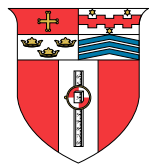
CURENT Course

Power System Toolbox

Prof. Joe H. Chow



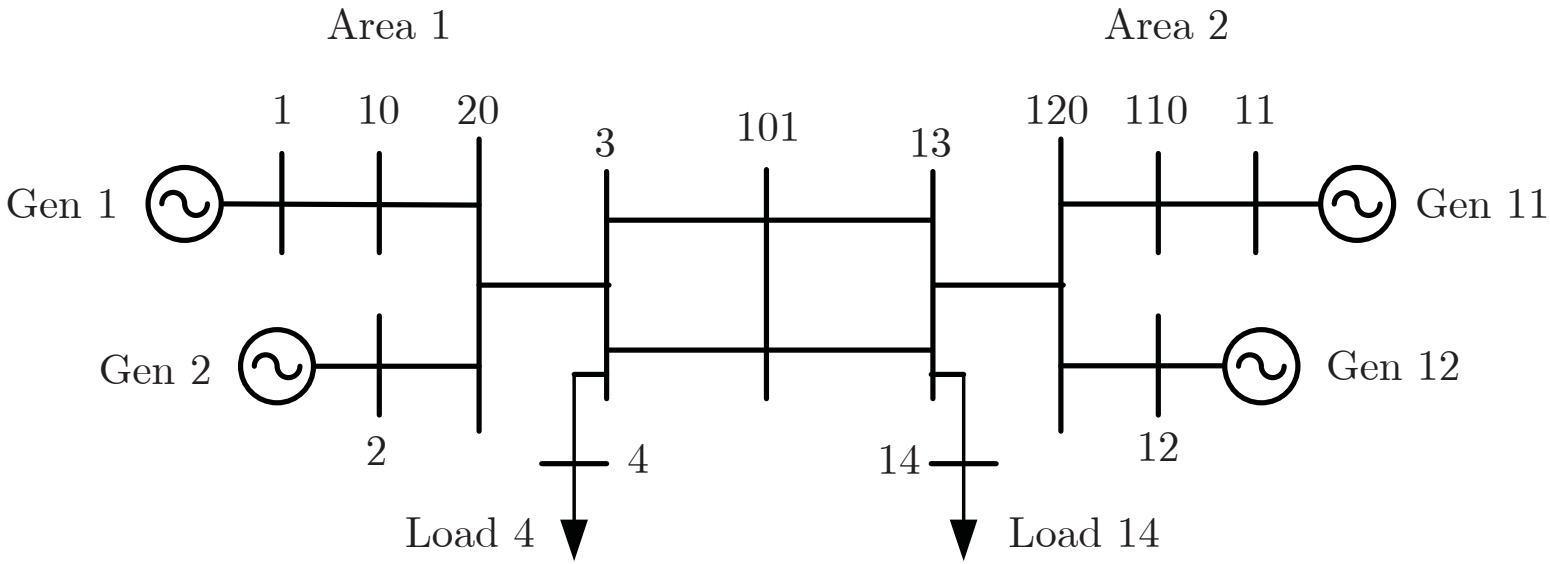
Rensselaer Polytechnic Institute
ECSE Department

September 1, 2015

Power System Toolbox

- Developers: Joe Chow, Kwok Cheung, and Graham Rogers (Ontario Hydro and Cherry Tree Scientific Software)
- Power System Toolbox uses MATLAB code to perform (1) power flow computation, (2) dynamic simulation, and (3) linear model generation
- In a power network, generators are supplying power to the loads. A power flow program uses the power network to compute the flow of power from generation to load
- In dynamic simulation, the impact of a disturbance on a power system, such as stability, is simulated. In the simulation part, dynamic models of generators, excitation systems, and governors of various details are used.
- Linearization uses small perturbations from equilibrium point to generate (A, B, C, D) matrices. (not covered in this lecture)

Power Flow Computation



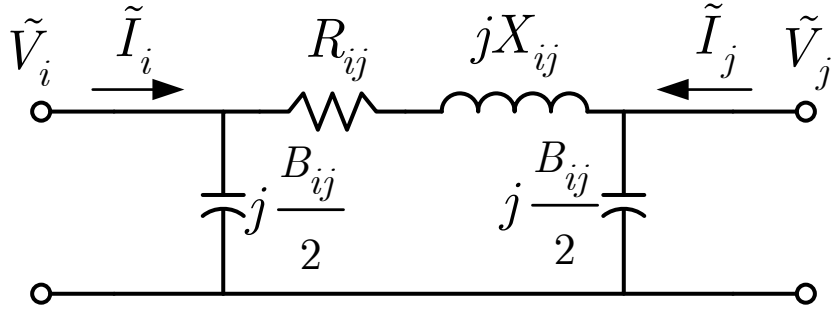
We need the following:

1. Active and reactive power flow on transmission lines and transformers
2. Network admittance matrix
3. Summation of flow at each node (bus)
4. Forming a Jacobian matrix for using the Newton-Raphson method
5. Others such as sparse factorization

Power Network Admittance Matrix

A power network admittance matrix to relate bus voltages and line currents can be developed iteratively.

1. Buses i and j connected via a transmission line with 3 parameters: resistance R_{ij}, reactance $X_{ij} = \omega L_{ij} = (2\pi f) L_{ij}$, and line charging (susceptance) $B_{ij} = \omega C_{ij}$ (that is why the value of B is positive)



The current $\tilde{I}_i$ is given by, with $Y_{ij} = 1/Z_{ij} = 1/(R_{ij} + jX_{ij})$

$$\tilde{I}_i = \text{current in series reactance} \; + \; \text{current in shunt susceptance}$$

$$= Y_{ij}(\tilde{V}_i - \tilde{V}_j) + \tilde{V}_i \cdot j\frac{B_{ij}}{2} = \left(Y_{ij} + j\frac{B_{ij}}{2}\right)\tilde{V}_i - Y_{ij}\tilde{V}_j$$

Similarly, the current $\tilde{I}_j$ is given by

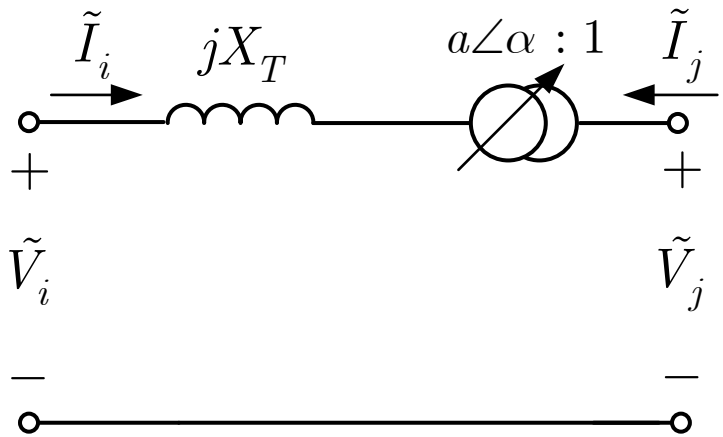
$$\tilde{I}_j = \left(Y_{ij} + j\frac{B_{ij}}{2} \right) \tilde{V}_j - Y_{ij}\tilde{V}_i$$

In matrix form

$$\begin{bmatrix} Y_{ij} + j\frac{B_{ij}}{2} & -Y_{ij} \\ -Y_{ij} & Y_{ij} + j\frac{B_{ij}}{2} \end{bmatrix} \begin{bmatrix} \tilde{V}_i \\ \tilde{V}_j \end{bmatrix} = \begin{bmatrix} \tilde{I}_i \\ \tilde{I}_j \end{bmatrix} \quad \Rightarrow \quad Y\tilde{V} = \tilde{I}$$

where Y is generically known as the admittance matrix, $\tilde{V}$ is the vector of voltage phasors, and $\tilde{I}$ is the (injected) current vector.

2. Buses i and j connected via a transformer with parameters: reactance $X_T = (2\pi f)L_T$, and tap ratio a for a regular transformer or tap ratio a and phase α for a phase-shifting transformer.
Circuit diagram (modified from an earlier diagram)



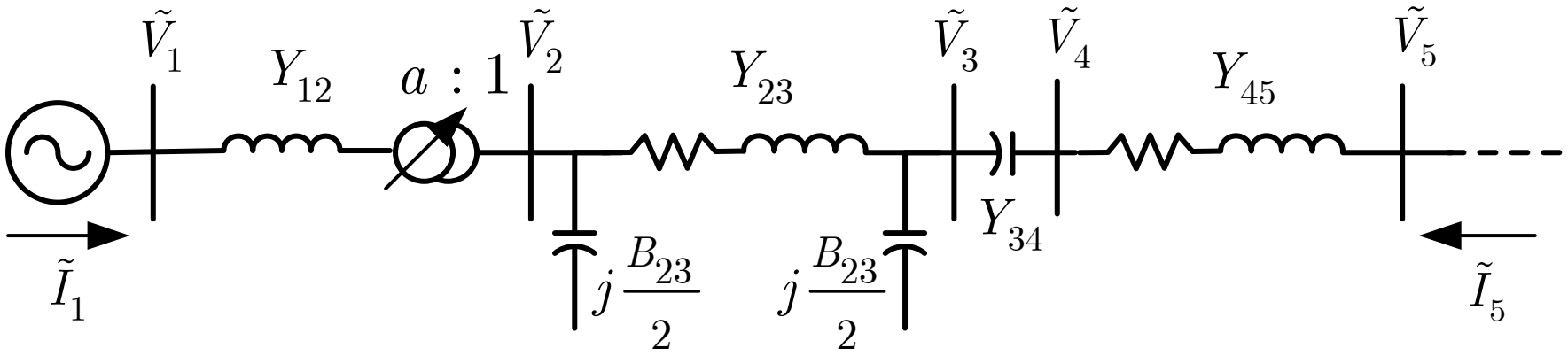
The voltage and current expression is

$$\begin{bmatrix} Y_T & -\tilde{a}Y_T \\ -\tilde{a}^* Y_T & a^2 Y_T \end{bmatrix} \begin{bmatrix} \tilde{V}_i \\ \tilde{V}_j \end{bmatrix} = \begin{bmatrix} \tilde{I}_i \\ \tilde{I}_j \end{bmatrix} \quad \Rightarrow \quad Y\tilde{V} = \tilde{I}$$

where $Y_T = 1/(jX_T)$.

Admittance Matrix Example

Consider a radial 5-bus system example



There are no injections (loads) at Buses 2, 3, and 4.
Line 1-2 is a generator step-up transformer, Line 2-3 has line charging, Line 3-4 is a series capacitor compensation, Line 4-5 is a line without charging, and Bus 5 is connected to other buses (not shown).

The network voltage-current equation $Y\tilde{V} = \tilde{I}$ ($\tilde{I}$ is the current injection into the bus) is given by

$$\begin{bmatrix} Y_{11} & -aY_{12} & 0 & 0 & 0 \\ -aY_{12} & Y_{22} & -Y_{23} & 0 & 0 \\ 0 & -Y_{23} & Y_{33} & -Y_{34} & 0 \\ 0 & 0 & -Y_{34} & Y_{44} & -Y_{45} \\ 0 & 0 & 0 & -Y_{45} & Y_{55} \end{bmatrix} \begin{bmatrix} \tilde{V}_1 \\ \tilde{V}_2 \\ \tilde{V}_3 \\ \tilde{V}_4 \\ \tilde{V}_5 \end{bmatrix} = \begin{bmatrix} \tilde{I}_1 \\ \tilde{I}_2 \\ \tilde{I}_3 \\ \tilde{I}_4 \\ \tilde{I}_5 \end{bmatrix} = \begin{bmatrix} \tilde{I}_1 \\ 0 \\ 0 \\ 0 \\ \tilde{I}_5 \end{bmatrix}$$

The direction of current injection is into the node (bus).
Rules for building the Y matrix:

1. If Buses i and j are not connected, then the (i, j) entry of Y is zero.
2. The (i, j) terms are the sum of admittances of all the direct connections between Buses i and j.
3. The self (i, i) terms of Y are the sum of the admittances of connections of Bus i to all the other buses, and all the shunt components representable as linear, passive circuit elements.

For example, the entry

$$Y_{22} = aY_{12} + Y_{23} + j\frac{B_{23}}{2}$$

Building the Y matrix: Y is built sequentially by adding one branch (transformer, transmission line) at a time, in the order of reading the branch input data file.
If a branch between Bus i and Bus j is read, then the (i,i), (i,j), and (j,j) entries of Y are updated.
Storing the (complex) Y matrix:

1. as a full $N_B \times N_B$ matrix, where N_B is the number of buses: not practical for large power networks
2. as a sparse matrix, in which each non-zero (i,j) entry of Y is stored as a three-tuple (i,j,y_{ij}): practical for large systems. A 1,000-bus system with an average of 4 connections per bus will result in a sparse matrix with approximately 5,000 non-zero entries, instead of a million entries. The 5-bus example has only 13 nonzero entries.

Using the network equation $Y\tilde{V} = \tilde{I}$:

- If the bus voltage $\tilde{V}$ at each of the bus is known, then $\tilde{I}$ can be directly computed.
  This is indeed the case in dynamic simulation when $\tilde{V}$ is the internal voltage of a generator. For non-source buses, the load is converted into an admittance (for a constant-impedance load) and folded into a diagonal entry of the overall admittance matrix. More on this aspect later when we discuss dynamic simulation.
- One cannot use $Y\tilde{V} = \tilde{I}$ directly for loadflow (steady-state dispatch) calculations, because loads are specified as $P + jQ = -\tilde{V}\tilde{I}^*$ (note negative sign as load is positive (power consumed) if $P > 0$ and $Q > 0$), resulting in

$$Y\tilde{V} = -\text{vec}\left(\frac{P_i + jQ_i}{\tilde{V}_i}\right)^* = -\text{vec}\frac{(P_i + jQ_i)^*}{\tilde{V}_i^*} = \tilde{I}$$

Note that the right-hand-side of the equation carries a negative sign, because P and Q are power consumptions by the load, and $\tilde{I}$ is current injection leaving the bus.

Loadflow Formulation

Loadflow calculates the bus voltage and line current phasors in a power network to satisfy a given set of generation, load, and voltage conditions.

Each bus is defined by 4 quantities:
Bus voltage magnitude and angle: $\tilde{V} = Ve^{j\theta}$ and
Bus generation and/or load: $P + jQ$ (or zero injection $P = Q = 0$)

3 main types of buses:

Bus type	V	θ	P	Q
PV (generator)	specified	computed	specified	computed
PQ (load bus)	computed	computed	specified	specified
Swing (slack) bus	specified	$\theta = 0°$	computed	computed

The phase of the swing bus is set to zero (or any fixed value), which is known as the reference.

Note that:

1. A swing bus is a generator bus whose active power generation accounts for the losses in a power network.
2. The angle of the swing bus is set to zero so that the angles of all the other buses are in reference to the swing bus.
3. The reactive power output Q of a generator bus is computed to support the specified voltage on the generator bus.
4. At a generator bus, if $Q > Q_{\max}$ or $Q < Q_{\min}$, then Q is set to the appropriate Q-limit Q_ℓ and the PV bus becomes a PQ bus with $Q = Q_\ell$.
5. Other types of buses such as HVDC (DC line) and voltage-source converters, need additional modeling capability.

Newton-Raphson (NR) Method

The NR method is an iterative method for solving a set of nonlinear equations

$$f(x) = b$$

where $x \in R^N$ is the vector of unknowns, $f \in R^N$ is the nonlinear vector function, and $b \in R^N$ is a constant vector:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, \quad f = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$$

That is, this is a system of N nonlinear equations in N unknowns.

Suppose we have an initial guess $x^{(0)}$ and the true solution is $x^{(0)} + \Delta x$, where

$$\Delta x = \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_N \end{bmatrix}$$

We consider Δx as a perturbation and expand $f(x)$ around $x^{(0)}$

$$f(x^{(0)} + \Delta x) = b = f(x^{(0)}) + \left.\frac{\partial f}{\partial x}\right|_{x=x^{(0)}} \Delta x + \frac{1}{2}\Delta x^T F|_{x=x^{(0)}} \Delta x$$

$$+ \text{ higher order terms}$$

where

$$\frac{\partial f}{\partial x} = J = \begin{bmatrix} J_{11} & J_{12} & \cdots & J_{1N} \\ \vdots & \vdots & \vdots & \vdots \\ J_{N1} & J_{N2} & \cdots & J_{NN} \end{bmatrix}, \quad J_{ij} = \frac{\partial f_i}{\partial x_j}$$

is the first derivative of f with respect to x and is known as the Jacobian, and F is the second derivative.

If Δx_1, Δx_2, ..., Δx_N are small, then $\Delta x_i \Delta x_j$ and higher order terms are negligible.

Thus

$$b = f(x^{(0)}) + J(x^{(0)})\Delta x \quad \Rightarrow \quad J\Delta x = b - f(x^{(0)}) = e(x^{(0)})$$

where e is the error.
For numerical stability, it is not recommended to solve for $\Delta x = J^{-1}e(x^{(0)})$ by computing J^{-1}, the inverse of J. In addition J^{-1} is in general a full matrix, even though J is sparse, that is, with many zero entries.

Instead, the proper way is to first to perform an LU decomposition of $J = LU$, where L is a lower triangular matrix, and U is an upper triangular matrix. Furthermore, the diagonal elements of L can be set to unity.

More on LU decomposition and sparse factorization later.

The update equation

$$LU\Delta x = L\Delta y = e(x^{(0)})$$

is solved in 2 stages:

1. Forward elimination: solve for Δy from $L\Delta y = e(x^{(0)})$
2. Back substitution: solve for Δx from $U\Delta x = \Delta y$

At iteration $k+1$, update the solution as

$$x^{(k+1)} = x^{(k)} + \Delta x$$

where Δx is the latest update. Continue the iteration until

$$|x^{(k+1)} - x^{(k)}| < \delta$$

where δ is a specified tolerance.

Near the solution point, the convergence of the NR method is quadratic.

Sometimes if the initial condition is poorly selected, then NR method may not converge.

Normally one can select sensible initial values.

NR Solution of Power Flow Equation

First we need to write the nonlinear power flow equations in terms of the unknown voltage variables:

PQ bus	$V_i e^{j\theta_i}$	2 unknowns: V_i, θ_i
PV bus	$V_i e^{j\theta_i}$	1 unknown: θ_i
Swing bus	$V_N e^{j0^\circ}$	no unknowns

Total number of unknowns

$$(N-1) + (N - N_g) = 2N - N_g - 1$$

where N is the total number of buses, and N_g is the total number of generator buses.

For each bus i, except for the swing bus, the power injection from the node into the system is

$$S_i = P_i + jQ_i = \tilde{V}_i \tilde{I}_i^*$$

From $Y\tilde{V} = \tilde{I}$, we obtain

$$\sum_{j=1}^{N} Y_{ij}\tilde{V}_j = \tilde{I}_i \quad \Rightarrow \quad S_i = \tilde{V}_i \left(\sum_{j=1}^{N} Y_{ij}^* \tilde{V}_j^* \right)$$

Thus

$$P_i + jQ_i = V_i e^{j\theta_i} \left(\sum_{j=1}^{N} |Y_{ij}| e^{-j\alpha_{ij}} V_j e^{-j\theta_j} \right) = V_i \left(\sum_{j=1}^{N} |Y_{ij}| V_j e^{j(\theta_i - \theta_j - \alpha_{ij})} \right)$$

that is,

$$P_i = V_i \sum_{j=1}^{N} |Y_{ij}| V_j \cos(\theta_i - \theta_j - \alpha_{ij}), \quad i = 1, 2, ..., N-1$$

$$Q_i = V_i \sum_{j=1}^{N} |Y_{ij}| V_j \sin(\theta_i - \theta_j - \alpha_{ij}), \quad i = 1, 2, ..., N-N_g$$

which are specified.

NR iteration: the unknown vector is

$$x = \left[\theta_1 \ \theta_2 \ \cdots \ \theta_{N-1} \ V_1 \ V_2 \ \cdots \ V_{N-N_g}\right]^T$$

with an initial value $x^{(0)}$.
Cold start: set all voltage magnitude values equal to 1, and all angle values equal to zero.
Warm start: use an existing solution to a problem with similar load conditions.

At iteration $k+1$, compute

$$x^{(k+1)} = x^{(k)} + \Delta x$$

where Δx is solved from

$$J(x^{(k)})\Delta x = e(x^{(k)})$$

where there are $2N - N_g - 1$ rows in both

$$e(x) = \begin{bmatrix} P_1 - V_1 \sum_{j=1}^{N} |Y_{ij}| V_j \cos(\theta_i - \theta_j - \alpha_{ij}) \\ P_2 - V_2 \sum_{j=1}^{N} |Y_{ij}| V_j \cos(\theta_i - \theta_j - \alpha_{ij}) \\ \vdots \\ Q_1 - V_1 \sum_{j=1}^{N} |Y_{ij}| V_j \sin(\theta_i - \theta_j - \alpha_{ij}) \\ \vdots \end{bmatrix}$$

$$J(x) = \begin{bmatrix} \frac{\partial P_1}{\partial \theta_1} & \frac{\partial P_1}{\partial \theta_2} & \cdots & \frac{\partial P_1}{\partial \theta_{N-1}} & \frac{\partial P_1}{\partial V_1} & \cdots & \frac{\partial P_1}{\partial V_{N-N_g}} \\ \frac{\partial P_2}{\partial \theta_1} & \frac{\partial P_2}{\partial \theta_2} & \cdots & \frac{\partial P_2}{\partial \theta_{N-1}} & \frac{\partial P_2}{\partial V_1} & \cdots & \frac{\partial P_2}{\partial V_{N-N_g}} \\ & & & \vdots & & & \\ \frac{\partial Q_1}{\partial \theta_1} & \frac{\partial Q_1}{\partial \theta_2} & \cdots & \frac{\partial Q_1}{\partial \theta_{N-1}} & \frac{\partial Q_1}{\partial V_1} & \cdots & \frac{\partial Q_1}{\partial V_{N-N_g}} \\ & & & \vdots & & & \end{bmatrix}$$

where

$$\frac{\partial P_1}{\partial \theta_1} = -V_1 \sum_{j=1}^{N} |Y_{1j}| V_j \sin(\theta_1 - \theta_j - \alpha_{1j})$$

$$\frac{\partial P_1}{\partial \theta_2} = -V_1 |Y_{12}| V_2 \sin(\theta_1 - \theta_2 - \alpha_{12})$$

$$\frac{\partial P_1}{\partial V_1} = \sum_{j=1}^{N} |Y_{1j}| V_j \cos(\theta_1 - \theta_j - \alpha_{1j}) - V_1 |Y_{11}| \cos(\alpha_{11})$$

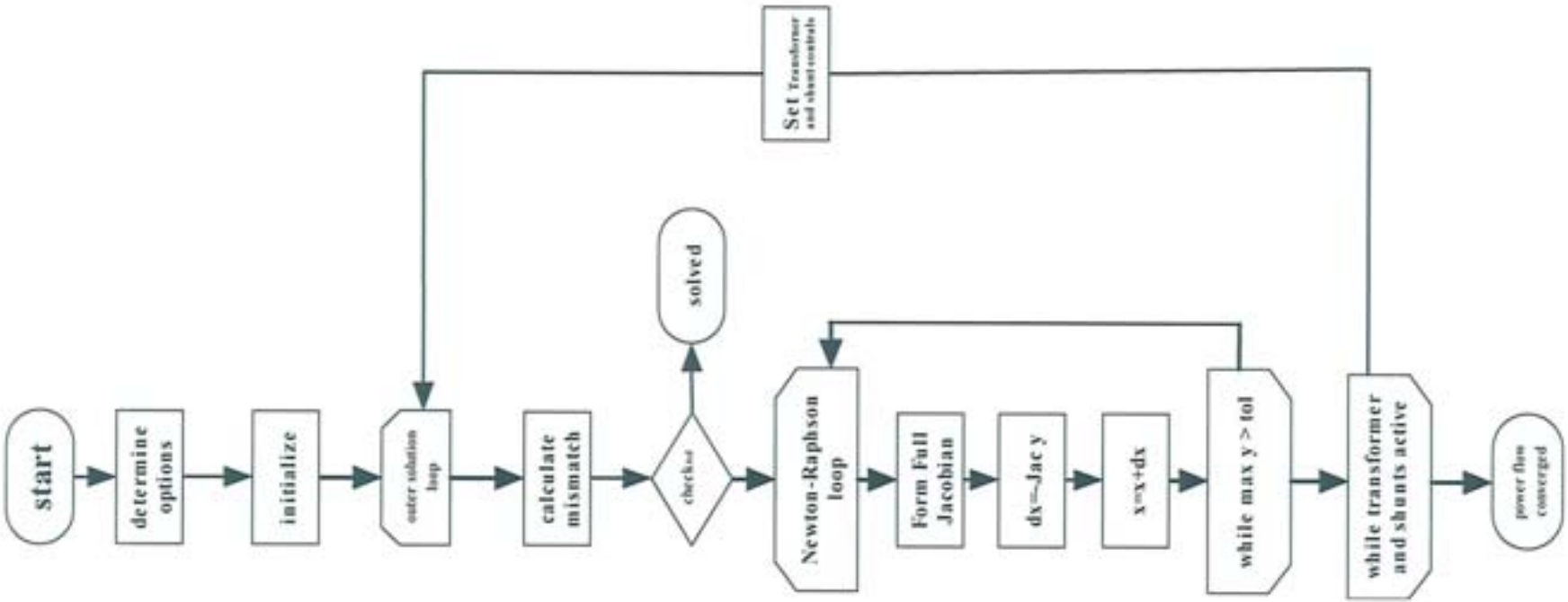
$$\frac{\partial P_1}{\partial V_2} = V_1 |Y_{12}| \sin(\theta_1 - \theta_2 - \alpha_{12})$$

Similar expressions can be obtained for the partial derivatives of Q_i.

Some general observations on the NR method:

1. The NR method is efficient. Convergence is quadratic near the solution.
2. The NR method allows a user to decelerate the algorithm by using a stepsize $\gamma < 1$, such that $x^{(k+1)} = x^{(k)} + \gamma \Delta x$, mostly to prevent divergence at the initial iterations.
3. Most loadflow programs display a squared error mismatch $(e(x^{(p)}))^T e(x^{(p)})$ where e is in pu. For example, the error history may look like 1×10^{-2}, 2.5×10^{-4}, 1.2×10^{-8}, etc.
4. If the error history looks like 1×10^{-2}, 2.5×10^{-3}, 1.2×10^{-4}, etc, the solution seems to be converging, but the Jacobian may not be correct. Also such less-than-quadratic convergence rate is to be expected with dishonest NR or decoupled power flow methods (explanations later).
5. Q limits at generator buses: if Q is outside the specified range, set Q equal to the upper or lower limit and change the generator bus to a PQ bus. Then resolve the loadflow.

Flow chart for loadflow solution



Dishonest NR Method

In the iterative solution of

$$J^{(k)} \begin{bmatrix} \Delta\theta^{(k)} \\ \Delta V^{(k)} \end{bmatrix} = \begin{bmatrix} \Delta P^{(k)} \\ \Delta Q^{(k)} \end{bmatrix} = e(x^{(k)})$$

where $e(x)$ is the mismatch vector, instead of updating J at every iteration, J is updated every two or three iterations. However, the mismatch $e(x)$ is updated at every iteration. Note that the LU decomposition of J can be saved, and thus no refactorization is needed.

Fast Decoupled Loadflow (FDLF)

Consider the NR method

$$J^{(k)} \begin{bmatrix} \Delta\theta^{(k)} \\ \Delta V^{(k)} \end{bmatrix} = \begin{bmatrix} \frac{\partial P^{(k)}}{\partial \theta} & \frac{\partial P^{(k)}}{\partial V} \\ \frac{\partial Q^{(k)}}{\partial \theta} & \frac{\partial Q^{(k)}}{\partial V} \end{bmatrix} \begin{bmatrix} \Delta\theta^{(k)} \\ \Delta V^{(k)} \end{bmatrix} = \begin{bmatrix} \Delta P^{(k)} \\ \Delta Q^{(k)} \end{bmatrix}$$

Because P is largely determined by θ and Q is largely determined by V, J is block-diagonally dominant. Thus neglecting the off-diagonal blocks of J results in

$$\begin{bmatrix} \frac{\partial P^{(k)}}{\partial \theta} & 0 \\ 0 & \frac{\partial Q^{(k)}}{\partial V} \end{bmatrix} \begin{bmatrix} \Delta\theta^{(k)} \\ \Delta V^{(k)} \end{bmatrix} = \begin{bmatrix} \Delta P^{(k)} \\ \Delta Q^{(k)} \end{bmatrix}$$

Note that even though the search direction

$$\frac{\partial P^{(k)}}{\partial \theta}\Delta\theta^{(k)} = \Delta P^{(k)}, \quad \frac{\partial Q^{(k)}}{\partial V}\Delta V^{(k)} = \Delta Q^{(k)}$$

may not be as good as the fully coupled NR method, FDLP will still converge to the right solution, because the mismatch is computed correctly. FDLF will achieve a quadratic convergence rate, as long as the off-diagonal blocks are sufficiently small. Also, FDLF works well when $R/X < 0.1$.

Further simplification:
From the $\partial P/\partial\theta$ and $\partial Q/V$ expressions, if we neglect line resistances and assume $|\theta_i - \theta_j|$ small ($\sin(\theta_i - \theta_j) = 0$ and $\cos(\theta_i - \theta_j) = 1$), then

$$\frac{\partial P}{\partial \theta} = -\text{diag}(V) \cdot B \cdot \text{diag}(V), \quad \frac{\partial Q}{\partial V} = -\text{diag}(V) \cdot B$$

where $B = \text{Im}(Y)$ is the susceptance matrix.

Thus we solve for

$$-B \cdot \text{diag}(V)\Delta\theta = (\text{diag}(V))^{-1}\Delta P, \quad -B\Delta V = (\text{diag}(V))^{-1}\Delta Q$$

for which the LU decomposition for B has to be computed only once. Convergence of this method is linear, but each iteration is very fast.

DC loadflow

In a DC loadflow, we assume that

1. all the bus voltage magnitudes equal to unity
2. line resistances are neglected
3. $|\theta_i - \theta_j|$ is small

Then the loadflow equation simplifies to

$$-B \cdot \theta = P$$

DC loadflow has 2 major uses:

1. to initialize solution for the NR method
2. used in unit commitment program in which the generators are assumed to be able to provide adequate reactive power support

Power Flow Data for PST

```
% npcc 48 machine system data
% loadflow data
% bus data % col 1 -- bus number
% col 2 -- bus voltage V
% col 3 -- bus angle in degrees
% col 4, 5 -- active and reactive power generation pu
% col 6, 7 -- active and reactive power load pu
% col 8, 9 -- G and B shunt pu
% col 10 -- bus type:  1 is swing, 2 is PV, 3 is PQ
bus = [ ...
 1 1.0000 0.     0.000   0.000    0.0   0.00 0.  0.    3;
20 1.0418 0.     0.000   0.000  2.840   0.27 0.  0.    3;
21 1.0000 0.     6.500   2.165    0.0   0.00 0.  0.    2;
45 1.0409 0.     0.000   0.000  2.560   0.42 0.  0.25  3;
78 1.0200 0.    46.321   0.719 60.000   6.00 0.  0.    1 ]
```

Power Flow Data for PST

```
% line data
% col 1 -- from bus
% col 2 -- to bus
% col 3 -- R pu
% col 4 -- X pu
% col 5 -- B pu % col 6 -- tap ratio
% col 7 -- phase shifter angle deg
line = [...
1 21 0.      0.0200 0.     1.07 0;
2  1 0.0004 0.0043 0.07   0.   0;
2 33 0.0007 0.0082 0.14   0.   0;
3  2 0.0016 0.0435 0.     1.06 0;
3  4 0.0016 0.0435 0.     1.06 0]
```

Power Flow Function in PST

```
loadflow(bus,line,tol,iter_max,vmin,vmax,acc,display,flag)
% Syntax:  [bus_sol,line_flow] =
% loadflow(bus,line,tol,iter_max,vmin,vmax,acc,display,flag)
% Purpose:  solve the load-flow equations of power systems
% Input:  bus - bus data
%         line - line data
%         tol - tolerance for convergence
%         iter_max - maximum number of iterations
%         vmin - voltage minimum limit
%         vmax - voltage maximum limit
%         acc - acceleration factor
%         display - 'y', generate load-flow study report
%                   else, no load-flow study report
%         flag - 1, form new Jacobian every iteration
%                2, form new Jacobian every other iteration
% Output:  bus_sol - bus solution
%          line_flow - line flow solution
```

Power Flow Solution in PST

```
mismatch is 4.34711.
mismatch is 2.13662.
mismatch is 0.495287.
mismatch is 0.00246548.
mismatch is 0.000033704.
mismatch is 8.84984e-012.
                    LOAD-FLOW STUDY
              REPORT OF POWER FLOW CALCULATIONS
13-Jul-94
SWING BUS                     : BUS 50
NUMBER OF ITERATIONS          : 5
SOLUTION TIME                 : 1.92 sec.
TOTAL TIME                    : 5.44 sec.
TOTAL REAL POWER LOSSES       : 6.59692 pu.
TOTAL REACTIVE POWER LOSSES: 45.2356 pu.
```

Power Flow Solution - Bus Voltages

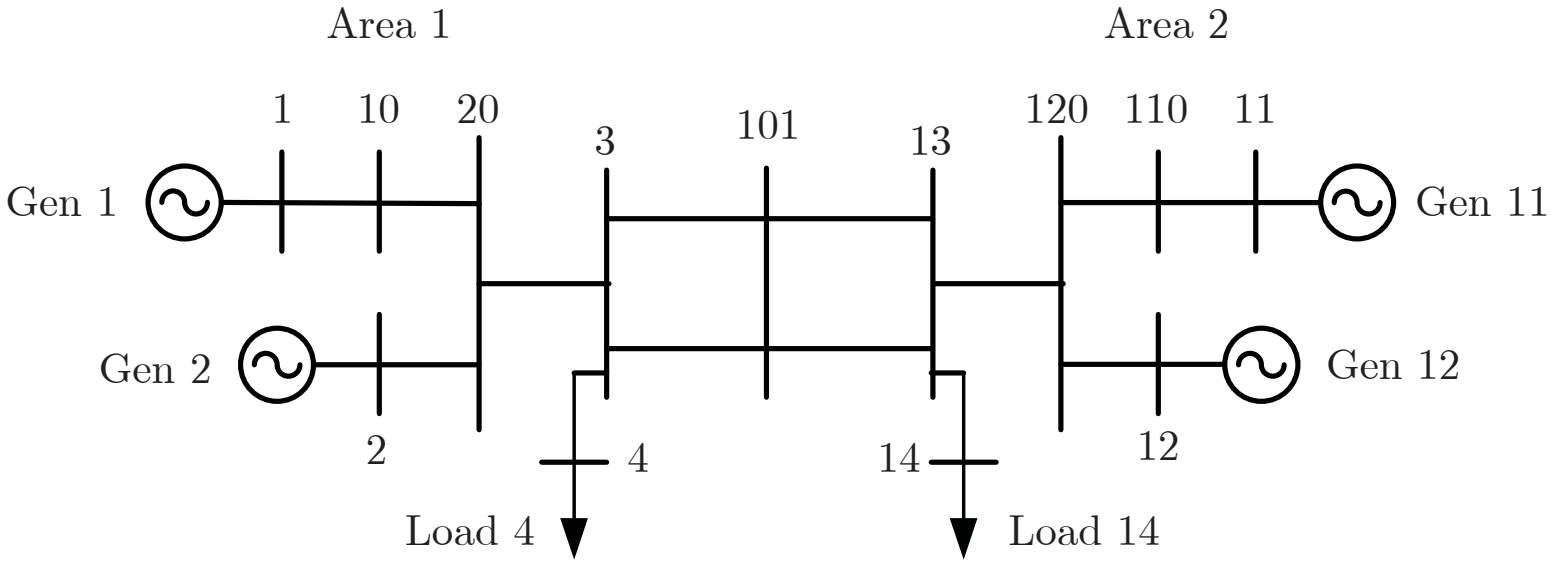
			GENERATION		LOAD	
BUS	VOLTS	ANGLE	REAL	REACTIVE	REAL	REACTIVE
10.0000	0.9988	24.4449	0	0	0.0000	0.0000
11.0000	0.9940	23.7896	0	0	0.0000	0.0000
12.0000	1.0326	23.4777	0	0	0.0900	0.8800
30.0000	0.9800	33.6919	7.3500	2.8746	0	0
31.0000	0.9900	29.5526	6.9500	1.6171	0	0
32.0000	1.0066	27.6191	5.1400	1.9541	0	0
50.0000	1.0500	8.6500	-17.824	7.2011	0	0
123.0000	1.0000	79.0885	6.0000	0.9563	7.5000	2.0000
124.0000	1.0200	77.0888	7.7000	0.3405	9.4600	-0.4000

Power Flow Solution - Line Flows

LINE FLOWS

LINE	FROM BUS	TO BUS	REAL	REACTIVE
1.0000	10.0000	30.0000	-7.3500	-1.5775
1.0000	30.0000	10.0000	7.3500	2.8746
2.0000	11.0000	10.0000	-2.7184	-0.8690
2.0000	10.0000	11.0000	2.7217	0.8347

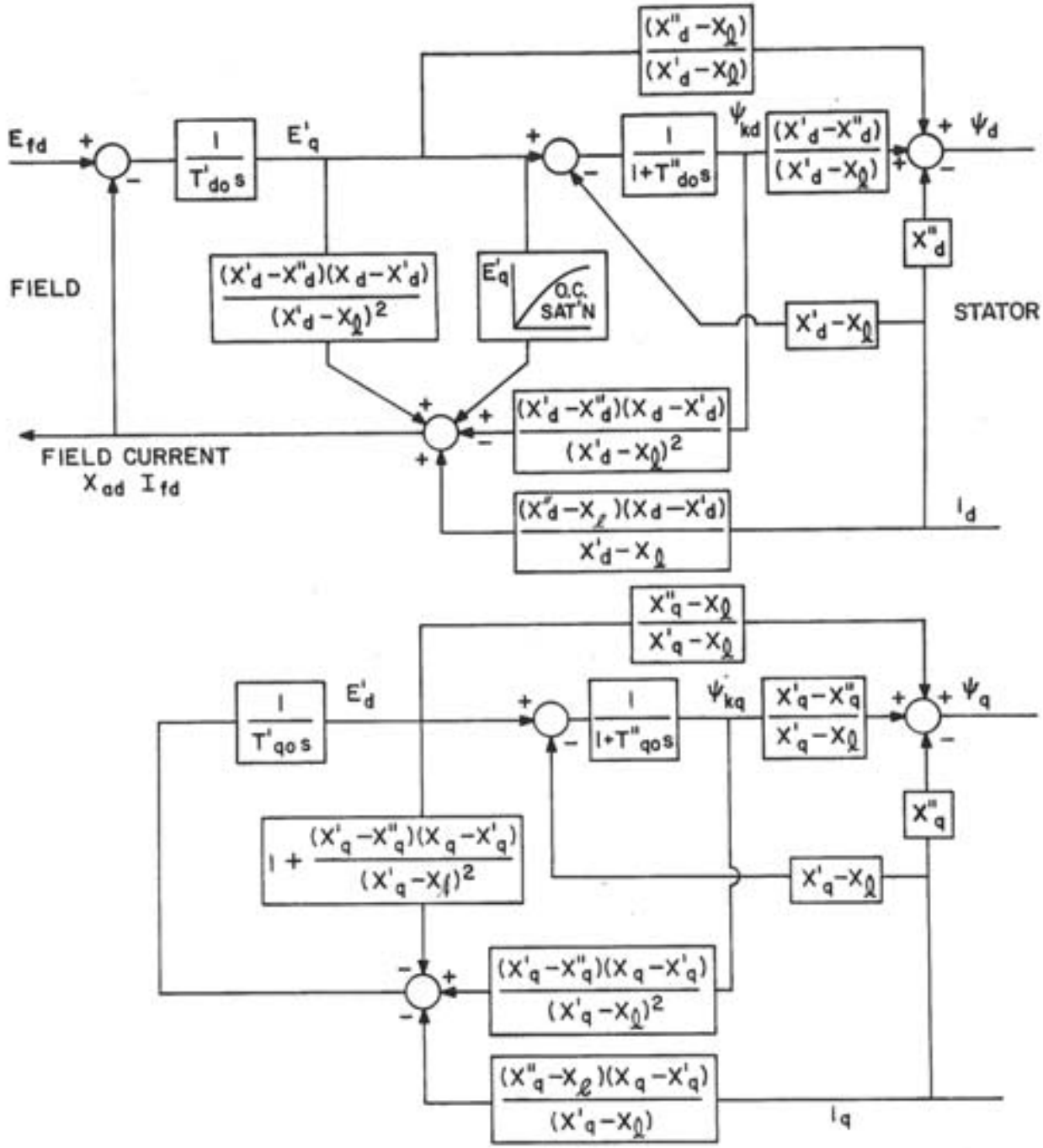
Dynamic Simulation



1. Starts with a solution of the power network in steady state
2. Apply the appropriate dynamic models for the synchronous machines, excitation systems, turbine governors, and power system stabilizers, and other control equipment
3. Initialize the dynamic models from the power flow solution
4. Apply a disturbance to the system
5. Simulate the system dynamics using an appropriate integration routine until either the trajectories converge to a (new) steady state or become unstable

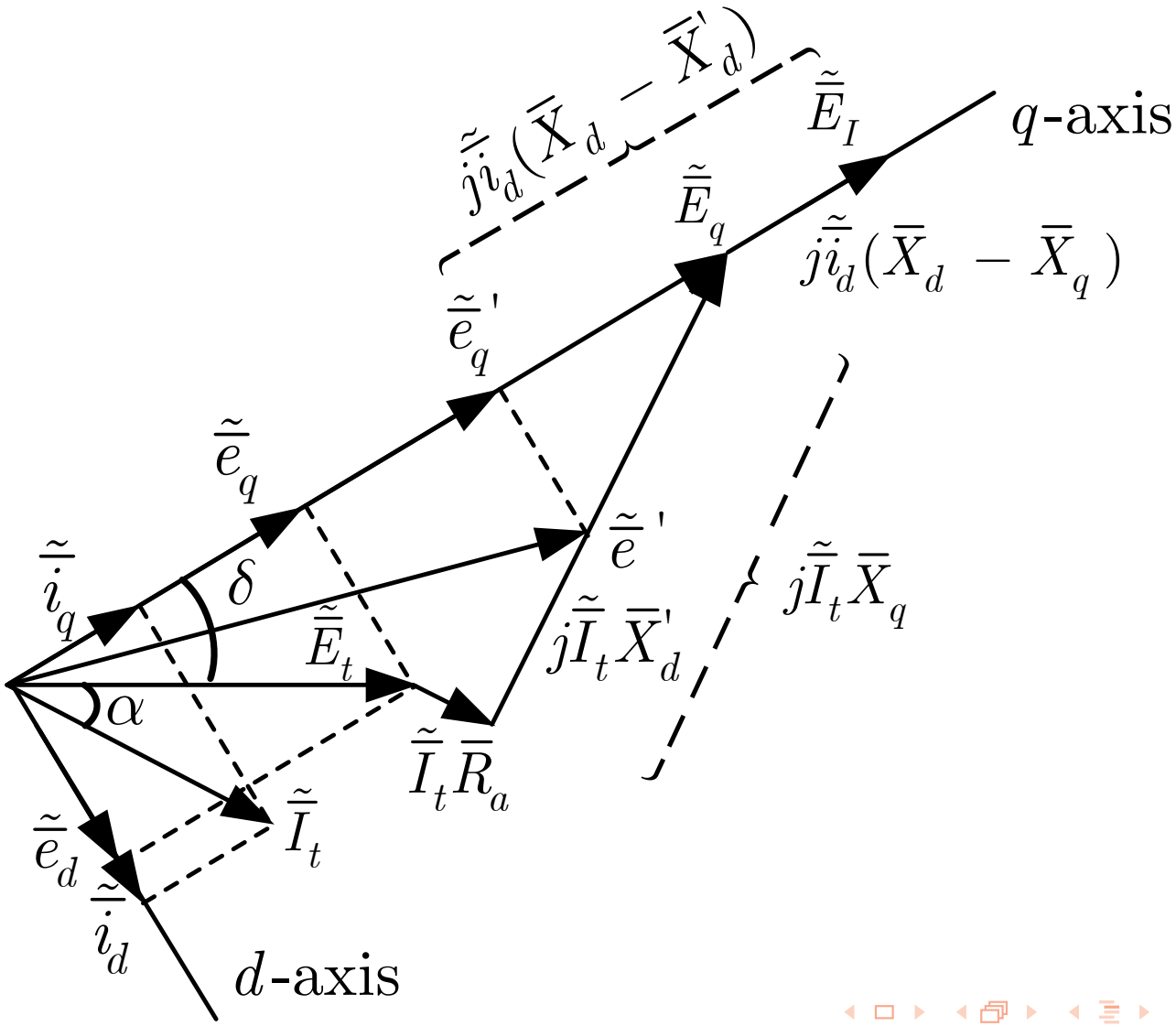
Synchronous Machine Dynamic Models

- Electromechanical model - 2 states, δ, ω
- Transient model - 4 states, δ, ω, E'_q, E'_d
- Subtransient model - 6 states, δ, ω, E'_q, E'_d, ψ_{kd}, ψ_{kq} (shown on next page)


E_{fd}
$\frac{1}{T'_{do}s}$
E'_q
$\frac{X''_d - X_\ell}{X'_d - X_\ell}$
$\frac{1}{1+T''_{do}s}$
ψ_{kd}
$\frac{X'_d - X''_d}{X'_d - X_\ell}$
ψ_d
X''_d
FIELD
$\frac{(X'_d - X''_d)(X_d - X'_d)}{(X'_d - X_\ell)^2}$
E'_q O.C. SAT'N
$X'_d - X_\ell$
STATOR
FIELD CURRENT
$X_{ad}\ I_{fd}$
$\frac{(X''_d - X_\ell)(X_d - X'_d)}{X'_d - X_\ell}$
i_d
$\frac{X''_q - X_\ell}{X'_q - X_\ell}$
$\frac{1}{T'_{qo}s}$
E'_d
$\frac{1}{1+T''_{qo}s}$
ψ_{kq}
$\frac{X'_q - X''_q}{X'_q - X_\ell}$
ψ_q
X''_q
$1 + \frac{(X'_q - X''_q)(X_q - X'_q)}{(X'_q - X_\ell)^2}$
$X'_q - X_\ell$
$\frac{(X'_q - X''_q)(X_q - X'_q)}{(X'_q - X_\ell)^2}$
$\frac{(X''_q - X_\ell)(X_q - X'_q)}{(X'_q - X_\ell)}$
i_q

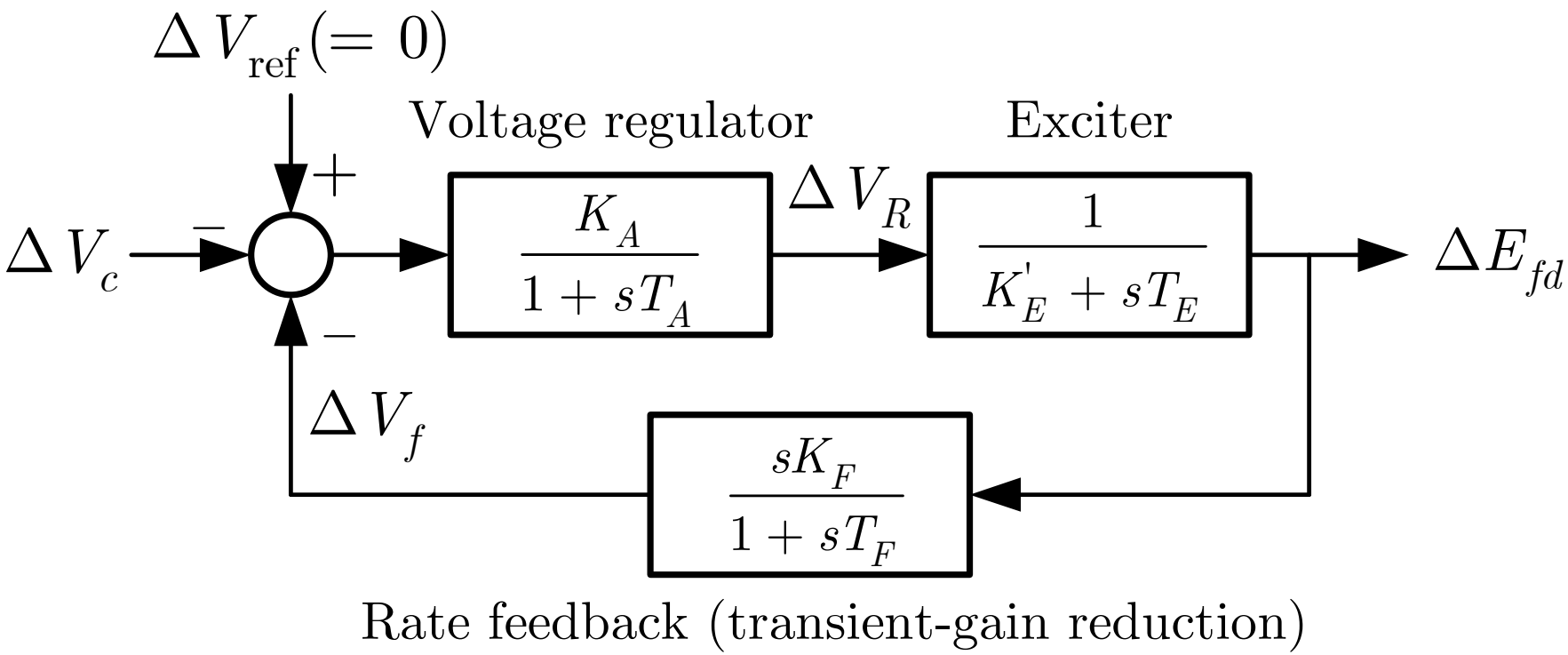
Synchronous Machine Vector Diagram

- For initialization



Excitation System Dynamic Models

- Type DC 1A



Rate feedback (transient-gain reduction)

- Type ST 1A

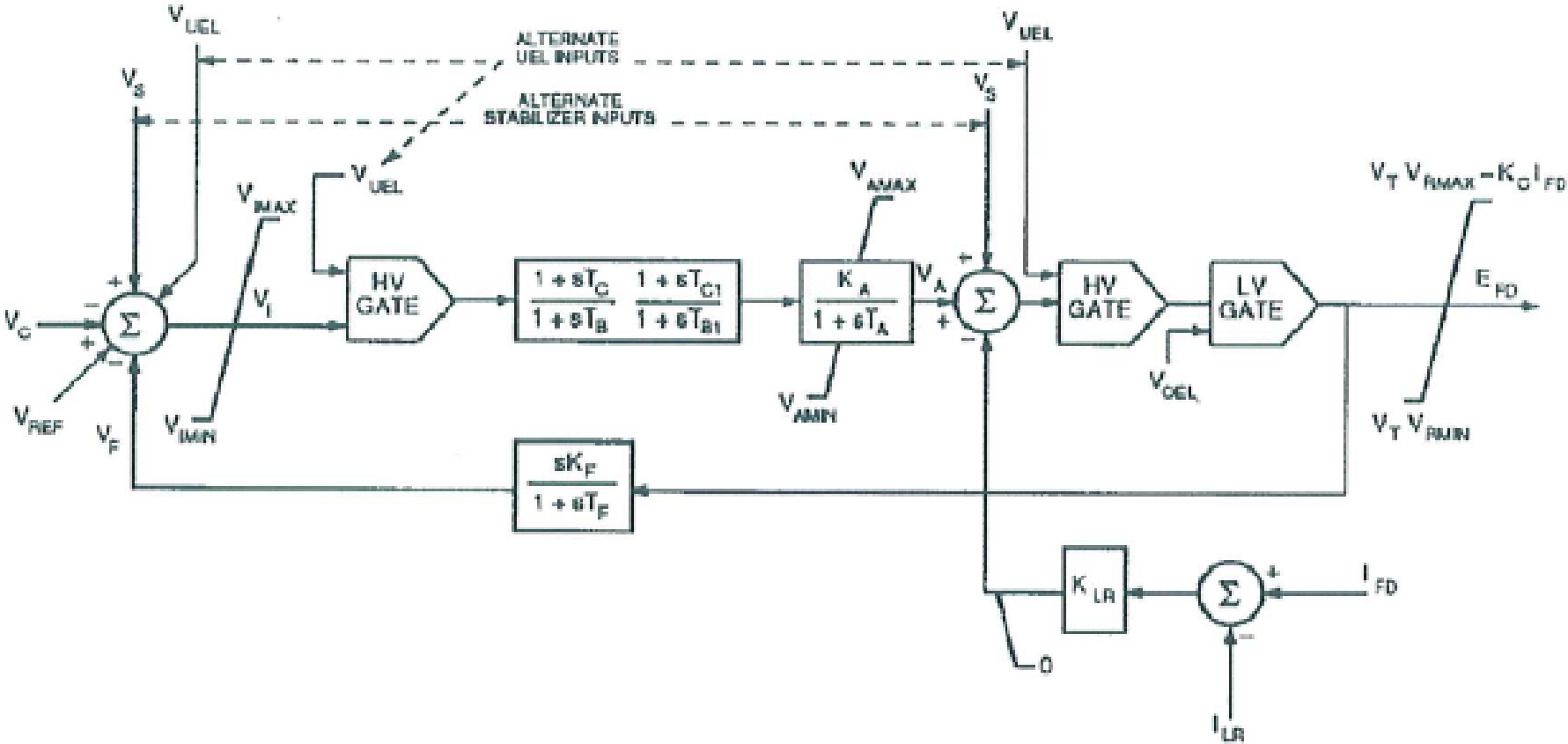
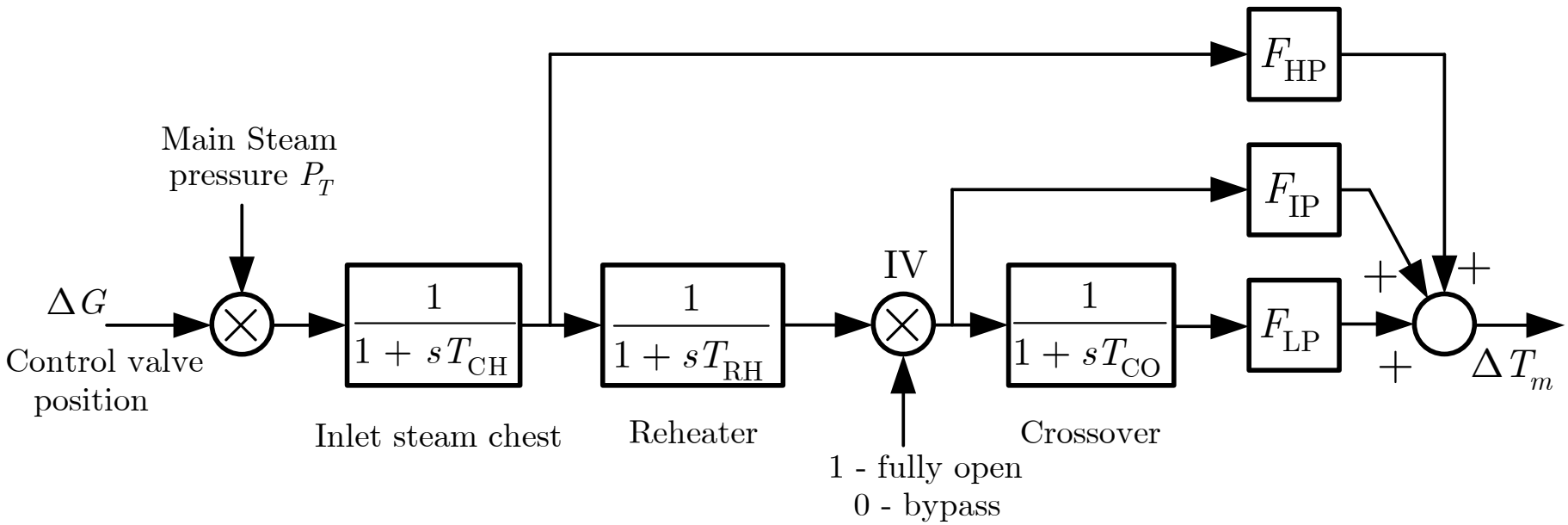


Figure 12—Type ST1A — Potential-Source Controlled-Rectifier Exciter

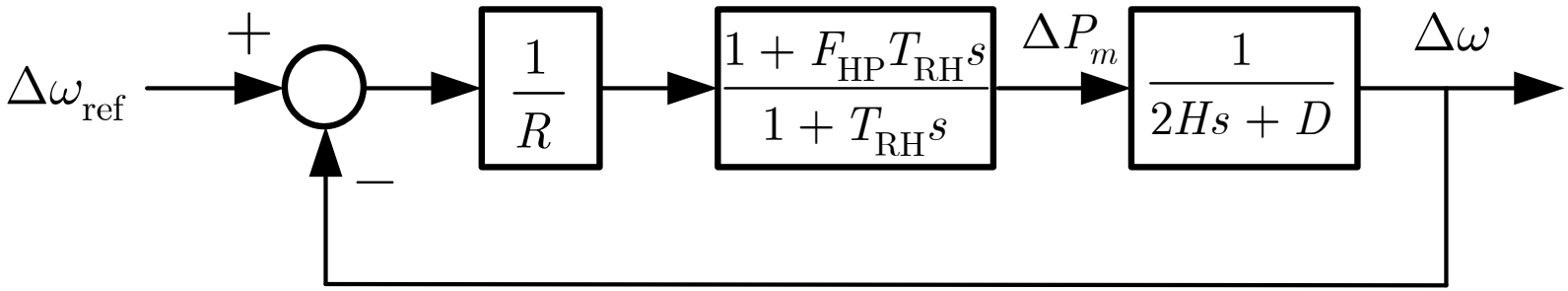
Turbine Dynamic Model

- Steam turbine model with reheat



Governor Dynamic Model

- Governor model for steam turbine model



Power System Dynamic Simulation

- Step-by-step integration of power system dynamic models represented as first-order differential equations (DE)
- The second-order swing equation

$$\frac{d^2\delta}{dt^2} = \frac{\Omega}{2H}(P_m - P_e)$$

is written as a system of 2 first-order DEs

$$\frac{d\delta}{dt} = \Omega\omega, \quad \frac{d\omega}{dt} = \frac{1}{2H}\left(P_m - \frac{E'V\sin\delta}{X_{\text{eq}}}\right)$$

where $\Omega = 2\pi f_o$ ($\Omega = 377$ for a 60 Hz system and 314 for a 50 Hz system). Units: δ is in radians and ω is in pu.

- In general, a power system consisting of an electrical network, generators, and controls (such as voltage regulators) can be modeled as a system of first-order equations

$$\dot{x} = f(x, u, t), \quad x, f \in R^n, \ u \in R^m$$

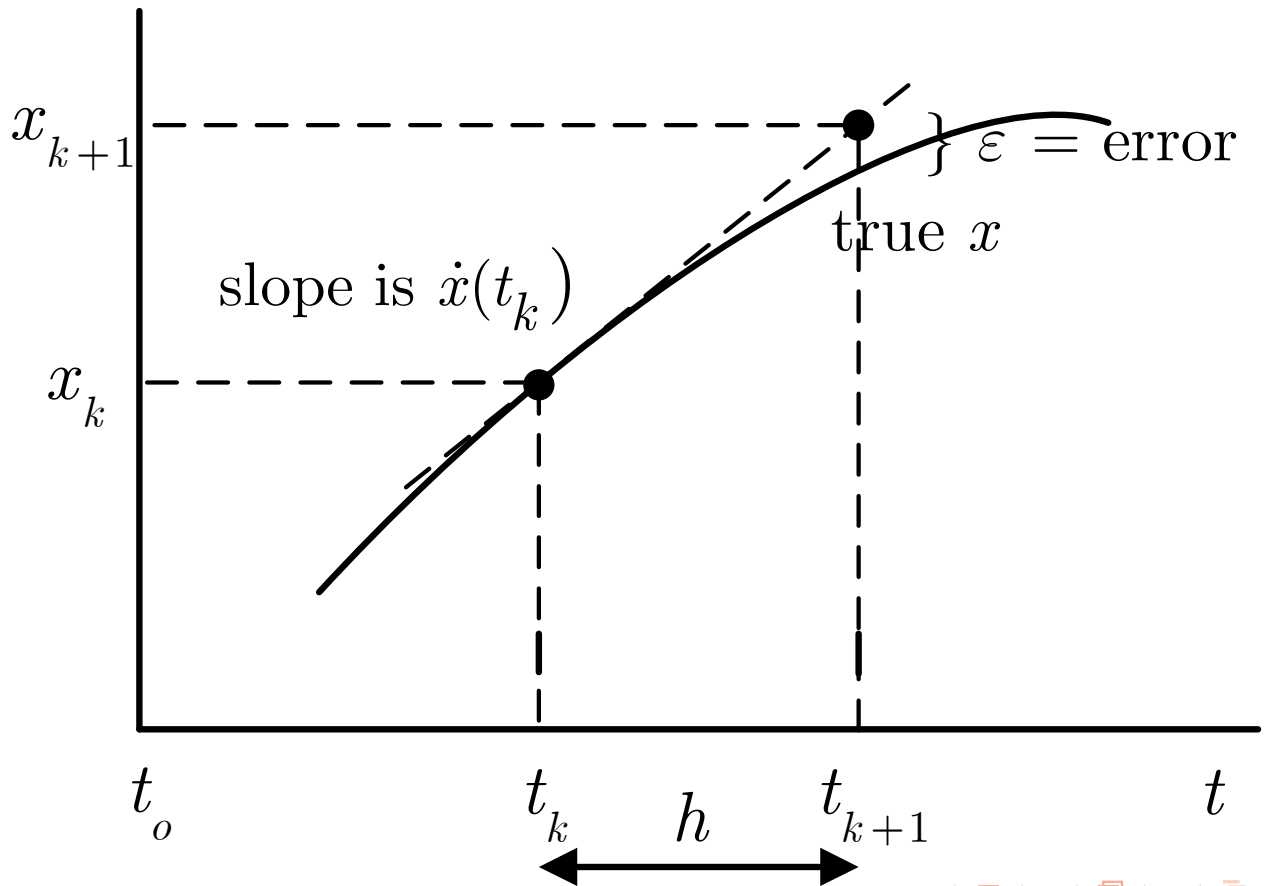
where x is the state vector of dimension n and u is the input (like P_m) vector of dimension m.

Reference: C. W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, 1971.

Euler's Method Integrate a nonlinear DE from $t = t_o$ to $t = t_f$ using fixed stepsize h.

Suppose the value of x at $t = t_k$ is $x(t_k) = x_k$. Then

$$x(t_{k+1}) = x(t_k) + h \cdot \dot{x}(t_k), \quad \text{that is,} \quad x_{k+1} = x_k + h \cdot \dot{x}_k$$



Taylor Series Expansion to $(n+1)$st Order

$$x_{k+1} = x_k + h\dot{x}_k + \frac{h^2}{2}\ddot{x}_k + \cdots + \frac{h^n}{n!}x_k^{(n)} + \frac{h^{n+1}}{(n+1)!}x_k^{(n+1)}(\xi)$$

where $t_k \leq \xi \leq t_{k+1}$. (Recall the midpoint theorem from advanced calculus. This is a higher order version.)

If $\dot{x} = f(x, u, t)$, then

$$\ddot{x} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\dot{x} + \frac{\partial f}{\partial u}\dot{u}$$

and so on.
If one only takes the first n terms, then the $(n+1)$st term becomes the local truncation error ε.

Runge-Kutta (RK) Methods for Integration of Differential Equations

- One-step method: to find x_{k+1}, one only needs information at the preceding point (x_k, t_k, u_k).
- Order n: agrees with Taylor series through h^n.
- Evaluation of $f(x, u, t)$ only: does not require any derivatives of $f(x, u, t)$.

Euler's method is a first-order RK. The local truncation error is

$$\varepsilon = \frac{h^2}{2}\ddot{x}(\xi) \approx \frac{h^2}{2}\ddot{x}_k \approx \frac{h^2}{2}\frac{(\dot{x}_{k+1} - \dot{x}_k)}{h} = \frac{h}{2}(\dot{x}_{k+1} - \dot{x}_k)$$

To keep the error small, one needs a small stepsize h. For power system dynamic simulation, we want to see about 100 points per second, taking into account time constants and control actions. For Euler's method a stepsize of $h = 10$ ms may not provide an accurate solution. One can improve on the accuracy by modifying the Euler's method.

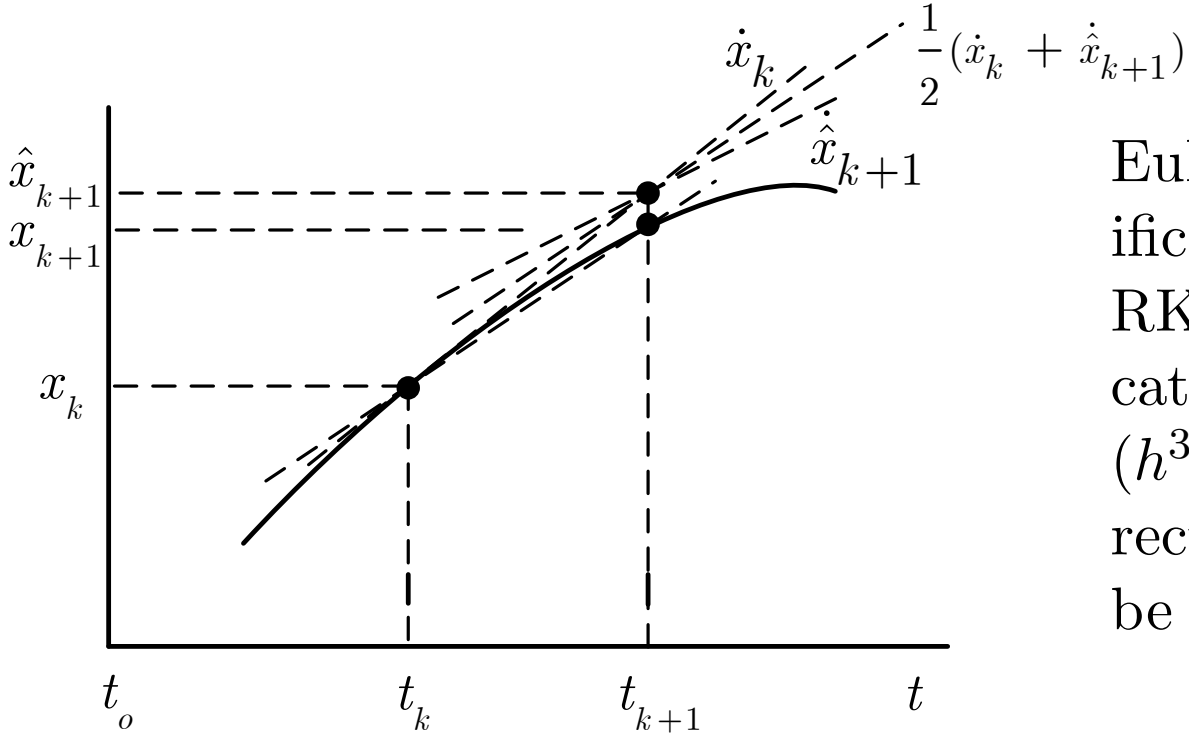
Euler Full-Step Modification: used in PST

Predictor:

$$\hat{x}_{k+1} = x_k + h \cdot \dot{x}_k = x_k + h \cdot f(x_k, u_k, t_k)$$

Corrector:

$$\dot{\hat{x}}_{k+1} = f(\hat{x}_{k+1}, u_{k+1}, t_{k+1}), \quad x_{k+1} = x_k + \frac{h}{2}(\dot{x}_k + \dot{\hat{x}}_{k+1})$$



Euler full-step modification is 2nd-order RK, as the local truncation error $\varepsilon \approx (h^3/6)\dddot{x}_k$. The corrector equation can be used repeatedly.

Adams-Bashforth Second-Order Method (1883)

The Siemens-PTI PSS/E program and the GE PSLF program both use the Adams-Bashforth Second-Order (AB2) Method to perform numerical integration. Given a system of nonlinear different equations with initial condition x_0

$$\dot{x}(t) = f(x(t)), \quad x(t_0) = x_0$$

the AB2 method performs the integration as

$$x_{k+1} = x_k + h[1.5f(x_k) - 0.5f(x_{k-1})]$$

where h is the stepsize. The Euler method is used to go from x_0 to x_1. Such a method is known as an explicit multi-step method because (1) there is no corrector part, and (2) the new value depends on more than just the previous value. The AB2 method depends on two previous values $f(x_k)$ and $f(x_{k-1})$.

Stability boundaries of integration methods

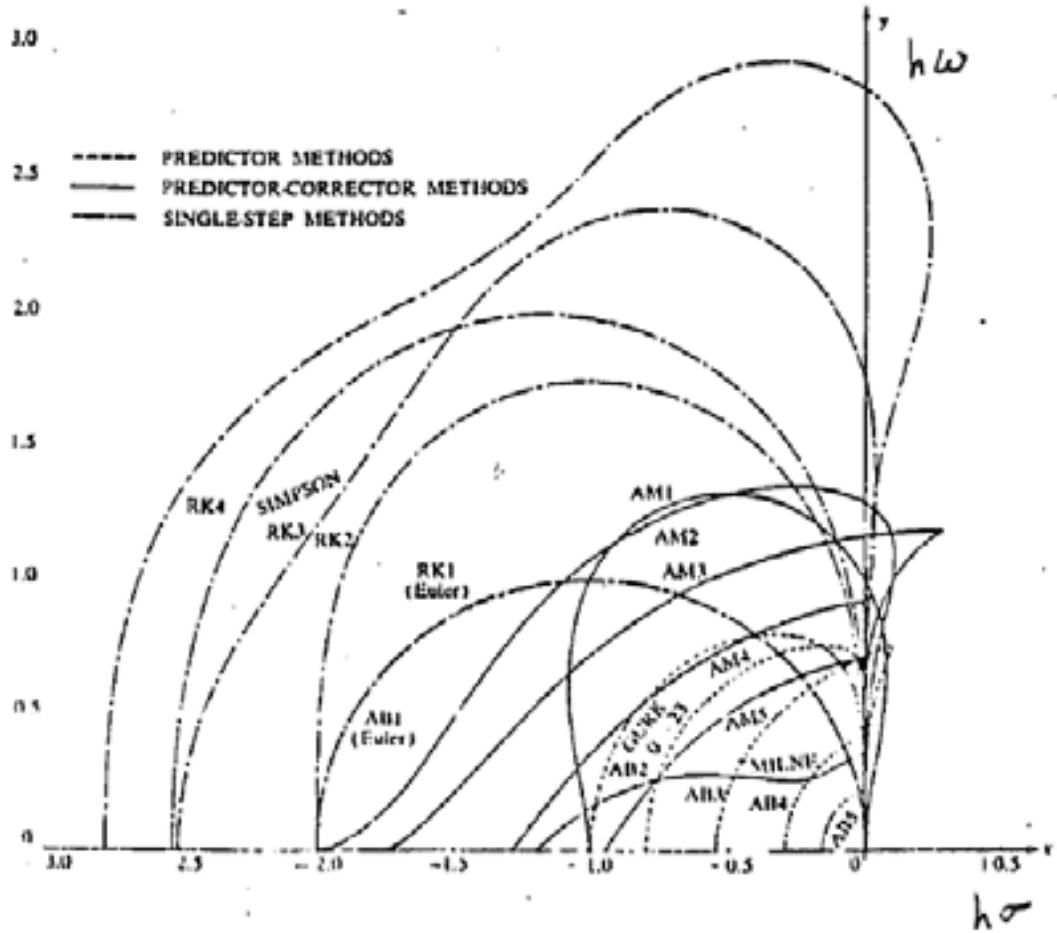
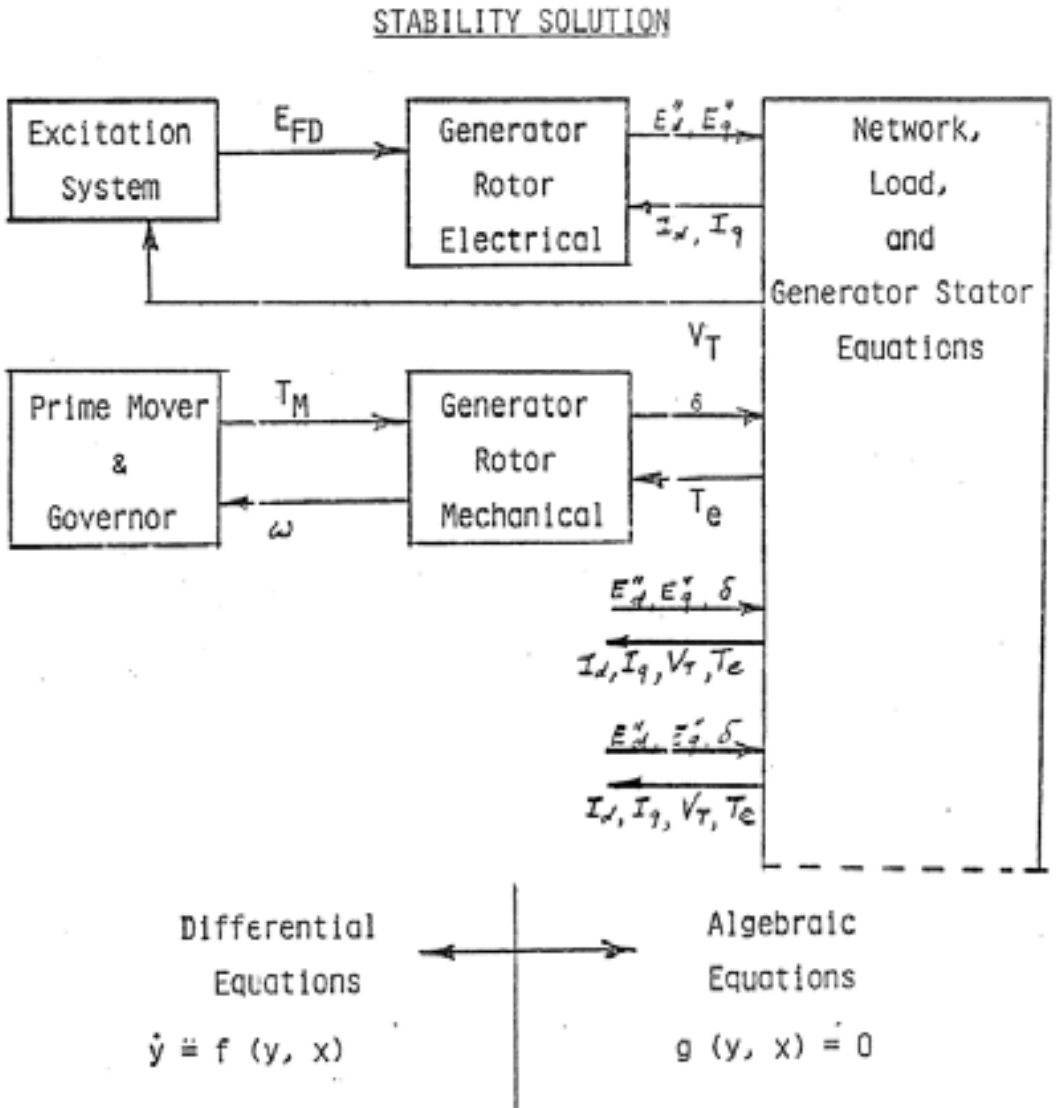


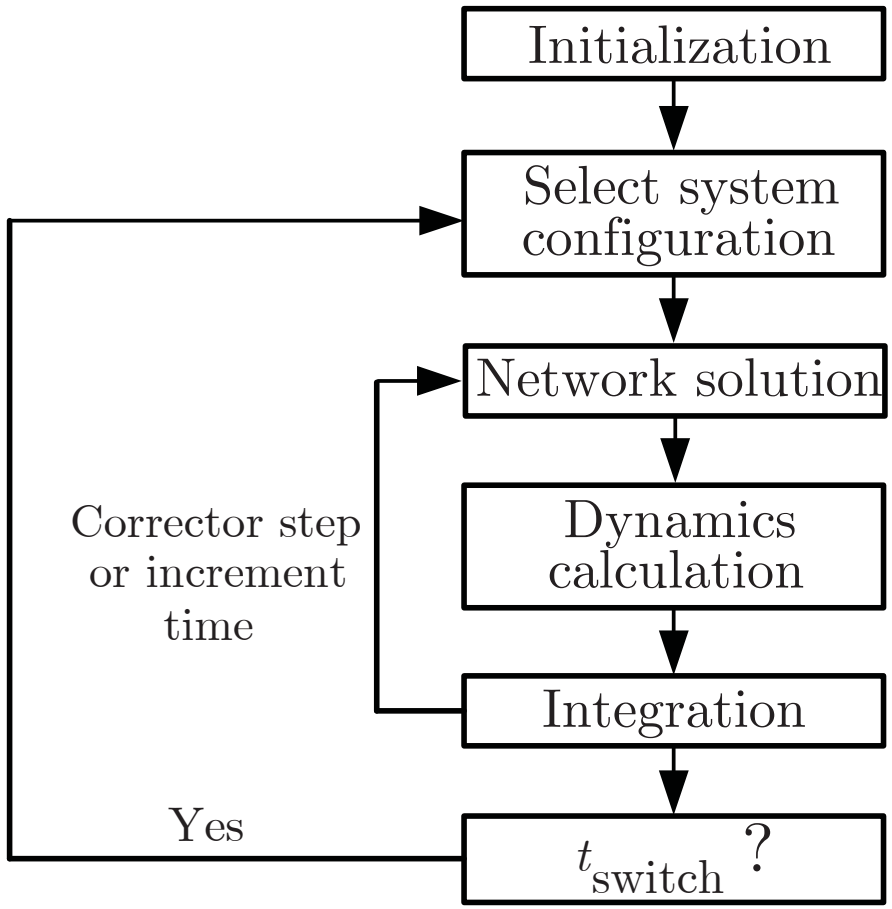
Figure 6 — Stability boundaries of various methods

Explicit integration methods can be unstable if h is too large. In the figure, $\sigma \pm j\omega$ are the largest eigenvalues of the linearized model.

Power System Computation Variable Flow Diagram (W. W. Price)



Dynamic Simulation Flowchart



From loadflow, back calculate to the machine internal node to find E', δ

Pre-fault, fault-on, post-fault

$Y\tilde{V} = \tilde{I}$, with appropriate Y

Compute $\dot{\delta}$, $\dot{\omega}$, etc

Note: for 3-phase short circuit at Bus k, $Y_{Lk} = \infty$, that is, set Y_{Lk} to a very large number, like 9999 pu.

- As a general-purpose software code, need to accommodate as many buses and machines as needed.
  - Y_N is built using bus and line data
  - Generators and other dynamic equipment are called from subroutines. A convenient way is to divide the code into 3 parts:
    1. Initialization code, used only once at the start.
    2. Network solution - setting up E'. In classical model E' is fixed. Detailed machine models require flux computation to determine machine voltages.
    3. (a) Dynamics (or time-derivative) computation - $\dot{\delta}$, $\dot{\omega}$, flux linkage states, etc.
       (b) Integration - modified Euler or higher-order methods

    Steps 2 and 3 are used repeatedly. Note that the dynamics code is usable in both the predictor and corrector parts.

Functions for Models

```
% synchronous machine subtransient model
function [f] = mac_sub(i,k,bus,flag) ...
  if flag == 0; % initialization
...
    fldcur(i,1) = E_Isat + mac_pot(i,6)*(eqprime(i,1)-...
                  psikd(i,1))+mac_pot(i,7)*curdg(i,1);
  if flag == 1 % network interface computation
...
    psi_re(i,k) = sin(mac_ang(i,k))*(-psiqpp) + ...
                  cos(mac_ang(i,k))*psidpp; % real part of psi
...
  if flag == 2 | flag == 3 % generator dynamics calculation
...
    dmac_spd(i,k) = (pmech(i,k)+pm_sig(i,k)-Te...
                  mac_con(i,17)*(mac_spd(i,k)-1)...
                  -mac_con(i,18)*(mac_spd(i,k)-sys_freq(k)))...
                  /(2*mac_con(i,16));
...
```

Generator data

```
% Column
% 1.  machine number
% 2.  bus number
% 3.  base mva
% 4.  leakage reactance x_l(pu)
% 5.  resistance r_a(pu)
% 6.  d-axis sychronous 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.  damping coefficient d_o(pu)
% 18.  dampling coefficient d_1(pu)
% 19.  bus number
% 20.  S(1.0) - saturation factor
% 21.  S(1.2) - saturation factor
%
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];
```

Excitation System Data

```
% Column
% 1.   exciter type - 3 for ST3
% 2.   machine number
% 3.   input filter time constant T_R
% 4.   voltage regulator gain K_A
% 5.   voltage regulator time constant T_A
% 6.   voltage regulator time constant T_B
% 7.   voltage regulator time constant T_C
% 8.   maximum voltage regulator output V_Rmax
% 9.   minimum voltage regulator output V_Rmin
% 10.   maximum internal signal V_Imax
% 11.   minimum internal signal V_Imin
% 12.   first stage regulator gain K_J
% 13.   potential circuit gain coefficient K_p
% 14.   potential circuit phase angle theta_p
```

```
% 15.  current circuit gain coefficient K_I
% 16.  potential source reactance X_L
% 17.  rectifier loading factor K_C
% 18.  maximum field voltage E_fdmax
% 19.  inner loop feedback constant K_G
% 20.  maximum inner loop voltage feedback V_Gmax
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];
```

Disturbance Script

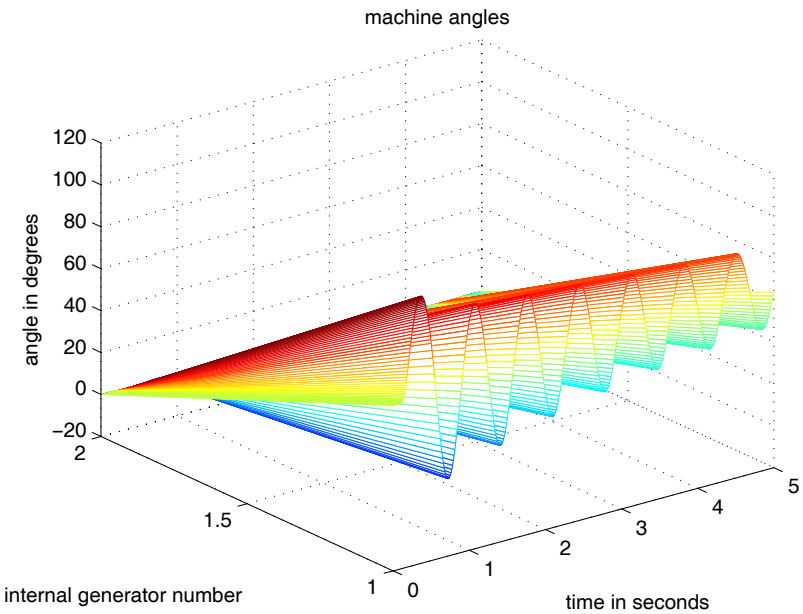
```
% row 1 col1 simulation start time (s) (cols 2 to 6 zeros)
%       col7 initial time step (s)
% row 2 col1 fault application time (s)
%       col2 bus number at which fault is applied
%       col3 bus number defining far end of faulted line
%       col4 zero sequence impedance in pu on system base
%       col5 negative sequence impedance in pu on system base
%       col6 type of fault - 0 three phase
%                          - 1 line to ground
%                          - 2 line-to-line to ground
%                          - 3 line-to-line
%                          - 4 loss of line with no fault
%                          - 5 loss of load at bus
%       col7 time step for fault period (s)
% row 3 col1 near end fault clearing time (s) (cols 2 to 6 zeros)
%       col7 time step for second part of fault (s)
```

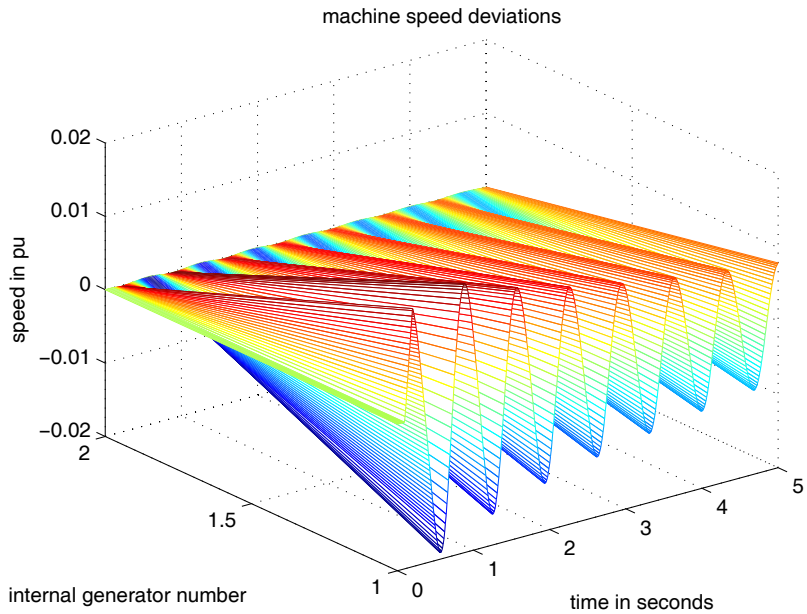
```
% row 4 col1 far end fault clearing time (s) (cols 2 to 6 zeros)
%       col7 time step for fault cleared simulation (s)
% row 5 col1 time to change step length (s)
%       col7 time step (s)
%
% row n col1 finishing time (s) (last row)
%
sw_con = [...
0     0 0 0 0 0 0.01;  % set initial time step
0.1   3 2 0 0 0 0.001; % apply 3-ph fault at bus 3 on line 3-2
0.167 0 0 0 0 0 0.001; % clear fault at bus 3
0.167 0 0 0 0 0 0.001; % clear remote end
0.50  0 0 0 0 0 0.01;  % increase time step
1.0   0 0 0 0 0 0.01;  % increase time step
5.0   0 0 0 0 0 0  ];  % end simulation
```

Simulation

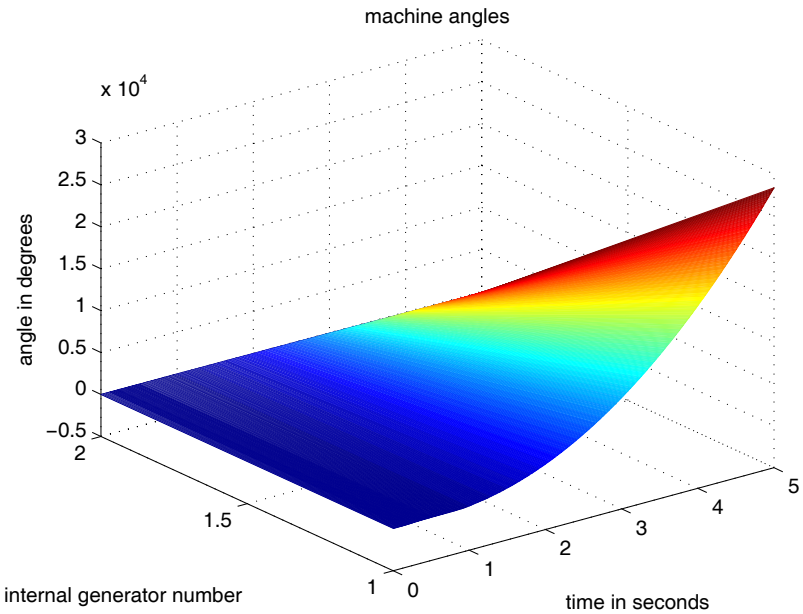
- PST command for simulation: `s_simu`
- Input appropriate MVA base (consistent with line data)
- Simulation results on a single-machine infinite-bus system: 3-phase short-circuit fault on the generator terminal bus, cleared by removing one of the two parallel lines to the infinite bus; with a high-gain excitation system but no PSS so the time response will be oscillatory.
- Two cases: fault cleared in 100 ms (about 6 cycles) - stable, and in 142 ms (8.5 cycles) - unstable.

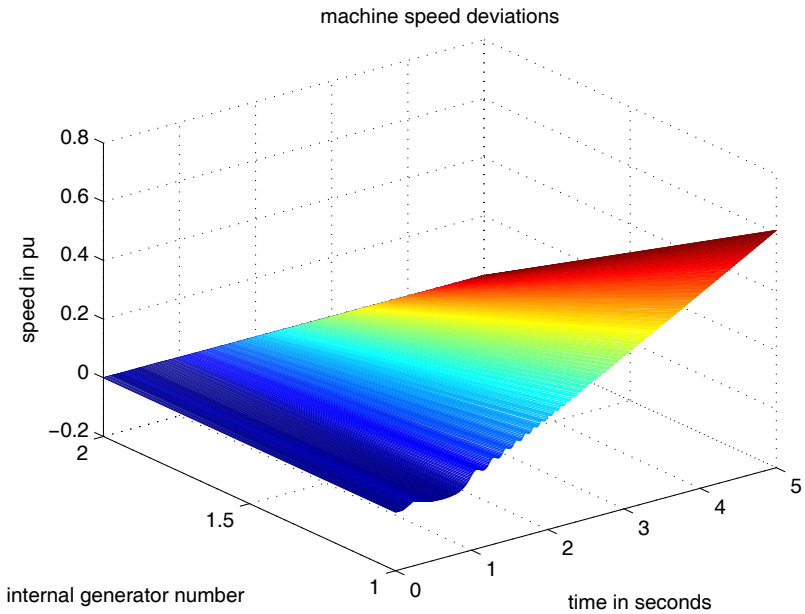
Simulation Results - Stable Case





Simulation Results - Unstable Case





Getting PST

- From Joe Chow's website
- Directly to the url http://www.eps.ee.kth.se/personal/vanfretti/pst/
- No cost to users; password to decompress the MATLAB code will be provided.