

ECEN 667

Power System Stability

Lecture 13: Governors, PID Controllers

Prof. Tom Overbye

Dept. of Electrical and Computer Engineering

Texas A&M University

overbye@tamu.edu



TEXAS A&M
UNIVERSITY

Announcements



- Read Chapter 4
- Exam 1 is Thursday October 10 during class; closed book, closed notes. One 8.5 by 11 inch note sheet and calculators allowed.

PID Controllers

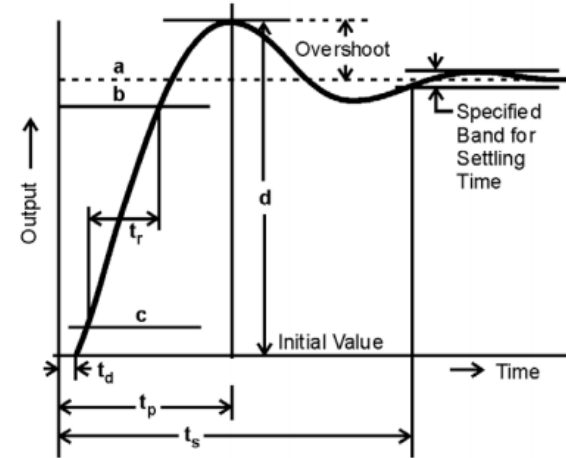


- Governors and exciters often use proportional-integral-derivative (PID) controllers
 - Developed in 1890's for automatic ship steering by observing the behavior of experienced helmsman
- PIDs combine
 - Proportional gain, which produces an output value that is proportional to the current error
 - Integral gain, which produces an output value that varies with the integral of the error, eventually driving the error to zero
 - Derivative gain, which acts to predict the system behavior. This can enhance system stability, but it can be quite susceptible to noise

PID Controller Characteristics



- Four key characteristics of control response are
 - 1) rise time,
 - 2) overshoot,
 - 3) settling time and
 - 4) steady-state errors



a - Steady-state value
 b - 90% of steady-state value
 c - 10% of steady-state value
 d - peak value
 t_d - Delay time
 t_p - Time to reach peak value
 t_s - Settling time
 t_r - Rise time

Figure F.1—Typical dynamic response of a turbine governing system to a step change

| Increasing Gain | Rise Time | Overshoot | Setting Time | Steady-State Error |
|-----------------|---------------|-----------|---------------|--------------------|
| K_p | Decreases | Increases | Little impact | Decreases |
| K_I | Decreases | Increases | Increases | Zero |
| K_D | Little impact | Decreases | Decreases | Little Impact |

Image source: Figure F.1, IEEE Std 1207-2011

PID Example: Car Cruise Control

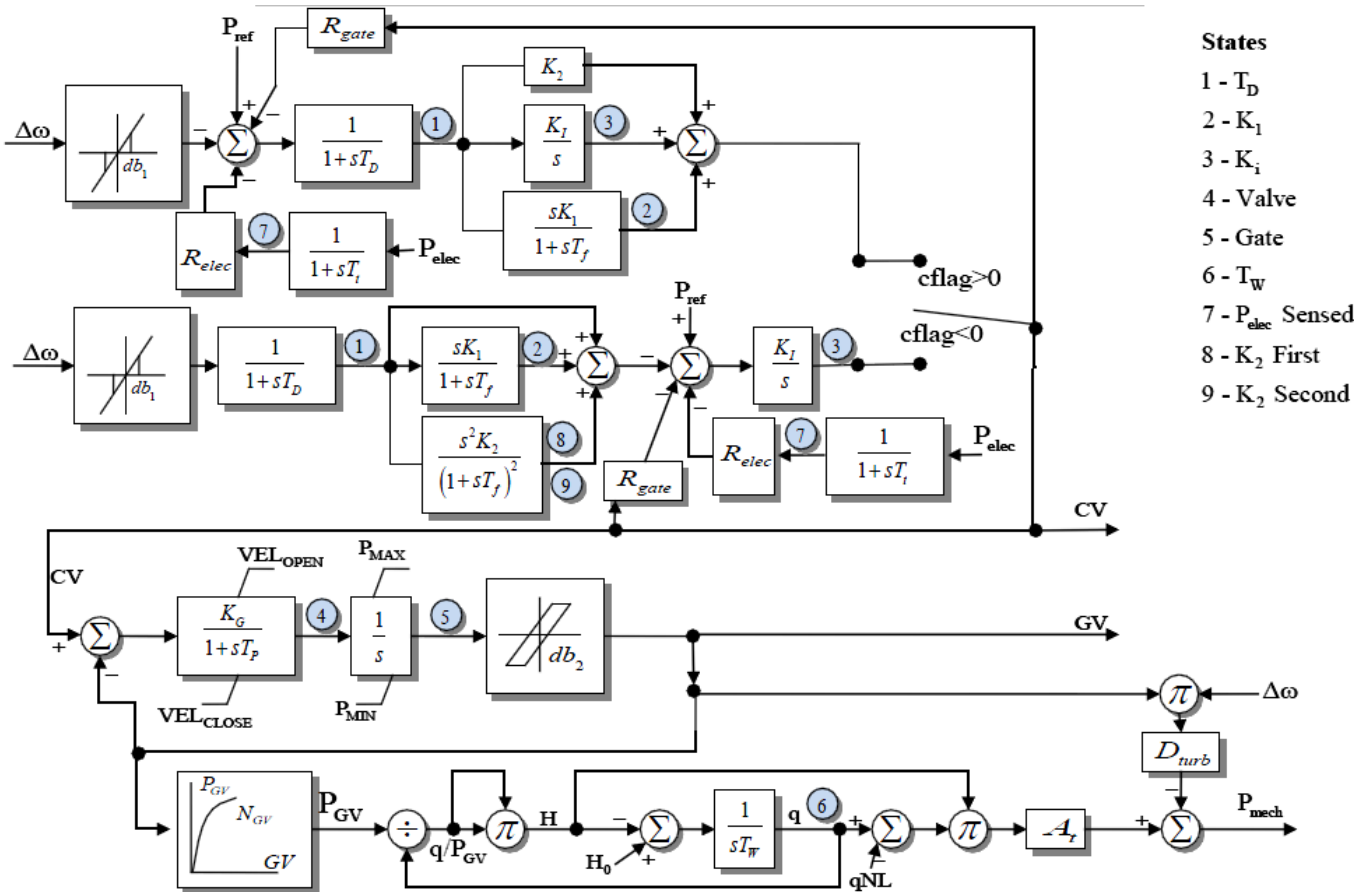


- Say we wish to implement cruise control on a car by controlling the throttle position
 - Assume force is proportional to throttle position
 - Error is difference between actual speed and desired speed
- With just proportional control we would never achieve the desired speed because with zero error the throttle position would be at zero
- The integral term will make sure we stay at the desired point
- With derivative control we can improve control, but as noted it can be sensitive to noise

HYG3



- The HYG3 models has a PID or a double derivative



- States
- 1 - T_D
 - 2 - K_1
 - 3 - K_1
 - 4 - Valve
 - 5 - Gate
 - 6 - T_w
 - 7 - P_{elec} Sensed
 - 8 - K_2 First
 - 9 - K_2 Second

Looks more complicated than it is since depending on cflag only one of the upper paths is used

About 15% of current WECC governors at HYG3

Tuning PID Controllers

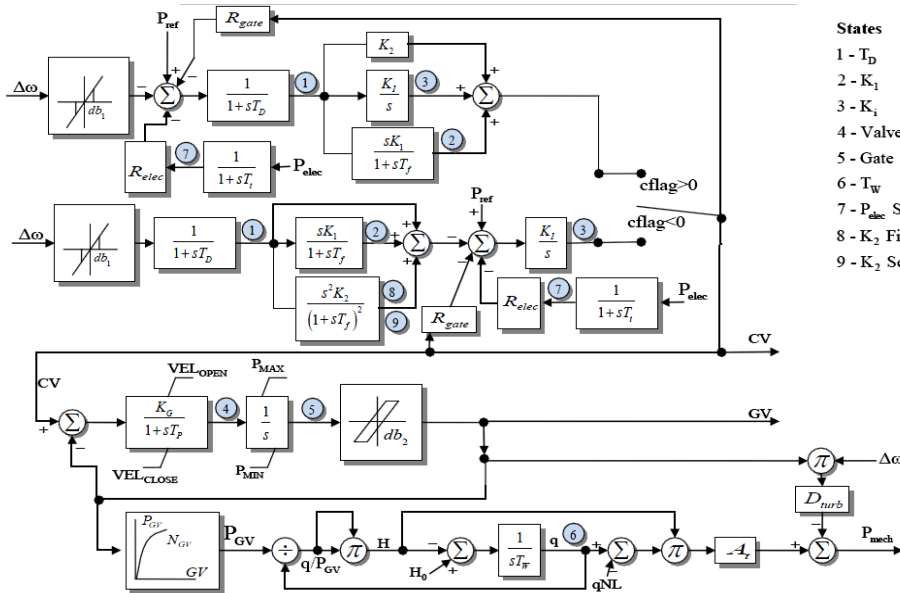


- Tuning PID controllers can be difficult, and there is no single best method
 - Conceptually simple since there are just three parameters, but there can be conflicting objectives (rise time, overshoot, setting time, error)
- One common approach is the Ziegler-Nichols method
 - First set K_I and K_D to zero, and increase K_P until the response to a unit step starts to oscillate (marginally stable); define this value as K_u and the oscillation period at T_u
 - For a P controller set $K_p = 0.5K_u$
 - For a PI set $K_p = 0.45 K_u$ and $K_I = 1.2 * K_p / T_u$
 - For a PID set $K_p = 0.6 K_u$, $K_I = 2 * K_p / T_u$, $K_D = K_p T_u / 8$

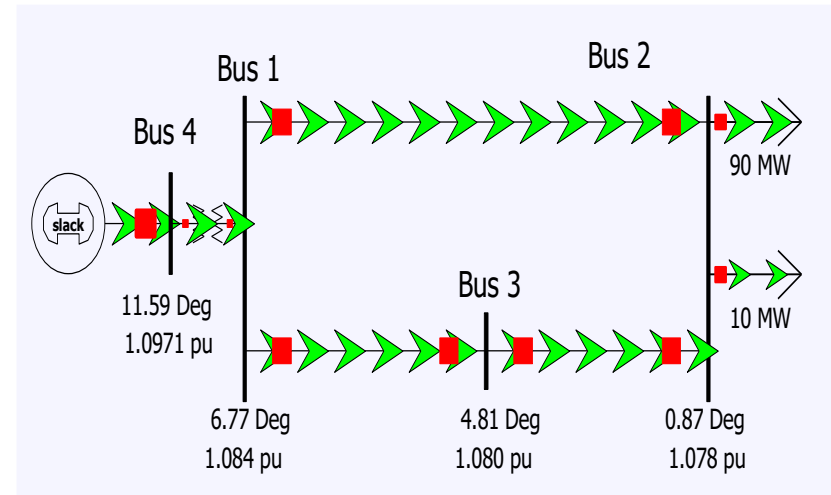
Tuning PID Controller Example



- Use the four bus case with infinite bus replaced by load, and gen 4 has a HYG3 governor with $cflag > 0$; tune K_P , K_I and K_D for full load to respond to a 10% drop in load (K_2 , K_I , K_I in the model; assume $T_f=0.1$)



- States
- 1 - T_D
 - 2 - K_I
 - 3 - K_I
 - 4 - Valve
 - 5 - Gate
 - 6 - T_w
 - 7 - P_{elec} Sensed
 - 8 - K_2 First
 - 9 - K_2 Second

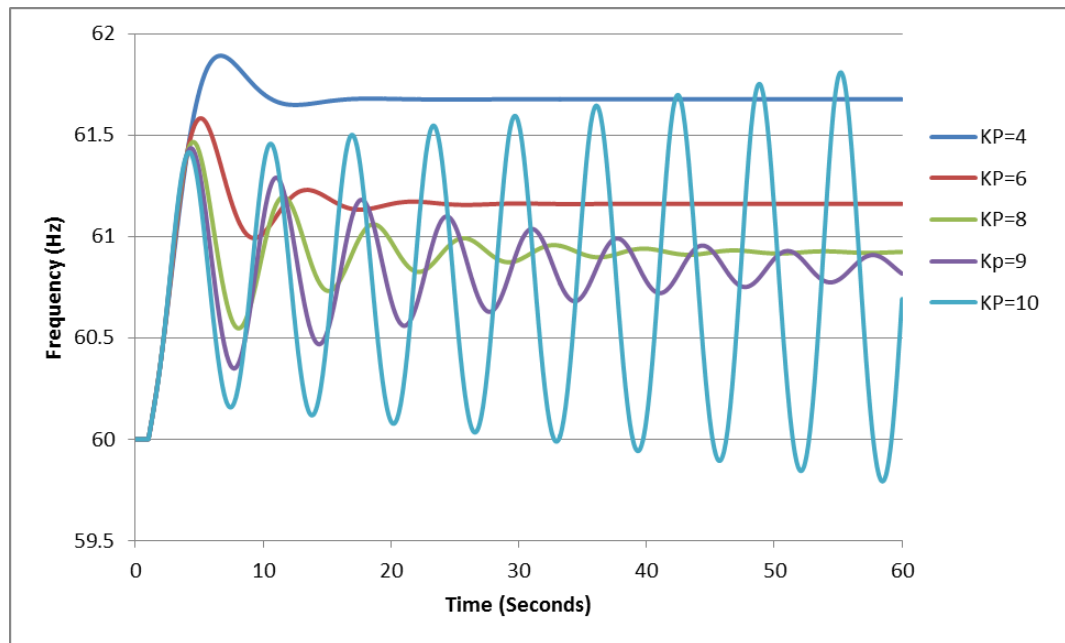


Case name: **B4_PIDTuning**

Tuning PID Controller Example



- Based on testing, K_u is about 9.5 and T_u is 6.4 seconds
- Using Ziegler-Nichols a good P value 4.75, is good PI values are $K_P = 4.3$ and $K_I = 0.8$, while good PID values are $K_P = 5.7$, $K_I = 1.78$, $K_D = 4.56$

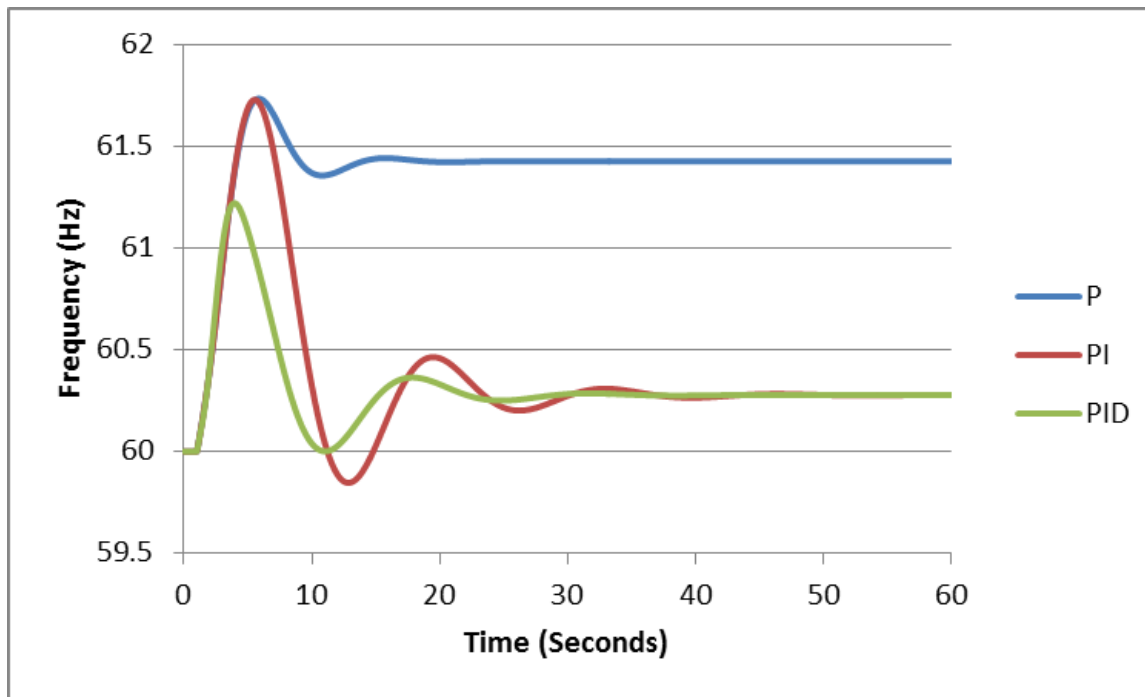


Further details on tuning are covered in IEEE Std. 1207-2011

Tuning PID Controller Example



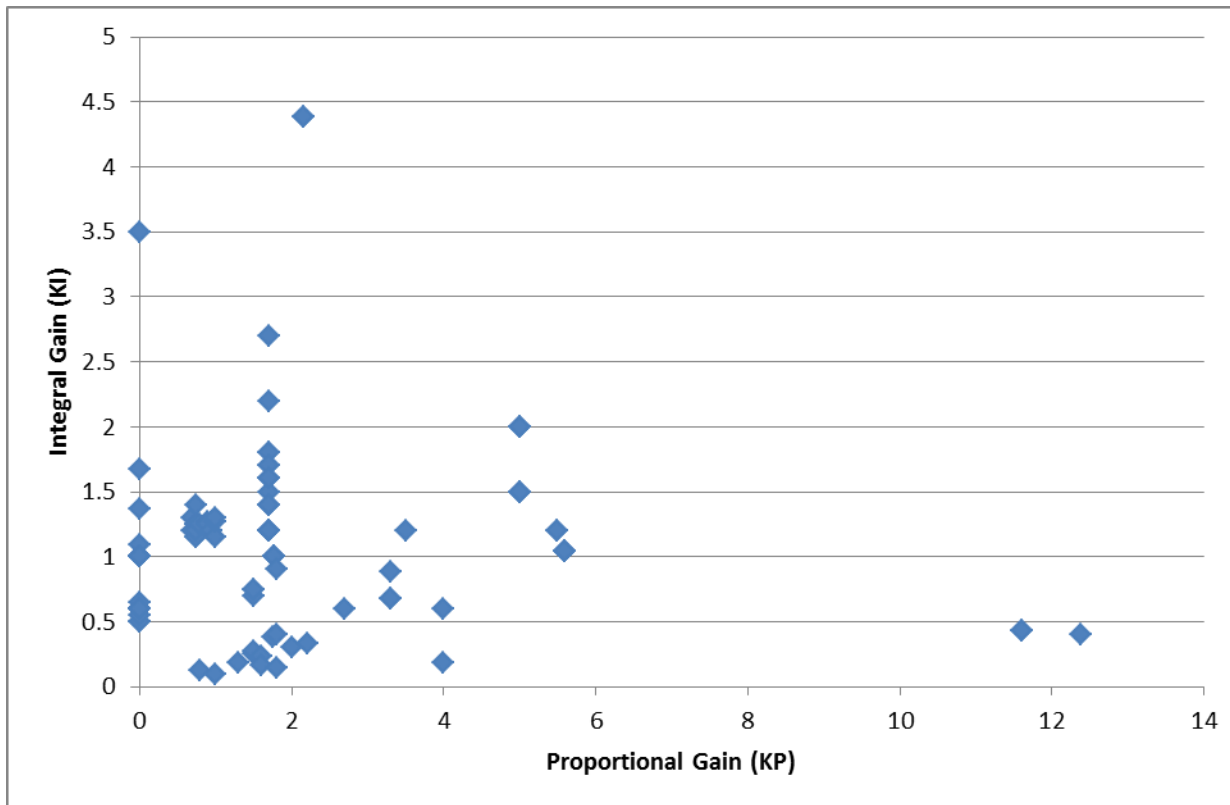
- Figure shows the Ziegler-Nichols for a P, PI and PID controls. Note, this is for stand-alone, not interconnected operation



Example K_I and K_P Values



- Figure shows example K_I and K_P values from an actual system case



About 60% of the models also had a derivative term with an average value of 2.8, and an average T_D of 0.04 sec

Non-windup Limits

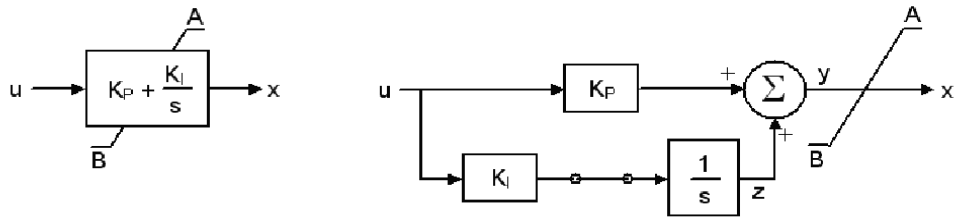


- An important open question is whether the governor PI controllers should be modeled with non-windup limits
 - Currently models show no limit, but transient stability verification seems to indicate limits are being enforced
- This could be an issue if frequency goes low, causing governor PI to "windup" and then goes high (such as in an islanding situation)
 - How fast governor backs down depends on whether the limit winds up

PI Non-windup Limits



- There is not a unique way to handle PI non-windup limits; the below shows two approaches from IEEE Std 421.5



$y > A$, then $x = A$ and $dz/dt = 0$

$y < B$, then $x = B$ and $dz/dt = 0$

Figure E.7—Non-windup proportional-integral block

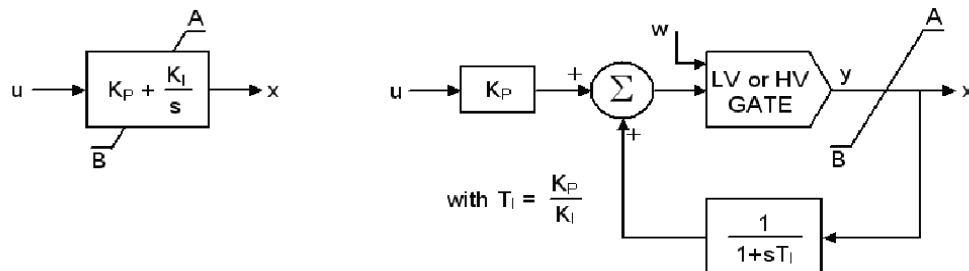


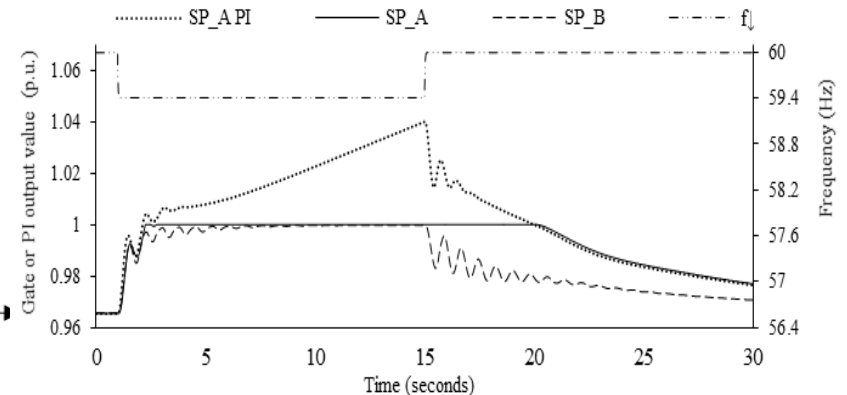
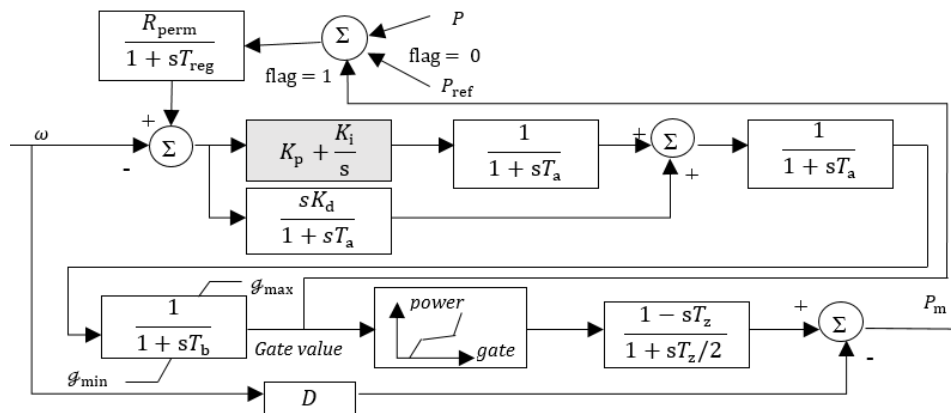
Figure E.8—Non-windup proportional-integral block

Another common approach is to cap the output and put a non-windup limit on the integrator

PI Limit Problems with Actual Hydro Models



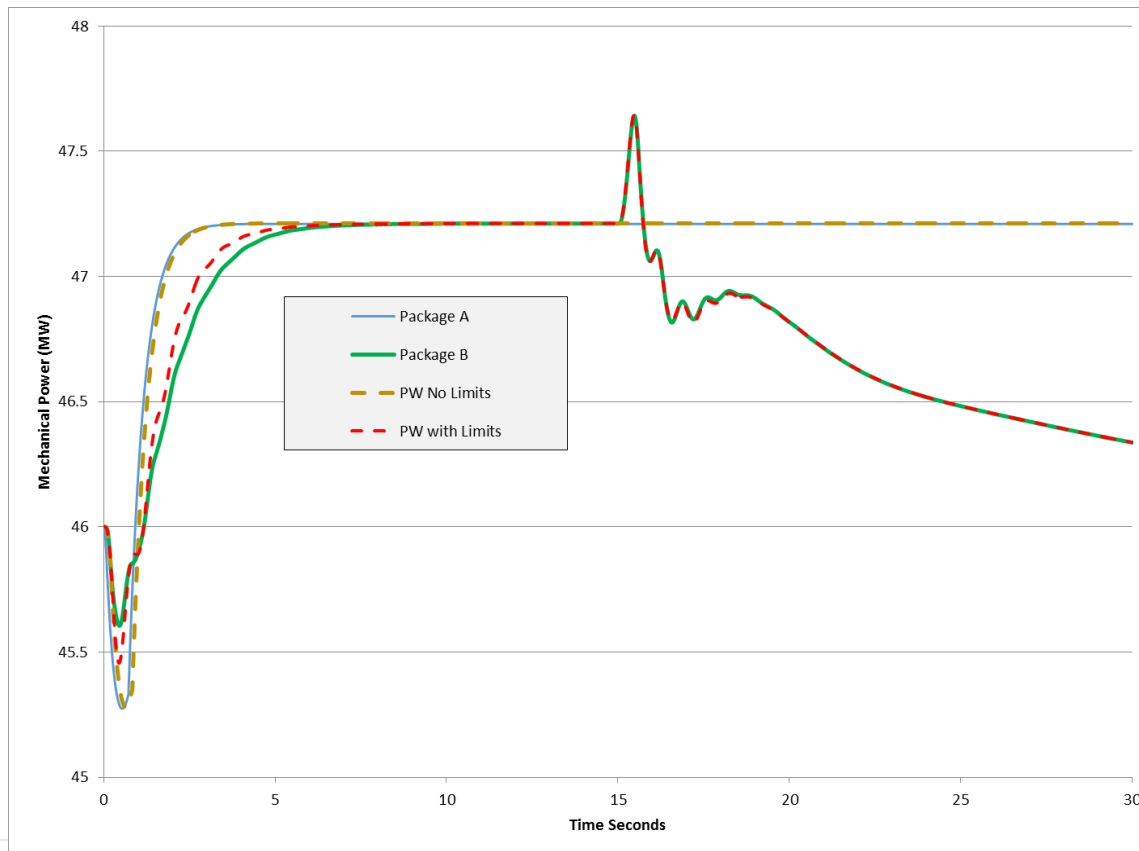
- A recent research project comparing transient stability packages found there were significant differences between hydro model implementations with respect to how PI limits were modeled
 - One package modeled limits but did not document them, another did not model them; limits were recommended at WECC MVWG in May 2014



PIDGOV Model Results



- Below graph compares the P_{mech} response for a two bus system for a frequency change, between three transient stability packages



Packages A and B both say they have no governor limits, but B seems to; PowerWorld can do either approach

GGOV1



- GGOV1 is a relatively newer governor model introduced in early 2000's by WECC for modeling thermal plants
 - Existing models greatly under-estimated the frequency drop
 - GGOV1 is now the most common WECC governor, used with about 40% of the units
- A useful reference is L. Pereira, J. Undrill, D. Kosterev, D. Davies, and S. Patterson, "A New Thermal Governor Modeling Approach in the WECC," *IEEE Transactions on Power Systems*, May 2003, pp. 819-829

GGOV1: Selected Figures from 2003 Paper

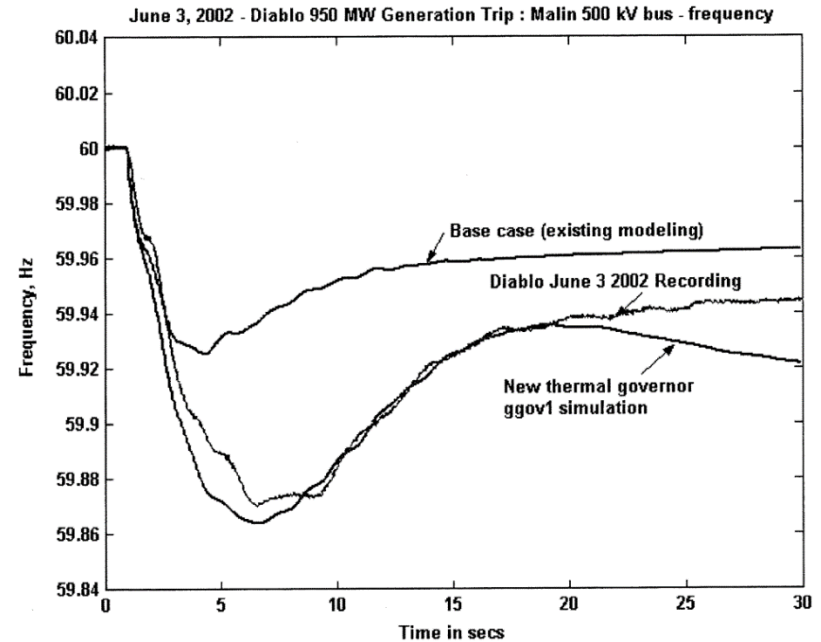
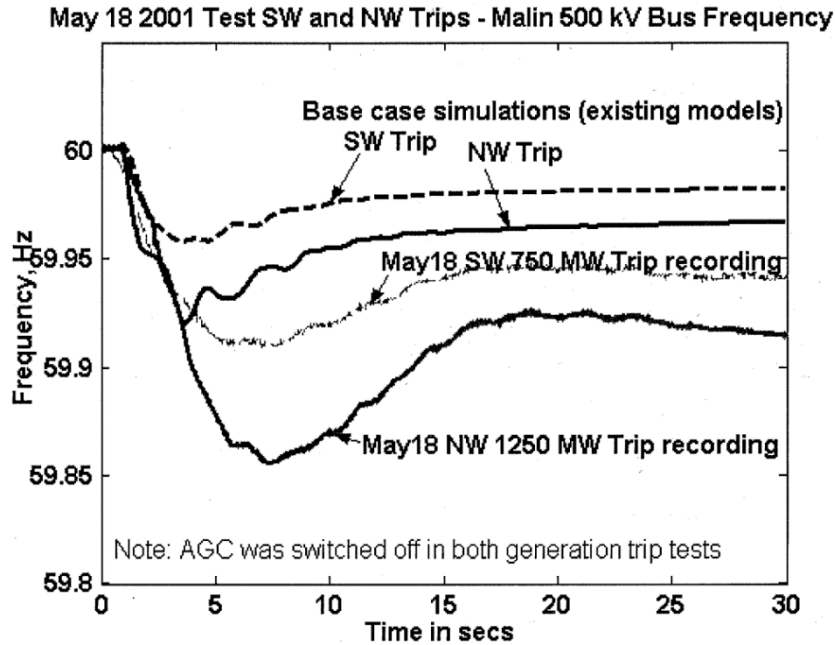
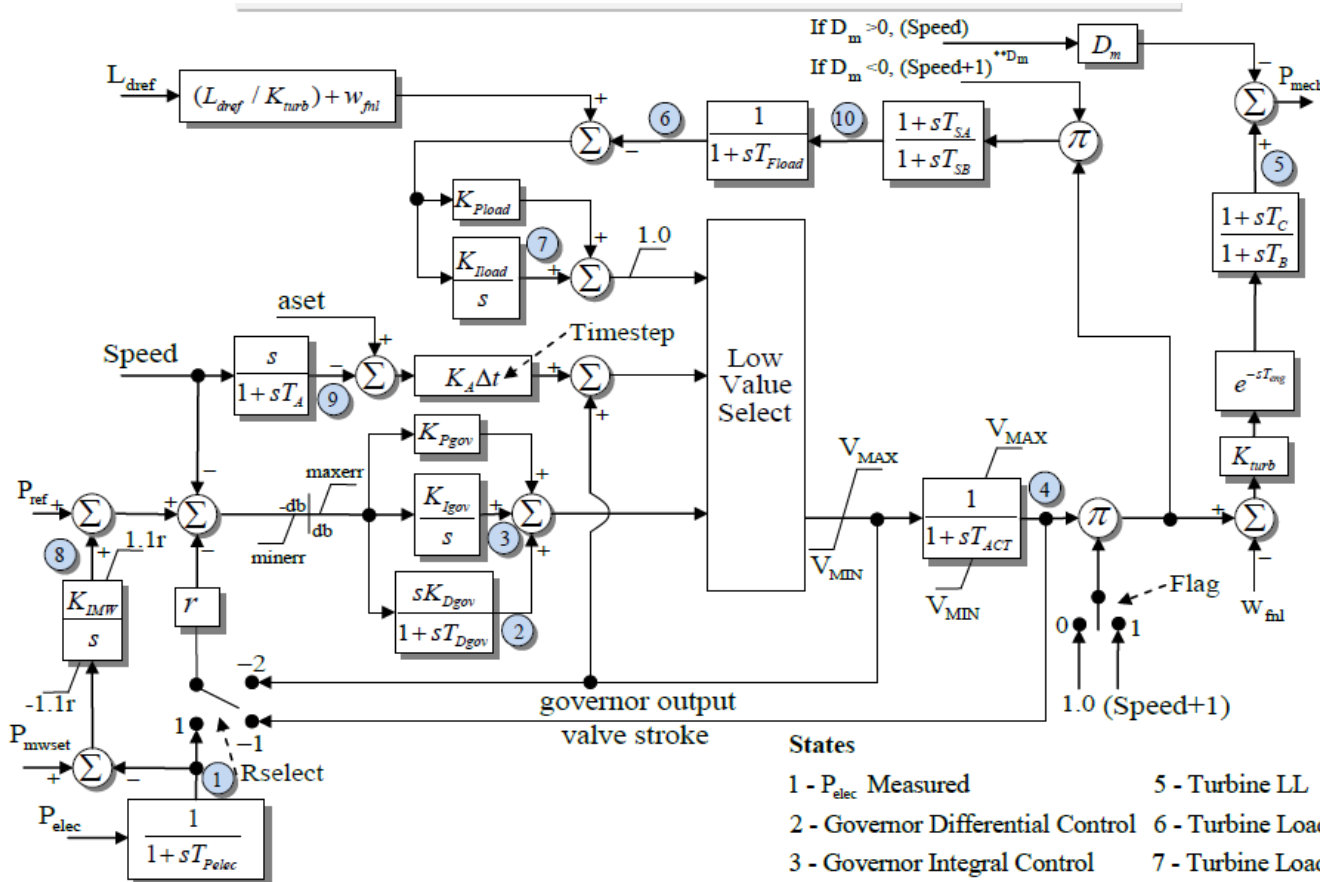


Fig. 1. Frequency recordings of the SW and NW trips on May 18, 2001. Also shown are simulations with existing modeling (base case).

Governor model verification—950-MW Diablo generation trip on June 3, 2002.

Diablo Canyon is California’s last nuclear plant, with Unit 1 now scheduled to shutdown in 2024 and Unit 2 in 2025.

GGOV1 Block Diagram



GGOV1 and the related GGOV3 are the most common governors in WECC, with more than 40% in 2019

- States
- 1 - P_{elec} Measured
 - 2 - Governor Differential Control
 - 3 - Governor Integral Control
 - 4 - Turbine Actuator
 - 5 - Turbine LL
 - 6 - Turbine Load Limiter
 - 7 - Turbine Load Integral Control
 - 8 - Supervisory Load Control
 - 9 - Accel Control
 - 10 - Temp Detection LL

Model supported by PSLF
 Model supported by PSS/E does not include non-windup limits on K_{IMW} block
 R_{UP} , R_{DOWN} , R_{CLOSE} , and R_{OPEN} inputs not implemented in Simulator

Transient Stability Multimachine Simulations



- Next, we'll be putting the models we've covered so far together
- Later we'll add in new model types such as stabilizers, loads and wind turbines
- By way of history, prior to digital computers, network analyzers were used for system stability studies as far back as the 1930's (perhaps earlier)
 - For example see, J.D. Holm, "Stability Study of A-C Power Transmission Systems," AIEE Transactions, vol. 61, 1942, pp. 893-905
- Digital approaches started appearing in the 1960's

Transient Stability Multimachine Simulations



- The general structure is as a set of differential-algebraic equations
 - Differential equations describe the behavior of the machines (and the loads and other dynamic devices)
 - Algebraic equations representing the network constraints

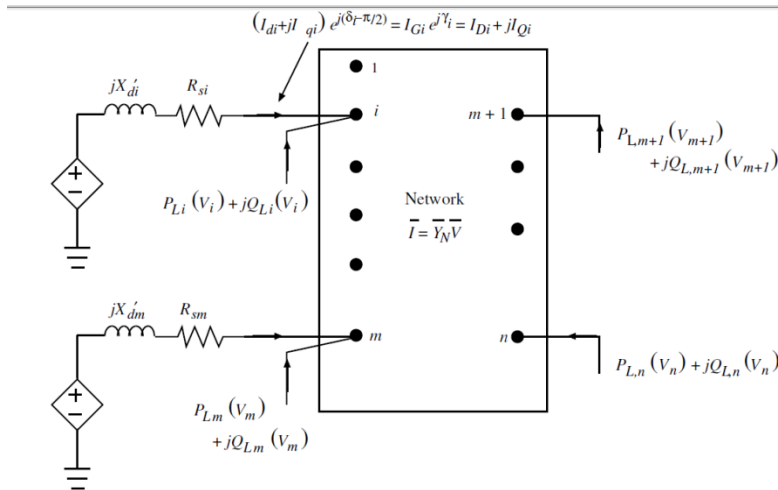


Figure 7.2: Interconnection of synchronous machine dynamic circuit and the rest of the network

In EMTP applications the transmission line delays decouple the machines; in transient stability they are assumed to be coupled together by the algebraic network equations

General Form



- The general form of the problem is solving

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{u})$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}, \mathbf{y})$$

where \mathbf{x} is the vector of the state variables (such as the generator δ 's), \mathbf{y} is the vector of the algebraic variables (primarily the bus complex voltages), and \mathbf{u} is the vector of controls (such as the exciter voltage setpoints)

Transient Stability General Solution



- General solution approach is
 - Solve power flow to determine initial conditions
 - Back solve to get initial states, starting with machine models, then exciters, governors, stabilizers, loads, etc
 - Set $t = t_{\text{start}}$, time step = Δt , abort = false
 - While ($t \leq t_{\text{end}}$) and (not abort) Do Begin
 - Apply any contingency event
 - Solve differential and algebraic equations
 - If desired store time step results and check other conditions (that might cause the simulation to abort)
 - $t = t + \Delta t$
 - End while

Algebraic Constraints



- The \mathbf{g} vector of algebraic constraints is similar to the power flow equations, but usually rather than formulating the problem like in the power flow as real and reactive power balance equations, it is formulated in the current balance form

$$\mathbf{I}(\mathbf{x}, \mathbf{V}) = \mathbf{Y} \mathbf{V} \quad \text{or} \quad \mathbf{Y} \mathbf{V} - \mathbf{I}(\mathbf{x}, \mathbf{V}) = \mathbf{0}$$

where \mathbf{Y} is the $n \times n$ bus admittance matrix ($\mathbf{Y} = \mathbf{G} + j\mathbf{B}$), \mathbf{V} is the complex vector of the bus voltages, and \mathbf{I} is the complex vector of the bus current injections

Simplest cases can have \mathbf{I} independent of \mathbf{x} and \mathbf{V} , allowing a direct solution; otherwise we need to iterate

Why Not Use the Power Flow Equations?

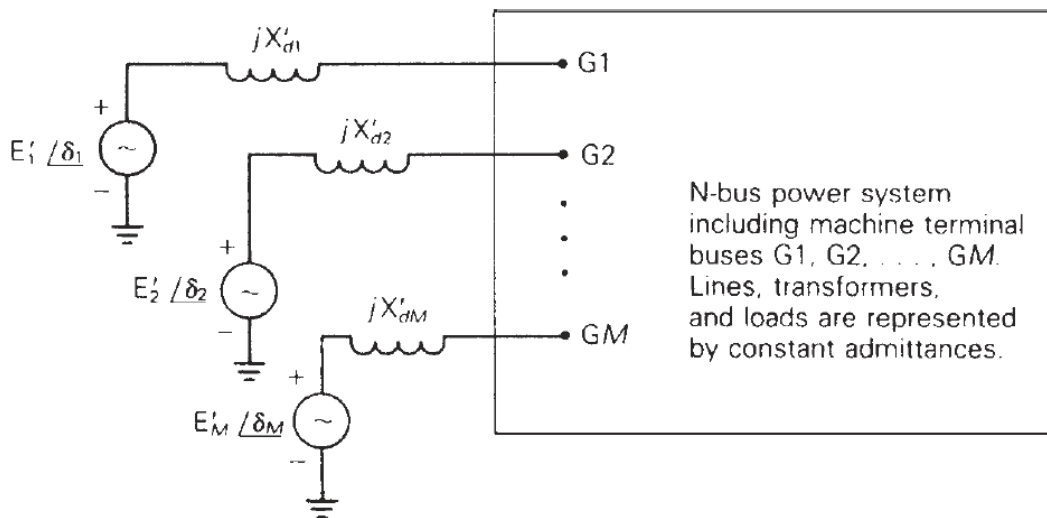


- The power flow equations were ultimately derived from
$$\mathbf{I}(\mathbf{x}, \mathbf{V}) = \mathbf{Y} \mathbf{V}$$
- However, the power form was used in the power flow primarily because
 - For the generators the real power output is known and either the voltage setpoint (i.e., if a PV bus) or the reactive power output
 - In the quasi-steady state power flow time frame the loads can often be well approximated as constant power
 - The constant frequency assumption requires a slack bus
- These assumptions do not hold for transient stability

Algebraic Equations for Classical Model



- To introduce the coupling between the machine models and the network constraints, consider a system modeled with just classical generators and impedance loads



In this example because we are using the classical model all values are on the network reference frame

We'll extend the figure slightly to include stator resistances, $R_{s,i}$

Algebraic Equations for Classical Model



- Replace the internal voltages and their impedances by their Norton Equivalent

$$\bar{I}_i = \frac{E'_i \angle \delta_i}{R_{s,i} + jX'_{d,i}}, \quad Y_i = \frac{1}{R_{s,i} + jX'_{d,i}}$$

- Current injections at the non-generator buses are zero since the constant impedance loads are included in \mathbf{Y}
 - We'll modify this later when we talk about dynamic loads
- The algebraic constraints are then $\mathbf{I} - \mathbf{Y} \mathbf{V} = \mathbf{0}$

Swing Equation



- The first two differential equations for any synchronous machine correspond to the swing equation

$$\frac{d\delta_i}{dt} = \omega_i - \omega_s = \Delta\omega_i$$

$$\frac{2H_i}{\omega_s} \frac{d\omega_i}{dt} = \frac{2H_i}{\omega_s} \frac{d\Delta\omega_i}{dt} = T_{Mi} - T_{Ei} - D_i (\Delta\omega_i)$$

$$\text{with } T_{Ei} = \psi_{de,i} i_{qi} - \psi_{qe,i} i_{di}$$

Swing Equation Speed Effects



- There is often confusion about these equations because of whether speed effects are included
 - Recognizing that often $\omega \approx \omega_s$ (which is one per unit), some transient stability books have neglected speed effects
- For a rotating machine with a radial torque, power = torque times speed
- For a subtransient model

$$\bar{E}'' = \bar{V} + (R_s + jX'')\bar{I}, \quad E_d'' + jE_q'' = (-\psi_q'' + j\psi_d'')\omega$$

$$T_E = \psi_d'' I_q - \psi_q'' I_d \quad \text{and}$$

$$P_E = T_E \omega = (E_d'' + jE_q'')(I_d - jI_q) = E_d'' I_d + E_q'' I_q$$

Classical Swing Equation



- Often in an introductory coverage of transient stability with the classical model the assumption is $\omega \approx \omega_s$ so the swing equation for the classical model is given as

$$\frac{d\delta_i}{dt} = \omega_i - \omega_s = \Delta\omega_i$$

$$\frac{2H_i}{\omega_s} \frac{d\Delta\omega_i}{dt} = P_{Mi} - P_{Ei} - D_i (\Delta\omega_i)$$

$$\text{with } P_{Ei} = (E'_i \angle \delta_i) (E'_i \angle \delta_i - \bar{V}_i) Y_i$$

- We'll use this simplification for our initial example

Numerical Solution



- There are two main approaches for solving

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{u})$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}, \mathbf{y})$$

- Partitioned-explicit: Solve the differential and algebraic equations separately (alternating between the two) using an explicit integration approach
- Simultaneous-implicit: Solve the differential and algebraic equations together using an implicit integration approach

Outline for Next Several Slides



- The next several slides will provide basic coverage of the solution process, partitioned explicit, then the simultaneous-implicit approach
- We'll start out with a classical model supplying an infinite bus, which can be solved by embedded the algebraic constraint into the differential equations

We'll start out just solving $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$

and then will extend to solving the full problem of

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{u})$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}, \mathbf{y})$$

Classical Swing Equation with Embedded Power Balance



- With a classical generator at bus i supplying an infinite bus with voltage magnitude V_{inf} , we can write the problem without algebraic constraints as

$$\frac{d\delta_i}{dt} = \omega_i - \omega_s = \Delta\omega_i = \Delta\omega_{i,pu} \omega_s$$

$$\frac{d\Delta\omega_{i,pu}}{dt} = \frac{1}{2H_i} \left(P_{Mi} - \frac{E'_i V_{\text{inf}}}{X_{th}} \sin \delta_i - D_i (\Delta\omega_{i,pu}) \right)$$

$$\text{with } P_{Ei} = \frac{E'_i V_{\text{inf}}}{X_{th}} \sin \delta_i$$

Note we are using the per unit speed approach

Explicit Integration Methods



- As covered on the first day of class, there are a wide variety of explicit integration methods
 - We considered Forward Euler, Runge-Kutta, Adams-Bashforth
- Here we will just consider Euler's, which is easy to explain but not too useful, and a second order Runge-Kutta, which is commonly used

Forward Euler



- Recall the Forward Euler approach is approximate

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t)) = \frac{d\mathbf{x}}{dt} \text{ as } \frac{\Delta\mathbf{x}}{\Delta t}$$

Then

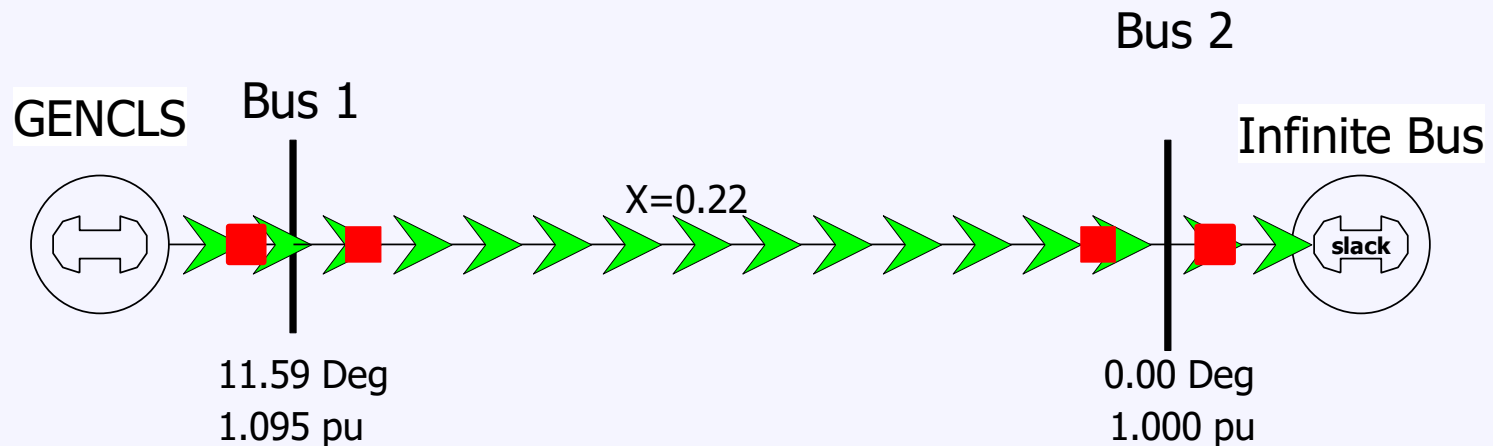
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t \mathbf{f}(\mathbf{x}(t))$$

- Error with Euler's varies with the square of the time step

Infinite Bus GENCLS Example using the Forward Euler's Method



- Use the four bus system from before, except now gen 4 is modeled with a classical model with $X_d'=0.3$, $H=3$ and $D=0$; also we'll reduce to two buses with equivalent



In this example $X_{th} = (0.22 + 0.3)$, with the internal voltage $\bar{E}'_1 = 1.281 \angle 23.95^\circ$ giving $E'_1 = 1.281$ and $\delta_1 = 23.95^\circ$

Infinite Bus GENCLS Example



- The associated differential equations for the bus 1 generator are

$$\frac{d\delta_1}{dt} = \Delta\omega_{1,pu} \omega_s$$

$$\frac{d\Delta\omega_{1,pu}}{dt} = \frac{1}{2 \times 3} \left(1 - \frac{1.281}{0.52} \sin \delta_1 \right)$$

- The value of $P_{M1} = 1$ is determined from the initial conditions, and would stay constant in this simple example without a governor
- The value $\delta_1 = 23.95^\circ$ is readily verified as an equilibrium point (which is 0.418 radians)

Infinite Bus GENCLS Example



- Assume a solid three phase fault is applied at the generator terminal, reducing P_{E1} to zero during the fault, and then the fault is self-cleared at time T^{clear} , resulting in the post-fault system being identical to the pre-fault system
 - During the fault-on time the equations reduce to

$$\frac{d\delta_1}{dt} = \Delta\omega_{1,pu} \omega_s$$
$$\frac{d\Delta\omega_{1,pu}}{dt} = \frac{1}{2 \times 3} (1 - 0)$$

That is, with a solid fault on the terminal of the generator, during the fault $P_{E1} = 0$

Euler's Solution



- The initial value of \mathbf{x} is

$$\mathbf{x}(0) = \begin{bmatrix} \delta(0) \\ \omega_{pu}(0) \end{bmatrix} = \begin{bmatrix} 0.418 \\ 0 \end{bmatrix}$$

- Assuming a time step $\Delta t = 0.02$ seconds, and a T^{clear} of 0.1 seconds, then using Euler's

$$\mathbf{x}(0.02) = \begin{bmatrix} 0.418 \\ 0 \end{bmatrix} + 0.02 \begin{bmatrix} 0 \\ 0.1667 \end{bmatrix} = \begin{bmatrix} 0.418 \\ 0.00333 \end{bmatrix}$$

- Iteration continues until $t = T^{\text{clear}}$

Note Euler's
assumes
 δ stays constant
during the first
time step

Euler's Solution



- At $t = T^{\text{clear}}$ the fault is self-cleared, with the equations changing to

$$\frac{d\delta}{dt} = \Delta\omega_{pu} \omega_s$$

$$\frac{d\Delta\omega_{pu}}{dt} = \frac{1}{6} \left(1 - \frac{1.281}{0.52} \sin \delta \right)$$

- The integration continues using the new equations

Euler's Solution Results ($\Delta t=0.02$)



- The below table gives the results using $\Delta t = 0.02$ for the beginning time steps

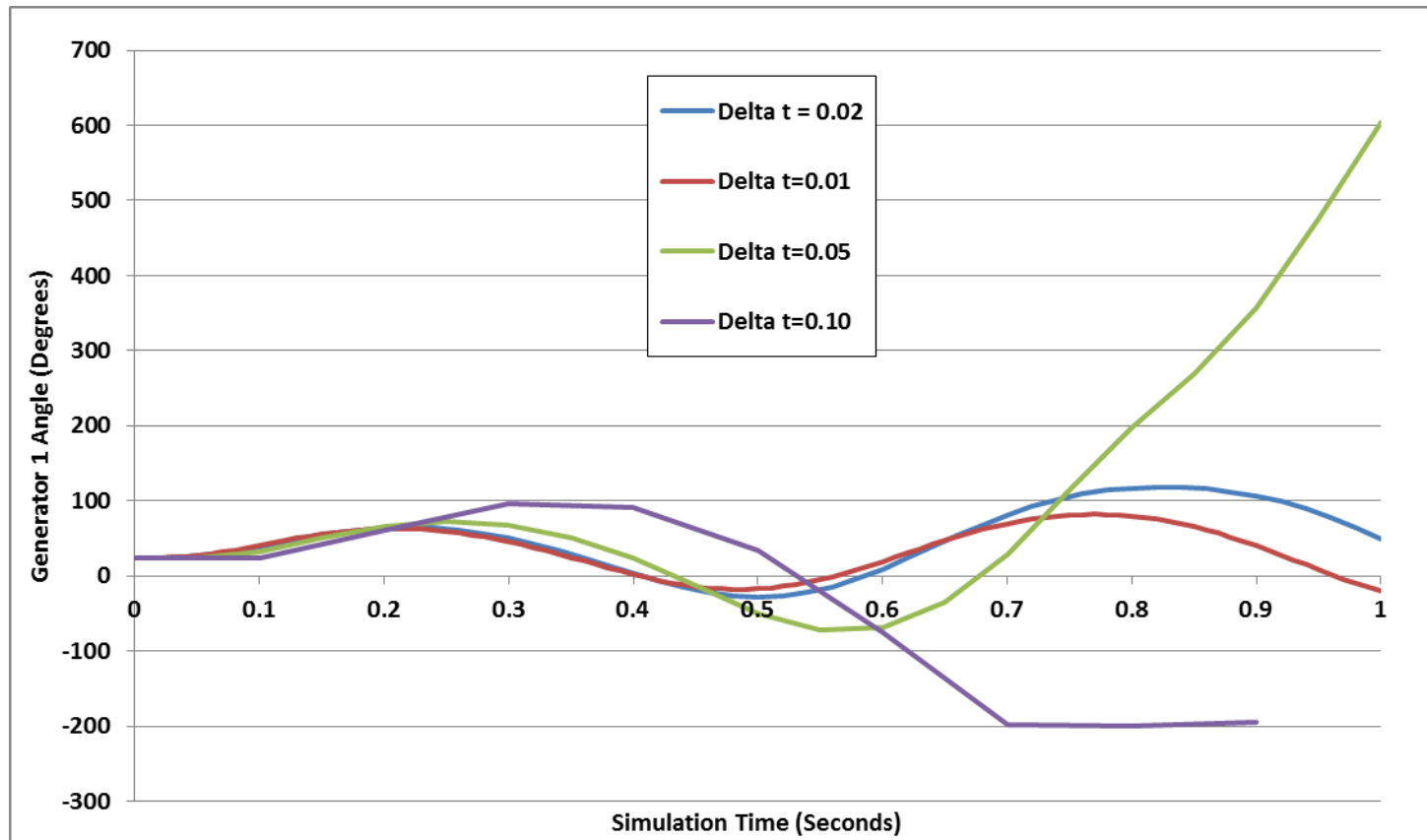
| Time | Gen 1 Rotor Angle, Degrees | Gen 1 Speed (Hz) |
|------|----------------------------|------------------|
| 0 | 23.9462 | 60 |
| 0.02 | 23.9462 | 60.2 |
| 0.04 | 25.3862 | 60.4 |
| 0.06 | 28.2662 | 60.6 |
| 0.08 | 32.5862 | 60.8 |
| 0.1 | 38.3462 | 61 |
| 0.1 | 38.3462 | 61 |
| 0.12 | 45.5462 | 60.8943 |
| 0.14 | 51.9851 | 60.7425 |
| 0.16 | 57.3314 | 60.5543 |
| 0.18 | 61.3226 | 60.3395 |
| 0.2 | 63.7672 | 60.1072 |
| 0.22 | 64.5391 | 59.8652 |
| 0.24 | 63.5686 | 59.6203 |
| 0.26 | 60.8348 | 59.3791 |
| 0.28 | 56.3641 | 59.1488 |

This is saved as PowerWorld case **B2_CLS_Infinite**. The integration method is set to Euler's on the Transient Stability, Options, Power System Model page

Generator 1 Delta: Euler's



- The below graph shows the generator angle for varying values of Δt ; numerical instability is clearly seen



Second Order Runge-Kutta



- Runge-Kutta methods improve on Euler's method by evaluating $\mathbf{f}(\mathbf{x})$ at selected points over the time step
- One approach is a second order method (RK2) in which

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \frac{1}{2}(\mathbf{k}_1 + \mathbf{k}_2)$$

where

$$\mathbf{k}_1 = \Delta t \mathbf{f}(\mathbf{x}(t))$$

$$\mathbf{k}_2 = \Delta t \mathbf{f}(\mathbf{x}(t) + \mathbf{k}_1)$$

This is also known as Heun's method or as the Improved Euler's or Modified Euler's Method

- That is, \mathbf{k}_1 is what we get from Euler's; \mathbf{k}_2 improves on this by reevaluating at the estimated end of the time step
- Error varies with the cubic of the time step

Second Order Runge-Kutta (RK2)



- Again assuming a time step $\Delta t = 0.02$ seconds, and a T^{clear} of 0.1 seconds, then using Heun's approach

$$\mathbf{x}(0) = \begin{bmatrix} \delta(0) \\ \Delta\omega_{pu}(0) \end{bmatrix} = \begin{bmatrix} 0.418 \\ 0 \end{bmatrix}$$

$$\mathbf{k}_1 = 0.02 \begin{bmatrix} 0 \\ 0.1667 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.00333 \end{bmatrix}, \quad \mathbf{x}(0) + \mathbf{k}_1 = \begin{bmatrix} 0.418 \\ 0.00333 \end{bmatrix}$$

$$\mathbf{k}_2 = 0.02 \begin{bmatrix} 1.257 \\ 0.1667 \end{bmatrix} = \begin{bmatrix} 0.0251 \\ 0.00333 \end{bmatrix}$$

$$\mathbf{x}(0.020) = \begin{bmatrix} 0.418 \\ 0 \end{bmatrix} + \frac{1}{2}(\mathbf{k}_1 + \mathbf{k}_2) = \begin{bmatrix} 0.431 \\ 0.00333 \end{bmatrix}$$

RK2 Solution Results ($\Delta t=0.02$)



- The below table gives the results using $\Delta t = 0.02$ for the beginning time steps

| Time | Gen 1 Rotor Angle, Degrees | Gen 1 Speed (Hz) |
|------|----------------------------|------------------|
| 0 | 23.9462 | 60 |
| 0.02 | 24.6662 | 60.2 |
| 0.04 | 26.8262 | 60.4 |
| 0.06 | 30.4262 | 60.6 |
| 0.08 | 35.4662 | 60.8 |
| 0.1 | 41.9462 | 61 |
| 0.1 | 41.9462 | 61 |
| 0.12 | 48.6805 | 60.849 |
| 0.14 | 54.1807 | 60.6626 |
| 0.16 | 58.233 | 60.4517 |
| 0.18 | 60.6974 | 60.2258 |
| 0.2 | 61.4961 | 59.9927 |
| 0.22 | 60.605 | 59.7598 |
| 0.24 | 58.0502 | 59.5343 |
| 0.26 | 53.9116 | 59.3241 |
| 0.28 | 48.3318 | 59.139 |

This is saved as PowerWorld case B2_CLS_Infinite. The integration method should be changed to Second Order Runge-Kutta on the Transient Stability, Options, Power System Model page

Generator 1 Delta: RK2



- The below graph shows the generator angle for varying values of Δt ; much better than Euler's but still the beginning of numerical instability with larger values of Δt

