Power System Toolbox
Version 2.0
Dynamic Tutorial and Functions

Cherry Tree Scientific Software
RR#5 Colborne
Ontario K0K 1S0
Canada

phone & fax: (905)349-2485
email: cherry@eagle.ca

Table of Contents

Dynamic Models: A Tutorial

Introduction

The purpose of the PST is to provide models of machines and control systems for performing transient stability simulations of a power system, and for building state variable models in small signal analysis and damping controller design. These dynamic models are coded as MATLAB functions.

Demonstration files are provided which enable a user to perform transient and small signal stability analysis. However, since the models are supplied as MATLAB m-files, by following a set of rules, the user can assemble customized models and applications.

In this tutorial we discuss the model conventions, structure and data requirements, and the method of interconnecting the models to form power system simulation models.

Necessary Data Requirements

The Power System Structure

This is defined by the **bus** and **line** specification matrices used in load flow calculations. A **solved** load flow case is required to set the operating condition used to initialize the dynamic device models. Load flow data which represents an unsolved case will lead to dynamic models which are not at equilibrium when initialized.

Generator Dynamic Data

This is supplied as the generator specification matrix **mac_con**. There are three types of generator model

1. the electromechanical (em) , or classical model (**mac_em**)
2. the transient model (**mac_tra**)
3. the subtransient model (**mac_sub**)

All use the same fields for data, but only the subtransient model uses every field. Thus all generator models are specified using a single specification matrix.

Simulation Control Data

For transient stability simulation, some method for instructing the simulation program to apply faults is required. The provided script file **y_switch** is an example of a simulation organization file. It uses the data specification file **sw_con** .

Optional Data

Depending on requirements additional data must be specified for the

- generator controls models
  - exciters - **exc_con**
  - power system stabilizers - **pss_con**
  - turbine-generators - **tg_con**
- induction motor models - **ind_con** and **mld_con**
- non-conforming loads - **load_con**
- static VAR compensator models - **svc_con**
- HVDC models
  - converters - **dcsp_con**
  - lines - **dcl_con**
  - controls - **dcc_con**

In small signal stability simulation generators may be specified as infinite buses using **ibus_con**.

Dynamic Model Functions

The models available in this version of PST include:

1. Generator models
   (a) **mac_em** -- electromechanical (classical) model
   (b) **mac_tra** -- model including transient effect
   (c) **mac_sub** -- model including subtransient effect [1]
   (d) **mac_ib** -- a generator as infinite bus model (used only in small signal stability simulation)
2. Excitation system models
   (a) **smpexc** -- simplified exciter model
   (b) **exc_dc12** -- IEEE type DC1 and DC2 models [2]
   (c) **exc_st3** -- IEEE type ST3 model [2]
3. Power system stabilizer model ... **pss**
4. Simplified turbine-governor model ... **tg**
5. Induction Motor Model...**mac_ind**
6. Induction Generator Model ... **mac_igen**
7. Static VAR compensator model -- **svc** [3]
8. Load Modulation Control ... **lmod**
9. HVDC line model – **dc_line, dc_cont**
10. Non-conforming load model -- **nc_load**
11. Line flow function -- **line_pq**
12. Utility functions -- **pss_des** (power system stabilizer design), **statef** (frequency response from state space), **step_res** (step response from state space system models).

Standard Dynamic Drivers

Driving functions are provided for transient stability (**s_simu**)and small signal stability (**svm_mgen**). These functions provide an environment which requires only the system data to be specified and act much like stand-alone transient and small signal stability programs. Details are given in the function descriptions section which follows this tutorial.

Expanding the Capabilities of PST

Since the source code for all functions is provided, a user may expand PST to meet special modelling or simulation requirements. The following indicates the preferred form of dynamic models.

Model Structure

Each model function consists of 3 parts

1. initialization of the state variables - **flag = 0**
2. network interface computation - **flag = 1**
3. calculation of the rates of change of state variables - **flag = 2**

In general, there are 4 input variables to a function, namely, **i** (the device number), **k** (time step), **bus** and **flag**. A convention used in all the supplied models is that if **i** is zero, the model calculations are made using vector methods. Additional variables are normally required for dynamic models. In PST these variables are normally specified as global. For consistency new global variables should be added to **pst_var**.

Most models require an interface mode, but some, such as the induction motor do not. If the mode does not exist, it is good practice to have a null section defined, see **mac_ind.m** for an example. In the case of the non-conforming load model, there are no state variables and hence no action is taken when this function is called with **flag = 2**.

New models should be coded so that they exit without error if the corresponding index or data specification matrix does not exist. In this way a single driver program, which calls all possible models, will not fail when the driver is run for a data set which does not contain the new model.

Vector Computation

In MATLAB, it is important to use vector computation whenever possible, and avoid loops in the computation process. In this version of PST, index functions are used to store data about the different types of similar models, e.g., generators. For example, if a new exciter model is added, the index function **exc_indx.m** must be modified to include the appropriate indexes which are passed on to the new exciter model as global variables.

Use of Templates

New dynamic models are most easily formed by modifying the existing models. This is the most efficient method. The data input format for the model should follow the same conventions as that of the existing models. The state variables should have meanings in new models similar to those in existing models. If there is no confusion, states already defined in **pst_var** should be used.

Transient Stability Simulation

A power system transient stability simulation model consists of a set of differential equations determined by the dynamic models and a set of algebraic equations determined by the power system network.

In PST, the dynamic generator models, with **flag = 1**, calculate the generator internal node voltages, i.e., the voltage behind transient impedance for the electromechanical generator, transient generator, and the voltage behind subtransient impedance for the subtransient generator. In the induction motor model the internal voltages behind transient impedance are the states vdprime and vqprime. These internal voltages are used with a system admittance matrix reduced to the internal nodes and the non-conforming load bus nodes to compute the current injections into the generators and motors. When there is an HVDC link in the model, the reduced Y matrix has additional rows and columns associated with the equivalent HT terminals of the HVDC links. The current injections are then used in the generator and motor models, and the non-conforming load voltages are used in the SVC and HVDC link models with **flag = 2**, to calculate the rates of change of their state variables.

All models should detect for the existence if valid model data, e.g., if the required data is not supplied, the model function exists with no changes. In this way, the driver can contain all existing models and relay on the data set to define those necessary for the required simulation.

Rather than build a new simulation driver from scratch for every additional simulation model, it is recommended that new models be added to the general transient stability driver **s_simu**. The structure is quite straightforward and well documented within the code.

Small Signal Stability

The stability of the operating point to small disturbances is termed small signal stability. To test for small signal stability we linearize the system dynamic equations about a steady state operating point to get a linear set of state equations

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

In some programs for small signal stability the state matrices are calculated analytically from the Jacobians of the non-linear state equations. In the Power System Toolbox, on the other hand, the linearization is performed by calculating the Jacobian numerically.

Starting from the states determined from model initialization, a small perturbation is applied to each state in turn. The change in the rates of change of all the states divided by the magnitude of the perturbation gives a column of the state matrix corresponding to the disturbed state. A permutation matrix **p_mat** is used to arrange the states in a logical order. Following each rate of change of state calculation, the perturbed state is returned to its equilibrium value and the intermediate variable values are reset to there initial values. Each step in this process is similar to a single step in a simulation program. The input matrix B, the output matrix C and the feed forward matrix D can be determined in a similar manner.

A single driver, **svm_mgen**, for small signal stability is provided. It is organized similarly to the transient stability simulation driver **s_simu**. New models should be designed to work satisfactorily in either driver. Generally, if a model is satisfactory in **s_simu**, it will be satisfactory in **svm_mgen**.

Damping Controller Design

The file **pss_des .m** provides a power system stabilizer design algorithm which uses the frequency response of a modified system in which the generator for which the stabilizer is to be designed has constant speed (very high inertia) and the other generators are either represented as infinite buses or netted as constant impedance loads. It requires the state matrices associated with the modified system as input. The required matrices are output from **svm_mgen**. The program displays the stabilizer ideal phase advance characteristic and allows the user to cut and try the PSS time constants until a satisfactory match over the frequency range of interest (0.2 to 1.5 Hz) has been found. The output of **pss_des** is the chosen set of time constants. The choice of PSS gain is made by running **svm_mgen** with the original system data and varying the gain from a low value until a satisfactory damping ratio is achieved. This process gives a stabilizer design which is quite robust to changes in system operating conditions.

Note: Any linear stabilizer design, should be checked for robustness using a transient stability simulation under a wide range of operating conditions. It is normal to set the PSS output limits so that the stabilizer has no adverse effects on a generator's response to a fault. Generally, the lower the negative output limit, the more effect the PSS has on the terminal voltage recovery following a fault.

References

1. R.P. Schulz, "Synchronous Machine Modeling," presented at the *Symposium "Adequacy and Philosophy of Modeling: System Dynamic Performance*," San Francisco, July 1972.
2. IEEE Committee Report, "Excitation System Models for Power System Stability Studies," *IEEE Transactions of Power Apparatus and Systems*, vol. PAS-100, pp. 494-509, 1981.
3. E.V. Larsen and J. H. Chow, "SVC Control Concepts for System Dynamic Performance," in *Application of Static VAR Systems for System Dynamic Performance*, IEEE Publications 87TH0187-5-PWR, 1987.
4. W.L. Brogan, *Modern Control Theory*, Quantum Publishers, New York, 1974.
5. J.H. Chow, editor, *Time-Scale Modeling of Dynamic Networks with Applications to Power Systems*, Springer-Verlag, Berlin, 1982.
6. V. Vittal, "Transient Stability Test Systems for Direct Stability Methods," IEEE Committee Report, IEEE Winter Power Meeting, Paper 91 WM 224-6 PWRS, 1991.
7. Graham Rogers and Joe Chow, " Hands-On Teaching of Power System Dynamics" *IEEE Computer Applications in Power*, January 1995, pp 12-16.

dbcage

Purpose:
Calculates the equivalent single cage resistance and reactance of a double cage induction motor as a function of slip.

Syntax:

[r,x]=dbcage(r1,x1,r2,x2,s)

Inputs:

r1	the first cage resistance (PU on motor base)
x1	the first cage leakage reactance (PU on motor base)
r2	the second cage resitance (PU on motor base)
x2	the inter-cage reactance (PU on motor base)
s	the motor slip

Outputs:

r	the equivalent rotor resistance at slip s (PU on motor base)
x	the equivalent rotor leakage reactance at slip s (PU on motor base)

Algorithm:

The rotor impedance is calculated at slip s and its real and imaginary parts used to define the the equivalent rotor resistance and reactance.

$$z = ix1 + (r1/s)(r2/s + ix2)/((r1 + r2)/s + ix2)$$

$$r = sreal(z); x = imag(z)$$

deepbar

Purpose:
Calculates the equivalent single cage resistance and reactance of a deep bar induction motor as a function of slip.

Syntax:

[r,x]=deepbar(rro,B,s)

Inputs:

rro	the resistance of the rotor bar at zero slip (PU on motor base)
B	the deep bar factor
s	the motor slip

Outputs:

r	the equivalent rotor resistance at slip s (PU on motor base)
x	the equivalent rotor leakage reactance at slip s (PU on motor base)

Algorithm:

The equivalent rotor resistance and reactance as a function of slip is

$$b = B\sqrt{|s|};$$

$$r_o = rro/2;$$

$$a = (1+i)b;$$

$$z = r_o a(\exp(a)+1)./(\exp(a)-1);$$

$$r = real(z); x = imag(z)./s;$$

Where B is the deep bar factor which depends on the depth of the rotor bar,

$$B = d\sqrt{2\omega\mu_o\sigma}$$

and,
ω is the angular frequency of the motor supply
μ_o is the permeability of free space
σ is the conductivity of the rotor bar

dc_cont

Purpose:
To model the action of HVDC link pole controllers in dynamic simulation

Syntax:

f = dc_cont(i,k,bus,flag)

Description:
dc_cont contains the equations required for the initialization, network interface and rate of change of state evaluation for the rectifier and inverter controls of HVDC links.

Inputs:

i — **i** = 0 all HVDC computations are performed using MATLAB vector methods

k — the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus — the solved bus specification matrix

flag — indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1. For proper initialization, the corresponding generators must be initialized first.
- The network interface calculation is performed when **flag** = 1, and the field voltage of the synchronous machine is set to the exciter output voltage.
- The rates of change of the exciter states are calculated when **flag** = 2, using the generator terminal voltage and the external system values at the time specified by **k**

Outputs:

f — a dummy variable

Global Variables:
basmva - system base MVA
dcsp_con - converter specification matrix
dcl_con - HVDC line specification matrix
dcc_con - HVDC pole control specification matrix
r_idx - rectifier index
i_idx - inverter index
n_dcl - number of HVDC lines
n_conv - number of HVDC converters
ac_bus - index of converter ac buses in the internal bus list
rec_ac_bus - index of rectifier ac buses in the internal bus list
inv_ac_bus - index of inverter ac buses in the internal bus list

Vdc - Matrix of HVDC voltages kV
i_dc - Matrix of HVDC line currents kA
dc_pot -
alpha - matrix of rectifier firing angles
gamma - matrix of inverter extinction angles
Vdc_ref - reference value for inverter extinction angle control
cur_ord - reference for current control at rectifier and inverter
dc_sig - external modulation control signal at rectifier and inverter
dcc_pot - matrix of pole control constants
i_dcr - rectifier line current kA
i_dci - inverter line current kA
v_dcc - HVDC line capacitance voltage kV
di_dcr - rate of change of rectifier HVDC line current
di_dci - rate of change of inverter HVDC line current
dv_dcc - rate of change of HVDC line capacitor voltage
v_conr - rectifier integral control state
dv_conr - rate of change of rectifier control state
v_coni - inverter integral control state
dv_coni - rate of change of inverter control state

Data Format:

The pole control data is specified in the matrix **dcc_con**

Table 1 HVDC Control Format

Column	Variable
1	Converter number
2	Proportional Gain
3	Integral Gain
4	Output Gain
5	Maximum Integral Limit
6	Minimum Integral Limit
7	Maximum Output Limit
8	Minimum Output Limit
9	Control Type

Note: the order of the converters in **dcc_con** must be the same as that in **dcsp_con**.

Algorithm:

Figure 1 shows the rectifier pole control block diagram. The control of the rectifier firing angle is by means of a proportional plus integral controller used to keep the HVDC line current at a value specified by **cur_ord**.

Figure 2 shows the inverter pole control block diagram. The control of the inverter extinction angle is by means of a proportional plus integral controller used to keep the inverter HVDC voltage at its initial value. If the inverter current falls below the inverter current order, the inverter pole control will take over current control.

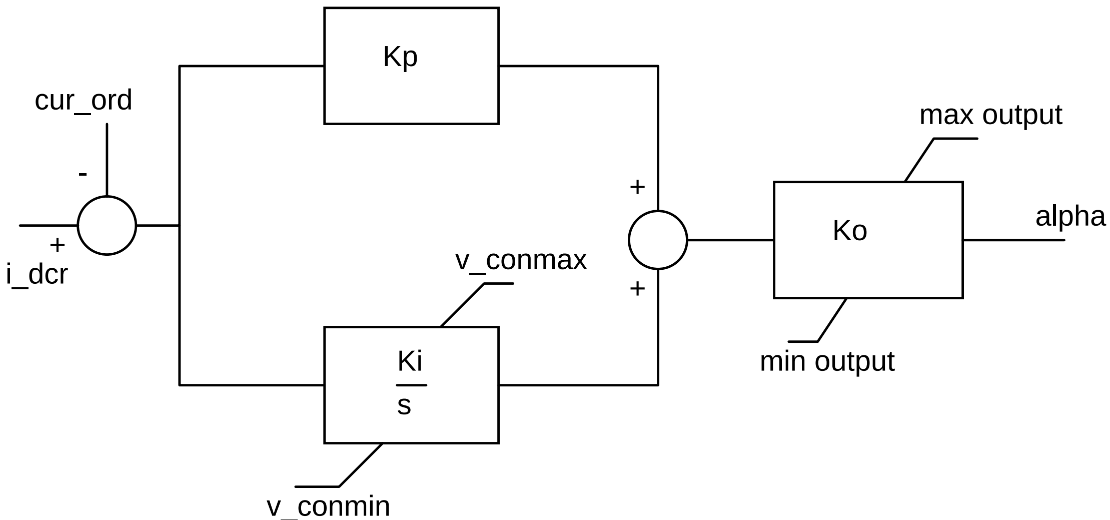


Figure 1 Rectifier Control Block Diagram

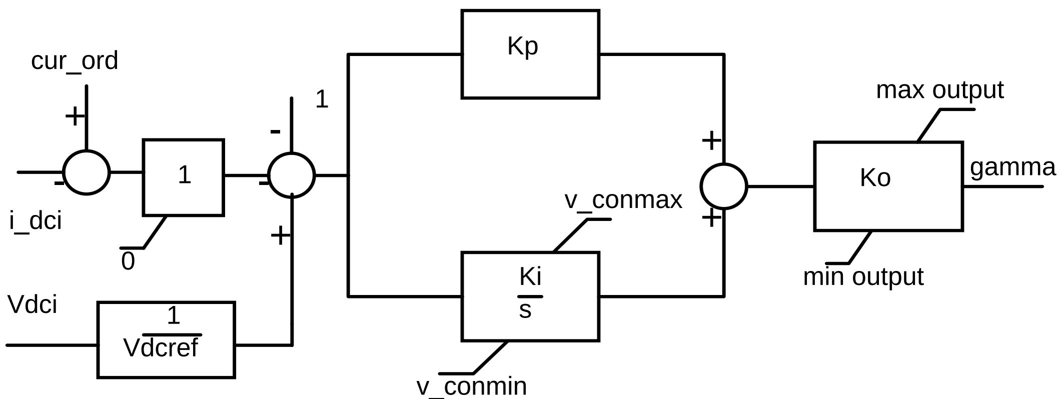


Figure 2 Inverter Pole Control Block Diagram

This algorithm is implemented in the M-file **dc_cont** in the POWER SYSTEM TOOLBOX.

dc_cur

Purpose:

Calculates the ac current load for use in the non-conforming load function **nc_load**

Syntax:

i_ac = dc_cur(V,k)

Description:

The function uses the current HT voltage estimate to determine the ac current load due to the HVDC links.

Inputs:

V - the current value of the equivalent HVDC HT terminal voltage
k - the current time step

Outputs:

i_ac - the ac load current in per unit due to the HVDC links

Global Variables:

r_idx - rectifier index
i_idx - inverter index
dcc_pot - dc control constant matrix
n_dcl - number of HVDC lines
basmva - system base MVA
i_dcr - rectifier current kA
i_dci - inverter current kA
alpha - rectifier firing angle
gamma - inverter extinction angle

Algorithm:

Calculates the HVDC voltages assuming that the currents, firing angle and extinction angle are constant. Calculates the equivalent active and reactive power load at the HVDC HT bus and from this calculates the equivalent alternating currents.

This algorithm is implemented in the M-file **dc_cur** in the POWER SYSTEM TOOLBOX.

dc_line

Purpose:
Forms the equations for HVDC line dynamics.

Syntax:

f = dc_line(i,k,bus,flag)

Description:

dc_line contains the equations necessary to model an HVDC line dynamically.

Inputs:

i   **i** = 0 all HVDC computations are performed using MATLAB vector methods
k   the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.
bus   the solved bus specification matrix
flag   indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1. For proper initialization, the corresponding generators must be initialized first.
- The network interface calculation is performed when **flag** = 1, and the field voltage of the synchronous machine is set to the exciter output voltage.
- The rates of change of the exciter states are calculated when **flag** = 2, using the generator terminal voltage and the external system values at the time specified by **k**

Outputs:

f   a dummy variable

Global Variables:
dcsp_con - HVDC converter specification matrix
dcl_con - HVDC line specification matrix
dcc_con - converter control specification matrix
dcc_pot - converter control constants matrix
dc_pot - line constants matrix
r_idx - rectifier index
i_idx - inverter index
n_dcl - number of HVDC lines
n_conv - number of HVDC converters
Vdc - HVDC voltages kV
i_dc - HVDC currents kA
no_cap_idx - index of HVDC lines with no capacitance specified
cap_idx - index of HVDC lines with capacitance specified

no_ind_idx - index of HVDC lines with no inductance specified
l_no_cap - number of HVDC lines with no capacitance
l_cap - number of HVDC lines with capacitance
i_dcr - rectifier HVDC line current kA
i_dci - inverter HVDC line current kA
v_dcc - HVDC line capacitance voltage kV
di_dcr - rate of change of rectifier dc line current
di_dci - rate of change of inverter dc line current
dv_dcc - rate of change of dc line capacitance voltage

Algorithm:

The HVDC line is modelled as a T equivalent. The smoothing reactors are included. The capacitance of the line may be set to zero. In this case, the inverter current is always equal to the rectifier current.

dc_load

Purpose:
Calculates the non-linear Jacobian elements for the changes in ac current injection changes in the real and imaginary parts of the equivalent HT terminal voltage

Syntax:

[Yrr,Yri,Yir,Yii] = dc_load(V,k)

Description:
Calculates:

$$Y_{rr} = \frac{\partial \, \mathrm{i}_{\mathrm{acr}}}{\partial \, \mathrm{V}_{\mathrm{r}}}$$

$$Y_{ri} = \frac{\partial \, \mathrm{i}_{\mathrm{acr}}}{\partial \, \mathrm{V}_{\mathrm{i}}}$$

$$Y_{ir} = \frac{\partial \, \mathrm{i}_{\mathrm{aci}}}{\partial \, \mathrm{V}_{\mathrm{r}}}$$

$$Y_{ii} = \frac{\partial \, \mathrm{i}_{\mathrm{aci}}}{\partial \, \mathrm{V}_{\mathrm{i}}}$$

Inputs:

V - the equivalent HT bus voltage
k - the current time step

Outputs:

$\mathbf{Y_{rr}}$, $\mathbf{Y_{ri}}$, $\mathbf{Y_{ir}}$, , $\mathbf{Y_{ii}}$

Global Variables:
i_dci - the inverter dc current
i_dcr - the rectifier dc current
dcc_pot - the dc control constants
alpha - the rectifier firing angle
gamma - the inverter extinction angle
basmva - the system base MVA
r_idx - the rectifier index
i_idx - the inverter index
n_conv - the number of HVDC converter buses
n_dcl - the number of HVDC lines

Algorithm:

This algorithm is implemented in the M-file **dc_load** in the POWER SYSTEM TOOLBOX.

desat

Purpose:
Calculates the describing function for saturation

Syntax:

g = dessat(a,isat)

Inputs:

a the input amplitude
isat the saturation amplitude

Outputs:

g the ratio of the amplitude of the fundamental of a sine wave of amplitude a clipped at isat to a

Algorithm:

The fundamental of the clipped sine wave is calculated from

$$
\begin{aligned}
g &= \frac{2}{\pi}(\tan^{-1} y + 0.5 * \sin(2y)) \quad & k \le 1 \\
&= 1 & k > 1 \\
&\text{where} \\
k &= \left|\frac{isat}{a}\right| \\
y &= \frac{k}{\sqrt{1-k^2}}
\end{aligned}
$$

Called by: mac_ind

The leakage inductances for the stator and rotor of an induction motor are calculated as

$$x_{sat} = x_{unsat}(1+g)/2$$

exc_dc12

Purpose:

Models IEEE Type DC1 and DC2 excitation system models

Synopsis:

f = exc_dc12(i,k,bus,flag)

Description:

exc_dc12(i,k,bus,flag) contains the equations of IEEE Type DC1 and DC2 excitation system models [1] (Figures 1, 2 and 3) for the initialization, machine interface and dynamics computation of the **i**$^{\mathbf{th}}$ excitation system.

Inputs:

i the number of the exciter
if **i** = 0 all dc exciters computations are performed using MATLAB vector methods. **This is the preferred mode.**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus the solved bus specification matrix

flag indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1. For proper initialization, the corresponding generators must be initialized first.
- The network interface calculation is performed when **flag** = 1, and the field voltage of the synchronous machine is set to the exciter output voltage.
- The rates of change of the exciter states are calculated when **flag** = 2, using the generator terminal voltage and the external system values at the time specified by **k**

Output:

f a dummy variable

Global Variables

Efd	E_{fd}	excitation output voltage (= field voltage) in pu
V_R	V_R	regulator output voltage in pu
V_A	V_A	regulator output voltage in pu
V_As	V_{As}	regulator voltage state variable in pu
R_f	R_f	stabilizing transformer state variable
V_FB	V_{FB}	feedback from stabilizing transformer
V_TR	V_{TR}	voltage transducer output in pu
V_B	V_B	potential circuit voltage output in pu
dEfd	dE_{fd}/dt	
dV_R	dV_R/dt	
dV_As	dV_{As}/dt	
dR_f	dR_f/dt	
dV_TR	dV_{TR}/dt	
exc_sig	V_{sup}	supplementary signal input to the summing junction
exc_pot		internally set matrix of exciter constants
exc_con		matrix of exciter data supplied by user

The m.file **pst_var.m** contains all the global variables required for **exc_dc12**, and should be loaded in the program calling **exc_dc12**.

Data Format

The exciter data is contained in the **i$^{\text{th}}$** row of the matrix variable **exc_con**. The data format for **exc_dc12** is shown in Table 1.

A constraint on using **exc_dc12** is that $T_F \neq 0$. All other time constants can be set to zero. If T_E is set to zero, then $E_{fd} = V_R$. K_F can be set to zero to model simple first order exciter models. The state V_R is prevented from exceeding its limits by a non_wind up limit.

If K_E is set to zero on input, its value will be computed during initialization to make $V_R=0$. If V_{Rmax} is set to zero on input, the values of V_{Rmax} and V_{Rmin} will be computed assuming that E_2 is the nominal ceiling value of E_{fd}.

Table 1 Data Format for exc_dc12

column	variable	unit
1	exciter type 1 for DC1 2 for DC2	
2	machine number	
3	input filter time constant T_R	sec
4	voltage regulator gain K_A	
5	voltage regulator time constant T_A	sec
6	voltage regulator time constant T_B	sec
7	voltage regulator time constant T_C	sec
8	max voltage regulator output V_{Rmax}	pu
9	min voltage regulator output V_{Rmin}	pu
10	exciter constant K_E	
11	exciter time constant T_E	sec
12	E_1	pu
13	saturation function $S_E(E_1)$	
14	E_2	pu
15	saturation function $S_E(E_2)$	
16	stabilizer gain K_F	
17	stabilizer time constant T_F	sec

Algorithm:

Based on the exciter block diagram, the exciter is initialized using the generator field voltage E_{fd} to compute the state variables. In the network interface computation, the exciter output voltage is converted to the field voltage of the synchronous machine. In the dynamics calculation, generator terminal voltage and the external signal is used to calculate the rates of change of the excitation system states.

This algorithm is implemented in the M-file **exc_dc12.m** in the POWER SYSTEM TOOLBOX.

See also: loadflow, pst_var, smpexc, exc_st3, mac_tra, mac_sub.

Reference:

1. IEEE Committee Report, "Excitation System Models for Power System Stability Studies," *IEEE Transactions of Power Apparatus and Systems*, vol. PAS-100, pp. 494-509, 1981.

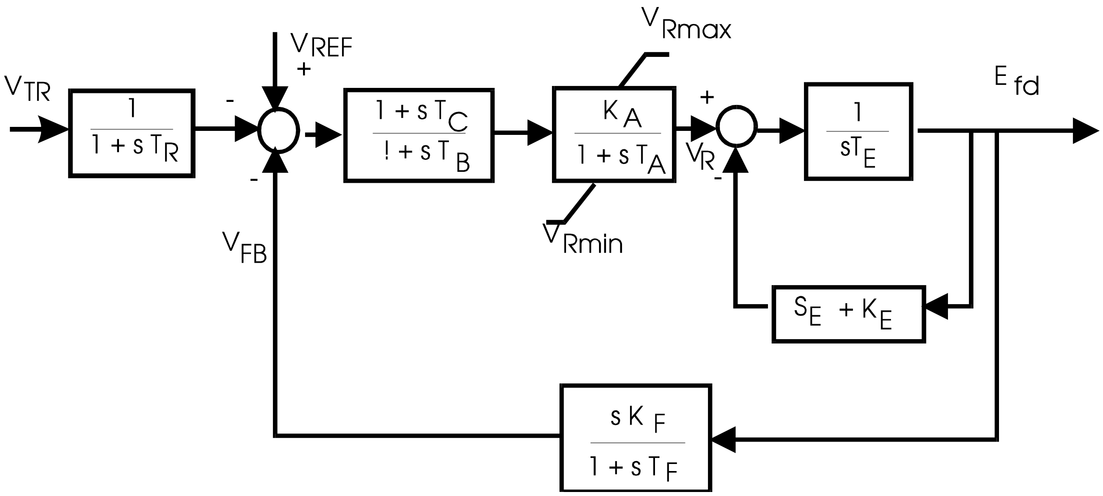


Figure 3 DC Exciter Type 1

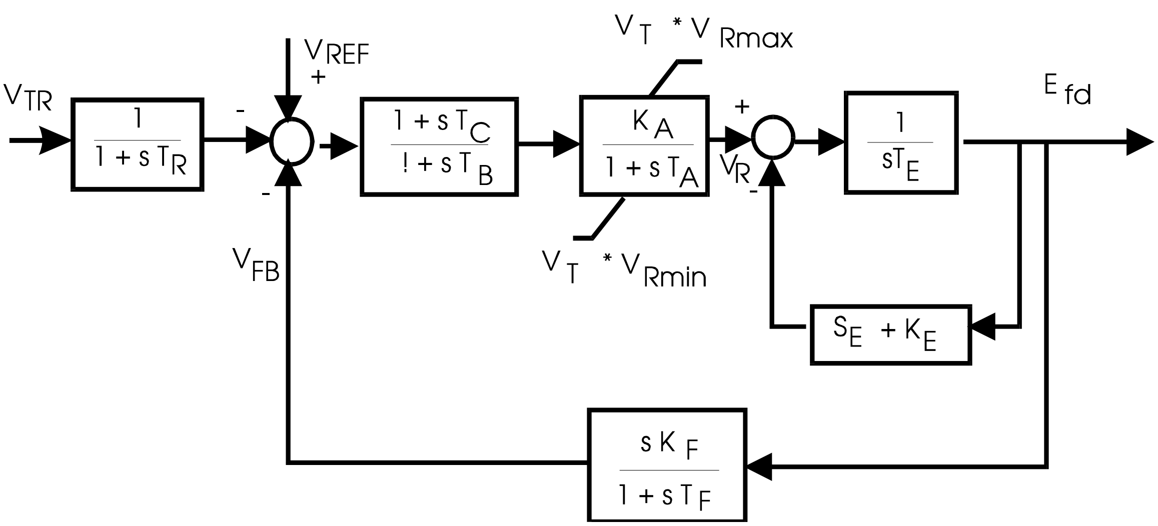


Figure 4 DC Exciter Type 2

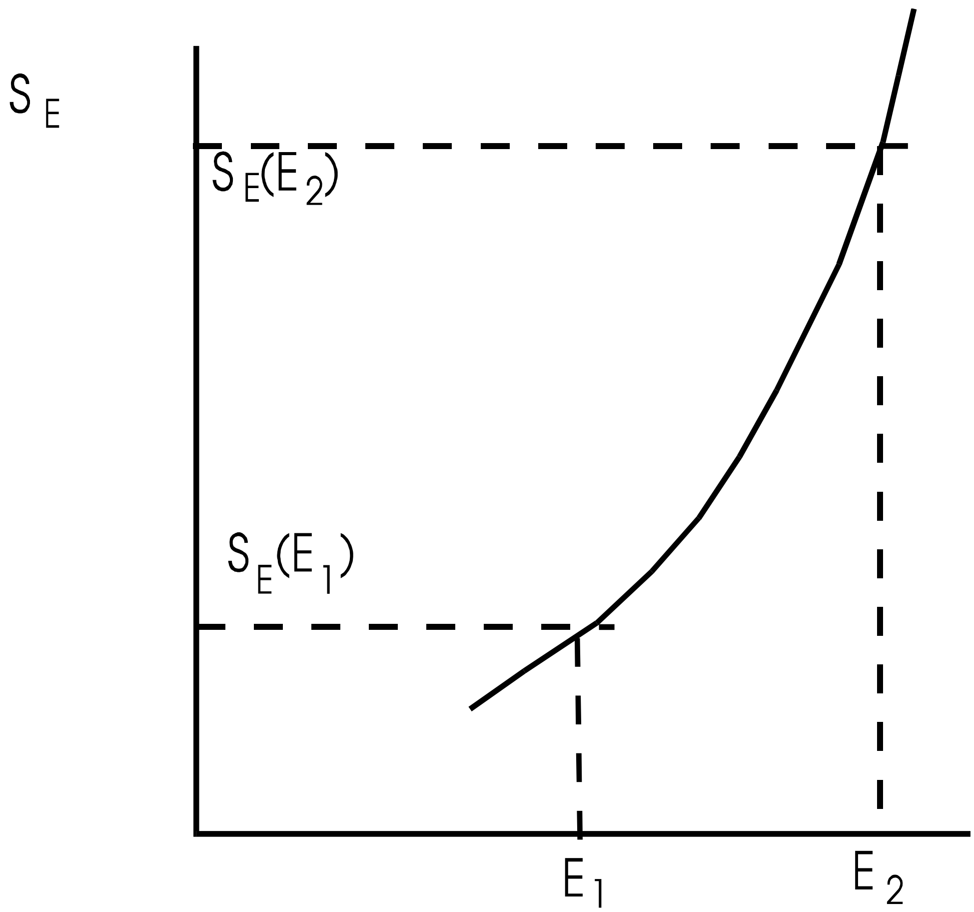


Figure 5 Exciter Saturation Function

exc_indx

Purpose:

Forms indexes for the exciters to enable vector computation to be used with mixed exciter models.

Syntax:

f = exc_indx

Description:

f = exc_indx checks the exciter input matrix **exc_con** for the type of exciter and the parameters specified. It produces indexes for the various exciter types and their parameters which are used in the corresponding model functions.

Outputs:

f is a dummy variable.

Global Variables

Exciter Indexes

exc_pot - exciter constants calculated on ititialization
exc_con - exciter data specification matrix
n_exc - number of exciters
smp_idx - index of simple exciters
n_smp - number of simple exciters
dc_idx - index of dc exciters
n_dc - number of dc exciters
dc2_idx - index of type 2 dc exciters
n_dc2 - number of type 2 dc exciters
st3_idx - index of st3 exciters
n_st3 - number of st3 exciters

Variable Indexes

smp_TA - the value of T_A for simple exciters (exc_con(smp_idx,5))
smp_TA_idx - the index of simple exciters having a T_A > 0.01s
smp_noTA_idx - the index of simple exciters having a T_A < 0.01s
smp_TB - the value of T_B for simple exciters
smp_TB_idx - the index of simple exciters having a T_B > 0.01s
smp_noTB_idx - the index of simple exciters having a T_B< 0.01s
smp_TR - the value of T_R for simple exciters
smp_TR_idx - the index of simple exciters having a T_R > 0.01s exciters
smp_noTR_idx - the index of simple exciters having a T_R < 0.01s

dc_TA - the value of T_A for dc exciters (exc_con(dc_idx,5))
dc_TA_idx - the index of dc exciters having a T_A > 0.01s
dc_noTA_idx - the index of dc exciters having a T_A < 0.01s
dc_TB - the value of T_B for dc exciters
dc_TB_idx - the index of dc exciters having a T_B > 0.01s
dc_noTB_idx- the index of dc exciters having a T_B < 0.01s
dc_TE - the value of T_E for dc exciters
dc_TE_idx - the index of dc exciters having a T_E > 0.01s
dc_noTE_idx - the index of dc exciters having a T_E < 0.01s
dc_TF - the value of T_F for dc exciters
dc_TF_idx - the index of dc exciters having a T_F > 0.01s
dc_TR - the value of T_R for dc exciters
dc_TR_idx - the index of dc exciters having a T_R > 0.01s
dc_noTR_idx - the index of dc exciters having a T_R < 0.01s
st3_TA - the value of T_A for st3 exciters
st3_TA_idx - the index of st3 exciters having a T_A > 0.01s
st3_noTA_idx - the index of st3 exciters having a T_A < 0.01s
st3_TB - the value of T_B for st3 exciters
st3_TB_idx - the index of st3 exciters having a T_B > 0.01s
st3_noTB_idx - the index of st3 exciters having a T_B < 0.01s
st3_TR - the value of T_R for st3 exciters
st3_TR_idx - the index of st3 exciters having a T_R > 0.01s
st3_noTR_idx - the index of st3 exciters having a T_R < 0.01s

Algorithm

This algorithm is implemented in the M-file **exc_indx.m** in the POWER SYSTEM TOOLBOX.

exc_st3

Purpose:

Models IEEE Type ST3 compound source rectifier exciter models

Synopsis:

f = exc_st3(i,k,bus,flag)

Description:

exc_st3(i,k,bus,flag) contains the equations of IEEE Type ST3 excitation system models [1] for the initialization, network interface and dynamics computation
of the **i**$^{\mathbf{th}}$ excitation system. The block diagram is shown in Figure 1.

The m.file **pst_var.m** containing all the global variables required for **exc_st3** should be loaded in the program calling **exc_st3**. The exciter data is contained in the **i**$^{\mathbf{th}}$ row of the matrix variable **exc_con**.

Inputs:

i the number of the exciter
if **i** = 0 all st_3 exciter computations are performed using MATLAB vector methods.
This is the preferred mode.

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus the solved bus specification matrix

flag indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1. For proper initialization, the corresponding generators must be initialized first.
- The network interface calculation is performed when **flag** = 1, and the field voltage of the synchronous machine is set to the exciter output voltage.
- The rates of change of the exciter states are calculated when **flag** = 2, using the generator terminal voltage and the external system values at the time specified by **k**

Output:

f a dummy variable

Global Variables:

System variables

psi_re	ψ_{re}	real and imaginary components of voltage
psi_im	ψ_{im}	source on system reference frame
cur_re	i_{re}	real and imaginary components of bus
cur_im	i_{im}	current on system reference frame
bus_int		array to store internal bus ordering

Synchronous Generator Variables

mac_ang	δ	machine angle in rad/sec
mac_spd	ω	machine speed in pu
eqprime	E_q'	pu on machine base
edprime	E_d'	pu on machine base
psikd	ψ_{kd}	pu on machine base
psikq	ψ_{kq}	pu on machine base
curd	i_d	d-axis current on system base
curq	i_q	q-axis current on system base
curdg	i_{dg}	d-axis current on machine base
curqg	i_{qg}	q-axis current on machine base
fldcur	i_{fd}	field current on machine base
psidpp	ψ_d''	pu on machine base
psiqpp	ψ_q''	pu on machine base
vex	V_{ex}	field voltage on machine base
eterm	E_T	machine terminal voltage in pu
theta	θ	terminal voltage angle in rad
ed	E_d	d-axis terminal voltage in pu
eq	E_q	q-axis terminal voltage in pu
pmech	P_m	mechanical input power in pu
pelect	P_e	electrical active output power in pu
qelect	Q_e	electrical reactive output power in pu
mac_int		array to store internal machine ordering
mac_pot		internally set matrix of machine constants
mac_con		matrix of generator parameters set by user
n_mac		number of generators
n_em		number of em (classical) generator models
n_tra		number of transient generator models
n_sub		number of subtransient generator models
mac_tra_idx		index of transient generator models
mac_sub_idx		index of subtransient generator models

Exciter Variables

Efd	E_{fd}	exciter output voltage - generator field voltage pu
V_R	V_R	regulator output voltage in pu

V_A	V_A	regulator output voltage in pu
V_As	V_{As}	regulator voltage state variable in pu
R_f	R_f	stabilizing transformer state variable
V_FB	V_{FB}	feedback from stabilizing transformer
V_TR	V_{TR}	voltage transducer output in pu
V_B	V_B	potential circuit voltage output in pu
dEfd	dE_{fd}/dt	
dV_R	dV_R/dt	
dV_As	dV_{As}/dt	
dR_f	dR_f/dt	
dV_TR	dV_{TR}/dt	
exc_sig	V_{sup}	supplementary input signal to exciter ref input
exc_pot		matrix of internally set exciter constants
exc_con		matrix of exciter data set by user
st3_idx		index of st3 exciters
n_st3		number of st3 exciters
st3_TA		**exc_con(st3_idx,5)**
st3_TA_idx		index of nonzero TA for st3 exciter
st3_noTA_idx		index of zero TA for st3 exciter
st3_TB		**exc_con(st3_idx,6)**
st3_TB_idx		index of nonzero TB for st3 exciter
st3_noTB_idx		index of zero TB for st3 exciter
st3_TR		**exc_con(st3_idx,3)**
st3_TR_idx		index of nonzero TR for st3 exciter
st3_noTR_idx		index of zero TR for st3 exciter

Data Format:

The data format for **exc_st3** is given in Table 1.

The time constants T_R and T_B can be set to zero if desired. However, T_A cannot be set to zero.

Table 1 Data format for model exc_st3

column	data	unit
1	exciter type	3 for ST3
2	machine number	
3	input filter time constant T_R	sec
4	voltage regulator gain K_A	
5	voltage regulator time constant T_A	sec
6	voltage regulator time constant T_B	sec
7	voltage regulator time constant T_C	sec
8	maximum voltage regulator output V_{Rmax}	pu
9	minimum voltage regulator output V_{Rmin}	pu
10	maximum internal signal V_{Imax}	pu
11	minimum internal signal V_{Imin}	pu
12	first state regulator gain K_J	
13	potential circuit gain coefficient K_P	
14	potential circuit phase angle q_P	degrees
15	current circuit gain coefficient K_I	
16	potential source reactance X_L	pu
17	rectifier loading factor K_C	
18	maximum field voltage E_{fdmax}	pu
19	inner loop feedback constant K_G	
20	maximum inner loop voltage feedback V_{Gmax}	pu

Example:

A typical data set for **st3** exciters is

exc_con =

3 1 0 7.04 0.4 6.67 1.0 7.57 0 0.2 -0.2 200 4.365 ....
20 4.83 0.091 1.096 6.53 1 6.53

Algorithm:

Based on the exciter block diagram, the exciter is initialized using the generator field voltage E_{fd} to compute the state variables. In the network interface computation, the exciter output voltage is converted to the field voltage of the synchronous machine. In the dynamics calculation, generator terminal voltage and the external signal is used to calculate the rates of change of the excitation system states.

This algorithm is implemented in the M-file **exc_st3.m** in the POWER SYSTEM TOOLBOX.

See also: loadflow, pst_var, exc_dc12, smpexc

Reference:

1. IEEE Committee Report, ``Excitation System Models for Power System Stability Studies," *IEEE Transactions of Power Apparatus and Systems*, vol. PAS-100, pp. 494-509, 1981.

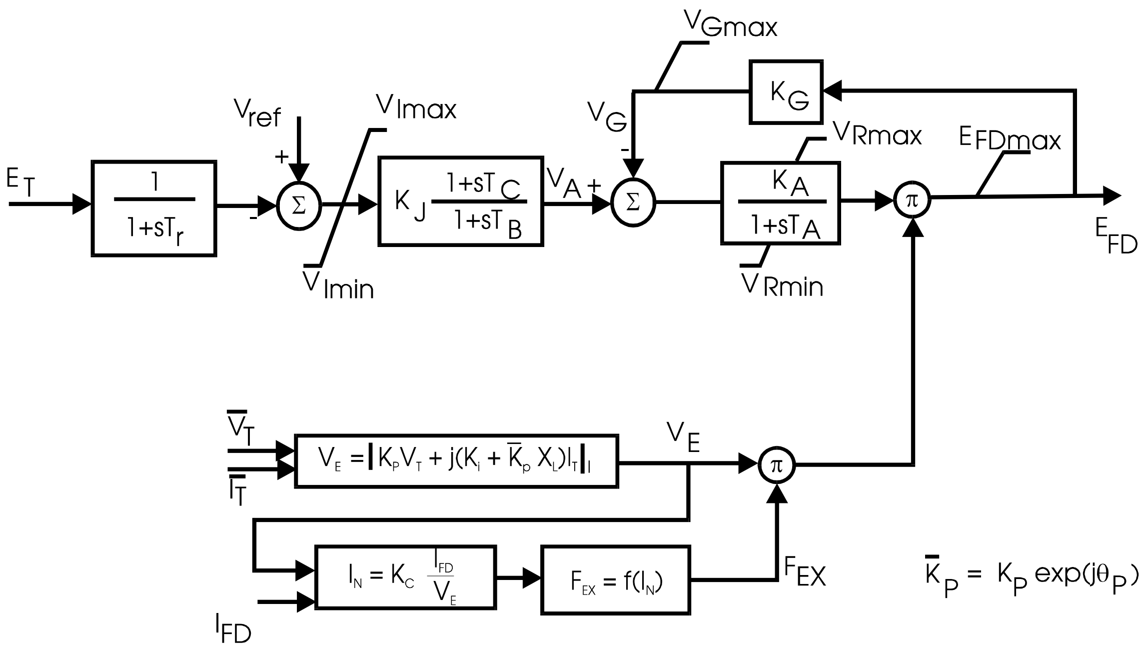


Figure 6 ST3 Excitation System

Imtspeed

Purpose:
Calculates the torque, power, reactive power and stator current as slip varies from 0 to 1.

Syntax:

[t,p,q,is,s]=imtspeed(V,rs,xs,Xm,rr,xr,rr2,xr2,dbf,isat)

Inputs:

V	stator voltage magnitude PU on motor base
rs	stator resistance PU on motor base
xs	stator leakage reactance PU on motor base
Xm	magnetizing reactance PU on motor base
rr	rotor reactance PU on motor base if double cage, the first cage resitance if deep bar the bar resistance at zero slip
xr	rotor leakage reactance PU on motor base if double cage, the leakage reactance of the first cage
rr2	the rotor resistance of the second cage PU on motor base zero if single cage or deep bar rotor
xr2	the rotor inter-cage leakage reactance PU on motor base zero if single cage or deep bar rotor
dbf	deep bar factor zero if motor single or double cage
isat	the current at which leakage inductance saturation occurs

Outputs:

t	torque
p	power
q	reactive power
is	stator current
s	slip

i_simu

Purpose:

To set the reduced Y matrix and the voltage recovery matrix to the appropriate values for the switching condition. Calculates the generator currents , the induction motor and generator currents and powers, the ac voltages (magnitudes and angles) and the HVDC voltages and currents.

Syntax:

function h_sol = i_simu(k,ks,k_inc,h,ntot,bus_sim,Y_r,rec_V,bo)

Inputs:

ks - indicates the switching times
k - the current time step
k_inc - the number of time steps between switching points
h - vector of time steps
ntot - total number of machines (gen + motor)
bus_sim - value of bus matrix at this switching time
Y_r - reduced Y matrix at this switching time
rec_V - voltage recovery matrix at this switching time
bo - bus order for this switching time

Outputs:

h_sol - the time step at this value of k_s

Global variables:

psi_re - real part of generator internal bus voltage
psi_im - imaginary part of generator internal bus voltage
vdp - induction motor d axis voltage behind transient impedance
vqp - induction motor q axis voltage behind transient impedance
n_mot - number of induction motors
n_conv - number of HVDC converters
nload - number of non-conforming load buses
bus_int - internal bus number vector
cur_re - real part of generator current
cur_im - imaginary part of generator current
idmot - d axis motor current
iqmot - q axis motor current
p_mot - motor active power
q_mot - motor reactive power
idig - d axis induction generator current
iqig - q axis induction generator current
pig - induction generator active power

qig - induction generator ractive power

Algorithm:

This algorithm is implemented in the M-file **i_simu** in the POWER SYSTEM TOOLBOX.

line_pq

Purpose:

Line power flow computation

Synopsis:

[S1,S2] = line_pq(V1,V2,R,X,B,tap,phi)

Description:

line_pq(V1,V2,R,X,B,tap,phi) computes the power flow on transmission lines. with resistance **R**, reactance **X**, line charging **B**, tap ratio **tap** and phase shifter angle **phi** (in degrees). The voltages **V1** and **V2** describe the from and to bus voltages respectively. They may be vectors, or they may be a matrix, such as that obtained at the end of a transient simulation,
i.e., **V1** may have the form **V1**(1:number of buses, 1:number of time steps).

The tap is at the from bus and represents the step down ratio,
i.e. V1' = V1/(tap*exp(j*phi*pi/180)); and i1' = i1*tap*exp(j*phi*pi/180)

Note: V1 and **V2** must have the same size

Inputs:

V1	from bus complex voltage matrix
V2	to bus complex voltage matrix
R	line resistance vector
X	line reactance vector
B	line charging vector
tap	tap ratio vector
phi	phase shifter angle vector in degrees

Outputs:

S1	complex power injection matrix at from bus
S2	complex power injection matrix at to bus

Algorithm:

This algorithm is implemented in the M-file **line_pq** in the POWER SYSTEM TOOLBOX.

Example:

To calculate the complex power flow from transient simulation records

Set: V1 = bus_v(bus_int(line(:,1)),:) the from bus voltages
Set: V2 = bus_v(bus_int(line(:,2)),:) the to bus voltages
Set: R = line(;,3); X = line(:,4); B = line(;,5)
Set: tap = line(:,6); phi = line(:,7)
make the call: [S1,S2] = line_pq(V1,V2,R,X,B,tap,phi)
The power flow at the from bus on any line may then be plotted using
plot(t,real(S1(line_number,:))

lmod

Purpose:

A load modulation control for transient simulation

Synopsis:

f = lmod(i,k,bus,flag)

Description:

f = lmod(i,k,bus,flag) contains the equations of a load modulation control system for the initialization, machine interface and dynamics computation of the **i**$^{\mathbf{th}}$ load modulation control.

Modulation is controlled through the global variable **lmod_sig**. This is modified by the function **ml_sig** which should be written by the user to obtain the required load modulation charactersitic.

The m.file **pst_var.m** containing all the global variables required for **lmod** should be loaded in the program calling **lmod**.

Inputs:

i	the number of the load modulation control if **i** = 0 all load modulation computations are performed using MATLAB vector methods. **This is the preferred mode.**
k	the integer time step in a simulation In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.
bus	the solved bus specification matrix
flag	indicates the mode of solution • Initialization is performed when **flag** = 0 and **k** = 1. • There is no need to perform a network interface calculation for **lmod** • The rates of change of the lmod state is calculated when **flag** = 2, using the modulating signal **lmod_sig** at the time specified by **k**

Output:

f is a dummy variable

Global Variables

System variables

basmva	system base MVA
bus_int	array to store internal bus ordering

Load Modulation Variables

lmod_st	*lm*	load modulation state
dlmod_st	*dlm/dt*	
lmod_sig	V_{sup}	supplementary signal into the reference input
lmod_con		matrix of lmod parameters supplied by user
lmod_pot		internally calculated matrix of lmod constants
n_lmod	number of load modulation controls	
lmod_idx		index of modulation controls included in **load_con**

Data Format

The load modulation control data is contained in the $\mathbf{i^{th}}$ row of the matrix **lmod_con**. The data format for **lmod_con** is given in Table 1.

Table 1. Data format for lmod

column	variable	unit
1	load modulation number	
2	bus number	
3	modulation base MVA	MVA
4	maximum conductance *lmod_max*	pu
5	minimum conductance *lmod_min*	pu
6	regulator gain K	pu
7	regulator time constant T_R	sec

Algorithm:

To use the **lmod** function, the load modulation buses must be declared via **load_con** as non-conforming load buses. The **lmod** buses may also have non-conforming loads In the network interface computation, the load modulation output is used to adjust the conductance at the control buses in the solution for the bus voltages in **nc_load**. In the dynamics calculation, the rate of change of the load modulation control state is adjusted according to the signal **lmod_sig**. An anti-windup limit is used to reset the state variable.

This algorithm is implemented in the M-file **lmod** in the POWER SYSTEM TOOLBOX.

See also: nc_load, pst_var, ml_sig.

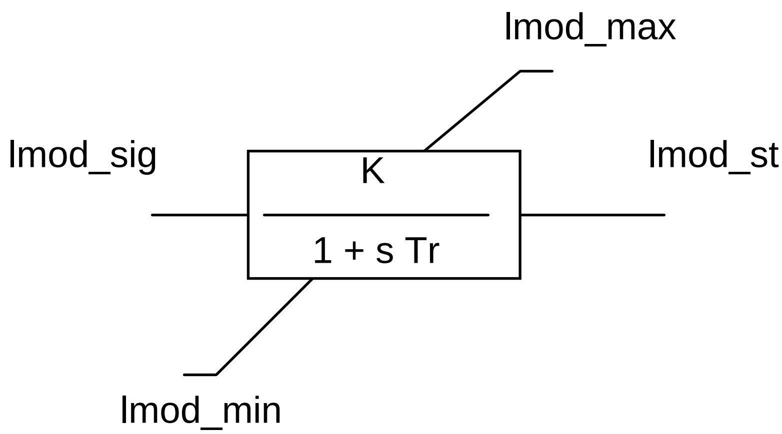


Figure 7 Load Modulation Control Block Diagram

mac_em

Purpose:

Model a synchronous machine with the classical electromechanical model

Synopsis:

f = mac_em(i,k,bus,flag)

Description:

mac_em(i,k,bus,flag) contains the electromechanical model equations for the initialization, network interface and dynamics computation of the **i^th^** synchronous machine.

The m.file **pst_var.m** containing all the global variables required for **mac_em** should be loaded in the program calling **mac_em**.

Inputs:

i the number of the generator
if **i** = 0 all em generator computations are performed using MATLAB vector methods. **This is the preferred mode.**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus the solved bus specification matrix

flag indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1.
- The network interface calculation is performed when **flag** = 1
- The rates of change of the em generator states are calculated when **flag** = 2, using the generator terminal voltage and the external system values at the time specified by **k**

Output:

f a dummy variable

Global Variables:

System variables

psi_re	y_{re}	real and imaginary components of voltage
psi_im	y_{im}	source on system reference frame
cur_re	i_{re}	real and imaginary components of bus
cur_im	i_{im}	current on system reference frame
bus_int		array to store internal bus ordering

Synchronous Generator Variables

mac_ang	δ	machine angle in rad/sec
mac_spd	ω	machine speed in pu
eqprime	E_q'	pu on machine base
edprime	E_d'	pu on machine base
psikd	ψ_{kd}	pu on machine base
psikq	ψ_{kq}	pu on machine base
curd	i_d	d-axis current on system base
curq	i_q	q-axis current on system base
curdg	i_{dg}	d-axis current on machine base
curqg	i_{qg}	q-axis current on machine base
fldcur	I_{fd}	field current on machine base
psidpp	ψ_d''	pu on machine base
psiqpp	ψ_q''	pu on machine base
vex	V_{ex}	field voltage on machine base
eterm	E_T	machine terminal voltage in pu
theta	θ	terminal voltage angle in rad
ed	E_d	d-axis terminal voltage in pu
eq	E_q	q-axis terminal voltage in pu
pmech	P_m	mechanical input power in pu
pelect	P_e	electrical active output power in pu
qelect	Q_e	electrical reactive output power in pu
mac_int		array to store internal machine ordering
mac_pot		internally set matrix of machine constants
mac_con		matrix of generator parameters set by user
n_mac		number of generators
n_em		number of em (classical) generator models
n_tra		number of transient generator models
n_sub		number of subtransient generator models
mac_tra_idx		index of transient generator models
mac_sub_idx		index of subtransient generator models

Data Format

The machine data is contained in the **i**$^{\mathbf{th}}$ row of the matrix variable **mac_con**. The data format for **mac_em** is shown in Table 1.

Table 1. Data for mac_em

column	variable	unit
1	machine number	
2	bus number	
3	base MVA	MVA
7	d-axis transient reactance x_d'	pu
16	Inertia Constant H	sec
17	damping coefficient d_o	pu
19	bus number	
22	active power fraction	
23	reactive power fraction	

Generators are numbered internally according to the order of the machines in **mac_con**. This information is contained in the array **mac_int** and is set up automatically by the Y matrix reduction function **red_ybus**.

Example:

The generator data in the 3 machine, 9 bus system [1] are

mac_con =

```
1  1  100  0  0  0  0.0608  0  0  0  0  0  0  0  0  23.64  9.6  0  1
2  2  100  0  0  0  0.1198  0  0  0  0  0  0  0  0  6.4    2.5  0  2
3  3  100  0  0  0  0.1813  0  0  0  0  0  0  0  0  3.01   1.0  0  3
```

Note that if the power fractions are left out of **mac_con**, they will be set to unity.

Algorithm:

Based on the generator vector diagram

- the initialization uses the solved loadflow bus voltages and angles to compute the internal voltage and the rotor angle. The d-axis voltage is identically zero for all time.
- the network interface computation generates the voltage behind the transient reactance on the system reference frame.
- in the dynamics calculation, the rotor torque imbalance and the speed deviation are used to compute the rates of change of the two state variables mac_ang and mac_spd.

This algorithm is implemented in the M-file **mac_em.m** in the POWER SYSTEM TOOLBOX.

See also: loadflow, mac_tra, mac_sub.

Reference:

1. J.H. Chow, editor, *Time-Scale Modeling of Dynamic Networks with Applications to Power Systems*, Springer-Verlag, Berlin, 1982.

mac_ib

Purpose:

Model a synchronous generator as an infinite bus

Synopsis:

f = mac_ib(i,k,bus,flag)

Description:

mac_ib(i,k,bus,flag) contains routines for the initialization, network interface and dynamics computation of the **i**$^{\mathbf{th}}$ synchronous machine modelled as an infinite bus.

The m.file **pst_var.m** containing all the global variables required for **mac_ib** should be loaded in the program calling **mac_ib**.

Inputs:

i the number of the generator
if **i** = 0 all em generator computations are performed using MATLAB vector methods. **This is the preferred mode.**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus the solved bus specification matrix

flag indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1.
- The network interface calculation is performed when **flag** = 1
- The rates of change of the em generator states are calculated when **flag** = 2, using the generator terminal voltage and the external system values at the time specified by **k**

Output:

f a dummy variable

Global Variables:

System variables

basmva		system base MVA
basrad		2 π* system frequency
sys_freq		system frequency in pu
bus_v	V	bus voltage magnitude in pu
bus_ang	θ	bus voltage angle in rad
psi_re	y_{re}	real and imaginary components of voltage
psi_im	y_{im}	source on system reference frame
cur_re	i_{re}	real and imaginary components of bus
cur_im	i_{im}	current on system reference frame
bus_int		array to store internal bus ordering

Synchronous Generator Variables

mac_ang	δ	machine angle in rad/sec
mac_spd	ω	machine speed in pu
eqprime	E_q'	pu on machine base
edprime	E_d'	pu on machine base
psikd	ψ_{kd}	pu on machine base
psikq	ψ_{kq}	pu on machine base
curd	i_d	d-axis current on system base
curq	i_q	q-axis current on system base
curdg	i_{dg}	d-axis current on machine base
curqg	i_{qg}	q-axis current on machine base
fldcur	I_{fd}	field current on machine base
psidpp	ψ_d''	pu on machine base
psiqpp	ψ_q''	pu on machine base
vex	V_{ex}	field voltage on machine base
eterm	E_T	machine terminal voltage in pu
theta	θ	terminal voltage angle in rad
ed	E_d	d-axis terminal voltage in pu
eq	E_q	q-axis terminal voltage in pu
pmech	P_m	mechanical input power in pu
pelect	P_e	electrical active output power in pu
qelect	Q_e	electrical reactive output power in pu
mac_int		array to store internal machine ordering
mac_pot		internally set matrix of machine constants
mac_con		matrix of generator parameters set by user
ibus_con		vector specifying infinite buses set by user
n_ib		number of generators modelled as infinite buses
n_ib_em		number of em (classical) generators modeled as infinite buses
n_ib_tra		number of transient generators modeled as infinite buses
n_ib_sub		number of subtransient generators modeled as infinite buses

mac_ib_idx index of generators modeled as infinite buses
not_ib_idx index of generators not modelled as infinite buses

Data Format

The infinite buses are specified in the vector **ibus_con**. The vector is of length equal to the number of generators. It has zero entries for non-infinite bus generators and unity for infinite bus generators.

Example:

To represent generator 2 in the single generator infinite bus system as an infinite bus

ibus_con = [0 1]';

```
 mac_con = [
1 1 991     0.15 0  2.0    0.245 0.2  5.0  0.031 ...
                    1.91   0.42  0.2  0.66 0.061 ...
                    2.8756 0.0   0    1    0       0;
2 2 100000 0.00  0  0.     0.01  0    0    0     ...
                    0      0     0    0    0     ...
                    3.0    2.0   0    2    0       0];
```

Algorithm:

On initialization the internal voltage behind either transient or subtransient impedance is determined. Thereafter this voltage is maintained constant.

This algorithm is implemented in the M-file **mac_ib.m** in the POWER SYSTEM TOOLBOX.

See also: loadflow, mac_em, mac_tra, mac_sub.

mac_igen

Purpose:

Models a single cage induction generator

Synopsis:

[bus_new] = mac_igen(i,k,bus,flag)

Description:

mac_igen(i,k,bus,flag) contains the model equations for the initialization, network interface and dynamics computation of induction generators.

The m.file **pst_var.m** containing all the global variables required for **mac_igen** should be loaded in the program calling **mac_igen**.

The induction generators are numbered internally according to the order of the machines in **igen_con**. This information is contained in the array **igen_int** and is set up automatically by the Y matrix reduction function **red_ybus**.

Note: The induction generator is modelled as a negative load in the loadflow, since induction generators cannot control voltage. The generator reactive power is not known until after the generator is initialized. After initialization, **bus_new** contains the load data with the generator active and reactive powers subtracted from the load specified in the original data file. This means that the induction generators must be initialized before the reduced y matrices are determined.

Inputs:

i = 0 ; **vector computation is the only option for induction generators**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus the solved bus specification matrix

flag indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1.
- The network interface calculation is performed when **flag** = 1
- The rates of change of the induction motor states are calculated when **flag** = 2, using the motor terminal voltage and the motor load torque at the time specified by **k**

Output:

bus_new a modified **bus** matrix, in which the induction generator active and reactive powers are subtracted from the original load active and reactive powers

Global Variables:

System Variables

basmva	system base MVA
basrad	2 π* system frequency
bus_int	array to store internal bus ordering

Induction Generator Variables

tmig	induction generator mechanical torque pu on motor base
pig	generator active power in p.u. on generator base
qig	generator reactive power in p.u. on generator base
vdig	generator direct axis stator voltage in p.u.
vqig	generator quadrature axis stator voltage in p.u.
idig	generator direct axis stator current in p.u.
iqig	generator quadrature axis voltage im p.u.
igen_con	matrix of induction generator parameters set by user
igen_pot	matrix of induction generator constants set internally
igen_int	index of internal induction generator buses
igbus	buses to which induction generators are connected
vdpig	V'd direct axis transient voltage for induction generators (state)
vqpig	V'q quadrature axis transient voltage for induction generators (state)
slig	fractional slip (state)
dvdpig	dV'_d/dt
dvqpig	dV'_q/dt
dslig	ds/dt

Data Format:

The induction generator data is contained in the **ith** row of the matrix variable **igen_con**. The data format for **mac_igen** is shown in Table 1.

Table 1. Data for mac_igen

column	variable	unit
1	generator number	
2	bus number	
3	generator base MVA	MVA
4	stator resistance r_s	pu
5	stator leakage reactance x_s	pu
6	magnetizing reactance X_m	pu
7	rotor resistance r_r	pu
8	rotor leakage reactance x_r	pu
9	inertia constant H of generator plus turbine	sec
15	fraction of active bus load	

Example:

The induction generator data in the 3 machine, 9 bus system are

igen_con =

1 8 60 0.001 0.01 3. 0.009 0.01 0.7 0 0 0 0 0 1

Algorithm:

Initialization (flag = 0) uses the solved load flow bus voltages and angles to compute the slip required to generate the specified power. The power is specified as a fraction of the load at the specified load bus. This should be set to a negative value in the load flow specification matrix. The slip is calculated using a Newton Raphson iteration. Failure to converge within 30 iterations causes an error message to be generated. Once the initial slip is known, the generator's reactive power is calculated. The generator's real and reactive powers are then subtracted from the corresponding bus loads.

The dynamic model is that formulated by Brereton, Lewis and Young[1] for an induction motor. In this model the three states are the d and q voltages behind transient reactance and the slip. For an induction generator, the initial slip is negative

This algorithm is implemented in the M-file **mac_igen.m** in the POWER SYSTEM TOOLBOX.

See also: loadflow, mac_tra, mac_sub, mac_ind, red_ybus.

Reference:

1. D.S. Brereton, D.G. Lewis and C.C. Young, " Representation of Induction Motor Loads during Power System Stability Studies", AIEE Trans, vol 76, Part III, August 1957, pp 451-460.

mac_ind

Purpose:

Models a single cage induction motor.

Synopsis:

[bus_new] = mac_ind(i,k,bus,flag)

Description:

mac_ind(i,k,bus,flag) contains the model equations for the initialization, network interface and dynamics computation of induction motors.

The m.file **pst_var.m** containing all the global variables required for **mac_ind** should be loaded in the program calling **mac_ind**.

The induction motors are numbered internally according to the order of the machines in **ind_con**. This information is contained in the array **ind_int** and is set up automatically by the Y matrix reduction function **red_ybus**.

Note: The motor reactive power is not known until after the motor is initialized. After initialization, **bus_new** contains the load data with the motor real and reactive load powers subtracted from the load specified in the original data file. This means that the motors must be initialized before the reduced y matrices are determined.

Inputs:

i the number of the induction motor
if **i** = 0 all induction motor computations are performed using MATLAB vector methods. **This is the preferred mode.**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus the solved bus specification matrix

flag indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1.
- The network interface calculation is performed when **flag** = 1
- The rates of change of the induction motor states are calculated when **flag** = 2, using the motor terminal voltage and the motor load torque at the time specified by **k**

Output:

bus_new a modified **bus** matrix, in which the motor active and reactive powers are subtracted from the original load active and reactive powers

Global Variables:

System Variables

basmva	system base MVA
basrad	2 π* system frequency
bus_int	array to store internal bus ordering

Induction Motor Variables

tload	motor load torque
t_init	initial motor load torque in pu. on motor base
p_mot	motor active power in pu. on system base
q_mot	motor reactive power in pu. on system base
vdmot	motor direct axis stator voltage in pu.
vqmot	motor quadrature axis stator voltage in pu.
idmot	motor direct axis stator current in pu.
iqmot	motor quadrature axis voltage im pu.
ind_con	matrix of induction motor parameters set by user
ind_pot	matrix of induction motor constants set internally
ind_int	index of internal induction motor buses
motbus	buses to which induction motors are connected
vdp	V'd direct axis transient voltage (state)
vqp	V'q quadrature axis transient voltage (state)
slip	fractional slip (state)
dvdp	dV'_d/dt
dvqp	dV'_q/dt
dslip	ds/dt

Table 2 ind_pot variable definitions

Index Number	Variable
1	Scaled MVA base
2	Motor Base KV
3	$X_s = x_s + X_m$
4	$X_r = x_r + X_m$
5	$X_s' = x_s + \frac{x_r X_m}{X_r}$
6	$X_s - X_s'$
7	$1/T_r = \omega_0 r_r / X_r$

With deep bar and double cage motors the ind_pot variables 3 to 7 vary with the motor slip, and are updated automatically during simulations. When leakage inductance saturation is specified, these variables change when the stator current exceeds the saturation current level.

Data Format:

The induction motor data is contained in the $\mathbf{i^{th}}$ row of the matrix variable **ind_con**. The data format for **mac_ind** is shown in Table 2.

Table 3 ind_con data format

column	variable	unit
1	motor number	
2	bus number	
3	motor base MVA	MVA
4	stator resistance r_s	pu
5	stator leakage reactance x_s	pu
6	magnetizing reactance X_m	pu
7	rotor resistance r_r	pu
8	rotor leakage reactance x_r	pu
9	inertia constant H	Sec
10	second cage resistance r_2	pu
11	intercage reactance x_2	pu
12	deep bar ratio	pu
13	leakage saturation current	pu
15	fraction of active bus load	

If the fraction of active bus load is set to zero, the induction motor will be initialized as though disconnected from the network. The motor will connect as soon as a simulation is started.

The motor load is a function of speed as calculated in the m-file **ind_ldto**. Data associated with the load torque is specified using the matrix **mld_con**. Each row of **mld_con** represents the motors load/speed characteristic. Its form is shown in Table 3.

Table 4 ind_ldto data format

column	variable	unit
1	motor number	
2	motor bus number	
3	stiction load coefficient - f_1	pu on motor base
4	stiction load index- i_1	
5	main load coefficient - f_2	pu on motor base
6	main load index - i_2	

The form of the motor load is as follows:

For a running motor the load torque is

$$t_l = \frac{t_{init}}{t_0}(f_1 s^{i_1} + f_2(1-s)^{i_2})$$

where

$$t_0 = f_1 s_0^{i_1} + f_2(1-s_0)^{i_2}$$ and s_0 is the initial slip

For a starting motor the load torque is

$$t_l = f_1 s^{i_1} + f_2(1-s)^{i_2}$$

Typical values are f_1=.1; i_1 = 1; f_2=.7; i_2=2

Example:

The induction motor data in the 3 machine, 9 bus system are

ind_con =

1	7	25	0.001	0.01	3.	0.009	0.01	0.7	0	0	0	0	0	0.15
1	9	25	0.001	0.01	3.	0.009	0.01	0.7	0	0	0	0	0	0.15

Algorithm:

Initialization (flag = 0) uses the solved load flow bus voltages and angles to compute the slip required for the motor to draw the specified power. The slip is calculated using a Newton Raphson iteration. Failure to converge within 30 iterations causes an error message to be generated. Once the initial slip is known, the motor's reactive power is calculated. The motor's real and reactive powers are then subtracted from the corresponding bus loads.

The dynamic model is that formulated by Brereton, Lewis and Young[1]. In this model the three states are the d and q voltages behind transient reactance and the motor slip.

If a double cage rotor is specified (non-zero values in columns 10 and 11 of ind_con), the effective rotor resistance and reactance (r_{re} and x_{re}) will vary with slip.

$$z = ix_r + (r_r./s).*(r_2./s + i*x_2)./((r_r+r_2)./s + ix_2);$$
$$r_{re} = s.*real(z);$$
$$x_{re} = imag(z);$$

If a deep bar rotor is specified (a non-zero value in column 12 of ind_con), the effective rotor resistance and reactance vary with slip

$$b = Bsqrt(abs(s));$$
$$r_o = r_r/2;$$
$$a = (1+i)b;$$
$$z = r_o a[(exp(a)+1)/(exp(a)-1)];$$
$$r_e = real(z); x_e = imag(z)/s;$$

where B is the deep bar factor.

If leakage inductance saturation is specified, the stator and rotor leakage reactances vary according to the describing function for saturation. For the stator current greater than the saturation current

$$\theta = atan2(i_{sat}, \sqrt{(i_s^2 - i_{sat}^2)})$$
$$g = (2/\pi)*(\theta + sin(2\theta)/2)$$
$$x_{sn} = x_s g/2$$
$$x_{rn} = x_r g/2$$

This algorithm is implemented in the M-file **mac_ind.m** in the POWER SYSTEM TOOLBOX.

See also: loadflow, mac_tra, mac_sub, red_ybus.

Reference:

1. D.S. Brereton, D.G. Lewis and C.C. Young, "Representation of Induction Motor Loads during Power System Stability Studies", AIEE Trans., vol. 76, Part III, August 1957, pp 451-460.

mac_sub

Purpose:

Models a synchronous machine with the voltage behind subtransient reactance model

Synopsis:

f = mac_sub(i,k,bus,flag)

Description:

mac_sub(i,k,bus,flag) contains the voltage behind the subtransient reactance model equations [1] for the initialization, network interface and dynamics computation of the **i**th synchronous machine (see block diagram in Figure 1).

The m.file **pst_var.m** containing all the global variables required for **mac_sub** should be loaded in the program calling **mac_sub**.

The generators are numbered internally according to their order in **mac_con**. This information is contained in the array **mac_int** and is set up automatically by the Y matrix reduction function **red_ybus**.

Inputs:

i the number of the generator
if **i** = 0 all subtransient model generator computations are performed using MATLAB vector methods. **This is the preferred mode.**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and their rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus the solved bus specification matrix

flag indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1.
- The network interface calculation is performed when **flag** = 1
- The rates of change of the subtransient generator states are calculated when **flag** = 2, using the generator terminal voltage and the external system values at the time specified by **k**

Output:

f a dummy variable

Global Variables

System variables

basmva		system base MVA
basrad		2 π* system frequency
syn_ref		synchronous reference
mach_ref		reference machine
sys_freq		system frequency in pu
bus_v	V	bus voltage magnitude in pu
bus_ang	θ	bus voltage angle in rad
psi_re	ψ_{re}	real and imaginary components of voltage
psi_im	ψ_{im}	source on system reference frame
cur_re	i_{re}	real and imaginary components of bus
cur_im	i_{im}	current on system reference frame
bus_int		array to store internal bus ordering

Synchronous Generator Variables

mac_ang	δ	machine angle in rad/sec
mac_spd	ω	machine speed in pu
eqprime	E_q'	pu on machine base
edprime	E_d'	pu on machine base
psikd	ψ_{kd}	pu on machine base
psikq	ψ_{kq}	pu on machine base
curd	i_d	d-axis current on system base
curq	i_q	q-axis current on system base
curdg	i_{dg}	d-axis current on machine base
curqg	i_{qg}	q-axis current on machine base
fldcur	I_{fd}	field current on machine base
psidpp	ψ_d''	pu on machine base
psiqpp	ψ_q''	pu on machine base
vex	V_{ex}	field voltage on machine base
eterm	E_T	machine terminal voltage in pu
theta	θ	terminal voltage angle in rad
ed	E_d	d-axis terminal voltage in pu
eq	E_q	q-axis terminal voltage in pu
pmech	P_m	mechanical input power in pu
pelect	P_e	electrical active output power in pu
qelect	Q_e	electrical reactive output power in pu
dmac_ang	$d\delta/dt$	
dmac_spd	$d\omega/dt$	
deqprime	dE_q'/dt	
dedprime	dE_d'/dt	
dpsikd	$d\psi_{kd}/dt$	
dpsikq	$d\psi_{kq}/dt$	
mac_int		array to store internal machine ordering
mac_pot		internally set matrix of machine constants

mac_con	matrix of generator parameters set by user
n_mac	number of generators
n_sub	number of subtransient generator models
mac_sub_idx	index of subtransient generator models

Data Format

The data format for **mac_sub** is given in Table 1.

A constraint on using **mac_sub** is that $x_q'' = x_d''$. This is because of the way in which the subtransient reactance is used in the network interface. **mac_sub** checks that the direct and quadrature subtransient reactances are equal, if they are not it makes them equal.

The definitions of the saturation factors are given in saturation curve diagram (Figure 2). It is assumed that there is no saturation for field current less than 0.8 pu. Setting the saturation factors to zero eliminates the saturation effect.

Table 1 Data Format for mac_sub

column	variable	unit
1	machine number	
2	bus number	
3	base MVA	MVA
4	leakage reactance x_l	pu
5	resistance r_a	pu
6	d-axis synchronous reactance x_d	pu
7	d-axis transient reactance x_d'	pu
8	d-axis subtransient reactance x_d''	pu
9	d-axis open circuit time constant T_{do}'	sec
10	d-axis open circuit subtransient time constant T_{do}''	sec
11	q-axis synchronous reactance x_q	pu
12	q-axis transient reactance x_q'	pu
13	q-axis subtransient reactance x_q''	pu
14	q-axis open circuit time constant T_{qo}'	sec
15	q-axis open circuit subtransient time constant T_{qo}''	sec
16	Inertia constant H	sec
17	local damping coefficient d_o	pu
18	system damping coefficient d_1	pu
19	bus number	
20	saturation factor $S(1.0)$	
21	saturation factor $S(1.2)$	
22	active power fraction	
23	reactive power fraction	

Example:

The machine data of a single machine infinite bus system are

```
mac_con = [
1 1 991     0.15 0  2.0    0.245 0.2  5.0  0.031 ...
                    1.91   0.42  0.2  0.66 0.061 ...
                    2.8756 0.0   0    1    0       0;
2 2 100000 0.00  0  0.     0.01  0    0    0     ...
                    0      0     0    0    0     ...
                    3.0    2.0   0    2    0       0];
```

The first generator data is that for a subtransient model, the second data is that for an electromechanical generator model used to represent the infinite bus. In small signal stability simulations, the second generator should be declared as an infinite bus (see **mac_ib**).

Algorithm:

Based on the machine vector diagram

- the initialization uses the solved load flow bus voltages and angles to compute the internal voltage and the rotor angle.
- In the network interface computation, the voltage behind the subtransient reactance on the system reference frame is generated.
- In the dynamics calculation, the power imbalance and the speed deviation are used to compute the time derivative of the state variables.

This algorithm is implemented in the M-file **mac_sub** in the POWER SYSTEM TOOLBOX.

See also: loadflow, pst_var, mac_em, mac_tra.

Reference:

1. R. P. Schulz, "Synchronous Machine Modeling," presented at the Symposium ``Adequacy and Philosophy of Modeling: System Dynamic Performance," San Francisco, July 9-14, 1972.

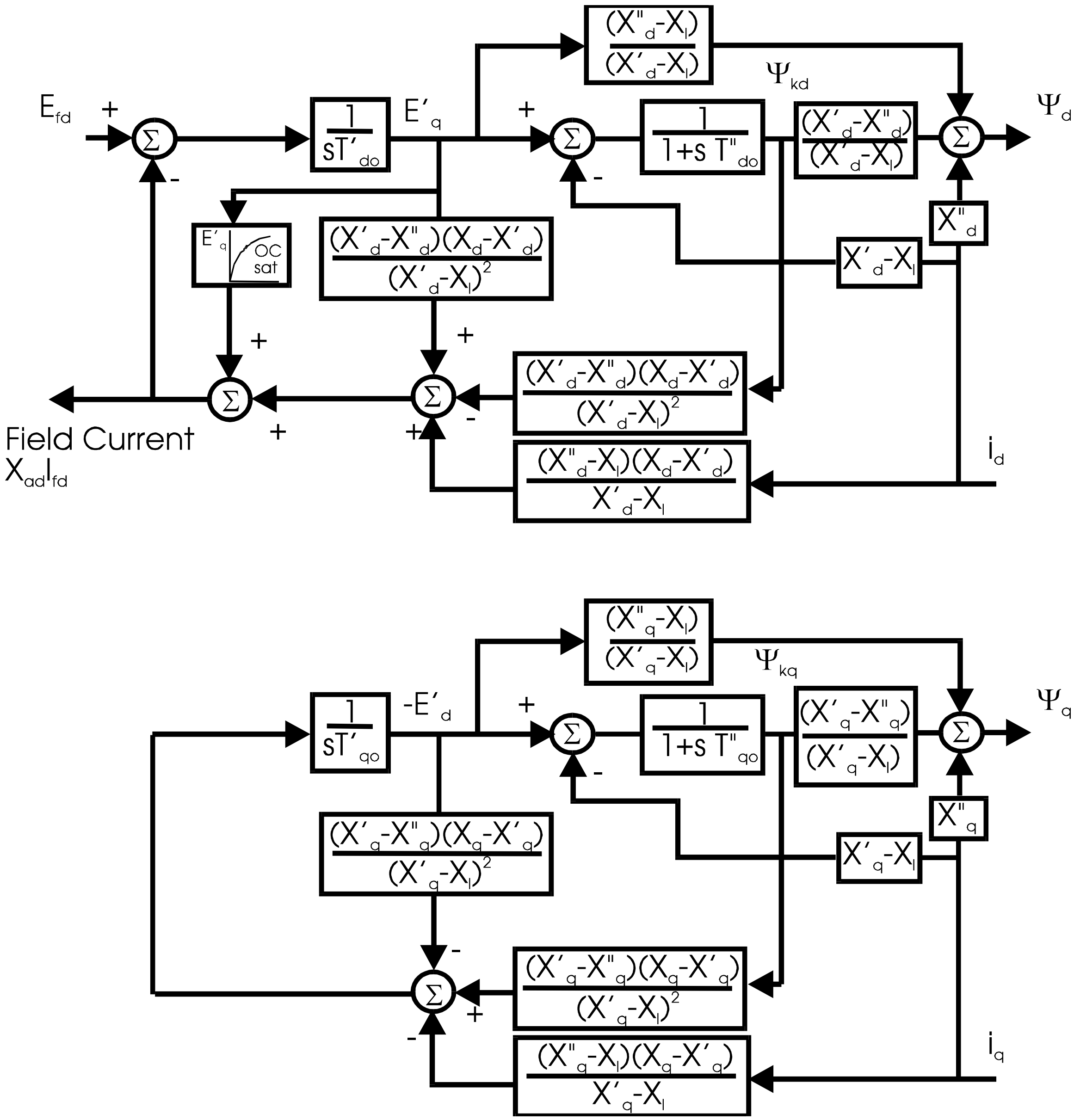


Figure 8 Block Diagram Of Direct and Quadrature Axes of Subtransient Generator Model

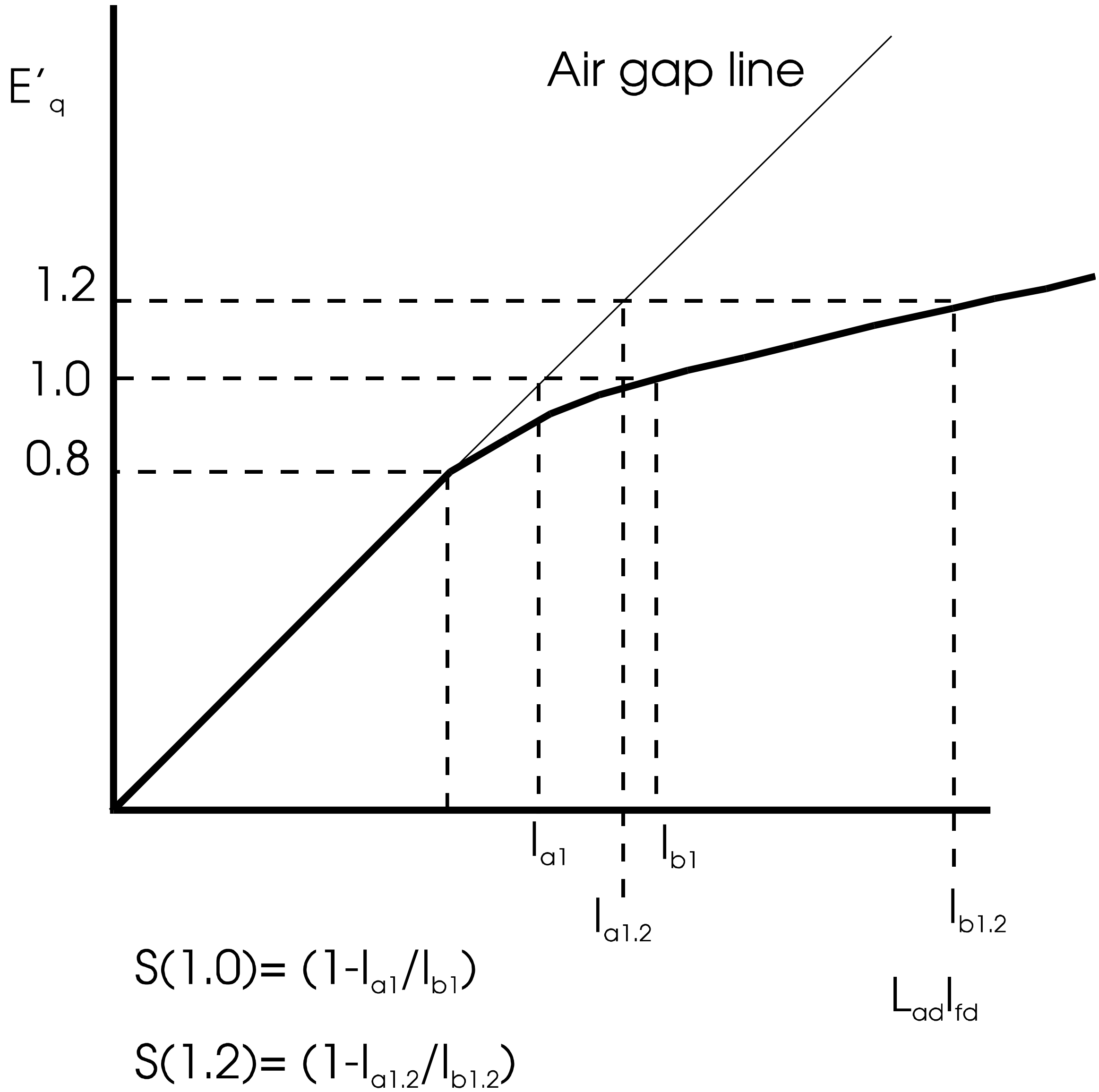


Figure 9 Synchronous Generator Field Saturation Characteristic

mac_tra

Purpose:

Models a synchronous machine with the voltage behind transient reactance model

Synopsis:

f = mac_tra(i,k,bus,flag)

Description:

mac_tra(i,k,bus,flag) contains the voltage behind the transient reactance model equations for the initialization, network interface and dynamics computation of the **i**th synchronous machine (see block diagram in Figure 1).

The m.file **pst_var.m** containing all the global variables required for **mac_tra** should be loaded in the program calling **mac_tra**.

The machines are numbered internally according to the order of the machines in **mac_con**. This information is contained in the array **mac_int** and is set up
automatically by the Y matrix reduction function **red_ybus**.

Inputs:

i the number of the generator
if **i** = 0 all transient model generator computations are performed using MATLAB vector methods. **This is the preferred mode.**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and their rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus the solved bus specification matrix

flag indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1.
- The network interface calculation is performed when **flag** = 1
- The rates of change of the transient generator states are calculated when **flag** = 2, using the generator terminal voltage and the external system values at the time specified by **k**

Output:

f a dummy variable

Global Variables

System variables

basmva		system base MVA
basrad		2 π* system frequency
syn_ref		synchronous reference
mach_ref		reference machine
sys_freq		system frequency in pu
bus_v	V	bus voltage magnitude in pu
bus_ang	θ	bus voltage angle in rad
psi_re	ψ_{re}	real and imaginary components of voltage
psi_im	ψ_{im}	source on system reference frame
cur_re	i_{re}	real and imaginary components of bus
cur_im	i_{im}	current on system reference frame
bus_int		array to store internal bus ordering

Synchronous Generator Variables

mac_ang	δ	machine angle in rad/sec
mac_spd	ω	machine speed in pu
eqprime	E_q'	pu on machine base
edprime	E_d'	pu on machine base
psikd	ψ_{kd}	pu on machine base
psikq	ψ_{kq}	pu on machine base
curd	i_d	d-axis current on system base
curq	i_q	q-axis current on system base
curdg	i_{dg}	d-axis current on machine base
curqg	i_{qg}	q-axis current on machine base
fldcur	I_{fd}	field current on machine base
psidpp	ψ_d''	pu on machine base
psiqpp	ψ_q''	pu on machine base
vex	V_{ex}	field voltage on machine base
eterm	E_T	machine terminal voltage in pu
theta	θ	terminal voltage angle in rad
ed	E_d	d-axis terminal voltage in pu
eq	E_q	q-axis terminal voltage in pu
pmech	P_m	mechanical input power in pu
pelect	P_e	electrical active output power in pu
qelect	Q_e	electrical reactive output power in pu
dmac_ang	$d\delta/dt$	
dmac_spd	$d\omega/dt$	
deqprime	dE_q'/dt	
dedprime	dE_d'/dt	
mac_int		array to store internal machine ordering
mac_pot		internally set matrix of machine constants
mac_con		matrix of generator parameters set by user

n_mac	number of generators
n_tra	number of subtransient generator models
mac_tra_idx	index of subtransient generator models

Data Format

The data format for **mac_tra** is given in Table 1.

The definitions of the saturation factors are given in saturation curve diagram (Figure 2). It is assumed that there is no saturation for field current less than 0.8 pu. Setting the saturation factors to zero eliminates the saturation effect.

Table 1. Data format for mac_tra

column	variable	unit
1	machine number	
2	bus number	
3	base MVA	MVA
5	resistance r_a	pu
6	d-axis synchronous reactance x_d	pu
7	d-axis transient reactance x_d'	pu
9	d-axis open circuit time constant T_{do}'	sec
11	q-axis synchronous reactance x_q	pu
12	q-axis transient reactance x_q'	pu
14	q-axis open circuit time constant T_{qo}'	sec
16	Inertia Constant H	sec
17	local damping coefficient d_o	pu
18	system damping coefficient d_1	pu
19	bus number	
20	saturation factor $S(1.0)$	
21	saturation factor $S(1.2)$	
22	active power fraction	
23	reactive power fraction	

Algorithm:

Based on the machine vector diagram
- the initialization uses the solved load flow bus voltages and angles to compute the internal voltage and the rotor angle.
- In the network interface computation, the voltage behind the transient reactance on the system reference frame is generated.

In the dynamics calculation, the power imbalance and the speed deviation are used to compute the time derivatives of the state variables
This algorithm is implemented in the M-file **mac_tra.m** in the POWER SYSTEM TOOLBOX.
See also: loadflow, pst_var, mac_em, mac_sub.

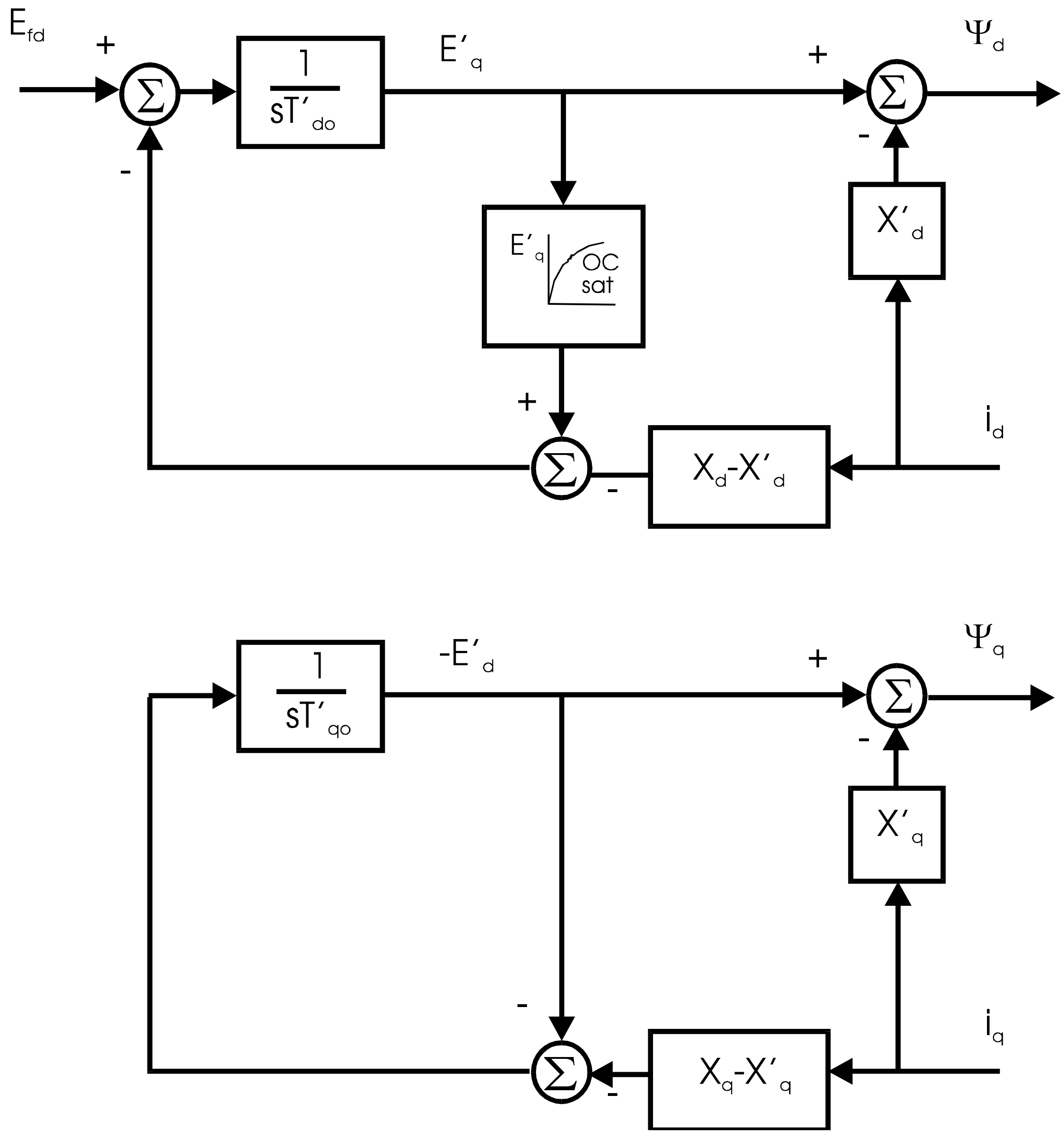


Figure 10 Block Diagram Transient Generator Model

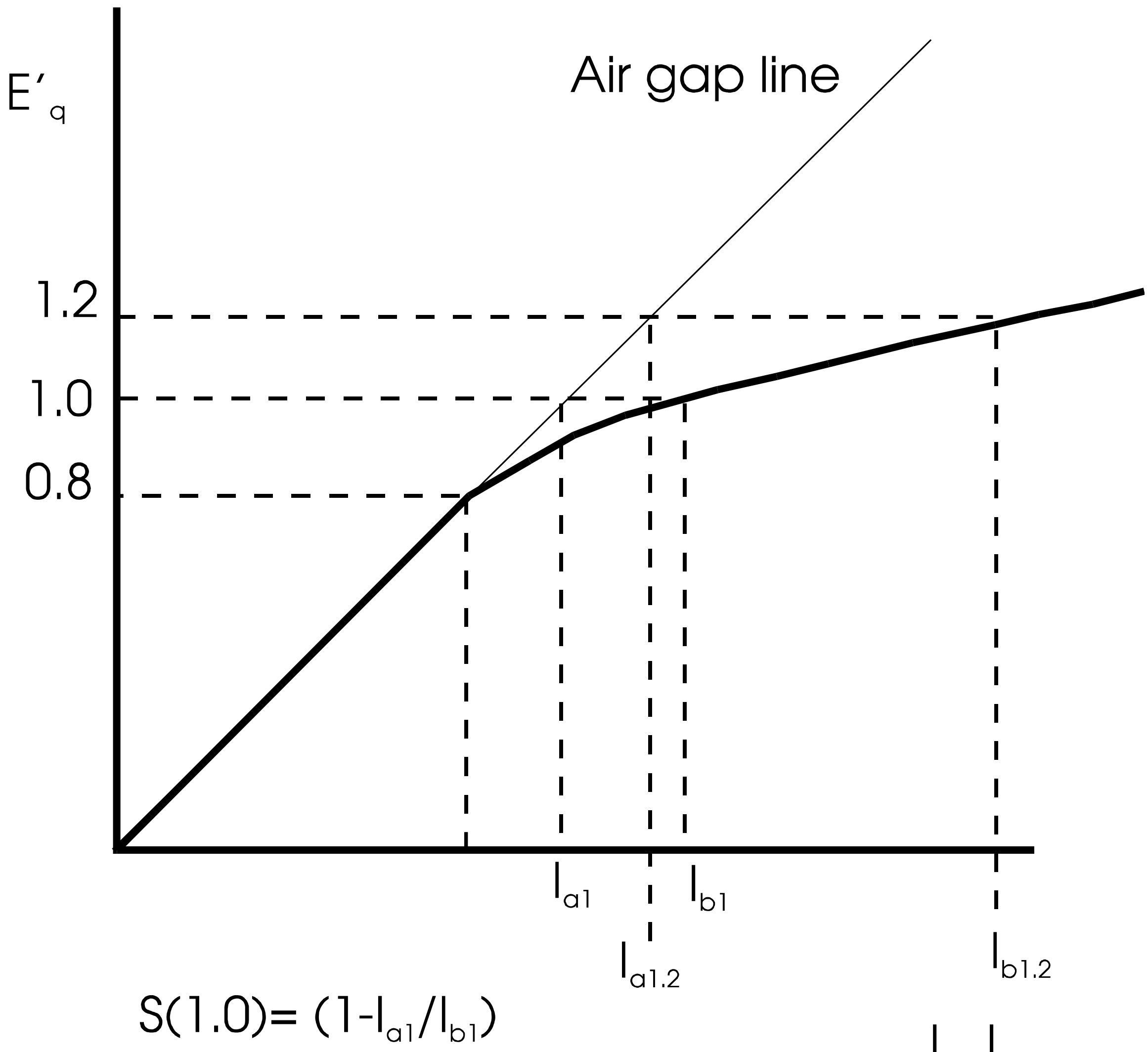


Figure 11 Field Saturation Characteristic

mdc_sig

Purpose:

Forms the dc controls modulation signal.

Synopsis:

f = mdc_sig(t, k)

Description:

f = mdc_sig forms the load modulation signal as a function of time. The modulation variable **dc_sig** is passed as a global variable.

The m.file **pst_var.m** containing all the global variables should be loaded in the program calling **mdc_sig**.

Inputs:

t the time in seconds corresponding to **k**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

Output:

f a dummy variable

Global Variable

dc_sig V_{sup} supplementary load modulation signal

n_conv number of HVDC converters

See also: **dc_cont**

Example

The following version of **mdc_sig** causes a step change in the first rectifier pole control reference after a time of 0.1 s.

```
function f = mdc_sig(t,k)
% Syntax: f = mdc_sig(t,k)
% defines modulation signal for svc control
global dc_sig n_conv r_idx i_idx
f=0; %dummy variable
if t<=0.1
   dc_sig(:,k) = zeros(n_conv,1);
else
   dc_sig(:,k) = zeros(n_conv,1);
   dc_sig(r_idx(1),k) = 0.1;
end
return
```

mexc_sig

Purpose:

Forms the exciter modulation signal.

Synopsis:

f = mexc_sig(t, k)

Description:

f = mexc_sig forms the exciter modulation signal as a function of time. The modulation variable **exc_sig** is passed as a global variable.

The m.file **pst_var.m** containing all the global variables should be loaded in the program calling **mexc_sig**.

Inputs:

t	the time in seconds corresponding to **k**
k	the integer time step in a simulation In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

Output:

f	a dummy variable

Global Variable

exc_sig	V_{sup}	supplementary load modulation signal
n_exc		number of exciters

See also: **exc_dc12, exc_st3, smpexc**

Example

The following version of **mexc_sig** causes a step change of 0.01 in Vref at exciter number 1 after a time of 0.1 s.

```
function f = mexc_sig(t,k)
% Syntax: f = mexc_sig(t,k)
% defines modulation signal for exciter control
global exc_sig n_exc
f=0; %dummy variable
exc_sig(:,k) = zeros(n_exc,1);
if t>=0.1
   exc_sig(1,k) = 0.01;
end
return
```

ml_sig

Purpose:

Forms the load modulation signal.

Synopsis:

f = ml_sig(t, k)

Description:

f = ml_sig forms the load modulation signal as a function of time. The modulation variable **lmod_sig** is passed as a global variable.

The m.file **pst_var.m** containing all the global variables should be loaded in the program calling **ml_sig**.

Inputs:

t	the time in seconds corresponding to **k**
k	the integer time step in a simulation In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

Output:

f	a dummy variable

Global Variable

lmod_sig	V_{sup}	supplementary load modulation signal
n_lmod		number of load modulation controls

See also: **lmod**

Example

The following version of ml_sig causes a step change in load after a time of 0.1 s.

```
function f = ml_sig(t,k)
% Syntax: f = ml_sig(t,k)
% defines modulation signal for lmod control
global lmod_sig n_lmod
f=0; %dummy variable
if t<=0.1
   lmod_sig(:,k) = zeros(n_lmod,1);
else
   lmod_sig(:,k) = 0.1*ones(n_lmod,1);
end
return
```

msvc_sig

Purpose:

Forms the svc modulation signal.

Synopsis:

f = msvc_sig(t, k)

Description:

f = msvc_sig forms the load modulation signal as a function of time. The modulation variable **svc_sig** is passed as a global variable.

The m.file **pst_var.m** containing all the global variables should be loaded in the program calling **msvc_sig**.

Inputs:

t the time in seconds corresponding to **k**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

Output:

f a dummy variable

Global Variable

svc_sig V_{sup} supplementary load modulation signal

n_svc number of svc controls

See also: **svc**

Example

The following version of **msvc_sig** causes a step change in all the svc reference voltages after a time of 0.1 s.

```
function f = msvc_sig(t,k)
% Syntax: f = msvc_sig(t,k)
% defines modulation signal for svc control
global svc_sig n_svc
f=0; %dummy variable
if t<=0.1
   svc_sig(:,k) = zeros(n_svc,1);
else
   svc_sig(:,k) = 0.1*ones(n_svc,1);
end
return
```

mtg_sig

Purpose:

Forms the turbine governor modulation signal.

Synopsis:

f = mtg_sig(t, k)

Description:

f = mtg_sig forms the turbine governor modulation signal as a function of time. The modulation variable **tg_sig** is passed as a global variable.

The m.file **pst_var.m** containing all the global variables should be loaded in the program calling **mtg_sig**.

Inputs:

t	the time in seconds corresponding to **k**
k	the integer time step in a simulation In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

Output:

f	a dummy variable

Global Variable

tg_sig	V_{sup}	supplementary power order modulation signal
n_tg		number of turbine governor controls

See also: **tg**

Example

The following version of mtg_sig causes a step change of 0.01 in governor power demand after a time of 0.1 s.

```
function f = mtg_sig(t,k)
% Syntax: f = mtg_sig(t,k)
% defines modulation signal for  turbine governor power demand control at all
% governors
global tg_sig n_tg
f=0; %dummy variable
if t<=0.1
   tg_sig(:,k) = zeros(n_tg,1);
else
   tg_sig(:,k) = 0.01*ones(n_tg,1);
end
return
```

nc_load

Purpose:

Solves the complex voltages at non-conforming load buses

Synopsis:

[V] = nc_load(bus,flag,Y22,Y21,psi,Vo,tol)

[V] = nc_load(bus,flag,Y22,Y21,psi,Vo,tol,k)

Description:

[V] = nc_load(bus,flag,Y22,Y21,psi,Vo,tol) computes the complex voltage **V** at the non-conforming load buses the SVC buses and the HVDC HT buses using a Newton-Raphson algorithm.

[V] = nc_load(bus,flag,Y22,Y21,psi,Vo,tol,k) is used in the simulation process at each network interface calculation.

The m.file **pst_var.m** containing all the global variables required for **nc_load** should be loaded in the program calling **nc_load**.

Inputs:

bus	solved loadflow bus data
flag	solution mode control
	0 - initialization
	1 - network interface computation
	2 - dynamic calculation not needed in this model
Y22	reduced Y matrix of non-conforming loads (output from red_ybus)
Y21	reduced Y matrix connecting non conforming load current to machine internal voltages
psi	machine internal voltage, not used in initialization
V_o	initial non conforming load bus voltage vector, not used in initialization
tol	tolerance for Newton's algorithm convergence, not used in initialization
k	integer time step (only for svc/facts models), not used in initialization

Outputs:

V_nc solved non-conforming load bus voltage vector

Global Variables:

load_con : non-conforming bus specification matrix
load_pot : non-conforming bus constants
bus_int : internal bus number vector
svc_con : svc specification matrix
svc_idx: svc index vector
n_svc : number of svcs
svc_pot : svc constants
B_cv: svc state
i_dci : inverter dc current
i_dcr : rectifier dc current
dcc_pot : dc controls constants
alpha: rectifier firing angle
gamma: inverter extinction angle
basmva : base MVA
r_idx: rectifier converter index
i_idx: inverter converter index
n_conv : number of HVDC converters
n_dcl : number of HVDC lines
ldc_idx: HVDC line index

Data Format

The non-conforming load data is contained in the $\mathbf{i^{th}}$ row of the matrix variable **load_con**. The data format for **load_con** is given in Table 1.

Table 1. Data format for load_con

column	variable	unit
1	bus number	
2	fraction of constant active power load	
3	fraction of constant reactive power load	
4	fraction of constant active current load	
5	fraction of constant reactive current load	

Note: SVCs obtain their initial values from the generator reactive power specified in bus. If an SVC bus has loads specified also, these may be defined as non conforming in the same way as any load bus. If there is no load, then the SVC bus must still be declared as non conforming, but with zero entries for the load fractions. HVDC buses are specified in the load flow as the Low Tension buses, these buses cannot have loads, other than the HVDC loads.

Algorithm:

The current balance equation at the non-conforming load buses is given by

$$Y21\psi + Y22V = (Icc(V) + Icp(V))$$

where *Icc* is the current injection due to the constant current components and *Icp* is the current injection due to the constant power components. These injections are functions of the bus voltage. The constant impedance components are included in **Y22** (which is computed in the function **red_ybus**). Sensitivities of these injections with respect to the voltage is used to formulate a Newton's algorithm to solve this nonlinear equation. The initial guess **Vo** is typically the bus solution at the previous time step.

See **s_simu.m** and **svm_mgen.m** for examples of use.

This algorithm is implemented in the M-file **nc_load.m** in the POWER SYSTEM TOOLBOX.

See also: pst_var, red_ybus, svc, s_simu, svm_mgen, i_simu

pss

Purpose:

Models power system stabilizers

Synopsis:

f = pss(i,k,bus,flag)

Description:

pss(i,k,bus,flag) contains the equations of a power system stabilizer (PSS) model shown in Figure 1 for the initialization, machine interface and dynamics computation of the **i**$^{\mathbf{th}}$ excitation system.

The m.file **pst_var.m** containing all the global variables required for **pss** should be loaded in the program calling **pss**.

Inputs:

i the number of the generator which the PSS is controlling
if **i** = 0 all PSS computations are performed using MATLAB vector methods. **This is the preferred mode.**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus the solved bus specification matrix

flag indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1. For proper initialization, the corresponding generators must be initialized first.
- The network interface calculation is performed when **flag** = 1, and the field voltage of the synchronous machine is set to the exciter output voltage.
- The rates of change of the exciter states are calculated when **flag** = 2, using the generator terminal voltage and the external system values at the time specified by **k**

Output:

f a dummy variable

Global Variables

System variables

basmva	system base MVA

Synchronous Generator Variables

mac_spd	ω	machine speed in pu

pelect	P_e	electrical active output power in pu on system base
mac_int		array to store internal machine ordering
mac_pot		internally set matrix of machine constants
mac_con		matrix of generator parameters set by user

Excitation System Variable

exc_sig	V_{sup}	supplementary input signal to exciter ref input

PSS variables

pss1	washout state variable
pss2	first lead-lag compensator state variable
pss3	second lead-lag compensator state variable
dpss1	
dpss2	
dpss3	
pss_con	matrix of pss parameters specified by user
pss_pot	internally computed matrix of pss constants
n_pss	number of pss
pss_idx	index of pss
pss_T	**pss_con(pss_idx,4)**
pss_T2	**pss_con(pss_idx,6)**
pss_T4	**pss_con(pss_idx,8)**
pss_T4_idx	index of nonzero T4 for pss
pss_noT4	index of zero T4 for pss
pss_sp_idx	index of pss with speed input
pss_p_idx	index of pss with power input

Data Format

The pss data is contained in the $\mathbf{i}^{\mathbf{th}}$ row of the matrix variable **pss_con**. The data format for **pss** is shown in Table 1.

Table 1. Data format for pss

column	data	unit
1	type 1 speed input 2 power input	
2	machine number	
3	gain K	
4	washout time constant T	sec
5	lead time constant T_1	sec
6	lag time constant T_2	sec
7	lead time constant T_3	sec
8	lag time constant T_4	sec
9	maximum output limit	pu
10	minimum output limit	pu

A constraint on using **pss** is that $T_1 \neq 0$ and $T_2 \neq 0$. The output of the power system stabilizer is limited by an upper and a lower limit.

Note: The PSS gain K is equal to the normally defined Kstab multiplied by T, the washout time constant.

Algorithm:

Based on the pss block diagram

- on initialization the washout state variable is set to
  - the generator speed for type = 1
  - the electrical power on the generator base if type = 2

  the remaining states are set to zero. The PSS output is also zero.
- In the network interface computation, the PSS output signals exc_sig are set.
- In the dynamics calculation, the input machine speed or electrical power is used to calculate the rates of change of the PSS states.

This algorithm is implemented in the M-file **pss** in the POWER SYSTEM TOOLBOX.

See also: pst_var, smpexc, exc_dc12, exc_st3.

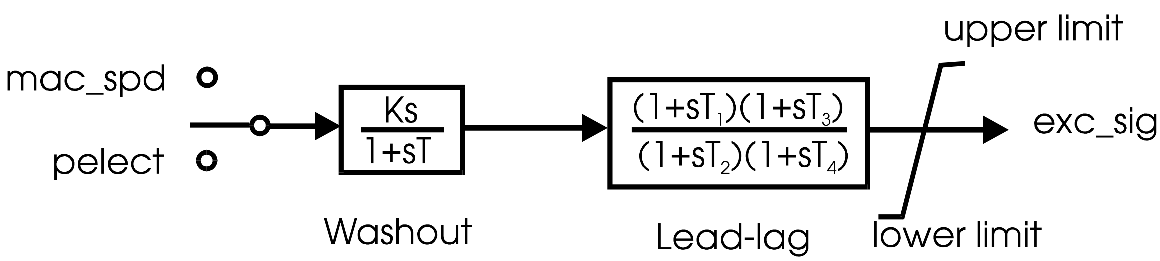


Figure 12 Power System Stabilizer Model Block Diagram

pss_des

Purpose:

Allows trial and error determination of PSS parameters to fit an ideal frequency response

Syntax:

[tw,t1,t2,t3,t4] = pss_des(a,b,c,d)

Global variables

There are no global variable in this file

Description:

This function allows the user to select, on a cut-and-try basis, power system stabilizer parameters which fit as closely as desired the ideal phase lead between V_{ref} and the generator electrical torque necessary to produce a damping torque over the matched frequency range.

Inputs:

a	the state matrix of the system for which the PSS is to be designed
b	the input matrix associated for the exciter reference input
c	the output matrix associated with the generator mechanical torque
d	the feed forward matrix between the voltage reference and the generator mechanical torque. Normally zero

The inputs are normally obtained by running **svm_mgen**

Outputs:

tw	the washout time constant (s)
t1	the first lead time constant (s)
t2	the first lag time constant (s)
t3	the second lead time constant (s)
t4	the second lag time constant (s)

Algorithm:

The user is asked to provide a set of PSS parameters - default settings are provided. The ideal stabilizer frequency response is calculated from the supplied state matrices using **statef**. This is plotted together with the stabilizer frequency response.

The user can then perform an additional check with new parameters in order to a close fit to the ideal frequency response characteristic.

This algorithm is implemented in the M-file **pss_des** in the POWER SYSTEM TOOLBOX.

pst_var

Purpose:

Declare global variables for functions in POWER SYSTEM TOOLBOX

Synopsis:

pst_var

Description:

pst_var declares all the global variables required for the functions in POWER SYSTEM TOOLBOX. All these variables can be displayed in matrix form or graphically by MATLAB. **pst_var** is inserted at the top of script files (m.files) for simulation and building state matrices. To start a new simulation, the memory should be cleared by typing clear and clear global.

Global Variables:

System variables

basmva		system base MVA
basrad		2 π* system frequency
syn_ref		synchronous reference
mach_ref		reference machine
sys_freq		system frequency in Hz
bus_v	V	bus voltage magnitude in pu
bus_ang	θ	bus voltage angle in rad
psi_re	ψ_{re}	real and imaginary components of voltage
psi_im	ψ_{im}	source on system reference frame
cur_re	I_{re}	real and imaginary components of bus
cur_im	I_{im}	current on system reference frame
bus_int		array to store internal bus ordering

Synchronous Generator Variables

mac_ang	δ	machine angle in rad/sec
mac_spd	ω	machine speed in pu
eqprime	E_q'	pu on machine base
edprime	E_d'	pu on machine base
psikd	ψ_{kd}	pu on machine base
psikq	ψ_{kq}	pu on machine base
curd	i_d	d-axis current on system base
curq	i_q	q-axis current on system base
curdg	i_{dg}	d-axis current on machine base

curqg	i_{qg}	q-axis current on machine base
fldcur	I_{fd}	field current on machine base
psidpp	ψ_d''	pu on machine base
psiqpp	ψ_q''	pu on machine base
vex	V_{ex}	field voltage on machine base
eterm	E_T	machine terminal voltage in pu
theta	θ	terminal voltage angle in rad
ed	E_d	d-axis terminal voltage in pu
eq	E_q	q-axis terminal voltage in pu
pmech	P_m	mechanical input power in pu
pelect	P_e	electrical active output power in pu
qelect	Q_e	electrical reactive output power in pu
dmac_ang	$d\delta/dt$	
dmac_spd	$d\omega dt$	
deqprime	dE_q'/dt	
dedprime	dE_d'/dt	
dpsikd	$d\psi_{kd}/dt$	
dpsikq	$d\psi_{kq}/dt$	
mac_int		array to store internal machine ordering
mac_pot		internally set matrix of machine constants
mac_con		matrix of generator parameters set by user
ibus_con		vector specifying infinite buses set by user
n_mac		number of generators
n_em		number of em (classical) generator models
n_tra		number of transient generator models
n_sub		number of subtransient generator models
n_ib		number of infinite buses
mac_em_idx		index of em generator models, i.e. mac_con(mac_em_idx,:) picks out the em data
mac_tra_idx		index of transient generator models
mac_sub_idx		index of subtransient generator models
mac_ib_idx		index of infinite buses
not_ib_idx		index of generators which are not modelled as infinite buses
mac_ib_em		index of em generatoirs modelled as infinite buses
mac_ib_tra		index of transient generators modelled as infinite buses
mac_ib_sub		index of subtransient generators modelled as infinite buses
n_ib_em		number of em generators modelled as infinite buses
n_ib_tra		number of transient generators modelled as infinite buses
n_ib_sub		number of subtransient generators modelled as infinite buses

Excitation System Variables

Efd	E_{fd}	exciter output voltage, equal to generator field voltage pu
V_R	V_R	regulator output voltage in pu
V_A	V_A	regulator output voltage in pu
V_As	V_{As}	regulator voltage state variable in pu
R_f	R_f	stabilizing transformer state variable
V_FB	V_{FB}	feedback from stabilizing transformer

V_TR	V_{TR}	voltage transducer output in pu
V_B	V_B	potential circuit voltage output in pu
dEfd	dE_{fd}/dt	
dV_R	dV_R/dt	
dV_As	dV_{As}/dt	
dR_f	dR_f/dt	
dV_TR	dV_{TR}/dt	
exc_sig	V_{sup}	supplementary input signal to exciter ref input
exc_pot		matrix of internally set exciter constants
exc_con		matrix of exciter data set by user
smp_idx		index of simple exciters, i.e., exc_con(smp_idx,:)
n_smp		number of simple exciters
dc_idx		index of dc exciters
n_dc		number of dc exciters
dc2_idx		index of type 2 dc exciters
n_dc2		number of type 2 dc exciters
st3_idx		index of st3 exciters
n_st3		number of st3 exciters
smp_TA		**exc_con(smp_idx,5)**
smp_TA_idx		index of non-zero TA for simple exciter
smp_noTA_idx		index of zero TA for simple exciter
smp_TB		**exc_con(smp_idx,6)**
smp_TB_idx		index of nonzero TB for simple exciters
smp_noTB_idx		index of zero TB for simple exciters
smp_TR		**exc_con(smp_idx,3)**
smp_TR_idx		index of nonzero TR for simple exciter
smp_no_TR_idx		index of zero TR for simple exciter
dc_TA		**exc_con(dc_idx,5)**
dc_TA_idx		index of nonzero TA for dc exciter
dc_noTR_idx		index of zero TA for dc exciter
dc_TB		**exc_con(dc_idx,6)**
dc_TB_idx		index of non-zero TB for dc exciter
dc_noTB_idx;		index of zero TB for dc exciter
dc_TE		**exc_con(dc_idx,11)**
dc_TE_idx		index of nonzero TE for dc exciter
dc_noTE_idx		index of zero TE for dc exciter
dc_TF		**exc_con(dc_idx,17)**
dc_TF_idx		index of TF for dc exciter
dc_TR		**exc_con(dc_idx, 3)**
dc_TR_idx		index of nonzero TR for dc exciter
dc_noTR_idx		index of zero TR for dc exciter
st3_TA		**exc_con(st3_idx,5)**
st3_TA_idx		index of nonzero TA for st3 exciter
st3_noTA_idx		index of zero TA for st3 exciter
st3_TB		**exc_con(st3_idx,6)**
st3_TB_idx		index of nonzero TB for st3 exciter
st3_noTB_idx		index of zero TB for st3 exciter
st3_TR		**exc_con(st3_idx,3)**
st3_TR_idx		index of nonzero TR for st3 exciter
st3_noTR_idx		index of zero TR for st3 exciter

Power System Stabilizer Variables

pss1	washout state variable
pss2	first lead-lag compensator state variable
pss3	second lead-lag compensator state variable
dpss1	
dpss2	
dpss3	
pss_con	matrix of pss parameters specified by user
pss_pot	Internally computed matrix of pss constants
n_pss	number of pss
pss_idx	index of pss
pss_T	**pss_con(pss_idx,4)**
pss_T2	**pss_con(pss_idx,6)**
pss_T4	**pss_con(pss_idx,8)**
pss_T4_idx	index of nonzero T4 for pss
pss_noT4	index of zero T4 for pss
pss_sp_idx	index of pss with speed input: pss_con(pss_sp_idx,1) = 1
pss_p_idx	index of pss with power input: pss_con(pss_p_idx,1) = 2

Turbine-governor Variables

tg1	governor state variable
tg2	servo state variable
tg3	reheater state variable
dtg1	
dtg2	
dtg3	
tg_con	matrix of turbine governor specifications set by user
tg_pot	internally set matrix of turbine governor constants
n_tg	number of turbine governors
tg_idx	index of turbine governors

Induction Motor Variables

tload		motor load torque as a fraction of the initial torque
t_init		initial motor load torque in pu. on motor base
pot		motor active power in pu. on motor base
qmot		motor reactive power in pu. on motor base
vdmot		motor direct axis stator voltage in pu.
vqmot		motor quadrature axis stator voltage in pu.
idmot		motor direct axis stator current in pu.
iqmot		motor quadrature axis voltage im pu.
ind_con		matrix of induction motor parameters set by user
ind_pot		matrix of induction motor constants set internally
ind_int		index of internal induction motor buses
motbus		buses to which induction motors are connected
vdp		V'd direct axis transient voltage (state)
vqp		V'q quadrature axis transient voltage (state)
slip		fractional slip (state)
dvdp	dV'_d/dt	
dvqp	dV'_q/dt	
dslip	ds/dt	

Induction genertaor variables

tmig	mechanical torque from driving turbine
pig	generator active power
qig	generator reactive power
vdig	d axis stator voltage
vqig	q axis stator current
idig	d axis stator current
iqig	q axis stator current
igen_con	matrix of induction generator data
igen_pot	matrix of induction generator constants
igen_int	internal numbers for induction generators
igbus	internal bus numbers for induction generators
n_ig	number of induction generators
vdpig	d axis voltage behind transient impedance
vqpig	d axis voltage behind transient impedance
slig	induction generator slip
dvdpig	rate of change of **vdpig**
dvqpig	rate of change of **vqpig**
dslig	rate of change of **slig**

Non Conforming Load Variables

load_con	matrix of non conforming load parameters set by user
load_pot	matrix of non-conforming load constants set internally

Static VAR Compensator Variables

B_cv	B_{cv}	svc susceptance in pu
dB_cv	dB_{cv}/dt	
svc_sig	V_{sup}	supplementary signal into the reference input
svc_con		matrix of svc parameters supplied by user
svc_pot		internally calculated matrix of svc constants
n_svc		number of svcs
svc_idx		index of svcs included in **load_con**

HVDC System Variables

dcsp_con	HVDC converter specification matrix
dcl_con	HVDC line specification matrix
dcc_con	HVDC pole control specification matrix
r_idx	rectifier converter index
i_idx	inverter converter index
n_dcl	number of HVDC lines
n_conv	number of HVDC converters
ac_bus	index of converter ac buses
rec_ac_bus	index of rectifier ac buses
inv_ac_bus	index of inverter ac buses
inv_ac_line	index of inverter ac lines
rec_ac_line	index of rectifier ac lines
ac_line	index of converter ac lines

dcli_idx	index of HVDC lines
tap	HVDC transformer tap settings
tapr	HVDC rectifier transformer tap settings
tapi	HVDC inverter transformer tap settings
tmax	HVDC tap maximum values
tmin	HVDC tap minimum values
tstep	HVDC tap steps
tmaxr	rectifier maximum tap values
tmaxi	inverter maximum tap values
tminr	rectifier minimum tap values
tmini	inverter minimum tap values
tstepr	rectifier tap step
tstepi	inverter tap step
Vdc	HVDC Voltage kV
i_dc	HVDC current kA
dc_pot	HVDC line constant matrix
alpha	rectifier firing angle
gamma	inverter extinction angle
dc_sig	HVDC external modulation signal
cur_ord	HVDC current order
Vdc_ref	inverter HVDC voltage reference
dcc_pot	HVDC pole controls constant matrix
no_cap_idx	index of HVDC lines having no capacitance
cap_idx	index of HVDC lines having capacitance
no_ind_idx	index of HVDC lines having no inductance
l_no_cap	number of HVDC lines having no capacitance
l_cap	number of HVDC lines having capacitance
i_dcr	rectifier HVDC current kA (state)
i_dci	inverter HVDC current kA (state)
v_dcc	HVDC line capacitance voltage kV (state)
di_dcr	rate of change of rectifier HVDC current
di_dci	rate of change of inverter HVDC current
dv_dcc	rate of change of HVDC line capacitance voltage
v_conr	HVDC rectifier pole control state
dv_conr	rate of change of HVDC rectifier pole control state
v_coni	HVDC inverter pole control state
dv_coni	rate of change of HVDC inverter pole control state

Load Modulation Variables

lmod_st	*lm*	load modulation state
dlmod_st	*dlm/dt*	
lmod_sig	V_{sup}	supplementary signal into the reference input
lmod_con		matrix of lmod parameters supplied by user
lmod_pot		internally calculated matrix of lmod constants
n_lmod		number of load modulation controls
lmod_idx		index of modulation controls included in **load_con**

Reactive Load Modulation Variables

rlmod_st	r*lm*	reactive load modulation state
drlmod_st	*drlm/dt*	
lmod_sig	V_{sup}	supplementary signal into the reference input

rlmod_con	matrix of rlmod parameters supplied by user
rlmod_pot	internally calculated matrix of rlmod constants
n_rlmod	number of reactive load modulation controls
rlmod_idx	index of reactive modulation controls included in **load_con**

red_ybus

Purpose:

Forms the reduced admittance matrix used in simulations.

Synopsis:

[red_Y,rec_V] = red_ybus(bus,line)

[Y11,Y12,Y21,Y22,rec_V1,rec_V2,bus_ord] = red_ybus(bus,line)

Description:

[red_Y,rec_V] = red_ybus(bus,line) uses the bus data in **bus**, the line data in **line** and the machine reactances in **mac_con** and **ind_con** to return the reduced admittance matrix **red_Y** and the voltage reconstruction matrix **rec_V** so that

$$Ig = red\_Y * Vg$$
$$Vb = rec\_V * Vg$$

where *Ig* is a column vector of generator current injection, *Vg* and *Vb* are column vectors of generator bus voltages and load bus voltages, respectively.

[Y11,Y12,Y21,Y22,rec_V1,rec_V2,bus_ord] = red_ybus(bus,line) gives the reduced admittance matrix in partitioned form. This is required when there are non-conforming load buses in the system. The function uses the bus data in **bus**, the line data in **line**, the machine reactance in **mac_con** and **ind_con** ,and the load data in **load_con** to return the reduced admittance matrices **Y11, Y12, Y21, Y22** and the voltage reconstruction matrix **rec_V1**, **rec_V2** so that

$$Ig = Y11 * Vg + Y12 * Vnc$$
$$Vb = rec\_V1 * Vg + rec\_V2 * Vnc$$

where *Vnc* is the column vector of the non-conforming load bus voltages. The matrices **Y21, Y22** and the bus reordering information contained in the column vector **bus_ord** are used in **nc_load**. If the full input is specified on calling when **load_con** is empty, the additional outputs are set to the null matrix [].

The output variables of **red_ybus** are all in full matrix form. The user can convert them to sparse matrix form if necessary.

Inputs:

bus	a solved bus data set
line	a solved line data set

Outputs:

Y11	the reduced admittance matrix connecting the generator current injections to the internal generator and induction motor voltages

Y12	the admittance matrix component which gives the generator and motor currents due to the voltages at non conforming load and SVC buses
Y21	the admittance matrix component which gives the non conforming load and SVC currents in terms of the generator and induction motor internal voltages
Y22	the admittance matrix connecting the non conforming load and SVC currents to the voltages at the non conforming load and SVC buses
rec_V1	The voltage reconstruction matrix which gives the original bus voltages components due to the generator and induction motor internal bus voltages
rec_V2	The voltage reconstruction matrix which gives the original bus voltages components due to the non conforming load and SVC bus voltages
bus_ord	An index vector giving the non conforming loads first followed by the conforming loads

Global Variables:

basmva	system base MVA
bus_int	array to store internal bus ordering
mac_int	array to store internal machine ordering
mac_con	matrix of generator parameters set by user
ind_con	matrix of induction motor parameters set by user
ind_int	index of internal induction motor buses
ind_pot	matrix of induction motor constants set internally
igen_con	matrix of induction generator parameters set by user
igen_int	index of internal induction generator buses
igen_pot	matrix of induction generator constants set internally
load_con	matrix of non conforming load parameters set by user

Example:

Consider the 11 bus, four generator, 2 Area System in **d2a_sub.m**.

The following is a diary record of a call to red_ybus

```
pst_var
d2a_sub
basmva = 100;
[Y_red, V_rec] = red_ybus(bus,line)
Y_red =

   1.2365 - 9.9183i   1.3727 + 6.9137i   0.4129 + 0.5492i   0.6755 + 0.8344i
   1.3727 + 6.9137i   2.5317 -11.7642i   0.6755 + 0.8344i   1.1017 + 1.2650i
   0.4129 + 0.5492i   0.6755 + 0.8344i   1.6591 -10.3017i   2.0111 + 6.2912i
   0.6755 + 0.8344i   1.1017 + 1.2650i   2.0111 + 6.2912i   3.4936 -12.7722i
V_rec =

   0.7245 - 0.0343i   0.1920 - 0.0381i   0.0153 - 0.0115i   0.0232 - 0.0188i
   0.1920 - 0.0381i   0.6732 - 0.0703i   0.0232 - 0.0188i   0.0351 - 0.0306i
   0.2746 - 0.0841i   0.4241 - 0.1467i   0.0498 - 0.0423i   0.0755 - 0.0688i
   0.5589 - 0.0550i   0.3075 - 0.0611i   0.0244 - 0.0184i   0.0371 - 0.0300i
   0.0153 - 0.0115i   0.0232 - 0.0188i   0.7138 - 0.0461i   0.1748 - 0.0559i
   0.0232 - 0.0188i   0.0351 - 0.0306i   0.1748 - 0.0559i   0.6452 - 0.0970i
   0.0498 - 0.0423i   0.0755 - 0.0688i   0.2358 - 0.1221i   0.3614 - 0.2039i
   0.3075 - 0.0611i   0.4768 - 0.1126i   0.0371 - 0.0300i   0.0563 - 0.0490i
   0.1688 - 0.0666i   0.2599 - 0.1134i   0.1485 - 0.0863i   0.2271 - 0.1431i
   0.0244 - 0.0184i   0.0371 - 0.0300i   0.5418 - 0.0738i   0.2798 - 0.0894i
   0.0371 - 0.0300i   0.0563 - 0.0490i   0.2798 - 0.0894i   0.4319 - 0.1554i
```

Note: It is necessary to have **basmva** specified before calling **red_ybus**. The calling sequence is more complex if induction motors or generators or non-conforming loads are specified. You can see the necessary calling sequence by looking in the **s_simu** code.

Algorithm:

The function **red_ybus** sets up an admittance matrix which includes of the generator and induction motor internal buses and the load buses: the load buses include the SVC and the HVDC equivalent HT buses. Then Kron reduction is performed to eliminate all the load buses not specified in **load_con**. The buses are reordered so that the non conforming load buses are first, in the order in which they are specified in **load_con**. The other buses follow in the order in which they are specified in **bus**. Initially, the HVDC ac buses are the transformer LT buses specified in the load flow. However, **red_ybus** transforms these so that in transient simulation the bus voltage retained in Y_red is the equivalent HT converter bus. This makes the ac/dc interface much easier and yet still gives the freedom to specify a Thevenin equivalent reactance for the HVDC commutating reactance.

This algorithm is implemented in the M-file **red_ybus.m** in the POWER SYSTEM TOOLBOX.

See also: loadflow, ybus, pst_var, mac_em, mac_ind, mac_igen, mac_tra, mac_sub, nc_load, s_simu, svm_mgen y_switch

rlmod

Purpose:

A reactive load modulation control for transient simulation

Synopsis:

f = rlmod(i,k,bus,flag)

Description:

f = rlmod(i,k,bus,flag) contains the equations of a reactive load modulation control system for the initialization, machine interface and dynamics computation of the **i**$^{\mathbf{th}}$ modulation control.

Modulation is controlled through the global variable **rlmod_sig**. This is modified by the function **rml_sig** which should be written by the user to obtain the required load modulation charactersitic.

The m.file **pst_var.m** containing all the global variables required for **rlmod** should be loaded in the program calling **rlmod**.

Inputs:

i — the number of the reactive load modulation control
if **i** = 0 all load modulation computations are performed using MATLAB vector methods.
This is the preferred mode.

k — the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus — the solved bus specification matrix

flag — indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1.
- There is no need to perform a network interface calculation for **rlmod**
- The rates of change of the rlmod state is calculated when **flag** = 2, using the modulating signal **rlmod_sig** at the time specified by **k**

Output:

f is a dummy variable

Global Variables

System variables

basmva	system base MVA
bus_int	array to store internal bus ordering

Load Modulation Variables

rlmod_st	r*lm*	reactive load modulation state
drlmod_st	*drlm/dt*	
rlmod_sig	V_{sup}	supplementary signal into the reference input
rlmod_con		matrix of rlmod parameters supplied by user
rlmod_pot		internally calculated matrix of rlmod constants
n_rlmod		number of reactive load modulation controls
rlmod_idx		index of reactive modulation controls included in **load_con**

Data Format

The load modulation control data is contained in the **i**$^{\text{th}}$ row of the matrix **rlmod_con**. The data format for **rlmod_con** is given in Table 1.

Table 1. Data format for rlmod

column	variable	unit
1	reactive load modulation number	
2	bus number	
3	modulation base MVA	MVA
4	maximum susceptance r*lmod_max*	pu
5	minimum susceptance *rlmod_min*	pu
6	regulator gain K	pu
7	regulator time constant T_R	sec

Algorithm:

To use the **rlmod** function, the reactive load modulation buses must be declared via **load_con** as non-conforming load buses. The **rlmod** buses may also have non-conforming loads In the network interface computation, the reactive load modulation output is used to adjust the susceptance at the control buses in the solution for the bus voltages in **nc_load**. In the dynamics calculation, the rate of change of the load modulation control state is adjusted according to the signal **rlmod_sig**. An anti-windup limit is used to reset the state variable.

This algorithm is implemented in the M-file **rlmod** in the POWER SYSTEM TOOLBOX.

See also: nc_load, pst_var, rml_sig.

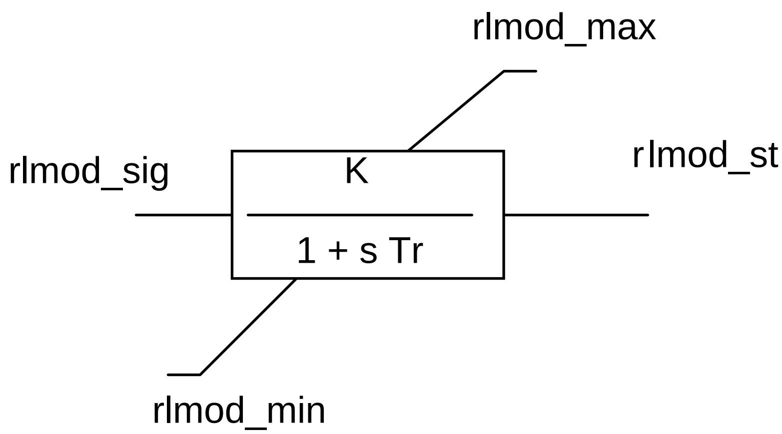


Figure 13 Reactive Load Modulation Control Block Diagram

rml_sig

Purpose:

Forms the reactive load modulation signal.

Synopsis:

f = rml_sig(t, k)

Description:

f = rml_sig forms the reactive load modulation signal as a function of time. The modulation variable **rlmod_sig** is passed as a global variable.

The m.file **pst_var.m** containing all the global variables should be loaded in the program calling **rml_sig**.

Inputs:

t	the time in seconds corresponding to **k**
k	the integer time step in a simulation In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

Output:

f	a dummy variable

Global Variable

rlmod_sig	V_{sup}	supplementary reactive load modulation signal
n_rlmod		number of reactive load modulation controls

See also: **rlmod**

Example

The following version of **rml_sig** causes a step change in reactive load after a time of 0.1 s.

```
function f = rml_sig(t,k)
% Syntax: f = rml_sig(t,k)
% defines modulation signal for rlmod control
global rlmod_sig n_rlmod
f=0; %dummy variable
if t<=0.1
   rlmod_sig(:,k) = zeros(n_rlmod,1);
else
   rlmod_sig(:,k) = 0.1*ones(n_rlmod,1);
end
return
```

s_simu

Purpose:

Acts as driver for transient simulation

Syntax:

s_simu

Description:

s_simu is a MATLAB script file which calls the models of the POWER SYSTEM TOOLBOX to

- select a data file
- perform a load flow
- initialize the non-linear simulation models
- do a step-by-step integration of the non-linear dynamic equations to give the response to a user specified system fault

Global variables

pst_var

Algorithm:

s_simu is the driver for transient stability analysis in the Power System Toolbox. It requires an input data set comprising of the following specification matrices

obligatory

- **bus** a bus specification matrix - not necessarily solved
- **line** a line specification matrix - not necessarily solved
- **mac_con** a generator specification matrix
- **sw_con** a switching specification file

optional

- **exc_con** an exciter specification matrix
- **pss_con** a power system stabilizer specification matrix
- **tg_con** a turbine governor specification matrix
- **ind_con** an induction motor specification matrix
- **mld_con** a motor load specification matrix
- **load_con** a non conforming load specification matrix
- **svc_con** an SVC specification matrix
- **dcsp_con** a dc converter specification matrix
- **dcl_con** a dc line specification matrix
- **dcc_con** a dc control specification matrix

Preliminary

1. After reading the data, **svm_mgen** performs a load flow if requested, otherwise the solved load flow data is extracted from a **mat** file with the same name as the data file.
   If the data contains dc specification files, a combined ac/dc load flow is performed.
2. The data is organized by calling the index m-files. These check to see which data is available.

Initialization

The non-linear models are initialized at the operating point set by the solved load flow. The induction motor, SVC and HVDC models are initialized before a reduced network admittance matrix is constructed since they alter the entries in the solved load flow bus specification matrix.

Reduced admittance matrices are constructed, using **red_ybus**, which relate the currents injected into the generators and motors to the internal generator and motor voltages and the voltages at the non conforming load and SVC buses (see **red_ybus**) under the fault conditions specified in **sw_con**.

The time vector **t** is defined based on the fault timing and time steps specified in **sw_con**. Switching points occur at the times specified in **sw_con**. To achieve this, the specified time steps are a guide only. The closest smaller time step which gives the required switching points is substituted for the time step specified.

Simulation

A predictor-corrector algorithm is used for the step-by-step integration of the system equations. At each time step

1. A network interface calculation is performed - flag = 1 in the device models. The non-linear equations for the load at the non-conforming load buses are solved to give the voltage at these buses. The current injected by the generators and absorbed by the motors is calculated from the reduced admittance matrix appropriate to the specified fault condition at that time step based on the machine internal voltages and the non-conforming load bus voltages.
2. The rates of change of the dynamic device model state variables is calculated - flag = 2 in the device models.
3. A predictor integration step is performed which gives an estimate of the states at the next time step.
4. A second network interface step is performed.
5. The rates of change of the dynamic device model state variables are recalculated.
6. A corrector integration step is performed to obtain the final value of the states at the next time step.

All calculations are performed using the MATLAB vector calculation facility. This results in a simulation time which is largely a function of the number of time steps. The time increases only slightly with the system size. However, in most simulations there are at least 500 time steps, and simulation is quite time consuming .

After every ten simulation time steps, the response of the bus voltage magnitude at the fault bus is shown on the screen. This allows the user to abort simulations which are clearly unsatisfactory (press control-c to abort the simulation).

At the completion of the simulation a menu of plots is presented to the user.

Many other variables are available for plotting if required. These include

- all dynamic states
- induction motor active and reactive powers (p_mot and q_mot)
- generator terminal voltage magnitudes (eterm)
- bus voltages (magnitude: abs(bus_v); angle: angle(bus_v))
- HDVC variables, Vdc, i_dc, alpha, gamma, dc control states, dc line states

For example, to plot all the generator terminal voltages against time use **plot(t,eterm)**

This algorithm is implemented in the M-file **s_simu** in the POWER SYSTEM TOOLBOX.

Example

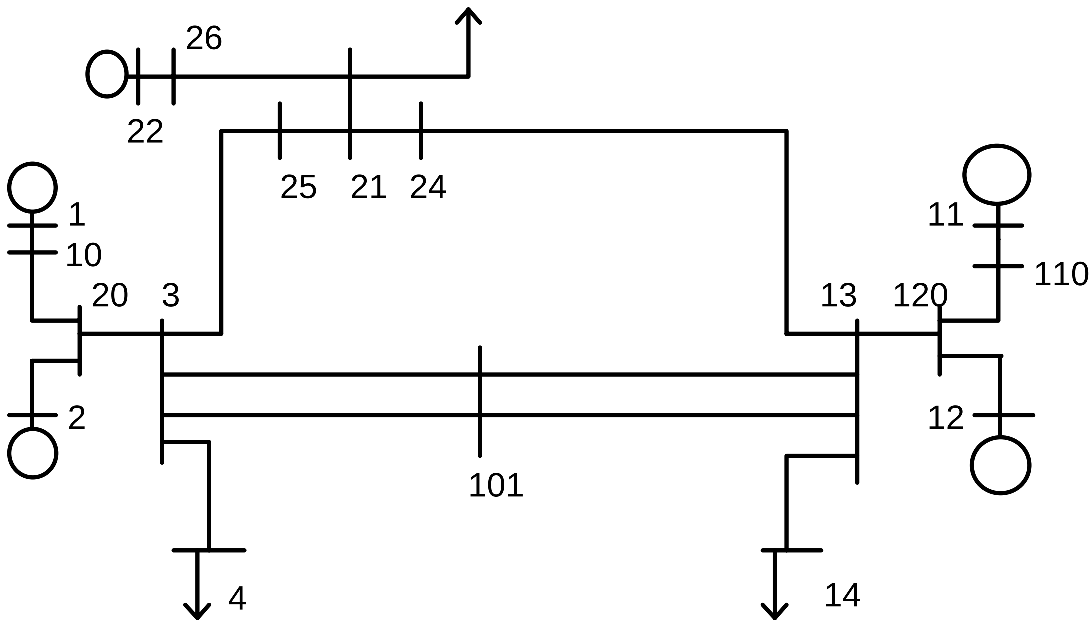


Figure 14 Two-Area System with added Load Area

The system shown in Figure 1 has the following data set.

```
bus = [...
1  1.03    18.5   7.00   1.61  0.00  0.00  0.00  0.00  1  99.0  -99.0 22.0  1.1  .9;
2  1.01    8.80   7.00   1.76  0.00  0.00  0.00  0.00  2  99.0  -99.0 22.0  1.1  .9;
3  0.9781  -6.1   0.00   0.00  0.00  0.00  0.00  0.00  3  0.0   0.0  500.0  1.5  .5;
4  0.95    -10    0.00   0.00  10.0  1.00  0.00  0.00  3  0.0   0.0  115.0  1.05 .95;
10 1.0103  12.1   0.00   0.00  0.00  0.00  0.00  0.00  3  0.0   0.0  230.0  1.5  .5;
11 1.03    -6.8   7.00   1.49  0.00  0.00  0.00  0.00  2  99.0  -99.0 22.0  1.1  .9;
12 1.01    -16.9  7.50   1.39  0.00  0.00  0.00  0.00  2  99.0  -99.0 22.0  1.1  .9;
13 0.9899  -31.8  0.00   0.00  0.00  0.00  0.00  0.00  3  0.0   0.0  500.0  1.5  .5;
14 0.95    -38    0.00   0.00  15.0  1.00  0.00  0.00  3  0.0   0.0  115.0  1.05 .95;
20 0.9876    2.1  0.00   0.00  0.00  0.00  0.00  0.00  3  0.0   0.0  230.0  1.5  .5;
21 1.0       0    0.00   0.00  5.0   2.0   0.00  0.0   3  0.00  0.00 115.0  1.5  .5;
22 1.0       0    1.50   1.5   0.00  0.00  0.00  0.00  2  99.0  -99.0 18.0  1.1  .9;
24 1.0       0    0      0     0     0     0     0     3  0     0    500.0  1.5  .5;
25 1.0       0    0      0     0     0     0     0     2  0     0    500.0  1.5  .5;
26 1.0       0    0      0     0     0     0     0     3  0     0    115.0  1.5  .5;
101 1.05     -19.3  0.00   8.00  0.00  0.00  0.00  0.00 2  99.0  -99. 500.0  1.5  .5;
110 1.0125   -13.4  0.00   0.00  0.00  0.00  0.00  0.00 3  0.0   0.0  230.0  1.5  .5;
120 0.9938   -23.6  0.00   0.00  0.00  0.00  0.00  0.00 3  0.0   0.0  230.0  1.5  .5 ];
```

```
line = [...
1   10  0.0     0.0167   0.00    1.0  0. 0.  0.  0.;
2   20  0.0     0.0167   0.00    1.0  0. 0.  0.  0.;
3    4  0.0     0.005     0.00   1.0  0. 1.2 0.8 0.05;
3   20  0.001   0.0100   0.0175  1.0  0. 0.  0.  0.;
3   101 0.011   0.110    0.1925  1.0  0. 0.  0.  0.;
3   101 0.011   0.110    0.1925  1.0  0. 0.  0.  0.;
3   25  0.011   0.110    0.1925  1.0  0  0   0   0 ;
13  24  0.019   0.19     0.3     1.0  0  0   0   0 ;
22  26  0.0     0.05     0.0     1.0  0  0   0   0 ;
24  21  0.0     0.01     0.0     1.0  0  0   0   0 ;
25  21  0.0     0.01     0.0     1.0  0  0   0   0 ;
26  21  0.02    0.2      0.375   1.0  0  0   0   0 ;
10  20  0.0025  0.025    0.0437  1.0  0. 0.  0.  0.;
11  110 0.0     0.0167   0.0     1.0  0. 0.  0.  0.;
12  120 0.0     0.0167   0.0     1.0  0. 0.  0.  0.;
13   14 0.0     0.005    0.00    1.0  0. 1.2 0.8 0.05;
13  101 0.011   0.11     0.1925  1.0  0. 0.  0.  0.;
13  101 0.011   0.11     0.1925  1.0  0. 0.  0.  0.;
13  120 0.001   0.01     0.0175  1.0  0. 0.  0.  0.;
110 120 0.0025  0.025    0.0437  1.0  0. 0.  0.  0.];

mac_con = [ ...

1 1 1000 0.200  0.0025  1.8  0.30  0.25 8.00  0.03...
                       1.7   0.55  0.25 0.4   0.05...
                       6.5   13    0    1;
2 2 1000 0.200  0.0025  1.8  0.30  0.25 8.00  0.03...
                       1.7   0.55  0.25 0.4   0.05...
                       6.5   13    0    2;
3 11 1000 0.200  0.0025  1.8 0.30  0.25 8.00  0.03...
                       1.7   0.55  0.25 0.4   0.05...
                       6.5   13    0    11;
4 12 1000 0.200  0.0025  1.8 0.30  0.25 8.00  0.03...
                       1.7   0.55  0.25 0.4   0.05...
                       6.5   13    0    12;
5 22 300 0.200  0.0025 1.8   0.3   0.25 5.00 0.03...
                       1.7   0.55  0.25 0.4  0.05...
                       5.0   10.0  0    22];
exc_con = [...
0 1 0.05 200.0  0     0     0    5.0  -5.0...
    0    0      0     0     0    0    0      0      0    0   0;
0 2 0.05 200.0  0     0     0    5.0  -5.0...
    0    0      0     0     0    0    0      0      0    0   0;
0 3 0.05 200.0  0     0     0    5.0  -5.0...
    0    0      0     0     0    0    0      0      0    0   0;
0 4 0.05 200.0  0     0     0    5.0  -5.0...
    0    0      0     0     0    0    0      0      0    0   0;
0 5 0.02 50.0   0.02  0.1   0.5  5.0  -2.0...
    0    0      0     0     0    0    0      0      0    0   0];

pss_con = [...
1 1 300.0  20.0  0.06  0.04  0.08  0.04  0.2  -0.05;
1 2 300.0  20.0  0.06  0.04  0.08  0.04  0.2  -0.05;
1 3 300.0  20.0  0.06  0.04  0.08  0.04  0.2  -0.05;
1 4 300.0  20.0  0.06  0.04  0.08  0.04  0.2  -0.05;
1 5 100.0  20.0  0.06  0.04  0.08  0.04  0.05 -0.01];

tg_con = [...
1  1  1  1.0  25.0  0.1  0.5 0.0 1.25 5.0;
1  2  1  1.0  25.0  0.1  0.5 0.0 1.25 5.0;
1  3  1  1.0  25.0  0.1  0.5 0.0 1.25 5.0;
1  4  1  1.0  25.0  0.1  0.5 0.0 1.25 5.0];

ind_con = [ ...
1  21  240.0 .001 .1 4 .015 .1  0.6 0 0 0 0 0 0.4];

mld_con = [ ...
1  21  .1  1  .7  5];
```

```
load_con = [21  0  0  0  0];

svc_con = [1  21 100  1  0  50  0.02];

sw_con = [...
0       0    0    0    0    0    0.01;%sets initial time step
0.1     25   3    0    0    0    0.005; %apply three phase fault at bus 25, on line 25-3
0.15    0    0    0    0    0    0.005; %clear fault at bus 25
0.20    0    0    0    0    0    0.005; %clear remote end
5.0     0    0    0    0    0    0.0 ]; % end simulation
```

The system has 5 generators at buses 1, 2, 11, 12 and 22. All generators have simple exciters and a power system stabilizer. The first four generators have turbine/governors modelled.

There are three load buses, 4, 14 and 21. The load at bus 21 has 40% motor content, the remaining loads are constant impedance.

There is an SVC set to control the voltage at bus 21.

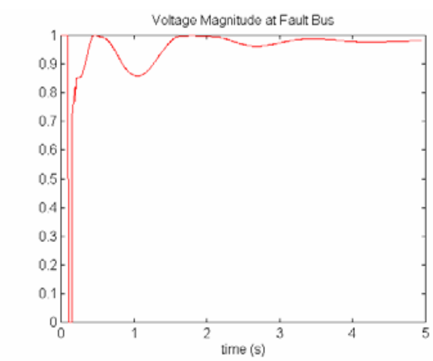
At 0.1s, a three phase fault is applied at bus 25 on line 3-25. At 0.15 s the line is disconnected at bus 25. The fault persists until 0.2 s when the line is disconnected from bus 3.

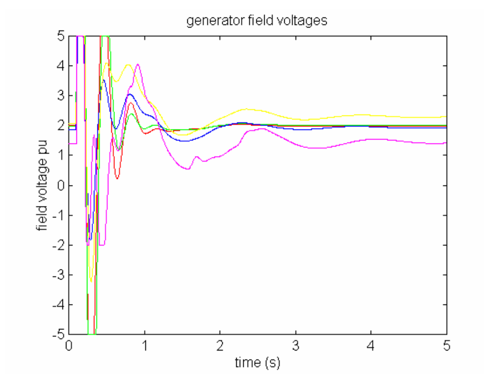
The simulation runs for 5 s. The time step is small (0,005 s) throughout because of the induction motor model.

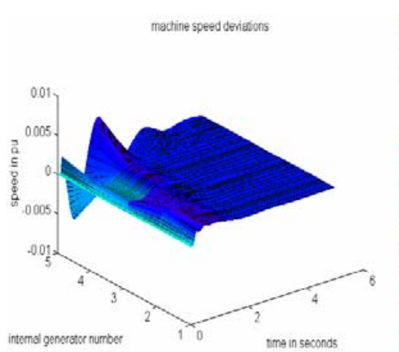
It is good practice to run a simulation for a short time before applying a fault. This allows a user to check that the system has a satisfactory, stable initial condition.

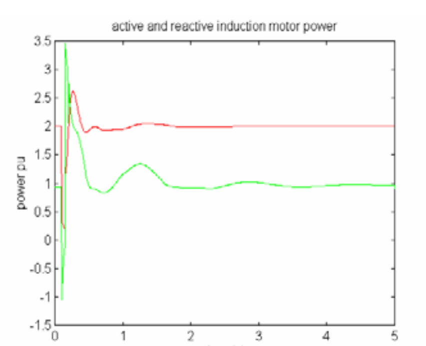
The following plots illustrate the

- fault bus voltage screen plot
- generator speed deviations
- exciter output voltages
- induction motor active and reactive loads









smpexc

Purpose:

Models simplified excitation systems

Synopsis:

f = smpexc(i,k,bus,flag)

Description:

smpexc(i,k,bus,flag) models the simplified excitation system shown in Figure 1. The m.file **pst_var.m** containing all the global variables required for **smpexc** should be loaded in the program calling **smpexc**.

Inputs:

i the number of the exciter
if **i** = 0 all simple exciter computations are performed using MATLAB vector methods. **This is the preferred mode.**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus the solved bus specification matrix

flag indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1. For proper initialization, the corresponding generators must be initialized first.
- The network interface calculation is performed when **flag** = 1, and the field voltage of the synchronous machine is set to the exciter output voltage.
- The rates of change of the exciter states are calculated when **flag** = 2, using the generator terminal voltage and the external system values at the time specified by **k**

Output:

f dummy variable

Global Variables:

Efd	E_{fd}	exciter output voltage, equal to generator field voltage pu
V_R	V_R	regulator output voltage in pu
V_A	V_A	regulator output voltage in pu
V_As	V_{As}	regulator voltage state variable in pu
R_f	R_f	stabilizing transformer state variable
V_FB	V_{FB}	feedback from stabilizing transformer
V_TR	V_{TR}	voltage transducer output in pu
V_B	V_B	potential circuit voltage output in pu
dEfd	dE_{fd}/dt	
dV_R	dV_R/dt	
dV_As	dV_{As}/dt	
dR_f	dR_f/dt	
dV_TR	dV_{TR}/dt	
exc_sig	V_{sup}	supplementary input signal to exciter ref input
exc_pot		matrix of internally set exciter constants
exc_con		matrix of exciter data set by user
smp_idx		index of simple exciters, i.e., exc_con(smp_idx,:)
n_smp		number of simple exciters

Data Format:

The exciter data are contained in the **i$^{\text{th}}$** row of the matrix variable **exc_con**. The data format for **smpexc** is shown in Table 1.

Table 1. Data format for smpexc

column	Variable	unit
1	exciter type	0
2	generator number	
3	transducer filter time constant T_R	sec
4	voltage regulator gain K_A	pu
5	voltage regulator time constant T_A	sec
6	transient gain reduction time constant T_B	sec
7	transient gain reduction time constant T_C	sec
8	maximum voltage regulator output V_{Rmax}	pu
9	minimum voltage regulator output V_{Rmin}	pu

If T_B is set to zero, then there will be no transient gain reduction.

Algorithm:

Based on the exciter block diagram, the exciter is initialized using the generator field voltage E_{fd} to compute the state variables. In the network interface computation, the exciter output voltage is converted to the field voltage of the synchronous machine. In the dynamics calculation, generator terminal voltage and the external signal is used to calculate the rates of change of the excitation system states.

This algorithm is implemented in the M-file **smpexc** in the POWER SYSTEM TOOLBOX.

See also: loadflow,pst_var,exc_dc12,exc_st3,mac_tra,mac_sub.

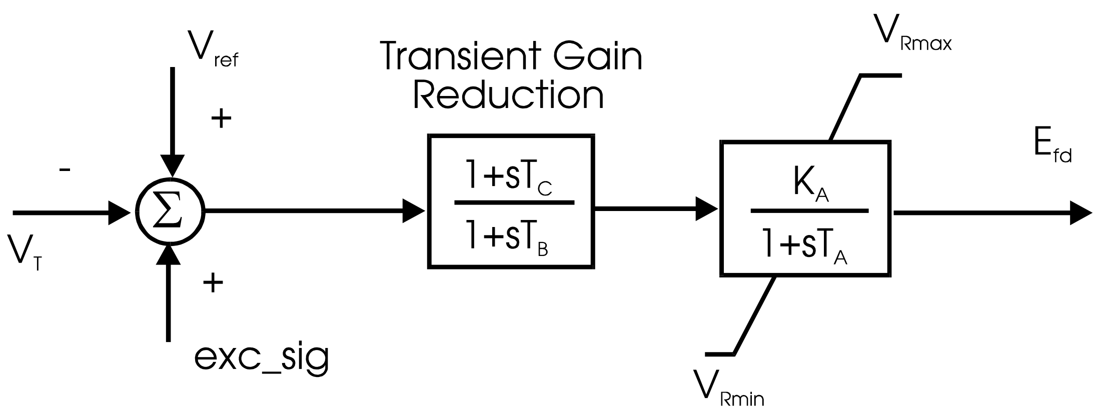


Figure 15 Simple Exciter

statef

Purpose:

Calculates the frequency response from system equations in state space form

Syntax:

[f,ymag,yphase]=statef(a,b,c,d,fstart,fstep,fend)

Description:

statef calculates the frequency response between a single input and a single output from the state space model of the system. It is used in **pss_des**.

Inputs:

a	the state matrix of the system for which frequency response is to be calculated
b	the input vector
c	the output row vector
d	the feed forward between input and output.
fstart	the starting frequency (Hz)
fstep	the frequency step (Hz)
fend	the end frequency (Hz)

Outputs:

f	the frequency vector
ymag	the output magnitude vector
yphase	the output phase vector (degrees)

Algorithm:

This algorithm is implemented in the M-file **statef.m** in the POWER SYSTEM TOOLBOX.

step_res

Purpose:

Step response from state space system definition

Synopsis:

[res t] = step_res(a,b,c,d,v_in,tmax)

Description:

step_res computes the step response from a state space system description.

$$\dot{x} = ax + bu$$

$$y = cx + du$$

The response is plotted on successful completion.

Inputs:

a	the state matrix of size ns by ns
b	the input matrix of size nx by nin
c	the out put matrix of size nout by nx
d	the feed forward matrix of size nout by nin
v_in	a column vector of length(nin) specifying the magnitude of the applied step
tmax	the maximum time of the response calculation (s)

nx - number of states (length(x))
nin - number of inputs (length(u))
nout - number of outputs (length(y))

Output:

res	a matrix of the response size(nx by length(t))
time	a vector of time

Algorithm:

The time step is chosen from the eigenvalues of a to give 5 time steps in the largest frequency or over the time constant of the fastest exponential decay.

The matrix exponential of (**a * t_step**) is calculated using **expm**.

The response is y is calculated from

$$x(:,k) = \exp(a * t\_step)\ x(:,k-1) - (I + \exp(a * t\_step)) inv(a)\ b\ v\_in$$
$$y(:,k) = c\ x(:,k) + d\ v\_in$$

The state matrices for a power system may be computed using **svm_mgen**.

svc

Purpose:

Models static VAR control systems

Synopsis:

bus_new = svc(i,k,bus,flag,v_sbus)

Description:

bus_new = svc(i,k,bus,flag,v_sbus) contains the equations of a static var control system [1] for the initialization, machine interface and dynamics computation of the **i**$^{\mathbf{th}}$ static var system.

A system oscillation damping control signal can be input to the static var system through the global variable **svc_sig** [1].

The m.file **pst_var.m** containing all the global variables required for **svc** should be loaded in the program calling **svc**.

Inputs:

i	the number of the SVC if **i** = 0 all SVC computations are performed using MATLAB vector methods. **This is the preferred mode.**
k	the integer time step in a simulation In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.
bus	the solved bus specification matrix
flag	indicates the mode of solution • Initialization is performed when **flag** = 0 and **k** = 1. • There is no need to perform a network interface calculation for **svc** • The rates of change of the SVC state is calculated when **flag** = 2, using the SVC terminal voltage value and the modulating signal **svc_sig** at the time specified by **k**
v_sbus	The SVC bus voltage

Output:

bus_new	On initialization **bus_new = bus** with the reactive generation at the SVC buses set to zero In other modes **bus_new = bus**

Global Variables

System variables

basmva	system base MVA
bus_int	array to store internal bus ordering

Static VAR Compensator Variables

B_cv	B_{cv}	svc susceptance in pu
dB_cv	dB_{cv}/dt	
svc_sig	V_{sup}	supplementary signal into the reference input
svc_con		matrix of svc parameters supplied by user
svc_pot		internally calculated matrix of svc constants
n_svc		number of svcs
svc_idx		index of svcs included in **load_con**

Data Format

The static var system data is contained in the **i**$^{\mathbf{th}}$ row of the matrix **svc_con**. The data format for **svc_con** is given in Table 1.

Table 1. Data format for svc

column	variable	unit
1	svc number	
2	bus number	
3	svc base MVA	MVA
4	maximum susceptance B_{cvmax}	pu
5	minimum susceptance B_{cvmin}	pu
6	regulator gain K_R	pu
7	regulator time constant T_R	sec

Algorithm:

To use the **svc** function, the static var system buses must be declared via **load_con** as non-conforming load buses with zero constant power and current components. The buses should be set to be generator buses, since the SVC picks up the reactive power generation to determine its initial susceptance setting. In the network interface computation, the static var system output is used to adjust the reduced network admittance matrix to solve for the bus voltages. This function is automatically performed in **nc_load**. In the dynamics calculation, the rate of change of the SVC state is adjusted according to the voltage error. An anti-windup limit is used to reset the susceptance state variable.

This algorithm is implemented in the M-file **svc** in the POWER SYSTEM TOOLBOX.

See also: nc_load, pst_var.

Reference:

1. E. V. Larsen and J. H. Chow, "SVC Control Concepts for System Dynamic Performance," in *Application of Static Var Systems for System Dynamic Performance*, IEEE Publications 87TH0187-5-PWR, 1987.

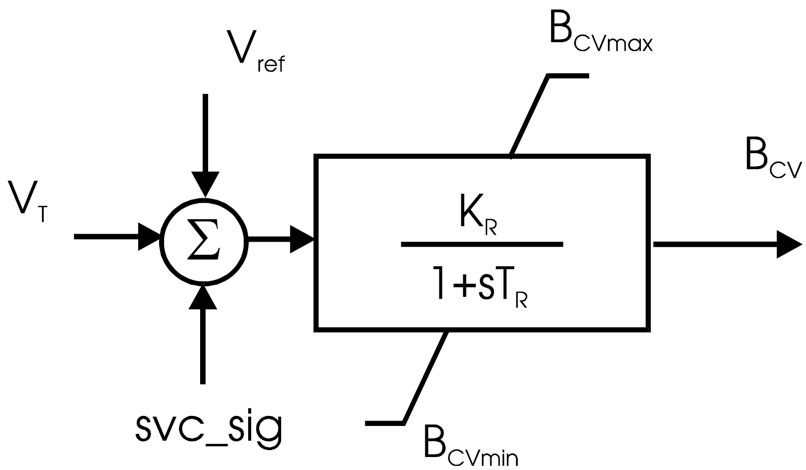


Figure 16 SVC Model Block Diagram

svc_indx

Purpose:

Forms indexes for svc calculation and checks for correct svc calling.

Syntax:

f = svc_indx

Outputs:

f a dummy variable

Global Variables:

Non Conforming Load Variables

load_con matrix of non conforming load parameters set by user

Static VAR Compensator Variables

svc_con matrix of svc parameters supplied by user
n_svc number of svcs
svc_idx index of svcs included in **load_con**

Algorithm:

Called before **svc** to set index. Finds the number of **svc's** and checks to see if they are declared correctly on **load_con**.

This algorithm is implemented in the M-file **svc_indx.m** in the POWER SYSTEM TOOLBOX.

svm_mgen

Purpose:

Forms the state matrices of a power system model, linearized about an operating point set by a load flow and performs modal analysis.

Syntax:

svm_mgen

Description:

svm_mgen is a MATLAB script file which calls the models of the POWER SYSTEM TOOLBOX to

- select a data file
- perform a load flow
- form a linearized model by perturbing each state in turn
- do a modal analysis of the system

Global variables

pst_var

Algorithm:

svm_mgen is the driver for small signal stability analysis in the Power System Toolbox. It requires an input data set comprising the following specification matrices

obligatory

- **bus** a bus specification matrix - not necessarily solved
- **line** a line specification matrix - not necessarily solved
- **mac_con** a generator specification matrix

optional

- **exc_con** an exciter specification matrix
- **pss_con** a power system stabilizer specification matrix
- **tg_con** a turbine governor specification matrix
- **ind_con** an induction motor specification matrix
- **mld_con** a motor load specification matrix
- **load_con** a non conforming load specification matrix
- **svc_con** an SVC specification matrix
- **ibus_con** an infinite bus specification vector
- **lmon_con** a line monitor specification vector

Preliminary

1. After reading the data, **svm_mgen** performs a load flow: the user is given the opportunity to revise **bus** and **line** to produce a post-fault, rather than pre-fault load flow.
2. The data is organized by calling the index m-files. These check to see which data is available
3. The number of system states are determined and a permutation matrix is formed which organizes the order of the states in the state matrix. In general, the states in the state matrix are ordered as follows:
    I. The generator and generator control states-in internal generator number order

II. The induction motor states in internal induction motor order
III. The svc states

The number of states in each device depends on the model data. However, the internal state matrices, as defined in **pst_var** have dimensions set only by the number of devices.

Initialization

All the devices are initialized at the operating point set by a system load flow. This gives the initial non-linear state vector. The infinite buses have no states, but their internal voltages are calculated from the original generator data and then stored. These voltages remain unchanged in following computations. The induction motor initialization (see **mac_ind**) determines the motor reactive power demand. This is subtracted from the bus reactive power load.

State matrix formation

Each state is perturbed in turn by a small value pert = (max(0.0001,0.001*state)). The rate of change **d_mat** of all the states is calculated. When the **i^th^** state is perturbed the **i^th^** column of the state matrix is calculated as

$$a\_mat(:,i) = p\_mat * d\_vector \,/\, pert$$

where *a_mat* is the state matrix and *p_mat* is the permutation matrix.

The input, output and feed forward matrices (**b**, **c**, **d**) are calculated at the same time for

inputs: exciter reference voltage $\mathbf{b_{vr}}$: turbine/governor power reference $\mathbf{b_{pr}}$: load modulation **b_lmod**: reactive load modulation **b_rlmod**

outputs: generator speed, $\mathbf{c_{sp}}$: generator electrical torque, $\mathbf{c_t}$: generator electrical power $\mathbf{c_p}$: line real and reactive power flow for monitored lines, **cpf1**, **cqfl**, **cpf2**, **cqf2.**
The monitored lines are specified in the input data by the vector
lmon_con which has length equal to the number of lines, entries of unity in the positions corresponding to the monitored lines and zero elsewhere.

feed forward: from V_{ref} to electrical torque, $\mathbf{d_{vrt}}$;to electrical power $\mathbf{d_{vrp}}$: from P_{ref} to electrical torque, $\mathbf{d_{prt}}$; to electrical power, $\mathbf{d_{prp}}$

Modal Analysis

Modal analysis is performed on the state matrix using the MATLAB **eig** function. This and storage considerations limits the total states of the modelled system to about 200.

The eigenvalues and right eigenvectors are calculated using **eig**. The left eigenvector is obtained by inverting the right eigenvector. The eigenvalues are ordered using **sort** and the columns of the eigenvector matrix are consistently permutated, They are stored in

l - eigenvalues vector
u - right eigenvector matrix ($\mathbf{i^{th}}$ column is the right eigenvector associated with **l**(i))
v - left eigenvector matrix ($\mathbf{i^{th}}$ row is the left eigenvector associated with **l**(i))

The participation vectors are stored as the columns of **p**. These values give the sensitivities of the eigenvalues to changes in the diagonal element of the state matrix. They are formed from

$$p(i,j) = u(i,j) * v(j,i)$$

The normalized participation vectors (the maximum modulus in each column is scaled to unity) are calculated and stored in **p_norm**. Values having a magnitude less than 0.1 are set to zero. The statement **sparse(abs(p_norm(:,j)))** indicates those states most influential in the control of the j^{th} eigenvalue.

Each of the columns of **p** and **p_norm** is associated with an eigenvalue, each of the rows is associated with a state.

Data about the structure of the state matrix is also available.

state(k) - gives the number of states associated with the kth generator
mac_state - has three columns

column 1 gives the overall state number

column 2 gives the state number within a particular generator and its controls

Generator
1 - δ
2 - ω
3 - E'_q
4 - ψ''_d
5 - E'_d
6 - ψ''_q

Exciter
7 - V_TR
8 - V_As
9 - V_R
10 - Efd
11 - R_f

Power System Stabilizer
12 - pss1
13 - pss2
14 - pss3

Turbine Governor
15 - tg1
16 - tg2
17 - tg3

column 3 gives the corresponding generator number

Thus, there are 17 possible states associated with each generator.

There are three states for each induction motor(v'_d, v'_q and s) which follow the generator states in the state vector in motor number order.

Each induction generator has three states which follow the induction motor states in the state vector in induction generator order.

Each svc has a single state (B_cv). The svc states follow the machine states in svc number order.

Each load modulation control has a single state (lmod_st). The load modulation states states follow the svc states in load modulation control number order.

Each HVDC link may have up to 5 states, these follow the svc states in the order, v_conr, v_coni, i_dcr, i_dci, v_dcc. If there is no line capacitor, the HVDC link model has only the first three states.

Thus the maximum number of states, which is the length of d_vector, is

$$17 * n\_mac + 3 * n\_mot + 3 * n\_ig + n\_svc + n\_l\mathrm{mod} + n\_rl\mathrm{mod} + 5 * n\_dcl$$

where n_mac is the number of generators, n_mot is the number of induction motors, n_ig is the number of induction generators, n_svc is the number of svcs, n_lmod is the number of load modulation controls, n_rlmod is the number of reactive load modulation controls and n_dcl is the number of HVDC lines.

Example

A two area system model data is contained in **data2a.m**. The m file listing is

```
% Two Area Test Case
% % bus data format
% bus:
% col1 number
% col2 voltage magnitude(pu)
% col3 voltage angle(degree)
% col4 p_gen(pu)
% col5 q_gen(pu),
% col6 p_load(pu)
% col7 q_load(pu)
% col8 G shunt(pu)
% col9 B shunt(pu)
% col10 bus_type
%       bus_type - 1, swing bus
%                - 2, generator bus (PV bus)
%                - 3, load bus (PQ bus)
% col11 q_gen_max(pu)
% col12 q_gen_min(pu)
% col13 v_rated (kV)
% col14 v_max  pu
% col15 v_min  pu

bus = [ ...
1   1.03    18.5   7.00   1.61  0.00  0.00  0.00  0.00 1  99.0 -99.0   22.0  1.1  .9;
2   1.01    8.80   7.00   1.76  0.00  0.00  0.00  0.00 2  5.0   -2.0   22.0  1.1  .9;
3   0.9781  -6.1   0.00   0.00  0.00  0.00  0.00  0.00 3  0.0    0.0  500.0  1.5  .5;
4   0.95    -10    0.00   0.00  9.76  1.00  0.00  0.00 3  0.0    0.0  115.0  1.05 .95;
10  1.0103  12.1   0.00   0.00  0.00  0.00  0.00  0.00 3  0.0    0.0  230.0  1.5  .5;
11  1.03    -6.8   7.16   1.49  0.00  0.00  0.00  0.00 2  5.0   -2.0   22.0  1.1  .9;
12  1.01    -16.9  7.00   1.39  0.00  0.00  0.00  0.00 2  5.0   -2.0   22.0  1.1  .9;
13  0.9899  -31.8  0.00   0.00  0.00  0.00  0.00  0.00 3  0.0    0.0  500.0  1.5  .5;
14  0.95    -38    0.00   0.00  17.67 1.00  0.00  0.00 3  0.0    0.0  115.0  1.05 .95;
20  0.9876   2.1   0.00   0.00  0.00  0.00  0.00  0.00 3  0.0    0.0  230.0  1.5  .5;
101 1.05    -19.3  0.00   8.00  0.00  0.00  0.00  0.00 2  99.0 -99.0  500.0  1.5  .5;
110 1.0125  -13.4  0.00   0.00  0.00  0.00  0.00  0.00 3  0.0    0.0  230.0  1.5  .5;
120 0.9938  -23.6  0.00   0.00  0.00  0.00  0.00  0.00 3  0.0    0.0  230.0  1.5  .5];

line = [...
1   10  0.0     0.0167   0.00    1.0  0. 0.  0.  0.;
2   20  0.0     0.0167   0.00    1.0  0. 0.  0.  0.;
3    4  0.0     0.005     0.00   1.0  0. 1.2 0.8 0.05;
3   20  0.001   0.0100   0.0175  1.0  0. 0.  0.  0.;
3   101 0.011   0.110    0.1925  1.0  0. 0.  0.  0.;
3   101 0.011   0.110    0.1925  1.0  0. 0.  0.  0.;
10  20  0.0025  0.025    0.0437  1.0  0. 0.  0.  0.;
11  110 0.0     0.0167   0.0     1.0  0. 0.  0.  0.;
12  120 0.0     0.0167   0.0     1.0  0. 0.  0.  0.;
13   14 0.0     0.005    0.00    1.0  0. 1.2 0.8 0.05;
13  101 0.011   0.11     0.1925  1.0  0. 0.  0.  0.;
13  101 0.011   0.11     0.1925  1.0  0. 0.  0.  0.;
13  120 0.001   0.01     0.0175  1.0  0. 0.  0.  0.;
110 120 0.0025  0.025    0.0437  1.0  0. 0.  0.  0.];

mac_con = [ ...

1 1 900 0.200  0.0025  1.8  0.30  0.25 8.00  0.03...
                       1.7  0.55  0.25 0.4   0.05...
  6.5  0  0  1;
2 2 900 0.200  0.0025  1.8  0.30  0.25 8.00  0.03...
                       1.7  0.55  0.25 0.4   0.05...
  6.5  0  0  2;
3 11 900 0.200  0.0025  1.8  0.30  0.25 8.00  0.03...
                       1.7  0.55  0.25 0.4   0.05...
  6.5  0  0  11;
4 12 900 0.200  0.0025  1.8  0.30  0.25 8.00  0.03...
                       1.7  0.55  0.25 0.4   0.05...
  6.5  0  0  12];
```

```
exc_con = [...
1 1 0.01 46.0   0.06  0     0    1.0   -0.9...
    0.0  0.46   3.1   0.33  2.3  0.1   0.1   1.0    0    0   0;
3 2 0.01 7.04   1.0   6.67  1.0 10.0  -10.0 ...
    0.2  -0.2   200.0 4.37  20   4.83  0.09  1.1    8.63 1.0 6.53;
0 3 0.01 200.0  0     0     0    5.0  -5.0...
    0    0      0     0     0    0    0      0      0    0   0;
2 4 0.01 300.0  0.01  0     0    4.95  -4.9...
    1.0  1.33   3.05  0.279 2.29 0.117 0.1   0.675  0    0   0];

pss_con = [...
1 3 300.0  20.0  0.06  0.04  0.08  0.04  0.2 -0.05];



tg_con = [...
1  1  1  1.0  25.0  0.1  0.5 0.0 1.25 5.0;
1  2  1  1.0  25.0  0.1  0.5 0.0 1.25 5.0;
1  3  1  1.0  25.0  0.1  0.5 0.0 1.25 5.0;
1  4  1  1.0  25.0  0.1  0.5 0.0 1.25 5.0];


load_con = [4  0  0  .5  0;
           14  0  0  .5  0;
           101 0  0   0  0];


svc_con = [1  101  100  10  -10  100  0.05];



sw_con = [...
0    0    0    0    0    0    0.01;%sets intitial time step
0.1  3    101  0    0    0    0.005; %apply three phase fault at bus 3, on line 3-101
0.15 0    0    0    0    0    0.005556; %clear fault at bus 3
0.20 0    0    0    0    0    0.005556; %clear remote end
0.50 0    0    0    0    0    0.01; % increase time step
1.0  0    0    0    0    0    0.02; % increase time step
5.0  0    0    0    0    0    0]; % end simulation
```

Note that this m file contains a switching file - this data is ignored in **svm_mgen**. It does not contain a line monitor specification ector and so the output matrices for line flow are not calculated.

Selected results of calling **svm_mgen** with this data set are given below.

The total number of states may be found from **sum(state)**

```
t_states = sum(state)

t_states =

    52
```

The distribution of the states between the system's devices can be seen from

```
state =

    13
    12
    13
    13
     1
```

State indicates that there are 13 states in the first, third and fourth generator models, 12 states in the second generator model and 1 state in the svc model.

The type of variable represented by each generator state can be found from mac_state

```
mac_state =

     1     1     1
     2     2     1
     3     3     1
     4     4     1
     5     5     1
     6     6     1
     7     7     1
     8     9     1
     9    10     1
    10    11     1
    11    15     1
    12    16     1
    13    17     1
    14     1     2
    15     2     2
    16     3     2
    17     4     2
    18     5     2
    19     6     2
    20     7     2
    21     8     2
    22     9     2
    23    15     2
    24    16     2
    25    17     2
    26     1     3
    27     2     3
    28     3     3
    29     4     3
    30     5     3
    31     6     3
    32     7     3
    33    12     3
    34    13     3
    35    14     3
    36    15     3
    37    16     3
    38    17     3
    39     1     4
    40     2     4
    41     3     4
    42     4     4
    43     5     4
    44     6     4
    45     7     4
    46     9     4
    47    10     4
    48    11     4
    49    15     4
    50    16     4
    51    17     4
```

Thus states 13, 25, 38 and 51 are the tg3 variables of the turbine governors at generators 1, 2, 3 and 4 respectively. There is a power system stabilizer on generator 3, the power system stabilizer states are 33, 34 and 35.

Eigenvalues

```
l =

 1.8614e-003
-1.4128e-002
-1.9983e-001
-1.9993e-001
-1.9993e-001
-2.2182e-001- 1.0403e-001i
-2.2182e-001+ 1.0403e-001i
-3.3501e-001- 6.4610e-001i
-3.3501e-001+ 6.4610e-001i
```

```
-4.1976e-001- 6.2672e-001i
-4.1976e-001+ 6.2672e-001i
-1.4866e+000- 8.9312e-001i
-1.4866e+000+ 8.9312e-001i
-1.9787e+000
-1.9949e+000
-1.9985e+000
-1.9994e+000
-2.7836e+000
-1.7140e-001- 3.5303e+000i
-1.7140e-001+ 3.5303e+000i
-3.5533e+000
-4.0663e+000
-4.8900e+000
-5.8495e-001- 6.8236e+000i
-5.8495e-001+ 6.8236e+000i
-1.3822e+000- 6.9188e+000i
-1.3822e+000+ 6.9188e+000i
-1.0003e+001
-1.0003e+001
-1.0004e+001
-1.0005e+001
-8.6348e+000- 9.9010e+000i
-8.6348e+000+ 9.9010e+000i
-2.0193e+001
-1.7033e+001- 1.3654e+001i
-1.7033e+001+ 1.3654e+001i
-2.8861e+001
-3.0240e+001
-3.1305e+001
-3.2859e+001
-3.5646e+001
-3.6145e+001
-3.6630e+001
-3.6737e+001
-5.0325e+001
-5.1437e+001- 3.0716e+001i
-5.1437e+001+ 3.0716e+001i
-9.6678e+001
-9.9912e+001
-9.9997e+001
-1.0040e+002
-1.2898e+002
```

The eigenvalues are ordered from minimum to maximum modulus using the MATLAB function **sort**.

The nature of the modes

There is a single unstable eigenvalue - this is an approximation to the zero eigenvalue which exists in all linearized power system models which do not have an infinite bus. The state matrix should be singular, since an equal change in all generator rotor angles has no effect on the system dynamics. Rounding errors normally cause the "zero" eigenvalue to have a small positive or negative value. With at least one infinite bus defined, the state matrix is not singular and in such a case there is no zero eigenvalue.

All other modes are stable - they have negative real parts. Some are real and some are complex.

There are 10 complex conjugate pairs of complex eigenvalues which represent the oscillatory system modes.

The least damped modes are 19 and 20 which have a damping ratio of 0.048494 and a frequency of 0.56187 Hz.

The states associated with this mode may be determined by

```
sparse(abs(p_norm(:,19)))

ans =

   (1,1)      8.7728e-001
   (2,1)      8.7961e-001
  (14,1)      4.7238e-001
  (15,1)      4.7363e-001
  (16,1)      2.5674e-001
  (26,1)      1.0000e+000
  (27,1)      9.4103e-001
  (28,1)      1.1731e-001
  (39,1)      8.4859e-001
  (40,1)      8.5085e-001
```

This indicates that the state with the largest normalized participation factor is state 26. The 26^{th} row of mac_state is

```
mac_state(26,:)

ans =

    26     1     3
```

This shows that state 26 is the rotor angle of generator 3.
State 27 is the speed of generator 3 and state 28 is E'_q for generator 3.

We can determine the nature of the other states in a similar manner, states 1 and 2 are the rotor states for generator 1, states 14 an 15 are the rotor states for generator 2, and states 39 and 40 are the rotor states for generator 4.

We thus see that this mode is an inter-area mode associated with all the system's generators.

The generator speed participation factors indicate possible sites for power system stabilizers which would add damping to this mode. Generator 3 has the largest speed participation. However, generator 3 has a power system stabilizer already fitted. Generators 1 and 4 are the next best sites for stabilizer placement.

The phase of the inter-area oscillations can be determined from the right eigenvector **u(:,19)**.

```
 [abs(u(1:51,19)) angle(u(1:51,19))*180/pi mac_state]

ans =

    0.1679   30.7260    1.0000    1.0000    1.0000
    0.0016  -62.0536    2.0000    2.0000    1.0000
    0.0029  155.3193    3.0000    3.0000    1.0000
    0.0129 -166.7025    4.0000    4.0000    1.0000
    0.0067  -65.2889    5.0000    5.0000    1.0000
    0.0090  -82.3560    6.0000    6.0000    1.0000
    0.0238  178.6530    7.0000    7.0000    1.0000
    0.3656  -56.3350    8.0000    9.0000    1.0000
    0.2251   34.3749    9.0000   10.0000    1.0000
    0.0621  111.1662   10.0000   11.0000    1.0000
    0.0015  137.7041   11.0000   15.0000    1.0000
    0.0008 -159.6786   12.0000   16.0000    1.0000
    0.0000  -70.1427   13.0000   17.0000    1.0000
    0.1259   24.1524   14.0000    1.0000    2.0000
    0.0012  -68.6272   15.0000    2.0000    2.0000
    0.0223   91.4914   16.0000    3.0000    2.0000
    0.0223  125.3269   17.0000    4.0000    2.0000
    0.0030  109.9738   18.0000    5.0000    2.0000
    0.0041   92.9067   19.0000    6.0000    2.0000
    0.0264  166.8798   20.0000    7.0000    2.0000
    0.2244   77.2283   21.0000    8.0000    2.0000
    0.1074    1.0440   22.0000    9.0000    2.0000
    0.0011  131.1306   23.0000   15.0000    2.0000
    0.0006 -166.2521   24.0000   16.0000    2.0000
    0.0000  -76.7162   25.0000   17.0000    2.0000
    0.1933 -163.9307   26.0000    1.0000    3.0000
    0.0018  103.2897   27.0000    2.0000    3.0000
    0.0478  112.8281   28.0000    3.0000    3.0000
    0.0476  108.7490   29.0000    4.0000    3.0000
    0.0138   86.6689   30.0000    5.0000    3.0000
    0.0187   69.6018   31.0000    6.0000    3.0000
    0.0264  103.4275   32.0000    7.0000    3.0000
    0.0000 -164.7407   33.0000   12.0000    3.0000
    0.0136  -69.4278   34.0000   13.0000    3.0000
    0.0274  -73.4155   35.0000   14.0000    3.0000
    0.0017  -56.9525   36.0000   15.0000    3.0000
    0.0009    5.6648   37.0000   16.0000    3.0000
    0.0000   95.2006   38.0000   17.0000    3.0000
    0.1950  179.1177   39.0000    1.0000    4.0000
    0.0018   86.3381   40.0000    2.0000    4.0000
    0.0049   83.3743   41.0000    3.0000    4.0000
    0.0108   47.3692   42.0000    4.0000    4.0000
    0.0186   29.2650   43.0000    5.0000    4.0000
    0.0253   12.1980   44.0000    6.0000    4.0000
    0.0097   89.0305   45.0000    7.0000    4.0000
    0.3125 -141.7445   46.0000    9.0000    4.0000
    0.0654  -62.5986   47.0000   10.0000    4.0000
    0.0257    7.0419   48.0000   11.0000    4.0000
    0.0018  -73.9042   49.0000   15.0000    4.0000
    0.0009  -11.2869   50.0000   16.0000    4.0000
    0.0000   78.2490   51.0000   17.0000    4.0000
```

State 52 is the svc state and u(52,19) = 1

The rotor states are 1,2 ; 14,15 ; 26,27 and 39,40. Thus in this mode the generators 2 and 4 swing against 1 and 2.

The largest eigenvector magnitude is associated with state 52 which is the svc state.

Note that the participation factor for this state is less than 1% of the maximum participation factor in this mode.

tg

Purpose:

Simplified turbine-governor system model

Synopsis:

f = tg(i,k,bus,flag)

Description:

tg(i,k,bus,flag) models the simplified turbine-governor system model shown in Figure 1.

Inputs:

i the number of turbine governor
if **i** = 0 all turbine governor computations are performed using MATLAB vector methods. **This is the preferred mode.**

k the integer time step in a simulation
In small signal simulation, only two values of **k** are used. At **k** = 1, the state variables and there rates of change are set to the initial values. At **k** = 2, the state variables are perturbed in turn and the rates of change of states correspond to those cause by the perturbation.

bus the solved bus specification matrix

flag indicates the mode of solution

- Initialization is performed when **flag** = 0 and **k** = 1. For proper initialization, the corresponding generators must be initialized first.
- The network interface calculation is performed when **flag** = 1, and the mechanical torque of the synchronous machine is set to the turbine output torque.
- The rates of change of the turbine governor states are calculated when **flag** = 2, using the generator speed deviation at the time specified by **k**

Output:

f a dummy variable

Global Variables

System variables

basmva	system base MVA

Synchronous Generator Variables

mac_spd	ω	machine speed in pu
pmech	P_m	mechanical input power in pu on generator base
pelect	P_e	electrical active output power in pu on system base
mac_int		array to store internal machine ordering

Turbine-governor Variables

tg1	governor state variable
tg2	servo state variable
tg3	reheater state variable
dtg1	
dtg2	
dtg3	
tg_con	matrix of turbine governor specifications set by user
tg_pot	internally set matrix of turbine governor constants
n_tg	number of turbine governors
tg_idx	index of turbine governors

Data Format

The data format for the specification file **tg_con** is shown in Table 1.

Table 1. Data format for tg

column	variable	unit
1	turbine model number (=1)	
2	machine number	
3	speed set point ω_f	pu
4	steady state gain $1/R$	pu
5	maximum power order T_{max}	pu on generator base
6	servo time constant T_s	sec
7	HP turbine time constant T_c	sec
8	transient gain time constant T_3	sec
9	time constant to set HP ratio T_4	sec
10	reheater time constant T_5	sec

No time constant is allowed to set to zero in this model. The function **tg** can be used to model either a steam unit or a hydro unit. The time constant T_4 should be set such that T_4/T_5 is the HP power fraction. For a hydro unit, T_4 would be negative.

Algorithm:

Based on the turbine-governor system model block diagram

- the initialization uses mechanical torque from the synchronous machine to compute the state variables on the integrators. If speed set point is not equal to 1 pu, the power order will be adjusted to give a torque output of the turbine which achieves steady state
- the network interface calculates the output mechanical torque for use by the corresponding generator
- the dynamics calculation determines the rates of change of the turbine governor state variables

This algorithm is implemented in **tg** in the POWER SYSTEM TOOLBOX.

See also: pst_var, mac_em, mac_tra, mac_sub

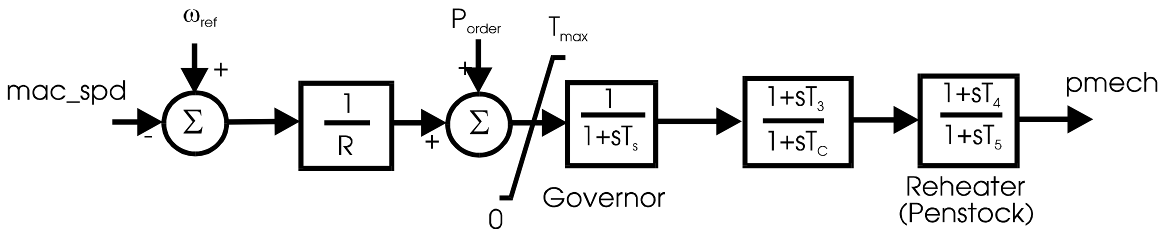


Figure 17 Simple Turbine Governor Model

tg_indx

Purpose:

Determines indexes for the turbine generators

Syntax:

f=tg_indx

Outputs:

f	a dummy variable

Global Variables:

Turbine-governor Variables

tg_con	matrix of turbine governor specifications set by user
n_tg	number of turbine governors
tg_idx	index of turbine governors

Algorithm:

determines the number of turbine governor models and sets the turbine governor index

This algorithm is implemented in the M-file **tg_indx** in the POWER SYSTEM TOOLBOX.

y_switch

Purpose:

Forms reduced admittance matrices to correspond with the switching conditions specified in **sw_con**.

Syntax:

y_switch

Description:

y_switch is a MATLAB script file which is called from **s_simu**. It is uses the switching data contained in **sw_con** to define the reduced admittance matrices required for transient simulation, i.e., for pre-fault, fault, immediate post-fault, final fault clear.

Data Format

The switching is specified in sw_con which has the format shown in Table 1.

Table 1 Switching File Format

time of fault(s)	fault bus number	far bus number	zero sequence fault impedance (pu)	negative sequence fault impedance (pu)	type of fault	time step for fault period(s)
start time	0	0	0	0	0	initial time step
fault on time	fb#	far b#	zs pu	zn pu	0 -three phase 1 - line to ground 2 - line-to-line-ground 3 - line-to-line 4 - loss of line no fault 5 - loss of load 6 - no fault	fault-on time step
initial fault clearing time	0	0	0	0	0	time step
final fault clearing time	0	0	0	0	0	time step
time to change time step						time step
end time						

There may be any number of entries changing the time step following final fault clearing. This allows the use of longer simulation time steps after any initial fast transients have decayed, so allowing faster computation time.

The no fault option is useful when the effect of modulation of control signals is to be studied.

Example

The switching data file for the two-area system in **data2a.m** is

```
sw_con = [...
0    0    0    0    0    0    0.01;     %sets intitial time step
0.1  3    101  0    0    0    0.005;    %three phase fault at bus 3 line 3 to 101
0.15 0    0    0    0    0    0.005556; %clear fault at bus 3
0.20 0    0    0    0    0    0.005556; %clear remote end
0.50 0    0    0    0    0    0.01;     % increase time step
1.0  0    0    0    0    0    0.02;     % increase time step
5.0  0    0    0    0    0    0];      % end simulation
```

Note: It is always worth while applying the fault at some short time after the start of the simulation. This allows a check on the unfaulted system which should remain in its initial state. If the initial states drift considerably, the initial rates of change of the states should be checked. These should all be zero, or very close to zero. Non-zero initial rates of change indicate the source of any problem.