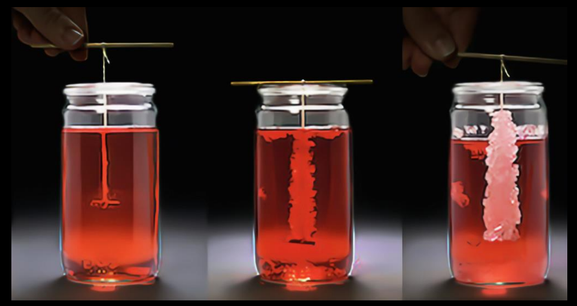
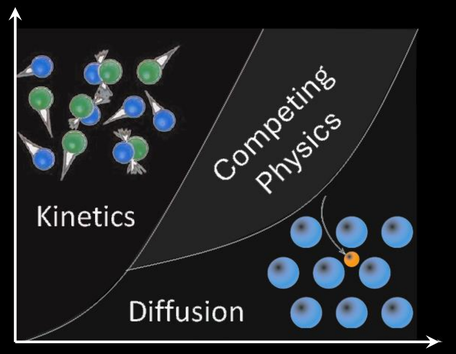
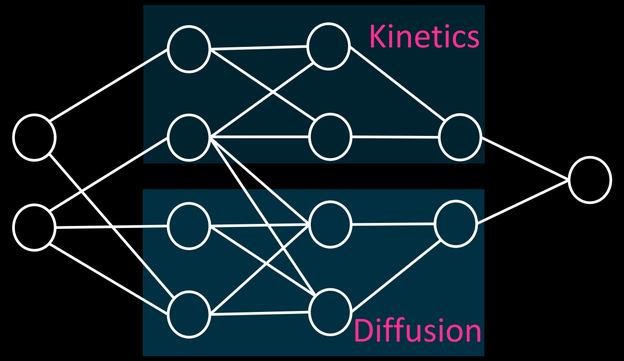
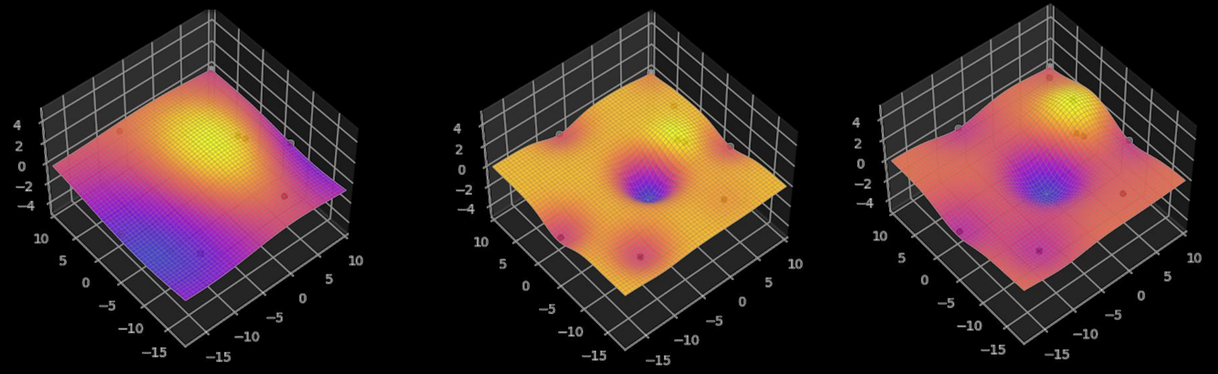
Physics Informed Machine Learning Chemical Reactions









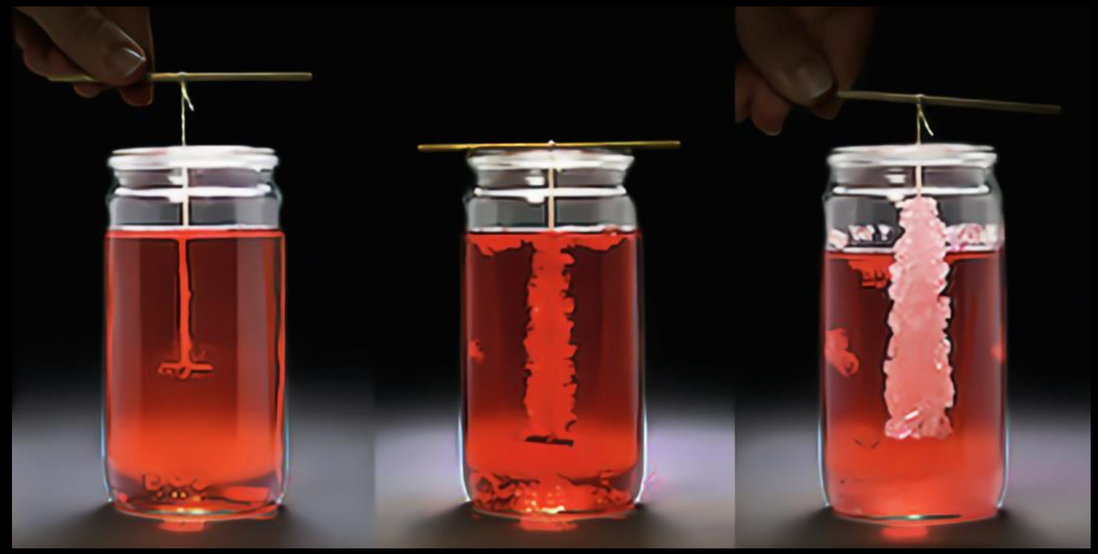
Navid Zobeiry, Associate Professor
Email: navidz@uw.edu
Materials Science & Engineering
https://composites.uw.edu/AI/

UNIVERSITY of WASHINGTON

Background: Chemical Reaction Kinetics

- **Reaction kinetics studies the rates of chemical reactions, helping predict reaction times and optimize conditions.**

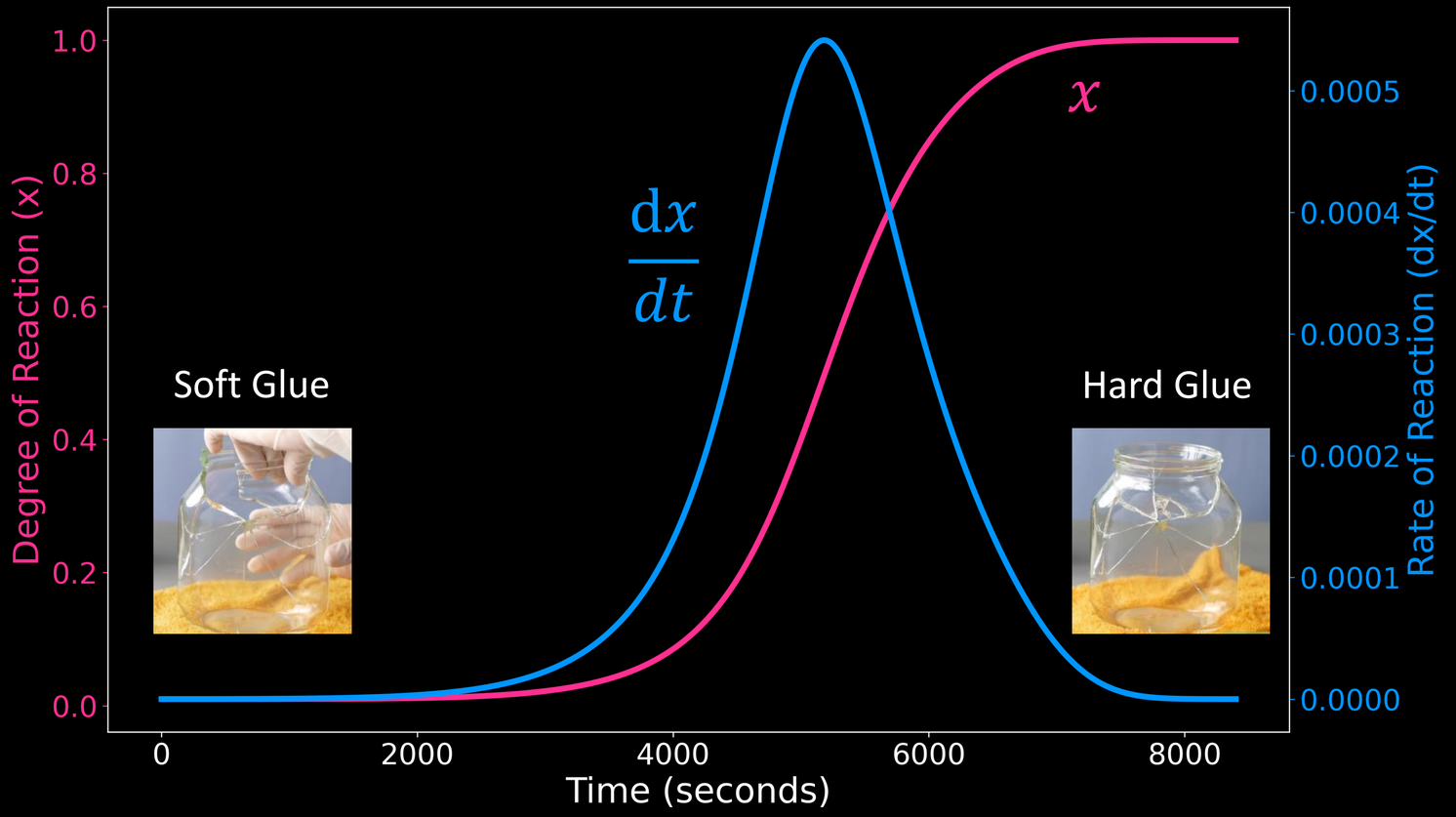
Crystallization



Diffusion

Background: Rate of Reaction

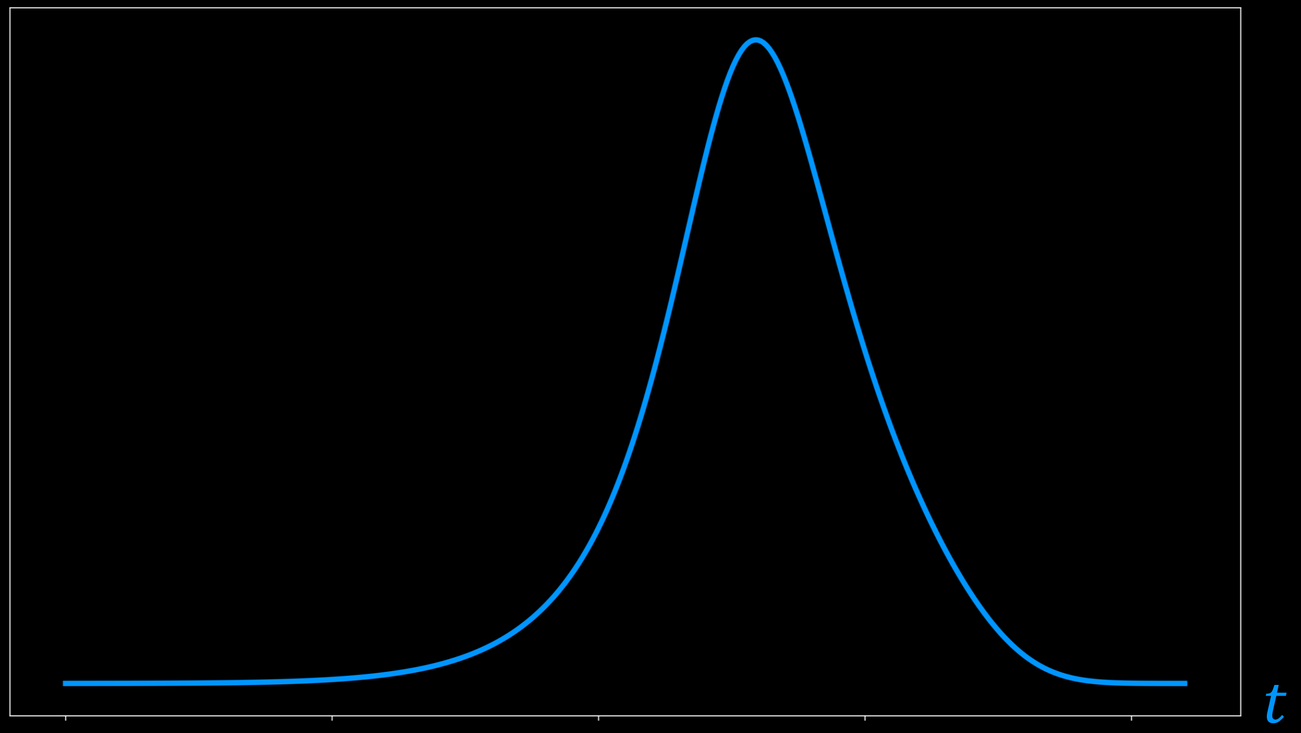
- To monitor a reaction's progress, we use a mathematical term, x, ranging from 0 to 1 during the reaction time.



Background: Rate of Reaction

- The reaction rate often depends on both temperature and the degree of reaction. The temperature dependency is usually expressed as an exponential term. The degree of reaction's effect is typically unknown.

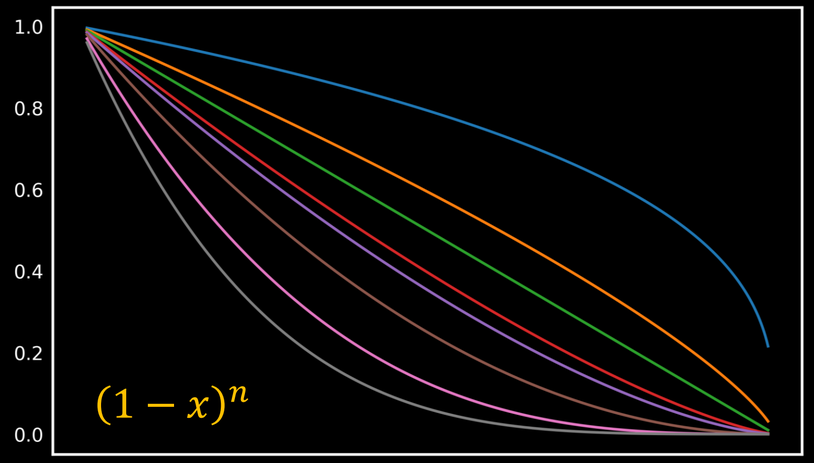
$$\frac{\mathrm{d}x}{dt} = Ae^{-\frac{E}{RT}} \times f(x)$$



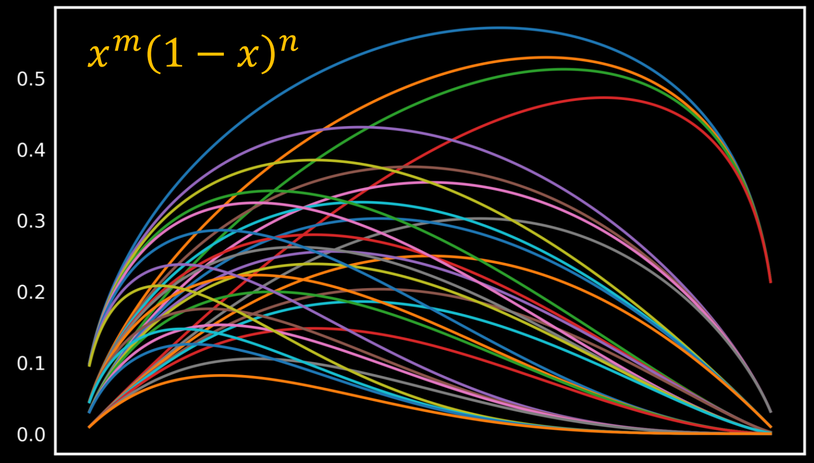
Background: Rate of Reaction

> $f(x)$ can have a wide range of shapes and orders:

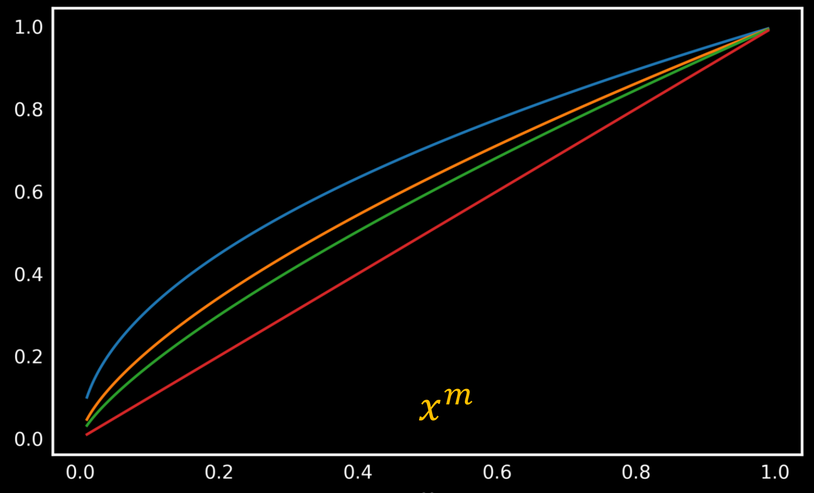
deceleratory reaction



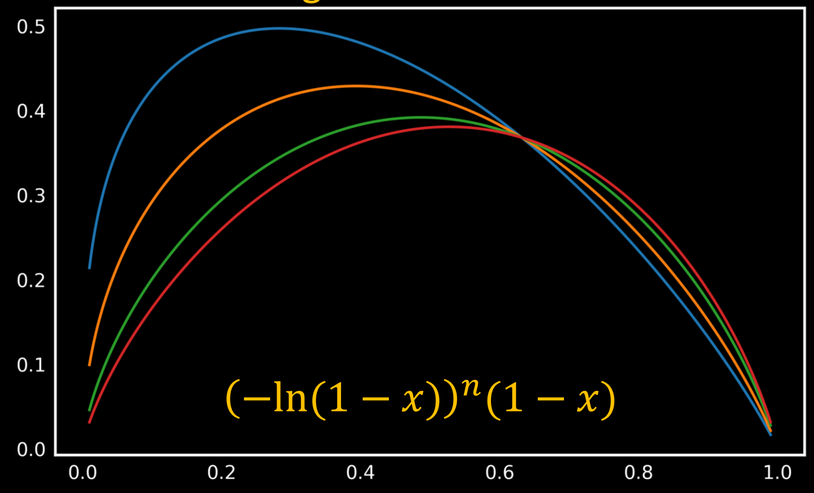
autocatalytic reaction



acceleratory reaction

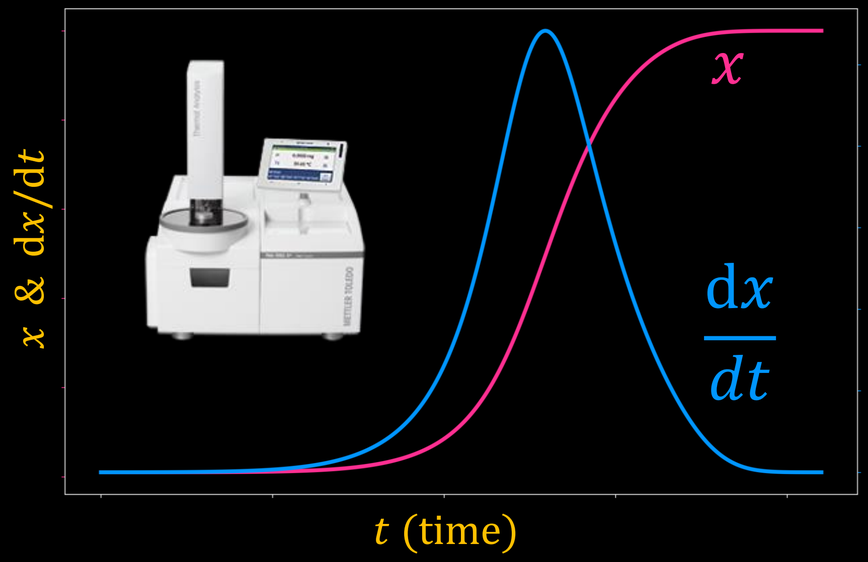


Sigmoid rate reaction



Background: Discovering Reaction Type

- Using experimental data for x and $\mathrm{d}x/dt$, we try to find $f(x)$ through data fitting and trial-and-error.
- Choosing the correct reaction type and determining the reaction orders (like m and n) are crucial.



$$\frac{dx}{dt} = \sum A_i e^{\frac{-E_i}{RT}} f_i(x) \quad \rightarrow f_i(x) = ?$$

$$f_i(x) = x^m(1-x)^n \rightarrow m \,\&\, n = ?$$

Example: Boeing 787 Composite Material

- The epoxy used in carbon-fiber composites for the Boeing 787 has a complex reaction mechanism involving three different reactions:

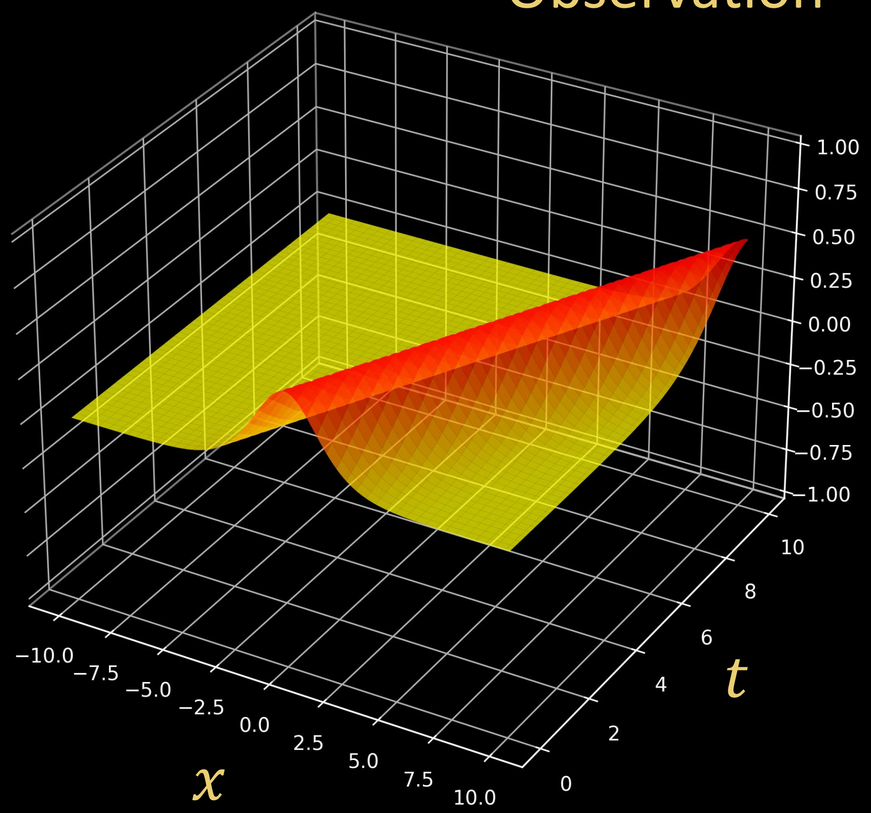
$$\frac{dx}{dt} = k_1(x)^{m_1}(1-x)^{n_1} + k_2(x)^{m_2}(1-x)^{n_2} + k_3(x)^{m_3}(1-x)^{n_3}$$

Dykeman, D. (2008). Minimizing uncertainty in cure modeling for composites manufacturing (T). University of British Columbia. Retrieved from https://open.library.ubc.ca/collections/ubctheses/24/items/1.0066334

Background: Sparse Identification of Nonlinear Dynamics (SINDy)

> **SINDy is used to discover physical laws and governing equations from time-series data:**

Observation



u

Governing Equation

$$\frac{\partial u}{\partial t} = -6u\frac{\partial u}{\partial x} - \frac{\partial^3 u}{\partial x^3}$$

Brunton, S., Proctor, J., Kutz, N. ”Discovering governing equations from data: Sparse identification of nonlinear dynamic systems.” arXiv:1509.03580v1. (2015).

Simple Introduction to SINDy

- **Step 1: Collect time-series data or spatial data:**

$$u(x,t)$$

- **Step 2: Numerically compute a library of functions:**

$$\Theta(u) = [1, u, \frac{\partial u}{\partial x}, u\frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^3 u}{\partial x^3} \dots]$$

- **Step 3: Only a few of the functions are active. Use sparse regression to identify which terms are active:**

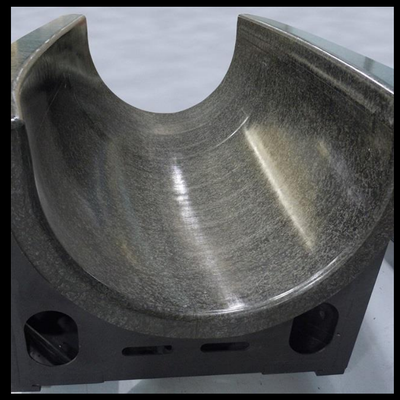
$$\frac{\partial u}{\partial t} = \Theta[\xi_1, \xi_2, \xi_3, \xi_4 \dots] \rightarrow \text{LASSO} \rightarrow \frac{\partial u}{\partial t} = \Theta[0,0,0,-6,0,-1 \dots]$$

$$\rightarrow \frac{\partial u}{\partial t} = -6u\frac{\partial u}{\partial x} - \frac{\partial^3 u}{\partial x^3}$$

Discovering Chemical Reactions with SINDy

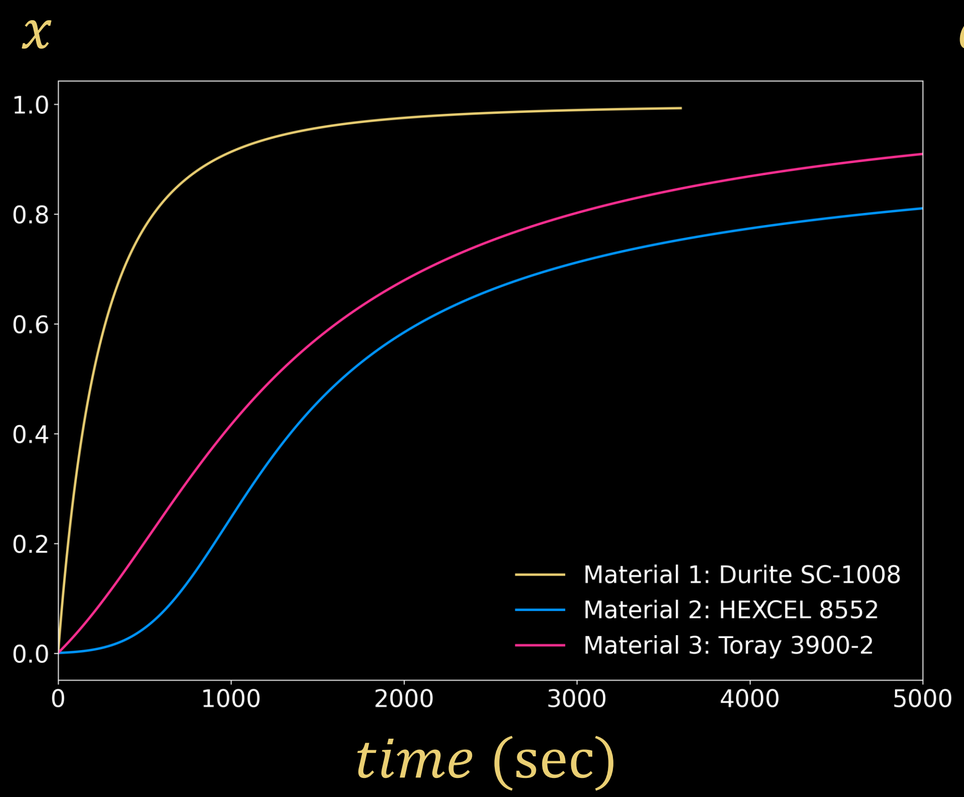
- **Three commercial materials are considered:**

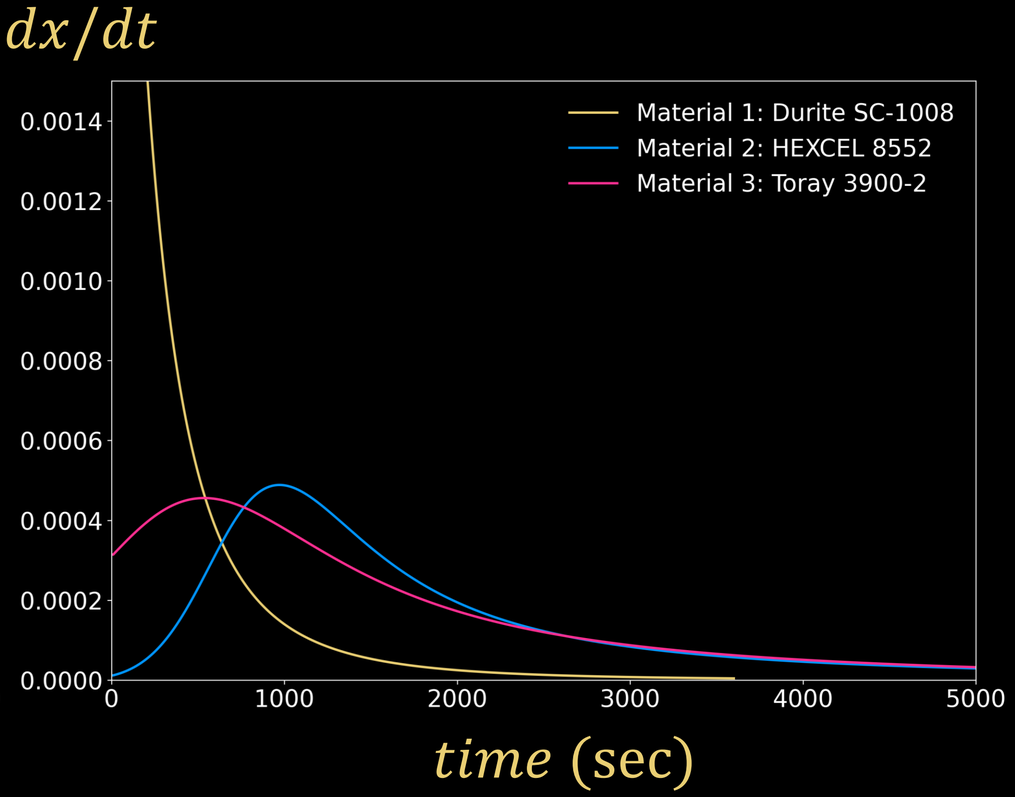
	Material 1	Material 2	Material 3
Name	Durite SC-1008	HEXCEL 8552	Toray 3900-2
Type	phenolic adhesive	epoxy adhesive	epoxy adhesive
Temperature	160 ℃	180 ℃	180 ℃



Discovering Chemical Reactions with SINDy

> All three materials are tested, and their degree of reaction as a function of time is provided:





SINDy/data_1.csv

SINDy/data_2.csv

SINDy/data_3.csv

SINDy Python Implementation

> **First, based on the literature, we created a library of 60 candidate functions:**

$$f(x) = (1-x)^n, n \in \left[\frac{1}{3}, \frac{3}{4}, 1, \frac{4}{3}, \frac{3}{2}, 2, 3, 4\right]$$

$$f(x) = x^m, m \in \left[-\frac{1}{2}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, 1\right]$$

$$f(x) = x^m(1-x)^n, m \in \left[-\frac{1}{2}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, 1\right], n \in \left[\frac{1}{3}, \frac{3}{4}, 1, \frac{4}{3}, \frac{3}{2}, 2, 3, 4\right]$$

$$f(x) = (-\ln(1-x))^n(1-x), n \in \left[-3, -2, -1, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \right]$$

```python
def generate_functions_and_names():
    # Chemical reaction (deceleration)
    def gen_func_1(n):
        return lambda x: (1 - x)**n, lambda _: f'(1-x)^{n}'

    # Chemical reaction (acceleration and deceleration autocatalytic)
    def gen_func_2(n, m):
        return lambda x: (x**m) * (1 - x)**n, lambda _: f'x^{m} * (1-x)^{n}'

    # Nucleation or Chemical reaction (acceleration)
    def gen_func_3(m):
        return lambda x: (x**m), lambda _: f'x^{m}'

    # Sigmoid rate reaction, random nucleation and growth
    def gen_func_4(n):
        return lambda x: (-np.log(1 - x))**n * (1 - x), lambda _: f'(-ln(1-x))^{n} * (1-x)'

    functions = []
    names = []

    # Chemical reaction (deceleration)
    for n in [1/3, 3/4, 1, 4/3, 3/2, 2, 3, 4]:
        func, name = gen_func_1(round(n, 3))
        functions.append(func)
        names.append(name)

    # Chemical reaction (acceleration and deceleration autocatalytic)
    for n in [1/3, 3/4, 1, 4/3, 3/2, 2, 3, 4]:
        for m in [-1/2, 1/2, 2/3, 3/4, 1]:
            func, name = gen_func_2(round(n, 3), round(m, 3))
            functions.append(func)
            names.append(name)

    # Chemical reaction (acceleration)
    for m in [-1/2, 1/2, 2/3, 3/4, 1]:
        func, name = gen_func_3(round(m, 3))
        functions.append(func)
        names.append(name)

    # Sigmoid rate reaction (random nucleation and growth)
    for n in [-3, -2, -1, 1/3, 1/2, 2/3, 3/4]:
        func, name = gen_func_4(round(n, 3))
        functions.append(func)
        names.append(name)
```

SINDy Python Implementation

Using “pysindy” library

```python
# Import Libraries
import pandas as pd
import numpy as np
import itertools
import pysindy as ps
from sklearn.linear_model import Lasso
import matplotlib.pyplot as plt
```

Load and prepare data

```python
# Prepare data for SINDy
df = pd.read_csv('data_'+str(option)+'.csv')

xdot_train_multi = [df[['xdot']].values]
x_train_multi = [df[['x']].values]
t_train_multi = [df['t'].values]
u_train_multi = [df[['T']].values]
```

Define LASSO optimizer, fit the model, and report

```python
# SINDy Model using Lasso with positive constraint
optimizer = Lasso(alpha=1.5E-7, max_iter=10000, fit_intercept=False, positive=True)
model = ps.SINDy(optimizer=optimizer, feature_library=feature_lib)
model.fit(x_train_multi, t=t_train_multi, multiple_trajectories=True)

# Get model coefficients and score
coefficients = model.coefficients()
feature_names = model.get_feature_names()
score = model.score(x_train_multi, t=t_train_multi, multiple_trajectories=True)
```

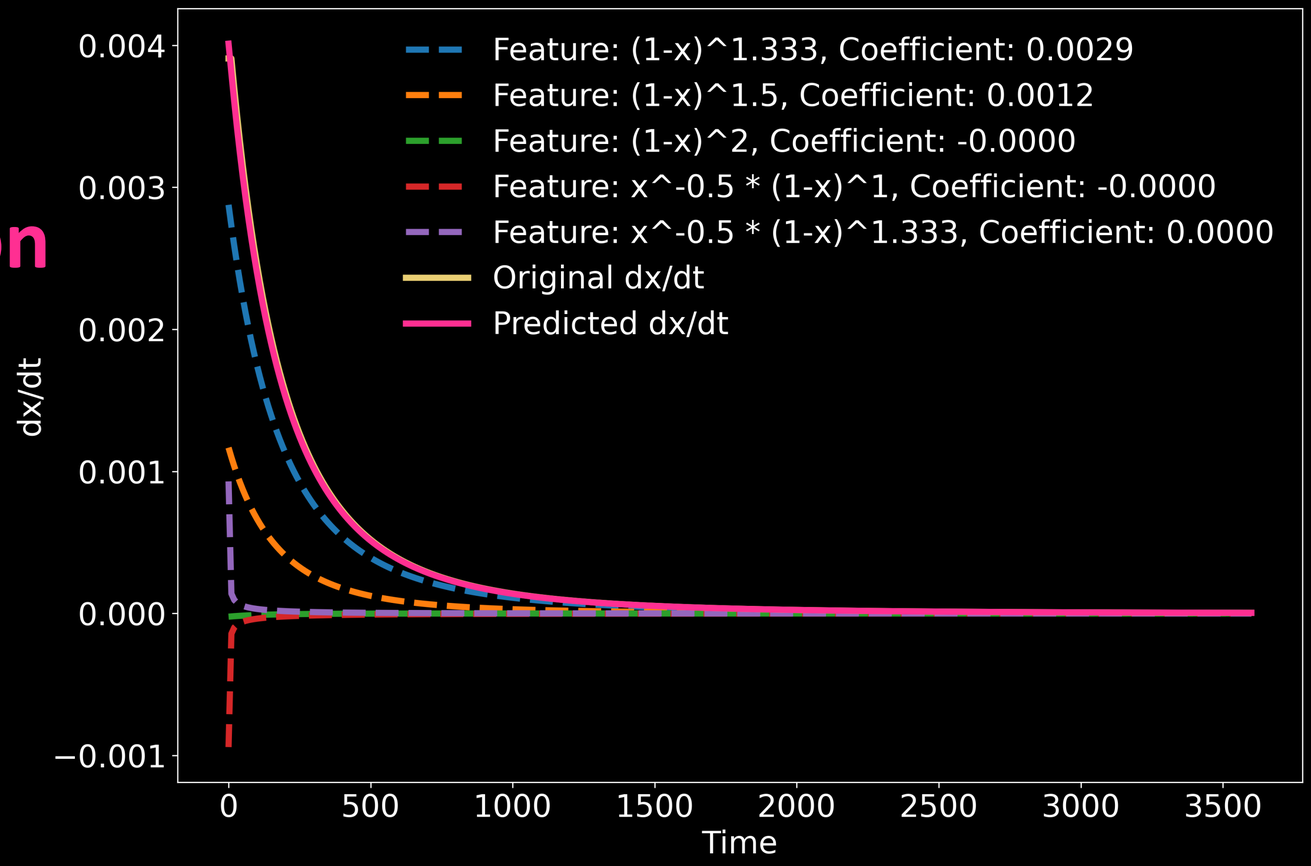
Example 1: Durite SC-1008

> SINDy's fitted model

$$\frac{\mathrm{d}x}{dt} \approx c_1(1-x)^{1.33} + c_2(1-x)^{1.50}$$

> Hidden Solution

$$\frac{\mathrm{d}x}{dt} = a(1-x)^{1.37}$$



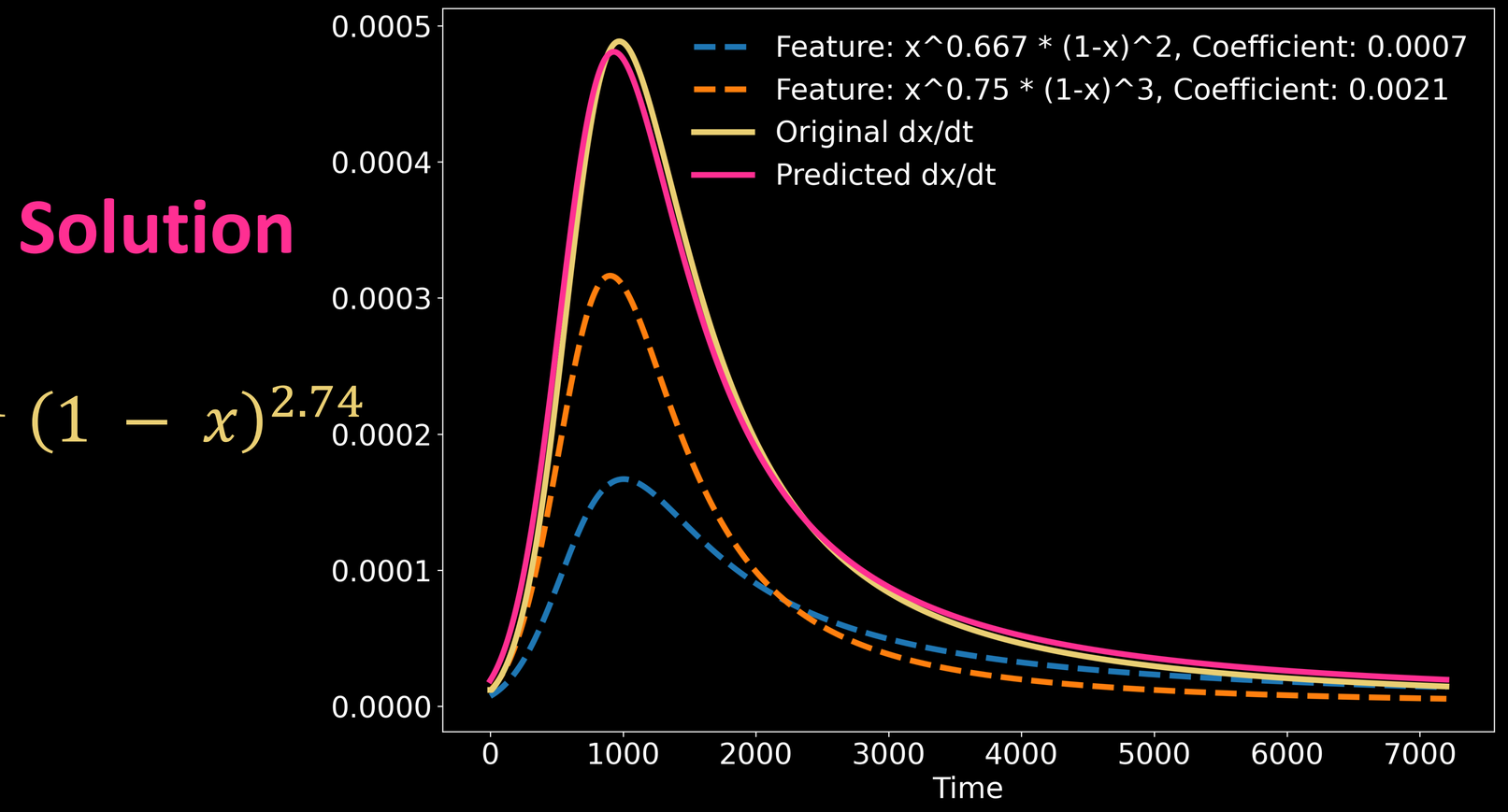
Example 2: HEXCEL 8552

> SINDy's fitted model:

$$\frac{\mathrm{d}x}{dt} \approx c_1 x^{0.67}(1-x)^2 + c_2 x^{0.75}(1-x)^3$$

> Hidden Solution

$$\frac{\mathrm{d}x}{dt} = a x^{0.81}\,(1\,-\,x)^{2.74}$$



Example 3: Toray 3900-2

> SINDy's fitted model

$$\frac{\mathrm{d}x}{dt} \approx c_1(1-x)^{1.33} + [c_2 x^{0.5}(1-x)^{1.5} + c_3 x^{0.5}(1-x)^2]$$

> Hidden Solution

$$\frac{\mathrm{d}x}{dt} = a_1(1-x) + a_2 x(1-x)^{2.5}$$