Time Domain Simulation and Optimization for Design

Lecture 1: Time Domain Simulation

S.D. Sudhoff

Spring 2019

Reference Material

[1] L.O. Chua, P.M. Lin, *Computer Aided Analysis of Electric Circuits: Algorithms & Computation Techniques*, Prentice-Hall, 1975.

[2] Walter Gautschi, *Numerical Analysis*, *Second Edition*, Birkhäuser, 2012.

Resistor-Companion Based Circuit Simulation

Some Definitions

- Model – A description of a system used to predict it's behavior

- Simulation – The process of predicting a system's behavior from the model

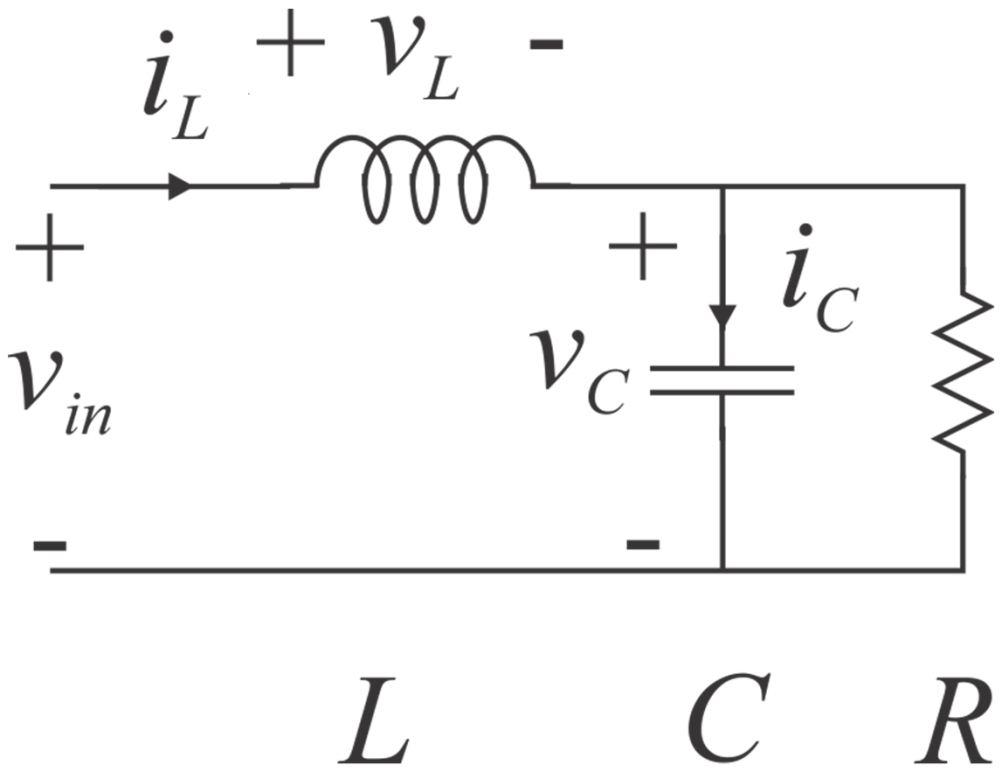
Time-Domain Simulation

- Resistor-Companion (Circuit) Simulation
  - SPICE, PSPICE, SABRE,PSCAD

- State-Variable (System) Simulation
  - Matlab, Simulink, ACSL, EASY5, Dymola, PLECS

Resistor-Companion Simulation [1]

- Consider the system



[1] L.O. Chua, P.M. Lin, *Computer Aided Analysis of Electric Circuits: Algorithms & Computation Techniques,* Prentice-Hall, 1975.

Resistor-Companion Simulation

- The inductor

Resistor-Companion Simulation

- The capacitor

Resistor-Companion Simulation

- The resistor

Resistor-Companion Simulation

- Putting it all together

Resistor-Companion Simulation

- Putting it all together

Resistor-Companion Simulation

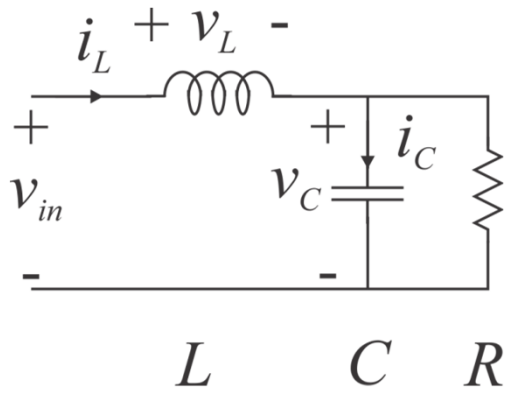
- Putting it all together

State Variable Based Models

State Variables

- Definition: $\mathbf{x} \in \mathbb{R}^O$ is a minimum set of variables that given their value at $t = t_0$ and the system inputs as a function of time can be used to predict system performance for $t > t_0$.

- Example:



State Variable Based Models

- Ordinary Differential Equations (ODEs)
- Standard explicit initial value problem

$$\frac{d\mathbf{x}}{dt} = p\mathbf{x} = \mathrm{f}(t, \mathbf{x})$$

$$t \in \left[t_0, t_f \right]$$

$$\mathbf{x}(t_0) = \mathbf{x}_0$$

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

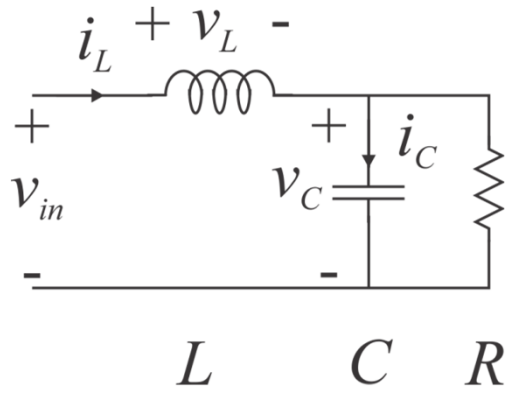
- For linear time-invariant systems

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}(t)$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}(t)$$

State Variable Based Models

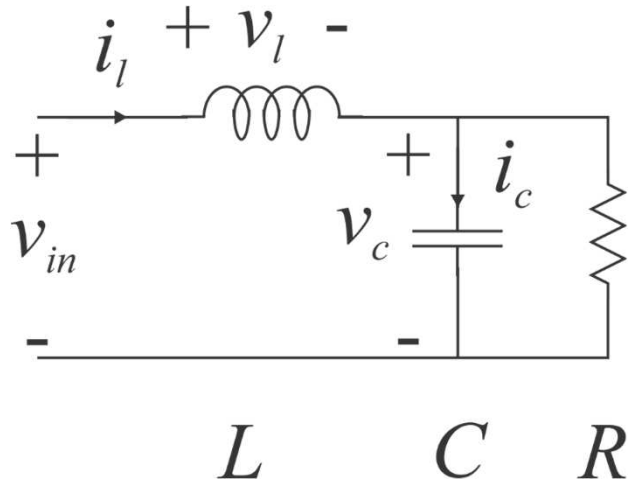
- Consider our example



State Variable Based Models

Uniqueness of State Variables

- Again consider our example



On the “Minimal Set of Variables”

- Consider the following system

Conditions for Solving ODEs

Existence and Solution of ODE's

- Theorem 5.3.1 (Paraphrased from [2])
- Suppose f(t, **x**) is continuous in $t \in \left[t_0, t_f \right]$ and satisfies the uniform Lipschitz condition

$$\left\| \mathrm{f}(t, \mathbf{x}) - \mathrm{f}(t, \mathbf{x}^*) \right\| \leq L \left\| \mathbf{x} - \mathbf{x}^* \right\|, \quad t \in \left[t_0, t_f \right], \quad x, x^* \in \mathbb{R}^n$$

- Then the initial value problem has a unique solution for arbitrary $\mathbf{x}(t_0)$ and $t \in \left[t_0, t_f \right]$

[2] Walter Gautschi, *Numerical Analysis*, *Second Edition*, Birkhäuser, 2012.

Lipschitz Continuity

- Example 1

Lipschitz Continuity

- Example 1 continued

Lipschitz Continuity

- Example 2

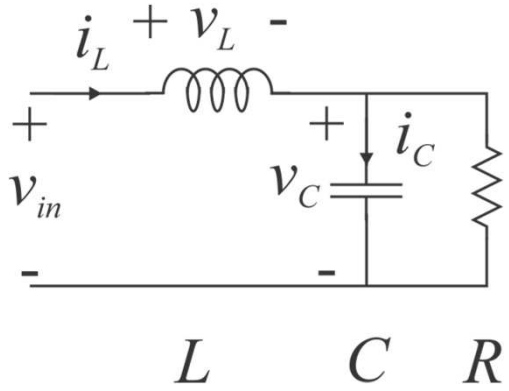
Life in a Non-Lipschitz Universe

- Suppose f(t,**x**) is Lipschitz in convex compact domain $D \subset \mathbb{R}^o$
- Then …

Function Formulation

Mathematical View

- Consider our inductor example



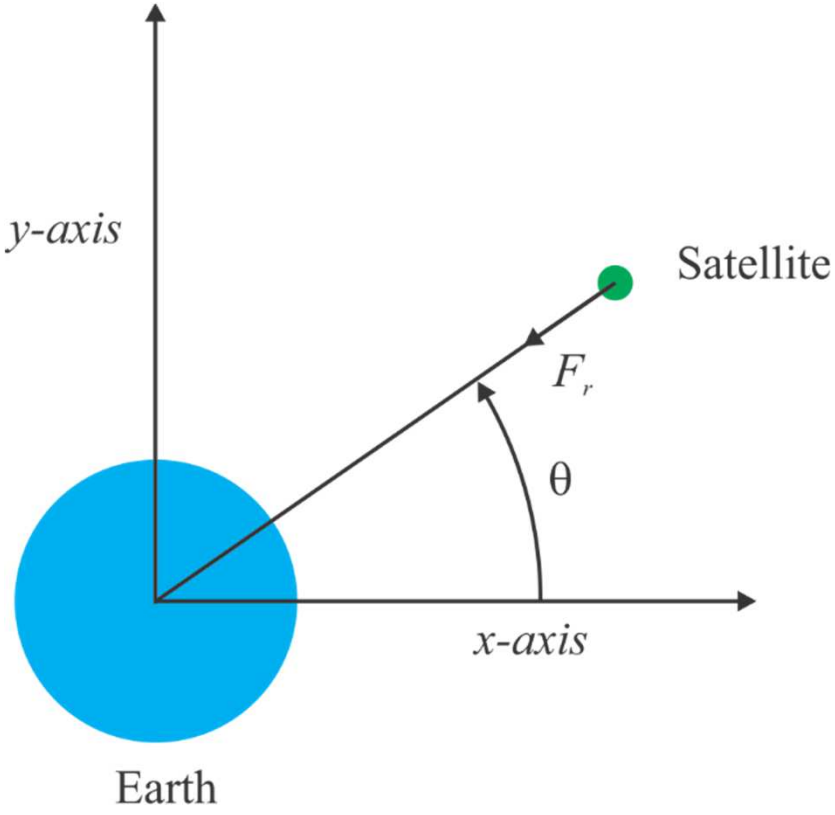
- Let

$$\mathbf{x} = \begin{bmatrix} x^1 \\ x^2 \end{bmatrix} = \begin{bmatrix} i_L \\ v_C \end{bmatrix}$$

$$\mathrm{f}(t, \mathbf{x}) = \begin{bmatrix} \frac{1}{L}(v_{in}(t) - x^2) \\ \frac{1}{C}(x^1 - x^2 / R) \end{bmatrix}$$

Coding View

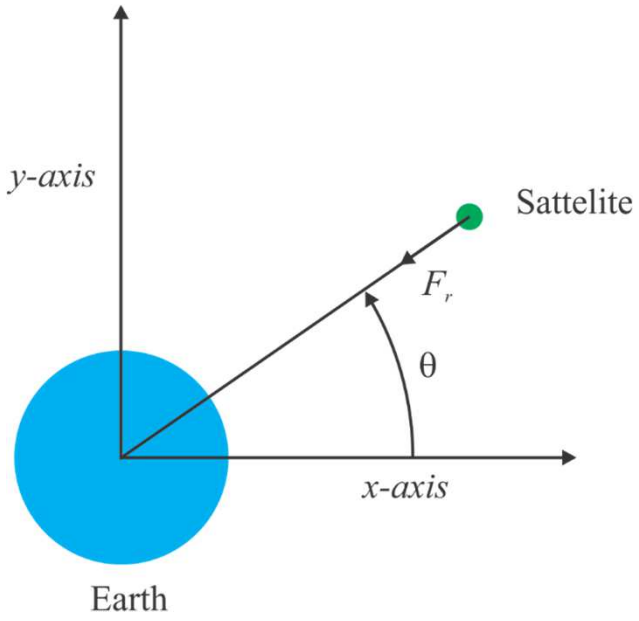
- Consider the orbit of a satellite



$$\mathrm{x} = \begin{bmatrix} x^1 \\ x^2 \\ x^3 \\ x^4 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$$

Coding View

- The physics



$$\theta = \text{angle}(p_x + jp_y)$$

$$r = \sqrt{p_x^2 + p_y^2}$$

$$F_r = \frac{Gm_e m_s}{r^2}$$

$$F_x = -F_r \cos\theta$$

$$F_y = -F_y \sin\theta$$

$$\frac{dp_x}{dt} = v_x$$

$$\frac{dp_y}{dt} = v_y$$

$$\frac{dv_x}{dt} = \frac{1}{M_s} F_x$$

$$\frac{dv_y}{dt} = \frac{1}{M_s} F_y$$

Coding View

```
function [px]=satellite_model(t,x,P)
% This routine contains the dynamics of a satellite whos
% mass is negligible compared to the earth
%
% [px] = satellite_model(t,x,P);
%
% Inputs:
% t        = time {not actually used} (s)
% x        = state vector
%  x(1)    = x-component of earth position (m)
%  x(2)    = y-component of earth position (m)
%  x(3)    = x-component of earth velocity (m/s)
%  x(4)    = y-component of earth velocity (m/s)
% P        = structure with parameters
%  P.me    = mass of earth (kg)
%  P.ms    = mass of satellite (kg)
%  P.G     = gravitational constant (N*(m/kg)^2)
%  P.px0   = x-component of initial satellite position(m)
%  P.py0   = y-component of initial satellite position(m)
%  P.vx0   = x-component of initial satellite velocity(m/s)
%  P.vy0   = y-component of initial satellite velocity(m/s)
%
% Outputs:
% px       = time derivative of state vector (see definition of x)
%
```

Coding View

```
% Internal:
% px        = x-component of satellite position (m)
% py        = y-component of satellite position (m)
% vx        = x-component of satellite velocity (m/s)
% vy        = y-component of satellite velocity (m/s)
% theta     = angular position of satellite (rad)
% r2        = square of earth-satellite distance {center to center} (m)
% f         = magnitude of earth-satellite gravitational force (N)
% fx        = gravitation force on satellite in x-direction (N)
% fy        = gravitation force on satellite in y-direction (N)
% ppx       = time derivative of x-component of satellite position (m/s)
% ppy       = time derivative of y-component of satellite position (m/s)
% pvx       = time derivative of x-component of satellite velocity (m/s^2)
% pvy       = time derivative of y-component of satellite velocity (m/s^2)
%
% Written by S.D. Sudhoff
%   School of Electrical and Computer Engineering
%   1285 Electrical Engineering Building
%   West Lafayette, IN 47907-1285
%   Phone: 765-494-3246
%   Fax: 765-494-0676
%   E-mail: sudhoff@ecn.purdue.edu
```

Coding View

```
% decompose state vector
px = x(1);
py = x(2);
vx = x(3);
vy = x(4);

% compute the gravitational force of satellite
theta=angle(px+1j*py);
r2=px^2+py^2;
fr=P.G*P.me*P.ms/r2;
fx=-fr*cos(theta);
fy=-fr*sin(theta);

% compute the derivative of states associated with asteroid
ppx=vx;
ppy=vy;
pvx=fx/P.ms;
pvy=fy/P.ms;

% pack the state derivatives
px=[ppx ppy pvx pvy]';

end
```

Sample Call

```
% This script sets parameter calls the satellite model
%
% Written by S.D. Sudhoff
%   School of Electrical and Computer Engineering
%   1285 Electrical Engineering Building
%   West Lafayette, IN 47907-1285
%   Phone: 765-494-3246
%   Fax: 765-494-0676
%   E-mail: sudhoff@ecn.purdue.edu

% model parameters
P.me    = 5.972e24;     % mass of earth (kg)
P.ms    = 1000;         % mass of satellite (kg)
P.G     = 6.674e-11;    % gravitational constant (N*(m/kg)^2)

vx    = 5e2;            % x-component of satellite velocity(m/s)
vy    = 30e2;           % y-component of satellite velocity(m/s)
px   = 4e7;             % x-component of satellite position(m)
py   = 0;               % y-component of satellite position(m)

x=[px py vx vy];
dxdt=satellite_model(0,x,P);
dxdt
```

Introduction to Simulation Engines: Forward Euler

State Model Solution Methods

- Analytical Approximation Methods vs **Discrete-Variable Methods**
- **One-Step Methods** vs Multistep Methods
- **Explicit Methods** vs **Implicit Methods (A-Stable)**

- Discrete-Variable, One Step, Explicit Methods
  - Forward Euler
  - Runga-Kutta
  - Many Others
- Discrete-Variable, One Step, Implicit Methods
  - Backward Euler
  - Trapezoidal
  - Many Others

Forward Euler

- Consider

$$\frac{d\mathbf{x}}{dt} = \mathrm{f}(t, \mathbf{x})$$

Quick Example

- Suppose we have

$$px^1 = -x^1 + 0.1\left(x^2\right)^2$$

$$px^2 = -x^2 + 0.1x^1$$

- And that

$$x^1(t_1) = 5$$

$$x^2(t_1) = 1$$

- Find $\mathbf{x}(t_2)$ where $t_2 = t_1 + h$; $h = 0.1$ s

Quick Example (Continued)

Observations from Euler's Method

- To predict a future value of state, we needed to find the present value of the time derivative of the state variables, based on what we know – that is the present value of state variables and present value of input variables.

- This is true of all other one-step integration algorithms as well

The Runga-Kutta Algorithm

The Runge-Kutta Algorithm

- The fourth-order implementation (RK4)

$$\mathbf{k}_1 = \mathrm{f}(t_n, \mathbf{x}_n)$$

$$\mathbf{k}_2 = \mathrm{f}\left(t_n + \frac{1}{2}h, \mathbf{x}_n + \frac{1}{2}h\mathbf{k}_1\right)$$

$$\mathbf{k}_3 = \mathrm{f}\left(t_n + \frac{1}{2}h, \mathbf{x}_n + \frac{1}{2}h\mathbf{k}_2\right)$$

$$\mathbf{k}_4 = \mathrm{f}\left(t_n + h, \mathbf{x}_n + h\mathbf{k}_3\right)$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

$$t_{n+1} = t_n + h$$

odefsrk.m

```
function [t,y]=odefsrk(fhandle,tspan,yic,par,maxt)
% This routine solves a ordinary differential equation
% using a 4th order Runga-Kutta method.
%
% [t,y] = odefsrk(fhandle,tspan,yic,par,maxt);
%
% Inputs:
%
% fhandle  = a handle to the function whose output is the time derivative
%            of the system model.  The inputs to this function are time,
%            state, and parameter vales.
% tspan    = a vector whose elements describe at which point in time the
%            solution is sought
% yic      = a vector which describes the initial condition of the system
%            being simulated
% par      = a structure which contains data or parameters needed to
%            evaluate the time derivative of the state variables
% maxt     = the maximum allowed time step
```

odefsrk.m

```
%
% Outputs:
%
% t         = a vector of times at which the state vector has been found
% y         = a matrix wherein each row contains the state vector at a
%             given time.  Each column is the time history of a particular
%             state
%
% Internal:
%
% Nt        = Number of points at which to calculate the state vector
% Ns        = Number of state variables
% i         = index variable for time
% deltat    = time span between current interval being simulated (s)
% nsteps    = number of integration steps required on interval
% h         = integration time step (s)
```

odefsrk.m

```
% ho2       = half of integration time step (s)
% y1        = state vector at beginning of integration step
% t1        = time at the beginning of the integration step (s)
% k1        = time derivative of state vector at beginning of integration
%             step (s)
% y2        = 1st est. of state vector in center of integration step
% k2        = time derivative of y2 at center of integration step
% y3        = 2nd est. of state vector in center of integration step
% k3        = time derivative of y3 at center of integration step
% y4        = estimate of state vector at end of integration step
% k4        = time derivative of state vector at end of integration step
% k         = weighted average value of time derivative of integration
%             step
%
% Written by S.D. Sudhoff
%   School of Electrical and Computer Engineering
%   1285 Electrical Engineering Building
%   West Lafayette, IN 47907-1285
%   Phone: 765-494-3246
%   Fax: 765-494-0676
%   E-mail: sudhoff@ecn.purdue.edu
```

odefsrk.m

```
% determine number of times at which state vector is recorded
Nt=length(tspan);

% determine the number of states
Ns=length(yic);

% assign the time vector, and initialize y matrix
t=tspan;
y=zeros(Nt,Ns);

% first value row of y matrix is determined in accordance with
% initial condition
y(1,:)=yic;

% iterate one communication interval at a time.  A given
communication
% interval may require several time steps
for i=2:Nt;
```

odefsrk.m

```
% find integration information for this communication interval
  deltat=tspan(i)-tspan(i-1);
  nsteps=ceil(deltat/maxt);
  h=deltat/nsteps;
  ho2=0.5*h;
  y1=y(i-1,:)';
  t1=tspan(i-1);

  % now perform enough integration steps to get through communication
  % interval
  for j=1:nsteps

     % perform Runga-Kutta update
     t2=t1+ho2;
     t3=t1+h;

     k1=fhandle(t1,y1,par);

     y2=y1+ho2*k1;
     k2=fhandle(t2,y2,par);
```

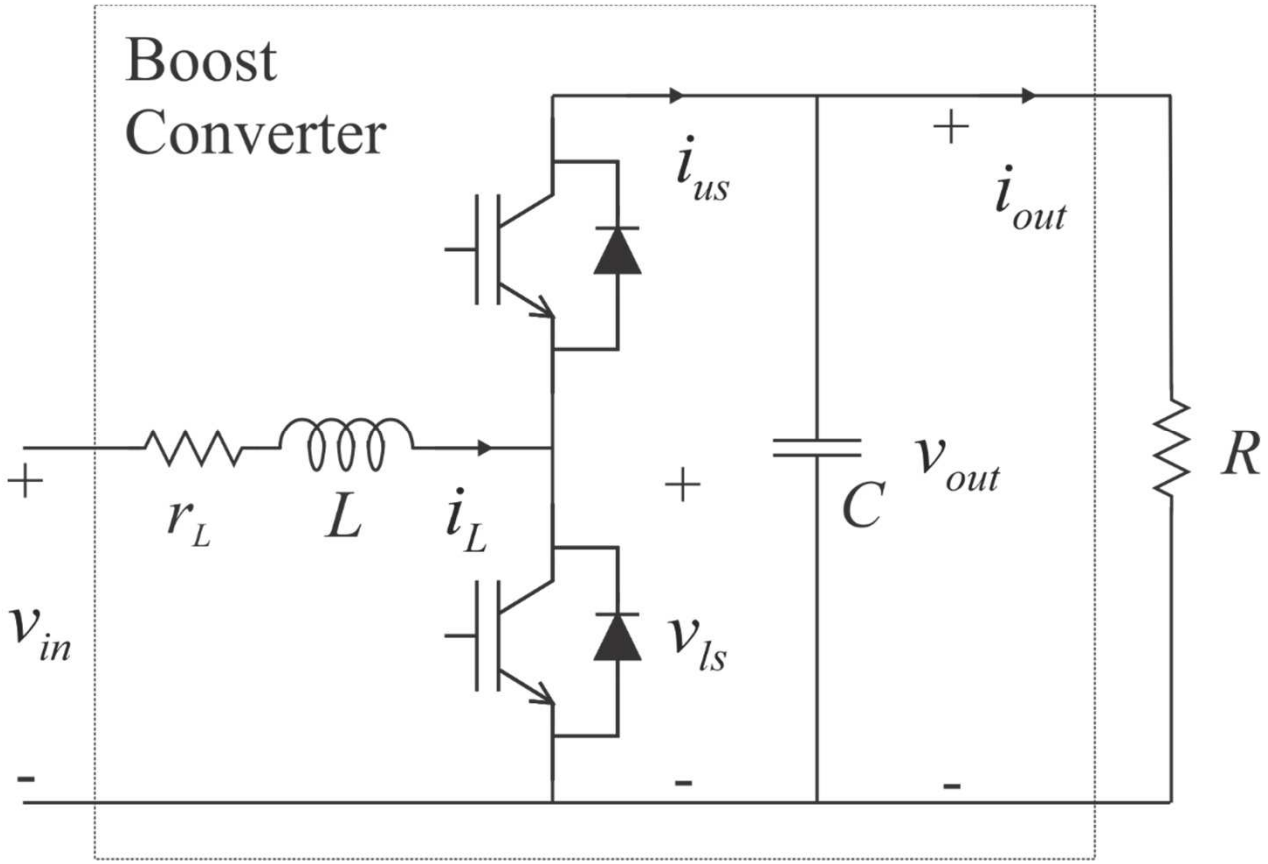
odefsrk.m

```
        y3=y1+ho2*k2;
        k3=fhandle(t2,y3,par);

        y4=y1+h*k3;
        k4=fhandle(t3,y4,par);

        k=(k1+2.0*k2+2.0*k3+k4)/6.0;
        y1=y1+k*h;

        t1=t1+h;

    end

    % fill in the y-matrix
    y(i,:)=y1';

end
```

An Example: A Boost Converter

Boost Converter Topology

- Circuit Diagram



Steady-State Operation with Resistive Load

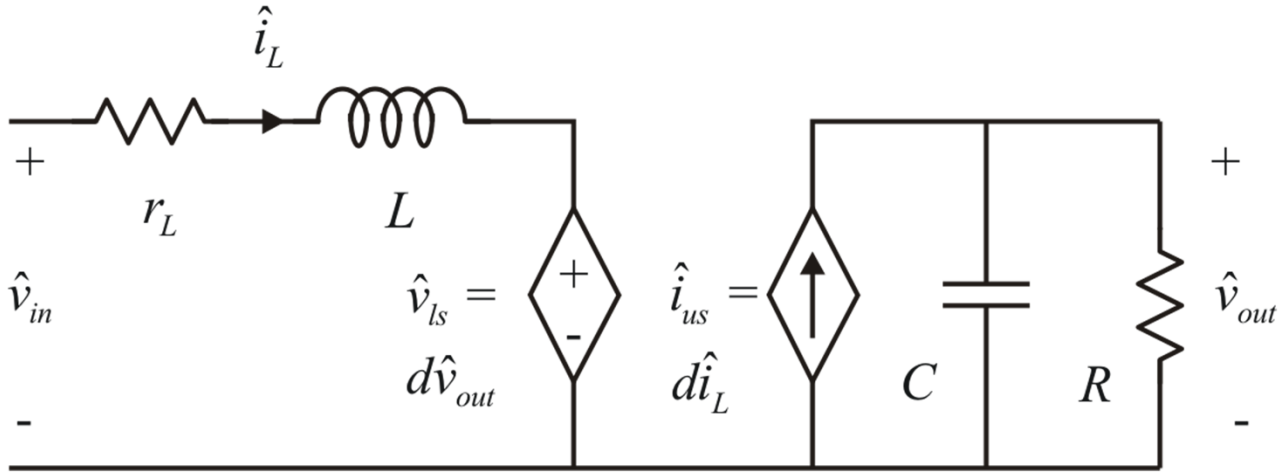
- Output voltage

$$\hat{v}_{out} = \frac{\hat{v}_{in}}{d + \frac{r_L}{Rd}}$$

- Inductor current

$$\hat{i}_L = \frac{\hat{v}_{out}}{dR}$$

Average-Value Model



$$\hat{v}_{ls} = d\hat{v}_{out}$$

$$\hat{i}_{us} = d\hat{i}_L$$

$$\hat{i}_{out} = \frac{\hat{v}_0}{R}$$

$$p\hat{i}_L = \frac{\hat{v}_{in} - \hat{v}_{ls} - r_L\hat{i}_L}{L}$$

$$p\hat{v}_{out} = \frac{\hat{i}_{us} - \hat{i}_{out}}{C}$$

Boost Converter Study

- System Parameters
  - R = 10 Ω
  - L = 5 mH
  - r_L = 0.5 Ω
  - C = 1000 uF
  - v_{in} = 300 V
- The converter is in the steady-state. Then the duty cycle steps from 0.5 to 0.7 at t = 20 ms.

Boost Converter Simulation in Matlab

Matlab Implementation

- msim_dcdc.m
- odefsrk.m
- mavm_dcdc.m

msim_dcdc.m

```
% This script sets parameter and simulates a
% simple matlab average value dc/dc (boost) converter using mavm_dcdc
%
% Written by S.D. Sudhoff
%   School of Electrical and Computer Engineering
%   1285 Electrical Engineering Building
%   West Lafayette, IN 47907-1285
%   Phone: 765-494-3246
%   Fax: 765-494-0676
%   E-mail: sudhoff@ecn.purdue.edu

% model parameters
P.L=5e-3;                    % inductor inductance (H)
P.rL=0.5;                    % inductor resistance (Ohms)
P.C=1000e-6;                 % capacitor capacitance (F)
P.Rload=10.0;                % load resistance (Ohms)
P.di=0.5;                    % initial duty cycle
P.df=0.7;                    % final duty cycle
P.tstep=0.02;                % time for duty cycle step
P.vin=300;                   % input voltage(V)
```

msim_dcdc.m

```
% compute intial conditions
vouti=P.vin/(P.rL/(P.Rload*P.di)+P.di);     % initial output
voltage (V)
ili=vouti/(P.di*P.Rload);                   % initial inductor
current (A)
xic=[ili vouti];

% time values where we want the answer
tvals=linspace(0,0.1,200);

% maximum time step
maxt=2e-5;

% perform the simulation
[t,x]=odefsrk(@mavm_dcdc,tvals,xic,P,maxt);

% grab the waveforms
iin=x(:,1);
vout=x(:,2);
```

msim_dcdc.m

```
% plot the voltage
figure(1)
plot(tvals,vout)
xlabel('t, s');
ylabel('v_{out}, A');
title('Output Voltage Versus Time');
axis([0 0.1 0 550]);
grid on

% plot the current
figure(2)
plot(tvals,iin)
xlabel('t, s');
ylabel('i_{in}, A');
title('Input Current Versus Time');
axis([0 0.1 0 120]);
grid on
```

mavm_dcdc

```
function [varargout]=mavm_dcdc(t,x,P)
% This routine contains the dynamics of an simple dc/dc (boost) converter
% model.
%
% [px] = mavm_dcdc(t,x,P);
% [px,vls,ius,iout] = mavm_dcdc(t,x,P);
%
% Inputs:
% t        = time (s)
% x        = state vector
%  x(1)    = fast average of inductor current (A)
%  x(2)    = fast average of output voltage (V)
% P        = structure with parameters
%  P.L     = inductor inductance (H)
%  P.rL    = inductor series resistance (Ohms)
%  P.C     = capacitor capacitance (F)
%  P.Rload = load resistance (Ohms)
%  P.vin   = input voltage (V)
%  P.di    = initial duty cycle (upper switch on / total period)
%  P.df    = final duty cycle
%  P.tstop = time of step in duty cycle (s)
%
```

mavm_dcdc

```
% Outputs:
% px        = time derivative of state vector
%  px(1)    = time derivative of fast average of inductor current (A)
%  px(2)    = time derivative of fast average of output voltage (V)
% vls       = fast average lower switch voltage (V)
% ius       = fast average current out ot upper switch (A)
% iout      = output current (A)
%
% Internal:
% d         = duty cycle
% pil       = time derivative of il (A/s)
% pvout     = time derivative of vout (V/s)
% il        = fast average inductor current (A)
% vout      = fast average output voltage (V)
%
% Written by S.D. Sudhoff
%   School of Electrical and Computer Engineering
%   1285 Electrical Engineering Building
%   West Lafayette, IN 47907-1285
%   Phone: 765-494-3246
%   Fax: 765-494-0676
%   E-mail: sudhoff@ecn.purdue.edu
```

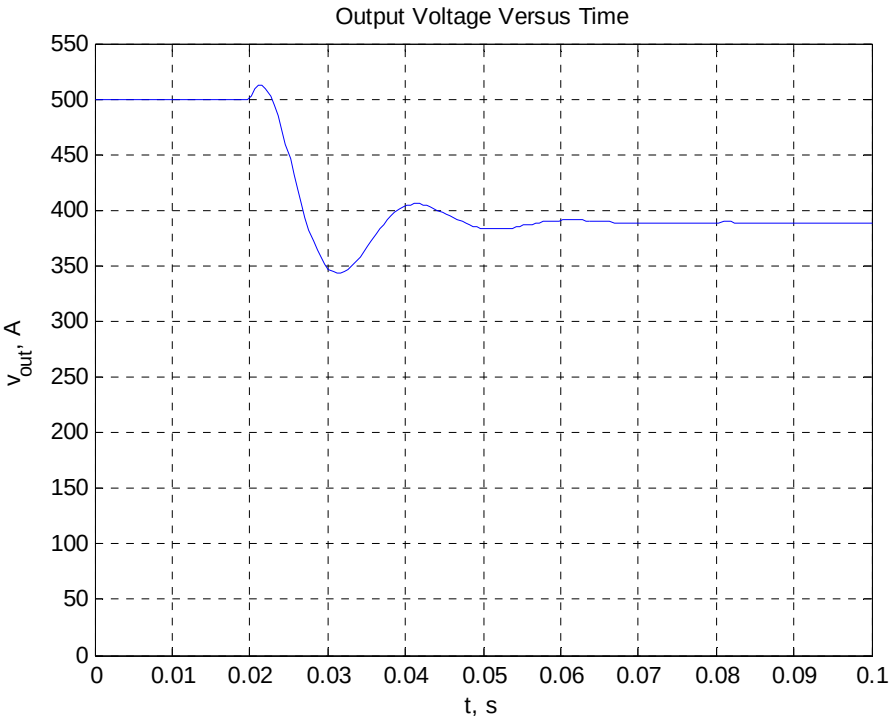
mavm_dcdc

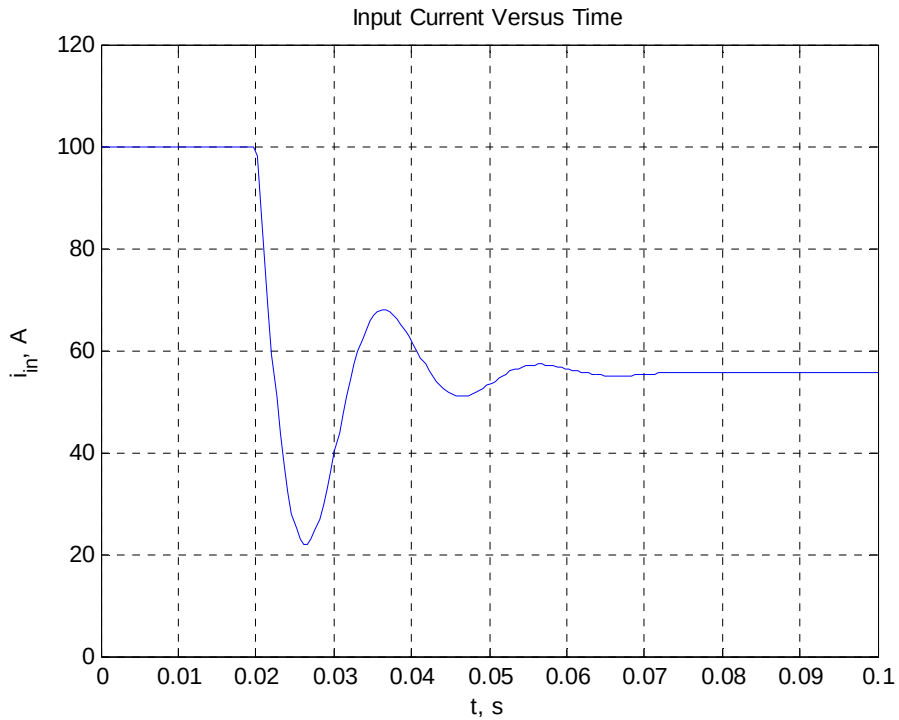
```
% decompose state vector
il=x(1);
vout=x(2);

% compute the duty cycle
if (t<P.tstep)
   d=P.di;
else
   d=P.df;
end

% variables of interest
vls=d*vout;
ius=d*il;
iout=vout/P.Rload;
```

mavm_dcdc

```
% compute derivatives of state
pil=(P.vin-P.rL*il-vls)/P.L;
pvout=(ius-iout)/P.C;

% pack the state vector
px=[pil pvout]';

% output variables
if nargout==1
   varargout={px};
else
   varargout={px,vls,ius,iout};
end

end
```

Results


Output Voltage Versus Time
550
500
450
400
350
300
250
200
150
100
50
0
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
t, s
v_out, A


Input Current Versus Time
120
100
80
60
40
20
0
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
t, s
i_in, A

Simulation Engines Metrics

One-Step Methods

- In general:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\Phi(t, \mathbf{x}_n; h)$$

- For Euler's Method

(Local) Truncation Error

- Let **z**(t) be the exact solution to our ODE

$$\frac{d\mathbf{z}}{d\tau} = p\mathbf{z} = \mathrm{f}(\tau, \mathbf{z}) \quad t \le \tau \le t + h$$

$$\mathbf{z}(t) = \mathbf{x}_n$$

- Our local truncation error is defined as

$$\mathbf{T}(t, \mathbf{x}; h) = \frac{1}{h}\left[\mathbf{x}_{n+1} - \mathbf{z}(t+h)\right]$$

Order

- Method Φ is said to have order p if

$$\left\| \mathbf{T}(t, \mathbf{x}; h) \right\| \leq C h^p$$

where C independent of t, $\mathbf{x}$, and h

A-Stability

- Consider a linear system

$$\frac{d\mathbf{x}}{dt} = \mathbf{Ax} + \mathbf{Bu}$$

$$\text{where } \mathrm{Re}\,\lambda_i(\mathbf{A}) \le 0 \qquad \forall i \subset [1,2,\ldots O]$$

A-Stability

- We can show $\mathbf{x}_{n+1}$ is related to $\mathbf{x}_n$ by the stability function

$$\mathbf{x}_{n+1} = \varphi(h\mathbf{A})\mathbf{x}_n$$

- To represent the system behavior we require

$$\left|\varphi(h\,\lambda_i(\mathbf{A}))\right| \le 1 \quad \forall i \subset [1,2,\ldots O]$$

- For A-Stable Methods

$$\left|\varphi(z)\right| \le 1 \quad \forall \operatorname{Re}(z) < 0$$

Proof – Part 1

- Consider special case of system model with non-repeated eigenvalues

Proof – Part 2

- Properties of Exact Solution

Proof – Part 4

- Similarity Transformation

Proof – Part 4

- Similarity Transformation

Proof – Part 3

- The Stability Function

Proof – Part 5

- Applying Similarity Transformation to Stability Function

Proof – Part 5

- Applying Similarity Transformation to Stability Function

Proof – Part 6

- Conclusion

Global Error

- The global error is defined as

$$e_n = \mathbf{z}(t_n) - \mathbf{x}_n$$

where

$$\frac{d\mathbf{z}}{dt} = p\mathbf{z} = \mathrm{f}(t, \mathbf{z})$$

$$\mathbf{z}(t_0) = \mathbf{x}_0$$

Convergence

- A method is <u>convergent</u> if

$$\lim_{h \to 0} \max_{n} \left\| \mathbf{e}_n \right\| = 0$$

Euler's Method Revisited

Stability Function of Euler Method

Order of the Forward Euler Method

Order of the Forward Euler Method

A-Stability (or lack thereof) of Euler's Method

A-Stability (or lack thereof) of Euler's Method

Runga Kutta Method Revisited

Stability Function of Runga Kutta Method

Stability Function of Runga Kutta Method

Stability Function of Runga Kutta Method

Order of Runga Kutta Method

Order of Runga Kutta Method

Stability of Runga Kutta Method

Stiffness

Stiffness

Stiffness

- In general

$$\left|\mathrm{Re}(\bar{\lambda})\right| \geq \left|\mathrm{Re}(\lambda_i)\right| \geq \left|\mathrm{Re}(\underline{\lambda})\right|$$

$$\mathrm{SR} = \frac{\left|\mathrm{Re}(\bar{\lambda})\right|}{\left|\mathrm{Re}(\underline{\lambda})\right|}$$

Stiffness

Model-Order Reduction

Ideal Case

A Less Ideal Case

A Less Ideal Case

A Less Ideal Case

A-Stable Methods: Backwards Euler

Backward Euler

- Update rule

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\Phi(t_n, \mathbf{x}_n; h)$$

$$\Phi(t, \mathbf{x}_n; h) = \mathrm{f}(t_{n+1}, \mathbf{x}_{n+1})$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\,\mathrm{f}(t_{n+1}, \mathbf{x}_{n+1})$$

Stability Function of Backward Euler

A-Stability of Backward Euler

Backwards Euler for Linear Systems

Update for Backwards Euler for Linear Systems

Trapezoidal Predictor Corrector

Trapezoidal

- Update rule

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\Phi(t_n, \mathbf{x}_n; h)$$

$$\Phi(t, \mathbf{x}_n; h) = \frac{1}{2}\left(\mathrm{f}(t_n, \mathbf{x}_n) + \mathrm{f}(t_{n+1}, \mathbf{x}_{n+1})\right)$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}\left(\mathrm{f}(t_n, \mathbf{x}_n) + \mathrm{f}(t_{n+1}, \mathbf{x}_{n+1})\right)$$

Trapezoidal Predictor-Corrector Method

- Again consider

$$\frac{d\mathbf{x}}{dt} = \mathrm{f}(\mathbf{x}, t)$$

- Algorithm
  - Step 1- Initialize

$$\mathbf{x}_{old} = \mathbf{x}_n + h\,\mathrm{f}(\mathbf{x}_n, t_n)$$

$$t_{n+1} = t_n + h$$

$$g = 1$$

Trapezoidal Predictor-Corrector Method

Note: this slide has been modified from lecture

- Algorithm (Continued)
  - Step 2 – Iterate

$$\mathbf{x}_{new} = \mathbf{x}_n + \frac{h}{2}\left(\mathrm{f}(\mathbf{x}_n, t_n) + \mathrm{f}(\mathbf{x}_{old}, t_n)\right)$$

$$\mathbf{x}_d = \left|\mathbf{x}_{old} - \mathbf{x}_{new}\right|$$

$$\mathbf{x}_s = \left|\mathbf{x}_{old} + \mathbf{x}_{new}\right|$$

$$\mathbf{a} = \left|\mathbf{x}_d\right| < \varepsilon_{aec}$$

$$\mathbf{r} = \mathbf{x}_d - \varepsilon_{rec}\mathbf{x}_s < \mathbf{0}$$

$$\mathbf{x}_{old} = \mathbf{x}_{new}$$

$$g = g + 1$$

until g=G or $\left(\mathbf{a}^i \; or \; \mathbf{r}^i\right) = 1 \; \forall i$

Trapezoidal Predictor-Corrector Method

- Algorithm
  - Step 3 – Finish $\mathbf{x}_{n+1} = \mathbf{x}_{new}$

Trapezoidal Predictor-Corrector Method

- Comments on the relative error criteria

A-Stability of Trapezoidal Method

Stability Function for Trapezoidal

A-Stability of Trapezoidal

Order of Trapezoidal

Order of Trapezoidal

Update for Backwards Euler for Linear Systems

Boost Converter Simulation in Simulink

Simulink Implementation

- savm_dcdc_setup.m
- savm_dcdc.slx
- savm_dcdc_plot.m
- savm_dcdc_pp.m
- savm_dcdc_study.m

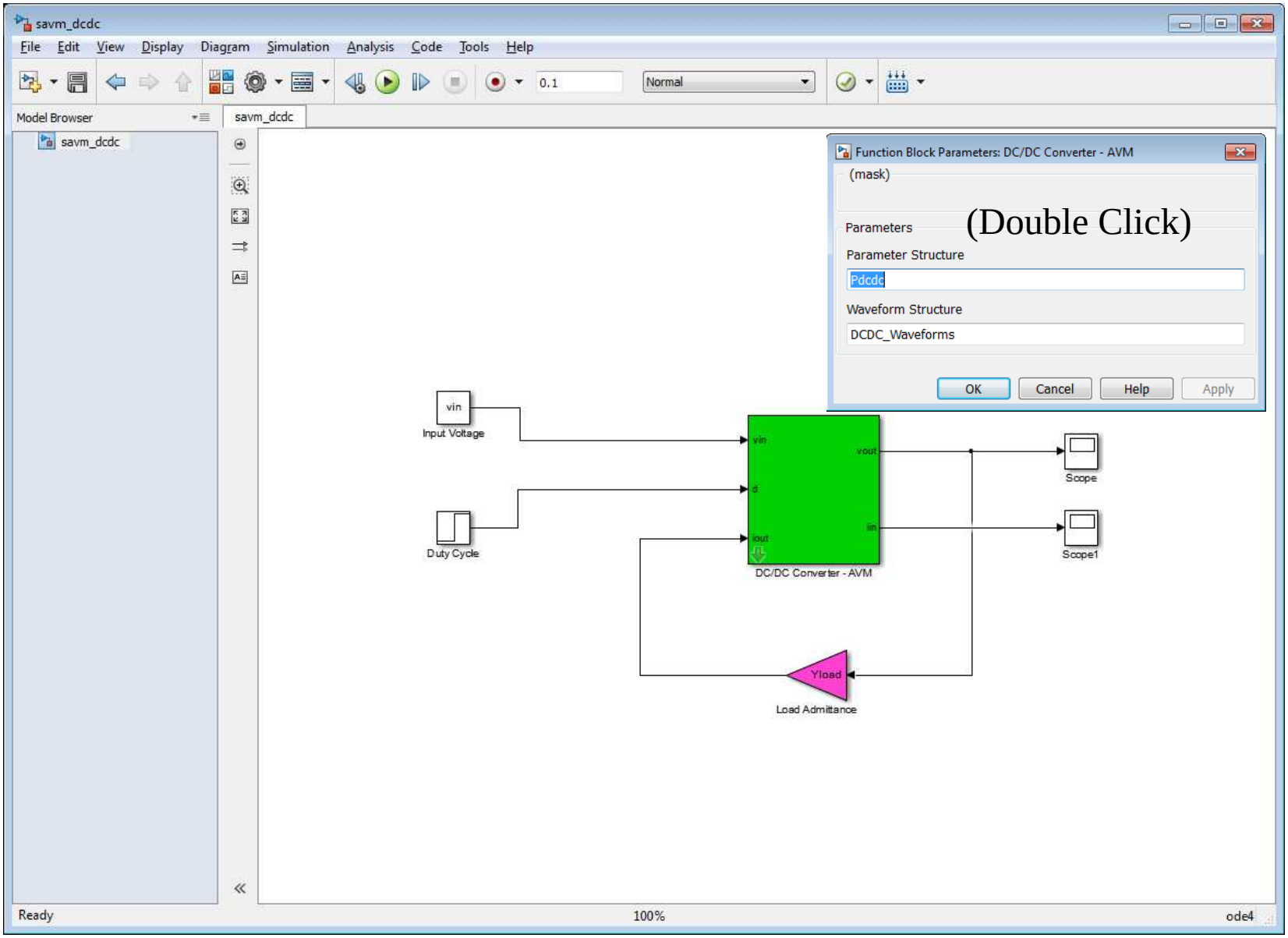
savm_dcdc_setup

```
% This script set parameter and performs intial calculations for
% simple simulink average value dc/dc converter simulation savm_dcdc

% Written by S.D. Sudhoff
%   School of Electrical and Computer Engineering
%   1285 Electrical Engineering Building
%   West Lafayette, IN 47907-1285
%   Phone: 765-494-3246
%   Fax: 765-494-0676
%   E-mail: sudhoff@ecn.purdue.edu

% input source
vin=300;                                    % input voltage (V)
```

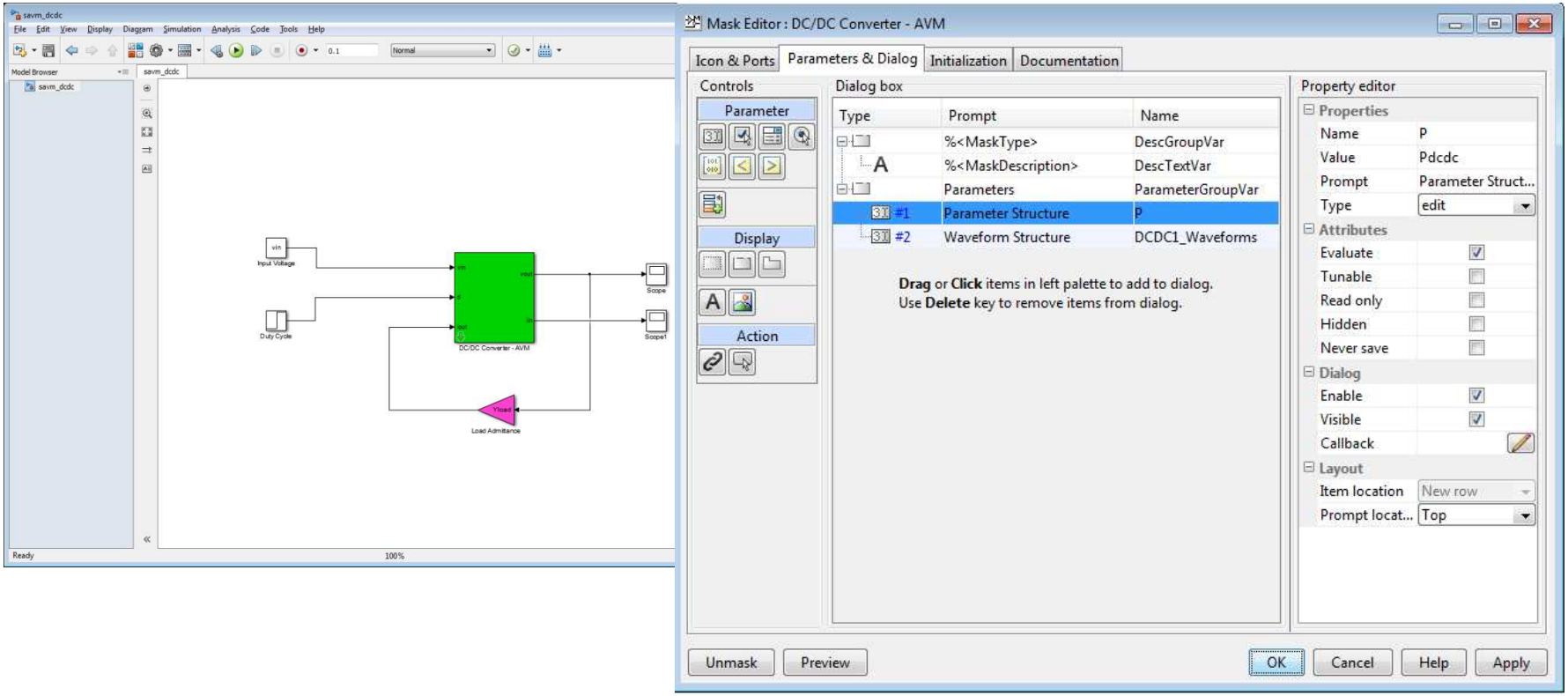
savm_dcdc_setup

```
% duty cycle
di=0.5;                                          % initial value
df=0.7;                                          % final value
tstep=0.02;                                      % time of step (s)

% load
Rload=10.0;                                      % load resistance (Ohms)
Yload=1/Rload;                                   % load conductance (Mhos)

% model parameters
Pdcdc.L=5e-3;                                    % inductor inductance (H)
Pdcdc.rL=0.5;                                    % inductor resistance (Ohms)
Pdcdc.C=1000e-6;                                 % capacitor capacitance (F)

% initial condition
vouti=vin/(Pdcdc.rL/(Rload*di)+di);              % initial output voltage (V)
ili=vouti/(di*Rload);                            % initial inductor current (A)
Pdcdc.xic=[ili vouti];                           % initial state vector
```

savm_dcdc

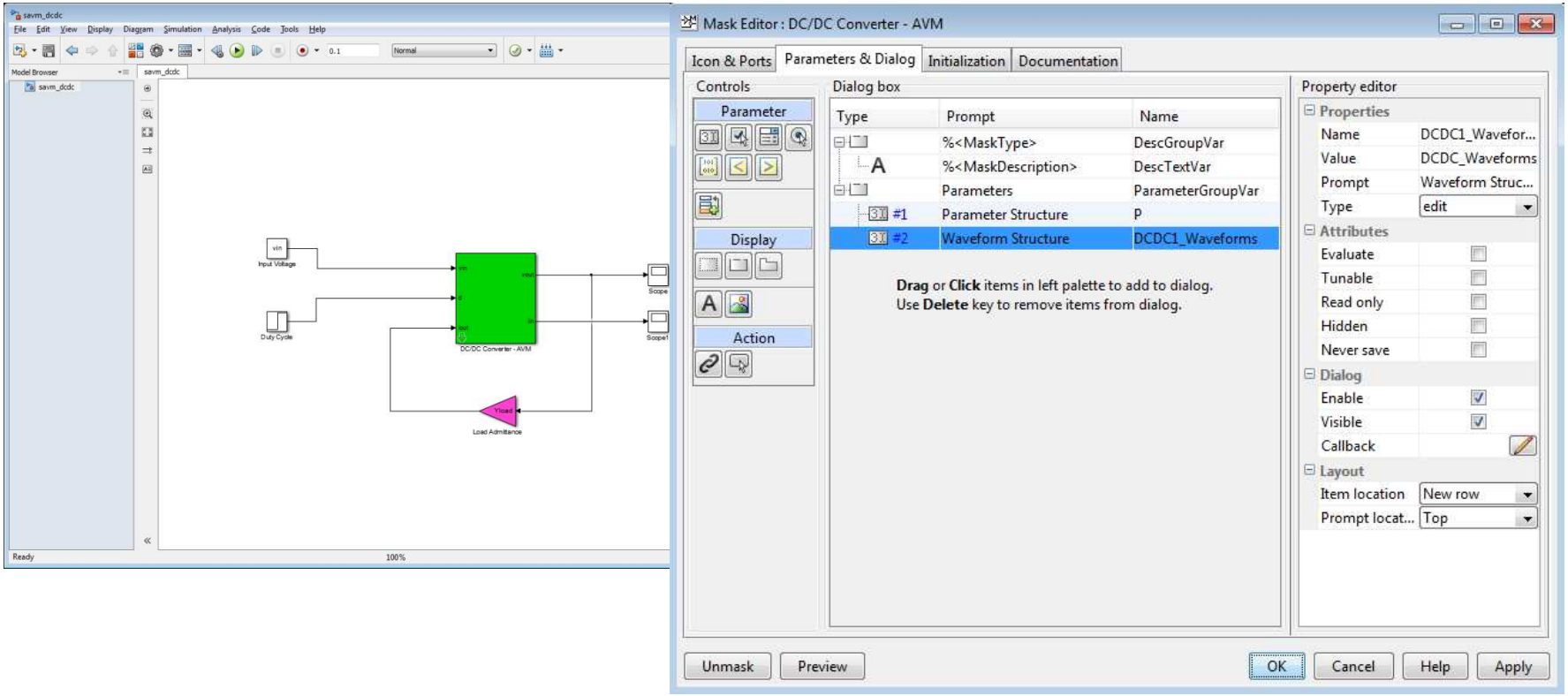

savm_dcdc
File Edit View Display Diagram Simulation Analysis Code Tools Help
0.1
Normal
Model Browser
savm_dcdc
savm_dcdc
Function Block Parameters: DC/DC Converter - AVM
(mask)
Parameters
(Double Click)
Parameter Structure
Pdcdc
Waveform Structure
DCDC_Waveforms
OK
Cancel
Help
Apply
vin
Input Voltage
Duty Cycle
vin
d
iout
vout
iin
DC/DC Converter - AVM
Scope
Scope1
Yload
Load Admittance
Ready
100%
ode4

savm_dcdc



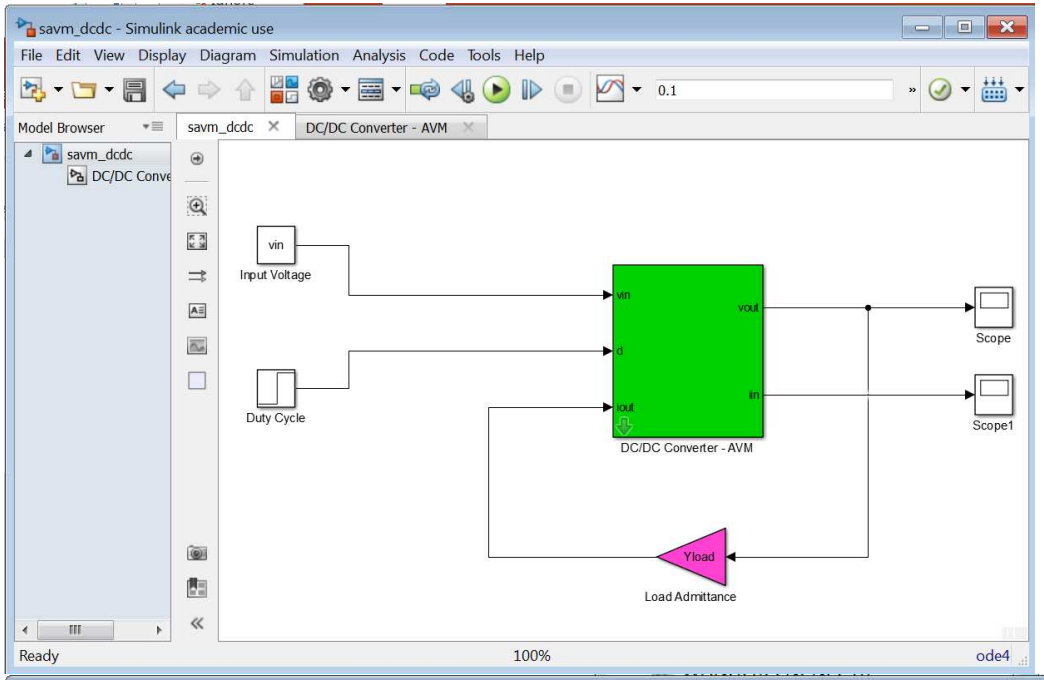
Right Click on DC/DC Converter – AVM – Mask – Edit Mask

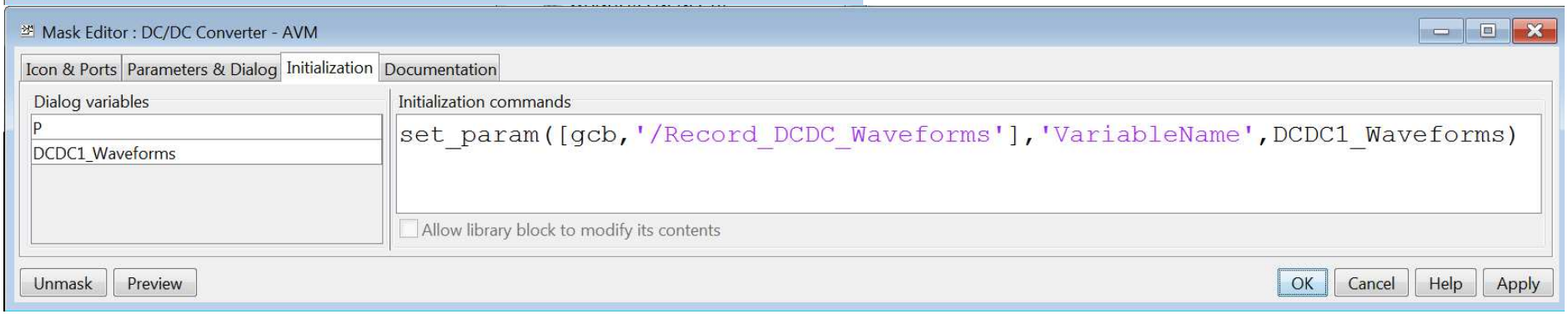
savm_dcdc



Right Click on DC/DC Converter – AVM – Mask – Edit Mask

savm_dcdc



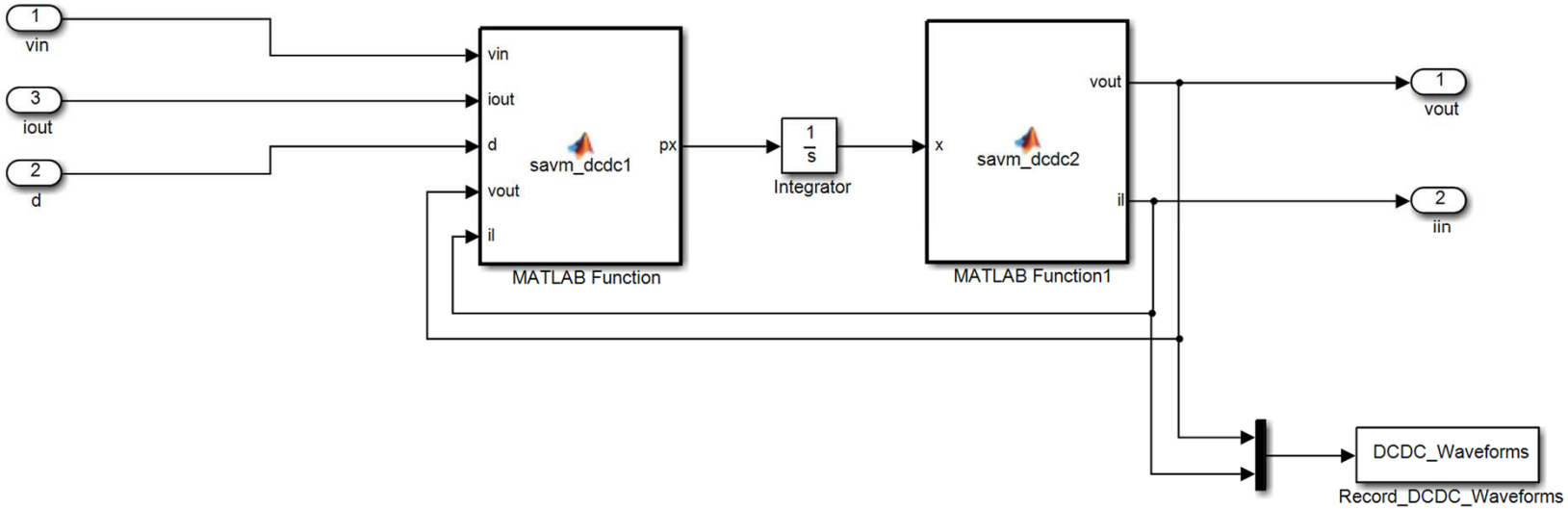


Right Click on DC/DC Converter – AVM – Mask – Edit Mask

Creating A Subsystem

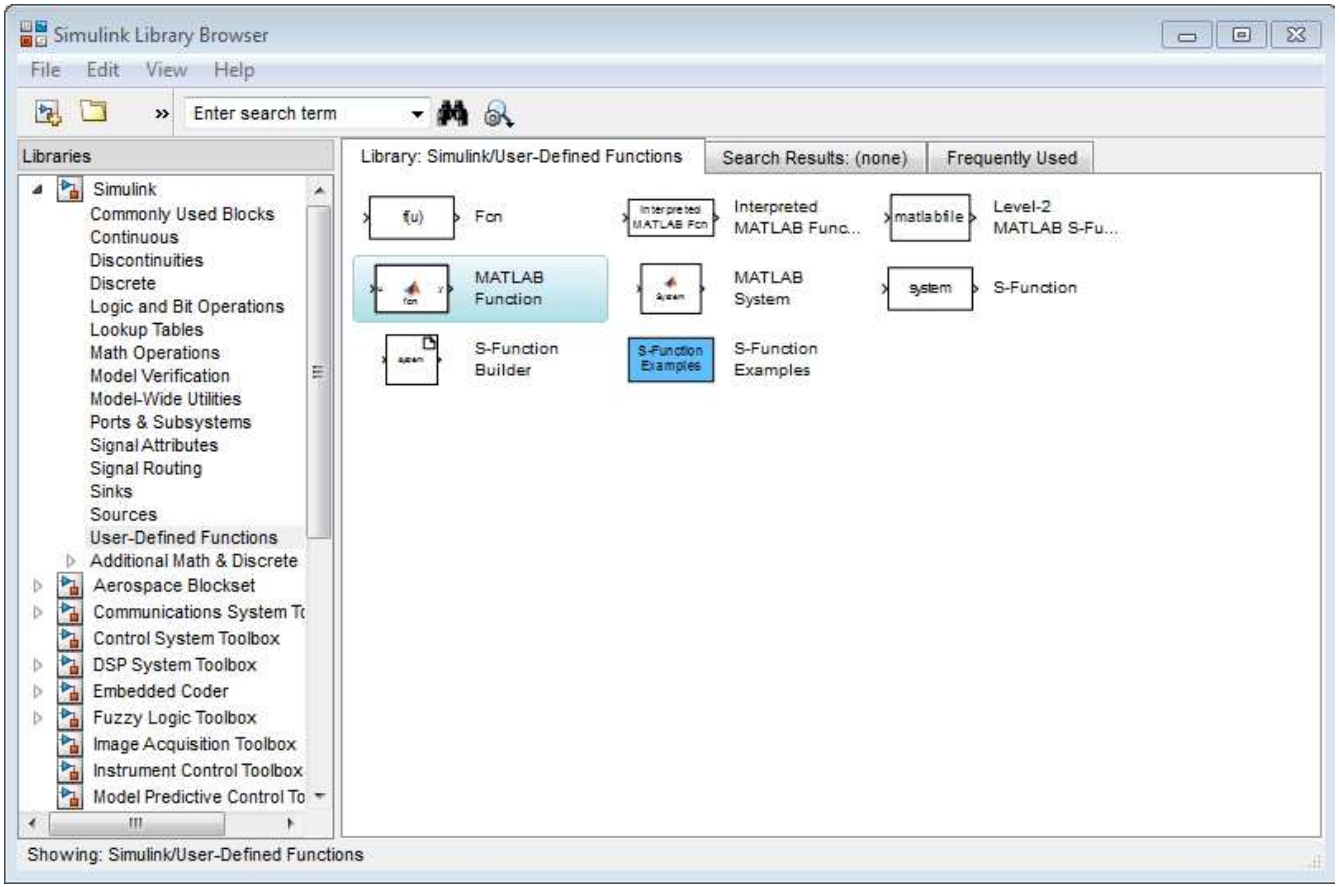
- Select Portion of Diagram of Interest
- Select *Diagram – Model & Subsystem Reference – Create Subsystem from Selection*

savm_dcdc


1
vin
3
iout
2
d
vin
iout
d
vout
il
savm_dcdc1
px
MATLAB Function
1
s
Integrator
x
savm_dcdc2
vout
il
MATLAB Function1
1
vout
2
iin
DCDC_Waveforms
Record_DCDC_Waveforms

Creating A Block

- Select *View* then *Library Browser*



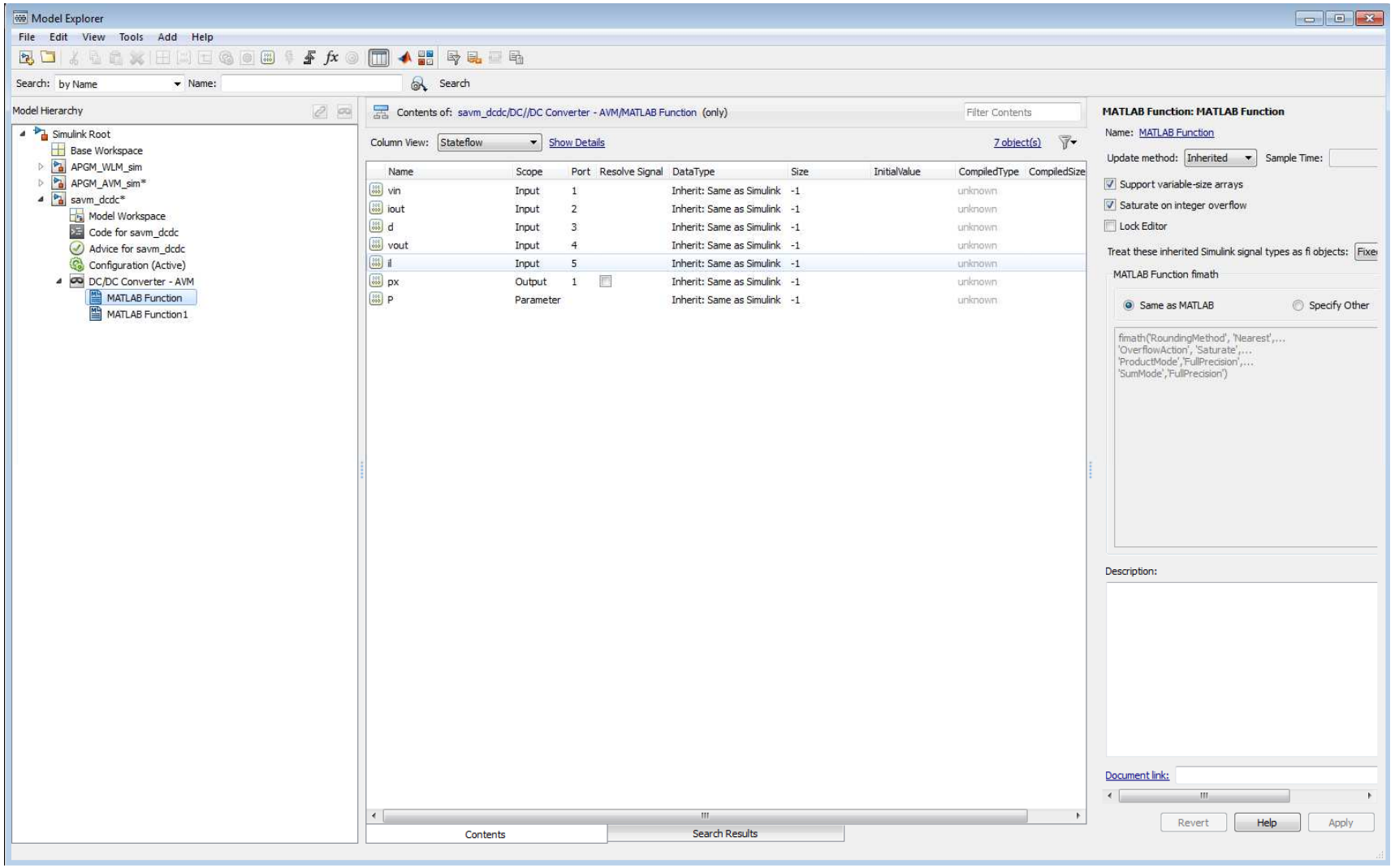
savm_dcdc1

```
function px = savm_dcdc1(vin,iout,d,vout,il,P)
% This routine contains the dynamics of an simple dc/dc (boost) converter
% model. To be used with savm_dcdc2.
%
% Inputs:
% vin       = input voltage (V)
% iout      = output curent (A)
% d         = duty cycle
% P         = structure with parameters
%  P.L      = inductor inductance (H)
%  P.rL     = inductor series resistance (Ohms)
%  P.C      = capacitor capacitance (F)
%
% Outputs:
% px        = time derivative of state vector
%  px(1)    = time derivative of fast average of inductor current (A)
%  px(2)    = time derivative of fast average of output voltage (V)
%
```

savm_dcdc1

```
% Internal:
% vls      = average voltage across lower switch/diode (V)
% ius      = average current into upper switch/diode (A)
% pil      = time derivative of il (A/s)
% pvout    = time derivative of vout (V/s)
%
% Written by S.D. Sudhoff
%   School of Electrical and Computer Engineering
%   1285 Electrical Engineering Building
%   West Lafayette, IN 47907-1285
%   Phone: 765-494-3246
%   Fax: 765-494-0676
%   E-mail: sudhoff@ecn.purdue.edu

% variables of interest
vls=d*vout;
ius=d*il;

% compute derivatives of state
pil=(vin-P.rL*il-vls)/P.L;
pvout=(ius-iout)/P.C;

% pack the state vector
px=[pil pvout]';
```

Matlab Function I/O

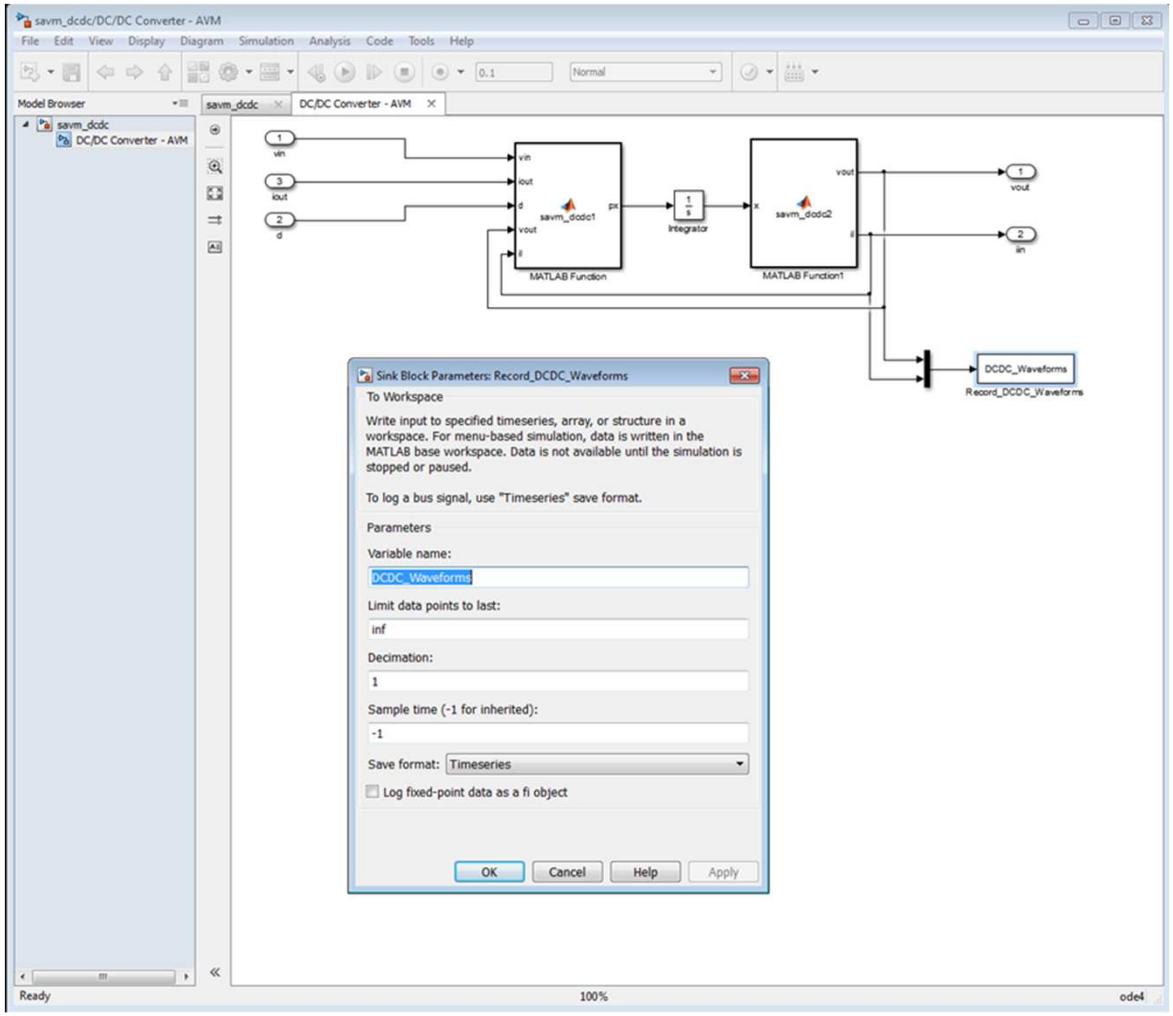


Right Click on MATLAB Function – Explore – Click on Scope to Change

savm_dcdc2

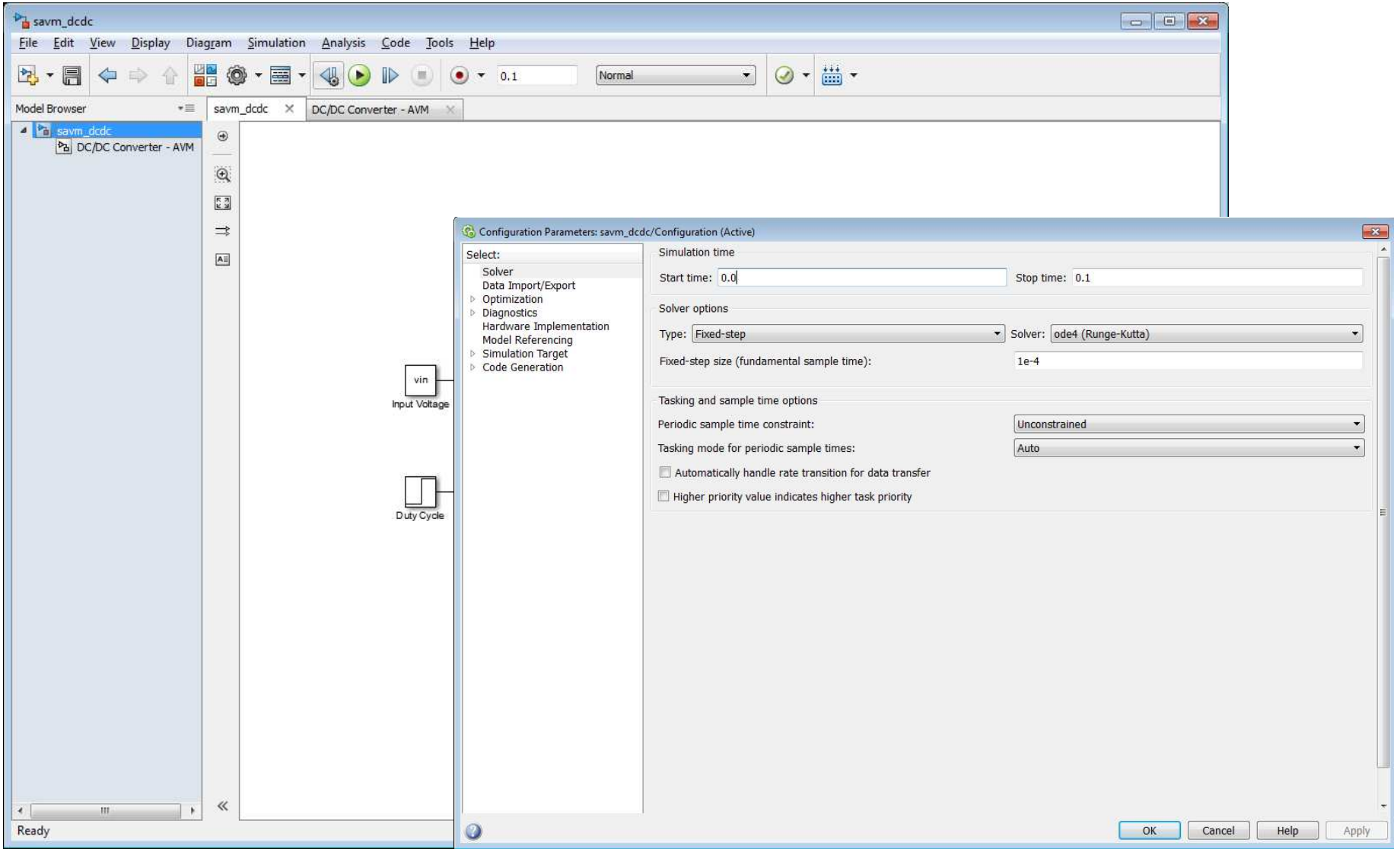
```
function [vout,il] = savm_dcdc2(x)
% This routine decomposes the state vector in a simple average value
% model of a dc/dc (boost) converter. To be used with savm_dcdc1.
%
% Inputs:
% x         = state vector
%  x(1)     = fast average of inductor current (A)
%  x(2)     = fast average of output voltage (V)
%
% Outputs:
% vout      = output voltage (V)
% il        = inductor (input) current (A)
%
% Written by S.D. Sudhoff
%   School of Electrical and Computer Engineering
%   1285 Electrical Engineering Building
%   West Lafayette, IN 47907-1285
%   Phone: 765-494-3246
%   Fax: 765-494-0676
%   E-mail: sudhoff@ecn.purdue.edu

% break apart state vector
il=x(1);
vout=x(2);
```

Output to Workspace



View – Library Browser – Simulink – Sinks – To Workspace

Simulation Settings



Simulation – Model Configuration Parameters

savm_dcdc_plot

```
function savm_dcdc_plot(Name,Wf,fn)
% savm_dcdc_plot plots waveforms associated with a dc/dc converter
%
% savm_dcdc_plot(Name,Wf)
% savm_dcdc_plot(Name,Wf,fn)
%
% Inputs:
% Name          = text Description of component
% Wf            = structure of waveform data
%  Wf.Time      = time vector (s)
%  WF.Data(:,1) = output voltage (V)
%  Wf.Data(:,2) = inductor current (A)
% fn            = figure number
%
% Internal:
% t             = time vector (s)
% il            = inductor current (A)
% vout          = output voltage (V)
%
% Written by:
% S.D. Sudhoff
% Purdue University
% School of Electrical and Computer Engineering
% 1285 Electrical Engineering Building
% West Lafayette, IN 47907-1285
% Phone: 765-494-3246
% Fax: 765-494-0676
% E-mail: sudhoff@ecn.purdue.edu
```

savm_dcdc_plot

```
% Get data
t=Wf.Time;
vout=Wf.Data(:,1);
il   =Wf.Data(:,2);

% plot dc output current
if nargin>2
   figure(fn)
else
   figure
end
plot(t,vout);
xlabel('t, s');
ylabel('v_{out}, V');
grid on;
title([Name, ' DC Output Voltage']);

% plot torque
if nargin>2
   figure(fn+1)
else
   figure
end
plot(t,il);
xlabel('t, s');
ylabel('i_l, A');
grid on;
title([Name, ' Inductor Current']);
```

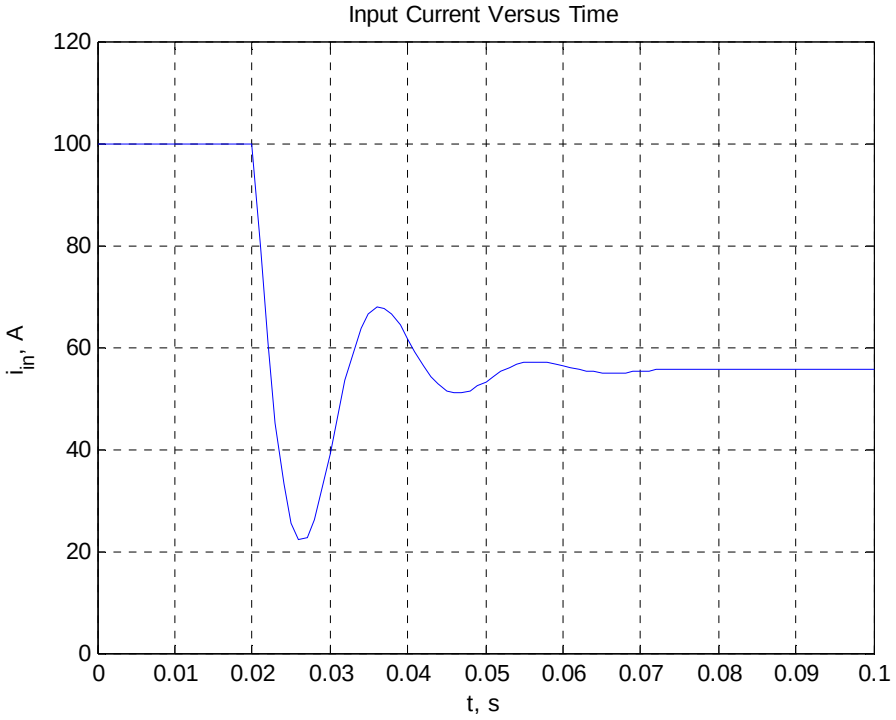
savm_dcdc_pp

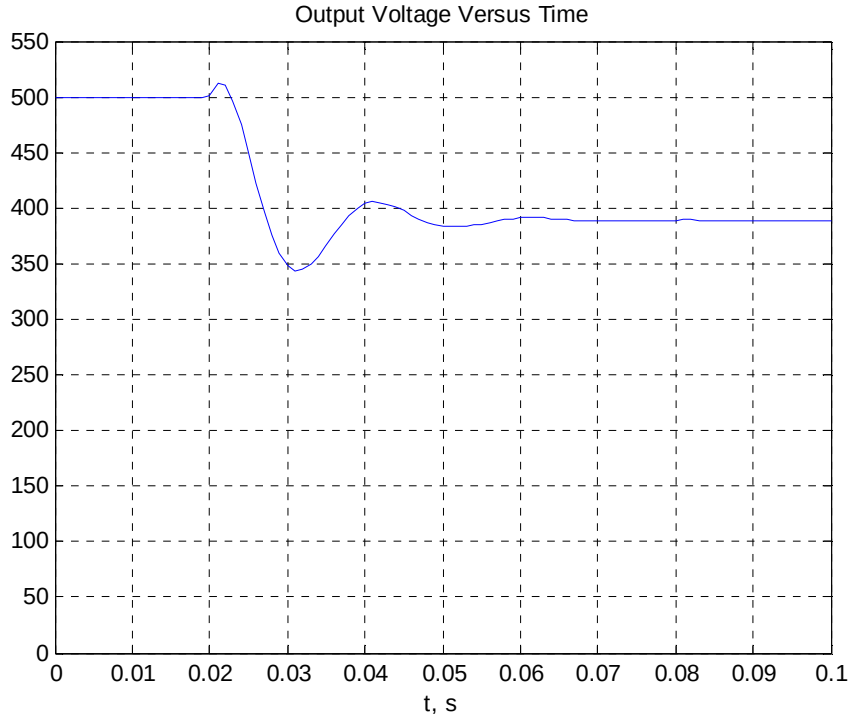
```
% This script imports and plots waveforms from the
% simple simulink average value dc/dc converter simulation savm_dcdc

% Written by S.D. Sudhoff
%   School of Electrical and Computer Engineering
%   1285 Electrical Engineering Building
%   West Lafayette, IN 47907-1285
%   Phone: 765-494-3246
%   Fax: 765-494-0676
%   E-mail: sudhoff@ecn.purdue.edu

savm_dcdc_plot('DCC Converter',DCDC_Waveforms,10);
```

savm_dcdc_study

```
% This script sets parameter and performs intial calculations for
% simple simulink average value dc/dc converter simulation, runs the
% study, and then plots the results

% Written by S.D. Sudhoff
%   School of Electrical and Computer Engineering
%   1285 Electrical Engineering Building
%   West Lafayette, IN 47907-1285
%   Phone: 765-494-3246
%   Fax: 765-494-0676
%   E-mail: sudhoff@ecn.purdue.edu

% initialize parameters
savm_dcdc_setup;

% run the study
sim('savm_dcdc');

% plot the results
savm_dcdc_plot('DCC Converter',DCDC_Waveforms,10);
```

Results


Input Current Versus Time
120
100
80
60
40
20
0
i_{in}, A
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
t, s


Output Voltage Versus Time
550
500
450
400
350
300
250
200
150
100
50
0
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
t, s

Comments On Modeling: Algebraic Loops

Sorting

- Recall: To predict a future value of state, we needed to find the present value of the time derivative of the state variables, based on what we know – that is the present value of state variables and present value of input variables
- Sorting is the evaluation order we need to follow when doing this

Sorting

- Consider the system

$$\frac{dx^1}{dt} = 3x^2 - x^1 + g$$

$$\frac{dx^2}{dt} = -3x^2 + z$$

$$g = t^2 + 2x^1 + z$$

$$z = \left(x^1\right)^2 x^2$$

Algebraic Loops

- Algebraic Loops occur when the system model is formatted such that it cannot be sorted
- Consider

$$\frac{dx^1}{dt} = -\left(x^1\right)^2 + x^2 + y$$

$$\frac{dx^2}{dt} = -5x^2 + z$$

$$y = 4z + x^1$$

$$z = 2y - x^2$$

Breaking Algebraic Loops

- Method 1 – Model Reformulation
- Start With

$$\frac{dx^1}{dt} = -\left(x^1\right)^2 + x^2 + y$$

$$\frac{dx^2}{dt} = -5x^2 + z$$

$$y = 4z + x^1$$

$$z = 2y - x^2$$

Breaking Algebraic Loops

Breaking Algebraic Loops

- Method 2 – Cheating
- Start With

$$\frac{dx^1}{dt} = -\left(x^1\right)^2 + x^2 + y$$

$$\frac{dx^2}{dt} = -5x^2 + z$$

$$y = 4z + x^1$$

$$z = 2y - x^2$$

Breaking Algebraic Loops

Breaking Algebraic Loops

- Method 3 – Numerical Solution

Simulink Artificial Algebraic Loops

Comments On Modeling: Discrete Events

Discrete Events and Scheduling

Closing Remarks

Simulation Research

- Hardware in the Loop (HIL)
- Power Hardware in the Loop (PHIL)
- Differential Algebraic Equations (DAE)
- Real Time Digital Simulators (RTDS)