# Model Predictive Control

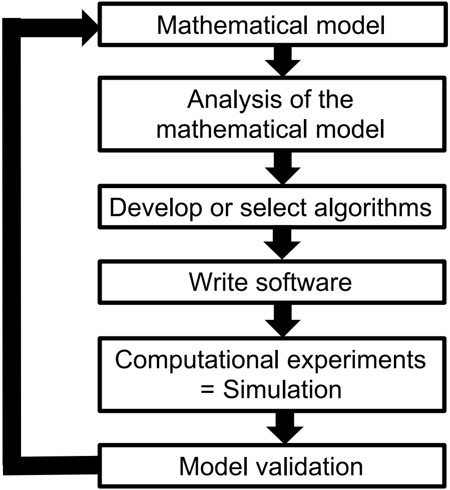
## Lecture 03C - Overview of models and methods

John Bagterp Jørgensen

*Department of Applied Mathematics and Computer Science*
*Technical University of Denmark*

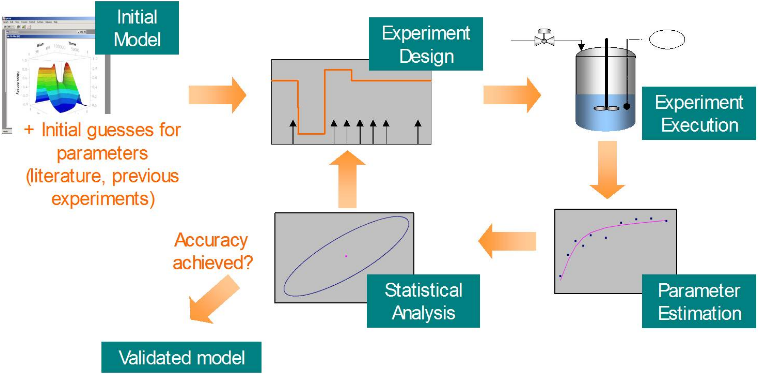
02619 Model Predictive Control

# Steps in Applied Mathematics




- **Mathematical model**
  Use physical, chemical, biological and economical principles to develop a system of differential equations that **approximately** describes the problem studied.
- **Analysis of the mathematical model**
  Determine the type of model and the appropriate methods for solving it.
- **Develop or select algorithms**
  Use library algorithms or develop new algorithms for solving the differential equations that model the system.
- **Write software**
  Implement the algorithm in software.
- **Simulation**
  Run the software on instances of the problem to understand the model and to use the model to answer engineering questions (design, operation, etc).
- **Model validation**
  Interpret the results of the simulation and decide if the model adequate or need to be revised.

# Systematic Mathematical Modeling - The Model Building Cycle




Mathematical model

$$\hat{x}(t_0) = x_0$$

$$\frac{d\hat{x}}{dt}(t) = f(\hat{x}(t), u(t), p)$$

$$\hat{y}(t_k) = g(\hat{x}(t_k), p)$$

Data

- Measurements: $y(t_k)$
- Error: $e(t_k) = \hat{y}(t_k) - y(t_k)$
- Least-squares error metric: $V = \frac{1}{2}\sum_k \|e(t_k)\|_2^2$
- Adjust the parameters, $p$, such that the model fits the data best possible

# Computational Tasks in Systematic Model Building

- Simulation = solution of intitial value problems in the form

$$\hat{x}(t_0) = x_0 \tag{1a}$$

$$\frac{d}{dt}\hat{x}(t) = f(t, \hat{x}(t), p) \tag{1b}$$

$$\hat{y}(t_k) = g(\hat{x}(t_k), p) \tag{1c}$$

- Parameter estimation = optimization problem with ODEs

$$\min_p \qquad V(p) = \frac{1}{2}\sum_k \|\hat{y}(t_k, p) - y(t_k)\|_2^2 \tag{2a}$$

$$s.t. \qquad \hat{x}(t_0) = x_0 \tag{2b}$$

$$\frac{d}{dt}\hat{x}(t, p) = f(t, \hat{x}(t, p), p) \tag{2c}$$

$$\hat{y}(t_k, p) = g(\hat{x}(t_k, p), p) \tag{2d}$$

$$p_l \leq p \leq p_u \tag{2e}$$

# Mathematical Modeling with Differential Equations

# Conservation Principle

Physical models are based on conservation principles.

1. Conservation of mass
2. Conservation of energy
3. Conservation of momentum (force)

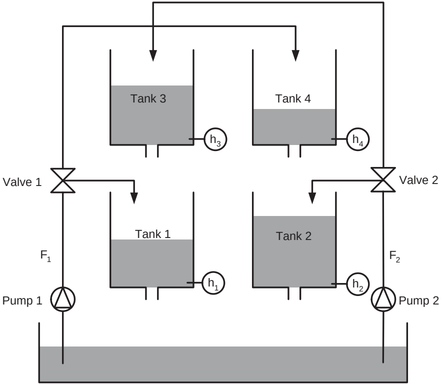
The general derivation of the system equations have the form

$$\text{Accumulated} = \text{Influx} - \text{Outflux} + \overbrace{\text{Produced} - \text{Consumed}}^{=\text{Generated}}$$

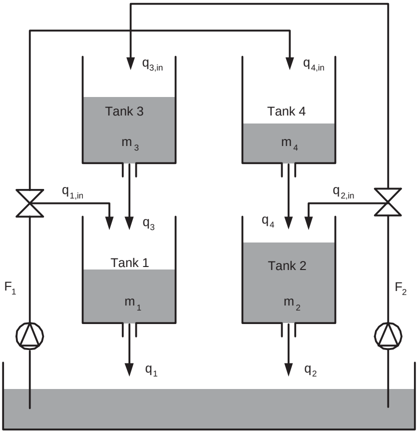
For non-reactive systems the generation term is absent

$$\text{Accumulated} = \text{Influx} - \text{Outflux}$$

# 4-Tank System - Motivating Example



# Example - Tank 1




$$\text{Accumulated} = \text{In - Out}$$

with

$$\begin{aligned} \text{Accumulated} &= m_1(t+\Delta t) - m_1(t) \\ \text{In} &= \rho q_{1,in}(t)\Delta t + \rho q_3(t)\Delta t \\ \text{Out} &= \rho q_1(t)\Delta t \end{aligned}$$

$$\underbrace{m_1(t+\Delta t) - m_1(t)}_{\text{Accumulated}} = \underbrace{\rho q_{1,in}(t)\Delta t + \rho q_3(t)\Delta t}_{\text{In}} - \underbrace{\rho q_1(t)\Delta t}_{\text{Out}}$$

# Example - Tank 1

1. Conservation of mass

$$\underbrace{m_1(t+\Delta t) - m_1(t)}_{\text{Accumulated}} = \underbrace{\rho q_{1,in}(t)\Delta t + \rho q_3(t)\Delta t}_{\text{In}} - \underbrace{\rho q_1(t)\Delta t}_{\text{Out}}$$

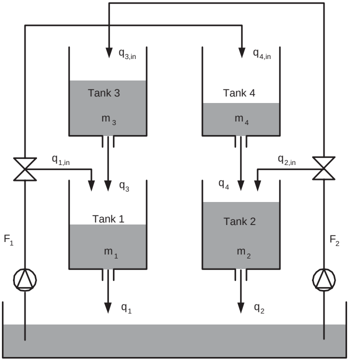
2. Divide by $\Delta t$

$$\frac{m_1(t+\Delta t) - m_1(t)}{\Delta t} = \rho q_{1,in}(t) + \rho q_3(t) - \rho q_1(t)$$

3. Let $\Delta t \to 0$

$$\frac{dm_1(t)}{dt} = \rho q_{1,in}(t) + \rho q_3(t) - \rho q_1(t)$$

# 4-Tank System - Model




Mass balances

$$\frac{dm_1}{dt}(t) = \rho q_{1,in}(t) + \rho q_3(t) - \rho q_1(t) \qquad m_1(t_0) = m_{1,0}$$
$$\frac{dm_2}{dt}(t) = \rho q_{2,in}(t) + \rho q_4(t) - \rho q_2(t) \qquad m_2(t_0) = m_{2,0}$$
$$\frac{dm_3}{dt}(t) = \rho q_{3,in}(t) - \rho q_3(t) \qquad m_3(t_0) = m_{3,0}$$
$$\frac{dm_4}{dt}(t) = \rho q_{4,in}(t) - \rho q_4(t) \qquad m_4(t_0) = m_{4,0}$$

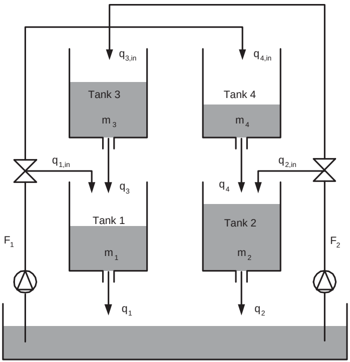
Inflows

$$q_{1,in}(t) = \gamma_1 F_1(t) \qquad q_{2,in}(t) = \gamma_2 F_2(t)$$
$$q_{3,in}(t) = (1-\gamma_2) F_2(t) \qquad q_{4,in}(t) = (1-\gamma_1) F_1(t)$$

Outflows

$$q_i(t) = a_i \sqrt{2 g h_i(t)} \qquad h_i(t) = \frac{m_i(t)}{\rho A_i} \qquad i \in \{1,2,3,4\}$$

# 4-Tank System - Model




System of ordinary differential equations

$$\dot{x}(t) = f(x(t), u(t), p) \qquad x(t_0) = x_0$$

with the vectors defined as

$$x = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} \qquad u = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}$$

$$p = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & A_1 & A_2 & A_3 & A_4 & \gamma_1 & \gamma_2 & g & \rho \end{bmatrix}^T$$

This is a non-stiff ODE system
as all processes take place on the same time-scale

# Generic Input-Output Model


$$\frac{dx(t)}{dt} = f(x(t), u(t), p) \qquad x(t_0) = x_0 \qquad \text{Process model}$$

$$y(t) = g(x(t), p) \qquad \text{Sensor function}$$

$$z(t) = h(x(t), p) \qquad \text{Output function}$$

# Simulation in Matlab

The model

$$\dot{x}(t) = f(t, x, u, p) \qquad x(t_0) = x_0$$

may be implemented in Matlab as

```
function xdot = ProcessModel(t,x,u,p)
% Process Model   dx/dt = f(t,x,u,p)
%
% Computes the RHS of dx/dt = f(t,x,u,p)
% It stores the result in xdot.

...
```

and called using

```
[T,X] = ode45(@ProcessModel,[t0 tf],x0,odeOptions,u,p)
```

# Model for the 4-Tank System

```
function xdot = FourTankSystem(t,x,u,p)
% FOURTANKSYSTEM   Model dx/dt = f(t,x,u,p) for 4-tank System
%
% This function implements a differential equation model for the
% 4-tank system.
%
% Syntax: xdot = FourTankSystem(t,x,u,p)

% Unpack states, MVs, and parameters
m     = x;                      % Mass of liquid in each tank [g]
F     = u;                      % Flow rates in pumps [cm3/s]
a     = p(1:4,1);               % Pipe cross sectional areas [cm2]
A     = p(5:8,1);               % Tank cross sectional areas [cm2]
gamma = p(9:10,1);              % Valve positions [-]
g     = p(11,1);                % Acceleration of gravity [cm/s2]
rho   = p(12,1);                % Density of water [g/cm3]

% Inflows
qin = zeros(4,1);
qin(1,1) = gamma(1)*F(1);       % Inflow from valve 1 to tank 1 [cm3/s]
qin(2,1) = gamma(2)*F(2);       % Inflow from valve 2 to tank 2 [cm3/s]
qin(3,1) = (1-gamma(2))*F(2);   % Inflow from valve 2 to tank 3 [cm3/s]
qin(4,1) = (1-gamma(1))*F(1);   % Inflow from valve 1 to tank 4 [cm3/s]

% Outflows
h = m./(rho*A);                 % Liquid level in each tank [cm]
qout = a.*sqrt(2*g*h);          % Outflow from each tank [cm3/s]

% Differential equations
xdot = zeros(4,1);
xdot(1,1) = rho*(qin(1,1)+qout(3,1)-qout(1,1));   % Mass balance Tank 1
xdot(2,1) = rho*(qin(2,1)+qout(4,1)-qout(2,1));   % Mass balance Tank 2
xdot(3,1) = rho*(qin(3,1)-qout(3,1));             % Mass balance Tank 3
xdot(4,1) = rho*(qin(4,1)-qout(4,1));             % Mass balance Tank 4
```

# Sensor function

Sensors measuring the level (height) of all tanks

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = \begin{bmatrix} \frac{m_1}{\rho A_1} \\ \frac{m_2}{\rho A_2} \\ \frac{m_3}{\rho A_3} \\ \frac{m_4}{\rho A_4} \end{bmatrix} = g(x, p) \tag{3}$$

Matlab implementation

```
function y = FourTankSystemSensor(x,p)
% FOURTANKSYSTEMSENSOR Level for each tank in the four tank system
%
% Syntax: y = FourTankSystemSensor(x,p)

% Extract states and parameters
m = x;
A = p(5:8,1);
rho = p(12,1);

% Compute level in each tank
rhoA = rho*A;
y = m./rhoA;
```

# Output function

In this case the output is the level (height) in tank 1 and tank 2

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} \frac{m_1}{\rho A_1} \\ \frac{m_2}{\rho A_2} \end{bmatrix} = h(x,p) \tag{4}$$

Matlab implementation

```
function z = FourTankSystemOutput(x,p)
% FOURTANKSYSTEMOUTPUT Level for the lower tanks in the four tank system
%
% Syntax: z = FourTankSystemOutput(x,p)

% Extract states and parameters
m = x(1:2,1);
A = p(5:6,1);
rho = p(12,1);

% Compute level in each tank
rhoA = rho*A;
z = m./rhoA;
```

# Define Simulation Parameters

```
% ------------------------------------------------------------
% Parameters
% ------------------------------------------------------------
a1 = 1.2272      %[cm2] Area of outlet pipe 1
a2 = 1.2272      %[cm2] Area of outlet pipe 2
a3 = 1.2272      %[cm2] Area of outlet pipe 3
a4 = 1.2272      %[cm2] Area of outlet pipe 4

A1 = 380.1327    %[cm2] Cross sectional area of tank 1
A2 = 380.1327    %[cm2] Cross sectional area of tank 2
A3 = 380.1327    %[cm2] Cross sectional area of tank 3
A4 = 380.1327    %[cm2] Cross sectional area of tank 4


gamma1 = 0.45;  % Flow distribution constant. Valve 1
gamma2 = 0.40;  % Flow distribution constant. Valve 2

g = 981;         %[cm/s2] The acceleration of gravity
rho = 1.00;      %[g/cm3] Density of water

p = [a1; a2; a3; a4; A1; A2; A3; A4; gamma1; gamma2; g; rho];
% ------------------------------------------------------------
```

# Simulation Scenario and Simulation

```
% --------------------------------------------------------------
% Simulation scenario
% --------------------------------------------------------------
t0 =    0.0;        % [s] Initial time
tf = 20*60;         % [s] Final time

m10 = 0.0;          % [g] Liquid mass in tank 1 at time t0
m20 = 0.0;          % [g] Liquid mass in tank 2 at time t0
m30 = 0.0;          % [g] Liquid mass in tank 3 at time t0
m40 = 0.0;          % [g] Liquid mass in tank 4 at time t0

F1 = 300;           % [cm3/s] Flow rate from pump 1
F2 = 300;           % [cm3/s] Flow rate from pump 2

x0 = [m10; m20; m30; m40];
u = [F1; F2]
% --------------------------------------------------------------
```

Simulate the system

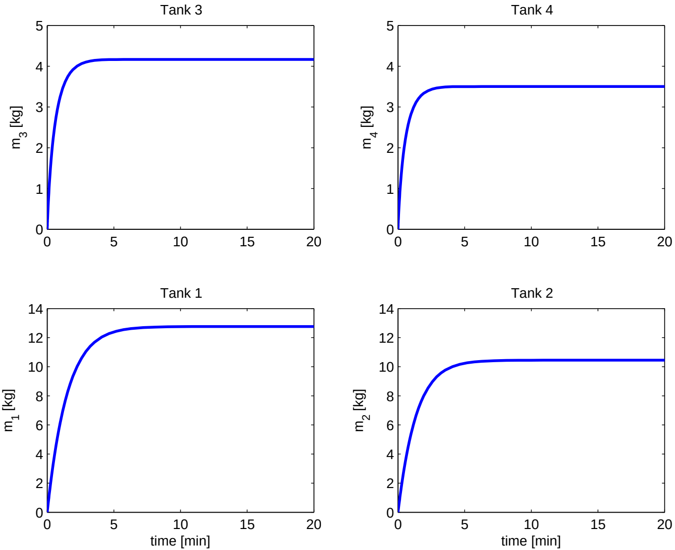
```
% --------------------------------------------------------------
% Compute the solution / Simulate
% --------------------------------------------------------------
% Solve the system of differential equations
[T,X] = ode45(@FourTankSystem,[t0 tf],x0,[],u,p);
```

# Computation of additional variables

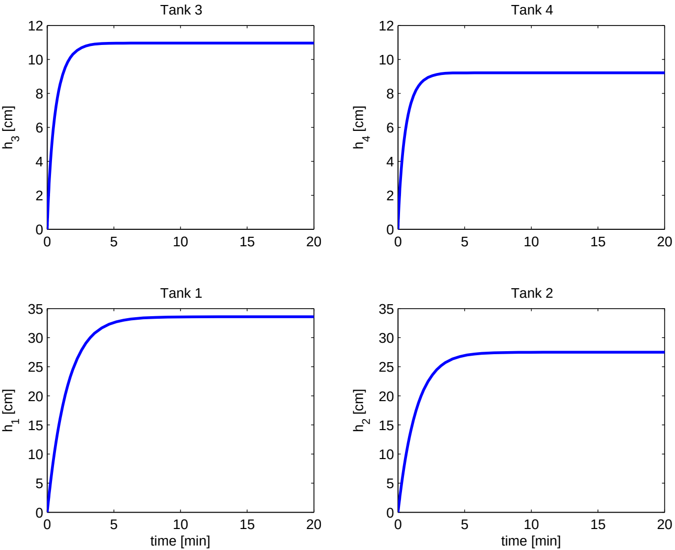
Compute additional variables for plotting

```
% ----------------------------------------------------------------
% help variables
[nT,nX] = size(X);
a = p(1:4,1)';
A = p(5:8,1)';

% Compute the measured variables
H = zeros(nT,nX);
for i=1:nT
H(i,:) = X(i,:)./(rho*A);
end

% Compute the flows out of each tank
Qout = zeros(nT,nX);
for i=1:nT
Qout(i,:) = a.*sqrt(2*g*H(i,:));
end
% ----------------------------------------------------------------
```

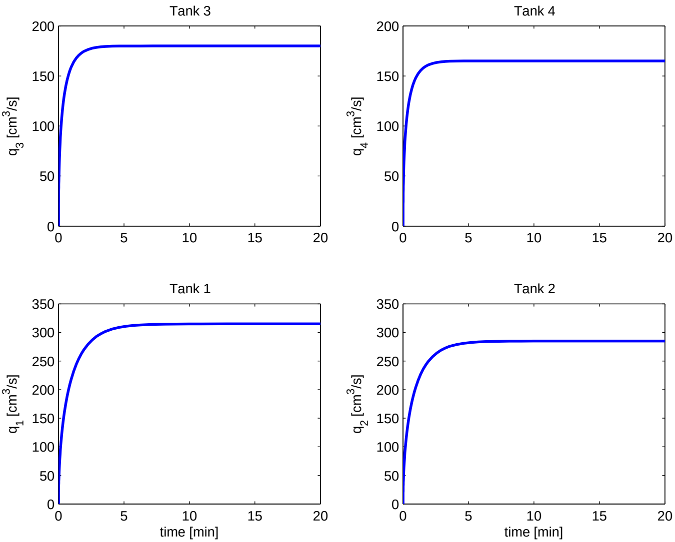
# Masses in the Tanks



# Levels in the Tanks



# Outflow Rates



# Discrete-Time Generic Input-Output Model


$$x_{k+1} = F(x_k, u_k, p) \qquad \text{Discrete-time process model}$$
$$y_k = g(x_k, p) \qquad \text{Sensor function}$$
$$z_k = h(x_k, p) \qquad \text{Output function}$$

Zero-order-hold for MVs

$$u(t) = u_k \qquad t_k \le t < t_{k+1}$$

Continuous-time process model to discrete-time process model

$$F(x_k, u_k, p) = x_k + \int_{t_k}^{t_{k+1}} f(x(t), u_k, p)dt$$

# Difference Equation

The difference equation

$$x_{k+1} = F(x_k, u_k, p)$$

with

$$F(x_k, u_k, p) = x_k + \int_{t_k}^{t_{k+1}} f(x(t), u_k, p) dt$$

can be computed by numerical solution of

$$\frac{dx}{dt}(t) = f(x(t), u_k, p) \qquad x(t_k) = x_k \qquad t_k \leq t < t_{k+1}$$

such that

$$x_{k+1} = x(t_{k+1})$$

# Discrete-Time Simulation using Matlab

The discrete-time model

$$x_{k+1} = F(x_k, u_k, p) \qquad F(x_k, u_k, p) = x_k + \int_{t_k}^{t_{k+1}} f(x(t), u_k, p) dt$$

$$y_k = g(x_k, p)$$

$$z_k = h(x_k, p)$$

may be simulated in Matlab using

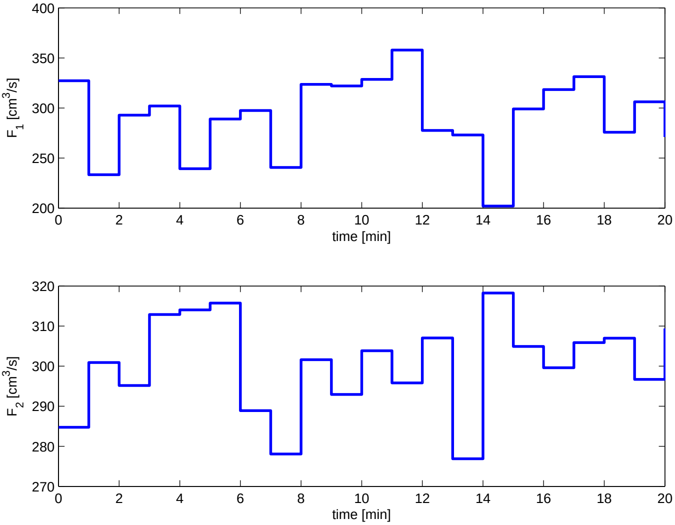
```
for i=0:N
    k=i+1;
    y(:,k) = g(x(:,k),p);
    z(:,k) = h(x(:,k),p);
    [Tk,Xk] = ode45(@f,[t(k) t(k+1)],x(:,k),odeOptions,u(:,k),p);
    x(:,k+1) = Xk(end,:)';
    T = [T; Tk];
    X = [X; Xk];
end
```

# Example - 4-Tank System

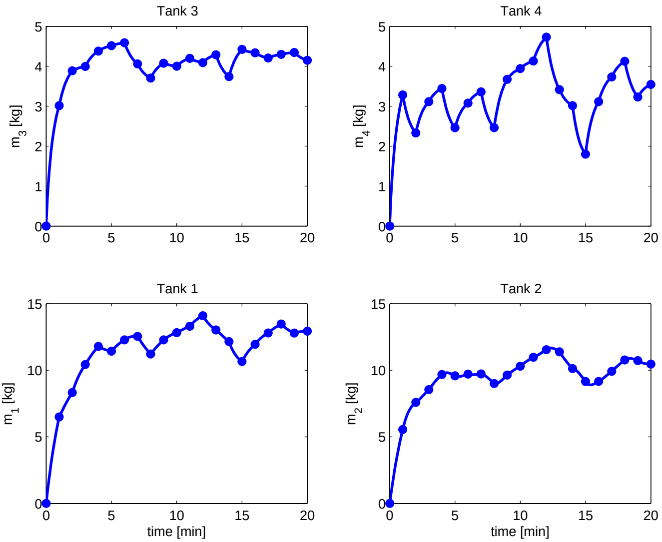

$$x_{k+1} = F(x_k, u_k, p)$$
$$y_k = g(x_k, p)$$
$$z_k = h(x_k, p)$$

```
X = zeros(0,nx);
T = zeros(0,1);

x(:,1) = x0;
for k = 1:N-1
    y(:,k) = FourTankSystemSensor(x(:,k),p); % Sensor function
    z(:,k) = FourTankSystemOutput(x(:,k),p); % Output function

    [Tk,Xk] = ode15s(@FourTankSystem,[t(k) t(k+1)],x(:,k),[],u(:,k),p);
    x(:,k+1) = Xk(end,:)';

    T = [T; Tk];
    X = [X; Xk];
end
k = N;
y(:,k) = FourTankSystemSensor(x(:,k),p);
z(:,k) = FourTankSystemOutput(x(:,k),p);
```

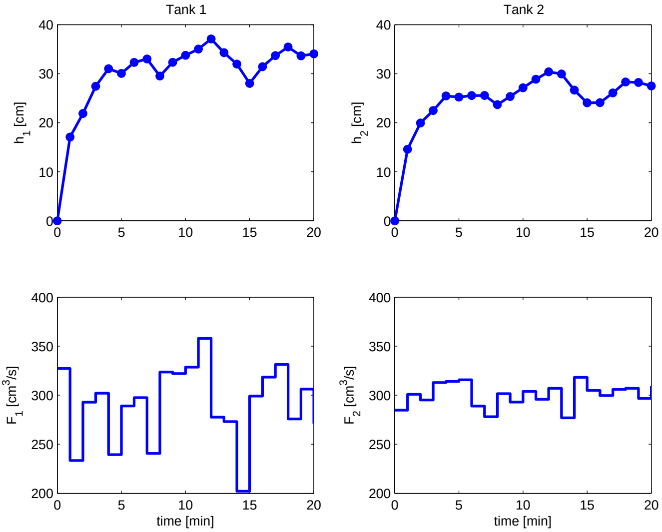
# Flow Rate Scenario



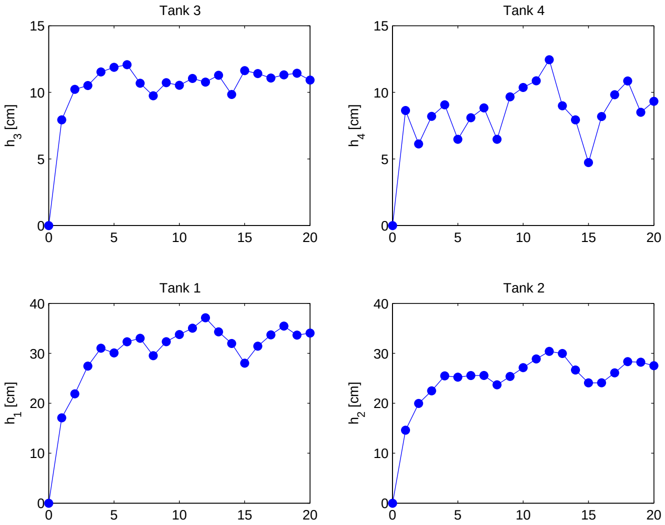
# Quadruple Tank Process - States



# Quadruple Tank Process - Inputs and Outputs



# Quadruple Tank Process - Sensors



# Stochastic Simulation

$$
\begin{aligned}
\boldsymbol{x}_{k+1} &= F(\boldsymbol{x}_k, u_k, p, \boldsymbol{w}_k) && \text{Process model} \\
\boldsymbol{y}_k &= g(\boldsymbol{x}_k, p) + \boldsymbol{v}_k && \text{Sensor function} \\
\boldsymbol{z}_k &= h(\boldsymbol{x}_k, p) && \text{Output function}
\end{aligned}
$$

$$
\begin{aligned}
&\boldsymbol{x}_0 \sim N(\bar{x}_0, P_0) && \text{Unknown initial state} \\
&\boldsymbol{w}_k \sim N_{iid}(0, Q) && \text{Process noise} \\
&\boldsymbol{v}_k \sim N_{iid}(0, R) && \text{Measurement (sensor) noise}
\end{aligned}
$$

# A Stochastic Realization

The stochastic variables

$$\begin{aligned} \boldsymbol{w}_k &\sim N_{iid}(0, Q) \qquad Q = LL' \\ \boldsymbol{e}_k &\sim N_{iid}(0, I) \end{aligned}$$

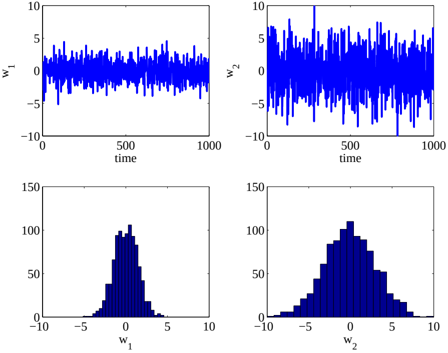
are related by

$$\boldsymbol{w}_k = L\boldsymbol{e}_k$$

As a normal distribution is completely characterized by its means and covariance, this relation can be proved by

$$\begin{aligned} &E\left\{\boldsymbol{w}_k\right\} = E\left\{L\boldsymbol{e}_k\right\} = LE\left\{\boldsymbol{e}_k\right\} = 0 \\ &V\left\{\boldsymbol{w}_k\right\} = \langle \boldsymbol{w}_k, \boldsymbol{w}_k \rangle = \langle L\boldsymbol{e}_k, L\boldsymbol{e}_k \rangle = L \underbrace{\langle \boldsymbol{e}_k, \boldsymbol{e}_k \rangle}_{=I} L' = LL' = Q \end{aligned}$$

# Stochastic Realization in Matlab




$$w_k \sim N_{iid}(0, Q)$$

$$Q = LL'$$

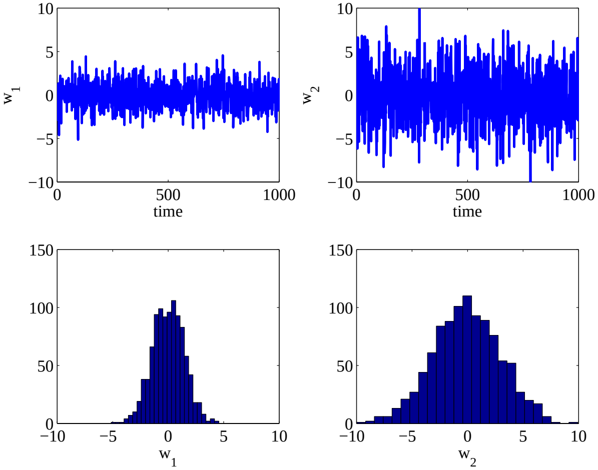
$$e_k \sim N_{iid}(0, I)$$

$$w_k = Le_k$$

```
Q = [2 1; 1 10];
L = chol(Q,'lower');

MySeed = 100;               % You can use any number
randn('state', MySeed);     % Makes the random sequence reproducable
w = L*randn(2,1000);        % Generation of the random sequence
```

# Stochastic Process Noise

$$w_k \sim N_{iid}(0, Q)$$



Process noise

$$\begin{bmatrix} \boldsymbol{F}_1 \\ \boldsymbol{F}_2 \end{bmatrix} = \begin{bmatrix} F_{1s} \\ F_{2s} \end{bmatrix} + \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \end{bmatrix}$$

$$\begin{bmatrix} F_{1s} \\ F_{2s} \end{bmatrix} = \begin{bmatrix} 300 \\ 300 \end{bmatrix} \qquad \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \end{bmatrix} \sim N_{iid}\left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 30^2 & 0 \\ 0 & 10^2 \end{bmatrix} \right)$$

Measurement noise

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \boldsymbol{y}_3 \\ \boldsymbol{y}_4 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \boldsymbol{v}_3 \\ \boldsymbol{v}_4 \end{bmatrix} \qquad \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \boldsymbol{v}_3 \\ \boldsymbol{v}_4 \end{bmatrix} \sim N\left( \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1^2 & 0 & 0 & 0 \\ 0 & 1^2 & 0 & 0 \\ 0 & 0 & 1^2 & 0 \\ 0 & 0 & 0 & 1^2 \end{bmatrix} \right)$$

Outputs

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

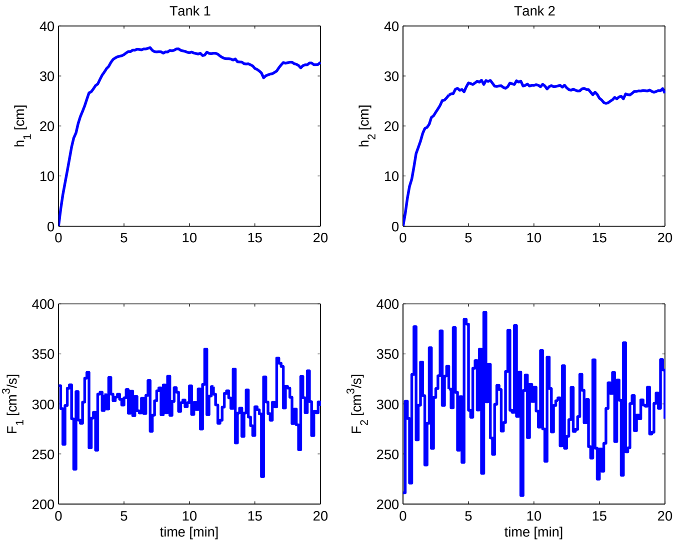
# Stochastic Simulation - Definition of Simulation Scenario

```
t0 =    0.0;        % [s] Initial time
tf = 20*60;         % [s] Final time
Ts = 10;            % [s] Sample Time
t = [t0:Ts:tf]';    % [s] Sample instants
N = length(t);

m10 = 0;          % [g] Liquid mass in tank 1 at time t0
m20 = 0;          % [g] Liquid mass in tank 2 at time t0
m30 = 0;          % [g] Liquid mass in tank 3 at time t0
m40 = 0;          % [g] Liquid mass in tank 4 at time t0

F1 = 300;           % [cm3/s] Flow rate from pump 1
F2 = 300;           % [cm3/s] Flow rate from pump 2

x0 = [m10; m20; m30; m40];
u = [repmat(F1,1,N); repmat(F2,1,N)];

% Process Noise
Q = [20^2 0;0 40^2];
Lq = chol(Q,'lower');
w = Lq*randn(2,N);

% Measurement Noise
R = eye(4);
Lr = chol(R,'lower');
v = Lr*randn(4,N);
```

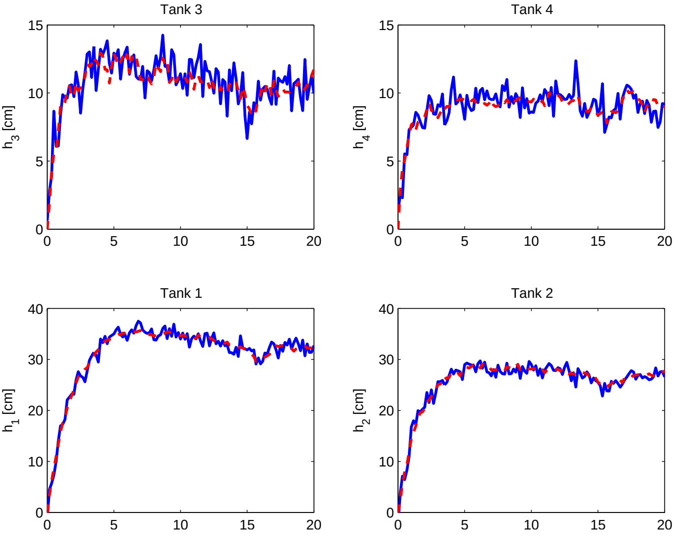
# Stochastic Simulation - Matlab Script

```
nx = 4; nu = 2; ny = 4; nz = 2;
x = zeros(nx,N);
y = zeros(ny,N);
z = zeros(nz,N);

X = zeros(0,nx);
T = zeros(0,1);

x(:,1) = x0;
for k = 1:N-1
    y(:,k) = FourTankSystemSensor(x(:,k),p)+v(:,k); % Sensor function
    z(:,k) = FourTankSystemOutput(x(:,k),p);        % Output function

    [Tk,Xk] = ode45(@FourTankSystem,[t(k) t(k+1)],x(:,k),[],...
                                                      u(:,k)+w(:,k),p);
    x(:,k+1) = Xk(end,:)';

    T = [T; Tk];
    X = [X; Xk];
end
k = N;
    y(:,k) = FourTankSystemSensor(x(:,k),p)+v(:,k); % Sensor function
    z(:,k) = FourTankSystemOutput(x(:,k),p);        % Output function
```

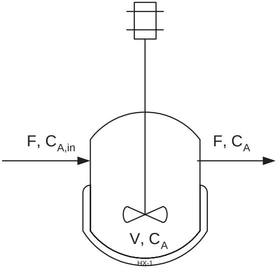
# Stochastic Process Simulation - Input-Output



# Stochastic Process Simulation - Measurements



# Chemical Reaction in a Continuous Stirred Tank Reactor

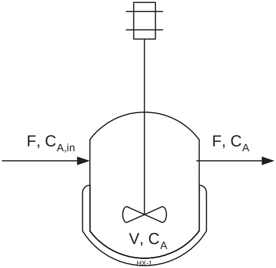



$$A \rightarrow P \qquad r = kC_A$$

The production rate of $A$ is

$$R_A = -r$$

# Chemical Reaction in a Continuous Stirred Tank Reactor




$$
\begin{aligned}
Accumulated &= VC_A(t+\Delta t) - VC_A(t) \\
Influx &= FC_{A,in}(t)\Delta t \\
Outflux &= FC_A(t)\Delta t \\
Generated &= R_A V \Delta t \qquad R_A = R_A(C_A(t))
\end{aligned}
$$

$$A \to P \qquad r = kC_A \qquad R_A = -r$$

$$Accumulated = Influx - Outflux + \overbrace{Produced - Consumed}^{Generated}$$

$$Accumulated = Influx - Outflux + \overbrace{Produced - Consumed}^{Generated}$$

$$\begin{aligned}
Accumulated &= VC_A(t+\Delta t) - VC_A(t)\\
Influx &= FC_{A,in}(t)\Delta t\\
Outflux &= FC_A(t)\Delta t\\
Generated &= R_A V \Delta t \qquad R_A = R_A(C_A(t))
\end{aligned}$$

1. Conservation of mass (mole balance for A):

   $$VC_A(t+\Delta t) - VC_A(t) = FC_{A,in}(t)\Delta t - FC_A(t)\Delta t + R_A(C_A(t))V\Delta t$$

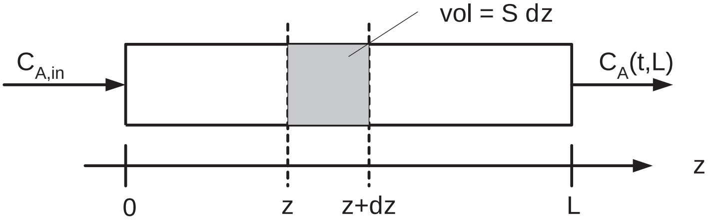
2. Divide by $\Delta t$ and $V$:

   $$\frac{C_A(t+\Delta t) - C_A(t)}{\Delta t} = \frac{F}{V}\left(C_{A,in}(t) - C_A(t)\right) + R_A(C_A(t))$$

3. $\Delta t \to 0$:

   $$\frac{dC_A}{dt}(t) = \frac{F}{V}\left(C_{A,in}(t) - C_A(t)\right) + R_A(C_A(t))$$

# Flow in a Pipe




$$Accumulated = [C_A(t+\Delta t, z) - C_A(t,z)]\, S\Delta z$$
$$Influx = N_A(t,z)S\Delta t$$
$$Outflux = N_A(t, z+\Delta z)S\Delta t$$

# Flow in a Pipe

$$
\begin{aligned}
Accumulated &= \left[C_A(t+\Delta t, z) - C_A(t,z)\right] S\Delta z \\
Influx &= N_A(t,z)S\Delta t \\
Outflux &= N_A(t, z+\Delta z)S\Delta t
\end{aligned}
$$

1. Conservation of mass (mole balance for A):

$$
\overbrace{\left[C_A(t+\Delta t, z) - C_A(t,z)\right] S\Delta z}^{Accumulated} = \overbrace{N_A(t,z)S\Delta t}^{Influx} - \overbrace{N_A(t,z+\Delta z)S\Delta t}^{Outflux}
$$

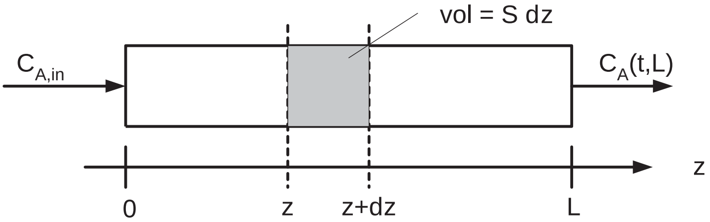
2. Divide by $\Delta t$, $\Delta z$, and $S$:

$$
\frac{C_A(t+\Delta t, z) - C_A(t,z)}{\Delta t} = -\frac{N_A(t, z+\Delta z) - N_A(t,z)}{\Delta z}
$$

3. $\Delta t \rightarrow 0$ and $\Delta z \rightarrow 0$:

$$
\frac{\partial C_A}{\partial t}(t,z) = -\frac{\partial N_A}{\partial z}(t,z)
$$

Differential Equation Model




$$\frac{\partial C_A}{\partial t}(t,z) = -\frac{\partial N_A}{\partial z}(t,z)$$

Initial condition

$$C_A(0,z) = C_{A0}(z) \qquad 0 \le z \le L$$

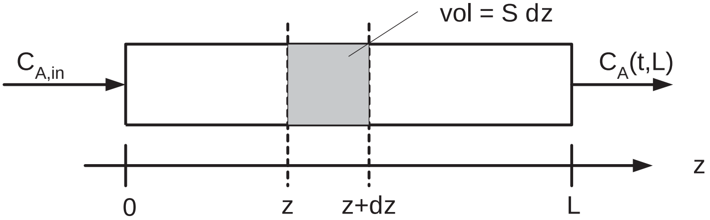
Boundary condition

$$C_A(t,0) = C_{A,in}(t) \qquad t \ge 0$$

Flux (convective flow)

$$N_A(t,z) = vC_A(t,z)$$

# Flow and Chemical Reaction in a Pipe




Chemical Reaction: $A \rightarrow P \quad r = kC_A \quad R_A = -r$

Flux for convective and diffusive flow: $N_A = vC_A + J_A \quad J_A = -D_A \dfrac{\partial C_A}{\partial z}$

$$\begin{aligned}
Accumulated &= \left[C_A(t+\Delta t, z) - C_A(t,z)\right] S\Delta z \\
Influx &= N_A(t,z)S\Delta t \\
Outflux &= N_A(t, z+\Delta z)S\Delta t \\
Generated &= R_A S\Delta z\Delta t
\end{aligned}$$

- Model (mass balance)

$$\frac{\partial C_A(t,z)}{\partial t} = -\frac{\partial N_A(t,z)}{\partial z} + R_A(t,z)$$

- Boundary conditions

$$\begin{aligned} z = 0: &\qquad N_A(t,0) = vC_{A,in} \\ z = L: &\qquad N_A(t,L) = vC_A(t,L) \end{aligned}$$

- Initial condition

$$t = 0: \qquad C_A(0,z) = C_{A0}(z)$$

- Flux

$$N_A(t,z) = \overbrace{vC_A(t,z)}^{\text{convection}} \overbrace{-D_A \frac{\partial C_A(t,z)}{\partial z}}^{\text{diffusion}}$$

- Stoichiometry and kinetics

$$A \rightarrow P \qquad r = kC_A$$

- Production rates

$$R_A = -r$$

- Equidistant spatial discretization

$$z_j = \left(j - \frac{1}{2}\right) \Delta z \qquad \Delta z = \frac{L}{N_z}$$

- Spatial discretization of partial differential equation (method of lines)

$$\frac{dC_{A,j}(t)}{dt} = -\frac{N_{A,j+1/2}(t) - N_{A,j-1/2}(t)}{\Delta z} + R_{A,j}(t) \qquad j = 1, 2, \ldots, N_z$$

- Fluxes

$$\begin{aligned}
N_{A,j+1/2}(t) &= vC_{A,in}(t) && j = 0 \\
N_{A,j+1/2}(t) &= vC_{A,j}(t) - D_A \frac{C_{A,j+1}(t) - C_{A,j}(t)}{\Delta z} && j = 1, 2, \ldots, N_z - 1 \\
N_{A,j+1/2}(t) &= vC_{A,j}(t) && j = N_z
\end{aligned}$$

- Reaction rates

$$r_j(t) = kC_{A,j}(t) \qquad j = 1, 2, \ldots, N_z$$

- Production rates

$$R_{A,j}(t) = -r_j(t) \qquad j = 1, 2, \ldots, N_z$$

## The model

- Equidistant spatial discretization

$$z_j = \left(j - \frac{1}{2}\right)\Delta z \qquad \Delta z = \frac{L}{N_z}$$

- Spatial discretization of partial differential equation (method of lines)

$$\frac{dC_{A,j}(t)}{dt} = -\frac{N_{A,j+1/2}(t) - N_{A,j-1/2}(t)}{\Delta z} + R_{A,j}(t) \qquad j = 1, 2, \ldots, N_z$$

- Fluxes

$$\begin{aligned} N_{A,j+1/2}(t) &= vC_{A,in}(t) && j = 0 \\ N_{A,j+1/2}(t) &= vC_{A,j}(t) - D_A \frac{C_{A,j+1}(t) - C_{A,j}(t)}{\Delta z} && j = 1, 2, \ldots, N_z - 1 \\ N_{A,j+1/2}(t) &= vC_{A,j}(t) && j = N_z \end{aligned}$$

- Reaction rates

$$r_j(t) = kC_{A,j}(t) \qquad j = 1, 2, \ldots, N_z$$

- Production rates

$$R_{A,j}(t) = -r_j(t) \qquad j = 1, 2, \ldots, N_z$$

can be represented as

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \qquad x(t_0) = x_0 \\ y(t) &= g(x(t)) \end{aligned}$$

with $x = [C_{A,1};\, C_{A,2};\, \ldots, C_{A,N_z}]$, $u = C_{A,in}$, and $y = C_{A,out} = C_{A,N_z}$

# Differential Equations

# Ordinary Differential Equation (ODE) System

- System of ordinary differential equations

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} = \begin{bmatrix} f_1(t, x_1(t), x_2(t), \ldots, x_n(t)) \\ f_2(t, x_1(t), x_2(t), \ldots, x_n(t)) \\ \vdots \\ f_n(t, x_1(t), x_2(t), \ldots, x_n(t)) \end{bmatrix} \tag{5}$$

- in compact notation

$$\frac{d}{dt}x(t) = f(t, x(t)) \tag{6}$$

- using the dot-notation

$$\dot{x}(t) = f(t, x(t)) \tag{7}$$

# Classes of Differential Equations

- Ordinary Differential Equation (ODE)

$$\frac{d}{dt}x(t) = f(t, x(t)) \quad \text{or} \quad \dot{x}(t) = f(t, x(t)) \tag{8}$$

- Differential Algebraic Equation (DAE)
  - Generalized Differential Equation

    $$\frac{d}{dt}g(t, y(t)) = f(t, y(t)) \tag{9}$$

    This is identical to the differential algebraic equation

    $$x(t) = g(t, y(t)) \tag{10a}$$
    $$\dot{x}(t) = f(t, y(t)) \tag{10b}$$

  - Differential Algebraic Equation in Semi-Explicit form

    $$G(t, x(t), y(t)) = 0 \tag{11a}$$
    $$\dot{x}(t) = F(t, y(t)) \tag{11b}$$

  - Fully Implicit Differential Algebraic Equations

    $$R(t, x(t), \dot{x}(t)) = 0 \tag{12}$$

# Autonomous vs Non-Autonomous Differential Equation

- Non-Autonomous ODE

$$\dot{x}(t) = f(t, x(t)) \tag{13}$$

- Autonomous ODE

$$\dot{x}(t) = f(x(t)) \tag{14}$$

---

A non-autonmous ODE can always be converted to an autonomous ODE by augmenting the state $y = \begin{bmatrix} x \\ t \end{bmatrix}$. Note that $\dot{t} = \frac{d}{dt}t = 1$. Consequently, the non-autonomous ODE, $\dot{x}(t) = f(t, x(t))$, can be expressed as

$$\dot{y}(t) = \begin{bmatrix} \dot{x} \\ \dot{t} \end{bmatrix} = \begin{bmatrix} f(t, x(t)) \\ 1 \end{bmatrix} = F(y(t)),$$

which is an autonomous ODE (system).

# Classes of Autonomous Differential Equations

- Ordinary Differential Equation (ODE)

$$\frac{d}{dt}x(t) = f(x(t)) \quad \text{or} \quad \dot{x}(t) = f(x(t)) \tag{15}$$

- Differential Algebraic Equation (DAE)
  - Generalized Differential Equation

$$\frac{d}{dt}g(y(t)) = f(y(t)) \tag{16}$$

    This is identical to the differential algebraic equation

$$x(t) = g(y(t)) \tag{17a}$$
$$\dot{x}(t) = f(y(t)) \tag{17b}$$

  - Differential Algebraic Equation in Semi-Explicit form

$$G(x(t), y(t)) = 0 \tag{18a}$$
$$\dot{x}(t) = F(y(t)) \tag{18b}$$

  - Fully Implicit Differential Algebraic Equations

$$R(x(t), \dot{x}(t)) = 0 \tag{19}$$

# Autonomous Differential Equations with a Forcing Function

- Ordinary Differential Equation (ODE)

$$\frac{d}{dt}x(t) = f(x(t), u(t)) \quad \text{or} \quad \dot{x}(t) = f(x(t), u(t)) \tag{20}$$

- Differential Algebraic Equation (DAE)
  - Generalized Differential Equation

    $$\frac{d}{dt}g(y(t)) = f(y(t), u(t)) \tag{21}$$

    This is identical to the differential algebraic equation

    $$x(t) = g(y(t)) \tag{22a}$$
    $$\dot{x}(t) = f(y(t), u(t)) \tag{22b}$$

  - Differential Algebraic Equation in Semi-Explicit form

    $$G(x(t), y(t)) = 0 \tag{23a}$$
    $$\dot{x}(t) = F(y(t), u(t)) \tag{23b}$$

  - Fully Implicit Differential Algebraic Equations

    $$R(x(t), \dot{x}(t), u(t)) = 0 \tag{24}$$

# Autonomous Differential Equations - General Form

- Ordinary Differential Equation (ODE)

$$\frac{d}{dt}x(t) = f(x(t), u(t), p) \quad \text{or} \quad \dot{x}(t) = f(x(t), u(t), p) \tag{25}$$

- Differential Algebraic Equation (DAE)
  - Generalized Differential Equation

$$\frac{d}{dt}g(y(t), p) = f(y(t), u(t), p) \tag{26}$$

    This is identical to the differential algebraic equation

$$x(t) = g(y(t), p) \tag{27a}$$

$$\dot{x}(t) = f(y(t), u(t), p) \tag{27b}$$

  - Differential Algebraic Equation in Semi-Explicit form

$$G(x(t), y(t), p) = 0 \tag{28a}$$

$$\dot{x}(t) = F(y(t), u(t), p) \tag{28b}$$

  - Fully Implicit Differential Algebraic Equations

$$R(x(t), \dot{x}(t), u(t), p) = 0 \tag{29}$$

# Autonomous ODE Systems - Different Representations

- ODE standard form

$$\dot{x}(t) = f(x(t)) \tag{30}$$

- ODE with a forcing function

$$\dot{x}(t) = f(x(t), u(t)) \tag{31}$$

- ODE with parameters

$$\dot{x}(t) = f(x(t), p) \tag{32}$$

- ODE with a forcing function and parameters

$$\dot{x}(t) = f(x(t), u(t), p) \tag{33}$$

## Non-Autonomous ODE Systems - Different Representations

- ODE standard form

$$\dot{x}(t) = f(t, x(t)) \tag{34}$$

- ODE with a forcing function

$$\dot{x}(t) = f(t, x(t), u(t)) \tag{35}$$

- ODE with parameters

$$\dot{x}(t) = f(t, x(t), p) \tag{36}$$

- ODE with a forcing function and parameters

$$\dot{x}(t) = f(t, x(t), u(t), p) \tag{37}$$

# Autonomous ODE and basic numerical solution methods

- Initial value problem (IVP)

$$x(t_0) = \bar{x}_0, \tag{38}$$
$$\dot{x}(t) = f(x(t)), \quad t_0 \leq t \leq t_N = t_f \tag{39}$$

- Discrete times

$$t_0 < t_1 < t_2 < \ldots < t_N = t_f \tag{40}$$

- Time steps

$$\Delta t_k = t_{k+1} - t_k, \qquad k = 0, 1, \ldots, N-1 \tag{41}$$

- Numerical solutions: $x_k = x(t_k)$

$$\text{Initial:} \quad x_0 = \bar{x}_0 \tag{42}$$
$$\text{Explicit:} \quad x_{k+1} - x_k = f(x_k)\Delta t_k, \quad k = 0, 1, \ldots, N-1 \tag{43}$$
$$\text{Implicit:} \quad x_{k+1} - x_k = f(x_{k+1})\Delta t_k, k = 0, 1, \ldots, N-1 \tag{44}$$

# Explicit Solution Methods

- System of ordinary differential equations (ODEs)

$$\dot{x}(t) = f(x(t)) \tag{45}$$

- Integral form

$$x(t_{k+1}) - x(t_k) = \int_{t_k}^{t_{k+1}} f(x(t))dt \tag{46}$$

- Explicit numerical method

$$x_{k+1} - x_k = f(x_k)\Delta t_k \tag{47}$$

with $\Delta t_k = t_{k+1} - t_k$

- Numerical procedure

$$x_{k+1} = x_k + f(x_k)\Delta t_k \tag{48}$$

# Implicit Solution Methods

- System of ordinary differential equations (ODEs)

$$\dot{x}(t) = f(x(t)) \tag{49}$$

- Integral form

$$x(t_{k+1}) - x(t_k) = \int_{t_k}^{t_{k+1}} f(x(t))dt \tag{50}$$

- Implicit numerical method

$$x_{k+1} - x_k = f(x_{k+1})\Delta t_k \tag{51}$$

with $\Delta t_k = t_{k+1} - t_k$

- Numerical procedure: Solve

$$R_k(x_{k+1}) = x_{k+1} - f(x_{k+1})\Delta t_k - x_k = 0 \tag{52}$$

by a Newton method.

# Differential Equations with Discrete Events

Differential equations with discrete events are systems of differential equations where we have one set of model equations, $f_I(t, x(t))$, in one set of the state space, $g(t, x(t)) \geq 0$, and another set of equations, $f_{II}(t, x(t))$, in the other set of the state space, $g(t, x(t)) < 0$. Such a model may be denoted

$$\dot{x}(t) = \begin{cases} f_I(t, x(t)) & g(t, x(t)) \geq 0 \\ f_{II}(t, x(t)) & g(t, x(t)) < 0 \end{cases} \tag{53}$$

where $g(t, x(t))$ is called the discrete event function.
Examples

- Bouncing ball
- Tank with overflow
- Phase equilibrium systems with appearance/disappearance of phases

# Differential Equations and Uncertainty Quantification

- Initial value problem (IVP)

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{54}$$

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{p}), \qquad t_0 \leq t \leq t_f \tag{55}$$

- Uncertain initial conditions

$$\boldsymbol{x}_0 \sim N(\bar{x}_0, P_0) \qquad \Rightarrow \qquad \left\{x_0^i\right\}_{i=1}^{N_x} \tag{56}$$

- Uncertain Parameters

$$p \sim N(\bar{p}, R_p) \qquad \Rightarrow \qquad \left\{p^j\right\}_{j=1}^{N_p} \tag{57}$$

- Solution for uncertain initial conditions and parameters

$$x^{(i,j)}(t_0) = x_0^i \tag{58}$$

$$\dot{x}^{(i,j)}(t) = f(x^{(i,j)}(t), p^j), \qquad t_0 \leq t \leq t_f \tag{59}$$

These $N_x \times N_p$ systems of differential equations may be solved in parallel

# Stochastic Differential Equations (SDEs)

- The stochastic system of differential equations is denoted as

$$d\boldsymbol{x}(t) = \overbrace{f(\boldsymbol{x}(t))dt}^{\text{drift term}} + \overbrace{g(\boldsymbol{x}(t))d\boldsymbol{w}(t)}^{\text{diffusion term}} \tag{60}$$

- Integral form

$$\boldsymbol{x}(t_{k+1}) - \boldsymbol{x}(t_k) = \overbrace{\int_{t_k}^{t_{k+1}} f(\boldsymbol{x}(t))dt}^{\text{Riemann integral}} + \overbrace{\int_{t_k}^{t_{k+1}} g(\boldsymbol{x}(t))d\boldsymbol{w}(t)}^{\text{Ito integral}} \tag{61}$$

- Numerical methods

$$\text{explicit-explicit:} \quad \boldsymbol{x}_{k+1} - \boldsymbol{x}_k = f(\boldsymbol{x}_k)\Delta t_k + g(\boldsymbol{x}_k)\Delta \boldsymbol{w}_k \tag{62}$$

$$\text{implicit-explicit:} \quad \boldsymbol{x}_{k+1} - \boldsymbol{x}_k = f(\boldsymbol{x}_{k+1})\Delta t_k + g(\boldsymbol{x}_k)\Delta \boldsymbol{w}_k \tag{63}$$

- $\{\boldsymbol{w}(t)\}$ is a standard Wiener process. This implies $d\boldsymbol{w}(t) \sim N_{iid}(0, I dt)$ such that $\Delta \boldsymbol{w}_k \sim N_{iid}(0, I\Delta t_k)$

# Linear Systems of Differential Equations

- Scalar (real)

$$\dot{x}(t) = \lambda x(t), \quad x(0) = x_0, \quad \lambda \in \mathbb{R} \tag{64}$$

- Scalar (complex)

$$\dot{x}(t) = \lambda x(t), \quad x(0) = x_0, \quad \lambda \in \mathbb{C} \tag{65}$$

- Multi dimensional

$$\dot{x}(t) = \Lambda x(t), \quad x(0) = x_0, \quad \Lambda \in \mathbb{R}^{n \times n} \tag{66}$$

- Scalar stochastic differential equation (SDE)

$$d\boldsymbol{x}(t) = \lambda \boldsymbol{x}(t) dt + \sigma d\boldsymbol{\omega}(t) \tag{67}$$

- Scalar SDE - state dependent diffusion

$$d\boldsymbol{x}(t) = \lambda \boldsymbol{x}(t) dt + \sigma \boldsymbol{x}(t) d\boldsymbol{\omega}(t) \tag{68}$$

# The Harmonic Oscillator Problem

- Physics – mass-spring system

$$m\ddot{x}(t) = -kx(t) \tag{69}$$

- Mathematical model - 2nd order differential equation

$$\ddot{x}(t) = -\beta x(t) \qquad \beta = \frac{k}{m} \tag{70}$$

- First order system of differential equations

$$x_1 = x \qquad \dot{x}_1 = \dot{x} \qquad \dot{x}_1(t) = x_2(t) \tag{71}$$

$$x_2 = \dot{x} \qquad \dot{x}_2 = \ddot{x} \qquad \dot{x}_2(t) = -\beta x_1(t) \tag{72}$$

- Matrix form ($\dot{x}(t) = Ax(t)$)

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\beta & 0 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{73}$$

# The Van der Pol Oscillator Problem

- Second order differential equation

$$\ddot{y}(t) = \mu(1 - y(t)^2)\dot{y}(t) - y(t) \tag{74}$$

- Equivalent system of first order differential equations

$$\dot{x}_1(t) = x_2(t) \tag{75}$$

$$\dot{x}_2(t) = \mu(1 - x_1(t)^2)x_2(t) - x_1(t) \tag{76}$$

- Version for optimal control problems

$$\dot{x}_1(t) = x_2(t) \tag{77}$$

$$\dot{x}_2(t) = \mu(1 - x_1(t)^2)x_2(t) - x_1(t) + u(t) \tag{78}$$

- SDE version (state independent diffusion)

$$d\boldsymbol{x}_1(t) = \boldsymbol{x}_2(t)dt + \sigma_{11}d\boldsymbol{\omega}_1(t) \tag{79}$$

$$d\boldsymbol{x}_2(t) = \left[\mu(1 - \boldsymbol{x}_1(t)^2)\boldsymbol{x}_2(t) - \boldsymbol{x}_1(t)\right] dt + \sigma_{22}d\boldsymbol{\omega}_2(t) \tag{80}$$

- SDE version (state dependent diffusion)

# Classes of Partial Differential Equations - 1D

- Hyperbolic PDE

$$\frac{\partial}{\partial t}s(t,x) = -\frac{\partial}{\partial x}s(t,x) + f(t,x) \tag{81}$$

- Parabolic PDE

$$\frac{\partial}{\partial t}s(t,x) = \frac{\partial^2}{\partial x^2}s(t,x) + f(t,x) \tag{82}$$

- Elliptic PDE

$$0 = \frac{\partial^2}{\partial x^2}s(t,x) + f(t,x) \tag{83}$$

# Classes of Partial Differential Equations - 2D

- Hyperbolic PDE

$$\frac{\partial}{\partial t}s(t,x,y) = -\left(\frac{\partial}{\partial x}s(t,x,y) + \frac{\partial}{\partial y}s(t,x,y)\right) + f(t,x,y) \tag{84}$$

- Parabolic PDE

$$\frac{\partial}{\partial t}s(t,x,y) = \left(\frac{\partial^2}{\partial x^2}s(t,x,y) + \frac{\partial^2}{\partial y^2}s(t,x,y)\right) + f(t,x,y) \tag{85}$$

- Elliptic PDE

$$0 = \left(\frac{\partial^2}{\partial x^2}s(t,x,y) + \frac{\partial^2}{\partial y^2}s(t,x,y)\right) + f(t,x,y) \tag{86}$$

# Classes of Partial Differential Equations - 3D

- Hyperbolic PDE

$$\frac{\partial}{\partial t}s(t,x,y,z) = -\left(\frac{\partial}{\partial x}s(t,x,y,z) + \frac{\partial}{\partial y}s(t,x,y,z) + \frac{\partial}{\partial z}s(t,x,y,z)\right) + f(t,x,y,z) \tag{87}$$

- Parabolic PDE

$$\frac{\partial}{\partial t}s(t,x,y,z) = \left(\frac{\partial^2}{\partial x^2}s(t,x,y,z) + \frac{\partial^2}{\partial y^2}s(t,x,y,z) + \frac{\partial^2}{\partial z^2}s(t,x,y,z)\right) + f(t,x,y) \tag{88}$$

- Elliptic PDE

$$0 = \left(\frac{\partial^2}{\partial x^2}s(t,x,y,z) + \frac{\partial^2}{\partial y^2}s(t,x,y,z) + \frac{\partial^2}{\partial z^2}s(t,x,y,z)\right) + f(t,x,y,z) \tag{89}$$

# Classes of Partial Differential Equations - General

- Hyperbolic PDE

$$\frac{\partial}{\partial t} s(t,x) = -\nabla \cdot e s(t,x) + f(t,x) \qquad e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tag{90}$$

- Parabolic PDE

$$\frac{\partial}{\partial t} s(t,x) = \nabla \cdot (\nabla s(t,x)) + f(t,x) \tag{91}$$

- Elliptic PDE

$$0 = \nabla \cdot (\nabla s(t,x)) + f(t,x) \tag{92}$$

# Convection-Diffusion-Reaction PDE - 1D

Mass balance

$$\frac{\partial}{\partial t}C_i(t,x) = \overbrace{-\frac{\partial}{\partial x}F_{x,i}(t,x)}^{\text{transport (flux)}} + \overbrace{R_i(t,x)}^{\text{reaction}} \tag{93}$$

Flux

$$F_{x,i}(t,x) = \overbrace{v_x(t,x)C_i(t,x)}^{\text{convection}} + \overbrace{J_{x,i}(t,x)}^{\text{diffusion}} \tag{94}$$

Diffusion (Fick's law)

$$J_{x,i}(t,x) = -D_i(t,x)\frac{\partial}{\partial x}C_i(t,x) \tag{95}$$

Production of component $i$ by reaction

$$R_i(t,x) = \sum_{j\in\mathcal{R}} \nu_{ij} r_j(C(t,x)) \tag{96}$$

# Convection-Diffusion-Reaction PDE - 2D

Mass balance

$$\frac{\partial}{\partial t}C_i(t,x,y) = \overbrace{-\left(\frac{\partial}{\partial x}F_{x,i}(t,x,y) + \frac{\partial}{\partial y}F_{y,i}(t,x,y)\right)}^{\text{transport (flux)}} + \overbrace{R_i(t,x,y)}^{\text{reaction}} \tag{97}$$

Flux

$$F_{x,i}(t,x,y) = \overbrace{v_x(t,x,y)C_i(t,x,y)}^{\text{convection}} + \overbrace{J_{x,i}(t,x,y)}^{\text{diffusion}} \tag{98a}$$

$$F_{y,i}(t,x,y) = \overbrace{v_y(t,x,y)C_i(t,x,y)}^{\text{convection}} + \overbrace{J_{y,i}(t,x,y)}^{\text{diffusion}} \tag{98b}$$

Diffusion (Fick's law)

$$J_{x,i}(t,x,y) = -D_i(t,x,y)\frac{\partial}{\partial x}C_i(t,x,y) \tag{99a}$$

$$J_{y,i}(t,x,y) = -D_i(t,x,y)\frac{\partial}{\partial y}C_i(t,x,y) \tag{99b}$$

Production of component $i$ by reaction

$$R_i(t,x,y) = \sum_{j\in\mathcal{R}} \nu_{ij} r_j(C(t,x,y)) \tag{100}$$

# Convection-Diffusion-Reaction PDE - 3D

Mass balance

$$\frac{\partial}{\partial t} C_i(t,x,y,z) = \overbrace{-\left( \frac{\partial}{\partial x} F_{x,i}(t,x,y,z) + \frac{\partial}{\partial y} F_{y,i}(t,x,y,z) + \frac{\partial}{\partial z} F_{z,i}(t,x,y,z) \right)}^{\text{transport (flux)}} + \overbrace{R_i(t,x,y,z)}^{\text{reaction}} \tag{101}$$

Flux

$$F_{x,i}(t,x,y,z) = \overbrace{v_x(t,x,y,z) C_i(t,x,y,z)}^{\text{convection}} + \overbrace{J_{x,i}(t,x,y,z)}^{\text{diffusion}} \tag{102a}$$

$$F_{y,i}(t,x,y,z) = \overbrace{v_y(t,x,y,z) C_i(t,x,y,z)}^{\text{convection}} + \overbrace{J_{y,i}(t,x,y,z)}^{\text{diffusion}} \tag{102b}$$

$$F_{z,i}(t,x,y,z) = \overbrace{v_z(t,x,y,z) C_i(t,x,y,z)}^{\text{convection}} + \overbrace{J_{z,i}(t,x,y,z)}^{\text{diffusion}} \tag{102c}$$

Diffusion (Fick's law)

$$J_{x,i}(t,x,y,z) = -D_i(t,x,y,z) \frac{\partial}{\partial x} C_i(t,x,y,z) \tag{103a}$$

$$J_{y,i}(t,x,y,z) = -D_i(t,x,y,z) \frac{\partial}{\partial y} C_i(t,x,y,z) \tag{103b}$$

$$J_{z,i}(t,x,y,z) = -D_i(t,x,y,z) \frac{\partial}{\partial z} C_i(t,x,y,z) \tag{103c}$$

Production of component $i$ by reaction

$$R_i(t,x,y,z) = \sum_{j \in \mathcal{R}} \nu_{ij} r_j(C(t,x,y,z)) \tag{104}$$

# Convection-Diffusion-Reaction PDE - General

Mass balance

$$\frac{\partial}{\partial t} C_i(t,x) = \overbrace{-\nabla \cdot F_i(t,x)}^{\text{transport (flux)}} + \overbrace{R_i(t,x)}^{\text{reaction}} \tag{105}$$

Flux

$$F_i(t,x) = \overbrace{v(t,x) C_i(t,x)}^{\text{convection}} + \overbrace{\nabla J_i(t,x)}^{\text{diffusion}} \tag{106}$$

Diffusion (Fick's law)

$$J_i(t,x) = -D_i(t,x) \nabla C_i(t,x) \tag{107}$$

Production of component $i$ by reaction

$$R_i(t,x) = \sum_{j \in \mathcal{R}} \nu_{ij} r_j(C(t,x)) \tag{108}$$

## Schrödinger's Equation - The Wave Function - Quantum Mechanics

The 1-dimensional time-dependent Schrödinger equation is

$$i\hbar\frac{\partial\psi}{\partial t} = \mathcal{H}\psi \tag{109}$$

with $\psi = \psi(t, x)$ being the wave function. The wave function is complex

$$\psi = \psi_{Re} + i\psi_{Im} \tag{110}$$

and $|\psi|^2 = \psi^*\psi$ is a probability density function such that

$$\int_{-\infty}^{\infty} |\psi(t, x)|^2 dx = 1 \tag{111}$$

The Hamiltonian operator is

$$\mathcal{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x) \tag{112}$$

with $V(x)$ being the potential function (in this case independent of time, $t$), $\hbar = h/(2\pi)$ with $h$ being Planck's constant, and $m$ the mass of the particle.

# Maxwell's Equations - Electromagnetism

- The electric flux leaving a volume is proportional to the charge inside
$$\nabla \cdot E = \frac{\rho}{\varepsilon_0} \tag{113}$$
- The total magnetic flux through a closed surface is zero
$$\nabla \cdot B = 0 \tag{114}$$
- The voltage induced in a closed loop is proportional to the rate of change of the magnetic flux the the loop encloses
$$\nabla \times E = -\frac{\partial B}{\partial t} \tag{115}$$
- The magnetic field induced around a closed loop is proportional to the electric current plus displacement current that the loop encloses
$$\nabla \times B = \mu_0 \left( J + \varepsilon_0 \frac{\partial E}{\partial t} \right) \tag{116}$$

# Navier-Stoke Equations - Computational Fluid Dynamics

- Momentum balance (Newtonian fluid, constant $\mu$ and $\rho$)

$$\rho \frac{D\boldsymbol{v}}{Dt} = -\nabla p + \mu \nabla^2 \boldsymbol{v} + \rho \boldsymbol{g} \tag{117}$$

$$\frac{D\boldsymbol{v}}{Dt} = \frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{v} \tag{118}$$

- Energy balance (Newtonian fluid, constant $\rho$ and $k$)

$$\rho \hat{C}_p \frac{DT}{Dt} = k \nabla^2 T + \mu \Phi_v \tag{119}$$

$$\frac{DT}{Dt} = \frac{\partial T}{\partial t} + \boldsymbol{v} \cdot \nabla T \tag{120}$$

- Mass balance (binary mixture, constant $\rho D_{AB}$)

$$\rho \frac{D\omega_A}{Dt} = \rho D_{AB} \nabla^2 \omega_A + r_A \tag{121}$$

$$\frac{D\omega_A}{Dt} = \frac{\partial \omega_A}{\partial t} + \boldsymbol{v} \cdot \nabla \omega_A \tag{122}$$

# Thermodynamic Process Systems and Phase Equilibrium

- Mass and energy conservation (1st law of thermodynamics)

$$\dot{n}_i = f_i - l_i(T, P, n_l) - v_i(T, P, n_v) \qquad i = 1, \ldots, N_C \tag{123}$$

$$\dot{U} = H_F - H_L(T, P, n^l) + H_V(T, P, n^v) + Q + W_s \tag{124}$$

- Vapor-liquid equilibrium (2nd law of thermodynamics)

$$\max_{T,P,n^l,n^v} \; S = S^l(T, P, n^l) + S^v(T, P, n^v) \tag{125}$$

$$s.t. \qquad U^l(T, P, n^l) + U^v(T, P, n^v) = U \tag{126}$$

$$V^l(T, P, n^l) + V^v(T, P, n^v) = V \tag{127}$$

$$n_i^l + n_i^v = n_i \qquad i = 1, \ldots, N_C \tag{128}$$

- Mathematical representation as semi-explicit DAE system

$$\dot{x} = F(y) \tag{129}$$

$$G(x, y, z) = 0 \tag{130}$$

Semi-explicit DAE systems occurs for equilibrium governed processes

# Problem Types

# Basic Problem Types (ODE Systems)

- Initial Value Problem (IVP)

$$x(t_a) = x_a \tag{131a}$$
$$\dot{x}(t) = f(x(t)) \quad t_a \le t \le t_b \tag{131b}$$

Compute $x_b = x(t_b)$ - that satisfies (131).

- Boundary Value Problem (BVP)

$$x(t_a) = x_a(p) \tag{132a}$$
$$\dot{x}(t) = f(x(t)) \quad t_a \le t \le t_b \tag{132b}$$
$$x(t_b) = x_b(p) \tag{132c}$$

Compute $p$ - that satisfies (132).

# Advanced Problem Types (ODE Systems) - IVP

- Mean-Covariance Pair [IVP]
  From the initial conditions, $\hat{x}_k(t_k) = \hat{x}_{k|k}$ and $P_k(t_k) = P_{k|k}$, solve

$$\frac{d}{dt}\hat{x}_k(t) = f(\hat{x}_k(t)) \tag{133a}$$

$$\frac{d}{dt}P_k(t) = \left[\frac{\partial f}{\partial x}(\hat{x}_k(t))\right] P_k(t) + P_k(t)\left[\frac{\partial f}{\partial x}(\hat{x}_k(t))\right]' + \sigma\sigma' \tag{133b}$$

  in the interval $t \in [t_k,\, t_{k+1}]$ to obtain

  $\hat{x}_{k+1|k} = \hat{x}_k(t_{k+1})$ and $P_{k+1|k} = P_k(t_{k+1})$.
- Sensitivities [IVP]

$$\dot{x}(t) = f(x(t), p), \qquad x(t_a) = x_a(p) \tag{134a}$$

$$\dot{S}_p(t) = \frac{\partial f}{\partial x}(x(t), p)S_p(t) + \frac{\partial f}{\partial p}(x(t), p), \quad S_p(t_a) = \frac{\partial}{\partial p}x_a(p) \tag{134b}$$

- Lyapunov Differential Equation [IVP]

$$\dot{P}(t) = A'P(t) + P(t)A + Q, \quad P(t_a) = P_a \tag{135}$$

- Riccati Differential Equation [IVP]

$$\dot{P}(t) = A'P(t) + P(t)A + Q - P(t)BR^{-1}B'P(t), \quad P(0) = P_N \tag{136}$$

  Compute $P_0 = P(T)$

# Optimal Control Problem (OCP)

- Bolza form

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \phi = \int_{t_a}^{t_b} g(x(t), u(t))dt + h(x(t_b)) && \text{(137a)} \\
s.t. \quad & x(t_a) = x_a && \text{(137b)} \\
& \dot{x}(t) = f(x(t), u(t)) \qquad t_a \le t \le t_b && \text{(137c)}
\end{aligned}
$$

- Lagrange form

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \phi = \int_{t_a}^{t_b} g(x(t), u(t))dt && \text{(138a)} \\
s.t. \quad & x(t_a) = x_a && \text{(138b)} \\
& \dot{x}(t) = f(x(t), u(t)) \qquad t_a \le t \le t_b && \text{(138c)}
\end{aligned}
$$

- Mayer form

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \phi = h(x(t_b)) && \text{(139a)} \\
s.t. \quad & x(t_a) = x_a && \text{(139b)} \\
& \dot{x}(t) = f(x(t), u(t)) \quad t_a \le t \le t_b && \text{(139c)}
\end{aligned}
$$

# Optimal Control Problem (OCP) in Bolza Form

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \phi = \int_{t_a}^{t_b} g(x(t), u(t))dt + h(x(t_b)) && \text{(140a)} \\
s.t. \quad & x(t_a) = x_a && \text{(140b)} \\
& \dot{x}(t) = f(x(t), u(t)) \qquad t_a \le t \le t_b && \text{(140c)}
\end{aligned}
$$

- The Hamiltonian

$$
\begin{aligned}
\mathcal{H} &= \mathcal{H}(x(t), u(t), \lambda(t)) \\
&= g(x(t), u(t)) + \lambda(t)' f(x(t), u(t))
\end{aligned} \tag{141}
$$

- The Euler-Lagrange equations
  System of differential equations that defines the optimal solution

# Numerical Methods (Solvers)

# Explicit Solvers - Nonstiff Systems

- Initial value problem (IVP)

$$x(t_0) = \bar{x}_0 \tag{142}$$

$$\dot{x}(t) = f(x(t)) \tag{143}$$

- Explicit Euler method

$$x_0 = \bar{x}_0 \tag{144}$$

$$x_{k+1} = x_k + f(x_k)\Delta t_k \tag{145}$$

# Implicit Solvers - Stiff Systems

- Initial value problem (IVP)

$$x(t_0) = \bar{x}_0 \tag{146}$$
$$\dot{x}(t) = f(x(t)) \tag{147}$$

- Implicit Euler method

$$x_0 = \bar{x}_0 \tag{148}$$
$$x_{k+1} = x_k + f(x_{k+1})\Delta t_k \tag{149}$$

- Solve

$$R_k(x_{k+1}) = x_{k+1} - f(x_{k+1})\Delta t_k - x_k = 0 \tag{150}$$

using a variant of Newton's method (Newton-Raphson)

# Newton's Method

- Nonlinear system of equations

$$R_k(x_{k+1}) = x_{k+1} - f(x_{k+1})\Delta t_k - x_k = 0 \tag{151}$$

- Iteration matrix and Jacobian

$$M_k = \frac{\partial R_k}{\partial x_{k+1}} = I - J(x_{k+1})\Delta t_k, \quad J(x_{k+1}) = \frac{\partial f}{\partial x}(x_{k+1}) \tag{152}$$

- Linear system ($Ax = b$)

$$M_k \Delta x_{k+1} = -R_k(x_{k+1}) \tag{153}$$

- Updated iterate

$$x_{k+1} := x_{k+1} + \Delta x_{k+1} \tag{154}$$

- Convergence

$$\|R_k(x_{k+1})\| \leq \epsilon \tag{155}$$

# Methods for Dense Matrices - Dense LU Factorization

- Linear system of equations

$$Ax = b \tag{156}$$

where $A \in \mathbb{R}^{n \times n}$ is a dense matrix

- LU factorization with pivoting (interchange of rows)

$$PA = LU \qquad \text{Computational complexity: } \mathcal{O}(n^3) \tag{157}$$

- Back substitution

$$PAx = L\overbrace{Ux}^{=y} = Pb = \bar{b} \tag{158}$$

$$\text{Interchange rows: } \bar{b} = Pb \qquad \mathcal{O}(n) \tag{159}$$

$$\text{Solve for } y\text{: } \quad Ly = \bar{b} \qquad \mathcal{O}(n^2) \tag{160}$$

$$\text{Solve for } x\text{: } \quad Ux = y \qquad \mathcal{O}(n^2) \tag{161}$$

# Direct Methods for Sparse Matrices - Sparse LU Factorization

- Matlab `sparse`
- Sparse matrix software libraries

# Newton Method Variants

- The exact Newton method
- The inexact Newton method
- Sequential substitution

# Iterative Methods for Sparse Matrices

- Large-scale systems (typical discretization of PDEs)
- GMRES with incomple LU (ILU) preconditioner

# Implementation (Software)

# Programming Languages

High-level programming languages:

- Matlab
- Octave
- Python
- R
- Julia

Low-level programming languages

- C
- Fortran

Object-oriented programming languages

- C++
- Java
- C#

Web-oriented languages

- Javascript

# Modeling Environments

- Simulink (Mathworks)
- gProms (PSEnterprise)
- Modelica (OpenModelica – `https://openmodelica.org/`)

# Software Libraries for Solution of Differential Equations

- Matlab
  - Nonstiff ODE (`ode45, ode23, ode113`)
  - Stiff ODE (`ode15s, ode23s`)
  - Stiff DAE (`ode23t, ode23tb`)
  - Fully Implicit DAE (`ode15i`)
- SUNDIALS
  SUite of Nonlinear and DIfferential ALgebraic equation Solvers
  http://computation.llnl.gov/projects/sundials
- NETLIB (www.netlib.org)

# Summary and Conclusion

# Summary and Conclusion

- Non-stiff systems of differential equations
  - Typical low dimensional ODEs
  - Explicit solver for the system of differential equations (ODEs)
  - Numerical implementation straightforward
- Stiff systems of differential equations
  - Implicit solver for the system of differential equations (ODEs, DAEs, spatially discretized PDEs in the MOL)
  - Some Newton based solver (exact, inexact) for the implicit equations
  - Linear system of equations must be solved (Dense, Sparse direct, Sparse iterative)