x_2 \) have some values \( x_1(0) \) and \( x_2(0) \) at time 0
Another example: a neural short-term memory model

\[
\begin{align*}
\frac{dx_1}{dt} &= -x_1 + \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2} \\
\frac{dx_2}{dt} &= -x_2 + \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2}
\end{align*}
\]
Another example: a neural short-term memory model

\[
\frac{dx_1}{dt} = -x_1 + \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2}
\]

\[
\frac{dx_2}{dt} = -x_2 + \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2}
\]

example of \(x_1(t)\) and \(x_2(t)\) for a given starting point \(x_1(0)\) and \(x_2(0)\)

\(x_0 = [10, 25]\)
Another example: a neural short-term memory model

\[
\begin{align*}
\frac{dx_1}{dt} &= -x_1 + \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2} \\
\frac{dx_2}{dt} &= -x_2 + \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2}
\end{align*}
\]

example of \(x_1(t)\) and \(x_2(t)\) for a different starting point \(x_1(0)\) and \(x_2(0)\)

\[x_0 = [25,100]\]
Another example: a neural short-term memory model

\[
\begin{align*}
\frac{dx_1}{dt} &= -x_1 + 3x_2 \\
\frac{dx_2}{dt} &= -x_2 + 3x_1
\end{align*}
\]

what happens without the nonlinearity?
Another example: a neural short-term memory model

\[
\frac{dx_1}{dt} = -x_1 + \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2}
\]

\[
\frac{dx_2}{dt} = -x_2 + \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2}
\]

global behavior with a vector field plot on a phase diagram
Another example: a neural short-term memory model.

\[
\begin{align*}
\frac{dx_1}{dt} &= -x_1 + 100 \left( 3x_2^2 \right)^2 + \left( 120 + a_1 \right)^2 + \left( 3x_2 \right)^2 \\
\frac{dx_2}{dt} &= -x_2 + 100 \left( 3x_1^2 \right)^2 + \left( 120 + a_2 \right)^2 + \left( 3x_1 \right)^2
\end{align*}
\]

Global behavior with a vector field plot on a phase diagram.

\[
\begin{align*}
x_{\text{min}} &= 0 \\
x_{\text{max}} &= 100 \\
x_{\text{step}} &= 2.5 \\
x_1 &= \text{arange}(0, x_{\text{max}} + x_{\text{step}}, x_{\text{step}}) \\
x_2 &= \text{arange}(0, x_{\text{max}} + x_{\text{step}}, x_{\text{step}}) \\
(X_1, X_2) &= \text{meshgrid}(x_1, x_2) \\
a_1 &= 0 \\
a_2 &= 0 \\
U &= -X_1 + \frac{100 \left( 3X_2^2 \right)^2}{\left( 120 + a_1 \right)^2 + \left( 3X_2 \right)^2} \\
V &= -X_2 + \frac{100 \left( 3X_1^2 \right)^2}{\left( 120 + a_2 \right)^2 + \left( 3X_1 \right)^2}
\end{align*}
\]
Another example: a neural short-term memory model

\[
\begin{align*}
\frac{dx_1}{dt} &= -x_1 + \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2} \\
\frac{dx_2}{dt} &= -x_2 + \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2}
\end{align*}
\]

global behavior with a vector field plot on a phase diagram

zoom in
Another example: a neural short-term memory model

\[
\frac{dx_1}{dt} = -x_1 + \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2}
\]

\[
\frac{dx_2}{dt} = -x_2 + \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2}
\]

global behavior with a vector field plot on a phase diagram
Another example:

A neural short-term memory model

dx_1 = -x_1 + 100((3x_2)^2) / ((120 + a_1)^2 + (3x_2)^2))

dx_2 = -x_2 + 100((3x_1)^2) / ((120 + a_2)^2 + (3x_1)^2))

Global behavior with a vector field plot on a phase diagram

x_min = 0
x_max = 100
x_step = 2.5

x1 = arange(0, x_max + x_step, x_step)
x2 = arange(0, x_max + x_step, x_step)
(X1, X2) = meshgrid(x1, x2)

a1 = 0
a2 = 0
U = -X1 + (100*((3*X2)^2) / ((120 + a_1)^2 + (3*X_2)^2))
V = -X2 + (100*((3*X_1)^2) / ((120 + a_2)^2 + (3*X_1)^2))
Another example: a neural short-term memory model

\[
\begin{align*}
\frac{dx_1}{dt} &= -x_1 + \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2} \\
\frac{dx_2}{dt} &= -x_2 + \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2}
\end{align*}
\]

global behavior with a vector field plot on a phase diagram

this dynamical system has three “nodes”

2 attract
1 repels
Another example: a neural short-term memory model

\[
\begin{align*}
\frac{dx_1}{dt} &= 0 \\
\frac{dx_2}{dt} &= 0
\end{align*}
\]

Where does the system refuse to change?

One way to further investigate the global behavior is through “isoclines”
Another example: a neural short-term memory model

\[
\frac{dx_1}{dt} = -x_1 + \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2} = 0
\]

\[
\frac{dx_2}{dt} = -x_2 + \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2} = 0
\]
Another example: a neural short-term memory model

\[ x_1 = \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2} \]

\[ x_2 = \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2} \]
Another example: a neural short-term memory model

\[ x_1 = \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2} \]

\[ x_2 = \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2} \]

Question: how does the qualitative behavior change with values of \( a \)?
Another example: a neural short-term memory model

\[
x_1 = \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2}
\]

\[
x_2 = \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2}
\]
Another example: a neural short-term memory model

\[ x_1 = \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2} \]

\[ x_2 = \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2} \]

\[ a = 25 \]
Another example:
a neural short-term memory model

\begin{align*}
x_1 &= (3x_2)^2 \\
120 &= (120 + a_1)^2 + (3x_2)^2 \\
25 &= (3x_1)^2 \\
\end{align*}

time

Neural Response
Another example: a neural short-term memory model

\[
x_1 = \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2}
\]

\[
x_2 = \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2}
\]

A bifurcation occurs at \(a=30\)

One equilibrium point gets split into two, or vice-versa
Another example: a neural short-term memory model

\[ x_1 = \frac{100(3x_2)^2}{(120 + a_1)^2 + (3x_2)^2} \]

\[ x_2 = \frac{100(3x_1)^2}{(120 + a_2)^2 + (3x_1)^2} \]

A catastrophe occurs at \( a > 30 \) that equilibrium point disappears.
Another example: a neural short-term memory model.
Another example: a neural short-term memory model

\[
\begin{align*}
\frac{dx_1}{dt} &= -x_1 + \frac{100(3x_2)^2}{(120 + a(t))^2 + (3x_2)^2} \\
\frac{dx_2}{dt} &= -x_2 + \frac{100(3x_1)^2}{(120 + a(t))^2 + (3x_1)^2}
\end{align*}
\]

imagine that \(a(t)\) varies with time …

… then the qualitative behavior of the system will change over time

the \(a(t)\) term could be the “adaptation” of the neurons
Lorenz Attractor (chaotic attractor)

\[
\frac{dx}{dt} = 10(-x + y)
\]

\[
\frac{dy}{dt} = -y + 28x - xz
\]

\[
\frac{dz}{dt} = -\frac{8}{3}z + xy
\]
Lorenz Attractor (chaotic attractor)

Published by meteorologist and mathematician Edward Lorenz in 1963 to `model some of the unpredictable behavior which we normally associate with the weather``

... x, y, z are not spatial coordinate. The "x is proportional to the intensity of the convective motion, while y is proportional to the temperature difference between the ascending and descending currents, similar signs of x and y denoting that warm fluid is rising and cold fluid is descending. The variable z is proportional to the distortion of vertical temperature profile from linearity, a positive value indicating that the strongest gradients occur near the boundaries."
Matlab
sensitivity to starting points for cycles, limit cycles, and chaotic attractors

Lotka-Volterra

Hodgkin-Huxley

Lorenz Attractor

each plot shows two trajectories differing in starting point by only 1%
sensitivity to starting points for cycles, limit cycles, and chaotic attractors

Each plot shows two trajectories differing in starting point by only 1%
sensitivity to starting points for cycles, limit cycles, and chaotic attractors
sensitivity to starting points for cycles, limit cycles, and chaotic attractors
sensitivity to starting points for cycles, limit cycles, and chaotic attractors
sensitivity to starting points for cycles, limit cycles, and chaotic attractors
sensitivity to starting points for cycles, limit cycles, and chaotic attractors

“Butterfly Effect”
Lorenz is truly an "attractor"
Lorenz is truly an "attractor"

\[
\begin{align*}
x(0) &= -300 \\
y(0) &= -1200 \\
z(0) &= 0
\end{align*}
\]

\[
\begin{align*}
x(0) &= 1200 \\
y(0) &= 1000 \\
z(0) &= 500
\end{align*}
\]
Lorenz is truly at "attractor"

\[
\begin{align*}
  x(0) &= -300 \\
  y(0) &= -1200 \\
  z(0) &= 0
\end{align*}
\]

\[
\begin{align*}
  x(0) &= 1200 \\
  y(0) &= 1000 \\
  z(0) &= 500
\end{align*}
\]
as with many dynamical systems, the Lorenz attractor is also sensitive to the values of its parameters

\[ \sigma = 20 \]

\[ \rho = 20 \]