



Optimization with Differential Algebraic Equations:

DAE Discretization and Pyomo DAE

Assoc. Prof. Karl Ezra Pilario, Ph.D.

Process Systems Engineering Laboratory
Department of Chemical Engineering
University of the Philippines Diliman

Outline

- Review: Differential Equations
- Optimization with DAEs
- Orthogonal Collocation

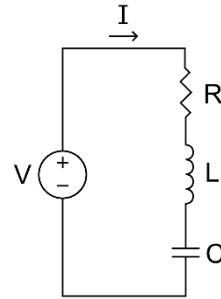
Review: Differential Equations

Differential Equations

- A *mathematical model* of a physical system.
- It is written in terms of the **derivatives** of some unknown function/s.
- To simulate the model, we need to find the “solution” to the differential equation, i.e. function/s that satisfy/ies the model.
- What we can do with the solution:
 - Graph it / plot it
 - Explore its properties
 - Interpret it in physical terms

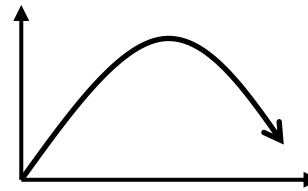
“All models are wrong, but some are useful.”

– George Box



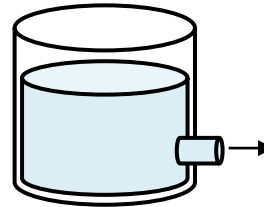
RLC Circuits

$$\frac{d^2 I}{dt^2} + \frac{R}{L} \frac{dI}{dt} + \frac{1}{LC} I = 0$$



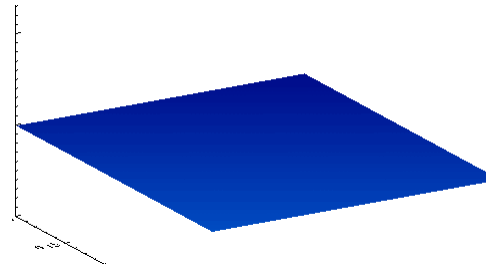
Projectile Motion

$$\begin{aligned} \frac{dv_x}{dt} &= \mu v_x \\ \frac{dv_y}{dt} &= -g - \mu v_y \end{aligned}$$



Draining a Tank

$$\frac{dh}{dt} = -C_v \sqrt{h}$$



Shallow Water Equations

$$\begin{aligned} \frac{\partial(\rho\eta)}{\partial t} + \frac{\partial(\rho\eta u)}{\partial x} + \frac{\partial(\rho\eta v)}{\partial y} &= 0, \\ \frac{\partial(\rho\eta u)}{\partial t} + \frac{\partial}{\partial x} \left(\rho\eta u^2 + \frac{1}{2} \rho g \eta^2 \right) + \frac{\partial(\rho\eta uv)}{\partial y} &= 0, \\ \frac{\partial(\rho\eta v)}{\partial t} + \frac{\partial(\rho\eta uv)}{\partial x} + \frac{\partial}{\partial y} \left(\rho\eta v^2 + \frac{1}{2} \rho g \eta^2 \right) &= 0. \end{aligned}$$

Chemical Kinetics

$$\begin{aligned} \frac{dC_A}{dt} &= \frac{F}{V} (C_{Af} - C_A) - k_1 C_A - k_3 C_A^2 \\ \frac{dC_B}{dt} &= -\frac{F}{V} C_B + k_1 C_A - k_2 C_B \end{aligned}$$

Transport Equations

$$\frac{\partial c}{\partial t} = D \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) - kc^2$$

Radioactive Decay

$$\frac{dA}{dt} = -kA$$

Review: Differential Equations

Differential Algebraic Equations (DAEs)

General form of the
DAE model with time:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y} = \mathbf{g}(t, \mathbf{x}(t), \mathbf{u}(t)) \end{cases}$$

If we expand all vectors:

$$\begin{cases} \dot{x}_1 = \frac{dx_1}{dt} = f_1(t, x_1(t), x_2(t), \dots, u_1(t), u_2(t), \dots) \\ \dot{x}_2 = \frac{dx_2}{dt} = f_2(t, x_1(t), x_2(t), \dots, u_1(t), u_2(t), \dots) \\ \vdots \end{cases}$$

$$\begin{cases} y_1 = g_1(t, x_1(t), x_2(t), \dots, u_1(t), u_2(t), \dots) \\ y_2 = g_2(t, x_1(t), x_2(t), \dots, u_1(t), u_2(t), \dots) \\ \vdots \end{cases}$$

Models with higher-order derivatives $\left(\frac{d^n x}{dt^n}\right)$ can be cast into a **system of purely first-order DEs** using a change of variables (recall ES 204).

Where:

$$\mathbf{x}(t) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix} = \text{state variables} \\ \text{(those that appear in } \textit{time-derivative} \text{ terms)}$$

$$\mathbf{u}(t) = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \end{bmatrix} = \text{input variables} \\ \text{(those that can be independently } \textit{adjusted})$$

$$\mathbf{y}(t) = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix} = \text{output variables} \\ \text{(those that are physically } \textit{measured})$$

$\mathbf{f}(\cdot)$ = state equations
(expressions found in the differential equations)

$\mathbf{g}(\cdot)$ = output equations
(expressions for computing the output variables)

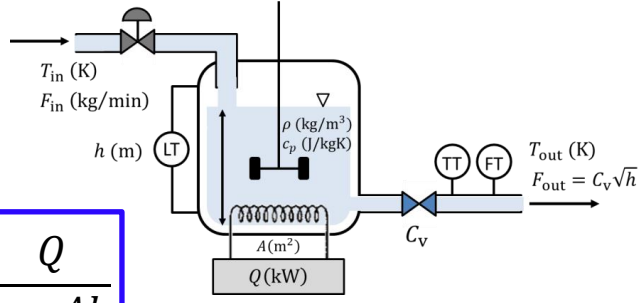
Review: Differential Equations

Differential Algebraic Equations (DAEs)

Example: Heated Stirred Tank

$$\frac{dx_1}{dt} \frac{dh}{dt} = \frac{1}{\rho A} (F_{in} - F_{out}) \quad f_1(t, x, u)$$

$$\frac{dx_2}{dt} \frac{dT_{out}}{dt} = \frac{F_{in}}{\rho A h} (T_{in} - T_{out}) + \frac{Q}{\rho c_p A h} \quad f_2(t, x, u)$$

$$y_1 \quad F_{out} = C_v \sqrt{h} \quad g_1(t, x, u)$$


State variables: $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} h \\ T_{out} \end{bmatrix}$

Input variables: $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} F_{in} \\ T_{in} \\ Q \end{bmatrix}$

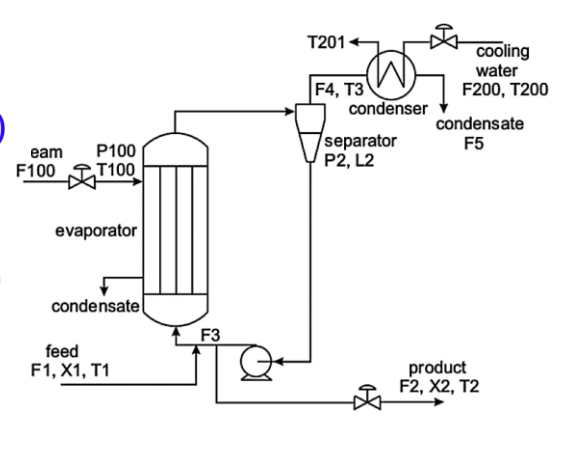
Output variables: $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} F_{out} \\ T_{out} \\ h \end{bmatrix}$

*Variables can be written in any order inside a vector, but you must consistently follow the same order all the time.

Example: Newell-Lee Evaporator

$$\frac{dx_1}{dt} \frac{dL_2}{dt} = \frac{F_1 - F_4 - F_2}{20} \quad f_1(t, x, u)$$

$$\frac{dx_2}{dt} \frac{dX_2}{dt} = \frac{F_1 X_1 - F_2 X_2}{20} \quad f_2(t, x, u)$$

$$\frac{dx_3}{dt} \frac{dP_2}{dt} = \frac{F_4 - F_5}{4} \quad f_3(t, x, u)$$


$$T_2 = 0.5616P_2 + 0.3126X_2 + 48.43$$

$$T_3 = 0.507P_2 + 55.0$$

$$F_4 = \frac{Q_{100} - 0.07F_1(T_2 - T_1)}{38.5}$$

$$T_{100} = 0.1538P_{100} + 90.0$$

$$Q_{100} = 0.16(F_1 + F_3)(T_{100} - T_2)$$

$$F_{100} = Q_{100}/36.6$$

$$Q_{200} = \frac{0.9576F_{200}(T_3 - T_{200})}{0.14F_{200} + 6.84}$$

$$T_{201} = T_{200} + Q_{200}/0.07F_{200}$$

$$F_5 = Q_{200}/38.5$$

$g_n(t, x, u)$

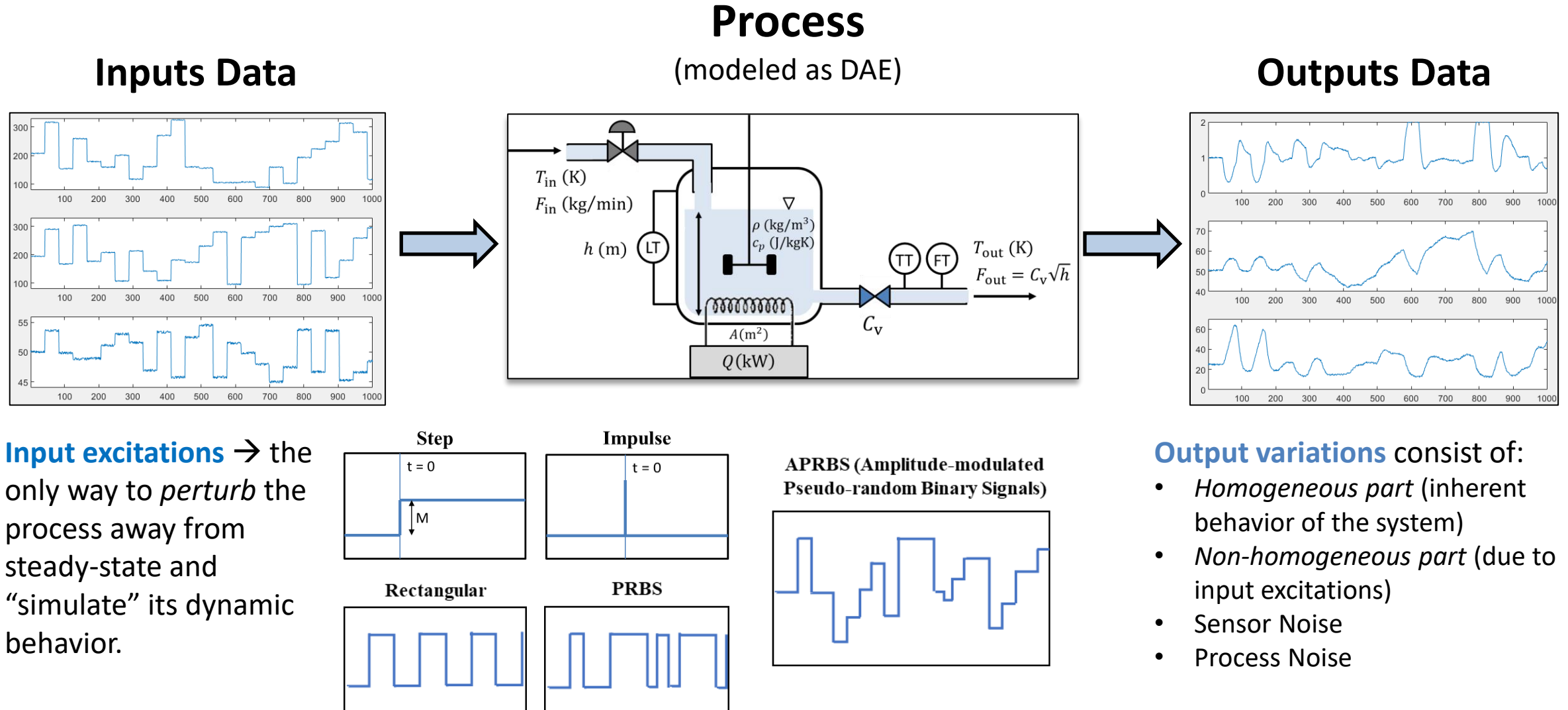
States: $\mathbf{x} = \begin{bmatrix} L_2 \\ P_2 \\ X_2 \end{bmatrix}$

Inputs: $\mathbf{u} = \begin{bmatrix} P_{100} \\ F_{200} \\ F_2 \end{bmatrix}$

Outputs: $\mathbf{y} = \begin{bmatrix} L_2 \\ P_2 \\ \vdots \end{bmatrix}$

Review: Differential Equations

What do we mean by “Simulate a DAE”?



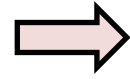
Optimization with DAEs

Standard NLP

Minimize: $f(\mathbf{x})$

Subject to: $h_i(\mathbf{x}) = 0 \quad i = 1, 2, \dots, l$
 $g_j(\mathbf{x}) \leq 0 \quad j = 1, 2, \dots, m$

and $x_k^L \leq x_k \leq x_k^U \quad k = 1, 2, \dots, n$



Optimization with DAE

Minimize: $\Psi(t, \mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t))$

Subject to: $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t))$ States
 $\mathbf{y} = \mathbf{g}(t, \mathbf{x}(t), \mathbf{u}(t))$ Outputs

and $h_i(t, \mathbf{x}, \mathbf{u}, \mathbf{y}) \leq 0 \quad i = 1, 2, \dots, m$

PARAMETER ESTIMATION

Given a real data set of $\mathbf{u}(t)$ and $\mathbf{y}(t)$,
find all parameter values within
 $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$ that fits the data.

DATA RECONCILIATION

Given a real data set of $\mathbf{u}(t)$ and $\mathbf{y}(t)$, and
fully known $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$, *reconcile* the
actual $\mathbf{y}(t)$ with the *simulated* $\mathbf{y}(t)$.

OPTIMAL CONTROL

Given a fully known $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$, *find*
 $\mathbf{u}(t)$ that achieves a *desired trajectory* in
 $\mathbf{y}(t)$ while satisfying other constraints.

OPTIMAL DESIGN

Find an operating point $\mathbf{x}(0)$, $\mathbf{u}(0)$ and
parameter values within $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$
that maximizes $\mathbf{y}(t)$ at steady-state.

(BATCH) SCHEDULING

Given a fully known $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$, *create a*
schedule for $\mathbf{u}(t)$ that minimizes total time
to achieve $\mathbf{y}(t)$.

REAL-TIME OPTIMIZATION (RTO)

Given a fully known $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$, *find*
 $\mathbf{u}(t)$ that maximizes $\mathbf{y}(t)$ over time
while satisfying other constraints.

Optimization with DAEs

Optimization with DAE

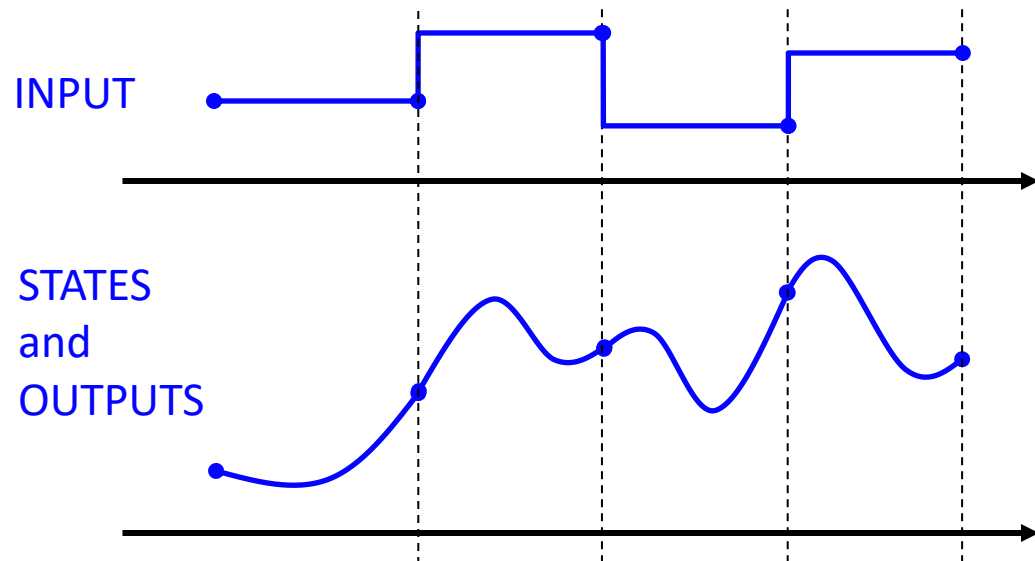
Minimize: $\Psi(t, x(t), y(t), u(t))$

Subject to: $\dot{x} = f(t, x(t), u(t))$ **States**

$y = g(t, x(t), u(t))$ **Outputs**

and $h_i(t, x, u, y) \leq 0 \quad i = 1, 2, \dots, m$

Discretization



To solve this, we need 2 routines:

DAE SOLVER

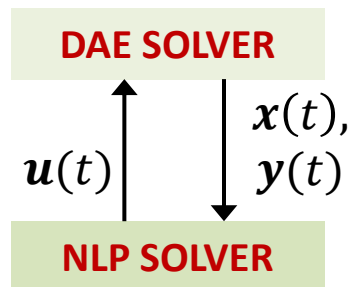
Integrate the DAE: Get $x(t)$ and $y(t)$ given $u(t)$ at the next time step, $t + \Delta t$.

NLP SOLVER

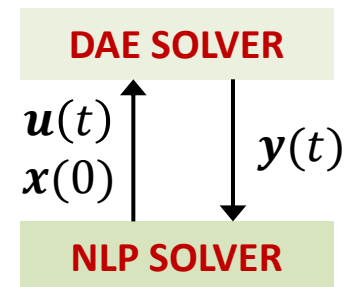
Do one iteration: Solve for next updates on the decision variables to minimize $\Psi(t, x(t), y(t), u(t))$

There are three approaches to combine the two:

Single Shooting



Multiple Shooting



Direct Collocation

NLP SOLVER is also the DAE SOLVER

NLP becomes bigger;
results are more accurate

Outline

- Review: Differential Equations
- Optimization with DAEs
- **Orthogonal Collocation**

Orthogonal Collocation

Optimization with DAE

Minimize: $\Psi(t, \mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t))$

Subject to: $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t))$ States

$\mathbf{y} = \mathbf{g}(t, \mathbf{x}(t), \mathbf{u}(t))$ Outputs

and $h_i(t, \mathbf{x}, \mathbf{u}, \mathbf{y}) \leq 0 \quad i = 1, 2, \dots, m$

In **orthogonal collocation**, our goal is to turn the differential equations into a linear system.

$$\mathbf{A}\mathbf{x} = \mathbf{B}$$

A single differential eq'n:

$$\frac{dx(\bar{t})}{d\bar{t}} = f(\bar{t}, x(\bar{t}), u(\bar{t}))$$

becomes $N_{FE} \times N_c + 2$ equalities:

$$\frac{dx(\bar{t})}{d\bar{t}} \cong \frac{1}{\Delta t} \sum_{i=0}^{N_c} \left[\frac{d\phi_i(\bar{t})}{d\bar{t}} \Big|_{t=t_j} x(\bar{t}_i) \right] = f(\bar{t}_j, x(\bar{t}_j), u(\bar{t}_j))$$

plus $N_{FE} - 1$ continuity constraints:

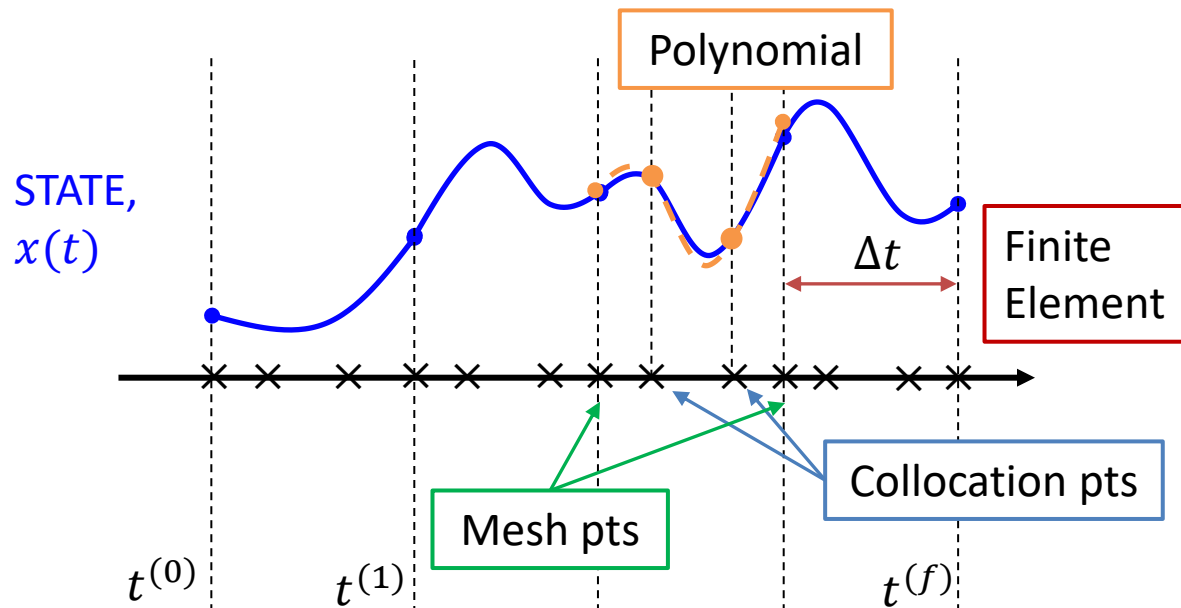
$$\sum_{m=0}^{N_c} \left[\frac{d\phi_m(\bar{t})}{d\bar{t}} \Big|_{\bar{t}_{N_c}^{(i)}} x(\bar{t}_m) \right] = \sum_{m=0}^{N_c} \left[\frac{d\phi_m(\bar{t})}{d\bar{t}} \Big|_{\bar{t}_0^{(i+1)}} x(\bar{t}_m) \right]$$

$$\phi_i(\bar{t}) = \prod_{\substack{m=0 \\ m \neq i}}^{N_c+2} \left(\frac{\bar{t} - \bar{t}_m}{\bar{t}_i - \bar{t}_m} \right)$$

Basis functions (polynomials) forced to be 1 at only one collocation point t_m at a time.

Polynomial

$$x(t) \approx \sum_{i=0}^{N_c+2} x(\bar{t}_i) \phi_i(\bar{t})$$



Orthogonal Collocation

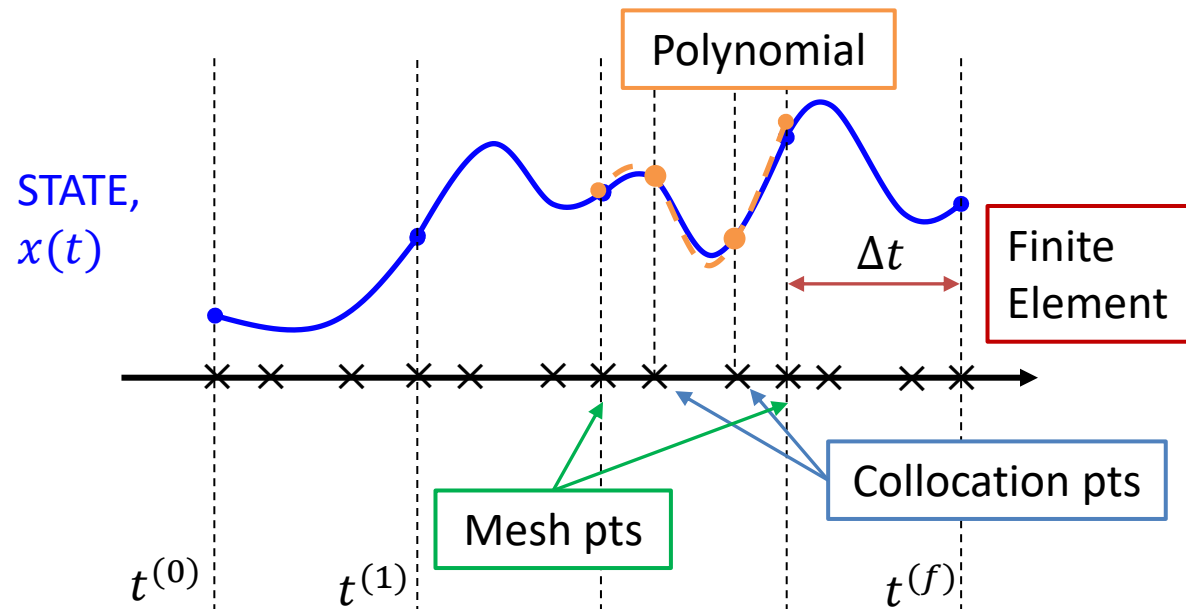
Optimization with DAE

Minimize: $\Psi(t, \mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t))$

Subject to: $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t))$ **States**

$\mathbf{y} = \mathbf{g}(t, \mathbf{x}(t), \mathbf{u}(t))$ **Outputs**

and $h_i(t, \mathbf{x}, \mathbf{u}, \mathbf{y}) \leq 0 \quad i = 1, 2, \dots, m$

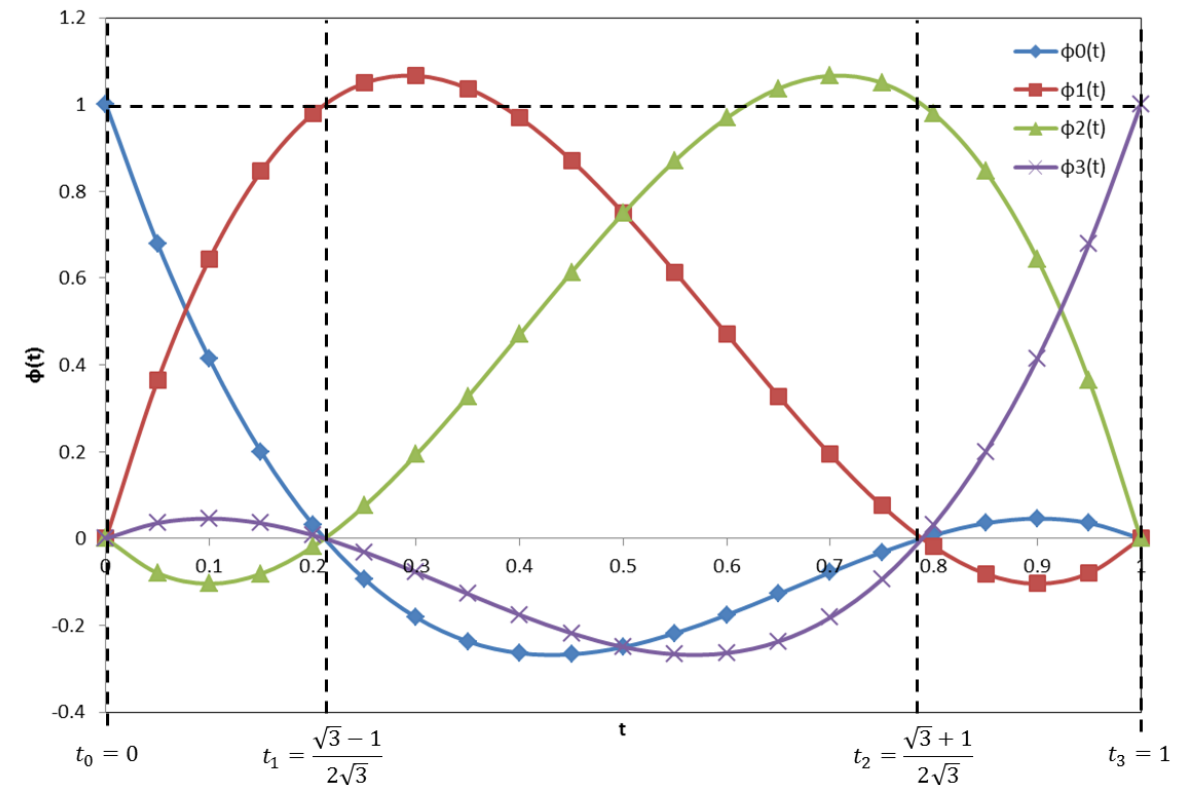


$$\phi_i(\bar{t}) = \prod_{\substack{m=0 \\ m \neq i}}^{N_c+2} \left(\frac{\bar{t} - \bar{t}_m}{\bar{t}_i - \bar{t}_m} \right)$$

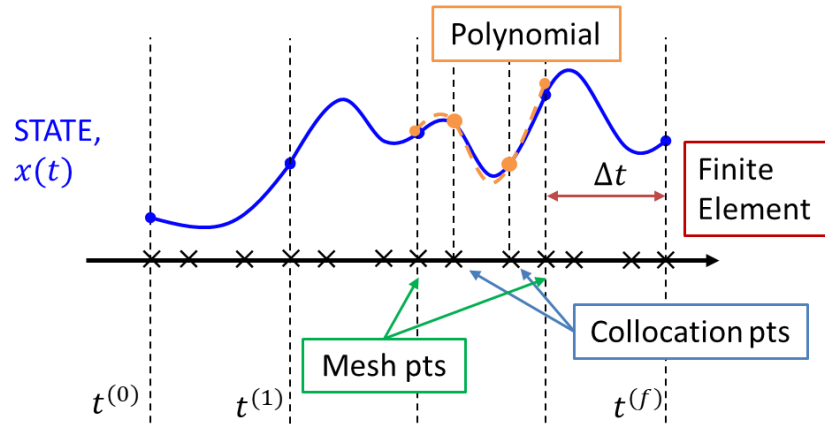
Basis functions (polynomials) forced to be 1 at only one collocation point t_m at a time.

For the 2nd order shifted **Legendre** polynomial, these are the basis functions:

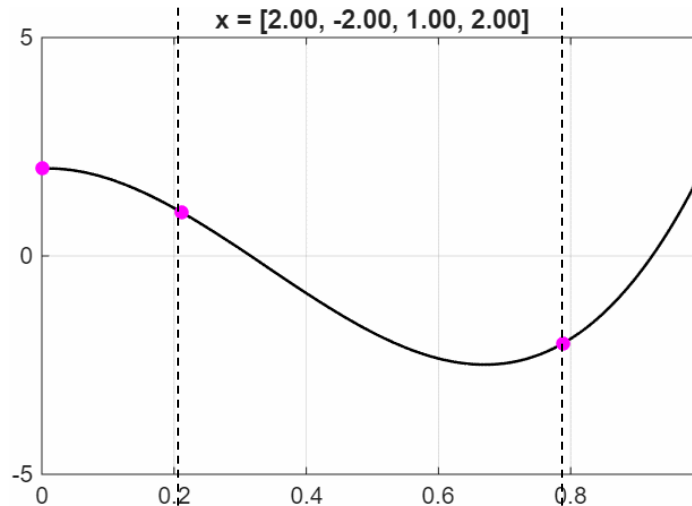
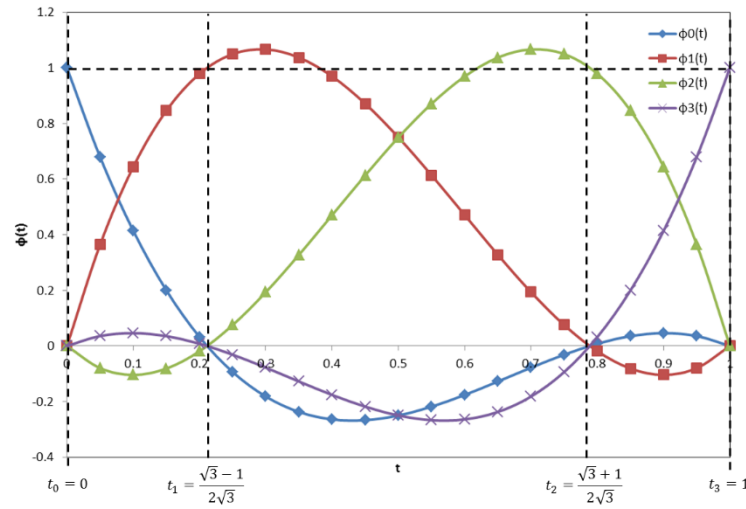
$$\tilde{P}_2(x) = 6x^2 - 6x + 1$$



Orthogonal Collocation

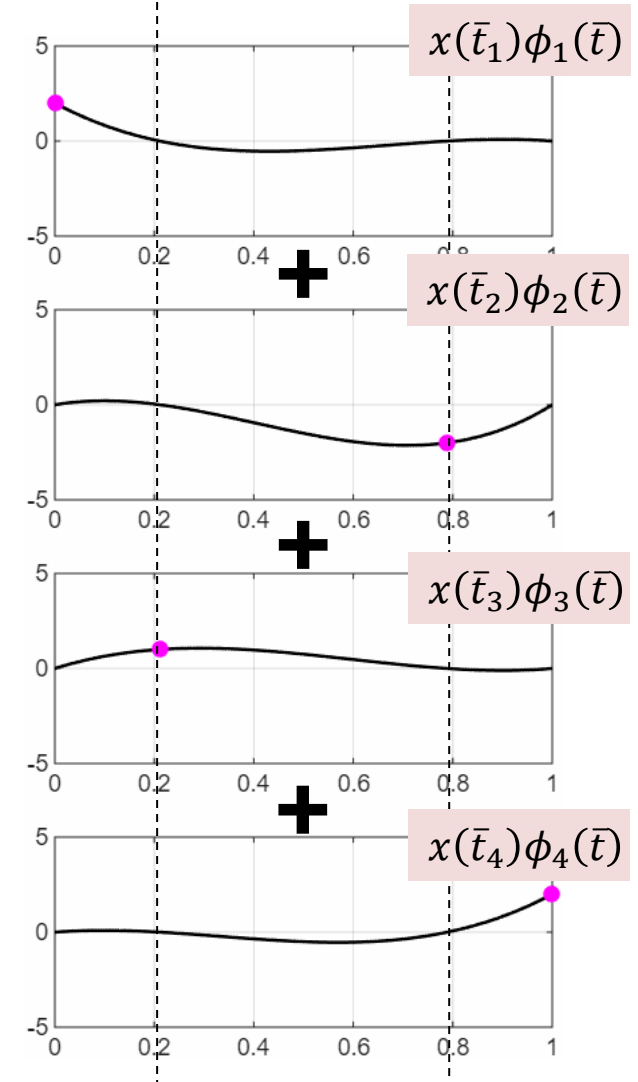


- All we need to fully define the curve within a finite element are 4 values: $x(t_i)$
- Each $x(t_i)$ defines its own cubic polynomial, which is 1 at one collocation point at a time, 0 at the others.
- The sum of 4 cubic polynomials are assumed to approximate the true solution $x(t)$.



$$x(t) \approx \sum_{i=0}^{N_c+2} x(\bar{t}_i) \phi_i(\bar{t})$$

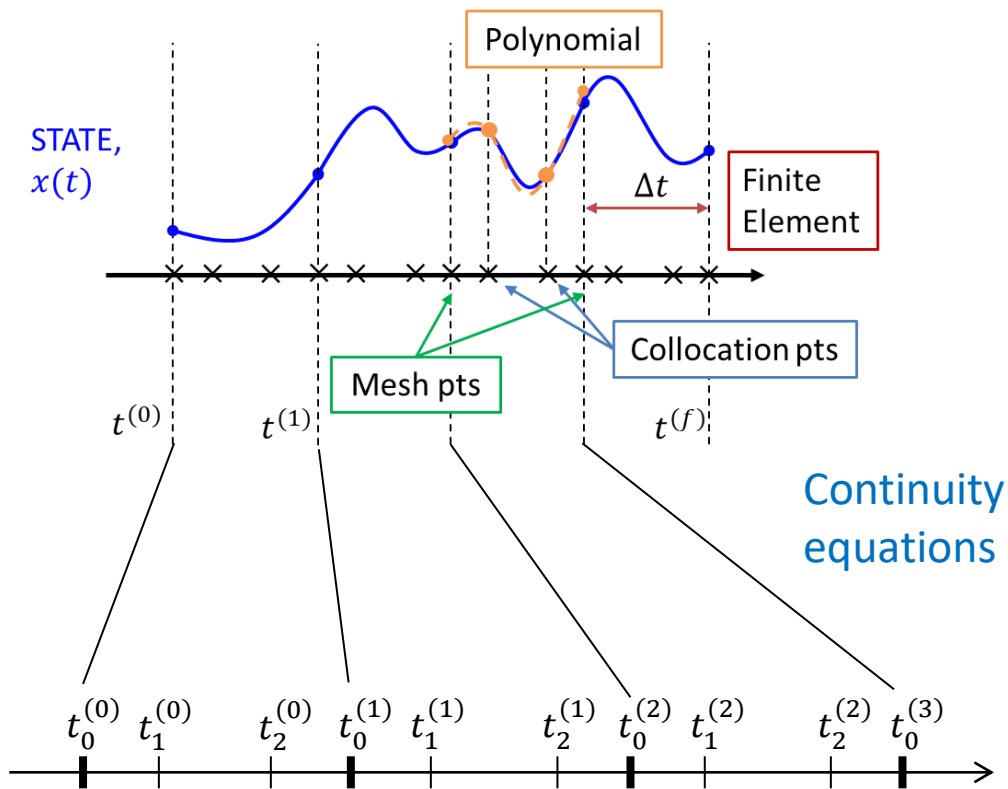
These are the 4 basis functions, one for each collocation point:



Orthogonal Collocation

In **orthogonal collocation**, our goal is to turn the differential equations into a linear system.

$$\begin{aligned} \dot{x} &= f(t, x(t), u(t)) \\ y &= g(t, x(t), u(t)) \end{aligned} \quad \Rightarrow \quad \mathbf{Ax} = \mathbf{B}$$



Here is an example of the resulting linear system for 2 states (or 2 differential equations), 2 collocation points, and 3 finite elements:
Legend: X = non-zero

States, x

x_1

x_2

Inputs, u

$t_0^{(0)}$	$t_1^{(0)}$	$t_2^{(0)}$	$t_0^{(1)}$	$t_1^{(1)}$	$t_2^{(1)}$	$t_0^{(2)}$	$t_1^{(2)}$	$t_2^{(2)}$	$t_0^{(3)}$	$t_0^{(0)}$	$t_1^{(0)}$	$t_2^{(0)}$	$t_0^{(1)}$	$t_1^{(1)}$	$t_2^{(1)}$	$t_0^{(2)}$	$t_1^{(2)}$	$t_2^{(2)}$	$t_0^{(3)}$	$A_0^{(0)}$	$A_0^{(1)}$	$A_0^{(2)}$	$A_0^{(3)}$	$T_0^{(0)}$	$T_0^{(1)}$	$T_0^{(2)}$	$T_0^{(3)}$
X	X	X	X								X										X						
X	X	X	X								X										X						
X	X	X	X									X									X						
X	X	X	X	X	X	X	X																				
				X	X	X	X								X							X					
				X	X	X	X								X							X					
				X	X	X	X	X	X														X				
					X	X	X	X															X				
					X	X	X	X															X				
						X	X	X	X															X			
						X	X	X	X															X			
							X	X	X	X															X		
								X	X	X	X														X		
									X	X	X	X														X	
										X	X	X	X														X
											X	X	X	X													
												X	X	X	X												
													X	X	X	X											
														X	X	X	X										
															X	X	X	X									
																X	X	X	X								
																	X	X	X	X							
																		X	X	X	X						
																			X	X	X	X					
																				X	X	X	X				
																					X	X	X	X			
																						X	X	X	X		
																							X	X	X	X	
																								X	X	X	X
																									X	X	X
																										X	X
																											X

Collocation matrix

Outline

- Review: Differential Equations
- Optimization with DAEs
- Orthogonal Collocation
- **Pyomo DAE**

Pyomo DAE

Simulate the following ODE in the domain
 $t \in [0, 2]$:

$$\frac{dz}{dt} = z^2 - 2z + 1$$

with the initial condition $z(0) = -3$.
Compare the result with the analytical
solution:

$$z(t) = \frac{4t - 3}{4t + 1}$$

and also with scipy's RK45 solver.

```
model = m = ConcreteModel()

m.t = ContinuousSet(bounds=(0,2))

m.z = Var(m.t)
m.dzdt = DerivativeVar(m.z)

m.obj = Objective(expr=1) # Dummy Objective

def _zdot(m, i):
    return m.dzdt[i] == m.z[i]**2 - 2*m.z[i] + 1

m.zdot = Constraint(m.t, rule=_zdot)

def _init_con(m):
    return m.z[0] == -3

m.init_con = Constraint(rule=_init_con)

# Discretize using collocation
discretizer = TransformationFactory('dae.collocation')
discretizer.apply_to(m, nfe=4, ncp=3,
                    scheme='LAGRANGE-RADAU')

# Solve using Pyomo IPOPT
solver = SolverFactory('cyipopt')
solver.solve(m, tee=True)

colloc_t = list(m.t)
colloc_z = [value(m.z[i]) for i in m.t]

plt.plot(colloc_t, colloc_z, 'ro--')
plt.xlabel('t')
plt.grid()
plt.show()
```

Pyomo DAE

Simulate the following ODE in the domain $t \in [0, 2]$:

$$\frac{dz}{dt} = z^2 - 2z + 1$$

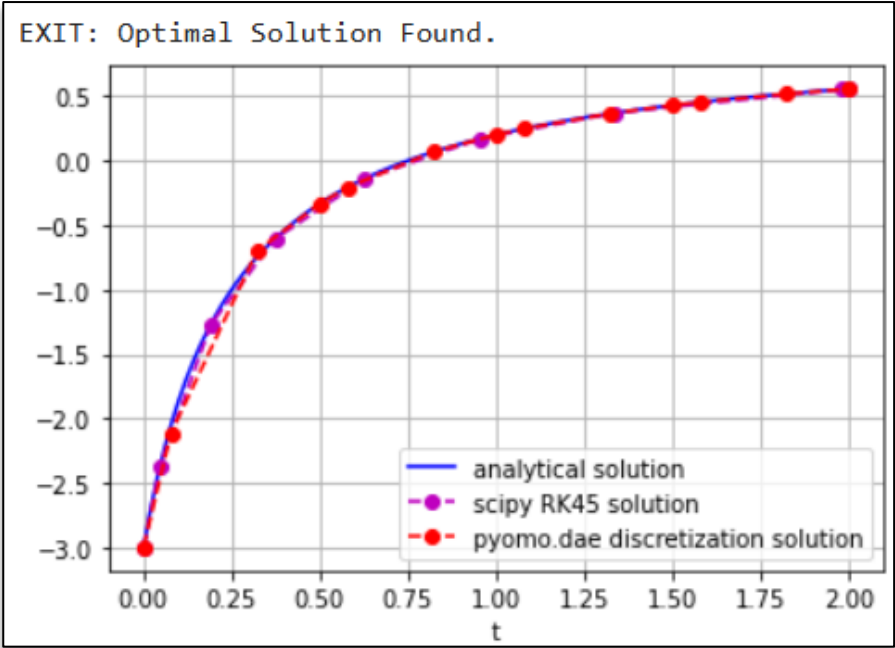
with the initial condition $z(0) = -3$.
Compare the result with the analytical solution:

$$z(t) = \frac{4t - 3}{4t + 1}$$

and also with scipy's RK45 solver.

Number of nonzeros in equality constraint Jacobian...	87
Number of nonzeros in inequality constraint Jacobian..	0
Number of nonzeros in Lagrangian Hessian.....	13
Total number of variables.....	26
variables with only lower bounds:	0
variables with lower and upper bounds:	0
variables with only upper bounds:	0
Total number of equality constraints.....	26
Total number of inequality constraints.....	0
inequality constraints with only lower bounds:	0
inequality constraints with lower and upper bounds:	0
inequality constraints with only upper bounds:	0
iter	objective
0	1.0000000e+00
1	1.0000000e+00
2	1.0000000e+00
3	1.0000000e+00
4	1.0000000e+00
5	1.0000000e+00

Number of Iterations.....: 5



Pyomo DAE

Solve the following BVP within
 $x \in [0, 10]$ then compare Pyomo
DAE with scipy BVP:

$$y'' - y'(\sin x) + xy = \cos x$$

$$\begin{aligned} \text{B. C. } y(0) &= 0.5, \\ y(10) &= -1 \end{aligned}$$

```
m = ConcreteModel()
m.xf = Param(initialize=10)
m.x = ContinuousSet(bounds=(0,m.xf))
m.y1 = Var(m.x) # This is dy/dx
m.y2 = Var(m.x) # This is the original y

m.dy1dt = DerivativeVar(m.y1) # This is dy^2/dx^2
m.dy2dt = DerivativeVar(m.y2) # This is dy/dx

m.obj = Objective(expr=1) # Dummy Objective

def _zdot1(m, i):
    return m.dy1dt[i] == m.y1[i]*np.sin(i) - i*m.y2[i] + np.cos(i)
m.zdot1 = Constraint(m.x, rule=_zdot1)

def _zdot2(m, i):
    return m.dy2dt[i] == m.y1[i]
m.zdot2 = Constraint(m.x, rule=_zdot2)

def _boundary_con(m):
    yield m.y2[0] == 0.5
    yield m.y2[m.xf] == -1

m.boundary_con = ConstraintList(rule=_boundary_con)

# Discretize using collocation
discretizer = TransformationFactory('dae.collocation')
discretizer.apply_to(m, nfe=20, ncp=3, scheme='LAGRANGE-RADAU')

# Solve using Pyomo IPOPT
solver = SolverFactory('cyipopt')
solver.solve(m, tee=True)

colloc_x = list(m.x)
colloc_y = [value(m.y2[i]) for i in m.x]

plt.plot(colloc_x, colloc_y, 'ro--')
plt.xlabel('x')
plt.grid()
plt.show()
```

Pyomo DAE

Solve the following BVP within $x \in [0, 10]$ then compare Pyomo DAE with scipy BVP:

$$y'' - y'(\sin x) + xy = \cos x$$

B. C. $y(0) = 0.5,$
 $y(10) = -1$

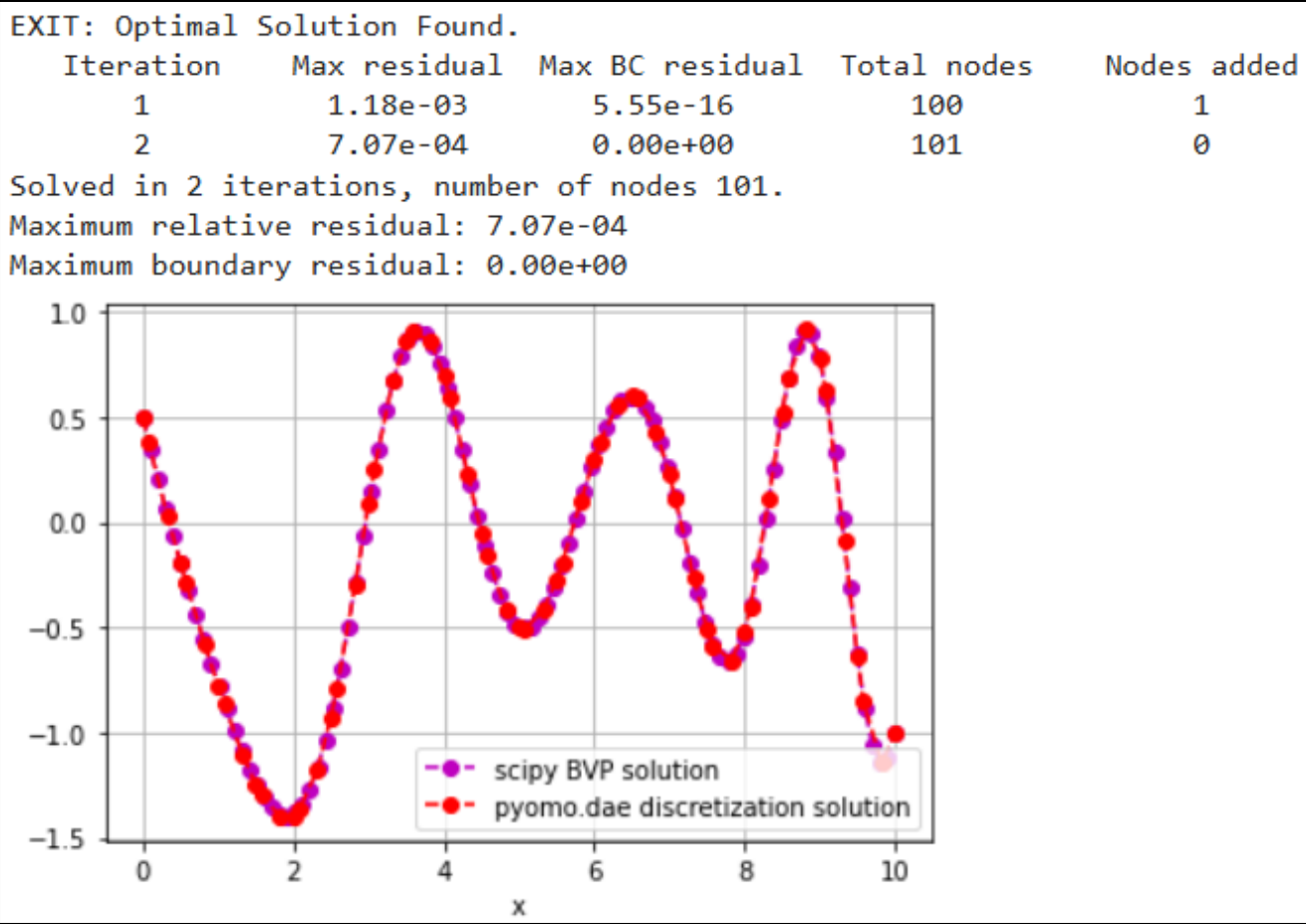
```
Number of nonzeros in equality constraint Jacobian...: 905
Number of nonzeros in inequality constraint Jacobian...: 0
Number of nonzeros in Lagrangian Hessian.....: 0

Total number of variables.....: 244
    variables with only lower bounds: 0
    variables with lower and upper bounds: 0
    variables with only upper bounds: 0
Total number of equality constraints.....: 244
Total number of inequality constraints.....: 0
    inequality constraints with only lower bounds: 0
    inequality constraints with lower and upper bounds: 0
    inequality constraints with only upper bounds: 0

iter  objective  inf_pr  inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr ls
  0  1.0000000e+00  1.00e+00  0.00e+00  -1.0  0.00e+00  -  0.00e+00  0.00e+00  0
  1  1.0000000e+00  7.11e-15  0.00e+00  -1.7  1.03e+01  -  1.00e+00  1.00e+00h  1

Number of Iterations.....: 1

              (scaled)              (unscaled)
Objective.....:  1.0000000000000000e+00  1.0000000000000000e+00
Dual infeasibility.....:  0.0000000000000000e+00  0.0000000000000000e+00
Constraint violation....:  7.1054273576010019e-15  7.1054273576010019e-15
Complementarity.....:  0.0000000000000000e+00  0.0000000000000000e+00
Overall NLP error.....:  7.1054273576010019e-15  7.1054273576010019e-15
```



Pyomo DAE

Simulate the following SIR model for $t \in [0, 200]$ days, with $\beta = 0.1$, $\gamma = 0.04$, and a population of $N = 100$:

$$\frac{dS}{dt} = -\frac{\beta IS}{N}$$

$$\frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

Use initial conditions $S(0) = 95$, $I(0) = 3$, $R(0) = 2$. Redo the simulation with $\beta = 0.08$ then compare.

```
m = ConcreteModel()
m.t = ContinuousSet(bounds=(0,200))

m.S = Var(m.t, initialize=95)
m.I = Var(m.t, initialize=3)
m.R = Var(m.t, initialize=2)
m.beta = Param(initialize=0.1, mutable=True)
m.gamma = Param(initialize=0.04)
m.N = Param(initialize=100)
m.dSdt = DerivativeVar(m.S)
m.dIdt = DerivativeVar(m.I)
m.dRdt = DerivativeVar(m.R)

m.obj = Objective(expr=1) # Dummy Objective
m.zdot1 = Constraint(m.t, rule=lambda m, i: \
                    m.dSdt[i] == -m.beta*m.I[i]*m.S[i]/m.N)
m.zdot2 = Constraint(m.t, rule=lambda m, i: \
                    m.dIdt[i] == m.beta*m.I[i]*m.S[i]/m.N - m.gamma*m.I[i])
m.zdot3 = Constraint(m.t, rule=lambda m, i: \
                    m.dRdt[i] == m.gamma*m.I[i])

def _init_con(m):
    yield m.S[0] == 95
    yield m.I[0] == 3
    yield m.R[0] == 2
m.init_con = ConstraintList(rule=_init_con)

# Discretize using collocation
discretizer = TransformationFactory('dae.collocation')
discretizer.apply_to(m, nfe=20, ncp=3, scheme='LAGRANGE-RADAU')

# Solve using Pyomo IPOPT
solver = SolverFactory('cyipopt')
solver.solve(m, tee=True)
colloc_t = list(m.t)
colloc_S = [value(m.S[i]) for i in m.t]
colloc_I = [value(m.I[i]) for i in m.t]
colloc_R = [value(m.R[i]) for i in m.t]
plt.plot(colloc_t, colloc_S, 'mo--', label='Susceptible (S)')
plt.plot(colloc_t, colloc_I, 'ro--', label='Infected (I)')
plt.plot(colloc_t, colloc_R, 'bo--', label='Removed (R)')
plt.legend(loc='best')
plt.xlabel('Days')
plt.grid()
```

Pyomo DAE

Simulate the following SIR model for $t \in [0,200]$ days, with $\beta = 0.1$, $\gamma = 0.04$, and a population of $N = 100$:

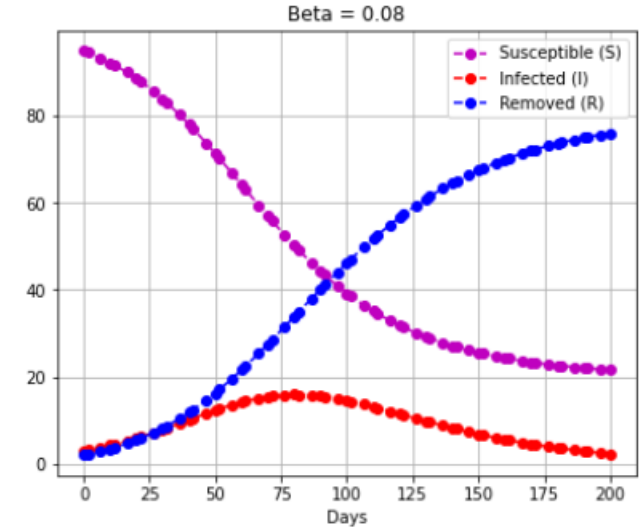
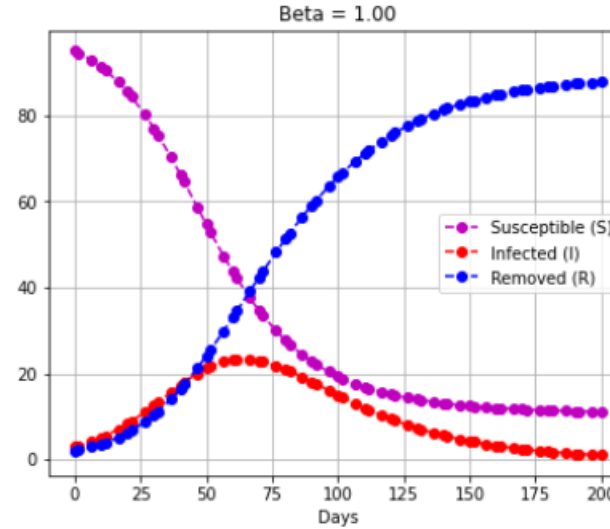
$$\frac{dS}{dt} = -\frac{\beta IS}{N}$$

$$\frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

Use initial conditions $S(0) = 95$, $I(0) = 3$, $R(0) = 2$.
Redo the simulation with $\beta = 0.08$ then compare.

EXIT: Optimal Solution Found.



```
Number of nonzeros in equality constraint Jacobian...: 1391
Number of nonzeros in inequality constraint Jacobian.: 0
Number of nonzeros in Lagrangian Hessian.....: 61

Total number of variables.....: 366
    variables with only lower bounds: 0
    variables with lower and upper bounds: 0
    variables with only upper bounds: 0
Total number of equality constraints.....: 366
Total number of inequality constraints.....: 0
    inequality constraints with only lower bounds: 0
    inequality constraints with lower and upper bounds: 0
    inequality constraints with only upper bounds: 0
```

iter	objective	inf_pr	inf_du	lg(mu)	d	lg(rg)	alpha_du	alpha_pr	ls
0	1.0000000e+00	2.85e-01	0.00e+00	-1.0	0.00e+00	-	0.00e+00	0.00e+00	0
1	1.0000000e+00	3.29e-01	0.00e+00	-1.7	1.22e+05	-	1.00e+00	2.44e-04h	13
2	1.0000000e+00	5.24e-01	0.00e+00	-1.7	4.77e+04	-	1.00e+00	4.88e-04h	12
3	1.0000000e+00	5.31e-01	0.00e+00	-1.7	1.82e+04	-	1.00e+00	9.77e-04h	11
4	1.0000000e+00	4.12e-01	0.00e+00	-1.7	8.15e+03	-	1.00e+00	3.91e-03h	9
5	1.0000000e+00	3.92e-01	0.00e+00	-1.7	2.41e+03	-	1.00e+00	7.81e-03h	8
6	1.0000000e+00	3.60e-01	0.00e+00	-1.7	1.05e+03	-	1.00e+00	3.13e-02h	6
7	1.0000000e+00	5.49e-01	0.00e+00	-1.7	3.08e+02	-	1.00e+00	1.25e-01h	4
8	1.0000000e+00	5.58e-01	0.00e+00	-1.7	7.61e+01	-	1.00e+00	5.00e-01h	2
9	1.0000000e+00	8.21e-02	0.00e+00	-1.7	1.02e+01	-	1.00e+00	1.00e+00h	1
iter	objective	inf_pr	inf_du	lg(mu)	d	lg(rg)	alpha_du	alpha_pr	ls
10	1.0000000e+00	1.83e-03	0.00e+00	-2.5	2.51e+00	-	1.00e+00	1.00e+00h	1
11	1.0000000e+00	1.67e-06	0.00e+00	-3.8	7.90e-02	-	1.00e+00	1.00e+00h	1
12	1.0000000e+00	2.16e-12	0.00e+00	-8.6	8.72e-05	-	1.00e+00	1.00e+00h	1

Outline

- Review: Differential Equations
- Optimization with DAEs
- Orthogonal Collocation
- **Pyomo DAE**
 - Optimal Control
 - Parameter Estimation
 - Data Reconciliation

OPTIMAL CONTROL

Given a fully known $f(\cdot)$ and $g(\cdot)$, find $u(t)$ that achieves a desired trajectory in $y(t)$ while satisfying other constraints.

PARAMETER ESTIMATION

Given a real data set of $u(t)$ and $y(t)$, find all parameter values within $f(\cdot)$ and $g(\cdot)$ that fits the data.

DATA RECONCILIATION

Given a real data set of $u(t)$ and $y(t)$, and fully known $f(\cdot)$ and $g(\cdot)$, reconcile the actual $y(t)$ with the simulated $y(t)$.