$f'(y) \approx \alpha_1 y_1 + \alpha_2 y_2 + \alpha_3 y_3 + \alpha_4 y_4 + O(h^2)$

$d_i = \psi_i(x_1, \ldots, x_n)$
Differentiation Matrices

How can numerical differentiation be cast as a matrix-vector operation?

\[ p_n(x) = \sum_{i=1}^{n} \alpha_i \phi_i(x) \]

\[ p'_n(x) = \sum_{i=1}^{n} \alpha_i \phi'_i(x) \]

\[ \mathbf{V}^T \mathbf{V}^{-1} \mathbf{f}(x) \approx \phi'(x) \]

**Demo:** Taking Derivatives with Vandermonde Matrices [cleared] (Build \( D \))
\[ f'(y) \approx \frac{f'(x_1) + \ldots + f'(x_n)}{n} \]

implies each row of \( D \) contains a finite difference rule:

\[ f'(x_i) \approx d_{i1} f(x_1) + \ldots + d_{in} f(x_n) \]
Shift:

- does not change $D$, does not obey $D$ rule

Scale:

$$\frac{f'(0)}{h} \approx -\frac{1}{h} f(0) + \frac{1}{2} f(h)$$

$$= \frac{f(h) - f(0)}{h}$$

- scaling node by $\delta$: become $\frac{\Delta}{\delta}$
Properties of Differentiation Matrices

How do I find second derivatives?

Does $D$ have a nullspace?
Numerical Differentiation: Shift and Scale

Does $D$ change if we shift the nodes $(x_i)_{i=1}^n \rightarrow (x_i + c)_{i=1}^n$?

Does $D$ change if we scale the nodes $(x_i)_{i=1}^n \rightarrow (\alpha x_i)_{i=1}^n$?
Finite Difference Formulas from Diff. Matrices

How do the rows of a differentiation matrix relate to FD formulas?

Assume a large equispaced grid and 3 nodes w/same spacing. How to use?
Finite Differences: via Taylor

\[ f'(x) \approx \frac{f(x+h) - f(x)}{h} + o(h^2) \]

\[ f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2} + \ldots \]

\[ \frac{f(x+h) - f(x)}{h} = f'(x) + o(h) \]
More Finite Difference Rules

Similarly:

\[ f'(x) = \frac{f(x + h) - f(x - h)}{2h} + O(h^2) \]

(Centered differences)

Can also take higher order derivatives:

\[ f''(x) = \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} + O(h^2) \]

Can find these by trying to match Taylor terms. Alternative: Use linear algebra with interpolate-then-differentiate to find FD formulas.

**Demo:** Finite Differences vs Noise [cleared]

**Demo:** Floating point vs Finite Differences [cleared]
Outline

Introduction to Scientific Computing
Systems of Linear Equations
Linear Least Squares
Eigenvalue Problems
Nonlinear Equations
Optimization
Interpolation
Numerical Integration and Differentiation

Initial Value Problems for ODEs
  Existence, Uniqueness, Conditioning
  Numerical Methods (I)
  Accuracy and Stability
  Stiffness
  Numerical Methods (II)

Boundary Value Problems for ODEs
Partial Differential Equations and Sparse Linear Algebra
Fast Fourier Transform
Additional Topics
What can we solve already?

- Linear Systems: yes
- Nonlinear systems: yes
- Systems with derivatives: no
### Some Applications

<table>
<thead>
<tr>
<th>IVPs</th>
<th>BVPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population dynamics</td>
<td>bridge load</td>
</tr>
<tr>
<td>$y_1' = y_1(\alpha_1 - \beta_1 y_2)$ (prey)</td>
<td>pollutant concentration (steady state)</td>
</tr>
<tr>
<td>$y_2' = y_2(-\alpha_2 + \beta_2 y_1)$ (predator)</td>
<td>temperature (steady state)</td>
</tr>
<tr>
<td>chemical reactions</td>
<td>waves (time-harmonic)</td>
</tr>
<tr>
<td>equations of motion</td>
<td></td>
</tr>
</tbody>
</table>

#### Demo: Predator-Prey System [cleared]
Initial Value Problems: Problem Statement

Want: Function $y : [0, T] \rightarrow \mathbb{R}^n$ so that

- $y^{(k)}(t) = f(t, y, y', y'', \ldots, y^{(k-1)})$ (explicit), or
- $f(t, y, y', y'', \ldots, y^{(k)}) = 0$ (implicit)

are called explicit/implicit $k$th-order ordinary differential equations (ODEs).

Give a simple example.

$$y' = ay \quad y(0) = c \cdot e^{kt}$$

Not uniquely solvable on its own. What else is needed?

$$y'' = f(y, y') \quad \text{needs } y \text{ and } y'$$

Need $y(0) = \ldots$ and $y^{(k-1)}(0) = \ldots$ as initial conditions.
Reducing ODEs to First-Order Form

A $k$th order ODE can always be reduced to first order. Do this in this example:

$$y''(t) = f(y)$$

\[
\begin{align*}
\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}' &= \begin{bmatrix} w_2 \\ \psi(w_1) \end{bmatrix} \\
\Rightarrow & \text{ For numerics: need only worry about (system of) first-order ODEs}
\end{align*}
\]
Properties of ODEs

What is a **linear** ODE?

\[
\dot{y}' = \Phi(t, \dot{y})
\]

\[
\dot{\Phi}(t, \dot{y}) = A(t) \dot{y} + \bar{b}(t)
\]

What is a **linear and homogeneous** ODE?

\[
\dot{\Phi}(t, \dot{y}) = A(t) \dot{y}
\]

What is a **constant-coefficient** ODE?

\[
\Phi(t, \dot{y}) = A \dot{y} + B
\]
Existence and Uniqueness

Consider the perturbed problem

\[
\begin{align*}
y'(t) &= f(y) \\
y(t_0) &= y_0
\end{align*}
\]

Then if \( f \) is \textit{Lipschitz continuous} (has ‘bounded slope’), i.e.

\[
\|f(y) - f(\hat{y})\| \leq L \|y - \hat{y}\|
\]

(where \( L \) is called the \textit{Lipschitz constant}), then...

- There exists a solution \( y \) in a neighborhood of \( t_0 \)
- \( \|y(t) - \hat{y}(t)\| \leq e^{L(t-t_0)} \|y_0 - \hat{y}_0\| \)

What does this mean for uniqueness?

Implicitly covers uniqueness as well

\( y_0 = \hat{y}_0 \Rightarrow y(t) = \hat{y}(t) \)
Conditioning

Unfortunate terminology accident: “Stability” in ODE-speak
To adapt to conventional terminology, we will use ‘Stability’ for

▶ the conditioning of the IVP, and
▶ the stability of the methods we cook up.

Some terminology:

An ODE is **stable** if and only if...

\[
\text{The solution is continuously dependent on the IC.}
\]

For all \( c > 0 \), there exists a \( \delta > 0 \) so that

\[
|| \tilde{y}_0 - \hat{y}_0 || < \delta \Rightarrow || \tilde{y}(t) - \hat{y}(t) || < c \quad \text{for all } t \geq t_0.
\]

An ODE is **asymptotically stable** if and only if

\[
|| \tilde{y}(t) - \hat{y}(t) || \rightarrow 0 \quad t \rightarrow \infty
\]
Example 1: Scalar, Constant-Coefficient

\[
\begin{align*}
  y'(t) &= \lambda y \\
  y(0) &= y_0
\end{align*}
\]

where \( \lambda = a + ib \)

Solution?

\[
y(t) = y_0 e^{\lambda t} = y_0 e^{(a + ib)t} = y_0 e^{at} e^{ibt}
\]

When is this stable?

- \( \Re \lambda > 0 \): unstable
- \( \Re \lambda = 0 \): stable, not asymptotically stable
- \( \Re \lambda < 0 \): asymptotically stable

Magnitude: \( |\lambda| \leq 1 \) implies "oscillation"
Example II: Constant-Coefficient System

\[
A = VDV^{-1}
\]

\[
\begin{align*}
\dot{y}(t) &= Ay(t) \\
y(t_0) &= y_0
\end{align*}
\]

Assume \( V^{-1}AV = D = \text{diag}(\lambda_1, \ldots, \lambda_n) \) diagonal. Find a solution.

\[
\tilde{w} = V^{-1}y \\
\dot{\tilde{w}} = \dot{y} = \dot{V}^{-1}y = V^{-1}A y = V^{-1}DV^{-1}y = D\tilde{w}
\]

When is this stable?

\[
\text{Re}(\lambda_i) < 0 \rightarrow \text{asympt. stable} \\
\text{Re}(\lambda_i) > 0 \rightarrow \text{unstable}
\]
Euler’s Method

Discretize the IVP

\[
\begin{align*}
\begin{cases}
    y'(t) &= f(y) \\
    y(t_0) &= y_0
\end{cases}
\end{align*}
\]

- Discrete times: \( t_1, t_2, \ldots \), with \( t_{i+1} = t_i + h \)
- Discrete function values: \( y_k \approx y(t_k) \).

\[ y(1) = y_0 + \int_{t_0}^{1} f(y(t)) \, dt \]

- Forward Euler: \( y(1) = y_0 + f(y_0) \cdot \Delta t \)
- "left rectangle rule": \( \int_{a}^{b} f(x)dx \approx f(a) \cdot (b-a) \)
Euler’s method: Forward and Backward

\[ y(t) = y_0 + \int_{t_0}^{t} f(y(\tau)) d\tau, \]

Use ‘left rectangle rule’ on integral:

\[ \tilde{y}_{n+1} = y_n + h \int f(y_n) \]

(explicit \rightarrow can just evaluate \[y_{k+1}\])

Use ‘right rectangle rule’ on integral:

\[ \tilde{y}_{n+1} = y_n + h f(y_{n+1}) \]

(implicit \rightarrow solve for \[ y_{n+1} \])

Demo: Forward Euler stability [cleared]