Ordinary Differential Equations (ODEs)

CHPC & NITheCS Summer School

Dr Graeme Pleasance

Department of Physics
Stellenbosch University

February 8, 2024

Outline

ODEs are ubiquitous in physical sciences.

Dynamical systems:

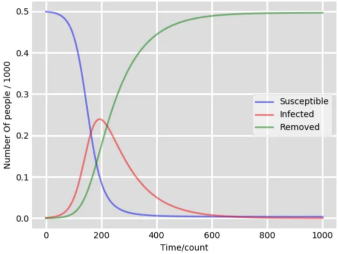
- Newton's laws of motion:

$$\vec{F}_{\text{tot}} = m\frac{d\vec{x}^2(t)}{dt^2}$$

- Population modelling:

$$\frac{dp(t)}{dt} = rp(t)(k - p(t))$$

- Epidemiology, economics, engineering, e.t.c.



https://medium.com/geekculture/modelling-a-modern-day-pandemic-developing-the-sir-model-8d77599050ce

Ordinary Differential Equations (ODEs)

Definition

Group of linear equations that relate an unknown function $y = f(x)$, $f : D \to \mathbb{R}$ to it's ordinary nth order derivatives $y^{(n)}$:

$$a_n(x)\frac{d^n y}{dx^n} + ... + a_1(x)\frac{dy}{dx} + a_0(x)y + b(x) = 0,$$

where $a_n(x), ..., a_1(x), b(x)$ are arbitrary differentiable functions.

Ordinary Differential Equations (ODEs)

Definition

Group of linear equations that relate an unknown function $y = f(x)$, $f : D \to \mathbb{R}$ to it's ordinary nth order derivatives $y^{(n)}$:

$$a_n(x)\frac{d^n y}{dx^n} + ... + a_1(x)\frac{dy}{dx} + a_0(x)y + b(x) = 0,$$

where $a_n(x), ..., a_1(x), b(x)$ are arbitrary differentiable functions.

- Order n corresponds to power of highest derivative of y, e.g.

$$\begin{aligned} &y' + a_0 y = 0, && n = 1 \\ &y'' + a_1(x)y' + y = 0, && n = 2 \end{aligned}$$

Ordinary Differential Equations (ODEs)

Definition

Group of linear equations that relate an unknown function $y = f(x)$, $f : D \to \mathbb{R}$ to it's ordinary nth order derivatives $y^{(n)}$:

$$a_n(x)\frac{d^n y}{dx^n} + ... + a_1(x)\frac{dy}{dx} + a_0(x)y + b(x) = 0,$$

where $a_n(x), ..., a_1(x), b(x)$ are arbitrary differentiable functions.

- Order n corresponds to power of highest derivative of y, e.g.

$$\begin{aligned} &y' + a_0 y = 0, && n = 1 \\ &y'' + a_1(x)y' + y = 0, && n = 2 \end{aligned}$$

- Homogenous if $b(x) = 0$.
- Non-linear: e.g. $(y')^{3/2} + \sin(y) = 0$.

Solution

For ODE of order n the solution in general depends on n constants.

- Uniqueness: for solution to be unique, there must be n boundary conditions to determine the n constants.
- Boundary conditions: externally imposed conditions on the solution. For nth order ODE, could be value of y at n different x-values, or any n combination of values of y, y', y''...

Solution

For ODE of order n the solution in general depends on n constants.

- Uniqueness: for solution to be unique, there must be n boundary conditions to determine the n constants.
- Boundary conditions: externally imposed conditions on the solution. For nth order ODE, could be value of y at n different x-values, or any n combination of values of y, y', y''...

Initial value problem: When the n boundary conditions are specified at the same initial value $x = x_0$: e.g.,

$$y'' + a_1(x)y' + y = 0, \quad n = 2,$$

with $y(x_0) = y_0$ and $y'(x_0) = y'_0$.

Examples

1. Population growth (1st order):

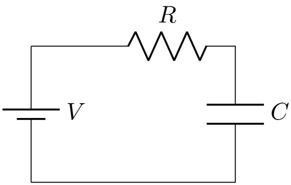
$$\dot{p}(t) = r(k - p(t))$$

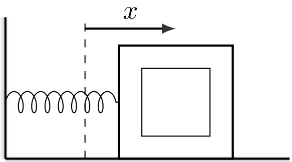
2. RC-circuit (1st order):

$$\dot{q}(t) + \frac{q(t)}{RC} = \frac{V(t)}{R}$$

3. Damped-driven oscillator (2nd order):

$$\ddot{x}(t) + \gamma\dot{x}(t) + \omega_0^2 x(t) = F_d(t)$$





First order ODEs

General ODE order $n = 1$:

$$\boxed{y' = F(g(y), h(x))}$$

Separable:

$$y' = g(y)h(x) \implies \int \frac{dy}{g(y)} = \int dx\, h(x)$$

First order ODEs

General ODE order $n = 1$:

$$\boxed{y' = F(g(y), h(x))}$$

Separable:

$$y' = g(y)h(x) \implies \int \frac{dy}{g(y)} = \int dx\, h(x)$$

Example (Population growth)

Solve $\dot{p}(t) = r(k - p(t))$ with $p(0) = p_0$.

First order ODEs

General ODE order $n = 1$:

$$\boxed{y' = F(g(y), h(x))}$$

Separable:

$$y' = g(y)h(x) \implies \int \frac{dy}{g(y)} = \int dx\, h(x)$$

Example (Population growth)

Solve $\dot{p}(t) = r(k - p(t))$ with $p(0) = p_0$.

Define $\quad z(t) = p(t) - k, \quad \dot{z}(t) = -rz(t)$:

$$\ln z(t) = -rt + \ln(h), \qquad C = \ln(h)$$

$$\implies z(t) = he^{-rt}$$

First order ODEs

General ODE order $n = 1$:

$$\boxed{y' = F(g(y), h(x))}$$

Separable:

$$y' = g(y)h(x) \implies \int \frac{dy}{g(y)} = \int dx\, h(x)$$

Example (Population growth)

Solve $\dot{p}(t) = r(k - p(t))$ with $p(0) = p_0$.

Define $z(t) = p(t) - k, \quad \dot{z}(t) = -rz(t)$:

$$\ln z(t) = -rt + \ln(h), \qquad C = \ln(h)$$

$$\implies z(t) = he^{-rt}$$

Transform back: $p(t) = k + (p_0 - h)e^{-rt}, \qquad p_0 = k + h \implies h = p_0 - k$:

$$\boxed{p(t) = k + (p_0 - k)e^{-rt}}$$

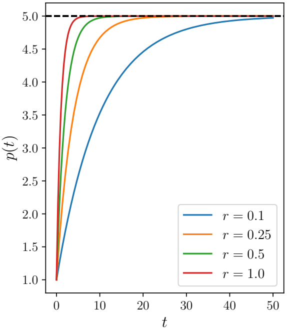
First order ODEs

Example (Population growth)

Solution $p(t) = k + (p_0 - k)e^{-rt}$:

- $r =$ Growth rate.
- $k =$ Population at equilibrium.

```python
import matplotlib.pyplot as plt
import numpy as np

tlist = np.linspace(0,50,150)
rlist = [0.1, 0.25, 0.5, 1.0]

p0 = 1  # initial popluation
k = 5   # equilibrium population

fig, ax = plt.subplots(figsize=(4,5))

for r in rlist:
    ax.plot(tlist, k + (p0-k)*np.exp(-r*tlist), label=r'$r={0}$'.format(r))
    ax.axhline(k, tlist[0], tlist[-1], linestyle='--', c='k')

ax.set_xlabel(r'$t$')
ax.set_ylabel(r'$p(t)$')
ax.legend(loc=0)
```



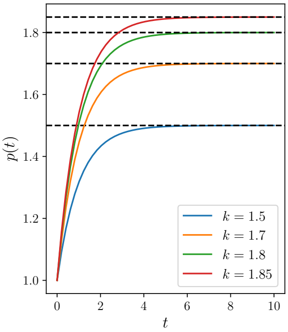
First order ODEs

Example (Population growth)

Solution $p(t) = k + (p_0 - k)e^{-rt}$:

- $r =$ Growth rate.
- $k =$ Population at equilibrium.

```python
import matplotlib.pyplot as plt
import numpy as np

tlist = np.linspace(0,10)
klist = [1.5, 1.7, 1.8, 1.85]

p0 = 1  # initial popluation
r = 1   # growth rate

fig, ax = plt.subplots(figsize=(4,5))

for k in klist:
    ax.plot(tlist, k + (p0-k)*np.exp(-r*tlist),
      label=r'$k={0}$'.format(k))
    ax.axhline(k, tlist[0], tlist[-1], linestyle
      ='--', c='k')

ax.set_xlabel(r'$t$')
ax.set_ylabel(r'$p(t)$')
ax.legend(loc=0)
```



First order ODEs

Inhomogeneous 1st order ODE:

$$\boxed{y' + P(x)y = Q(x)}$$

First order ODEs

Inhomogeneous 1st order ODE:

$$\boxed{y' + P(x)y = Q(x)}$$

Integrating factor $I(x)$

Multiply both sides by $I(x)$:

$$I(x)y' + I(x)P(x)y := \frac{d}{dx}(Iy) = Q(x)I(x)$$

$$I(x)y = \int dx\, Q(x)I(x)$$

$$\frac{d}{dx}(Iy) = I'(x)y + I(x)y' \implies I'(x) = P(x)I(x), \quad I(x) = e^{\int dx\, P(x)}$$

First order ODEs

Inhomogeneous 1st order ODE:

$$\boxed{y' + P(x)y = Q(x)}$$

Integrating factor $I(x)$

Multiply both sides by $I(x)$:

$$I(x)y' + I(x)P(x)y := \frac{d}{dx}(Iy) = Q(x)I(x)$$

$$I(x)y = \int dx\, Q(x)I(x)$$

$$\frac{d}{dx}(Iy) = I'(x)y + I(x)y' \implies I'(x) = P(x)I(x), \quad I(x) = e^{\int dx\, P(x)}$$

Solution:

$$y(x) = e^{-\int dx\, P(x)} \left(\int^x dx'\, Q(x')e^{\int dx'\, P(x')} + C \right)$$

All homogenous 1st order ODEs are separable.

First order ODEs

Example (RC circuit)

Charge $q(t)$ across an RC circuit with capacitance C, resistance R, and electromotive force $V(t)$:

$$\frac{dq(t)}{dt} + \frac{q(t)}{RC} = V(t)$$

where $q(t=0) = q_0$.

First order ODEs

Example (RC circuit)

Charge $q(t)$ across an RC circuit with capacitance C, resistance R, and electromotive force $V(t)$:

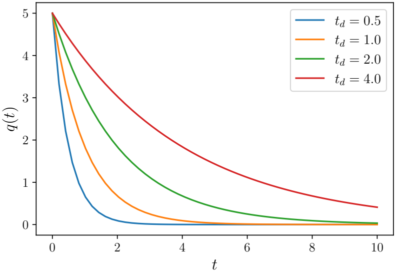
$$\frac{dq(t)}{dt} + \frac{q(t)}{RC} = V(t)$$

where $q(t=0) = q_0$.

Plot solution $q(t)$:

- Homogenous $q(t) = q_0 e^{-t/t_d}$ ($t_d = RC$):

```python
import matplotlib.pyplot as plt
import numpy as np

td_list = [0.5, 1.0, 2.0, 4.0]
tlist = np.linspace(0,10)

# Parameters
q0 = 5

fig, ax = plt.subplots()
for td in td_list:
    ax.plot(tlist, q0*np.exp(-tlist/td), label=r'$t_d={0}$'.format(td))
ax.set_xlabel(r'$t$')
ax.set_ylabel(r'$q(t)$')
ax.legend(loc=0)
```

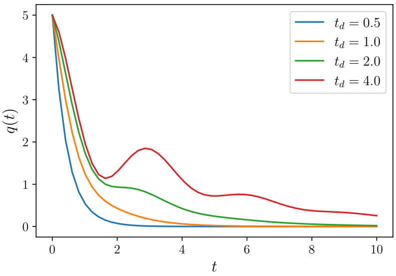


First order ODEs

Plot solution $q(t)$:

- Inhomogenous $q(t) = q_0 e^{-t/t_d} + \int_0^t dt'\, V(t') e^{-(t-t')/t_d}$:
- $V(t) = V_0 \sin(\omega t)$

```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import quad

td_list = [0.5, 1.0, 2.0, 4.0]
tlist = np.linspace(0,10)

# Parameters
q0 = 5
V0 = 4
om = 2

fig, ax = plt.subplots()
for td in td_list:
    qlist = []
    err_list = []
    integrand = lambda x, t: V0*np.sin(om*x)*np.exp((-t-x)/td)
    for t in tlist:
        q, _ = q0*np.exp(-t/(td)) + quad(integrand, t, 0, args=(t,))
        qlist.append(q)
    ax.plot(tlist, qlist, label=r'$t_d={0}$'.format(td))
ax.set_xlabel(r'$t$')
ax.set_ylabel(r'$q(t)$')
ax.legend(loc=0)
```



First order ODEs

Example (non-linear ODE)

Bernoulli equation.

For $n \neq 0, 1$:

$$y' + P(x)y = Q(x)y^n,$$

where $y(x = x_0) = y_0$.

First order ODEs

Example (non-linear ODE)

Bernoulli equation.

For $n \neq 0, 1$:

$$y' + P(x)y = Q(x)y^n,$$

where $y(x = x_0) = y_0$.

Linearlize using substitution $z = y^{1-n}$:

$$z'y^n = (1-n)y' \quad \text{(chain rule)}$$

$$\implies \boxed{z' + (1-n)P(x)z = (1-n)Q(x)}$$

First order ODEs

Example (non-linear ODE)

Bernoulli equation (linearized).

For $n \neq 0, 1$:

$$z' + (1-n)P(x)z = (1-n)Q(x)$$

where $y(x = x_0) = y_0$.

Examples $n = 2$, $\quad P(x) = Q(x) = \sin(x)$:

$$I(x) = e^{-\int dx \, \sin(x)} = e^{\cos(x)}$$

$$\begin{aligned} z(x) &= e^{-\cos(x)} \left(\int^x dx' \, \sin(x') e^{\cos(x')} + C \right) \\ &= -1 + Ce^{-\cos(x)} \end{aligned}$$

From $z = y^{1-n}$:

$$\implies \boxed{y^{-1} = -1 + Ce^{-\cos(x)}}$$

Euler method

First order ODE $(n = 1)$:

$$y' = F(x, y)$$

Iterative solution

Taylor expansion around x_k:

$$y(x_k + h) = y(x_k) + y'(x_k)h + \frac{y''(x_k)}{2!}h^2 + \ldots$$

Notation $x_{k+1} = x_k + h$, $\quad y(x_k) = y_k$:

$$\begin{aligned} k = 0: &\quad y_1 \approx y_0 + F(x_0, y_0)h \\ k = 1: &\quad y_2 \approx y_1 + F(x_1, y_1)h \\ \vdots &\qquad\quad \vdots \end{aligned}$$

Recurrence relation:

$$\boxed{\hat{y}_{k+1} = \hat{y}_k + F(x_k, \hat{y}_k)h}$$

Solution at some x_k obtained iteratively starting from x_0.

Euler method

First order ODE $(n = 1)$:

$$y' = F(x, y)$$

Iterative solution

Taylor expansion around x_k:

$$y(x_k + h) = y(x_k) + y'(x_k)h + \frac{y''(x_k)}{2!}h^2 + \ldots$$

Local truncation error:

$$\boxed{y_{k+1} - \hat{y}_{k+1} = \frac{y''(x_k)}{2!}h^2 + O(h^3)}$$

For $h \ll 1$, error proportional to h^2.

Recurrence relation:

$$\boxed{\hat{y}_{k+1} = \hat{y}_k + F(x_k, \hat{y}_k)h}$$

Solution at some x_k obtained iteratively starting from x_0.

Euler method

Example (Euler)

Numerically solve

$$y' = 2y^{3/2}, \quad y(0) = 1,$$

using the Euler method, starting from $x_0 = 0$ to $x_k = 0.5$, with varying step size h.

Analytical solution: $y = \dfrac{1}{(1-x)^2}$

Euler method

Example (Euler)

Numerically solve

$$y' = 2y^{3/2}, \quad y(0) = 1,$$

using the Euler method, starting from $x_0 = 0$ to $x_k = 0.5$, with varying step size h.

Analytical solution: $y = \dfrac{1}{(1-x)^2}$

```python
h = 0.1        # step size
x0 = 0         # initial x-value
xmax = 0.5     # final x-value
y0 = 1         # initial y-value

y_euler = [y0]

f = lambda x,y: 2*y**(3/2)

while x0<xmax:
    y1 = y0 + f(x0,y0)*h
    y_euler.append(y1)
    y0=y1
    x0 = round(x0+h,10)
```

Euler method

Example (Euler)

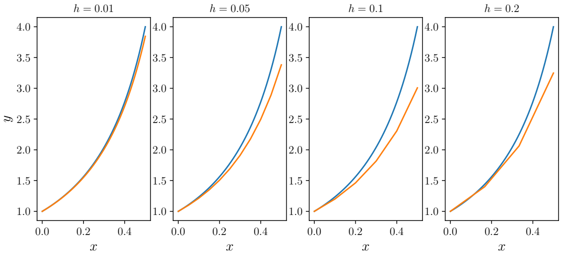
Numerically solve

$$y' = 2y^{3/2}, \quad y(0) = 1,$$

using the Euler method, starting from $x_0 = 0$ to $x_k = 0.5$, with varying step size h.

Analytical solution: $y = \dfrac{1}{(1-x)^2}$

```python
h = 0.1        # step size
x0 = 0         # initial x-value
xmax = 0.5     # final x-value
y0 = 1         # initial y-value

y_euler = [y0]

f = lambda x,y: 2*y**(3/2)

while x0<xmax:
    y1 = y0 + f(x0,y0)*h
    y_euler.append(y1)
    y0=y1
    x0 = round(x0+h,10)
```



Runge-Kutta method

First order ODE $(n = 1)$:

$$y' = F(x, y)$$

Iterative solution

Taylor expansion around x_k:

$$y(x_k + h) = y(x_k) + y'(x_k)h + \frac{y''(x_k)}{2!}h^2 + \dots$$

Write: $\hat{y}_{k+1} = \hat{y}_k + F(x_k, \hat{y}_k)\frac{h}{2} + F(x_k + \frac{1}{2}h, \hat{y}_k + \frac{1}{2}F(x_k, \hat{y}_k)h))\frac{h}{2}$

$$y_{k+1} = y_k + F(x_k, y_k)h + \left(\frac{\partial F}{\partial x} + F\frac{\partial F}{\partial y}\right)\frac{h^2}{2} + O(h^3)$$

Local truncation error:

$$\boxed{y_{k+1} - \hat{y}_{k+1} = \frac{1}{3!}y_k^{(3)}h^3 + O(h^4)}$$

For $h \ll 1$, error proportional to h^3.

Runge-Kutta method

Runge-Kutta method $(n = 1)$:

$$y' = F(x, y)$$

Recurrence relation (order h^2):

$$\begin{aligned}
a_1 &= F(x_k, \hat{y}_k)h \\
a_2 &= F(x_k + \tfrac{1}{2}h, \hat{y}_k + \tfrac{1}{2}a_1)h \\
\hat{y}_{k+1} &= \hat{y}_k + \frac{1}{2}(a_1 + a_2)
\end{aligned}$$

Possible to consider higher order schemes:

- To order h^4 (RK4):

$$\begin{aligned}
b_1 &= F(x_k, \hat{y}_k)h \\
b_2 &= F(x_k + \tfrac{1}{2}h, \hat{y}_k + \tfrac{1}{2}b_1)h \\
b_3 &= F(x_k + \tfrac{1}{2}h, \hat{y}_k + \tfrac{1}{2}b_2)h \\
b_4 &= F(x_k + h, \hat{y}_k + b_3)h \\
\hat{y}_{k+1} &= \hat{y}_k + \frac{1}{6}b_1 + \frac{1}{3}b_2 + \frac{1}{3}b_3 + \frac{1}{6}b_4
\end{aligned}$$

Runge-Kutta method

Example (RK2)

Numerically solve

$$y' = 2y^{3/2}, \quad y(0) = 1,$$

using 2nd order Runge-Kutta, starting from $x_0 = 0$ to $x_k = 0.5$, with varying step size h.

Analytical solution: $y = \dfrac{1}{(1-x)^2}$

Runge-Kutta method

Example (RK2)

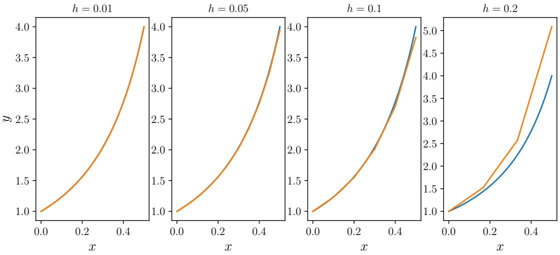
Numerically solve

$$y' = 2y^{3/2}, \quad y(0) = 1,$$

using 2nd order Runge-Kutta, starting from $x_0 = 0$ to $x_k = 0.5$, with varying step size h.

Analytical solution: $y = \dfrac{1}{(1-x)^2}$

```python
h = 0.1        # step size
x0 = 0         # initial x-value
xmax = 0.5     # final x-value
y0 = 1         # initial y-value

y_rk2 = [y0]

f = lambda x,y: 2*y**(3/2)

while x0<xmax:
    a1 = f(x0,y0)*h
    a2 = f(x0 + h, y0 + a1)*h
    y1 = y0 + 0.5 * (a1 + a2)
    y_rk2.append(y1)
    y0=y1
    x0 = round(x0+h,10)
```



Runge-Kutta method

Example (RK4)

Numerically solve

$$y' = y\sin^2(x), \quad y(0) = 1$$

using 4th order Runge-Kutta, starting from $x = 0$ to $x_k = 10$, with step size $h = 0.5$.

Analytical solution: $y = \exp\left(\frac{1}{2}\left[x - \frac{1}{2}\sin(2x)\right]\right)$

Runge-Kutta method

Example (RK4)

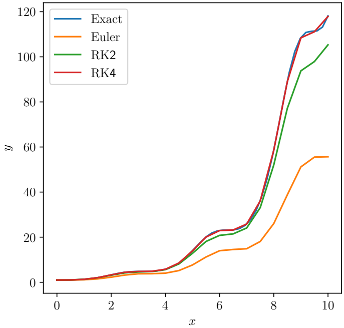
Numerically solve

$$y' = y\sin^2(x), \quad y(0) = 1$$

using 4th order Runge-Kutta, starting from $x = 0$ to $x_k = 10$, with step size $h = 0.5$.

Analytical solution: $y = \exp\left(\frac{1}{2}\left[x - \frac{1}{2}\sin(2x)\right]\right)$

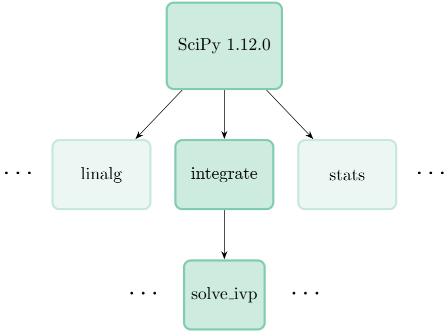
```python
import numpy as np

h = 0.5       # step size
x0 = 0        # initial x-value
xmax = 5      # final x-value
y0 = 0        # initial y-value

y_rk4 = [y0]

f = lambda x,y: np.sin(x)**2 * y

while x0<xmax:
    b1 = f(x0,y0)*h
    b2 = f(x0 + 0.5*h, y0 + 0.5*b1)*h
    b3 = f(x0 + 0.5*h, y0 + 0.5*b2)*h
    b4 = f(x0 + h, y0 + b3)*h
    y1 = y0 + 1/6 * (b1 + 2*b2 + 2*b3 + b4)
    y_rk4.append(y1)
    y0=y1
    x0 = round(x0+h,10)
```



SciPy ODE solver

SciPy: Python library for scientific computing built on NumPy.

- scipy.integrate contains routines for numerical integration and ODE solvers.



https://docs.scipy.org/doc/scipy/index.html

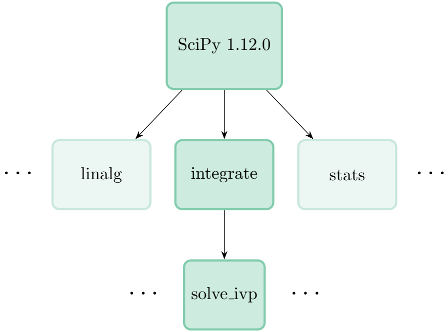
SciPy ODE solver

SciPy: Python library for scientific computing built on NumPy.

- scipy.integrate contains routines for numerical integration and ODE solvers.
- scipy.integrate.solve_ivp function for solving initial value problems:

$$\vec{y}\,' = \vec{F}(t, \vec{y}), \quad \vec{x}(t_0) = \vec{x}_0$$

- $\vec{y} \in \mathbb{R}^d$, d-dimensional real vector.
- $\vec{F}(\vec{y}, t)$ function of $\vec{y}$.



https://docs.scipy.org/doc/scipy/index.html

SciPy ODE solver

Example (solve_ivp)

Numerically solve

$$y' = y\sin^2(t), \quad y(0) = 1$$

using solver_ivp function of scipy.integrate, starting from $t_0 = 0$ up to $t_{\text{max}} = 10$.

Analytical solution: $y = \exp\left(\frac{1}{2}\left[t - \frac{1}{2}\sin(2t)\right]\right)$

SciPy ODE solver

Example (solve_ivp)

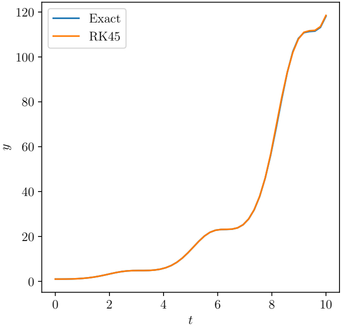
Numerically solve

$$y' = y\sin^2(t), \quad y(0) = 1$$

using solver_ivp function of scipy.integrate, starting from $t_0 = 0$ up to $t_{\text{max}} = 10$.

Analytical solution: $y = \exp\left(\frac{1}{2}\left[t - \frac{1}{2}\sin(2t)\right]\right)$

```python
import numpy as np
from scipy.integrate import solve_ivp

# Right-hand side of ODE
f = lambda t,y: np.sin(t)**2 * y

# t-values for which ODE is solved
t_eval = np.linspace(0, 10)

sol = solve_ivp(f, [t_eval[0],t_eval[-1]],
                y0=[1], method='RK45', t_eval=t_eval)

sol.t   # Returns t-values      shape=(len(t_eval),)
sol.y   # Returns y-values      shape=(len(y0),len(t_eval))

sol.y.reshape(sol.t.shape)   #  shape=(len(t_eval),)
```



SciPy ODE solver

Example (solve_ivp)

Numerically solve

$$y_1' = y_2$$
$$y_2' = -k_1 y_2 - k_2 \sin(y_1)$$

using solver_ivp function of scipy.integrate, starting from $t_0 = 0$ up to $t_{\text{max}} = 100$.

SciPy ODE solver

Example (solve_ivp)

Numerically solve

$$y_1' = y_2$$
$$y_2' = -k_1 y_2 - k_2 \sin(y_1)$$

using solver_ivp function of scipy.integrate, starting from $t_0 = 0$ up to $t_{\text{max}} = 100$.

Written compactly as:

$$\vec{y}' = \vec{F}(t, \vec{y}) : \qquad \vec{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \in \mathbb{R}^2, \qquad \vec{F}(t, \vec{y}) = \begin{pmatrix} y_2 \\ -k_1 y_2 - k_2 \sin(y_1) \end{pmatrix}$$

SciPy ODE solver

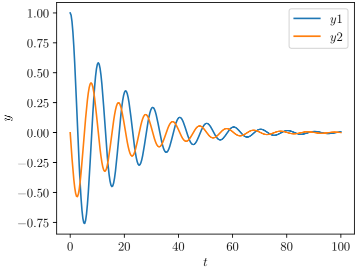
Example (solve_ivp)

Numerically solve

$$y_1' = y_2$$
$$y_2' = -k_1 y_2 - k_2 \sin(y_1)$$

using solver_ivp function of scipy.integrate, starting from $t_0 = 0$ up to $t_{\text{max}} = 100$.

```python
import numpy as np
from scipy.integrate import solve_ivp

k1 = 0.1
k2 = 0.4

# Right-hand side of ODE
def f(t,y):
    # y is 2d-array
    dydt = [y[1], -k1*y[1] - k2*np.sin(y[0])]
    return dydt

# t-values for which ODE is solved
t_eval = np.linspace(0, 100, 300)

sol = solve_ivp(f, [t_eval[0],t_eval[-1]],
                y0=[1,0], method='RK45', t_eval=t_eval)

y1 = sol.y[0,:]
y2 = sol.y[1,:]
```



SciPy ODE solver

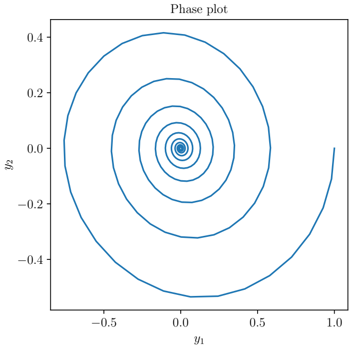
Example (solve_ivp)

Numerically solve

$$
\begin{aligned}
y_1' &= y_2 \\
y_2' &= -k_1 y_2 - k_2 \sin(y_1)
\end{aligned}
$$

using solver_ivp function of scipy.integrate, starting from $t_0 = 0$ up to $t_{\text{max}} = 100$.

```python
import numpy as np
from scipy.integrate import solve_ivp

k1 = 0.1
k2 = 0.4

# Right-hand side of ODE
def f(t,y):
    # y is 2d-array
    dydt = [y[1], -k1*y[1] - k2*np.sin(y[0])]
    return dydt

# t-values for which ODE is solved
t_eval = np.linspace(0, 100, 300)

sol = solve_ivp(f, [t_eval[0],t_eval[-1]],
                y0=[1,0], method='RK45', t_eval=t_eval)

y1 = sol.y[0,:]
y2 = sol.y[1,:]
```



Solution to higher order ODEs

Linear nth order ODE:

$$F(x, y, y' ... y^{(n)}) := a_n(x)\frac{d^n y}{dx^n} + ... + a_1(x)\frac{dy}{dx} + a_0(x)y + b(x) = 0.$$

Unique solution found for n boundary conditions.

Homogeneous equation $b(x) = 0$

Homogenous nth order ODE:

$$F(x, y, y' ... y^{(n)}) := a_n(x)\frac{d^n y}{dx^n} + ... + a_1(x)\frac{dy}{dx} + a_0(x)y = 0,$$

Since linear, general solution to ODE has the form ($c_i \in \mathbb{R}$ $i = 1, ..., n$):

$$y(x) = c_1 y_1(x) + c_2 y_2(x) + ... + c_n y_n(x),$$

where $F(x, y_i, y_i' ... y_i^{(n)}) = 0$, and $y_i(x)$ are linearly independent.

Solution to higher order ODEs

Constant coefficients:

$$a_n \frac{d^n y}{dx^n} + ... + a_1 \frac{dy}{dx} + a_0 y = 0.$$

Method for solution, try ansatz $y = Ae^{\lambda x}$

Auxiliary equation

Polynomial nth order

$$a_n \lambda^n + ... + a_1 \lambda + a_0 = 0.$$

Has n roots $\lambda_1, ..., \lambda_n$.

Solution to higher order ODEs

Constant coefficients:

$$a_n \frac{d^n y}{dx^n} + \ldots + a_1 \frac{dy}{dx} + a_0 y = 0.$$

Method for solution, try ansatz $y = Ae^{\lambda x}$

Auxiliary equation

Polynomial nth order

$$a_n \lambda^n + \ldots + a_1 \lambda + a_0 = 0.$$

Has n roots $\lambda_1, \ldots, \lambda_n$.

- λ_i all real: $y = c_1 e^{\lambda_1 x} + \ldots + c_n e^{\lambda_n x}$
- λ_i real and complex conjugate pairs:

$$c_1 e^{\alpha + i\beta} + c_2 e^{\alpha - i\beta} = (d_1 \cos(\beta x) + d_2 \sin(\beta x)) e^{\alpha t}$$

- λ_i degenerate (e.g., $\lambda_1 = \lambda_2$)

Solution to higher order ODEs

Example (damped harmonic oscillator)

Solve

$$\ddot{x}(t) + 2\gamma\dot{x}(t) + \omega_0^2 x(t) = 0$$

with initial conditions $x(0) = x_0$, $\dot{x}(0) = v_0$.

Solution to higher order ODEs

Example (damped harmonic oscillator)

Solve

$$\ddot{x}(t) + 2\gamma\dot{x}(t) + \omega_0^2 x(t) = 0$$

with initial conditions $x(0) = x_0$, $\dot{x}(0) = v_0$.

Substitute in $x(t) = Ae^{\lambda t}$:

$$\lambda^2 + 2\gamma\lambda + \omega_0^2 = 0$$

$$\implies \lambda_{1,2} = -\gamma \pm i\sqrt{\omega_0^2 - \gamma^2} := -\gamma \pm i\Omega$$

Solution to higher order ODEs

Example (damped harmonic oscillator)

Solve

$$\ddot{x}(t) + 2\gamma\dot{x}(t) + \omega_0^2 x(t) = 0$$

with initial conditions $x(0) = x_0$, $\dot{x}(0) = v_0$.

Substitute in $x(t) = Ae^{\lambda t}$:

$$\lambda^2 + 2\gamma\lambda + \omega_0^2 = 0$$

$$\implies \lambda_{1,2} = -\gamma \pm i\sqrt{\omega_0^2 - \gamma^2} := -\gamma \pm i\Omega$$

$$\begin{aligned} x(t) &= c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} \\ &= \underbrace{(c_1 + c_2)}_{=d_1} \cos\Omega t\, e^{-\gamma t} + \underbrace{i(c_1 - c_2)}_{=d_2} \sin\Omega t\, e^{-\gamma t} \end{aligned}$$

Solution to higher order ODEs

Example (damped harmonic oscillator)

Solve

$$\ddot{x}(t) + 2\gamma\dot{x}(t) + \omega_0^2 x(t) = 0$$

with initial conditions $x(0) = x_0$, $\dot{x}(0) = v_0$.

General solution:

$$x(t) = (d_1 \cos \Omega t + d_2 \sin \Omega t)e^{-\gamma t}$$

Determine $d_{1,2}$ from initial conditions:

$$x(0) = x_0 = d_1$$

$$\dot{x}(0) = v_0 = \Omega d_2 - \gamma x_0 \implies d_2 = \frac{v_0 - \gamma x_0}{\Omega}$$

Solution to higher order ODEs

Example (damped harmonic oscillator)

Solve

$$\ddot{x}(t) + 2\gamma\dot{x}(t) + \omega_0^2 x(t) = 0$$

with initial conditions $x(0) = x_0$, $\dot{x}(0) = v_0$.

Solution:

$$x(t) = \left[x_0 \cos\Omega t + \left(\frac{v_0 - \gamma x_0}{\Omega}\right)\sin\Omega t\right] e^{-\gamma t}$$

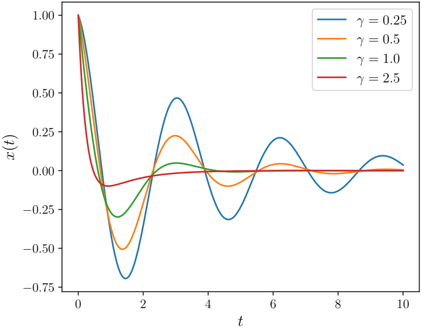
Regimes of behavior:

- Underdamped $\omega_0 > \gamma$, $\Omega \in \mathbb{R}$, damped coherent oscillations
- Overdamped $\omega_0 < \gamma$, $\Omega = i\tilde{\Omega}$, incoherent damping
- Critical damping $\omega_0 = \gamma$, $\lambda_1 = \lambda_2$

Solution to higher order ODEs

Plots of solution:

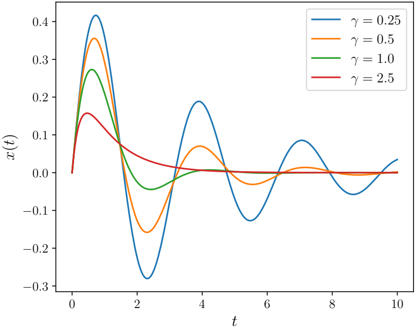
```python
import matplotlib.pyplot as plt
import numpy as np

tlist = np.linspace(0,10,250)
gmlist = [0.25, 0.5, 1.0, 2.5]
om0 = 2
x0 = 1
v0 = 0

x = lambda gm, Om, t: (x0*np.cos(Om*t) + ((v0-gm*x0)/Om)*np.sin(Om*t))*np.exp(-gm*t)

fig, ax = plt.subplots(figsize=(6,5))

for gm in gmlist:
    if om0>gm:
        Om = np.sqrt(om0**2 - gm**2)
    else:
        Om = 1.0j*np.sqrt(gm**2 - om0**2)

    ax.plot(tlist, x(gm,Om,tlist), label=r'$\gamma={0}$'.format(gm))

ax.set_xlabel(r'$t$')
ax.set_ylabel(r'$x(t)$')
ax.legend(loc=0)
```



Solution to higher order ODEs

Plots of solution:

```python
import matplotlib.pyplot as plt
import numpy as np

tlist = np.linspace(0,10,250)
gmlist = [0.25, 0.5, 1.0, 2.5]
om0 = 2
x0 = 0
v0 = 1

x = lambda gm, Om, t: (x0*np.cos(Om*t) + ((v0-gm*x0)/Om)*np.sin(Om*t))*np.exp(-gm*t)

fig, ax = plt.subplots(figsize=(6,5))

for gm in gmlist:
    if om0>gm:
        Om = np.sqrt(om0**2 - gm**2)
    else:
        Om = 1.0j*np.sqrt(gm**2 - om0**2)

    ax.plot(tlist, x(gm,Om,tlist), label=r'$\gamma={0}$'.format(gm))

ax.set_xlabel(r'$t$')
ax.set_ylabel(r'$x(t)$')
ax.legend(loc=0)
```



Solution to higher order ODEs

Constant coefficients:

$$a_n \frac{d^n y}{dx^n} + ... + a_1 \frac{dy}{dx} + a_0 y = 0.$$

Method for solution, try ansatz $y = Ae^{\lambda x}$:

Auxiliary equation

Polyonmial nth order

$$a_n \lambda^n + ... + a_1 \lambda + a_0 = 0.$$

Has n roots $\lambda_1, ..., \lambda_n$.

If k-fold degenerate ($\lambda_1 = \lambda_2 = ... = \lambda_k$):

$$y = (c_1 + c_2 x + c_2 x^2 + ... c_k x^{k-1}) e^{\lambda_1 x} + c_{k+1} e^{\lambda_{k+1} x} + ... + c_n e^{\lambda_n x}$$

Solution to higher order ODEs

Constant coefficients:

$$a_n \frac{d^n y}{dx^n} + ... + a_1 \frac{dy}{dx} + a_0 y = 0.$$

Method for solution, try ansatz $y = Ae^{\lambda x}$:

Auxiliary equation

Polyonmial nth order

$$a_n \lambda^n + ... + a_1 \lambda + a_0 = 0.$$

Has n roots $\lambda_1, ..., \lambda_n$.

If k-fold degenerate ($\lambda_1 = \lambda_2 = ... = \lambda_k$):

$$y = (c_1 + c_2 x + c_2 x^2 + ...c_k x^{k-1})e^{\lambda_1 x} + c_{k+1} e^{\lambda_{k+1} x} + ... + c_n e^{\lambda_n x}$$

In case where $n = 2$ and $\lambda_1 = \lambda_2$:

$$y = (c_1 + c_2 x)e^{\lambda_1 x}$$

Solution to higher order ODEs

Example (damped harmonic oscillator)

Solve

$$\ddot{x}(t) + 2\gamma\dot{x}(t) + \omega_0^2 x(t) = 0$$

with initial conditions $x(0) = x_0$, $\dot{x}(0) = v_0$.

Critical damping $\omega_0 = \gamma$:

$$\lambda_{1,2} = -\gamma \pm i\sqrt{\omega_0^2 - \gamma^2} = -\gamma$$

Solution to higher order ODEs

Example (damped harmonic oscillator)

Solve

$$\ddot{x}(t) + 2\gamma\dot{x}(t) + \omega_0^2 x(t) = 0$$

with initial conditions $x(0) = x_0$, $\dot{x}(0) = v_0$.

Critical damping $\omega_0 = \gamma$:

$$\lambda_{1,2} = -\gamma \pm i\sqrt{\omega_0^2 - \gamma^2} = -\gamma$$

Solution:

$$x(t) = (c_1 + c_2 t)e^{-\gamma t}$$

Determine c_1 and c_2 from initial conditions:

$$x(t=0) = x_0 = c_1$$

$$\dot{x}(t) = c_2 e^{-\gamma t} - \gamma(x_0 + c_2 t)e^{-\gamma t} \implies v_0 = c_2 - \gamma x_0$$

Solution to higher order ODEs

Example (damped harmonic oscillator)

Solve

$$\ddot{x}(t) + 2\gamma\dot{x}(t) + \omega_0^2 x(t) = 0$$

with initial conditions $x(0) = x_0$, $\dot{x}(0) = v_0$.

Critical damping $\omega_0 = \gamma$:

$$\lambda_{1,2} = -\gamma \pm i\sqrt{\omega_0^2 - \gamma^2} = -\gamma$$

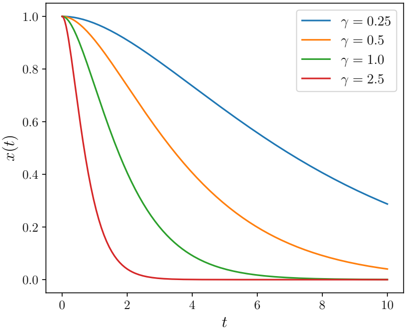
Solution:

$$x(t) = (x_0 + (v_0 + \gamma x_0)t)e^{-\gamma t}$$

Solution to higher order ODEs

Plots of solution:

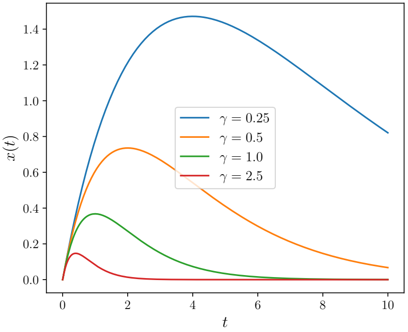
```
import matplotlib.pyplot as plt
import numpy as np

tlist = np.linspace(0,10,250)
gmlist = [0.25, 0.5, 1.0, 2.5]
x0 = 1
v0 = 0

x = lambda gm, t: (x0+(v0 + gm*x0)*t)*np.exp(-gm
      *t)

fig, ax = plt.subplots(figsize=(6,5))

for gm in gmlist:
    ax.plot(tlist, x(gm,tlist), label=r'$\gamma
      ={0}$'.format(gm))

ax.set_xlabel(r'$t$')
ax.set_ylabel(r'$x(t)$')
ax.legend(loc=0)
```



Solution to higher order ODEs

Plots of solution:

```python
import matplotlib.pyplot as plt
import numpy as np

tlist = np.linspace(0,10,250)
gmlist = [0.25, 0.5, 1.0, 2.5]
x0 = 0
v0 = 1

x = lambda gm, t: (x0+(v0 + gm*x0)*t)*np.exp(-gm*t)

fig, ax = plt.subplots(figsize=(6,5))

for gm in gmlist:
    ax.plot(tlist, x(gm,tlist), label=r'$\gamma={0}$'.format(gm))

ax.set_xlabel(r'$t$')
ax.set_ylabel(r'$x(t)$')
ax.legend(loc=0)
```



Solution to higher order ODEs

Linear nth order ODE:

$$F(x, y, y'...y^{(n)}) := a_n(x)\frac{d^n y}{dx^n} + ... + a_1(x)\frac{dy}{dx} + a_0(x)y + b(x) = 0.$$

Unique solution found for n boundary conditions.

General solution

Since linear, general solution to ODE has the form

$$y(x) = y_c(x) + y_p(x)$$

where $y_c(x)$ is the solution to the complimentary equation with $b(x) = 0$.

The complimentary solution $y_c(x)$ and particular integral $y_p(x)$ are linearly independent.

Solution to higher order ODEs

Constant coefficients:

$$a_n \frac{d^n y}{dx^n} + ... + a_1 \frac{dy}{dx} + a_0 y + b(x) = 0.$$

Method for solution, for complimentary equation $b(x) = 0$ try ansatz $y = Ae^{\lambda x}$.

No general method of finding $y_p(x)$:

Solution to higher order ODEs

Constant coefficients:

$$a_n \frac{d^n y}{dx^n} + \ldots + a_1 \frac{dy}{dx} + a_0 y + b(x) = 0.$$

Method for solution, for complimentary equation $b(x) = 0$ try ansatz $y = Ae^{\lambda x}$.

No general method of finding $y_p(x)$:

- If $b(x) = e^{rx}$, try
$$y_p(x) = be^{rx}$$
- If $b(x) = a_1 \sin rx + a_2 \cos rx$ (a_1 or a_2 may be zero), try
$$y_p(x) = b_1 \sin rx + b_2 \cos rx$$
- If $b(x) = a_0 + a_1 x + \ldots + a_N x^N$, (some a_n may be zero) try
$$y_p(x) = b_0 + b_1 x + \ldots + b_N x^N$$

Thank you