ECEN 615
Methods of Electric Power Systems Analysis

Lecture 26: Optimal Power Flow, Security Constrained OPF, Optimization

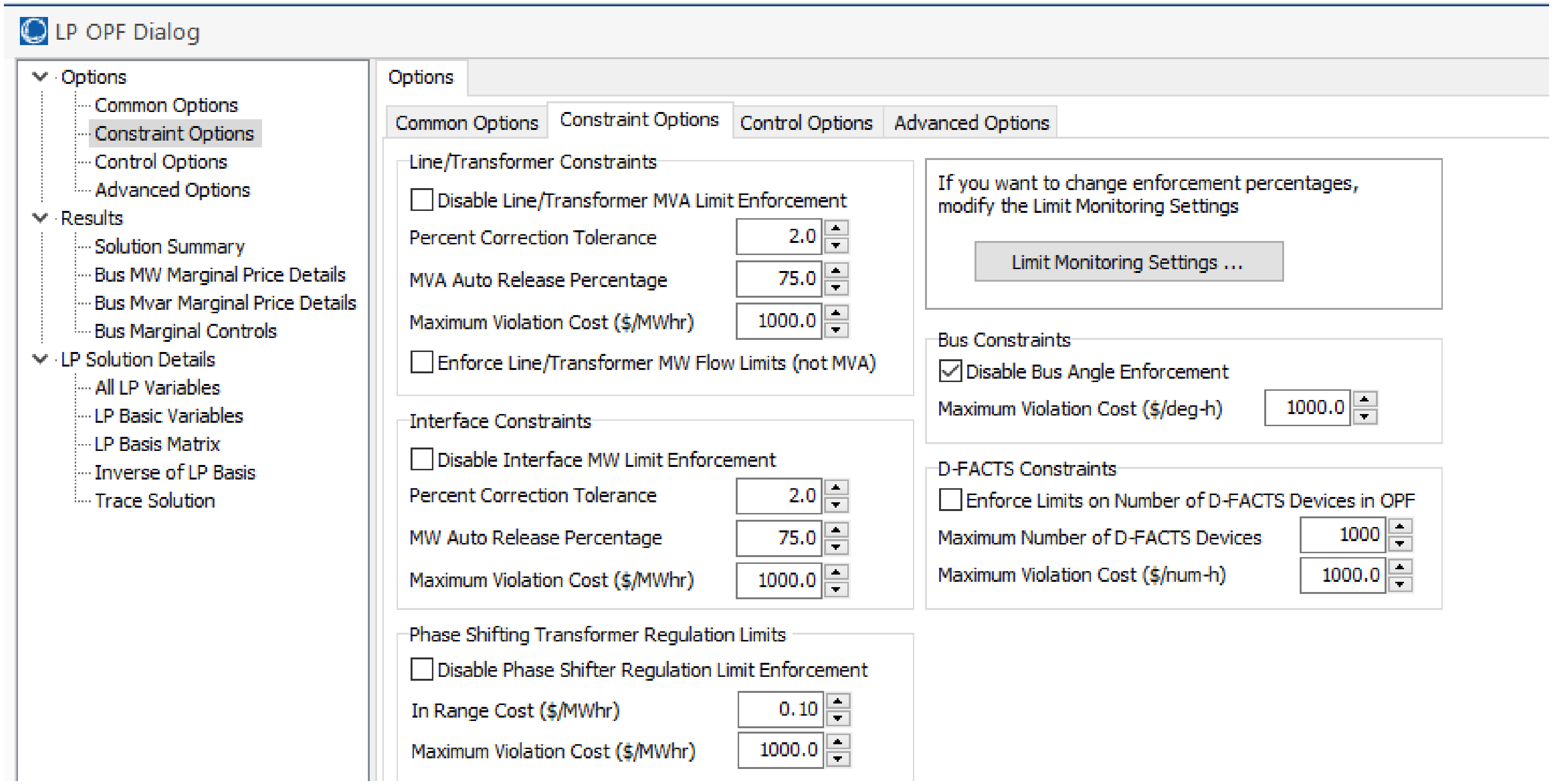
Prof. Tom Overbye
Dept. of Electrical and Computer Engineering
Texas A&M University
overbye@tamu.edu

Announcements

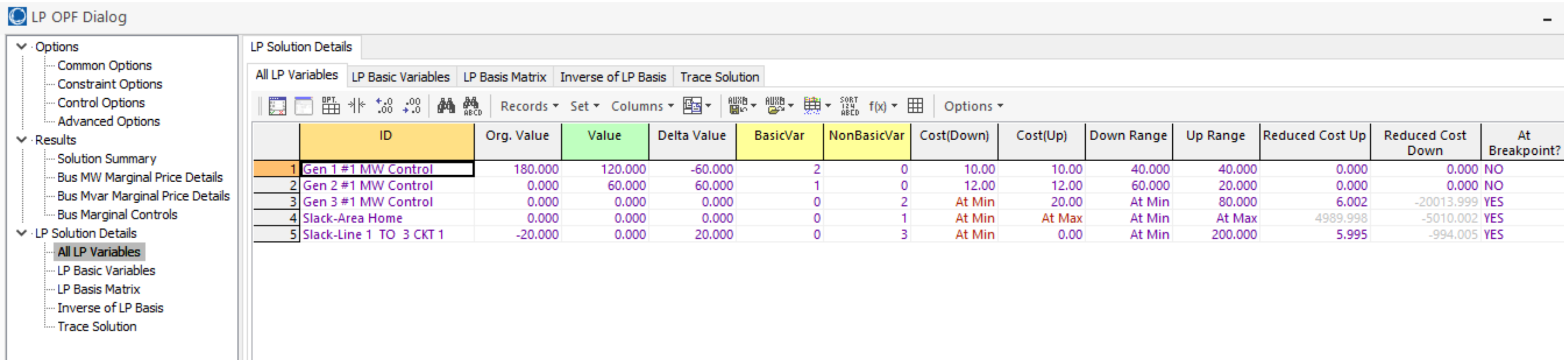
- Homework 6 is due now
- Course evaluations are now available. Goto pica.tamu.edu
  - Please do the evaluation!!
- Final exam is Wednesday Dec 12, 1 to 3pm
  - Closed book, closed notes. Two 8.5 by 11 inch notesheets allowed; calculators allowed

LP OPF Introductory Example, cont

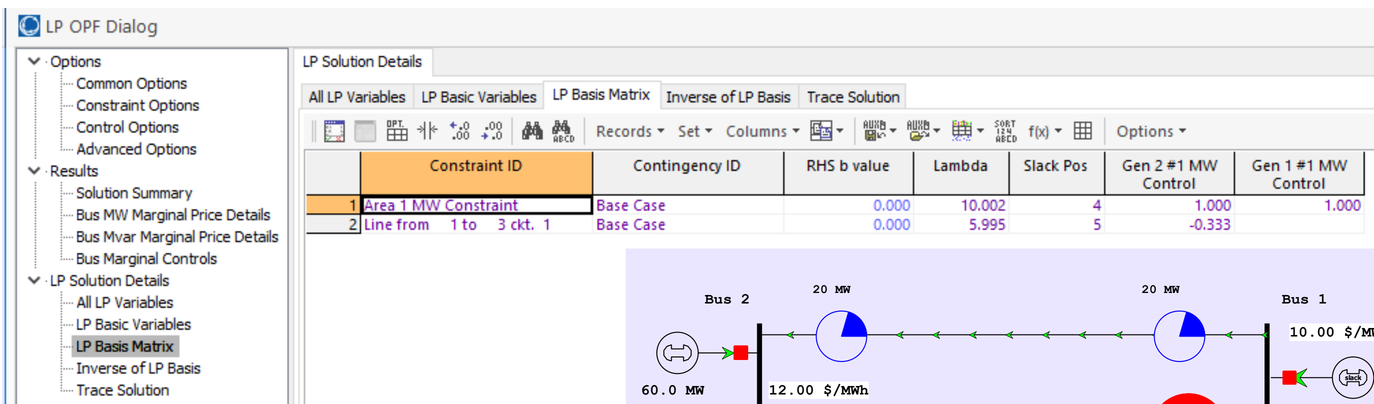
- On use **Options, Constraint Options** to enable the enforcement of the Line/Transformer MVA limits



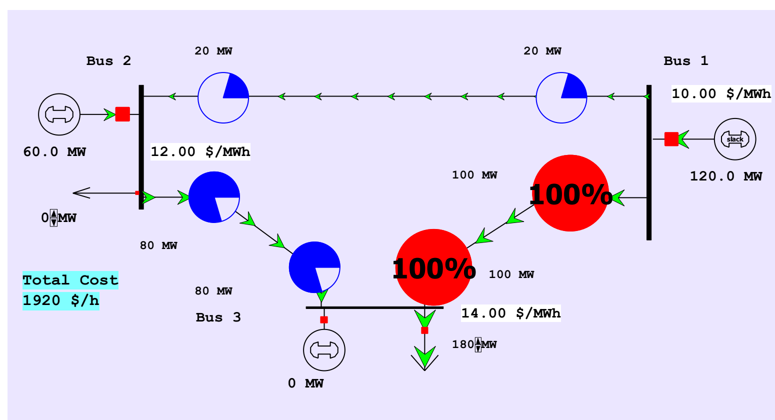
LP OPF Introductory Example, cont



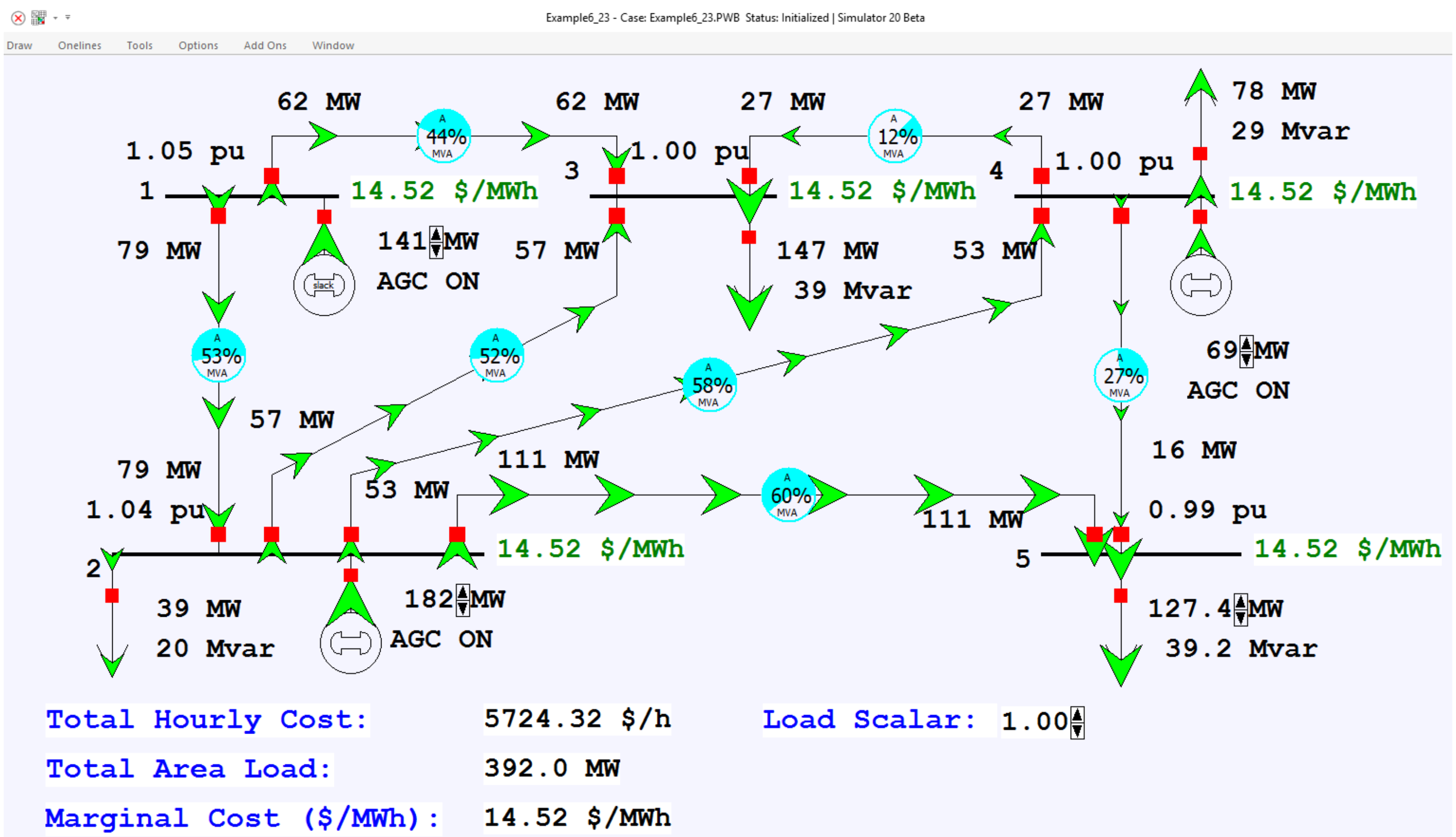
	ID	Org. Value	Value	Delta Value	BasicVar	NonBasicVar	Cost(Down)	Cost(Up)	Down Range	Up Range	Reduced Cost Up	Reduced Cost Down	At Breakpoint?
1	Gen 1 #1 MW Control	180.000	120.000	-60.000	2	0	10.00	10.00	40.000	40.000	0.000	0.000	NO
2	Gen 2 #1 MW Control	0.000	60.000	60.000	1	0	12.00	12.00	60.000	20.000	0.000	0.000	NO
3	Gen 3 #1 MW Control	0.000	0.000	0.000	0	2	At Min	20.00	At Min	80.000	6.002	-20013.999	YES
4	Slack-Area Home	0.000	0.000	0.000	0	1	At Min	At Max	At Min	At Max	4989.998	-5010.002	YES
5	Slack-Line 1 TO 3 CKT 1	-20.000	0.000	20.000	0	3	At Min	0.00	At Min	200.000	5.995	-994.005	YES



	Constraint ID	Contingency ID	RHS b value	Lambda	Slack Pos	Gen 2 #1 MW Control	Gen 1 #1 MW Control
1	Area 1 MW Constraint	Base Case	0.000	10.002	4	1.000	1.000
2	Line from 1 to 3 ckt. 1	Base Case	0.000	5.995	5	-0.333	



Example 6_23 Optimal Power Flow

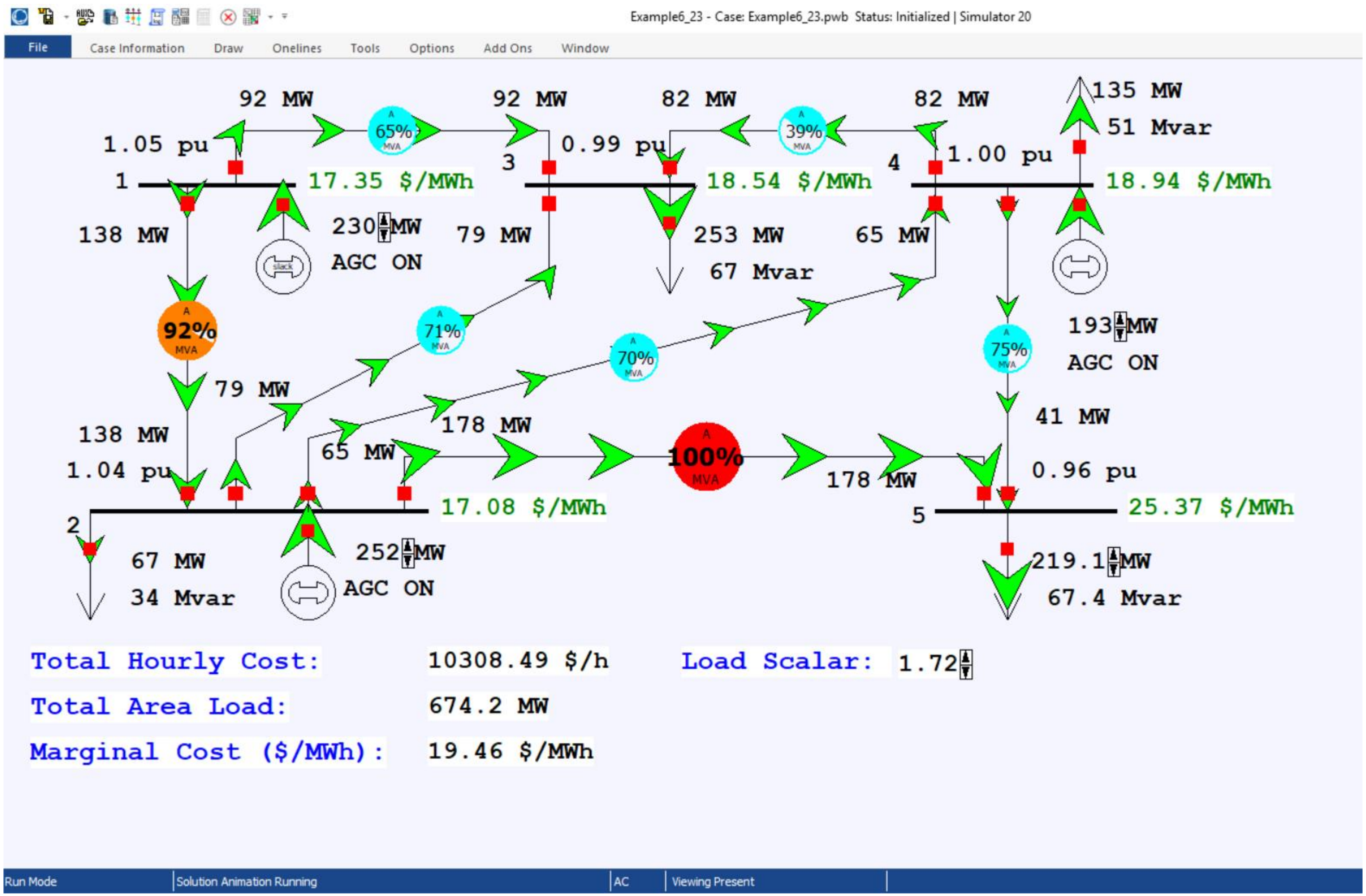


In the example the load is gradually increased

Locational Marginal Costs (LMPs)

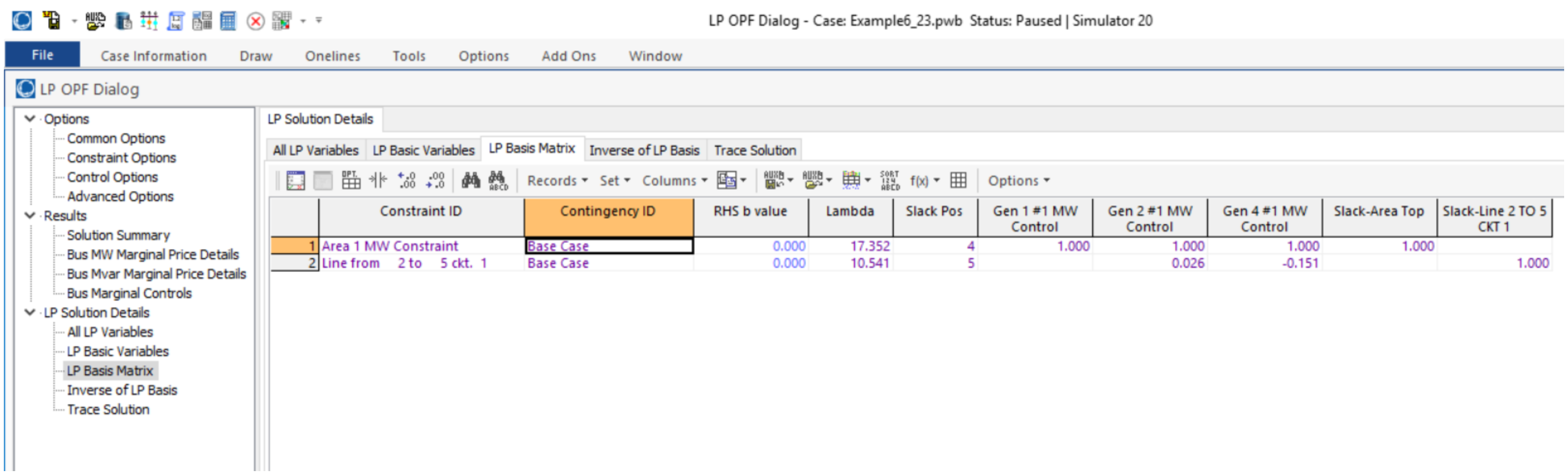
- In an OPF solution, the bus LMPs tell the marginal cost of supplying electricity to that bus
- The term "congestion" is used to indicate when there are elements (such as transmission lines or transformers) that are at their limits; that is, the constraint is binding
- Without losses and without congestion, all the LMPs would be the same
- Congestion or losses causes unequal LMPs
- LMPs are often shown using color contours; a challenge is to select the right color range!

Example 6_23 Optimal Power Flow with Load Scale = 1.72



Example 6_23 Optimal Power Flow with Load Scale = 1.72

- LP Sensitivity Matrix (**A** Matrix)



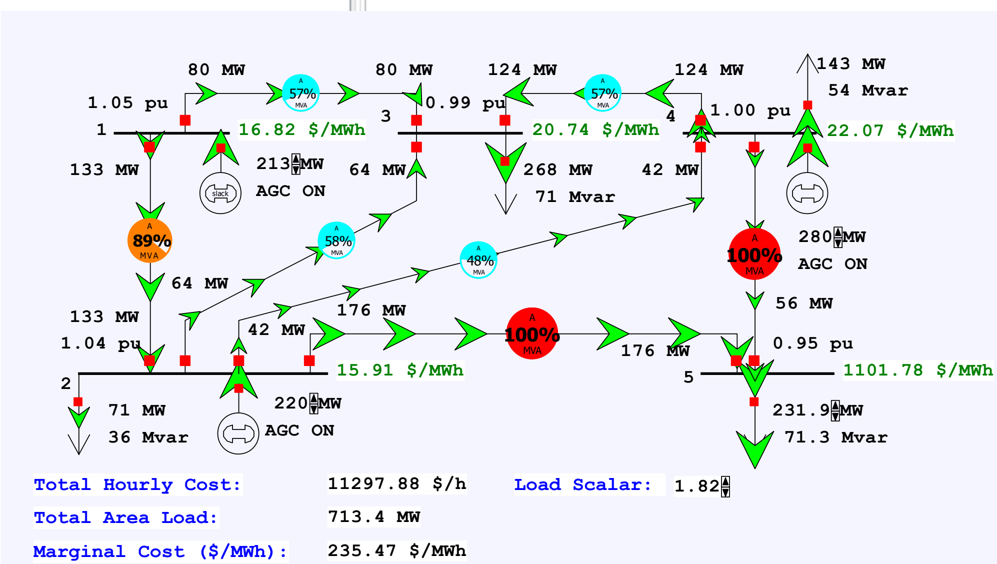
	Constraint ID	Contingency ID	RHS b value	Lambda	Slack Pos	Gen 1 #1 MW Control	Gen 2 #1 MW Control	Gen 4 #1 MW Control	Slack-Area Top	Slack-Line 2 TO 5 CKT 1
1	Area 1 MW Constraint	Base Case	0.000	17.352	4	1.000	1.000	1.000	1.000	
2	Line from 2 to 5 ckt. 1	Base Case	0.000	10.541	5		0.026	-0.151		1.000

The first row is the power balance constraint, while the second row is the line flow constraint. The matrix only has the line flows that are being enforced.

Example 6_23 Optimal Power Flow with Load Scale = 1.82

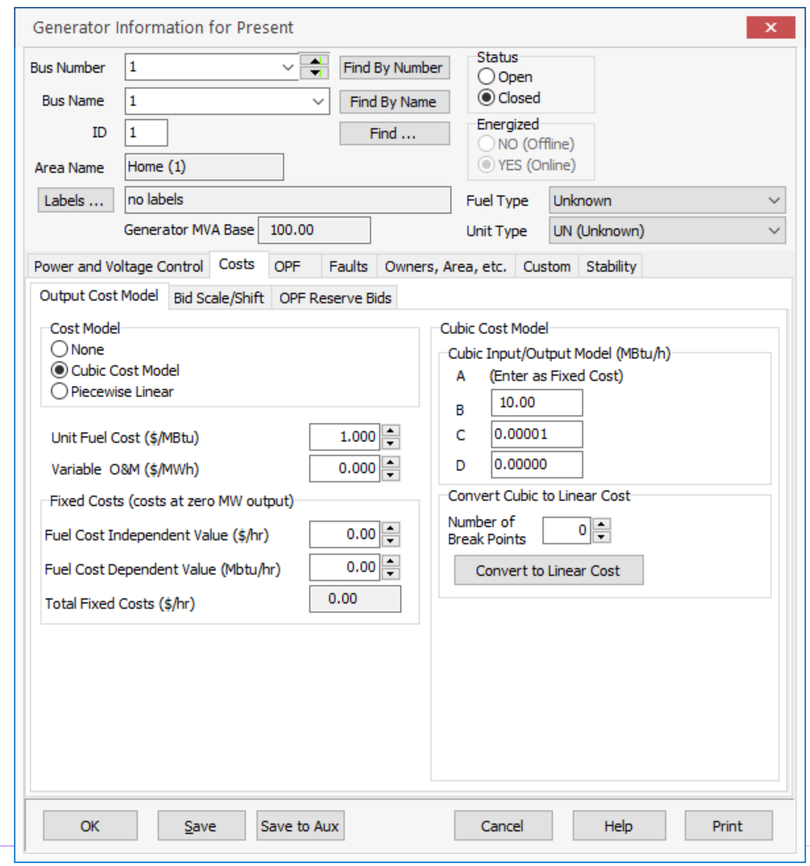
- This situation is infeasible, at least with available controls. There is a solution because the OPF is allowing one of the constraints to violate (at high cost)

						Control	Control	Control		CKT 1
1	Area 1 MW Constraint	Base Case	0.000	16.824	4	1.000	1.000	1.000	1.000	
2	Line from 2 to 5 ckt. 1	Base Case	0.000	993.664	5		0.026	-0.146		1.000
3	Line from 4 to 5 ckt. 1	Base Case	-0.002	1000.000	6		-0.024	0.140		



Generator Cost Curve Modeling

- LP algorithms require linear cost curves, with piecewise linear curves used to approximate a nonlinear cost function
- Two common ways of entering cost information are
  - Quadratic function
  - Piecewise linear curve
- The PowerWorld OPF supports both types



Security Constrained OPF

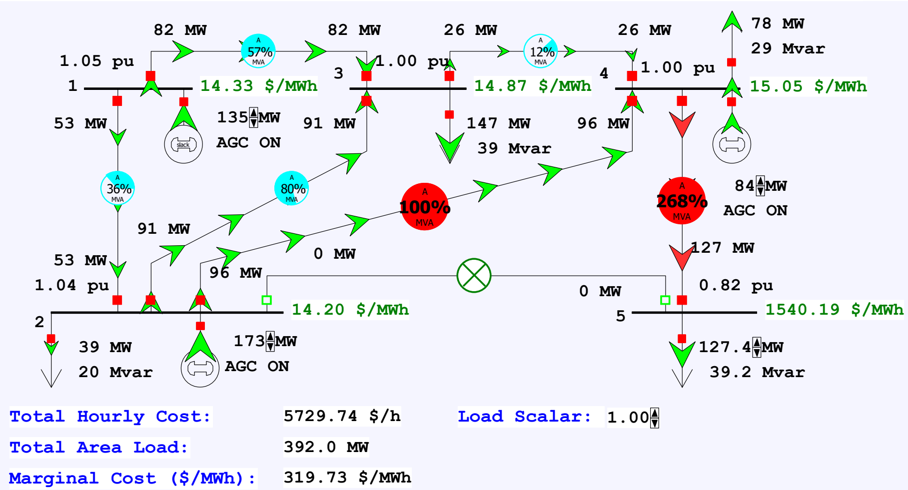
- Security constrained optimal power flow (SCOPF) is similar to OPF except it also includes contingency constraints
  - Again the goal is to minimize some objective function, usually the current system cost, subject to a variety of equality and inequality constraints
  - This adds significantly more computation, but is required to simulate how the system is actually operated (with N-1 reliability)
- A common solution is to alternate between solving a power flow and contingency analysis, and an LP

Security Constrained OPF, cont.

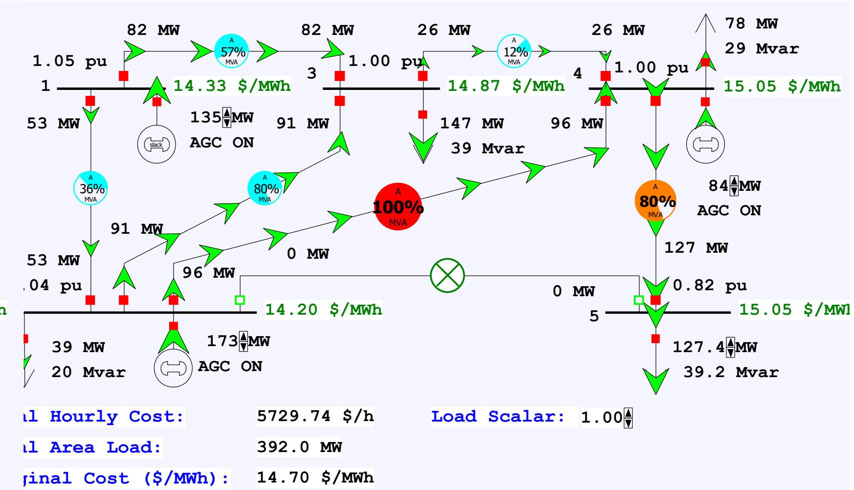
- With the inclusion of contingencies, there needs to be a distinction between what control actions must be done pre-contingent, and which ones can be done post-contingent
  - The advantage of post-contingent control actions is they would only need to be done in the unlikely event the contingency actually occurs
- Pre-contingent control actions are usually done for line overloads, while post-contingent control actions are done for most reactive power control and generator outage re-dispatch

SCOPF Example

- We’ll again consider Example 6_23, except now it has been enhanced to include contingencies and we’ve also greatly increased the capacity on the line between buses 4 and 5



Original with line 4-5 limit of 60 MW with 2-5 out

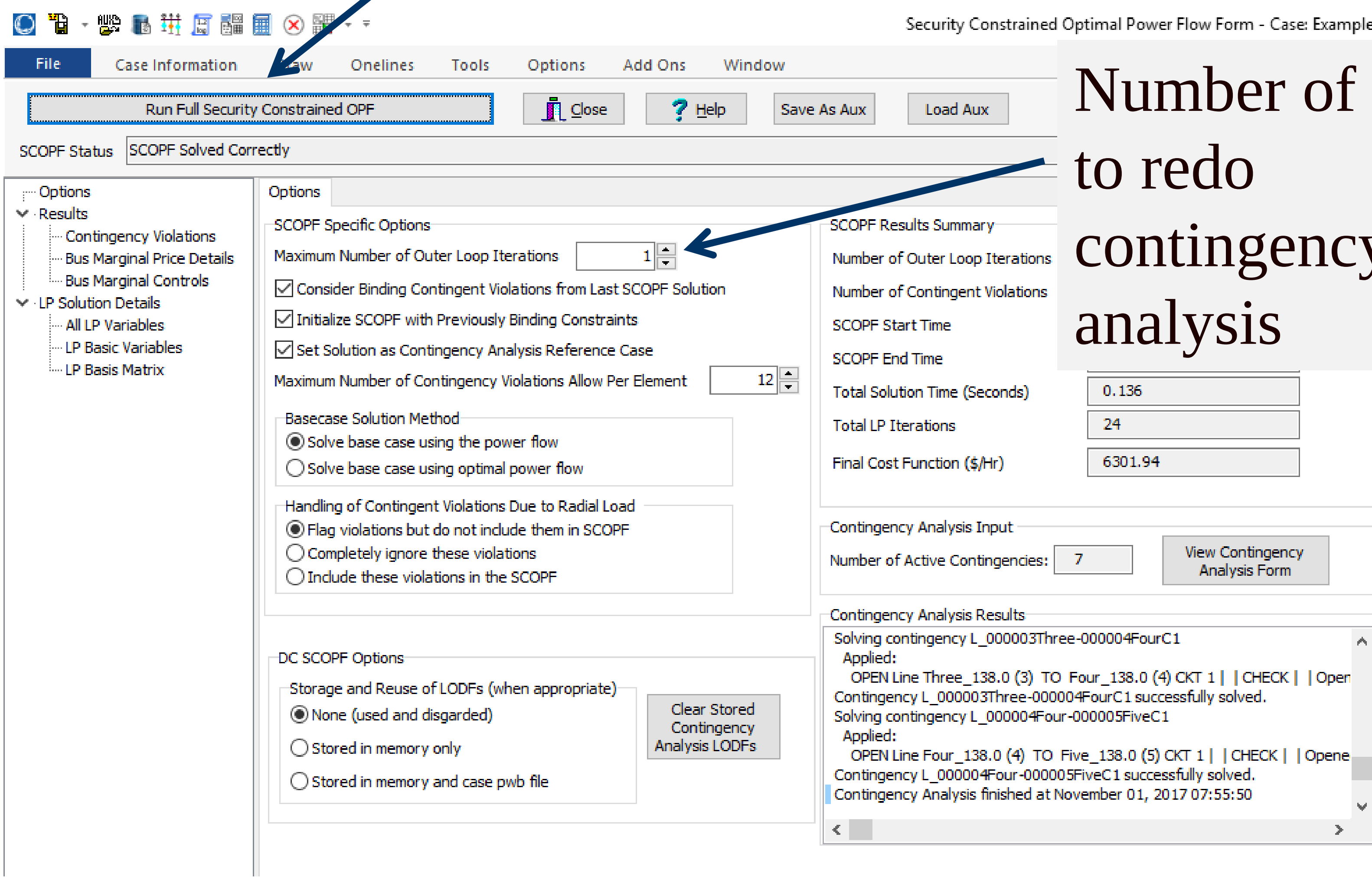


Modified with line 4-5 limit of 200 MVA with 2-5 out

PowerWorld SCOPF Application

Just click the button to solve

Number of times to redo contingency analysis



LP OPF and SCOPF Issues

- The LP approach is widely used for the OPF and SCOPF, particularly when implementing a dc power flow approach
- A key issue is determining the number of binding constraints to enforce in the LP tableau
  - Enforcing too many is time-consuming, enforcing too few results in excessive iterations
- The LP approach is limited by the degree of linearity in the power system
  - Real power constraints are fairly linear, reactive power constraints much less so

OPF Solution by Newton's Method

- An alternative to using the LP approach is to use Newton's method, in which all the equations are solved simultaneously
- Key paper in area is
  - D.I. Sun, B. Ashley, B. Brewer, B.A. Hughes, and W.F. Tinney, "Optimal Power Flow by Newton Approach", *IEEE Trans. Power App and Syst.*, October 1984
- Problem is

$$\begin{aligned} &\text{Minimize } f(\mathbf{x}) \\ &\text{s.t.} \qquad \mathbf{g}(\mathbf{x})=\mathbf{0} \\ &\qquad\qquad \mathbf{h}(\mathbf{x})\leq \mathbf{0} \end{aligned}$$

For simplicity **x** represents all the variables and we can use **h** to impose limits on individual variables

OPF Solution by Newton's Method

- During the solution the inequality constraints are either binding (=0) or nonbinding (<0)
  - The nonbinding constraints do not impact the final solution
- We'll modify the problem to split the **h** vector into the binding constraints, $\mathbf{h}_1$ and the nonbinding constraints, $\mathbf{h}_2$

$$\text{Minimize } f(\mathbf{x})$$

$$\text{s.t.} \quad \mathbf{g}(\mathbf{x}) = \mathbf{0}$$

$$\mathbf{h}_1(\mathbf{x}) = \mathbf{0}$$

$$\mathbf{h}_2(\mathbf{x}) < \mathbf{0}$$

OPF Solution by Newton's Method

- To solve first define the Lagrangian

$$L(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) = f(\mathbf{x}) + \boldsymbol{\mu}^T \mathbf{g}(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}_1(\mathbf{x})$$

Let $\mathbf{z} = \begin{bmatrix} \mathbf{x} & \boldsymbol{\mu} & \boldsymbol{\lambda} \end{bmatrix}$

- A necessary condition for a minimum is that the gradient is zero

$$\nabla L(\mathbf{z}) = \mathbf{0} = \begin{bmatrix} \dfrac{\partial L(\mathbf{z})}{\partial z_1} \\ \dfrac{\partial L(\mathbf{z})}{\partial z_2} \\ \vdots \end{bmatrix}$$

Both $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ are Lagrange Multipliers

OPF Solution by Newton's Method

- Solve using Newton's method. To do this we need to define the Hessian matrix

$$\nabla^2 L(\mathbf{z}) = \mathbf{H}(\mathbf{z}) = \left[\frac{\partial^2 L(\mathbf{z})}{\partial z_i \partial z}\right] = \begin{bmatrix} \frac{\partial^2 L(\mathbf{z})}{\partial x_i \partial x_j} & \frac{\partial^2 L(\mathbf{z})}{\partial x_i \partial \mu_j} & \frac{\partial^2 L(\mathbf{z})}{\partial x_i \partial \lambda_j} \\ \frac{\partial^2 L(\mathbf{z})}{\partial \mu_i \partial x_j} & \mathbf{0} & \mathbf{0} \\ \frac{\partial^2 L(\mathbf{z})}{\partial \lambda \partial x_{ji}} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

- Because this is a second order method, as opposed to a first order linearization, it can better handle system nonlinearities

OPF Solution by Newton's Method

- Solution is then via the standard Newton's method. That is

Set iteration counter k=0, set k_{max}

Set convergence tolerance ε

Guess $\mathbf{z}^{(k)}$

While $\left(\left\|\nabla L(\mathbf{z})\right\| \geq \varepsilon\right)$ and $\left(k < k_{max}\right)$

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} - \left[\mathbf{H}(\mathbf{z})\right]^{-1} \nabla L(\mathbf{z})$$

k = k + 1

End While

No iteration is needed for a quadratic function with linear constraints

Example

- Solve

Minimize $x_1^2 + x_2^2$ such that $x_1 + x_2 - 2 \geq 0$

Solve initially assuming the constraint is binding

$$L(\mathbf{x},\lambda) = x_1^2 + x_2^2 + \lambda(3x_1 + x_2 - 2)$$

$$\nabla L(\mathbf{x},\lambda) = \begin{bmatrix} \frac{\partial L}{\partial x_1} \\ \frac{\partial L}{\partial x_2} \\ \frac{\partial L}{\partial \lambda} \end{bmatrix} = \begin{bmatrix} 2x_1 + 3\lambda \\ 2x_2 + \lambda \\ 3x_1 + x_2 - 2 \end{bmatrix}$$

No iteration is needed so any "guess" is fine. Pick (1,1,0)

$$\nabla^2 L(\mathbf{x},\lambda) = \mathbf{H}(\mathbf{x},\lambda) = \begin{bmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 3 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} - \begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 1 \\ 1 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.2 \\ 0.4 \end{bmatrix}$$

Because λ is positive the constraint is binding

Newton OPF Comments

- The Newton OPF has the advantage of being better able to handle system nonlinearities
- There is still the issue of having to deal with determining which constraints are binding
- The Newton OPF needs to implement second order derivatives plus all the complexities of the power flow solution
  - The power flow starts off simple, but can rapidly get complex when dealing with actual systems
- There is still the issue of handling integer variables

Mixed-Integer Programming

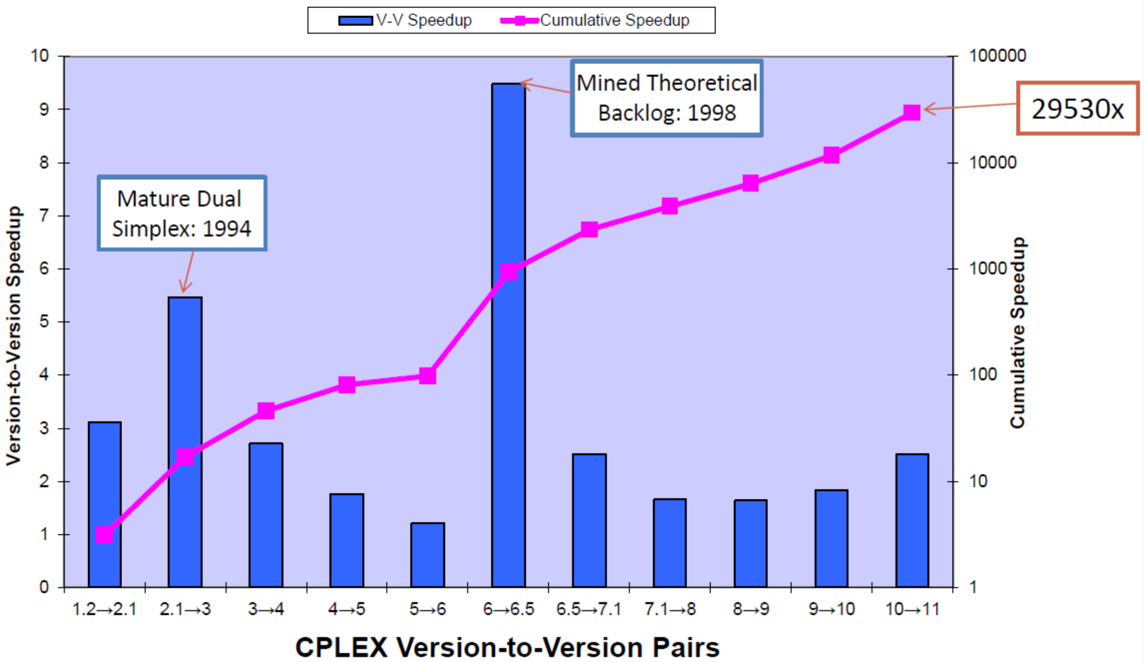
- A mixed-integer program (MIP) is an optimization problem of the form

$$
\begin{array}{ll}
\text{Minimize} & \mathbf{c}\,\mathbf{x} \\
\text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b} \\
 & \mathbf{x} \geq \mathbf{0} \\
\text{where} & \mathbf{x} = \text{n-dimensional column vector} \\
 & \mathbf{c} = \text{n-dimensional row vector} \\
 & \mathbf{b} = \text{m-dimensional column vector} \\
 & \mathbf{A} = \text{m} \times \text{n matrix} \\
 & \text{some or all } x_j \text{ integer}
\end{array}
$$

Mixed-Integer Programming

- The advances in the algorithms have been substantial

Speedups 1991-2008



Speedups from 2009 to 2015 were about a factor of 30!

Notes are partially based on a presentation at Feb 2015 US National Academies Analytic Foundations of the Next Generation Grid by Robert Bixby from Gurobi Optimization titled "Advances in Mixed-Integer Programming and the Impact on Managing Electrical Power Grids"

Mixed-Integer Programming

- Suppose you were given the following choices?
  - Solve a MIP with today's solution technology on a 1991 machine
  - Solve a MIP with a 1991 solution on a machine from today?
- The answer is to choose option 1, by a factor of approximately 300
- This leads to the current debate of whether the OPF (and SCOPF) should be solved using generic solvers or more customized code (which could also have quite good solvers!)

Notes are partially based on a presentation at Feb 2015 US National Academies Analytic Foundations of the Next Generation Grid by Robert Bixby from Gurobi Optimization titled "Advances in Mixed-Integer Programming and the Impact on Managing Electrical Power Grids"

More General Solvers Overview

- OPF is currently an area of active research
- Many formulations and solution methods exist…
  - As do many ***tools*** for highly complex, large-scale computing!

- While many options exist, some may work better for certain problems or with certain programs you already use

- Consider experimenting with a new language/solver!

Gurobi and CPLEX

- Gurobi and CPLEX are two well-known commercial optimization solvers/packages for **linear programming (LP),** quadratic programming (QP), quadratically constrained programming (QCP), and the mixed integer (MI) counterparts of LP/QP/QCP
- Gurobi and CPLEX are accessible through object-oriented interfaces (C++, Java, Python, C), matrix-oriented interfaces (MATLAB) and other modeling languages (AMPL, GAMS)

Solver Comparison

Algorithm Type ------------------ *Solver*	**LP/MILP** **linear/mixed integer linear program**	**QP/MIQP** quadratic/mixed integer quadratic program	**SOCP** second order cone program	**SDP** semidefinite program
*CPLEX**	x	x	x	
GLPK	x			
*Gurobi**	x	x	x	
IPOPT		x		
*Mosek**	x	x	x	x
SDPT3/SeDuMi			x	x

Linear programming can be solved by quadratic programming, which can be solved by second-order cone programming, which can be solved by semidefinite programming.

Modeling Tools

	AMPL	**CVX** (Matlab)	**GAMS**	**Pyomo** (Python)	**YALMIP** (Matlab)
CPLEX	**x**		**x**	**x**	**x**
GLPK				**x**	**x**
Gurobi	**x**	**x**	**x**	**x**	**x**
IPOPT	**x**			**x**	**x**
Mosek	**x**	**x**	**x**		**x**
SDPT3/SeDuMi		**x**	**x**		**x**

Introduction to AMPL

- AMPL (**A** **M**athematical **P**rogramming **L**anguage) is a modeling language that enables the compact and logical representation of optimization models
- Visit http://www.ampl.com to download and start using a student version.
  - To actually solve problems with AMPL, you'll also need a solver!
  - CPLEX is a good one to start with (already included in Windows download)
- There is an AMPL book and many examples also available for download at http://www.ampl.com

Introduction to AMPL

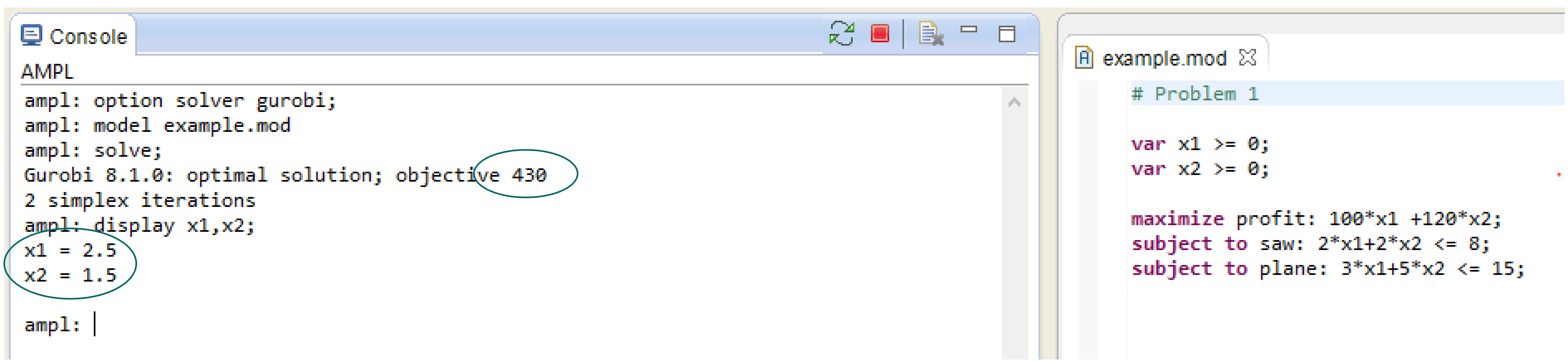
- The **model file** (.mod) declares the ***data parameters***, the *variables*, the *objective function* and the *constraints*
- The ***data*** is provided in the **data file** (.dat)
  - You don't need to change the model for every small change in the data!

- Every declaration ends in a semicolon **;**
- Comments begin with a pound sign #
- Parameters in the data file must first be declared in the model file

An AMPL Example

- Example 1 (lumber mill problem) from Lecture 24

$$
\begin{array}{ll}
\text{Maximize} & 100x_1 + 120x_2 \\
\text{s.t.} & 2x_1 + 2x_2 \le 8 \\
 & 3x_1 + 5x_2 \le 15 \\
 & x_1, x_2 \ge 0
\end{array}
$$



An (Improved) AMPL Example

- Example 1 (lumber mill problem) from Lecture 24

example_improved.mod | example_improved.dat

```
# Problem 1, improved

set B;  # Boards (products)
set R;  # Resources

param c {i in B};
param b {j in R};
param A {j in R, i in B};

var x{i in B} >= 0;

maximize profit: sum {i in B} c[i] * x[i];
subject to R_constr {j in R}: sum {i in B} A[j,i] * x[i] <= b[j];
```

example_improved.mod | example_improved.dat

```
set B := construction finish;         # grade boards
set R := saw plane;                   # resources

param c :=
construction     100
finish           120;

param b :=
saw     8
plane   15;

param A: construction   finish :=
saw         2             2
plane       3             5;
```

AMPL

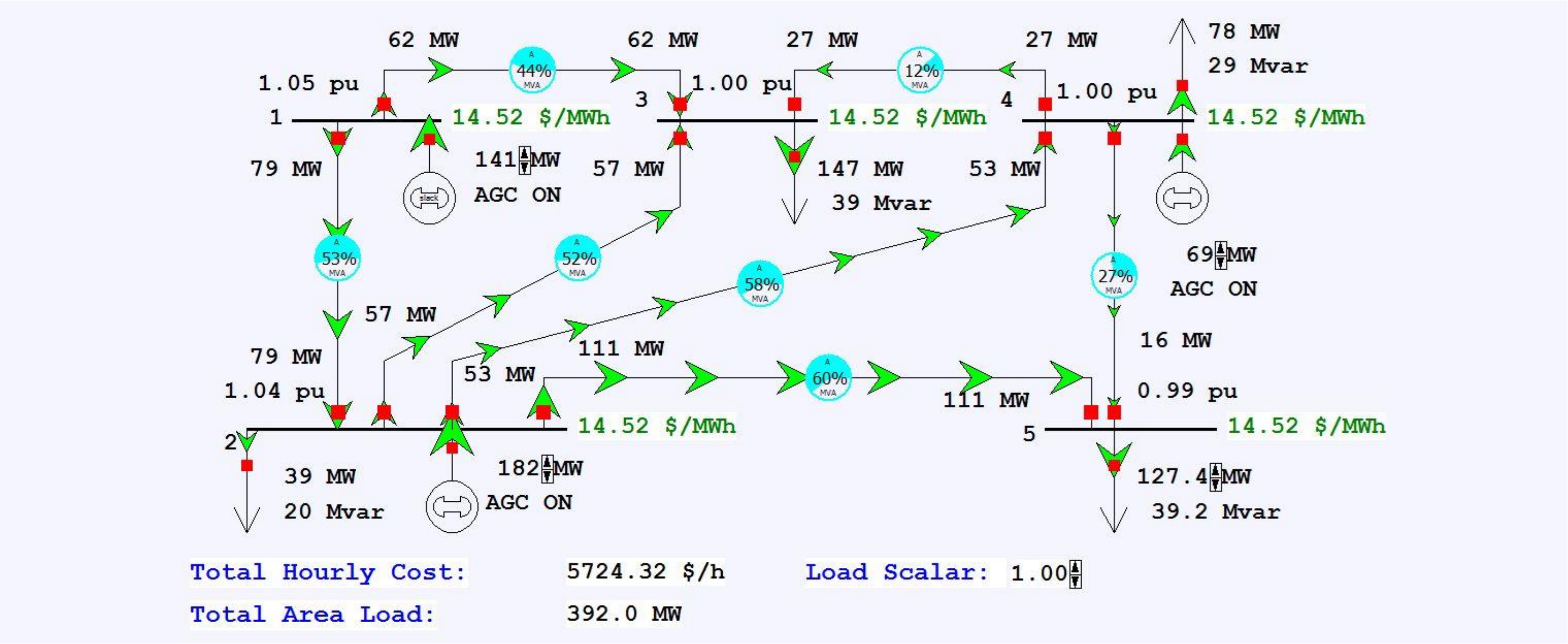
```
ampl: option solver gurobi;
ampl: model example_improved.mod
ampl: data example_improved.dat
ampl: solve;
Gurobi 8.1.0: optimal solution; objective 430
2 simplex iterations
ampl: display x, profit;
x [*] :=
construction  2.5
      finish  1.5
;

profit = 430
```

Can change the values in the .dat file, change solver, etc. with just a few quick clicks!

DCOPF in AMPL

- Example 6_23



DCOPF in AMPL: Parameters

- LOTS of parameters (not all are used in this example, but the framework is there for more involved examples)

case4.mod

```
set BUS;
set BR within {1..4000} cross BUS cross BUS;
set GEN;

## Parameters ##
#param bus_number    {BUS};
param bus_type       {BUS};
param bus_p_load     {BUS};
param bus_q_load     {BUS};
param bus_g_shunt    {BUS};
param bus_b_shunt    {BUS};
param bus_area       {BUS};
param bus_voltage    {BUS};
param bus_angle0     {BUS};
param base_volt      {BUS};
param loss_zone      {BUS};
param vmax           {BUS};
param vmin           {BUS};
param lam_p          {BUS};
param lam_q          {BUS};
param mu_vmax        {BUS};
param mu_vmin        {BUS};
```

```
param f_bus {BR};
param t_bus {BR};
param br_type {BR};
param br_r  {BR};
param br_x  {BR};
param br_b  {BR};
param rate_a {BR};
param rate_b {BR};
param rate_c {BR};
param tap   {BR};
param shift {BR};
param br_status {BR};
param angmin    {BR};
param angmax    {BR};
param pf    {BR};
param qf    {BR};
param pt    {BR};
param qt    {BR};
param mu_sf {BR};
param mu_st {BR};
param mu_angmin {BR};
param mu_angmax {BR};
```

```
param gen_bus {GEN};
param pg0       {GEN};
param qg0       {GEN};
param qmax      {GEN};
param qmin      {GEN};
param vg        {GEN};
param mbase     {GEN};
param gen_status {GEN};
param pmax      {GEN};
param pmin      {GEN};
#param pc1      {GEN};
#param pc2      {GEN};
#param qc1min   {GEN};
#param qc1max   {GEN};
#param qc2min   {GEN};
#param qc2max   {GEN};
#param ramp_agc {GEN};
#param ramp_10  {GEN};
#param ramp_30  {GEN};
#param ramp_q   {GEN};
#param apf      {GEN};
#param mu_pmax  {GEN};
#param mu_pmin  {GEN};
#param mu_qmax  {GEN};
#param mu_qmin  {GEN};
```

DCOPF in AMPL: Parameters, cont.

```
param deg2rad := 3.14156/180;
param base_mva := 100;

# Make YBUS
set YBUS := setof{i in BUS} (i,i)        union
            setof {(l,k,m) in BR} (k,m) union
            setof {(l,k,m) in BR} (m,k);

param B{(k,m) in YBUS} := if(k ==m)     then (sum{(l,k,i) in BR} (0)
                                                +sum{(l,i,k) in BR} (0))
                          else if(k != m)   then (sum{(l,k,m) in BR} 1/br_x[l,k,m]
                                                    +sum{(l,m,k) in BR} 1/br_x[l,m,k]);
```

- Initialization

```
# data init

for{i in BUS} {
   let bus_angle[i]    := bus_angle0[i]*deg2rad;
   let pg0[i]    := pg0[i]/base_mva;
   let pmin[i]   := pmin[i]/base_mva;
   let pmax[i]   := pmax[i]/base_mva;
   let bus_p_load[i]   := bus_p_load[i]/base_mva;
  };

# fixing variables

fix {i in BUS : bus_type[i] == 3} bus_angle[i];                        # slack angle fixed
```

DCOPF in AMPL: Model

- Objective: Polynomial

```
## Available Controls
var bus_angle   {BUS};
var pflow       {BR}; # used for output

# generator MW outputs
var pg {k in BUS} = bus_p_load[k]
                    + sum{(k,m) in YBUS} (B[k,m]*(bus_angle[k] - bus_angle[m]));

## Objective ##
minimize cost: 0.016*(pg[1]*base_mva)^2+10*pg[1]*base_mva+373.50 +
                0.018*(pg[2]*base_mva)^2+8*pg[2]*base_mva+403.60 +
                0.018*(pg[4]*base_mva)^2+12*pg[4]*base_mva+253.20;

## Equality Constraints
# bus real power balance
subject to pL {k in BUS}: pg[k] - bus_p_load[k]
                        - sum{(k,m) in YBUS} (B[k,m]*(bus_angle[k] - bus_angle[m])) = 0;

## Inequality Constraints
# generator MW limits
subject to pG {k in BUS}:
  pmin[k] <=  pg[k] <= pmax[k];
```

DCOPF in AMPL: Data

- Bus, Branch, and Generator data saved in .txt files

```
data;

param: BUS: bus_type bus_p_load bus_q_load
        bus_g_shunt bus_b_shunt bus_area bus_voltage bus_angle0
        base_volt loss_zone vmax vmin lam_p lam_q mu_vmax mu_vmin :=
include 'C:\Users\Documents\AMPL\amplide.mswin64\amplide\busdat.bus.txt';

data;
param: BR: br_r br_x br_b
        rate_a rate_b rate_c tap shift br_status angmin angmax
        pf qf pt qt mu_sf mu_st mu_angmin mu_angmax :=
include 'C:\Users\Documents\AMPL\amplide.mswin64\amplide\brdat.br.txt';

data;
param: GEN: pg0 qg0 qmax qmin vg mbase gen_status
        pmax pmin #pc1 pc2 qc1min qc1max qc2min qc2max
        #ramp_agc ramp_10 ramp_30 ramp_q apf mu_pmax mu_pmin mu_qmax mu_qmin
:= include 'C:\Users\Documents\AMPL\amplide.mswin64\amplide\gendat.gen.txt';
```

case4.mod | gendat.gen.txt

```
1   141.31  15.03   9900.00 -9900.00    1.0500  100.00  1    400.00  100.00
2   181.50  70.93   9900.00 -9900.00    1.0400  100.00  1    500.00  150.00
3   0.0     0.0     0.0     0.0         0.0     0.0     0.0  0.0     0.0
4   69.19   6.26    9900.00 -9900.00    1.0000  100.00  1    300.00  0.00
5   0.0     0.0     0.0     0.0         0.0     0.0     0.0  0.0     0.0
```

DCOPF in AMPL: Solution

```
Gurobi 8.1.0: optimal solution; objective 5724.293529
7 barrier iterations
No basis.
bus_angle [*] :=
1    0.130103
2    0.082881
3   -0.0196945
4   -0.0114151
5   -0.0504711
;
```

Ang (deg)
7.454
4.748
-1.128
-0.654
-2.892

AMPL+Gurobi DCOPF with polynomial

```
MATPOWER Version 4.1, 14-Dec-2011 -- AC Optimal Power Flow
MATLAB Interior Point Solver -- MIPS, Version 1.0, 07-Feb-2011
Converged!

Converged in 0.47 seconds
Objective Function Value = 5724.29 $/hr
```

Bus #	Voltage Mag(pu)	Ang(deg)
1	1.020	7.454*
2	1.021	4.846
3	1.004	-0.916
4	1.012	-0.448
5	0.983	-2.763

MATPOWER ACOPF with polynomial objective function

Angle (Deg)
7.45
4.97
-0.73
-0.26
-2.48

PowerWorld OPF solution

Total Hourly Cost: 5724.32 $/h

MATPOWER

- Uses data stored in MATLAB struct

```
>> runopf(Example6_23)

MATPOWER Version 4.1, 14-Dec-2011 -- AC Optimal Power Flow
MATLAB Interior Point Solver -- MIPS, Version 1.0, 07-Feb-2011
Converged!

Converged in 1.13 seconds
Objective Function Value = 5724.29 $/hr
================================================================================
|     System Summary                                                           |
================================================================================

How many?                How much?              P (MW)            Q (MVAr)
---------------------    -------------------  -------------  -----------------
Buses              5     Total Gen Capacity    1200.0       -29700.0 to 29700.0
Generators         3     On-line Capacity      1200.0       -29700.0 to 29700.0
Committed Gens     3     Generation (actual)    392.0              93.0
Loads              4     Load                   392.0             127.4
  Fixed            4       Fixed                392.0             127.4
  Dispatchable     0       Dispatchable          -0.0 of -0.0      -0.0
Shunts             0     Shunt (inj)             -0.0               0.0
Branches           7     Losses (I^2 * Z)         0.00             41.01
Transformers       0     Branch Charging (inj)     -               75.4
Inter-ties         0     Total Inter-tie Flow     0.0               0.0
Areas              1
```

MATPOWER Solvers

- Section 6.5 of manual
  - Originally used MATLAB's Optimization Toolbox
  - Now can use MINOPF/TSPOPF packages, IPOPT solver (open-source), CPLEX/MOSEK/Gurobi (for DC OPF), KNITRO (for AC OPFs)
  - Default: own primal-dual interior point method implementation MIPS (MATPOWER Interior Point Solver) for AC and DC (QP solver)

http://www.pserc.cornell.edu/matpower/

Questions

- What changes would you make to the .mod file to do a full ACOPF?
- What sensitivity analysis could you do by only changing the .dat/.txt files?
- What might you consider when comparing solvers?

- **What tools best fit your needs?**