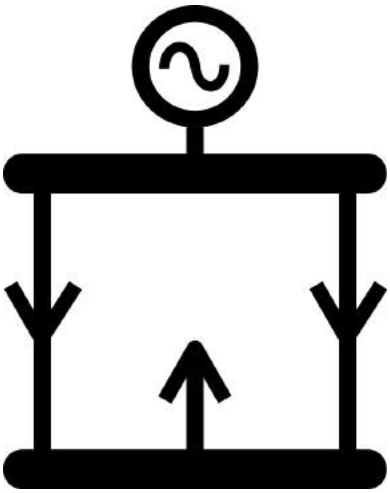
Electricity Markets and the Simplex Algorithm



With worked examples using the Simplex Nodal app

Introduction

This document explains nodal electricity markets and the simplex algorithm, supported by worked examples.

The first section provides an overview of electricity markets and electricity networks. This overview is followed by a series of tutorials that add detail by building models of electricity markets and electricity networks and solving them using the simplex algorithm. You can build and solve the models yourself by using the Simplex Nodal app, which is free software available for iPhone, iPad or Mac.

The first tutorial demonstrates how a very simple nodal electricity market can be represented mathematically as a linear programming model and solved using the simplex algorithm. When the model is solved, the results are explained.

The remaining tutorials follow a similar pattern of demonstrating features of electricity markets by building and solving example models, and then explaining the results. Electricity market features that are covered include the modelling of the transmission system, line losses, parallel flows, binding constraints, the spring washer effect, risk

and reserve, ramp rates, HVDC links, HVDC risk and the risk subtractor, HVDC reserve sharing, and modelling part of an actual electricity market.

In every tutorial the feature being explained is expressed in terms of the mathematical model that can be solved by the simplex algorithm. The final tutorial explains the simplex algorithm itself in detail, covering the initial tableau, basic and non-basic variables, the steps that the simplex algorithm takes in order to improve the objective value, and how the results are extracted from the final tableau.

The remainder of the document provides a user guide for the Simplex Nodal app, with a section that describes in how to use the app to build and solve electricity market models in general, followed by a section that describes how to use all of the controls and displays.

Electricity Networks and Markets

This section describes electricity markets and electricity networks, to provide background before we start building the models.

Electricity Networks

Generation and load

A generator produces electricity. A load consumes electricity. The electricity is transmitted from the generation location to the load location via the transmission system.

Power, voltage and current

The power of the electricity is the product of the quantity of electrical flow (the current) and the pressure of that flow (the voltage).

Transmission voltage

As the electricity flows through the conductors (e.g., wires) the electrons interact with the material of the conductor (e.g., copper) and some of the power is lost as heat.

The more electricity that is flowing, i.e., the higher the current, the higher the losses. Therefore, before the power is sent over long distances its voltage is

increased, which will allow the same amount of power to be sent with a lower current, thereby decreasing the losses.

High voltage electricity can jump across an air gap therefore it needs to be transported either by cables with very good insulation, or by towers where the wires are kept well away from the ground and from each other. These transportation mechanisms are expensive, hence high voltages are only used when long distances are involved and the corresponding reduction in losses is significant. The transport of electricity over long distances at high voltages is referred to as transmission.

Transformers

After the electricity is generated, it is increased to a higher voltage in preparation for transmission.

The voltage of the electricity is changed using transformers. Inside the transformer the electricity passes through a coil of wire, this creates an electromagnetic field that induces electricity in an adjacent coil of wire within the transformer.

If the adjacent coil has more windings, then it will have a higher voltage across the end of its windings than the voltage across the windings of the first coil;

if the adjacent coil has fewer windings then its voltage will be lower.

Transmission system

The transmission system transmits the power to the load location, where a transformer decreases the voltage to a level that can be safely and easily distributed to the end consumer. The power is distributed to the end consumer via the distribution network.

The transmission system consists of the lines, cables, and transformers that transmit the power from the generators to the distribution network.

Scope of the electricity market

The electricity market models the generators, the transmission system and the load. The location of the load is not the location of the actual load but rather the point at which the power exits the transmission system and enters the distribution network.

Branches

In the electricity market model the lines, cables and transformers that transmit the electricity are referred to as branches.

Substations

The transmission system also includes substations. The substations are the nodes on the network where power can be sent off in different directions. The substation is also where the power from the generation enters the transmission system and where the power for the load leaves the transmission system.

Buses

Within a substation there are one or more busbars that act as common connection points. Physically a busbar may consist of steel or copper bars, or it may be a short run of overhead wires.

A busbar is usually referred to as a bus. Because a substation can contain several buses operating at different voltages, it is the buses rather than the substations themselves that function as nodes in the electricity market model.

The power system and the System Operator

The power system consists of the generators that produce the electrical power, the transmission system that transports the power, and the load that consumes the power. The purpose of the power system is to meet the power requirements of the load.

The System Operator is the entity responsible for ensuring that the power system achieves its purpose. Hence, the System Operator is responsible for ensuring that there is always enough generation available to meet the power requirements of the load, and that the transmission system is capable of transporting the necessary power.

System security

Part of ensuring that the power requirements can be met it is ensuring that the power system is secure. For example, the System Operator enforces system security by ensuring that the power flow does not exceed the maximum capacity of the branches, and that, as far as possible, there is no single event that could result in the power requirements of the load not being met. These requirements are enforced by the electricity market model.

Network model and Power flow

The System Operator assesses system security by using a network model. The network model is a computer representation of the power system, complete with generators, branches, buses and loads.

The network model is combined with power flow software that simulates the physical rules of power flow. The power flow software runs a power flow study to assess the security of the power system, e.g., to confirm that branches do not exceed their limits.

Electricity Markets

Generator offers

Generators in an electricity market submit offers to generate the power that will meet the requirements of the load. The offers consist of a price and a quantity.

Load

The majority of load will be consumed regardless of price. This load is referred to as required load. There is also some price-sensitive load, which will only be consumed if the price is below a certain level.

Price sensitive load is bid with a price and quantity. Required load can either be modelled as a constraint requiring that the load be supplied, or as a load with a bid price that is higher than any possible generation offer price.

Units of measure

An electricity market generally covers a large area such as a state or a country. The load requirement is in the order of millions of watts, i.e., megawatts, abbreviated as MW.

Offer quantity and price

The generator's offer quantity is specified in MW, whereas the price is $/MWh. The MW represents the maximum quantity of power that can be provided at the specified price. The MWh associated with the price represents the metered energy, with one MWh being one MW of power supplied for an hour. Bids also use MW for quantity and $/MWh for price.

Time intervals and schedules

The results of the electricity market model provide a schedule of generation to supply load over a specific interval of time. This time interval is referred to as a trading interval or a trading period.

For forecast schedules, i.e., schedules that predict future prices and quantities, a trading period of 30 minutes or 60 minutes is common, and the schedule will contain multiple trading periods, e.g., 48 half hour trading periods to cover a day.

For schedules that are solved to meet the load requirements in real time, a time interval of 5 minutes is common.

Nodal electricity markets

An electricity market could clear generation offers in order to meet load without considering how the generation would reach the load.

However, in nodal electricity markets the electricity market model includes a network model and a power flow, similar to that used by the System Operator to study the Power System. In the nodal electricity market, the generation offers that are cleared are modelled as reaching the load by obeying the mathematical rules of power flow.

When the nodal electricity market model is solved, the result is a schedule of nodal prices and generation quantities. These prices and quantities are calculated based on the price of the generation offers combined with the requirements of the power flow.

Tutorial 1: Explaining Prices

This tutorial explains how the electricity market is represented as a Linear Programming (LP) mathematical model and how solving the LP model produces the electricity prices.

Default parameters and settings

Most of the parameters used in the following tutorials are the default values, as shown in Figure 1, Figure 2 and Figure 3. If you have set other default values and want to reset to the original values, then you can use the "Reset Default Values" button on the Settings display.

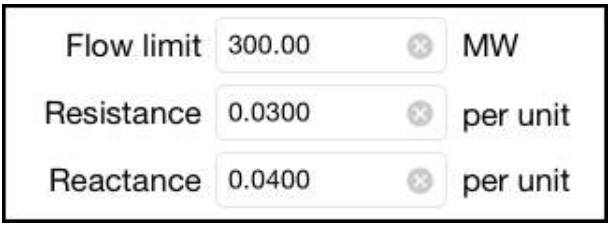


Figure 1: Default branch parameters

Figure 2: Default energy offers

Figure 3: Default load bids

Unless otherwise stated, the Solve Settings used by the worked examples are assumed to be those shown in Figure 4. There will be changes later on, for example when line losses are examined the instructions will specify that Include Losses is set to ON, but Figure 4 is our starting point.

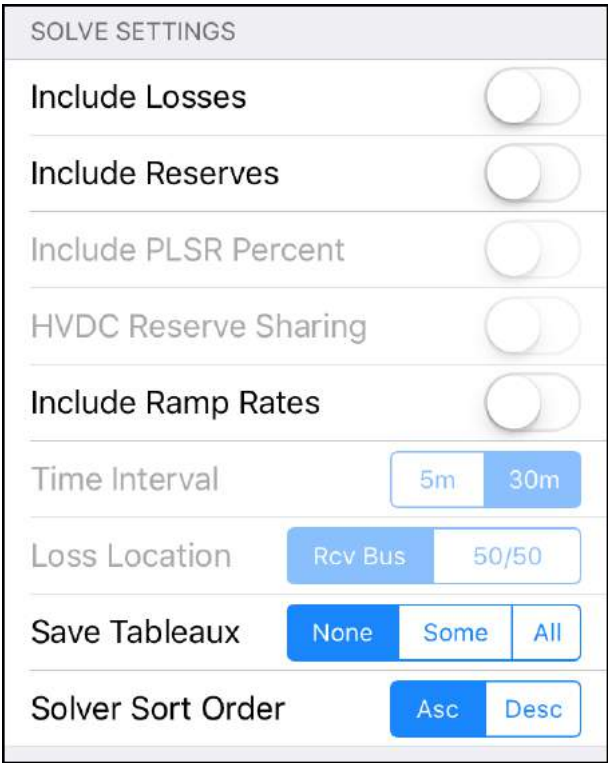


Figure 4: Solve Settings for tutorial 1

Build the model

Build the electricity market model for tutorial 1 by tapping the following buttons on the Build menu: Bus-Gen-Load.

Leaving the default parameters in place, solve the model by tapping the Solve button, checking that all solve options are selected OFF, then tapping the Solve Now button. The software builds a

mathematical model of your electricity market and then solves that model using the simplex algorithm. The resulting prices and quantities are displayed in Figure 5.

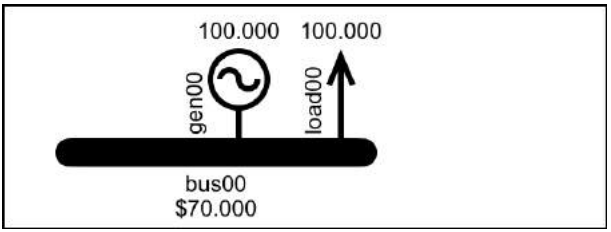


Figure 5: Small electricity market

Linear programming

The mathematical model of the electricity market is a Linear Programming model, referred to as an LP model.

The following excerpt from a website biography of George Dantzig explains the origin of the term "Linear Programming":

In 1947 George Dantzig made the contribution to mathematics for which he is most famous, the simplex method of optimisation. It grew out of his work with the U.S. Air Force where he become an expert on planning methods solved with desk calculators. In fact this was known as "programming", a military term that, at that time, referred to plans or schedules for training, logistical supply or deployment of men.

Dantzig mechanised the planning process by introducing "programming in a linear structure", where "programming" has the military meaning explained above.

Objective function and constraints

The general form of an LP model consists of an equation that defines the objective of the model and a series of equation constraints that represent the behaviour of the system being modelled. As the simplex algorithm pursues the objective, it must ensure that the requirements of the constraints are met. This concept is illustrated in Figure 6.

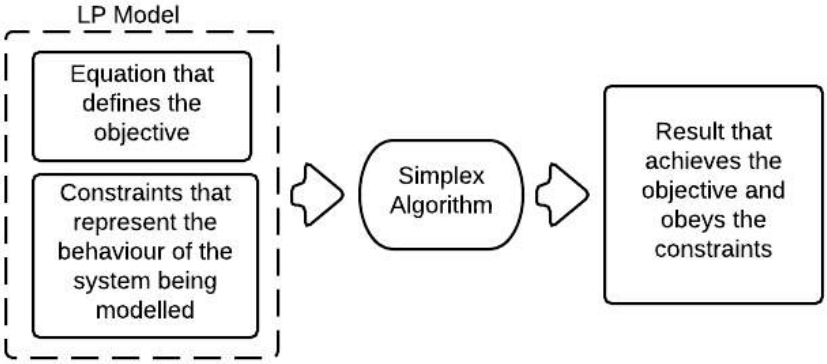


Figure 6: LP Model, solved using the Simplex Algorithm

Variables and constraints

The constraints that model the behaviour of the electricity market are associated with the

components of the model, i.e., the buses, branches, generators and loads.

To view the constraints for bus00, double-tap bus00 to view its Data Display, then on the Data Display tap the ∑ button indicated in Figure 7. This will take you to the variables and constraints display shown in Figure 8.

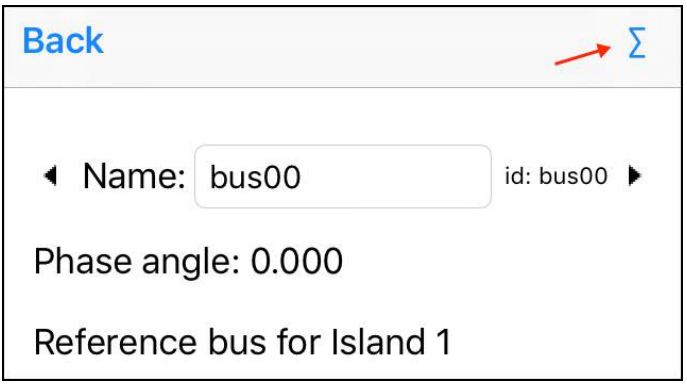


Figure 7: Data Display for bus00 with the "variables and constraints" button indicated

Figure 8: Variables and constraints display for bus00

Node balance constraint

In this single-bus model there are no variables associated with bus00. However, there is a constraint, which is the node balance constraint. The general form of the node balance constraint is shown in Equation 1. This constraint models the physical reality that the power flowing into the bus must equal the power flowing out.

$$\sum Bus_{FlowIn} - \sum Bus_{FlowOut} = 0$$

Equation 1: Node balance constraint

When we clicked the "Solve Now" button the software created constraints to represent the physical reality of the model. Figure 8 shows that the implementation of the node balance constraints ensures that the cleared generation of gen00 matches the cleared bids of load00.

The constraint presented in Equation 1 is an equality constraint (i.e., *a* equals *b*), but the simplex algorithm requires that all constraints are expressed as <= equations. Hence, the node balance constraint is expressed as a combination of <= and >= constraints, which together have the same effect as a single equality constraint… then the >= constraint has its signs reversed, to become a <= constraint.

Bid and Offer constraints

Bids and offers consist of a maximum quantity and a price. The portion of the maximum quantity that is scheduled is the cleared quantity. A constraint is required to limit the cleared quantity to be no more than the maximum quantity. These constraints are shown in Equation 2 and Equation 3.

$$ClearedQuantity_{LoadBid} \leq MaxQuantity_{LoadBid}$$

Equation 2: Cleared bid constraint, i.e., bid max constraint

$$ClearedQuantity_{GenOffer} \leq MaxQuantity_{GenOffer}$$

Equation 3: Cleared offer constraint, i.e., offer max constraint

You can view these constraints via the "Variables and Constraints" displays for load00 and gen00, as shown in Figure 9 and Figure 10. As with the bus, these displays are accessed by double tapping the component to access the Data Display, then tapping the ∑ button in the toolbar.

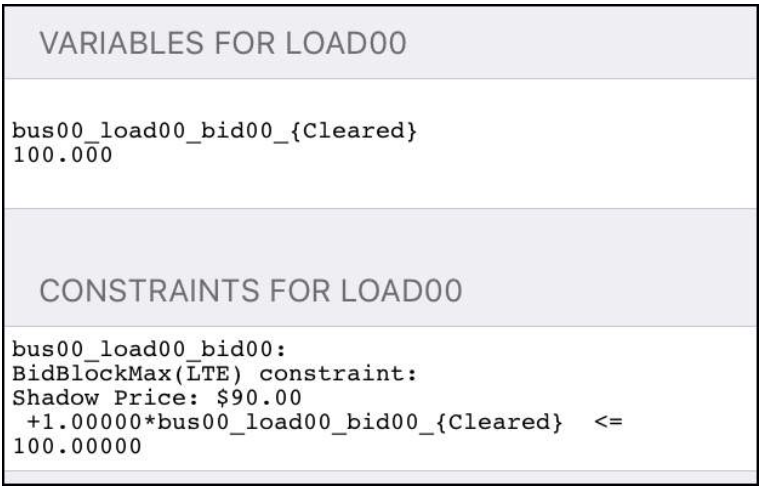


Figure 9: Variables and constraints for load00

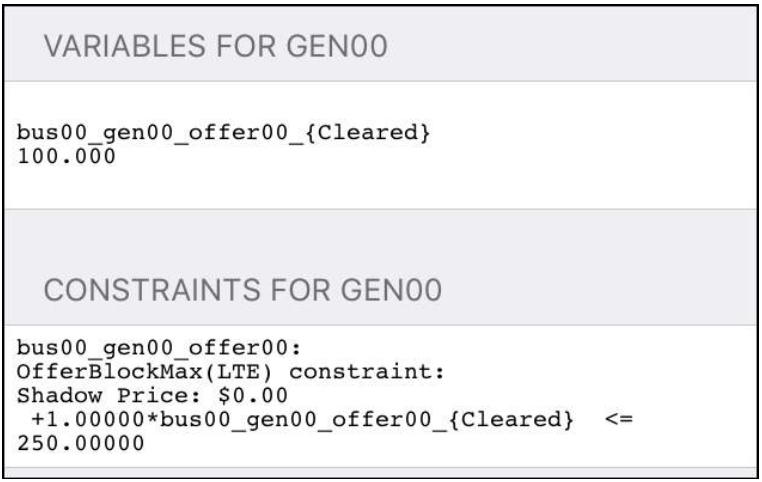


Figure 10: Variables and constraints for gen00

The Objective Function

The LP model consists of constraints representing the behaviour of the system being modelled, and an equation that defines the objective value. Maximising the objective value is what drives the actions of the simplex algorithm.

The equation that defines the objective value is called the objective function. Equation 4 describes the objective function for the electricity market model. This shows that the objective of solving the electricity market model is to maximise the benefit of the cleared bids while minimizing the cost of the cleared offers.

Maximize: $ObjectiveValue$
$$= loadBid_{Cleared} \times loadBid_{Price} - genOffer_{Cleared} \times genOffer_{Price}$$

Equation 4: Objective function for the electricity market model

The objective value for the latest solve is shown on the Results display, along with how much the objective value has changed relative to the previous solve.

To see the details of how the objective value was calculated, tap the Objective row on the Results display, which will take you to the Objective display shown in Figure 11.

Figure 11: The Objective display shows how the objective value is calculated

The electricity price

The bus price of $70/MWh represents the value of electricity at the bus. In terms of the LP model, this bus price represents the rate of change in the objective value that would occur if the node balance constraint for the bus was relaxed. This is explained as follows.

What it means to relax a constraint

If we add 1MW to the node balance constraint, as shown in Equation 5, we have relaxed the node balance constraint by 1MW.

$$Gen_{bus00} - Load_{bus00} + 1\text{MW} = 0$$

Equation 5: Node balance constraint relaxed by 1MW

Relaxing the constraint by 1MW effectively adds 1MW of power to the bus. Observing how the objective value changes due to this additional 1MW will tell us the value of that 1MW, i.e., the value of electricity at the bus.

Relaxing the node balance constraint

We can simulate relaxing the node balance constraint by adding the extra 1MW via a new generator with a $0 offer price.

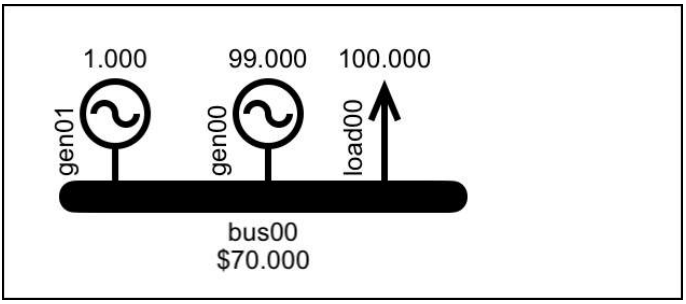


Figure 12: Add new generator, gen01, to the model and drag it into place on the bus

Add the new generator by tapping the Gen button. The new generator will probably appear in the right place… but, if necessary drag it into place on the bus so that it is located as shown in Figure 12.

Double-tap the new generator to view its Data Display and modify its offer block 1 to be 1MW at $0/MWh as shown in Figure 13.

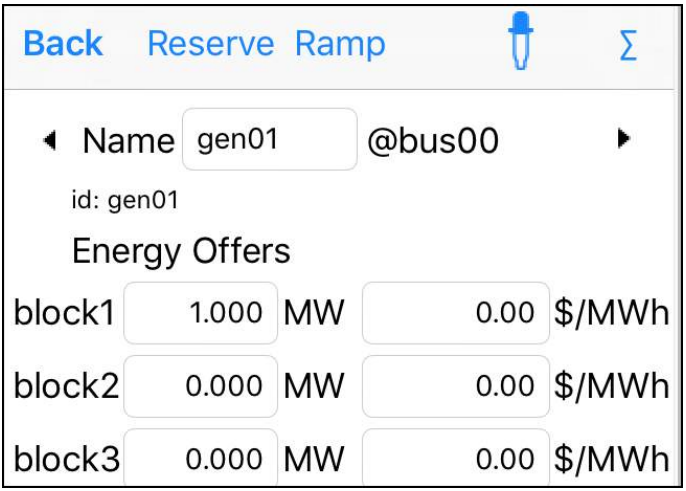


Figure 13: Double-tap gen01 and edit its block 1 offer to be 1MW at $0/MWh

Solve the model, with all solve settings set to OFF. To view what impact this has had on the objective value, tap the Results button. As shown in Figure 14 the objective value has increased by $70.

To achieve this $70 increase in the objective value we relaxed the node balance constraint by 1MW,

hence the rate of change of the objective value due to relaxing the node balance constraint for bus00 is $70/MW.

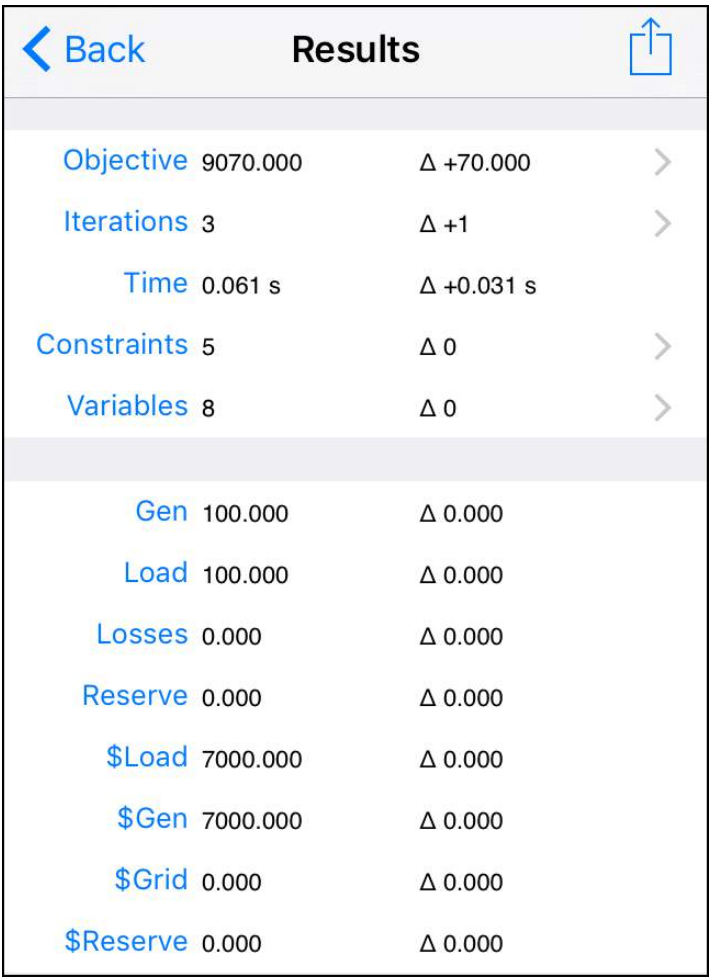


Figure 14: Objective △ $70 for 1MW at $0 → bus price is $70

This tells us that the benefit due to generation at the bus is $70/MW. To clear generation with a non-zero offer price the objective value must incur the cost of the offer, which will reduce the objective value. Therefore, provided the offer price is less than $70/MW (offered as $/MWh), overall it will improve the objective value and the offer will clear.

Explaining the bus price

The above example demonstrates that the value of extra electricity at a bus is determined by the impact that it has on the objective value. The example also allows us to explain how this impact occurred by looking at how the results have changed.

The objective value increased because the additional 1MW at $0/MWh allowed the generation of gen00 to be reduced by 1MW, reducing the overall cost by 1MW x $70/MWh = $70, thereby increasing the objective value by $70.

On the Results display, tapping the Objective row takes you to the Objective display shown in Figure 15, which details the calculation of the objective value. Note that because the generation offer that was used to relax the node balance constraint has a cost of $0/MWh, its offer price had no impact on the objective.

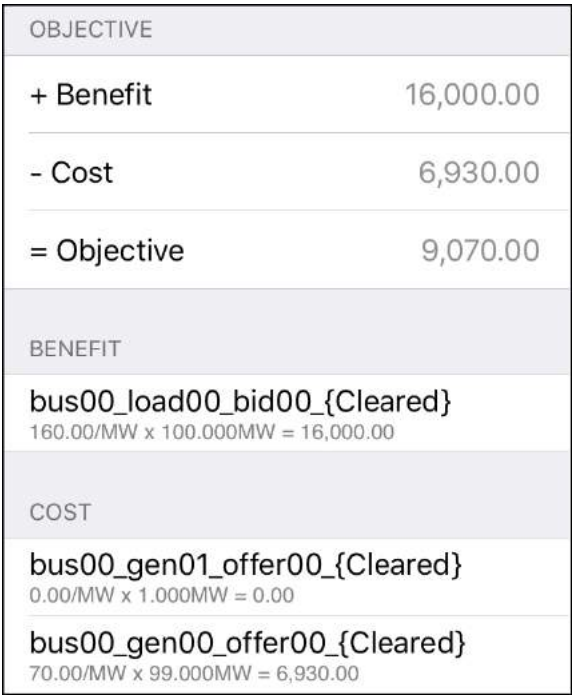


Figure 15: Details of the calculation of the objective value

The shadow price of a constraint

The $/MW improvement in objective value due to relaxing a constraint is referred to as the shadow price of the constraint. The shadow price of a bus's node balance constraint determines the electricity price at the bus.

The shadow price of *any* constraint in the LP model is the $/MW change in the objective value due to relaxing that constraint. The only shadow price that

is immediately useful is the shadow price of the node balance constraint because it sets the electricity price.

You can view the shadow price of any of the constraints by looking on the Constraints display. If a constraint has a non-zero shadow price, then this indicates that the objective value would be changed if the constraint is relaxed. This in turn indicates that the constraint is binding, i.e., the quantity being constrained would have been used more, because it can improve the objective value, if it were not for the constraint.

Summary

In this section we introduced Linear Programming (LP) and built a small model to demonstrate how a nodal electricity market is represented as an LP model.

We explained that the shadow price of a constraint is the benefit to the objective value of relaxing that constraint. We saw that the shadow price of the node balance constraint determines the bus price. We also saw how to use the app to relax the node balance constraint, and thereby explain the bus price.

Tutorial 2: Modelling Transmission

In our previous example the load and generation were at the same bus. In this tutorial we will see how to model the transmission of power from one bus to another.

The Branch component

A branch transmits power from one bus to another. A branch may be a transmission circuit or cable connecting buses at different substations, or a transformer connecting buses that are at different voltages in the same substation.

Adding a Branch

Clear the previous model by tapping the  icon on the Build toolbar.

Build the model shown in Figure 16 by adding two buses (tap the Bus button twice), then add a generator and a load (tap the Gen button and then the Load button), finally add a branch (by tapping the Branch button).

On the Main toolbar tap the Solve button, select all the solve settings to OFF, then tap the Solve Now button.

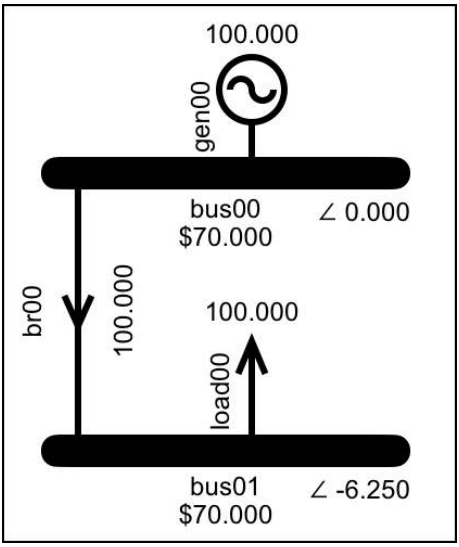


Figure 16: An electricity market model with transmission

Modelling a Branch

As discussed in Tutorial 1: Explaining Prices, the constraints of the LP model enforce the physical reality of the system being modelled. The physical reality of power flowing from one bus to another is enforced by two constraints; the power flow constraint and the maximum flow constraint.

The Power Flow constraint

The power flowing in the transmission system is Alternating Current (AC) electricity. The physical reality of AC power flow is modelled by an AC power flow equation.

The power flow constraint in the LP model implements a *simplified* AC power flow equation as shown in Equation 6.

$$Flow_{Branch} = (Angle_{FromBus} - Angle_{ToBus}) \times Susceptance_{Branch}$$

Equation 6: Simplified AC power flow equation

The simplified AC power flow equation looks similar in *form* to a DC power flow equation; hence a model that implements the simplified AC power flow is sometimes referred to as a DC power flow model.

Bus Angle

In Equation 6 the angle term refers to the phase angle of the voltage at the bus. In an AC system the voltage and current rise and fall, this happens 50 times per second in a 50Hz system. The voltage and current do not necessarily rise and fall at the same time. The voltage phase angle represents the timing difference between the rise and fall of the voltage and the rise and fall of the current.

How much AC power flows through a branch is proportional to the difference between the magnitude and phase angle of the voltage at one end of the branch and the magnitude and phase angle of the voltage at the other end.

In the simplified AC power flow, only the difference in the phase angle is used in the calculation; the magnitude of the voltage is taken to be the same at each end.

Because it is only the *difference* in phase angle that is used, it is only the relative phase angle of the buses that is important. Hence, in the LP model the phase angle at one of the buses is set to zero, this is the reference bus, and the solver is free to adjust all other phase angles relative to this.

The phase angle values determine how much power is scheduled to flow through the branches.

As shown in the Figure 16 result, after the solution is complete each bus has an associated phase angle. In this model bus00 is the reference bus for the electrical island. The first bus that is added becomes the reference bus, but the reference bus can be changed by selecting another bus as the reference bus, via its Data Display. Try changing the reference bus and re-solving to confirm that the phase angles are swapped around but the branch flow remains the same.

Branch Susceptance

The *impedance* of the branch impedes the flow of AC power. In the simplified AC power flow model, the

impedance is represented by branch susceptance. Susceptance is related to the *inverse* of the impedance; hence susceptance is a multiplier not a divisor in Equation 6.

Equation 7 shows how susceptance is calculated from the branch's resistance R and reactance X.

$$Susceptance = \frac{-X}{R^2 + X^2}$$

Equation 7: Calculation of susceptance

The Maximum Flow constraint

The other branch constraint in this model is the maximum flow constraint shown in Equation 8.

$$Flow_{Branch} \leq MaxFlow_{Branch}$$

Equation 8: Branch Maximum Flow Constraint

This constraint ensures that the LP model does not allow more power to flow through the branch than the branch is capable of safely transmitting. We will see the impact of this constraint in the Binding Branch section.

The Node Balance constraint

As discussed in the single bus example, the node balance constraint ensures that the power that flows into a bus is equal to the power that flows out. In the single bus example this ensured that the

cleared bid quantity was equal to the cleared offer quantity.

Figure 17: Node balance constraints including branch flow

Now that the model includes transmission, i.e., a branch, the node balance constraints include branch flow, as shown in Figure 17, which is the Variables and Constraints display for bus00.

Branch Data Display

The branch parameters are viewed and edited via the branch's Data Display, shown in Figure 18.

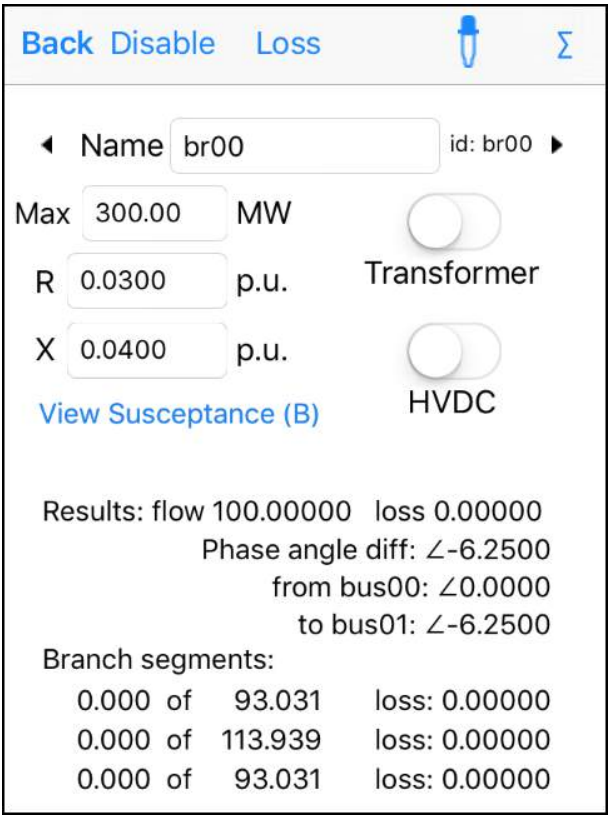


Figure 18: Branch Data Display

The power flow constraint uses the susceptance of the branch, calculated from the resistance and reactance. To view the susceptance, tap the "View Susceptance (B)" button and the display will change to show the calculated susceptance, see Figure 19.

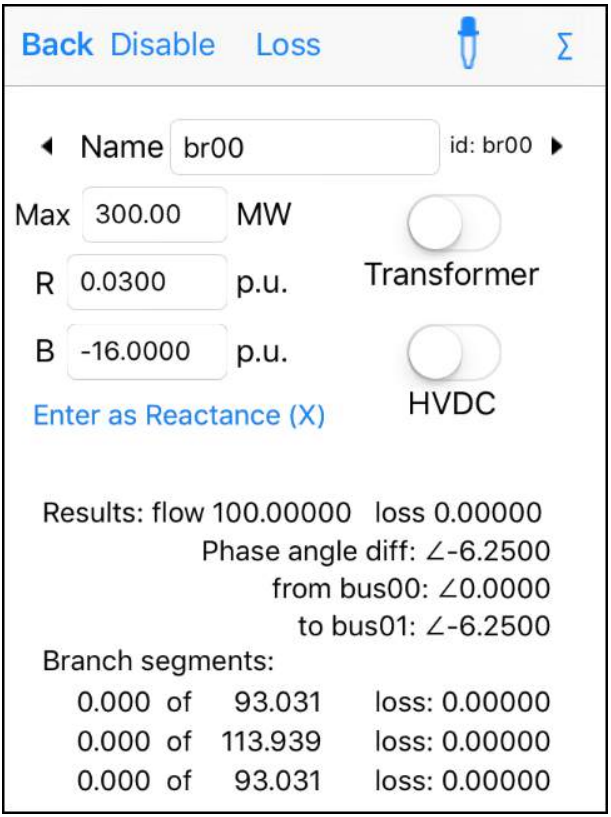


Figure 19: Branch Data Display, Susceptance option selected

Susceptance can be entered directly, in which case the reactance value will be set to zero. If a non-zero reactance value is entered, then susceptance will be re-calculated.

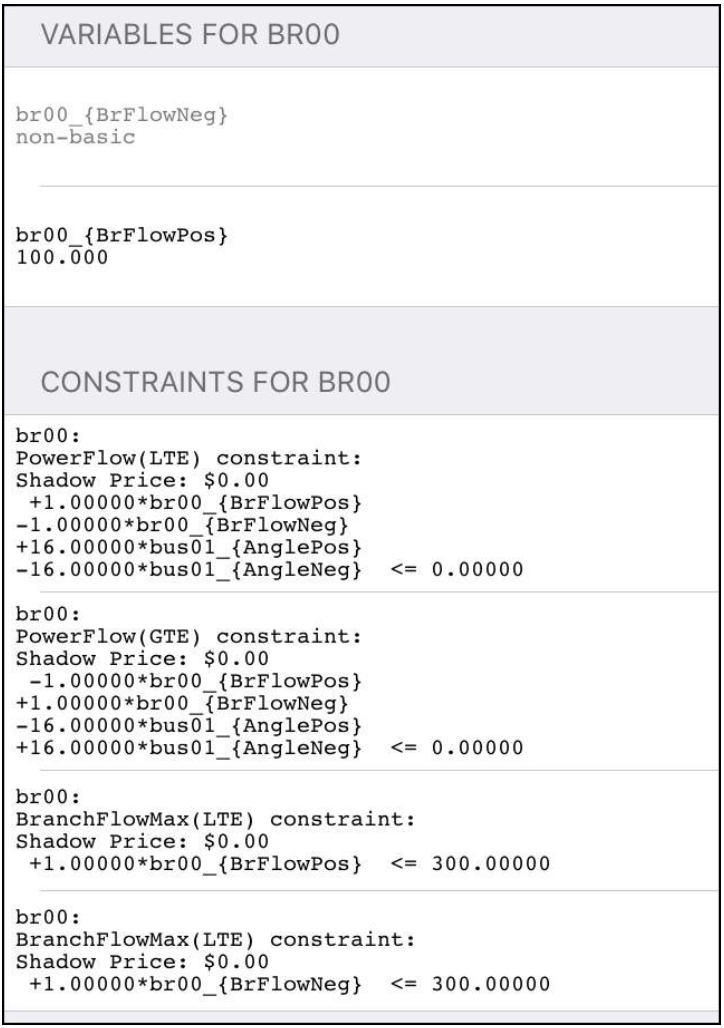


Figure 20: Variables and Constraints display for a branch

Branch variables and constraints

To view the branch's variables and constraints tap the ∑ button on the branch Data Display. This will present the Variables and Constraints display as shown in Figure 20, where you can see the implementation of the power flow and branch maximum flow constraints.

"From bus" and "To bus"

Positive branch flow is defined as being from the bus that has the lower alphabetical name, the "from bus", to the bus that has the higher alphabetical name, the "to bus".

Unrestricted variables

In the variables section of Figure 20 you will notice that there are two branch flow variables; a positive flow and a negative flow. This is because the simplex algorithm restricts all variables to have only positive values (the complete workings of the algorithm are covered in Tutorial 9: Simplex Explained).

If we only had one branch flow variable, we could only schedule branch flow in the positive direction. Hence, branch flow is modelled using two variables; one variable that represents flow in the positive direction and another that represents flow in the

negative direction… both of these variables can only take positive values.

The displayed branch flow result is calculated from the positive and negative variables. The calculation takes place in post-processing after the simplex algorithm has solved the LP Model.

Non-basic variables

On the Variables and Constraints display a variable that is greyed out is non-basic. A non-basic variable has been set to zero by the simplex algorithm. This is because in order to find a feasible solution the simplex algorithm must set a certain number of variables to zero. The variables that are set to zero are the non-basic variables… they are not zero because a constraint limits them to zero, they are zero because the algorithm has determined that allowing them to be non-zero would not improve the result.

We can see from the result in Figure 20 that the reverse flow on the branch is a non-basic variable. The simplex algorithm did not find any way that allowing this variable to be non-zero could produce a better objective value.

For a full explanation of basic and non-basic variables see Tutorial 9: Simplex Explained.

Bus variables and constraints

When we looked at the variables and constraints in Tutorial 1: Explaining Prices, we saw that the bus did not have any associated variables. This was because the model did not contain any transmission, hence there were no power flow constraints and therefore no need for a phase angle variable.

Now that the model includes transmission, bus01 has a phase angle variable. Bus00 is the reference bus so its phase angle is set to zero, i.e., not a variable. The variables and constraints for bus00 and bus01 are as shown in Figure 21 and Figure 22.

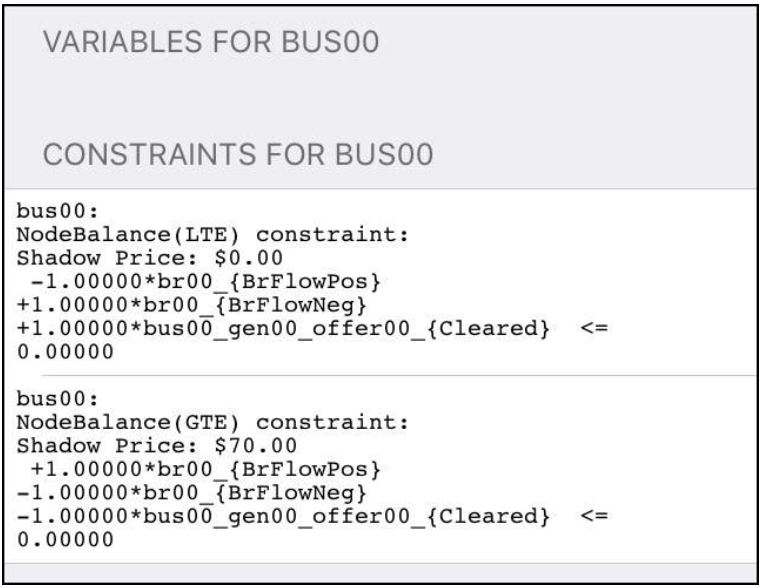


Figure 21: Variables and constraints for bus00

Figure 22: Variables and constraints for bus01

Note that bus01 has two phase angle variables. This is because the phase angle can be positive or negative, i.e., it is an un-restricted variable in the same way as the branch flow variable, and hence the same explanation applies… the simplex algorithm requires that all variables remain >=0, hence the phase angle is modelled as a positive

angle variable and a negative angle variable, both of which can only take non-negative values.

Binding Branch

A branch is binding if its scheduled flow is equal to the limit set by its maximum flow constraint. To see the impact of a binding branch, lower the maximum flow on br00 to 40MW, i.e., less than the scheduled flow of 100MW.

To lower the branch's maximum flow, i.e., the branch limit, double tap branch br00 and lower its Max from 300MW to 40MW. Solve. The result is shown in Figure 23.

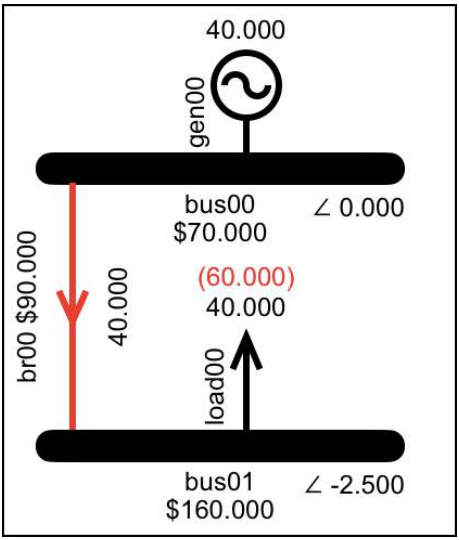


Figure 23: Binding branch

Price separation

Lowering the branch limit has produced a result where the simplex algorithm has not been able to fully clear the bids. This has led to price separation, whereby the prices at bus00 and bus01 are no longer related.

As explained in Tutorial 1: Explaining Prices, the bus price at bus00 is $70/MW due to the fact that relaxing the node balance constraint would allow the solver to reduce the existing $70/MWh generation offer, benefitting the objective value at a corresponding rate.

At bus01 things have changed due to the binding branch… relaxing the node balance constraint at bus01 would not allow the cleared generation to decrease. What it would do is allow more of the $160/MWh bid to clear, which would improve the objective at a rate of $160/MW; hence $160/MW is the price at bus01. Before the binding constraint was applied the $160/MWh bid was cleared to its limit, relaxing the node balance constraint would not allow it to clear any more, hence the $160/MWh price did not influence the bus price.

Minimizing price is not the objective

Note that the objective of the simplex algorithm is not to minimize the bus prices. Rather, the objective is to maximize the objective value, which will be achieved by maximizing the benefit of the cleared load bids while minimizing the cost of the cleared generation offers.

The prices are signals to indicate the potential for improving the objective value. Hence, in Figure 23 there is a higher price at bus01 than at bus00, because generation sited at bus01 would have a greater impact on the objective value.

Capacity Price

Looking at the result shown in Figure 23 we see that the branch is red, which indicates that the branch's maximum flow constraint is binding. Also, the branch has a price of $90 after its name. This $90/MW is the shadow price of the branch's maximum flow constraint.

Every constraint has a shadow price and these shadow prices are shown on the Constraints display. The shadow price of the branch maximum flow constraint only appears on the *main* display if it is non-zero, which only happens when the branch is binding. This non-zero shadow price indicates

that relaxing the branch's maximum flow constraint will improve the objective value.

The shadow price of the br00 maximum flow constraint is \$90/MW because an incremental increase in branch capacity would allow more generation to be cleared at bus00 and transmitted to bus01 where it would allow more bids to clear; the cleared bids would contribute another \$160 of value to the objective, while the corresponding cleared generation offer would cost \$70, for a net benefit of \$90. You can confirm this by increasing the branch capacity by 1MW, viewing the corresponding change in scheduled quantities and observing on the Results display that the objective value increases by \$90.

Congestion Charges

The financial impact of the binding branch is that the load pays more for purchasing electricity than the generation is paid for producing it. The total quantities are shown on the Results display in Figure 24. Here you can see that the load pays \$6400, the generation is paid \$2800. The \$3600 difference is referred to as a congestion charge. This value is labelled \$Grid on the display because it is a cost that arises due to the transmission network, also referred to as the grid.

Later when we model line losses, we will see that they can also give rise to the load paying more than the generation receives even when there are no binding branches. In this case the extra payments are referred to as transmission rentals.

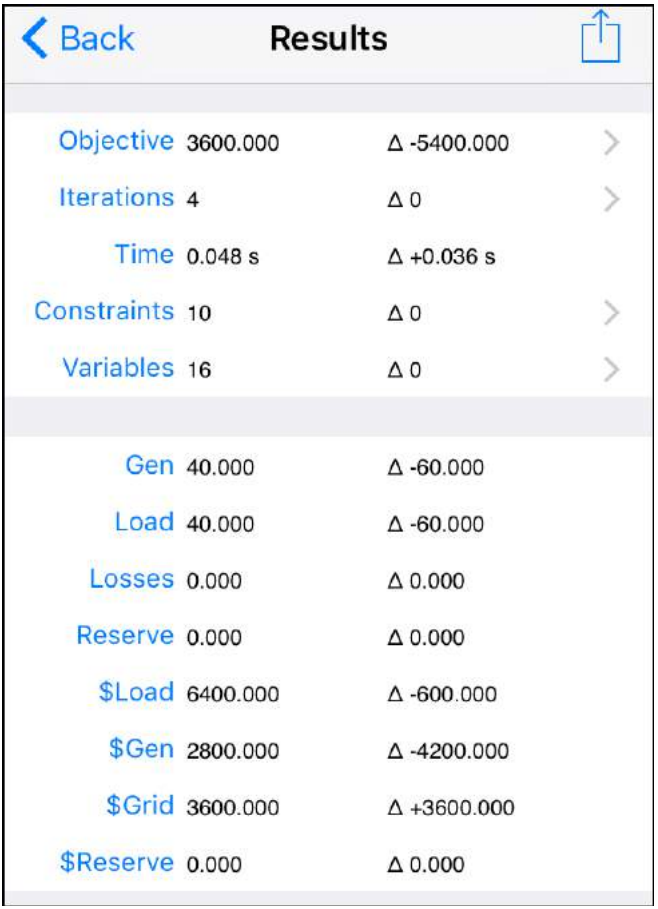


Figure 24: Binding branch leads to congestion charges, as indicated by the non-zero $Grid value

Shadow price and 1MW relaxation

We can also use this binding branch model to demonstrate that the shadow price indicates the per MW change to the objective value due to an *incremental* relaxation of the node balance constraint, which is not necessarily the value of the next 1 MW. Even though we have successfully been using 1MW to demonstrate how changes in the objective value relate to the shadow price, we are about to demonstrate that there are times when 1MW may be too large an increment.

To make this point, change the binding branch example by lowering the bid at load00 from 100MW to 40.5MW. If we solve now then the result is the same, the branch is still binding at 40MW.

To confirm the price at bus01, we are going to relax its node balance constraint by adding a generator with a 1MW offer and a $0 price to bus01 (similar to what we did in Tutorial 1: Explaining Prices). Tap the Gen button to add a generator to the model. Drag the new generator to bus01. Double tap the new generator and edit its offer block1 to be 1MW at $0. Solve, and the result is shown in Figure 25.

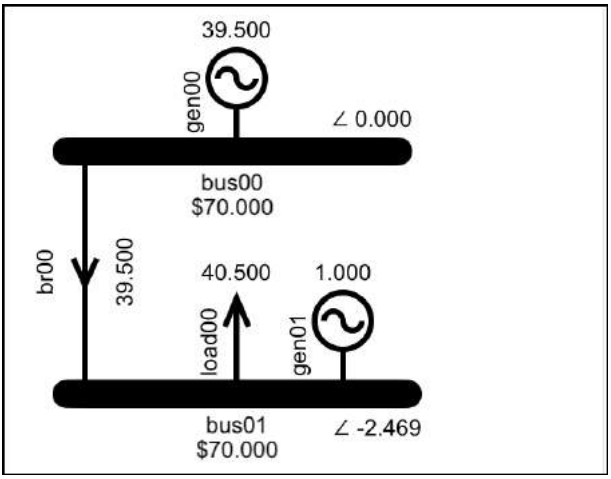


Figure 25: Bid at load00 changed to 40.5MW and gen01 added with an offer of 1MW at $0

The extra 1MW has resulted in the bid fully clearing and therefore the branch is no longer binding… the result has fundamentally changed, and we have *not* succeeded in proving the original $160/MW bus price. To prove the price, we need to relax only the constraint that is setting the bus01 price in the original result… that constraint was the node balance constraint at bus01, but by adding the 1MW of generation we have also relaxed the branch constraint.

Before we try again we need to return to the binding line result by setting the offered gen quantity at gen01 to 0MW then solving the model. Next, to relax just the node balance constraint and

leave the branch binding, edit gen01 so that its offer quantity is 0.1MW at $0.

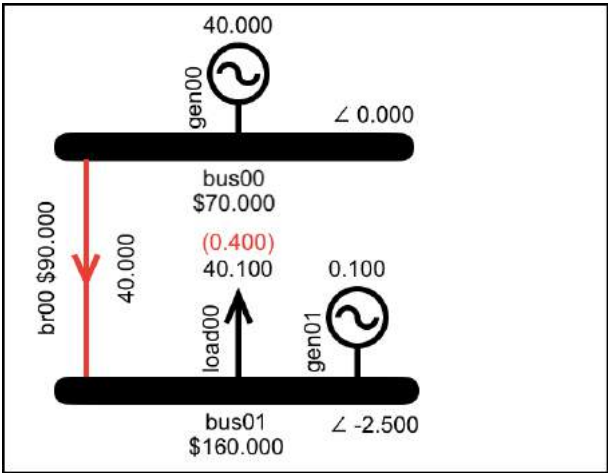


Figure 26: Bid 40.5MW and gen01 offering 0.1MW at $0

Solve again and the result is as shown in Figure 26. We still have the $160 bus price and we can see that the only change is that the bid has now cleared 40.1MW. Checking the Results display shown in Figure 27 we see that the 0.1MW increase in the cleared bid quantity has benefited the objective by $16. Dividing the benefit of $16 by the incremental quantity of 0.1MW we confirm that the per-MW value of generation at bus01 is $160/MW.

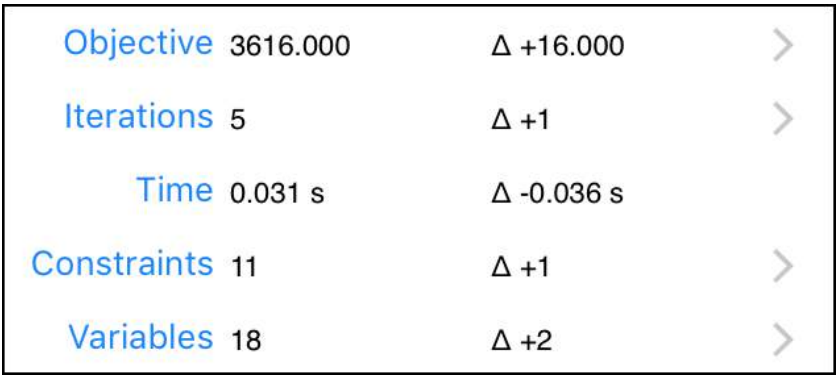


Figure 27: Adding 0.1MW at $0 results in objective value change of +$16, i.e., $160/MW

Summary

In this tutorial we added transmission to the model and saw how it is modelled by the power flow constraint, the branch limit constraint, and the inclusion of branch flow in the node balance constraints.

We saw how the power flow constraint leads to the solver using the phase angle variable to determine branch flow. We also saw that when the branch limit binds, the bus prices at either end of the branch become separated, i.e., a binding branch constraint causes price separation.

The binding branch example was also a useful opportunity to present an example showing that it is the value of an *incremental* change in the power

available at the bus that determines the bus price, i.e., while it may be possible to use a 1MW change to illustrate the impact of an incremental change (as we have done earlier), it is also the case that 1MW may be too large to represent an *incremental* change.

Tutorial 3: Parallel Flows & Spring Washer

In the previous tutorial we added transmission to the model, which involved introducing the power flow constraint. Now we will investigate the impact that the power flow constraint has on a model with parallel branches and how this can lead to the spring washer effect.

Parallel Flows

Unconstrained parallel flows

Build and solve the parallel branch model shown in Figure 28 by tapping Bus-Bus-Gen-Load-Branch-Branch. Solve the model with the default values and all solve settings selected to OFF.

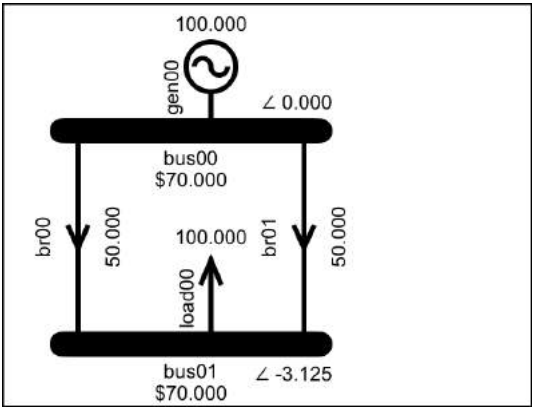


Figure 28: Parallel branches model

As described in the previous section, the power flow constraint calculates branch flow as the product of the branch susceptance and the phase angle difference between the from-bus and the to-bus. Because both branches in this example have the same susceptance, the same flow limit, and are connected to the same buses, their power flow is the same.

Parallel flows with a binding branch

Parallel branches get more interesting when one of the branches is binding. To see this, reduce the maximum flow limit on br00 from 300MW to 40MW. The result is shown in Figure 29.

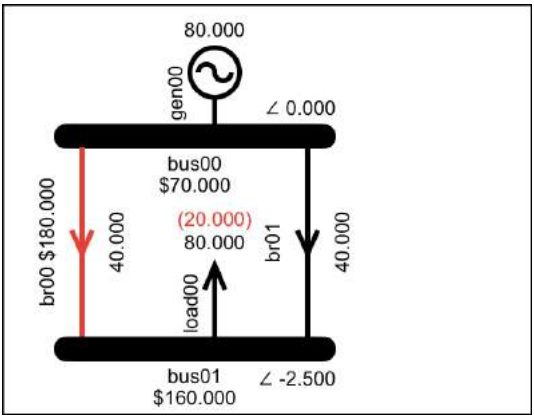


Figure 29: Binding branch restricts flow on parallel branches

We see that although the maximum flow on br01 remained at 300MW, its scheduled flow is now

40MW. This is because both branches have the same susceptance and are connected to the same buses, hence their scheduled flow must be the same... because br00 is binding on its flow limit of 40MW, br01 is also limited to 40MW.

Capacity cost

When br00 was binding and there was no parallel branch (see the binding branch example in the previous tutorial), the branch had a capacity cost of $90/MW. Now this cost is double because an incremental increase in flow on br00 will also allow a corresponding increase on br01.

Parallel flows with different susceptance values

The previous example shows that parallel branches with the same susceptance and the same end points must have the same flow. Not unexpectedly, if parallel branches have different susceptance values then they will have different flows.

To see this, edit br00 to double its susceptance from -16 to -32. The flow on br00 will be twice that on br01. Branch br00 will still bind, but this time the flow on br01 will be half that of br00, as shown in Figure 30.

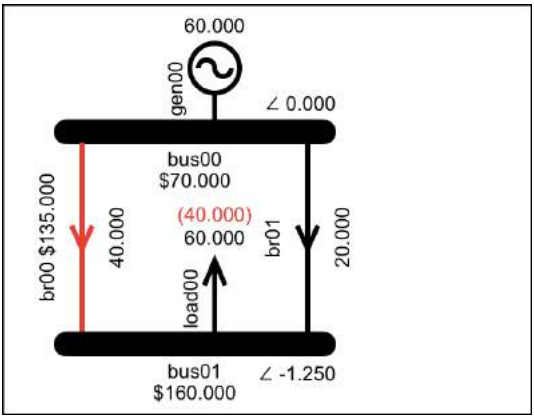


Figure 30: Branch br00 susceptance twice that of br01

Conversely you can change the branch parameters to see what happens when the susceptance of the non-binding branch br01 is increased. This will result in br01 transmitting more of the power. If the br01 susceptance is increased enough then it will take enough of the flow so that br00 will no longer bind.

Susceptance and reactance

To investigate the impact of different susceptance values we could also change the reactance, because susceptance is calculated from reactance and resistance, as shown in Equation 7.

Changing the susceptance as we have been doing provides a more direct demonstration because it is the susceptance value that is used in the constraint,

e.g., halving the susceptance will halve the flow, whereas doubling the reactance will not have the same effect because the reactance is combined with the resistance to calculate the susceptance. You can check for yourself by changing the reactance and seeing what happens.

The Spring Washer effect

Building and solving a three-bus model

Now that we have seen how a binding branch can limit the flow on parallel branches, we will take this a step further and look at what happens when there are multiple paths between the generation and the load. This model is then used to investigate and explain the spring washer effect.

Create the model shown in Figure 31 by adding three buses (tap Bus-Bus-Bus), add a generator and a load (tap Gen-Load), and then add two branches (tap Branch-Branch).

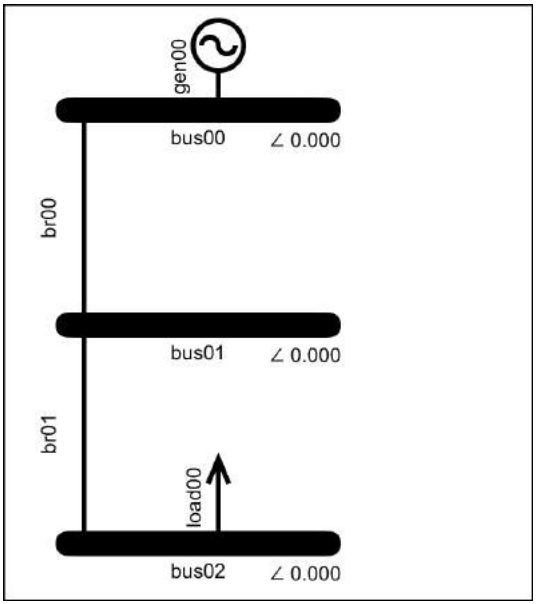


Figure 31: Building the three-bus model, step 1

Add the third branch as shown below in Figure 32... to make adding this branch as easy as possible, first drag the ends of bus00 and bus02 to make them wider than bus01, then add another branch, which will automatically connect to the end of these two buses.

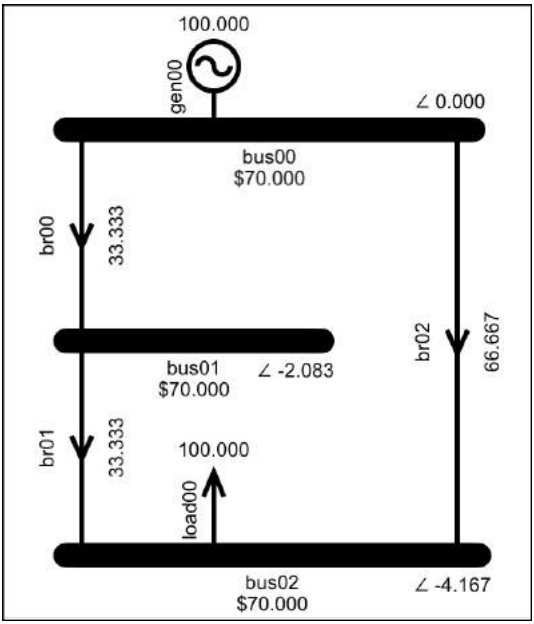


Figure 32: Three bus model

Solve the model with all Solve Settings selected OFF, to produce the result shown in Figure 32. The branches have the same susceptance. Therefore, at first glance, it may seem that they should all have the same flow. The reason that they don't is because they are not all connected to the same buses.

The branches cannot have the same flow because for br00 and br02 to have the same flow the power flow constraint would require them to have the same phase angle difference, which would require bus01 to have the same phase angle as bus02. But

that would result in a phase angle difference of zero across br01, which would only work if br01 flow was zero, which it can't be because the node balance constraint requires that the flow into bus01 must match the flow out.

The result shown in Figure 32 meets the requirements of the constraints. Due to the way that the power is flowing, br00 and br01 are in parallel with br02, hence the sum of their phase angle difference must be the same as the phase angle difference across br02. Because all of the branches are the same, the phase angle difference across each of br00 and br01 must be half that across br02, and therefore each must have half the flow of br02.

Another way of looking at this is that the total impedance across br00 and br01 is twice that across br02, hence the flow down that path is halved.

Losses were not included, and no branches are binding, hence all bus prices are the same.

This example shows how a meshed network affects power flow. This result is the pre-cursor to explaining the spring washer effect.

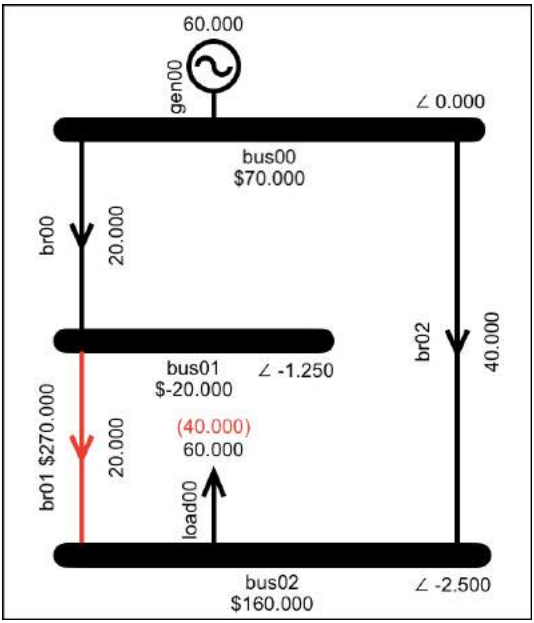


Figure 33: The spring washer effect

Creating the spring washer effect

Create the spring washer effect by double tapping br01 and lowering its maximum flow from 300MW to 20MW; currently it is scheduled at 33.33MW, so reducing its maximum to 20MW will cause it to bind. Solving this model produces the result shown in Figure 33.

Why it is called the spring washer effect

The "spring washer" in the spring washer effect refers to the similarity between the "shape" of the

prices shown in Figure 33 and the shape of an actual spring washer, shown schematically in Figure 34.

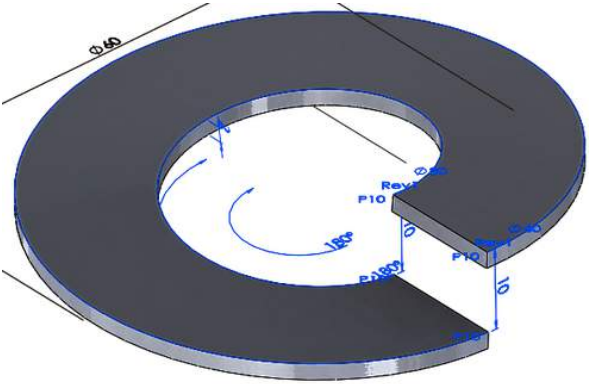


Figure 34: Representation of an actual spring washer

The gap between the highest and lowest points of the actual spring washer are represented by the binding branch which is the gap between the highest and lowest prices.

The curve of the spring washer is represented by the prices passing from the lowest price at bus01 on the upstream side of the binding branch (the gap), through the medium price of $70 at the generator bus, bus00, then via br02 to the highest price of $160 at the load bus, bus02 (the other side of the gap).

Explaining the positive prices

The prices at bus00 and bus02 are as explained in Tutorial 2: Modelling Transmission, i.e., the uncleared bids at bus02 set the bus price at bus02 because the value of electricity at this bus is the value of the bids that could be cleared. The cleared offers at bus00 set the price at bus00 because the value of extra electricity at this bus is the value caused by not having to clear the existing offers.

Explaining the negative price

The explanation of any bus price is that it indicates the $/MW rate of change to the objective value due to relaxing the node balance constraint at that bus.

For the spring-washer example we need to expand on this slightly. A positive bus price indicates that an increase in available power will increase the objective value, while a negative bus price indicates that an increase in available power will decrease the objective value. These statements are also true if both cause and effect are reversed; a positive bus price indicates that a decrease in available power will decrease the objective value, while a negative bus price indicates that a decrease in available power at the bus will increase the objective value.

As before, we can investigate the price at bus01 by seeing what happens when its node balance constraint is relaxed. In previous examples we relaxed the node balance constraint by adding a 1MW offer at $0/MWh. In the spring washer example that we are looking at now, a $0/MWh generation offer will not clear at bus01 because the negative price indicates that there is a disincentive to making power available at bus01.

The disincentive of extra power

For an offer to clear at a bus, the *cost* of its power must be less than or equal to the benefit of extra power at that bus. The benefit of extra power at bus01 is -$20, therefore only an offer with a cost less than or equal to -$20 will clear, i.e., a generator that is prepared to *pay* $20/MWh to generate.

Figure 35 shows the result of adding a generator to bus01 with an offer of 1MW at -$20/MWh. The offer clears and we can use the result to explain how the extra 1MW at bus01 impacts the objective value.

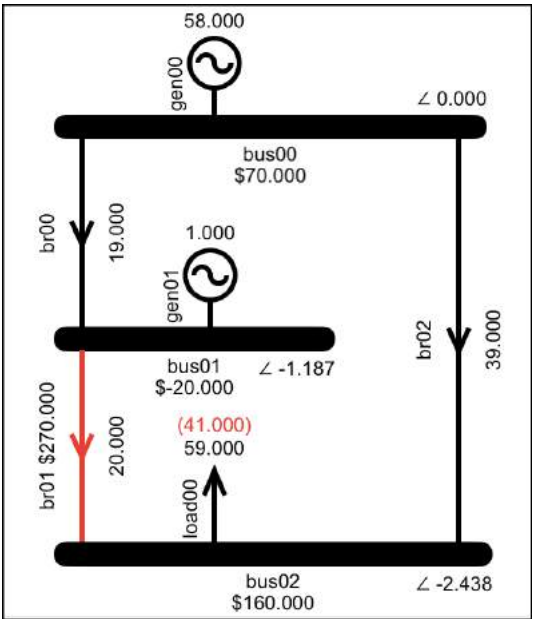


Figure 35: Impact of an extra 1MW at bus01

The changes in the results due to relaxing the node balance constraint at bus01 with a -$20/MW offer are explained as follows:

The extra 1MW at bus01 heads toward the load via br01, hence the existing inflow to bus01 via br00 must be reduced by 1MW to keep br01 on its limit.

To reduce the inflow to bus01 from br00 the phase angle at bus01 is reduced. Because br01 is connected to bus01, in order to keep the flow on br01 at its limit the phase angle at bus02 must also be reduced.

Reducing the phase angle on bus02 forces the flow on br02 to reduce by 1MW. The direct impact of this is that the cleared bid quantity at bus02 must reduce by 1MW.

Overall the flow out of bus00 was reduced by 1MW on br00 and 1MW on br02, therefore the cleared offers at bus00 must reduce by 2MW.

Table 1: Change in objective for 1MW extra at bus01

Component	△Quantity	Price	△Objective
gen00	-2MW	$70	+$140
load00	-1MW	$160	-$160
objective			-$20

The impact on the objective value of the 1MW of extra power at bus01 is detailed in Table 1. The calculated change in the objective value is $-20, which lines up with the bus price at bus01.

Note that if bus01 had been the reference bus then it would have been the phase angle at bus00 that would have been reduced… the overall effect would have been the same, as shown in Figure 36.

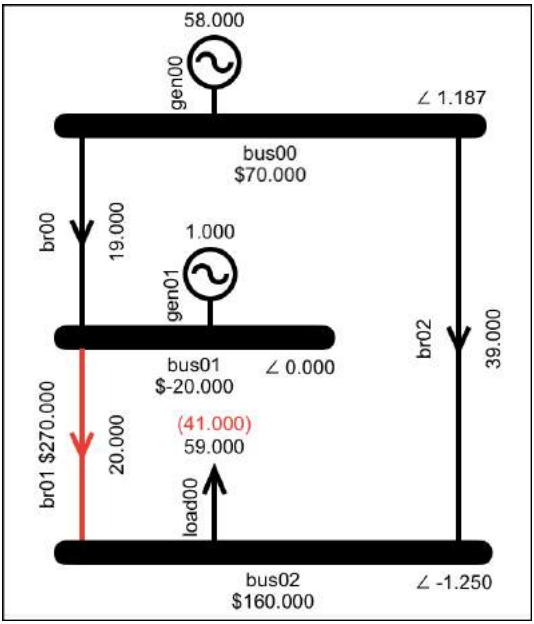


Figure 36: Overall the same result with bus01 as reference

No change in objective value

This experiment to explain the objective value is different to those conducted previously in that the calculated change in the objective value presented in Table 1 is not reflected by the observed change in objective value. This is because the price of the offer that relaxed the node balance constraint is not $0, we had to make it $-20 so that it would clear. Therefore, even though the *calculated* impact on the objective value was $-20, this is cancelled out by the

negative cost, i.e., the benefit, of clearing the offer with the -$20/MWh price.

The -$20 bus price indicates that generation must *pay* at least $20/MWh in order to be cleared at bus01. It also indicates that load will be *paid* $20/MWh if it is situated at bus01. We can see this by deleting the generator at bus01 and adding a load.

Relaxing node balance by adding load

The negative price at bus01 indicates that more power flowing into the bus will make the objective value worse. This also indicates that more power leaving the bus will make the objective value better; we can demonstrate this by adding load at bus01.

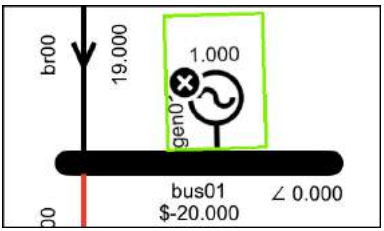


Figure 37: Press and hold gen01 to initiate the delete, tap gen01 to delete, tap anywhere else to cancel

The first thing that we need to do is delete the extra generator that we added. Delete gen01 by pressing it until it starts to wobble, and a cross appears as

shown in Figure 37. Then tap gen01 to perform the delete or tap anywhere else to cancel.

Now add a load to bus01 and edit it so that it has a bid of 1MW at $0. Solve, and the result is as shown in Figure 38.

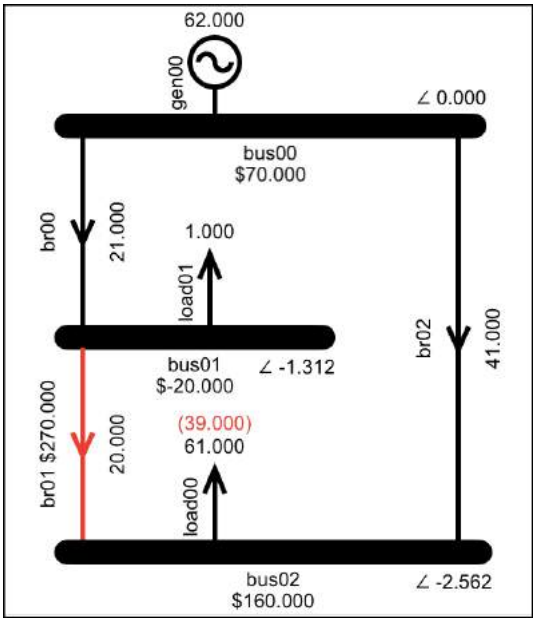


Figure 38: Spring washer with node balance relaxed by adding load

The explanation of the result when load is added is generally similar to the generation example we worked through above. Power is supplied to the extra 1MW of load by increasing the flow on br00. This increased flow requires an increase in the

phase angle at bus01, which in turn requires an increase to the phase angle at bus02. The increase in the bus02 phase angle allows more flow along br02, which allows more of the bids to clear at bus02. Overall, increasing the load at bus01 provides a benefit due to allowing an increased bid quantity at bus02. This benefit outweighs the cost of increasing the cleared offer quantity, hence the objective value has improved.

Compared to the original result shown in Figure 33, the impact that the extra 1MW of load has on the objective value is calculated in Table 2.

Table 2: Change in objective for 1MW load at bus01

Component	△Quantity	Price	△Objective
gen00	+2MW	$70	-$140
load00	+1MW	$160	+$160
load01	+1MW	$0	$0
objective			+$20

This is confirmed by the Results in Figure 39 which show that the extra 1MW of load resulted in a $20 increase in the objective value. Because this increase was due to *less* power at the bus, this translates to a bus price of -$20.

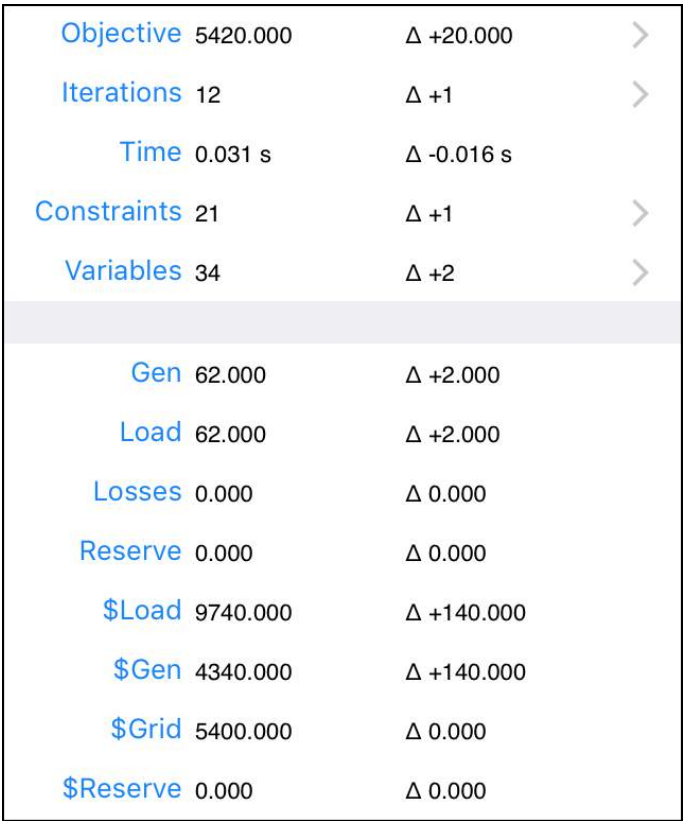


Figure 39: Node balance constraint relaxed by 1MW of load improves objective by $20

Spring washer prices not always negative

The characteristic of the spring washer effect is that the bus with the lowest price, i.e., the bus on the upstream (generator) side of the binding constraint, will have a price that is lower than the bus prices

either side of it… but it is not necessarily a negative price.

We can demonstrate this by increasing the offer price at gen00 from \$70/MWh to \$100/MWh, which produces the result shown in Figure 40. The price at bus01 is still the lowest but is no longer negative.

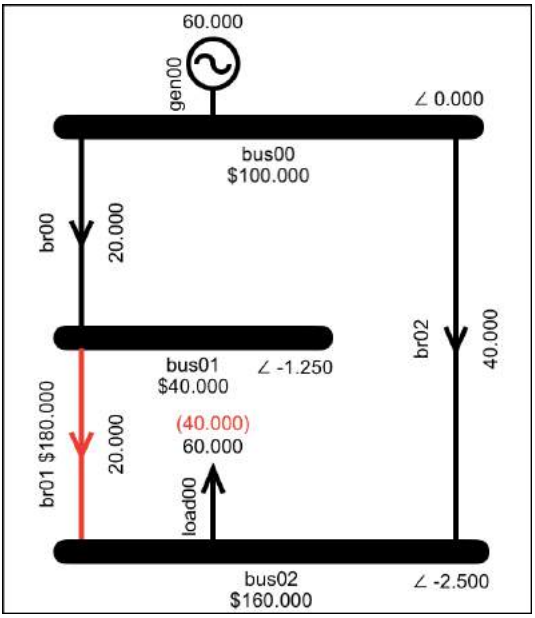


Figure 40: Spring washer price not negative, due to higher offer price

Returning gen00 to its original offer price of \$70/MWh but lowering the susceptance of br02 from -16 to -5, we see that by making br02 less attractive to power flow the benefit of load at bus01

has decreased, producing the result shown in Figure 41.

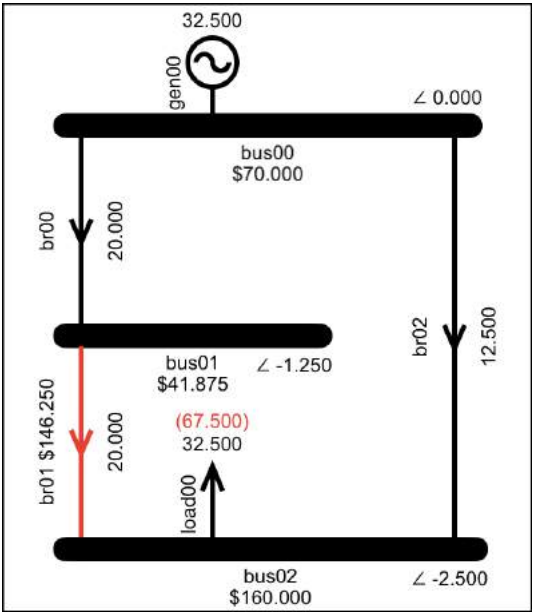


Figure 41: Spring washer price not negative, due to lower susceptance on br02

Summary

The examples in this section showed that because the power flow constraint links branch flow to bus angle, a branch's flow may be restricted due to limits that are binding on parallel branches.

We also saw that the impact of parallel branches can result in the spring washer effect, whereby the price at a bus situated between generation and load

can end up with a negative price, indicating that load at that bus will improve the objective value. We saw that this is because such load would influence the phase angle difference across the binding branch, thereby enabling increased flow on branches parallel to the binding branch, allowing more paying load to clear.

Tutorial 4: Transmission Losses

This tutorial explains how transmission losses are included in the electricity market model and the impact that including transmission losses has on the results.

Modelling transmission losses

Transmission losses = Power lost as heat

When electrical power flows through a conductor the movement of electrons causes the conductor to heat up. The more power that flows, the more heat. If a conductor gets too hot, then its structure is permanently altered… the maximum flow limit is set to prevent this from happening.

Power that is lost as heat is power that is not transmitted to the bus at the other end of the branch. In the model, this lost power is referred to as branch losses.

Our market model uses the simplified power flow equation to calculate a branch's power flow. The branch loss calculation corresponding to the simplified power flow equation is shown in Equation 9.

$$Losses_{branch} = Resistance_{branch} \times MWFlow^2_{branch}$$

Equation 9: Branch loss calculation for simplified AC power flow

Modelling branch losses

Because the branch loss calculation in Equation 9 includes a quadratic term, it represents a curve (specifically a parabola) and therefore it cannot be directly included in the LP (*Linear* Programming) model used by the simplex algorithm because this model requires its constraints to be linear, i.e., straight lines not curves.

To meet the requirement for linear constraints, the loss curve is modelled as a series of straight lines. To see this, build a two-bus, one-branch model (tap Bus-Bus-Gen-Load-Branch) and solve with the "Include Losses" option set to ON as shown in Figure 42. The result is shown in Figure 43.

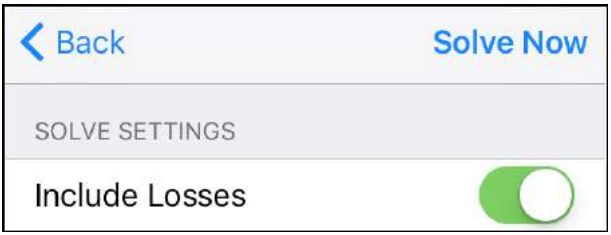


Figure 42: Setting branch losses to ON

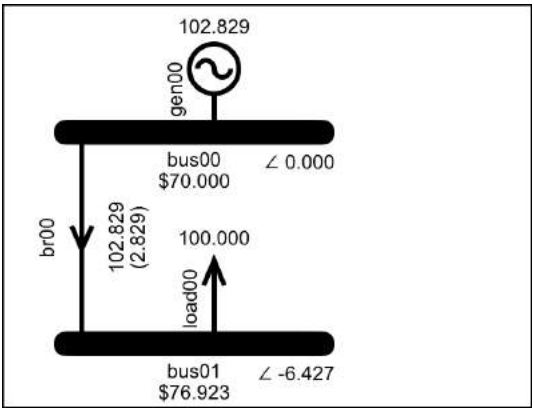


Figure 43: Result that demonstrates branch losses

To view the branch loss parabola, double tap the branch to see its Data Display then tap the "Loss" button on the toolbar, as indicated in Figure 44

Figure 44: Branch Data Display with Loss button indicated

The "Loss" button leads to the "Branch Segments" display in Figure 45, which shows the individual loss segments in table form and also as a plot. The display also shows the settings that were used to calculate the segments.

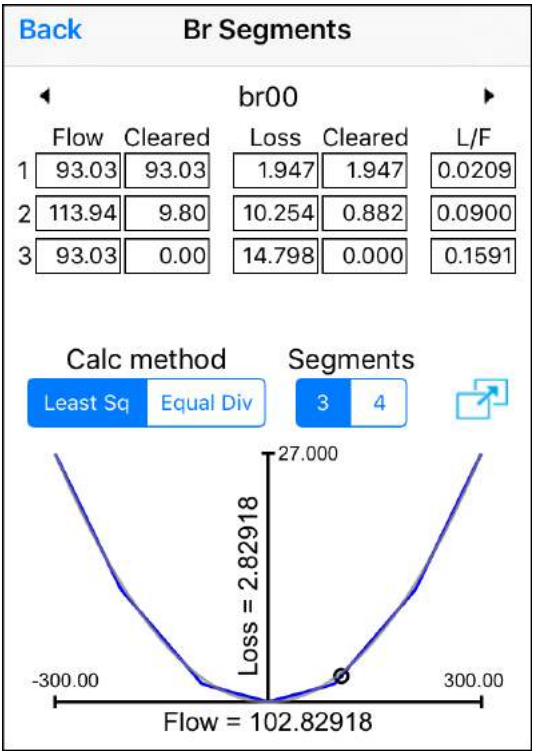


Figure 45: Branch flow-loss segments

In the plot the flow-loss result from the latest solution is indicated by a dot with a circle around it.

The display shows the individual linear segments in blue, with the parabola that they are estimating shown in grey. Using the pinch gesture to zoom in, you can see a close up of the differences between the piecewise curve and the parabola, as shown by the detail in Figure 46.

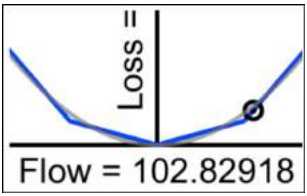


Figure 46: Zoom in to see details of parabola vs segments

To see a full screen version of the plot, as shown in Figure 47, tap the  button.

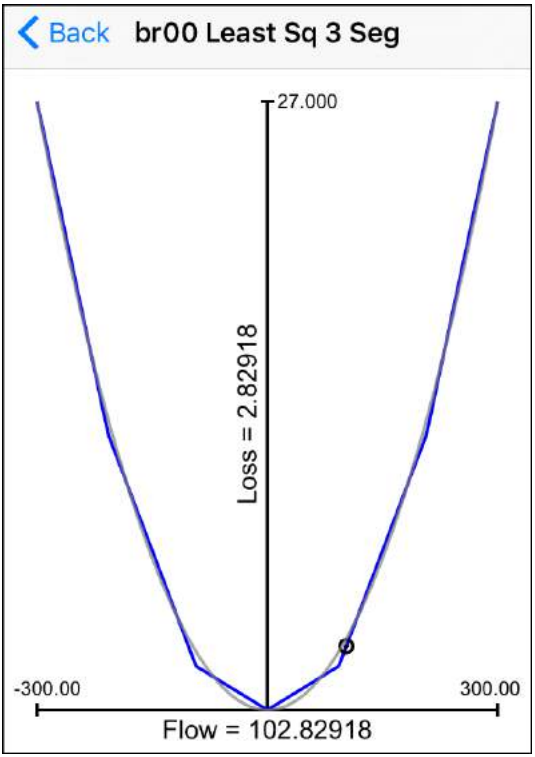


Figure 47: Full screen flow-loss curve

How the segments are calculated

There are different ways of modelling the branch segments. The app allows you to investigate the impact of the following modelling decisions:

- How many branch segments
- What algorithm is used to determine the endpoints of the segments

Number of segments vs total segments

The number of segments refers to the number of segments in each direction, i.e., selecting three segments results in the model using three segments to model flow in the forward direction and three segments in the reverse direction.

Segment endpoints

To determine the endpoints of the segments, the New Zealand electricity market uses an algorithm that minimizes the error between the parabola and the segments. The algorithm is derived from a least squares error function. The app refers to this as the "Least Sq" (least squares) method.

The Singapore electricity market calculates the segments by dividing the maximum flow into equal divisions. The loss point at the end of the segments is taken directly from the parabola. In the app this

is referred to as the "Equal Div" (equal divisions) method.

Note: Unless otherwise stated, the tutorial examples all use "Least Sq" and three segments.

Branch constraints to model losses

As shown in Equation 10, the branch flow is constrained to be equal to the sum of the flow on the individual flow-loss segments.

$$Flow_{branch} = \sum_{Segments_{branch}} Flow_{segment}$$

Equation 10: Branch flow is sum of segment flows

The "branch flow is sum of segment flows" constraints can be viewed on the Constraints display, as shown in Figure 48.

```
br00:
BrFlowIsSumOfBrSeg(LTE) constraint:
Shadow Price: $6.92
 +1.00000*br00_{BrFlowPos}
-1.00000*br00_brSeg00_{SegFlowPos}
-1.00000*br00_brSeg01_{SegFlowPos}
-1.00000*br00_brSeg02_{SegFlowPos}  <= 0.00000

br00:
BrFlowIsSumOfBrSeg(GTE) constraint:
Shadow Price: $0.00
 -1.00000*br00_{BrFlowPos}
+1.00000*br00_brSeg00_{SegFlowPos}
+1.00000*br00_brSeg01_{SegFlowPos}
+1.00000*br00_brSeg02_{SegFlowPos}  <= 0.00000

br00:
BrFlowIsSumOfBrSeg(LTE) constraint:
Shadow Price: $0.00
 +1.00000*br00_{BrFlowNeg}
-1.00000*br00_brSeg00_{SegFlowNeg}
-1.00000*br00_brSeg01_{SegFlowNeg}
-1.00000*br00_brSeg02_{SegFlowNeg}  <= 0.00000

br00:
BrFlowIsSumOfBrSeg(GTE) constraint:
Shadow Price: $0.00
 -1.00000*br00_{BrFlowNeg}
+1.00000*br00_brSeg00_{SegFlowNeg}
+1.00000*br00_brSeg01_{SegFlowNeg}
+1.00000*br00_brSeg02_{SegFlowNeg}  <= 0.00000
```

Figure 48: Constraints display: Branch flow is sum of segment flows

Because the segment constraints link branch flow to segment flows, in order to schedule flow on the branch the solver must schedule flow on one or more of the branch segments…. and in order to schedule flow on a branch segment the solver must also schedule the corresponding loss, due to the

constraint shown in Equation 11 which requires that every MW of segment flow has a corresponding loss, determined by the segment's loss-flow-ratio. Figure 49 shows the "loss for flow" constraints for one of the three segments.

$$Loss_{segment} = LossFlowRatio_{segment} \times Flow_{segment}$$

Equation 11: Segment loss-for-flow constraint

Each segment also has its own flow limit, as described by Equation 12, and shown in Figure 50.

$$Flow_{segment} \leq MaxFlow_{segment}$$

Equation 12: Segment maximum flow constraint

This segment flow limit is necessary in order to effectively model the parabola… without it the solver would schedule all of the flow on the segment with the lowest loss-flow ratio.

```
br00_brSeg00:
BrSegLossForFlow(LTE) constraint:
Shadow Price: $0.00
 +1.00000*br00_brSeg00_{SegLossPos}
-0.02093*br00_brSeg00_{SegFlowPos}  <= 0.00000

br00_brSeg00:
BrSegLossForFlow(GTE) constraint:
Shadow Price: $76.92
 -1.00000*br00_brSeg00_{SegLossPos}
+0.02093*br00_brSeg00_{SegFlowPos}  <= 0.00000

br00_brSeg00:
BrSegLossForFlow(LTE) constraint:
Shadow Price: $0.00
 +1.00000*br00_brSeg00_{SegLossNeg}
-0.02093*br00_brSeg00_{SegFlowNeg}  <= 0.00000

br00_brSeg00:
BrSegLossForFlow(GTE) constraint:
Shadow Price: $0.00
 -1.00000*br00_brSeg00_{SegLossNeg}
+0.02093*br00_brSeg00_{SegFlowNeg}  <= 0.00000
```

Figure 49: Constraints display: Segment flow linked to segment loss (one segment shown)

```
br00_brSeg00:
BrSegFlowMax(LTE) constraint:
Shadow Price: $5.31
 +1.00000*br00_brSeg00_{SegFlowPos}  <= 93.03062

br00_brSeg00:
BrSegFlowMax(LTE) constraint:
Shadow Price: $0.00
 +1.00000*br00_brSeg00_{SegFlowNeg}  <= 93.03062

br00_brSeg01:
BrSegFlowMax(LTE) constraint:
Shadow Price: $0.00
 +1.00000*br00_brSeg01_{SegFlowPos}  <=
113.93877

br00_brSeg01:
BrSegFlowMax(LTE) constraint:
Shadow Price: $0.00
 +1.00000*br00_brSeg01_{SegFlowNeg}  <=
113.93877

br00_brSeg02:
BrSegFlowMax(LTE) constraint:
Shadow Price: $0.00
 +1.00000*br00_brSeg02_{SegFlowPos}  <= 93.03072

br00_brSeg02:
BrSegFlowMax(LTE) constraint:
Shadow Price: $0.00
 +1.00000*br00_brSeg02_{SegFlowNeg}  <= 93.03072
```

Figure 50: Constraints display: Each branch segment has its own limit

Options for assigning branch losses

The constraints shown above describe how losses are calculated. There also needs to be some way of subtracting these losses from the branch flow. This

is achieved by including the branch losses as an outflow in the node balance constraint.

The app offers two options for applying the branch losses to the node balance constraint:

- Assign half the branch losses to the sending bus and half to the receiving bus.
- Assign all branch losses to the receiving bus.

The app provides the option of selecting either method, via the solver setting shown in Figure 51.

Figure 51: Solver setting for loss allocation

Bus constraints to model branch losses

If the "Rcv Bus" option for Loss Location is selected then a branch's losses are assigned to the bus that receives power from the branch, and the node balance constraint is as shown in Equation 13.

$$\sum Bus_{FlowIn} - \sum Bus_{FlowOut} - \sum_{\substack{segment \\ FlowIn}} Loss_{segment} = 0$$

Equation 13: Node balance: Losses assigned to receiving bus

If the "50/50" option is selected then the node balance constraint is as shown in Equation 14.

$$\sum Bus_{FlowIn} - \sum Bus_{FlowOut} - 0.5 \times \sum_{\substack{segment \\ FlowIn \\ FlowOut}} Loss_{segment} = 0$$

Equation 14: Node balance: Losses assigned 50/50 to sending bus and receiving bus

Viewed via the app these constraints appear as shown in Figure 52 and Figure 53.

Note: Unless otherwise stated the examples use the "Rcv Bus" setting, i.e., losses are assigned to the receiving end bus. The New Zealand electricity market originally assigned dynamic losses 50-50 but now assigns them at the receiving end.

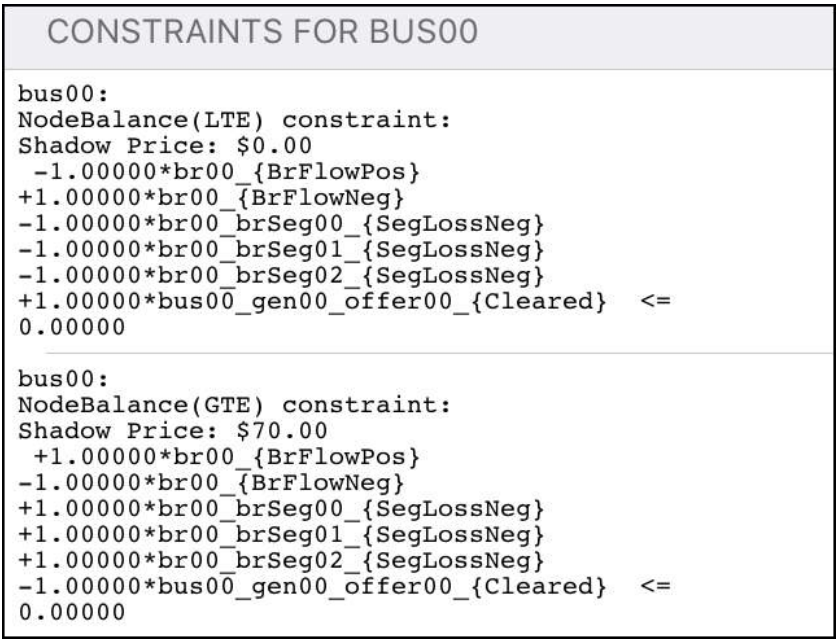


Figure 52: Bus constraints for losses at receiving end

CONSTRAINTS FOR BUS00

```
bus00:
NodeBalance(LTE) constraint:
Shadow Price: $0.00
 -1.00000*br00_{BrFlowPos}
+1.00000*br00_{BrFlowNeg}
-0.50000*br00_brSeg00_{SegLossPos}
-0.50000*br00_brSeg01_{SegLossPos}
-0.50000*br00_brSeg02_{SegLossPos}
-0.50000*br00_brSeg00_{SegLossNeg}
-0.50000*br00_brSeg01_{SegLossNeg}
-0.50000*br00_brSeg02_{SegLossNeg}
+1.00000*bus00_gen00_offer00_{Cleared}  <=
0.00000

bus00:
NodeBalance(GTE) constraint:
Shadow Price: $70.00
 +1.00000*br00_{BrFlowPos}
-1.00000*br00_{BrFlowNeg}
+0.50000*br00_brSeg00_{SegLossPos}
+0.50000*br00_brSeg01_{SegLossPos}
+0.50000*br00_brSeg02_{SegLossPos}
+0.50000*br00_brSeg00_{SegLossNeg}
+0.50000*br00_brSeg01_{SegLossNeg}
+0.50000*br00_brSeg02_{SegLossNeg}
-1.00000*bus00_gen00_offer00_{Cleared}  <=
0.00000
```

Figure 53: Bus constraints for losses assigned 50/50

Per-unit values and loss calculation

The calculation of power flow is potentially complicated by the fact that different parts of the power system run at different voltages. In order to be able to ignore these voltage differences the simplified power flow equation uses susceptance values that have been adjusted so that they take

into account the branch's nominal voltage. This is referred to as using per-unit values. As you may have noticed, the resistance, reactance and susceptance values used by the model are all quoted in per-unit, abbreviated to p.u., e.g., see Figure 54.

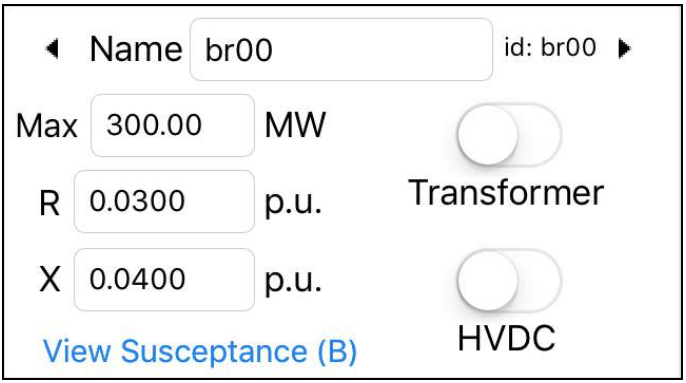


Figure 54: The model uses per-unit values

These per-unit values were calculated prior to being entered into the app. Per-unit values are calculated by dividing the actual values by a base value, e.g., per-unit resistance is calculated as follows:

$R_{per\text{-}unit}=R_{actual}/R_{base}$

The base value depends on the voltage. The base resistance is calculated from a base voltage value and a base power value…

$R_{base}=V_{base}^2/P_{base}$

…where the base voltage is the nominal voltage of the component, e.g., for a branch designed to run at 220kV, the base voltage is 220kV.

To calculate the base resistance there also needs to be a base power value. The base value is arbitrary but the same value must be used consistently in the calculation of all the per-unit values that are provided to a model.

For a power system the per-unit values are commonly specified in terms of a 100MVA base power value. When combined with the typical voltages of a power system, the 100MVA base power value results in per-unit resistance, reactance and susceptance values that are not too big and not too small.

For the power flow calculation where all flows are relative, the fact that the per-unit values are calculated using a 100MVA base can be ignored. However, to calculate the branch losses the 100MVA base will impact the calculations. Therefore, as part of the branch loss calculation the app divides the per-unit resistance value by 100 in order to account for the presence of the 100MVA base in the original per-unit calculation.

If you want to enter per-unit values that did not use a 100MVA base, and you want to model losses, then

you will need to convert the per unit values before entering them into the app. For example, if you had per-unit resistance and reactance values that were calculated using a 10MVA base then you would need to multiply their values by 10 before using them in the app.

Transmission rentals

Congestion charges and transmission rentals

In the transmission tutorial we saw that a binding branch can result in the total payments made by the load exceeding the total payments received by the generator. The difference between these amounts is displayed as the $Grid value on the Results display.

Branch losses can also give rise to $Grid, as shown by the results in Figure 55, which are for the single branch model of Figure 43, where the branch is not binding. These $Grid are due to line losses and are referred to as transmission rentals.

Figure 55: Branch losses give rise to transmission rentals

Non-parabolic loss model has no transmission rentals

The cause of transmission rentals is the parabolic nature of the relationship between branch flow and branch losses. If the branch losses are not parabolic then there are no transmission rentals. We can demonstrate this by editing the single branch model so that the branch only uses the first flow-loss segment; if the solver *only* uses the first segment

then the relationship between flow and loss is a straight line.

For the line loss example that we already have (Figure 43), the branch segments in Figure 45 show that the first segment has a maximum flow of 93.03MW. To only use this first segment, edit the load to reduce its quantity from 100MW to 90MW. As shown by the result in Figure 56 the transmission rentals are now $0.

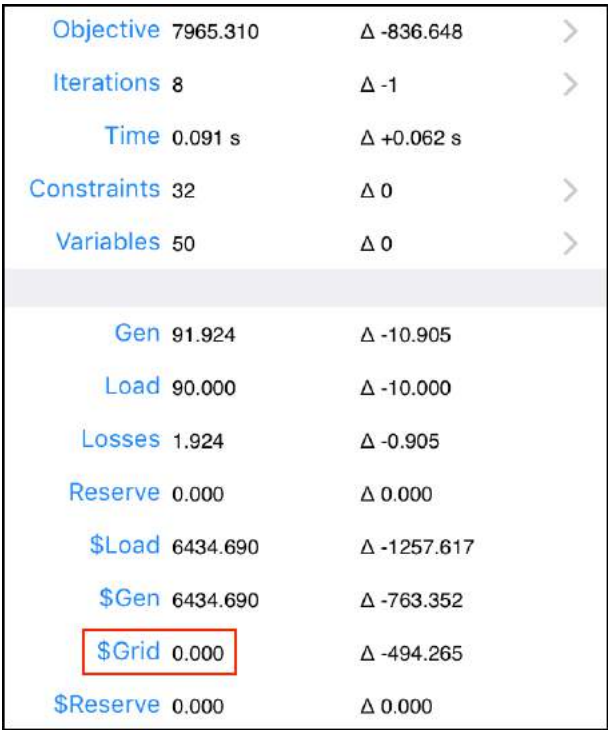


Figure 56: Rentals $0 when only first branch segment used

Why straight line losses have no loss rentals

When only the first branch segment is used, the relationship between flow and losses is a straight line. Although there is a price difference between the load bus and the generation bus this price difference only represents the value of the power that was dissipated as losses; the load pays a higher $/MW price because it is effectively paying for the quantity of power that was generated, but only receiving the power that was delivered. Overall the load pays the same amount as the generator receives and there is no extra payment.

Because the bus price represents the $/MW cost of the *next* MW, with a straight-line loss function the $/MW cost of the next MW is the same as the cost of the scheduled MW, because the losses per MW for the next MW are the same as the losses per MW for the scheduled MW.

Why parabolic losses have loss rentals

The actual physical losses are parabolic. Hence, if we could model a parabola then the $/MW losses associated with the *next* incremental MW would not be the same as the per MW losses associated with the scheduled MW.

As a crude example… with parabolic losses if a scheduled flow of 1MW incurs a loss that is 10% of flow, then we can infer that the resistance is 0.1 per unit, i.e., loss = 0.1 x flow^2. An "incremental" 1MW would take the flow to 2MW and incur a loss of 0.1 x 2^2 = 40%. The bus price is set by the cost of the incremental MW therefore the bus *price* would be set based on a 40% loss, even though the scheduled load is only incurring a 10% loss. The load pays for losses that are not actually incurred, but which would be incurred by the "next" MW. Hence the load pays more than the generator receives.

How the piece-wise parabola causes loss rentals

The LP model cannot include a parabola. But when the piece-wise linear loss model schedules flow that uses more than the first flow-loss segment, it will begin to approximate a parabola, and this will be reflected in the loss rentals.

To see how this happens, we will adjust the model so that it is using more than the first flow loss segment. We could achieve this by adjusting the load again, but it will be more interesting if we increase the number of flow loss segments… on the Branch Segments display for br00, tap the "4" button to increase the number of branch segments from 3 to 4. As shown in Figure 57, the display

adjusts immediately to show the new segments and we can see that going to a four-segment model will move the scheduled flow into the second segment.

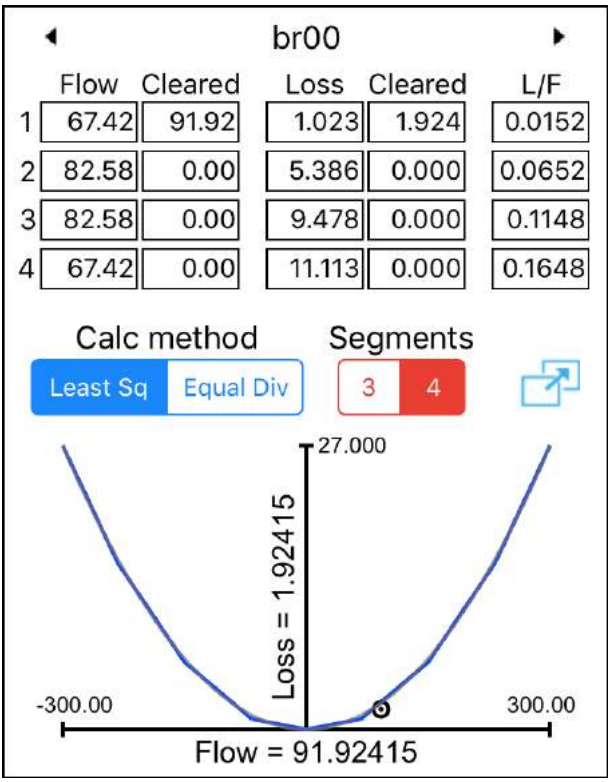


Figure 57: Change to 4 segments (not yet re-solved)

The "Segments" button is highlighted red to indicate that this change has not yet been applied to the model… this will happen when you leave the display via the Back button. Tap the Back button and then

tap "OK" when you are prompted to apply the change.

After solving with 4 segments, the flow-loss result is shown in Figure 58 and the corresponding loss rentals are shown in Figure 59.

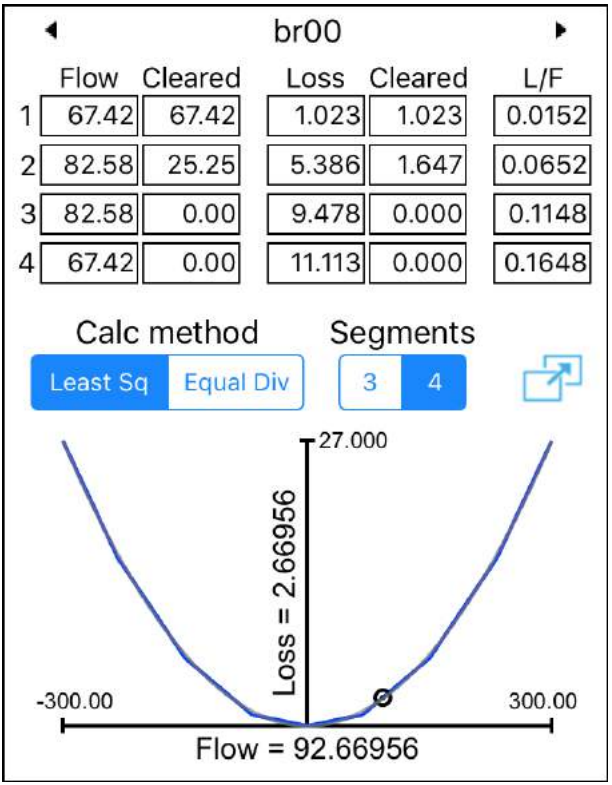


Figure 58: Flow-loss result after solving with 4 segments

Figure 59: Increasing to 4 segments has brought back the loss rentals

This result includes loss rentals because the load receives the final 25.25MW of its 90MW via the second flow-loss segment. The value of the next MW, and hence the bus price, reflects the flow-loss ratio of the second segment. Hence, the load pays as if all of the 90MW were delivered with this flow-loss ratio.

If all of the 90MW had incurred losses at the rate associated with the second segment, then the payments made by the load would match the payments received by the generation.

However, the first 67.42MW is only subject to the 0.0152 flow-loss ratio of the first segment, compared to the 0.0652 of the second segment that set the price. The generation *quantity*, and hence the amount paid to the generator, reflects the fact that some of the losses were incurred at a lower ratio, but the load pays at a price that was set as if all of the generation quantity was subject to the higher flow-loss ratio of the second segment. Hence the load pays more than the generation receives… with the difference referred to as transmission rentals.

Non-physical losses

Non-physical losses due to negative prices

When branch losses are included in the model, negative prices can lead to non-physical losses. As explained in the Spring Washer tutorial, negative prices occur at a bus where a decrease in available power would benefit the objective value.

One way to decrease the available power at a bus is via load. Another way is via branch losses. Hence, a negative price at a bus indicates that there is a

benefit to increasing the branch losses assigned to that bus.

Increasing branch losses

Within the constraints of the model it is possible to increase branch losses because while branch flow is the sum of the branch segment flows, the only thing ensuring that the "correct" segments are used is that usually the solver is trying to minimize the branch losses. Using the "correct" segments means that as the scheduled flow increases, its flow-loss ratio approximates a parabola.

When there are negative prices the solver can use branch segments that maximize the losses. The scheduled flow-loss no longer follows the parabola and the resulting losses are an over-estimate of the actual physical losses that would occur for the scheduled flow.

The losses that are over-estimated are referred to as non-physical losses. The associated result schedules more generation than would actually be required.

Demonstrating non-physical losses

To demonstrate non-physical losses, build the spring washer model shown in Figure 60. Leave all components with their default values except change

the limit on br01 to be 20MW. Ensure that branch losses are set to "Least Squares" and 3 segments (via the Branch-Losses display), with losses assigned at the receiving end of the branch (via Solve Settings).

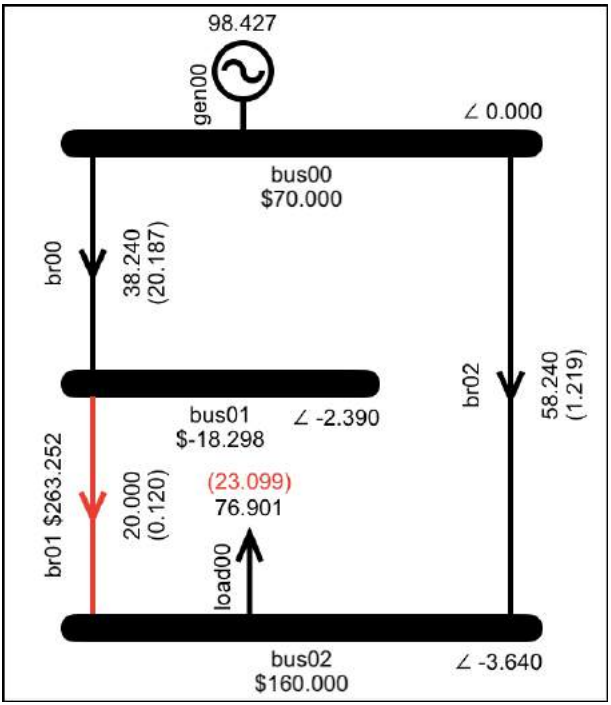


Figure 60: Spring washer with losses

Figure 60 also shows the results. This is the same model that we built in the spring washer example, and again we have negative prices at bus01. What is different is that this time the model includes line losses and the line losses on br00 are 20.187MW.

This level of losses is not physically realistic on a branch that is only transmitting 38.24MW. The loss curve in Figure 61 shows how the solver has achieved the non-physical losses on br00.

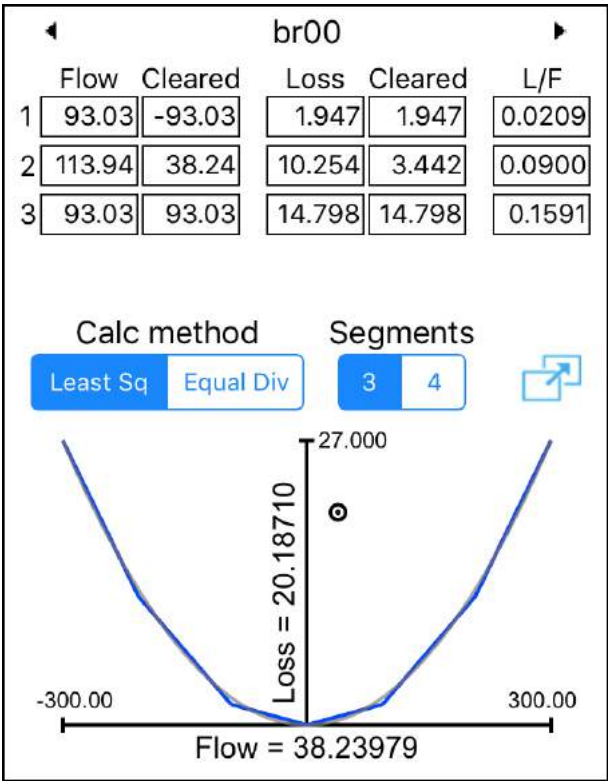


Figure 61: Spring washer with non-physical losses on br00

Because the solver can use any of the segments including those in the reverse directions, it has fully scheduled segment 1 in the reverse direction, balancing this flow by fully scheduling segment 3 in

the forward direction… the solver is scheduling circulating branch flows in order to increase the losses assigned to bus01. While the scheduled flow of 38.24 is transferred via segment 2, the losses are the sum of the cleared losses on all three segments.

Branch segments as viewed by the solver

Although the app displays the flow loss curve as a parabola, a graphical view of the constraints is as shown in Figure 62.

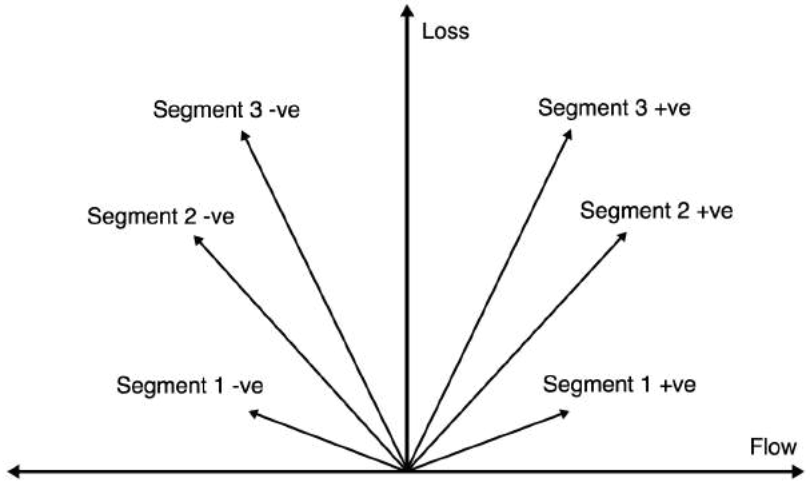


Figure 62: Branch segment constraints

In the normal situation, with no negative prices, the solver is minimising losses and will use the segments with the lowest loss-flow ratio first. The solver will schedule flow on segment 1 to its maximum, then segment 2 and then segment 3… in

this way the scheduled flow and loss will track the parabola.

When negative prices incentivise the solver to maximise losses it can utilize circular flows to schedule the segment flows "out of order", resulting in the total scheduled flow leaving the parabola as we saw in Figure 61.

Removal of non-physical losses

In an actual electricity market, any non-physical losses are detected in post-processing and removed by re-solving the schedule with some of the branch segments removed.

The removal of branch segments from the model is achieved either via post-processing logic which removes the branch segments and then re-runs the solve, or via post-processing logic that re-runs the solve as a mixed integer solution whereby the solver is forced to make binary decisions regarding which segments to include in the solution.

Summary

In this section we demonstrated how parabolic transmission losses are modelled in a linear solver by using a piece-wise linear curve.

We explained why the modelling of transmission losses gives rise to transmission rentals, whereby

the amount paid by the load is greater than the amount paid to the generator.

We demonstrated that negative prices have the potential to cause the solver to produce circulating branch flows that result in non-physical losses, and we explained how and why this happens.

Tutorial 5: Risk and Reserve

This tutorial describes risk and reserve, investigates how they are modelled, and explains the results that they produce.

Managing risk

The System Operator is responsible for ensuring that the power requirements of the load are reliably met. This includes ensuring that if the generator with the largest power output (the *risk* generator) were to suddenly become unavailable (to *trip*), then the load requirement could still be met. This reliability is achieved by ensuring that there is spare generating capacity sufficient to replace the power output of the largest risk generator.

Because the replacement generating capacity is held in reserve, it is referred to as generating reserve, or reserve. The power output of the largest risk generator is referred to as the risk. Hence it is said that reserve is scheduled to cover the risk.

Switch losses off for the reserve examples

In order to make it easier to explain risk and reserve we will take branch losses out of the equation (literally) by solving all the examples in this section with losses selected to OFF.

Risk setters

The generator with the largest scheduled power output is the risk setter, but only if it is flagged as a potential risk setter.

A "generator" that offers into the electricity market may consist of a single physical generating unit, or it may represent a generating station that consists of a number of separate generating units.

If a generator consists of a single generating unit then if that unit trips all of the generator's power output is lost. If this generator is large enough to be significant, it is flagged as a potential risk setter.

However, if a generator represents a generating station that consists of a number of generating units, then provided each individual unit is small enough to not significantly impact the system, and provided there is no single credible contingency that could result in a significant number of units tripping at the same time, this generator (i.e., the generating station) is not flagged as a potential risk setter.

Reserve providers

If a generator trips, then the power that it is was generating is lost from the power system… the total generation on the system will no longer match the

total load and therefore the frequency of the electricity will drop. A certain level of frequency deviation can be tolerated, but in response to a significant drop in frequency the generators that are providing reserve will automatically increase their generation… their spare generating capacity, which is functioning as reserve, will respond by becoming generation.

If the frequency drops too low then electrical equipment can be damaged, and if it drops further still then other generators will trip, leading to a cascade failure. Hence the reserve must respond to arrest the frequency decline in a timely fashion, e.g., 6 seconds or less. Not all generators are capable of meeting this requirement… while a generator may have spare capacity, if this capacity is not able to respond quickly enough then it cannot be offered as reserve. Hence not all generators are reserve providers, i.e., while all generators have energy offers, not all of them have reserve offers.

Example of not covering the risk

Produce the result shown in Figure 63 by tapping Bus-Bus-Gen-Gen-Gen-Load-Branch, then editing the offer for gen00 to be 60MW at $65/MWh.

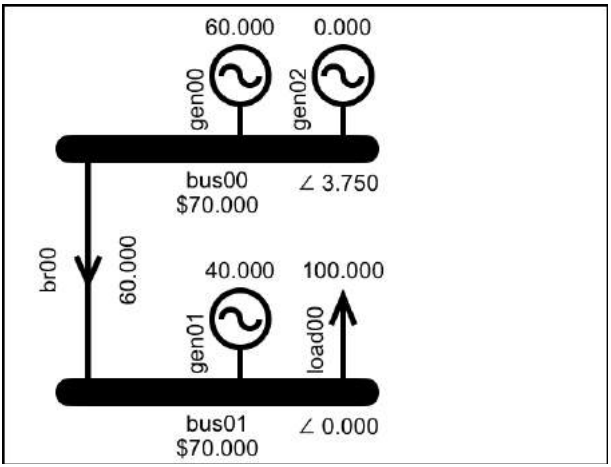


Figure 63: A model with the risk not covered

In this model we will suppose that gen00 is not a potential risk setter and also that it is the only generator capable of providing reserve. In this scenario if gen01 were to trip then there would be no reserve available to replace its lost capacity in a timely fashion; gen00 has no spare generating capacity because all of its 60MW of capacity is cleared as generation, and gen02 is not capable of providing reserve.

Example of reserve covering risk

To ensure that there is sufficient reserve to cover the risk, we will now include reserve cover as a requirement in our model. The first step is to add some reserve offers.

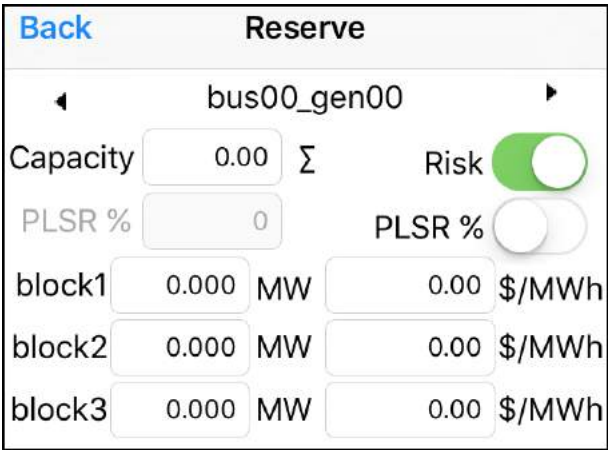


Figure 64: Reserve display for gen00 before data entry

From the gen00 Data Display, tap the Reserve button. Before any changes are made, the Reserve display will appear as shown in Figure 64.

On the Reserve display for gen00:

- Enter a reserve offer of 60MW at $2
- Switch the "Risk" button to OFF
- Tap the "∑" button to set the Capacity to be equal to the sum of the Energy Offer quantities

We will discuss PLSR% later, but for now we will leave PLSR% switched OFF. After making these changes the Reserves display for gen00 is shown in Figure 65.

Figure 65: Reserve display for gen00 after data entry

After making these changes, open the Solve Settings display (by tapping the Solve button). Change the "Include Reserves" setting to ON as shown in Figure 66.

Figure 66: Solve option "Include Reserves" selected to ON

Now, before solving, go back to the network model display. It should look like Figure 67. The "Include Reserves" button has switched on the risk indicators… the black scissor symbols indicate

generators that have their Risk switch set to YES, i.e., they are the potential risk setters.

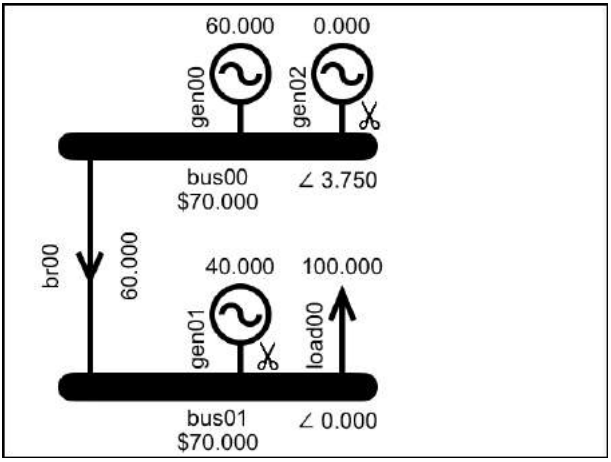


Figure 67: Scissor symbols indicate potential risk setters

Now go back to the Solve menu and tap the "Solve Now" button. In the result, shown in Figure 68, the red scissors indicate potential risk setters that have ended up setting the risk.

Both of the risk setters are presenting a risk of 40MW, which is covered by 40MW of reserve on gen00.

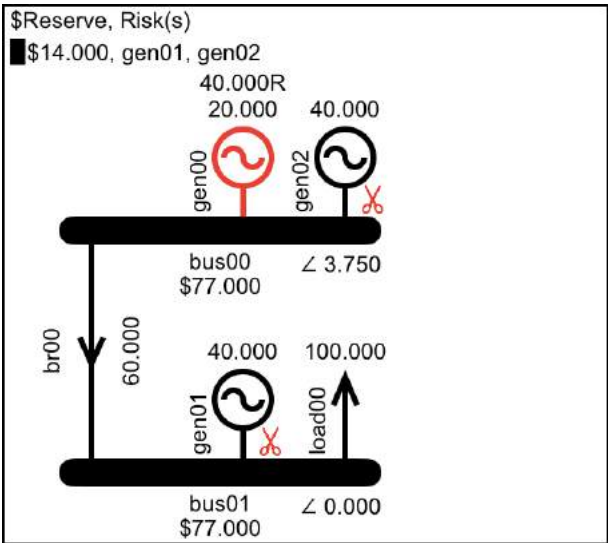


Figure 68: Result with reserve, indicated by R, scheduled on gen00 covering risk presented by gen01 and gen02

Capacity constraint

In the result shown in Figure 68, gen00 is coloured red because it is binding on its capacity constraint. The capacity constraint, shown in Equation 15, enforces the requirement that, when a generator has reserve offers, the total of its scheduled generation and reserve must not exceed the capacity limit of the generator.

$$ClearedEnergy_{Gen} + ClearedReserve_{Gen} \leq Capacity_{Gen}$$

Equation 15: Generator capacity constraint

If the generator does not have any reserve offers, i.e., it only has energy offers, then the capacity constraint is not created because it is not required… when there are only energy offers then the sum of the energy offers sets the maximum that can be requested from the generator. When there are energy and reserve offers, if there was no overarching restriction in place then there would be nothing to stop the cleared energy and the cleared reserve from both reaching the capacity of the generator. The capacity constraint forces the solver to decide how to apportion the generator's capacity between cleared energy and cleared reserve, i.e., to co-optimise the energy and reserve.

Avoiding a zero-capacity constraint

If a generator has reserve offers and the solve has reserve enabled, then a capacity of zero would limit the sum of the generator's energy and reserve to zero.

The app could be written so that the capacity limit was always set to be the sum of the energy offers but this would somewhat defeat the purpose of including the capacity limit as an enterable parameter.

Currently the default capacity limit is zero, because until you enter reserve offers, the capacity limit

does not apply. Once reserve offers are entered, if there are non-zero reserve offers but the capacity is zero then the capacity field will be highlighted red and the ∑ button will be displayed, as shown in Figure 69.

If the capacity is non-zero but does not match the sum of the offers then the ∑ button will be displayed, but the colour field will be black.

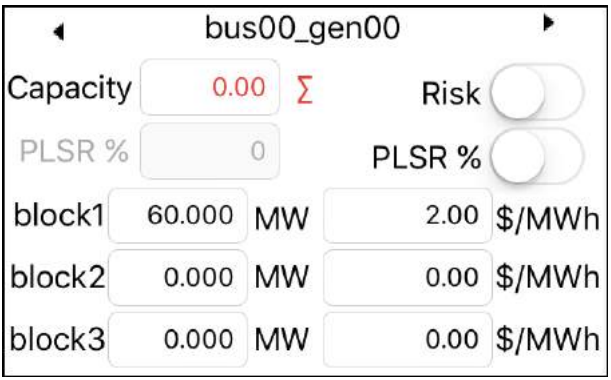


Figure 69: Zero capacity is highlighted red if non-zero reserve offers exist

To set the capacity to be the sum of the energy offers, tap the ∑ button. When the capacity matches the sum of then energy offers then the ∑ button will no longer be displayed.

At some stage you may want to model a situation where the capacity is not equal to the sum of the

energy offers, in which case you can manually enter a capacity value.

If you leave the display with non-zero reserve offers in place, but the capacity is still set to zero, then you will be presented with the alert shown in Figure 70.

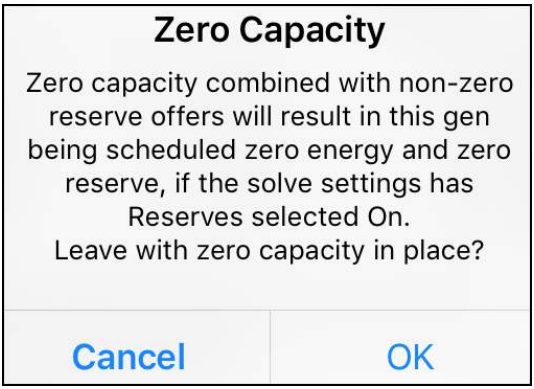


Figure 70: Zero capacity warning

Island largest risk

The largest risk in each electrical island is determined by the risk calculation constraints. For now, we are only looking at one electrical island, and generator risks, i.e., AC risks. Multiple islands and HVDC risks are covered in Tutorial 7: HVDC Link.

When a model is solved with reserves enabled, each island is assigned a LargestRisk variable and the risk calculation constraint shown in Equation 16 is created for each potential risk setter. The constraint

requires that the island's LargestRisk variable be >= the risk presented by any potential risk setter in the island.

$$ClearedEnergy_{RiskGen} + ClearedReserve_{RiskGen} \leq LargestRiskAC_{Island}$$

$$\forall RiskGen\ in\ Island$$

Equation 16: Largest Risk Constraint for generators

While it is only the loss of energy that will impact the system frequency, the generator risk calculation includes cleared reserve. This allows the reserve on a risk generator to be used to cover the risk of any other generator, while not covering its own risk.

Viewing the risk constraint

There is a risk calculation associated with every potential risk setter. Figure 71 shows the risk calculation constraint for gen02.

CONSTRAINTS FOR GEN02

```
bus00_gen02_offer00:
OfferBlockMax(LTE) constraint:
Shadow Price: $0.00
 +1.00000*bus00_gen02_offer00_{Cleared}  <=
250.00000

bus00_gen02:
CalcRiskEachGen(LTE) constraint:
Shadow Price: $2.00
 +1.00000*bus00_gen02_offer00_{Cleared}
-1.00000*island01_{LargestRiskAC}  <= 0.00000
```

Figure 71: Constraints for a potential risk setter

"Reserve covers risk" constraint

Each island has the "reserve covers risk" constraint shown in Equation 17 to ensure that the sum of the cleared reserve offers in the island is sufficient to replace the island's largest risk.

Equation 17: Reserve covers risk constraint

$$\sum_{\substack{Gen \\ in\ Island}} ClearedReserve_{Gen} \geq LargestRiskAC_{Island}$$

The constraints for an island can be viewed via the Constraints option on the Results display, as shown in Figure 72.

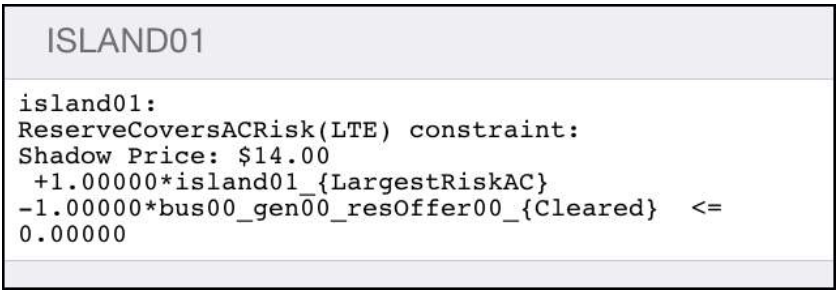


Figure 72: The "reserve covers risk" constraint on the Constraints display

The risk constraints become more interesting when multiple islands are involved, as shown Tutorial 7: HVDC Link.

Reserve offers in the objective function

Cleared reserve offers are included as a cost to the objective value, as shown in Equation 18.

Equation 18: Objective function including reserve offers

Maximize:

$$ObjectiveValue$$
$$= loadBid_{Cleared} \times loadBid_{Price}$$
$$-genOffer_{Cleared} \times genOffer_{Price}$$
$$-reserveOffer_{Cleared} \times reserveOffer_{Price}$$

The objective value calculation for the latest solve can be viewed via the Objective option on the Results display, as shown in Figure 73.

Figure 73: Reserve offers in the objective value calculation

Explaining the reserve price

To explain the reserve price in Figure 68 we can use the same mechanism that was used to explain bus prices, i.e., add a $0 quantity and see how it improves the objective value. To this end, add gen03 to bus01 as shown in Figure 74, change its energy offer to be 0MW and add a 1MW reserve offer at $0, with capacity set to 1MW and "Risk" selected to OFF.

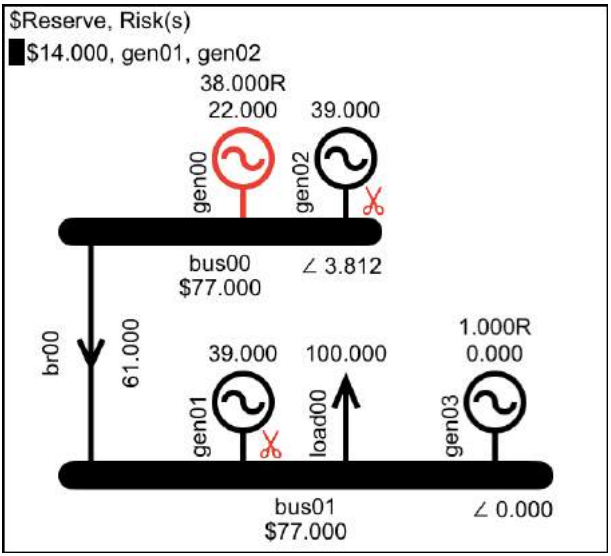


Figure 74: Explain reserve price by adding 1MW of $0 reserve

From the results display we can confirm that adding the $0 reserve has resulted in a $14 improvement in the objective value. From the result in Figure 74 we can explain this $14 improvement as follows…

Because gen00 was binding on its capacity constraint, the extra 1MW of free reserve allowed reserve on gen00 to reduce by 2MW… 1MW because of the 1MW of $0 reserve and 1MW because the risk was reduced by 1MW… where the risk was reduced by 1MW because the 2MW decrease in reserve freed up capacity on gen00 allowing for a 2MW increase in generation, which in turn allowed the generation on the risk setters to

reduce by 1MW each. This reduction in risk saves 2MW of scheduled reserve on gen00 at 2 x $2 = $4, and the energy at gen00 is cheaper by $5/MWh than the gen01 and gen02 energy that it replaced, saving 2 x $5, for a total of $14.

Explaining the energy price

We are going to explain the $77/MW energy price by zeroing the reserves on the dummy generator gen03 and solving to take us back to our original result, then giving gen03 an energy offer of 1MW at $0. The result is shown in Figure 75.

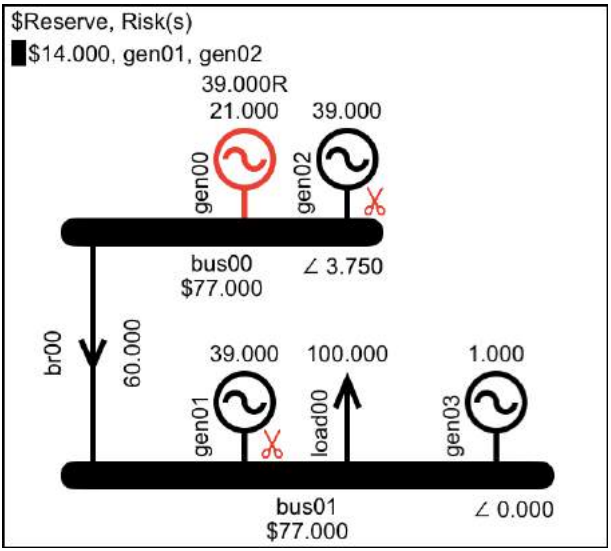


Figure 75: Explain energy price by adding 1MW of $0 gen

The 1MW of $0 energy allows the risk to be reduced by 1MW on both of the risk setters (gen01 and gen02)… 1MW of energy from gen01 is replaced by the extra 1MW, and 1MW of energy from gen02 is replaced by increasing gen00 generation by 1MW, which is possible within the limits of the capacity constraint because the reserve requirement has decreased by 1MW.

Overall the benefit is due to 1MW less energy from gen01, which saves $70, plus the $2 reserve saving due to 1MW less risk, plus the benefit due to the increased generation at gen00 being $5 cheaper than the energy at gen02 that it replaces, for a total benefit of $77.

Note that because the result that explained the reserve price, i.e., Figure 74, still had the $77/MWh energy price, we could have added the $0 energy and explained the $77/MW price using that model (i.e., with the $0 reserve offer still in place), but it would be a different explanation because it explains a different result (even though it would explain the same price).

Co-optimisation of energy and reserve

As shown above, by adding 1MW of energy we can explain the energy price, and by adding 1MW of reserve we can explain the reserve price. However,

because of the various trade-offs that are being made when the model is solved, it would be difficult to explain the result without employing these methods.

The process of making the trade-offs between energy, risk, reserve and capacity is referred to as the co-optimisation of energy and reserve.

PLSR% Constraint

A reserve provider's current level of generation can influence its ability to increase generation in response to falling frequency. For example, a generator that is generating 10MW may not be able to provide another 30MW of generation within the rapid timeframe required when providing reserve. However, if the reserve provider was generating 100MW, then another 30MW may not be a problem.

To model the situations where this limitation will impact on the ability of a generator to effectively provide reserve, the LP model includes the Partly Loaded Spinning Reserve percentage (PLSR%) constraint.

The PLSR acronym refers to reserve (R) that is available when the generator is already generating, i.e., spinning (S), and has spare capacity, i.e., is partly loaded (PL). There are other means of

providing reserve (not mentioned here) that do not involve PLSR; the PLSR% constraint only applies to PLSR.

The PLSR% constraint restricts the maximum cleared reserve (specifically PLSR) to a percentage of the cleared energy, as described by Equation 19.

Equation 19: PLSR% constraint

$$ReserveCleared_{Gen} \leq PLSR\%_{Gen} \times EnergyCleared_{Gen}$$

Demonstrating the PLSR% Constraint

Demonstrate the PLSR% constraint by building and solving the model shown in Figure 76, using the default energy offers, and the reserve offers shown in Table 3. Note that you can move between the reserve displays for the various generators by using the navigation buttons highlighted in Figure 77.

	Reserve offer quantity	**Reserve offer price**	**Can Set Risk**	**PLSR%**
gen00	60MW	\$2/MWh	Y	50
gen01	60MW	\$80/MWh	Y	OFF
gen02	0	0	Y	OFF

Table 3: Reserve data for PLSR% example

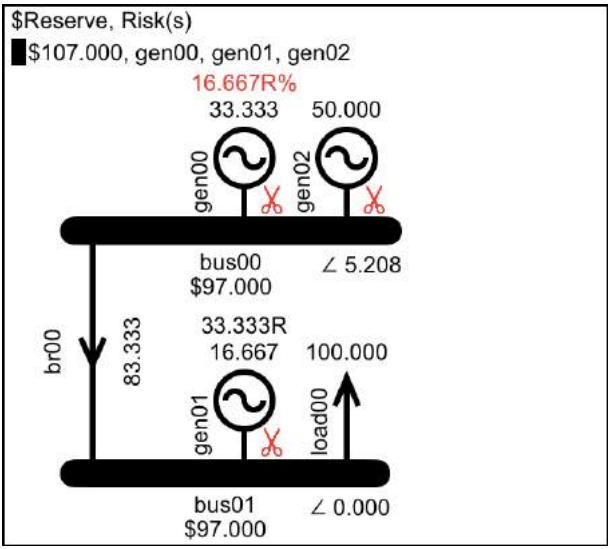


Figure 76: PLSR% constraint applied to gen00

The PLSR% value of 50 is entered by switching on the PLSR% option, as shown in Figure 77, and then entering the value. As per Equation 19, this PLSR% value will result in the maximum reserve that can be cleared by gen00 being capped at 50% of its cleared generation.

Figure 77: Switch ON PLSR% to enter a PLSR% value (navigation buttons indicated)

Solve the PLSR example with the solve settings for "Include Reserve" and "Include PLSR%" set to ON.

The result in Figure 76 shows that the solver has wanted to schedule gen00's cheap reserve but the PLSR% constraint has capped the reserve to 50% of the generation. The reserve quantity for gen00 is displayed in red with a % symbol after it to indicate that the scheduled reserve is binding on the PLSR% constraint, i.e., the solver would have scheduled more reserve on gen00 if it were not for the PLSR% constraint.

Summary

This tutorial demonstrated how risk and reserve constraints are employed to ensure that generation capacity is available to respond in timely fashion in

the event that energy supply (generation) is unexpectedly disconnected from the power system.

We saw how the result balances the requirement to schedule reserve with the limits of generation capacity and the driving objective of minimizing overall cost, by co-optimising the cleared energy and cleared reserve quantities. Co-optimisation makes the resulting prices less intuitive, but they can still be explained by adding a suitably small reserve or generation quantity priced at $0 to demonstrate the value of an incremental MW.

We also explained the PLSR% constraint, which models the relationship between a generator's scheduled energy output and its ability to provide reserve.

Tutorial 6: Ramp Rates

This tutorial looks at how and why generator ramp rates are included in the model. It also explains how pre-processing works to minimise the number of constraints that are created.

The ramp rate constraint

The rate at which a generator can increase its generation in response to a *scheduled* change in quantity (as opposed to a reserve provider's *automatic* generation increase in response to a drop in frequency) is limited by the rate at which its fuel intake can be increased such that the increase is sustainable. This rate limitation is modelled by the ramp rate constraint shown in Equation 20.

$$EnergyCleared_{Gen} \leq$$
$$InitialMW_{Gen} + RampRate_{Gen} \times TimeInterval$$

Equation 20: Ramp Rate Constraint

Ramp rate example

To demonstrate the ramp-rate constraint, build the model shown in Figure 78 by tapping the buttons Bus-Bus-Gen-Gen-Load-Branch.

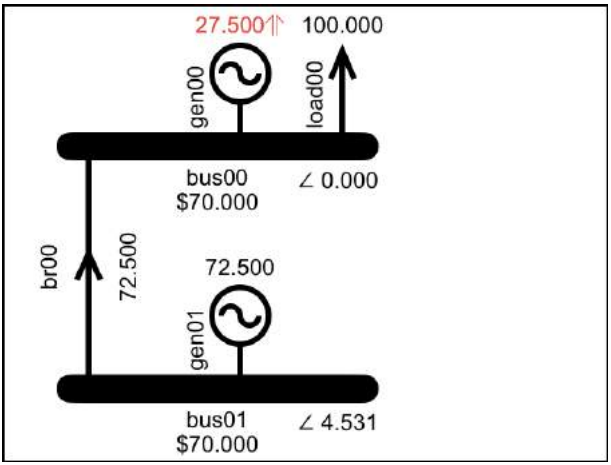


Figure 78: Model for ramp rate example

Edit gen00 to change its energy offer price to $7/MWh. Tap the Ramp button in the toolbar and enter a Ramp Rate Up value of 55, as shown in Figure 79. Leave the Initial MW as zero.

Solve the model with Losses and Reserves selected off, Ramp Rates selected on and a Time Interval of 30 minutes, as shown in Figure 80.

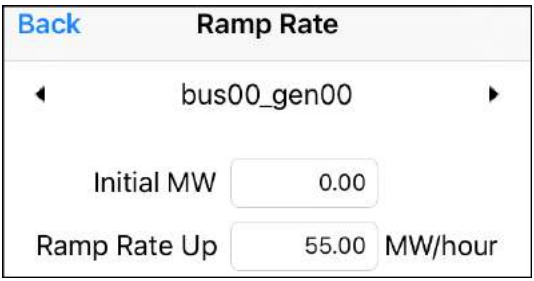


Figure 79: Ramp Rate data for gen00

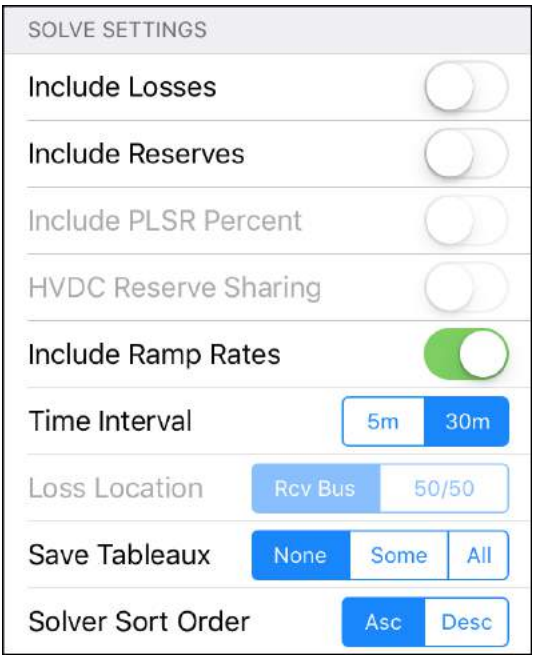


Figure 80: Solve Settings for ramp rate example

Explaining the ramp rate result

The ramp rate result is shown in Figure 78. The initial MW value entered on the Ramp Rate display is the generation at the beginning of the time interval. The time interval on the Solve Settings display is how much time the generator has to change its generation output from its initial MW value to the scheduled value… the results of the solve indicate the state of the system at the end of the time interval.

Gen00 has an initial MW value of 0MW and a ramp rate of 55MW/hour. Therefore, because we selected a time interval of 30 minutes, the maximum that gen00 can be scheduled is:

0MW + 55MW/hour x (30/60)hours = 27.5MW.

Because gen00 is cheaper than gen01 the solver schedules as much generation as it can from gen00 but is constrained by the 27.5MW limit set by the ramp rate constraint.

Indication of binding ramp rate

To indicate that generation is limited by the ramp rate, i.e., that the ramp rate is binding, the generation quantity for gen00 is coloured red and has an up-arrow suffix, as shown in Figure 78.

Ramp rate constraints

The ramp rate constraint that the app creates for gen00 is shown in Figure 81 (accessed via the ∑ button on the Data Display for the Gen component).

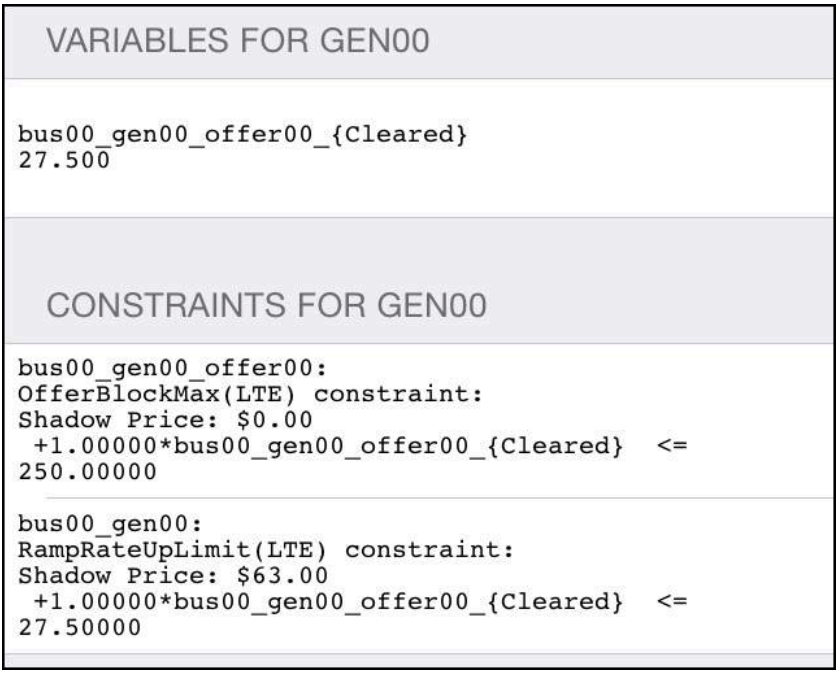


Figure 81: Constraints for gen00, including ramp-rate

Pre-processor exclusion of constraints

As you may have already noticed, there are no constraints created for bids and offers that have a quantity of zero. The creation of these constraints is excluded by pre-processing code that is run prior to the schedule being solved.

Similarly, when ramp-rates are enabled, the pre-processing code excludes the ramp-rate constraints for those generators where the sum of the generation offers is less than the maximum

generation limit that would be set by the ramp-rate constraint.

For example, if we look at the constraints for gen01 with the default up-ramp-rate of 99999, as shown in Figure 82 there is no ramp-rate constraint.

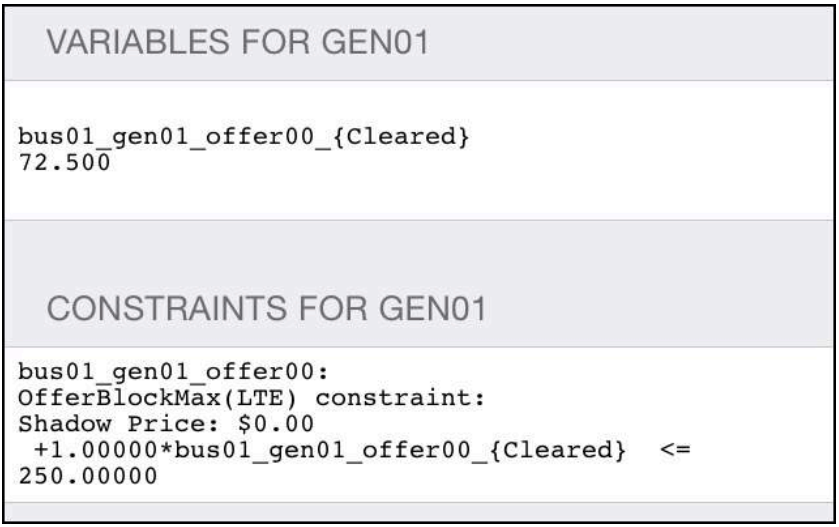


Figure 82: No ramp-rate constraint for gen01 because limit would be higher than sum of generation offers

Because of this pre-processing, and the high value that is used for the default up-ramp-rate, any generator that has the default up-ramp-rate is unlikely to have ramp-rate constraints created.

This makes it easy to investigate the impact of ramp-rates because only those generator's that have their ramp-rates edited will be affected by the

ramp-rate constraint… other generators can be left with the default ramp-rate in place and they are unlikely to have ramp-rate constraints applied.

Summary

In this tutorial we introduced the ramp-rate constraint that models the physical reality of the time it takes for a generator to increase its generation in a sustainable manner. Then we created an example to demonstrate the ramp-rate constraint in action.

We also saw how pre-processing excludes constraints that do not need to be applied, for example when bid or offer quantities are zero, or when the limit that would be imposed by the ramp-rate constraint is higher than the sum of the generation offer quantities.

Tutorial 7: HVDC Link

This tutorial explains how separate electrical islands are connected via an HVDC link, allowing for the controlled transmission of energy between the islands. The tutorial looks at how the HVDC link can set the risk, and how this risk can be mitigated if the HVDC link has more than one branch. Also covered is how the HVDC control system can be used to share reserve between islands via the HVDC link.

We start with a simple model and progressively add to it to demonstrate the various features of the HVDC link.

The finished model is also available in the Sample models, where it is named "HVDC Link".

Electrical Islands

The power produced by generators and consumed by the loads is alternating current (AC) power.

An AC electrical island represents a set of electrically connected components, i.e., generators, buses, branches and loads.

Connecting AC islands

If two isolated AC electrical islands are connected by an AC branch, i.e., the branches we have dealt

with so far, this will form a single AC electrical island. For example, build the model shown in Figure 83 using the default parameters. This model has two separate electrical islands.

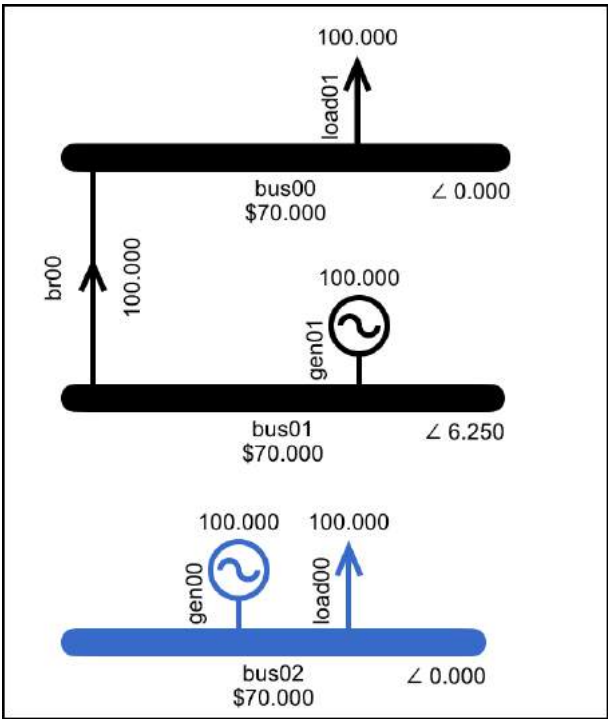


Figure 83: Two separate electrical islands

As soon as an AC branch is added between the two islands, they form a single island as shown in Figure 84.

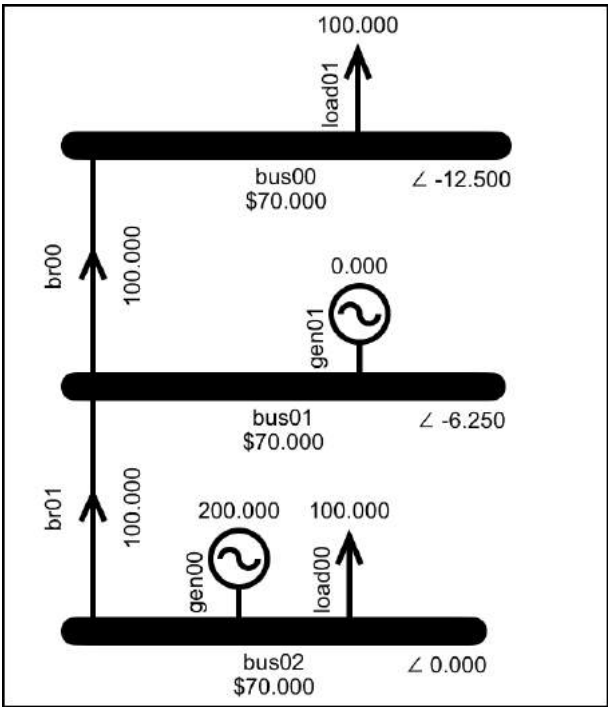


Figure 84: AC islands connected via an AC branch form a single island

HVDC Link

An HVDC link connects two AC systems while allowing them to remain independent. The control system of the HVDC link controls the power that flows between the two islands. The "HVDC" in HVDC link refers to the voltage of the link, i.e., High Voltage Direct Current. Power entering the HVDC link from an AC system is converted to HVDC and at

the other end of the link the power is converted back to AC and enters the other AC system.

Adding an HVDC link

The HVDC power is transmitted via one or more HVDC branches. An HVDC branch is a cable or transmission circuit. Physically the conductor is the same as an AC branch, with resistance, reactance and a maximum flow.

To add an HVDC link with a single HVDC branch, double tap the AC branch br01 and select the HVDC switch to the on position, as indicated in Figure 85.

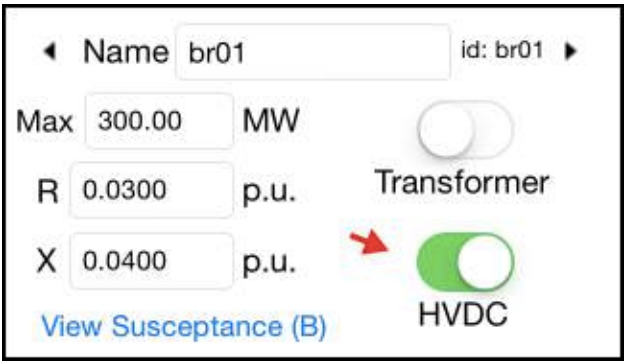


Figure 85: Branch Data Display with HVDC switch indicated

With br01 as an HVDC branch the two islands are separate, as shown in Figure 86. The symbol on the HVDC branch represents a thyristor. Thyristors are large semiconductors that are the key component in the conversion between AC voltage and DC voltage.

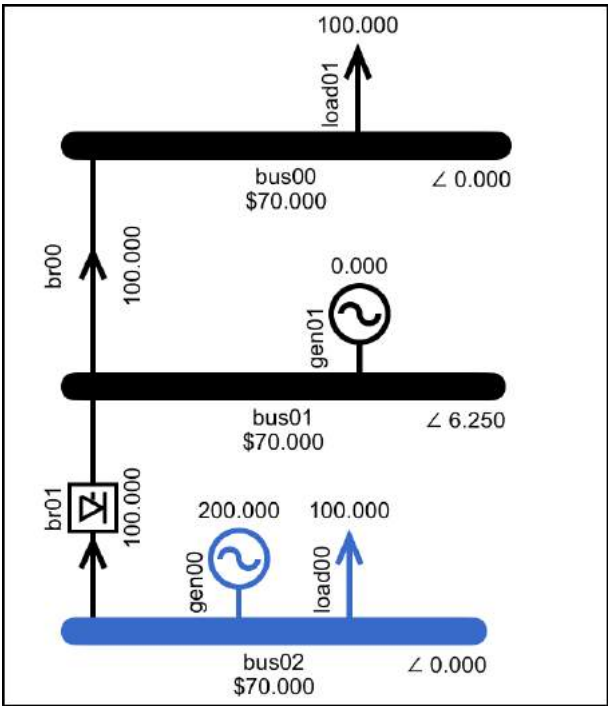


Figure 86: AC islands joined by an HVDC link

HVDC branch has no power flow constraint

The power flow in an AC branch is determined by the power flow constraint, as described in Tutorial 2: Modelling Transmission. The power flow constraint models the physical reality that a change in the voltage phase angle across a branch will result in a change in power flow.

With an HVDC branch it is the HVDC control system that determines the flow. Hence if the phase angle

changes at one end of the HVDC link there is no change in the HVDC branch flow, unless initiated by the control system.

The electricity market model represents this independence by not subjecting HVDC branches to the power flow constraint. This can be seen in Figure 86 where the phase angles either side of the HVDC branch are not related to the branch flow.

To see this more clearly, double tap bus01 and select it to be the reference bus for island 1 (as shown in Figure 87), then re-solve and the result is shown in Figure 88. Notice that the phase angles either side of the HVDC link are zero, and that despite this the flow is non-zero.

Figure 87: Setting bus01 to be the reference bus in island 1

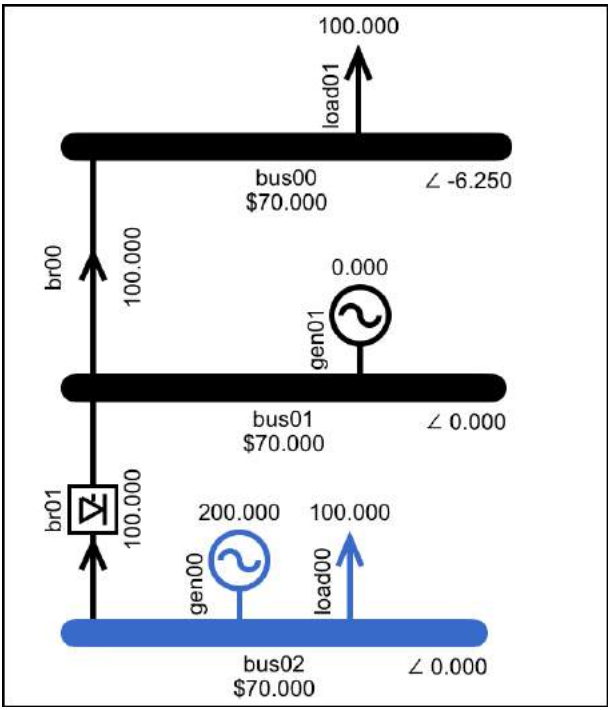


Figure 88: With bus01 as reference bus in island 1 it is easy to see that the HVDC flow is not dependent on phase angles

HVDC Risk

With two islands connected by an HVDC link, if the scheduled generation is in the opposite island to the load then the generation must be dispatched *and* a signal issued to the HVDC control system in order to send the generation from the sending island (with the generation) to the receiving island (with the load) via the HVDC link.

In the receiving island, the power from the HVDC link is a set amount resulting from an instruction issued to a control system. In this way the HVDC link is similar to a generator and can set the risk in the receiving island.

The constraint that identifies an HVDC branch as a risk is shown in Equation 21. This constraint acts in parallel with the generator risk constraint discussed in Tutorial 5: Risk and Reserve. Whichever is the larger of the AC generator risk and the HVDC branch risk will set the island risk (or if they represent the same risk quantity then they can both set the risk).

Equation 21: Island largest risk constraint for HVDC branch

$$BranchFlowToIsland_{HVDCBranch} - BranchLosses_{HVDCBranch} \leq LargestRiskHVDC_{Island}$$

To demonstrate the HVDC as a risk setter, increase gen01's energy offer price to $110, give it 250MW of $9 reserve and switch its Risk to off, so that it is not a risk unit. Edit gen00 so that it is also not a risk unit. Solve the model with Reserves enabled and Reserve Sharing disabled. The result is shown in Figure 89.

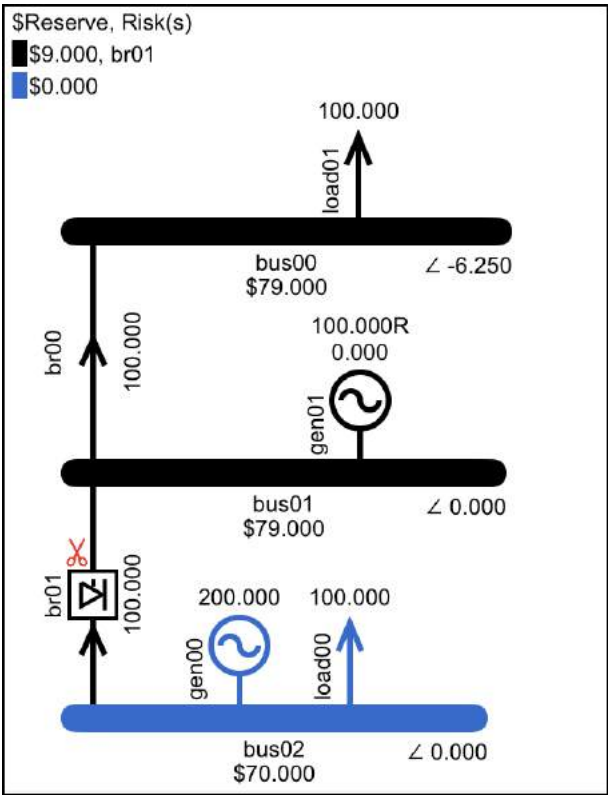


Figure 89: HVDC link as risk setter

Because the power at gen00 is cheaper than gen01 the solver uses the HVDC link to transport power from gen00 in island 2 (blue) to load01 in island 1 (black). The red risk setter symbol on the HVDC branch indicates that it is setting the risk in Island 1. The $9 cost of the reserve that covers the HVDC risk is incorporated into the energy price for Island 1.

Island 2 has no risk setter hence its reserve price is $0.

Note that it is the received transfer that is covered, i.e., flow – losses. Confirm this by solving with losses set to on, to get the result shown in Figure 90.

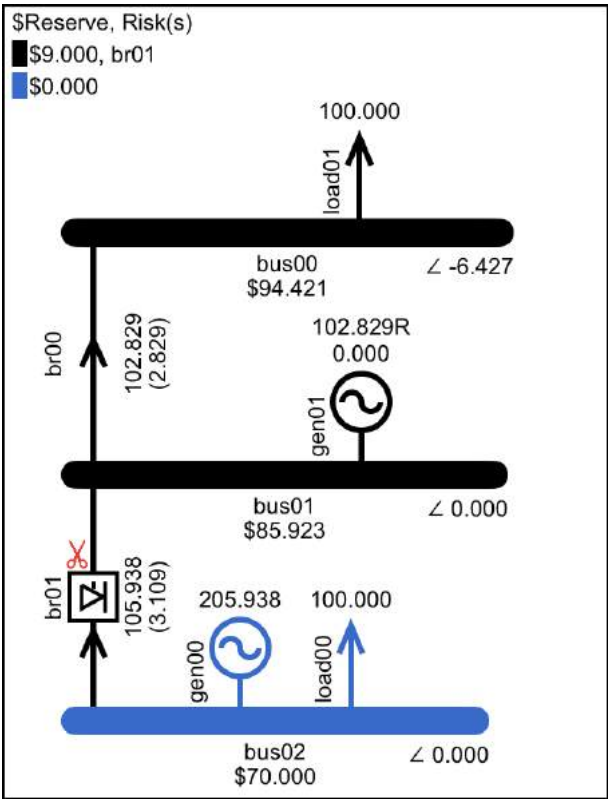


Figure 90: HVDC risk is net transfer

Viewing the HVDC risk constraints

The HVDC risk constraints that the solver creates for br01, with losses included, are shown in Figure 91. The risk of the flow minus losses is assigned to the LargestRiskDC variable for the island.

```
br01:
CalcRiskEachHVDC(LTE) constraint:
Shadow Price: $0.00
 +1.00000*br01_{BrFlowPos}
-1.00000*br01_brSeg00_{SegLossPos}
-1.00000*br01_brSeg01_{SegLossPos}
-1.00000*br01_brSeg02_{SegLossPos}
-1.00000*br01_{HVDCSubtractorToToIsland}
-1.00000*island02_{LargestRiskDC}  <= 0.00000

br01:
CalcRiskEachHVDC(LTE) constraint:
Shadow Price: $9.00
 +1.00000*br01_{BrFlowNeg}
-1.00000*br01_brSeg00_{SegLossNeg}
-1.00000*br01_brSeg01_{SegLossNeg}
-1.00000*br01_brSeg02_{SegLossNeg}
-1.00000*br01_{HVDCSubtractorToFromIsland}
-1.00000*island01_{LargestRiskDC}  <= 0.00000
```

Figure 91: HVDC risk calculation with losses included

Note that the constraints include an HVDC Subtractor, which is explained in the next sub-section.

The island constraints that cover the risk are shown in Figure 92. Note that the HVDC risk is covered separately from the AC risk… if the HVDC link is used to share reserve between islands, then the shared reserve cannot be used to cover the risk of

the HVDC link that transports it. Reserve sharing is covered later on in this tutorial.

ISLAND01

```
island01:
ReserveCoversACRisk(LTE) constraint:
Shadow Price: $0.00
 +1.00000*island01_{LargestRiskAC}
-1.00000*bus01_gen01_resOffer00_{Cleared}  <=
0.00000
```

```
island01:
ReserveCoversHVDCRisk(LTE) constraint:
Shadow Price: $9.00
 +1.00000*island01_{LargestRiskDC}
-1.00000*bus01_gen01_resOffer00_{Cleared}  <=
0.00000
```

ISLAND02

```
island02:
ReserveCoversACRisk(LTE) constraint:
Shadow Price: $0.00
 +1.00000*island02_{LargestRiskAC}  <= 0.00000
```

```
island02:
ReserveCoversHVDCRisk(LTE) constraint:
Shadow Price: $0.00
 +1.00000*island02_{LargestRiskDC}  <= 0.00000
```

Figure 92: Constraints that schedule reserve to calculate risk

HVDC Subtractor

When the HVDC link consists of more than one HVDC branch then the spare capacity on one of the

branches can offset the potential risk presented by the transfer on the other. To demonstrate this, add another HVDC branch between island 1 and island 2. Solve with Losses selected OFF, to produce the result shown in Figure 93.

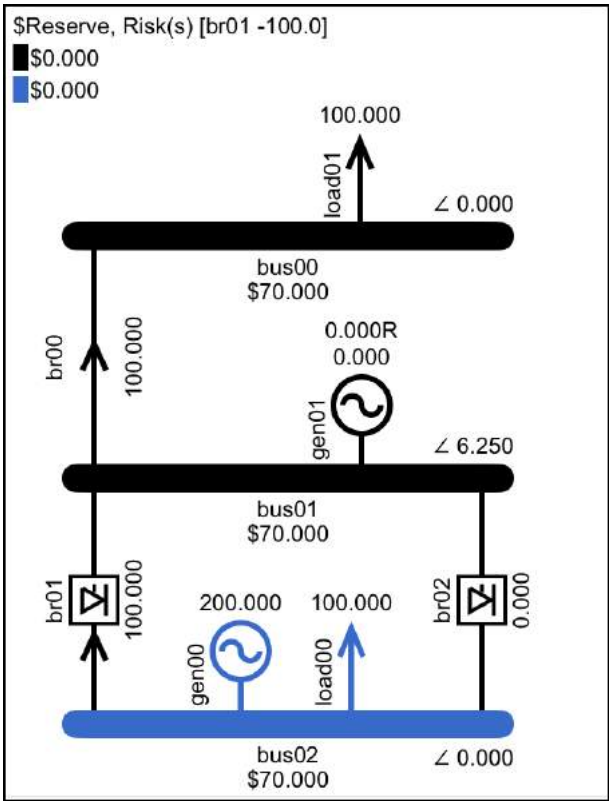


Figure 93: Parallel HVDC branch offsets HVDC risk

With the parallel HVDC branch, the HVDC risk in island 1 is now zero. This is because the transfer on

br01 is 100MW, while the spare capacity on br02 is at least 100MW. If br01 were to trip, then the HVDC control system would use br02 to replace the lost transfer.

If the solver uses a subtractor to reduce the HVDC risk then the reserve heading at the top of the display lists any HVDC branch with non-zero flow, followed by the subtractor that was applied to the risk presented by this flow. Here br01 is the potential risk setter and the solver has applied a subtractor of -100 (which it obtained from the spare capacity on br02).

Subtractor constraints

The subtractor constraints for br01 are shown in Figure 94.

```
br01:
CalcHVDCSubtractorToToIsland(LTE) constraint:
Shadow Price: $0.00
 +1.00000*br01_{HVDCSubtractorToToIsland}
+1.00000*br02_{BrFlowPos}   <= 300.00000

br01:
CalcHVDCSubtractorToFromIsland(LTE) constraint:
Shadow Price: $0.00
 +1.00000*br01_{HVDCSubtractorToFromIsland}
+1.00000*br02_{BrFlowNeg}   <= 300.00000
```

Figure 94: Subtractor calculation constraints

There are two constraints that calculate the branch's subtractor; one for forward flow and one for reverse flow (where forward flow is flow from the branch's from-island to the branch's to-island, and reverse flow is vice-versa).

The RHS of the branch's subtractor equation is the sum of the capacity of all *other* HVDC branches that have the same from-island and to-island as the branch in question. The LHS includes the subtractor variable, along with the flow from all other HVDC branches, effectively subtracting these flows from their capacity on the RHS.

HVDC Reserve Sharing

If the HVDC control system has the appropriate settings then the HVDC link can be used to share reserve between islands, i.e., reserve can be scheduled in one island to cover a generation risk in the other island.

In the event of a generator tripping, the island's frequency will drop. The HVDC control system will respond to the frequency drop by increasing the HVDC transfer to replace the lost generation. The increase in HVDC transfer will drop the frequency in the reserve-sending island, causing the reserve in the reserve-sending island to respond, effectively covering the increase in HVDC transfer.

The amount of reserve that can be shared must be covered by cleared reserve in the reserve-sending island, this is enforced by the constraint shown in Equation 22.

$$ReserveShared_{FromIsland} \leq \sum_{\substack{Gen \\ in\ FromIsland}} ClearedReserve_{Gen}$$

Equation 22: Reserve shared is covered by cleared reserve

To demonstrate reserve sharing, first simplify things by removing the HVDC branch br02. Then set gen00 as a risk unit. Solve this model with Reserves and Reserve Sharing enabled. The result is shown in Figure 95.

Other island risk

In the reserve sharing result shown in Figure 95, gen00 is the risk setter in island 2 (the blue island). The 200MW of reserve to cover this risk is provided by gen01 in island 1 and shared to island 2 via HVDC branch br01.

The risk setter for island 1 is listed as “other island”. The HVDC is no longer the risk setter in island 1, because the highest risk is now the risk presented by the reserve being shared to the other island.

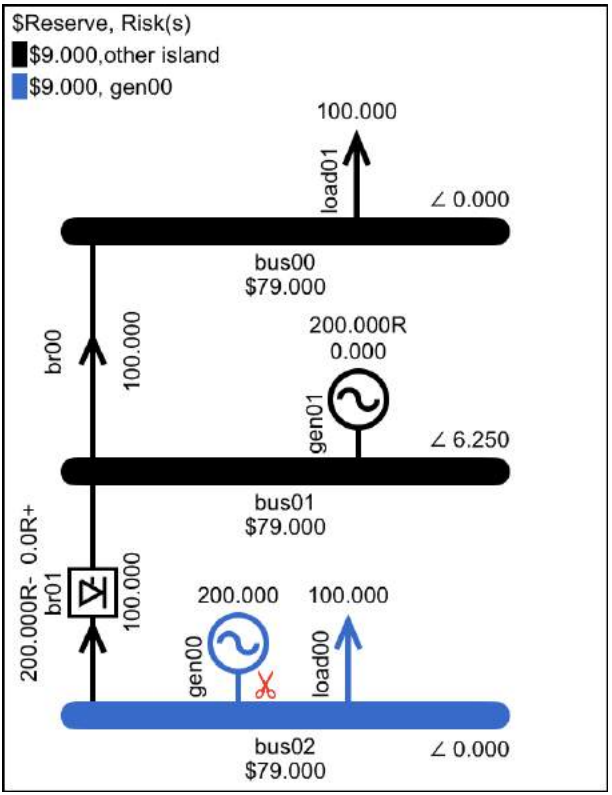


Figure 95: HVDC reserve sharing: reserve on gen01 in island 1 covers risk presented by gen00 in island 2

Reserve sharing direction

When reserve sharing is enabled, each HVDC branch that is sharing reserve has a label showing how much reserve it is sharing and in which direction. The reserve quantity has a "+" if it is shared in the

same direction as energy flow and a "-" if it is in the opposite direction.

If there is no energy flow then the "+" is for reserve that is shared in the direction that would be positive flow, i.e., from the branch's from-bus to the branch's to-bus (where the from-bus is the bus with the name that is alphabetically before the name of the to-bus… the from-bus and the to-bus are explicitly identified on the branch's Data Display).

Reserve Sharing "Reserve covers risk" constraint

When reserve sharing is enabled, the reserve from the other island is included in the "reserve covers risk constraint", as shown by Equation 23.

Equation 23: Reserve covers risk constraint, with reserve sharing

$$\sum_{\substack{Gen \\ in\ Island}} ClearedReserve_{Gen} + ReserveShared_{OtherIsland} \leq LargestACRisk_{Island}$$

The actual constraints for Island01 are shown in Figure 96. Note that, as discussed above, the reserve from the other island can only be used to cover the AC risk.

ISLAND01

```
island01:
ReserveCoversACRisk(LTE) constraint:
Shadow Price: $0.00
 +1.00000*island01_{LargestRiskAC}
-1.00000*bus01_gen01_resOffer00_{Cleared}
-1.00000*br01_{ResFromToIsland}  <= 0.00000

island01:
ReserveCoversHVDCRisk(LTE) constraint:
Shadow Price: $0.00
 +1.00000*island01_{LargestRiskDC}
-1.00000*bus01_gen01_resOffer00_{Cleared}  <=
0.00000

island01:
ResShareCappedByClearedRes(LTE) constraint:
Shadow Price: $9.00
 -1.00000*bus01_gen01_resOffer00_{Cleared}
+1.00000*br01_{ResFromFromIsland}  <= 0.00000
```

Figure 96: Reserve sharing constraints for Island01

HVDC setting risk and sharing reserve

Shared reserve received from the other island cannot be used to cover the risk presented by the HVDC. However, provided that the HVDC risk itself is covered by cleared reserve from within the island, the HVDC link can be used to share some or all of this cleared reserve to cover the AC risk in the other island.

To see the HVDC setting the risk in one island and also sharing reserve to the other island, add a new generator gen02 to bus02, edit its energy price to be

$120 and its reserve to be 30MW of reserve at $7. Edit load01 so that its quantity is only 10MW.

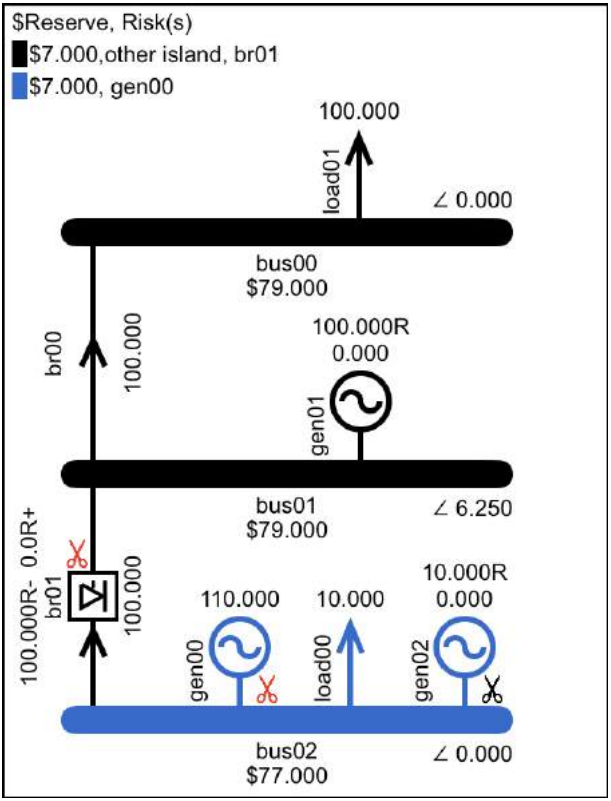


Figure 97: HVDC link as risk setter and sharing reserves

The result is shown in Figure 97. The HVDC risk of 100MW in island 1 is being covered by reserve of 100MW scheduled on gen01. This reserve is also being shared by the HVDC to cover 100MW of the risk presented by gen00 in island 2.

Reserve sharing in both directions

The reserve shared by the HVDC link can be shared in both directions at the same time. To demonstrate this, add a new generator gen03 to bus01, leaving its default parameters unchanged.

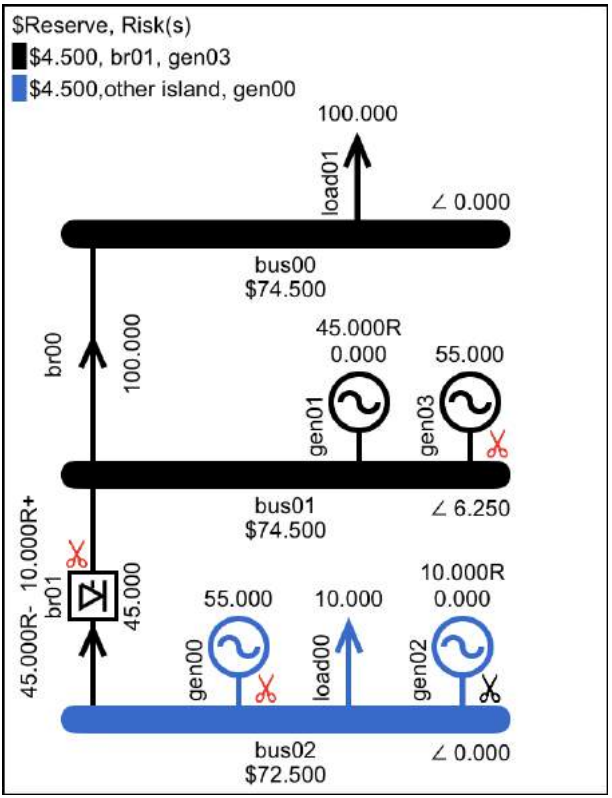


Figure 98: Reserve sharing in both directions

The results are shown in Figure 98. The risk in each island is covered partly by reserve from the other island.

HVDC capacity limit constraint

In the event of a tripping that results in the loss of generation, the reserve that is scheduled to cover the risk will respond to replace the missing energy. If reserve that is shared via the HVDC link is called upon then it will need to share the capacity of the link with the energy transfer that is already scheduled.

The requirement to ensure that the HVDC link is capable of providing the shared reserve as energy, while still maintaining its scheduled energy transfer, is represented by the constraint shown in Equation 24. This constraint limits the sum of the energy and reserve scheduled on the HVDC branch to be within the capacity limit of the branch, i.e., the branch's maximum flow.

Equation 24: Capacity limit on HVDC branch

$$Energy_{HVDCBranch} + Reserve_{HVDCBranch} \leq MaxFlow_{HVDCBranch}$$

The HVDC branch capacity constraint can be demonstrated by editing HVDC branch br01 to

reduce its max flow to 50MW. The result is shown in Figure 99.

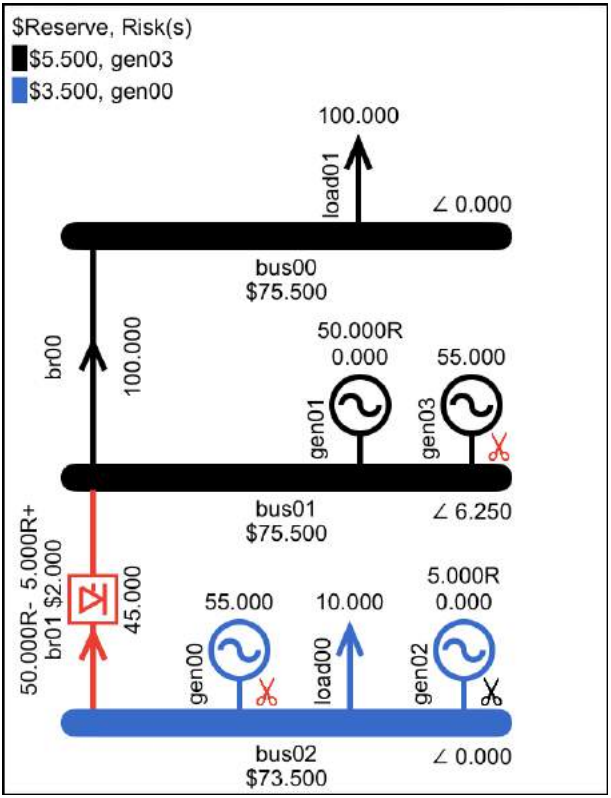


Figure 99: HVDC link binding on capacity limit of 50MW

Note that the HVDC capacity constraint takes account of direction. From island 2 to island 1 the HVDC link is scheduling energy and reserve, hence the energy of 45MW limits the scheduled reserve to 5MW. In the opposite direction, with no scheduled

energy transfer, the scheduled reserve is only limited by the 50MW capacity of the branch.

Reserve sharing price separation

In the result shown in Figure 99, the binding HVDC capacity constraint results in different reserve prices between the two islands. With the binding constraint separating the two islands, extra reserve in one island will not have a *direct* impact on reserve supply in the other island. However, because the capacity constraint is binding due to a combination of energy and reserve transfer, rather than being a hard constraint that restricts only the reserve, the inter-island reserve prices are not completely un-related, there relationship is dependent on the value of the energy trade-off that would need to be made in order to allow more reserve to be transferred.

Summary

In this tutorial we looked at HVDC links that connect AC power systems. We saw that the conductors in the HVDC link have the same physical properties as the branches in the AC system, and that it is the ability to control their flow via the HVDC control system that makes an HVDC link special in terms of how it is modelled.

We built models to investigate the special features of the HVDC link. Apart from the fact that the power flowing in the HVDC link is not directly related to the phase angles, most of the special features relate to risk and reserve. We saw how the HVDC link is modelled as a potential risk setter, and how this risk can be mitigated if the HVDC link has more than one HVDC branch. The HVDC link can also share reserve between islands, and we built models to demonstrate the various levels of complexity that arise when this feature is implemented.

Note that the HVDC investigation model is available pre-built in the Samples section of the app.

Tutorial 8: Actual Market Data

This tutorial looks at how to model a portion of the actual New Zealand electricity market, detailing where to locate the source files and how to extract the necessary modelling data. The results produced by the app are compared with those from the actual system.

We also use these sample models to further investigate transmission rentals, and to look at the impact that the ordering of the inputs has on the progress of the simplex algorithm.

Hawkes Bay Model

The portion of the New Zealand electricity market that will be modelled is the Hawkes Bay. The data is obtained from freely available online sources.

Source of real-world data

A diagram of the network model used by the New Zealand electricity market is available on the System Operator's website:

https://www.transpower.co.nz/system-operator/key-documents/maps-and-diagrams

The diagram is named the "Spd Locator drawing" so you can also find it using Google.

The input and output data for final pricing schedules from the New Zealand electricity market is available by going to the Electricity Authority (EA) website…

https://www.emi.ea.govt.nz/

…and then following the links: Wholesale Datasets, FinalPricing, CaseFiles.

Two sample models

The market data from the case files on the EA website has been used to build sample models that come pre-loaded with the app.

There are two versions of these samples. The larger of these more closely follows the actual market system, modelling the 110kV transmission system and the supply transformers that convert the voltage from 110kV to the distribution voltage of 33kV or 11kV. The load quantity is the value metered at the 33kV or 11kV buses. This model is referred to as the 033 model (because it models down to the 33kV/11kV level).

The smaller model uses the *scheduled* flow on the supply transformers to model the load at the 110kV level. This removes the need to model the supply transformers and makes the model faster to solve. It also allows us to account for the fixed losses of the

supply transformers, which is something that the actual market solver includes but the app does not. This is referred to as the 110 model.

Loading a sample model

Sample models are loaded via the Models display, which is accessed via the Folder button indicated in Figure 100.

Figure 100: Button that leads to the Models display

Once on the Models display the Samples display is accessed via the Samples button located on the lower toolbar, as shown in Figure 101.

Figure 101: The Samples button on the lower toolbar of the Models display

The Samples display is shown in Figure 102. The sample models of the Hawkes Bay are the 033 model and the 110 model as described above.

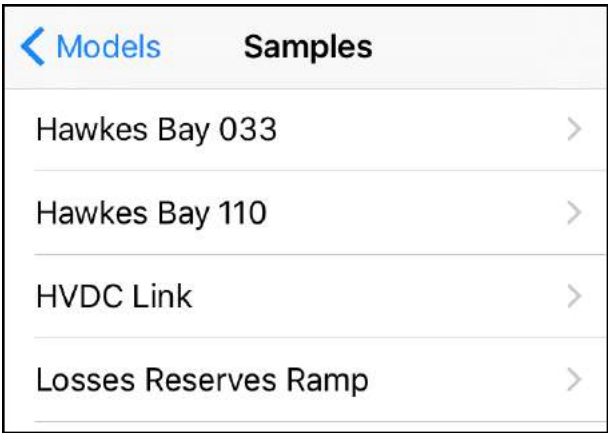


Figure 102: Samples display

View a screenshot of the 033 model by tapping its name, then set it as the current model by tapping the Load button on the toolbar. On an iPhone the model is larger than the display… you can zoom in and out using the pinch gesture. When the model is zoomed in you can move about (scroll) by dragging the background.

Real-world network model

The portion of the New Zealand electricity market that is modelled is shown in Figure 103. This is a section of the network model diagram from the System Operator's website, as described above.

The orange lines are 220kV, red lines are 110kV, green 33kV and black 11kV.

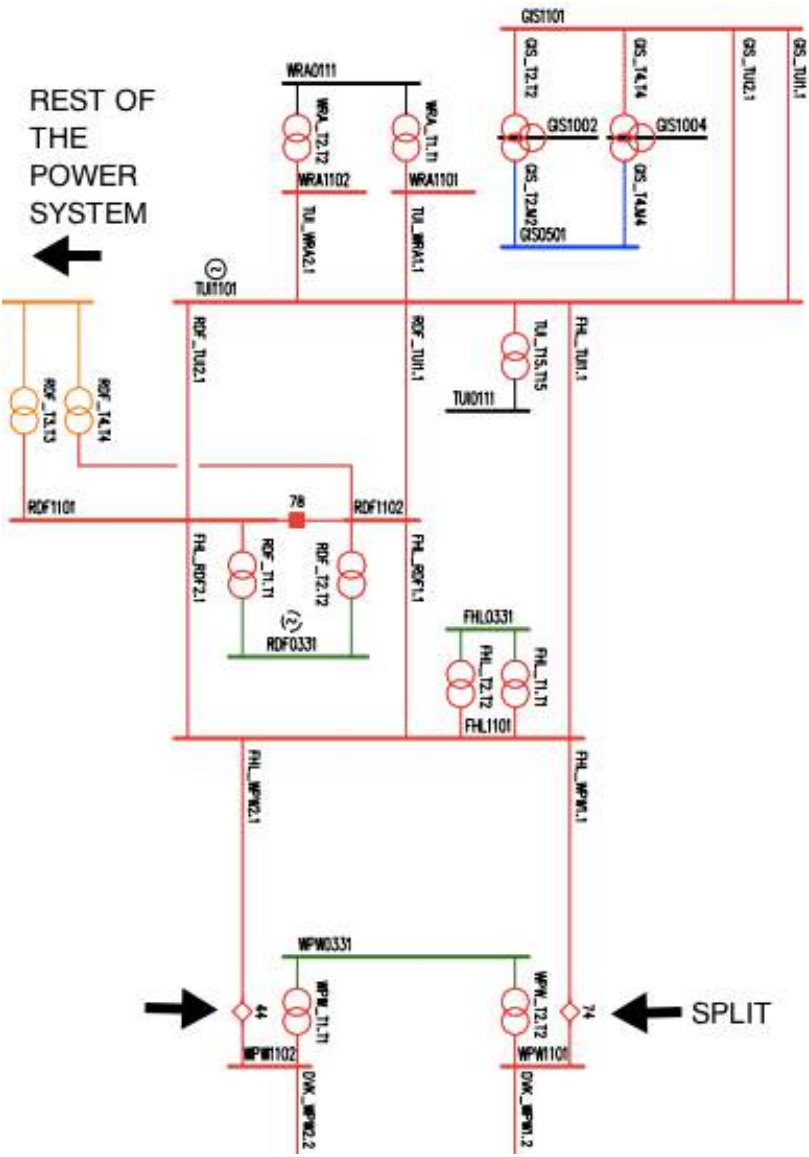


Figure 103: Hawkes Bay portion of the NZ electricity network

This portion of the electrical network covers the Hawkes Bay province of New Zealand. It connects to the rest of the power system via two interconnecting transformers, T3 and T4 at the Redcliffe (RDF) substation.

Modelling inflow from the rest of the system

In the sample models we use a dummy generator to model the inflow of power from the rest of the system via the T3 and T4 interconnecting transformers at RDF. The offer price and offer quantity of the dummy generator will be based on the real-world result at these interconnectors… the sum of the scheduled flow on T3 and T4 will be the offer quantity, while the offer price will be the bus price at the RDF 110kV bus.

There are transmission circuits to the south of the Hawkes Bay. These are capable of connecting to the rest of the electrical network, but they are not connected because there is an operational split in place to prevent overloads on the 110kV… the 110kV would effectively be in parallel with the 220kV if the split was closed.

Using real-world data

The sample models have been built using data from case files obtained from the Electricity Authority website, as described above.

The following sections describe how the case file data relates to the input data used by the app. There are some caveats. The first is that as of 01-April-15 the actual system moved from modelling branch losses using three flow-loss segments, i.e., three in each direction, to using six segments in each direction. The app is currently only set up to use three or four segments. Hence, while you can still use input data from after 01-April-15, there will be small differences in the results due to the differences in the number of segments.

The other caveat relates to the Hawkes Bay location. As of 01-April-15 (coincidentally the same date as the loss segments change) the substations at Gisborne (GIS) and Wairoa (WRA), which are both in the sample models, are no longer under the control of the System Operator, hence they have been removed from the electricity market model, and therefore you won't find their data in case files after this date.

Details of real-world data

The market data on the Electricity Authority website consists of zip files containing the 48 individual half hours (trading periods) that make up a trading day for the electricity market.

The date and time of the first period of the schedule are in the file name, e.g., for the file…

MSS_9111201501100714_0X

…the first period is 09-JAN-2015 11:00. Except, this is UTC time, so the local time is actually 10-JAN-2015 00:00. The schedule type is 111, which is a final pricing schedule. The 714 on the end is a random three-digit number.

Each zip file contains input files and result files for all of the trading periods covered by the case. For each trading period there is one results file and several input files. The data for the sample models is obtained from the input files shown in Table 4.

Table 4: Input file data to build a sample model

File extension	Section	Model data
PERIOD	BIDSANDOFFERS	ENOF = Energy offers
PERIOD	PNODEINT	MV90LOAD.

		Metered load at the supply bus. Used as load by the 033 model
MSSMOD	BRANCHLIMIT	Branch max flow
MSSNET	BRANCHBUS	Susceptance and Resistance

We also use data from the results file. As shown in Table 5 the results data is used to model the T3 and T4 interconnecting transformers as a dummy generator, and to provide the scheduled flow on the supply transformers, which we use as the load in the 110 model.

Table 5: Result file data used to build sample model

File extension	Section, Column	Model data
SPDSOLVED	BRANCH, TO_MW	Used as offer quantity for the dummy gen that represents the 220kV-110kV transformers T3 & T4
SPDSOLVED	BUS, PRICE	Used as offer price for the dummy gen that represents the 220kV-110kV

		transformers T3 & T4
SPDSOLVED	BRANCH, FROM_MW	Flow on the supply transformers, used as load by the 110 model

The above tables provide an overview of where the source data is located. The following sections show the details of how the data is extracted.

Generator offers

The energy offers for the actual generators (as opposed to the dummy generator that models T3 and T4) are from the BIDSANDOFFERS section of the PERIOD file, as shown in Figure 104.

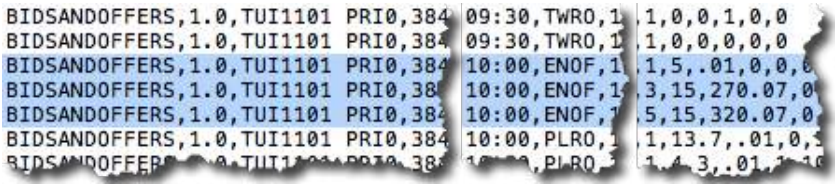


Figure 104: Energy offers from BIDSANDOFFERS section of PERIOD file, showing offers for generator PRI at the TUI bus

In-flow data

In our model the dummy generator named "T3&T4" at the RDF bus represents the in-flow of power from the rest of the power system to the Hawkes Bay area via interconnecting transformers T3 and T4.

The offer quantity for the dummy generator is the sum of the scheduled branch flows on the T3 and T4 branches. The offer price for the dummy generator is the scheduled bus price for the RDF 110kV bus. Note that the scheduled quantities are *results* from the actual system.

The scheduled branch flow is from the BRANCH section of the SPDSOLVED file as shown in Figure 105.

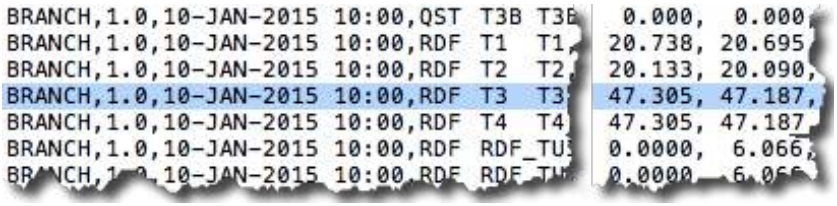


Figure 105: Scheduled branch flow from BRANCH section of SPDSOLVED file (used as offer quantity for dummy gen)

The scheduled bus price is from the BUS section of the SPDSOLVED file, as shown in Figure 106.

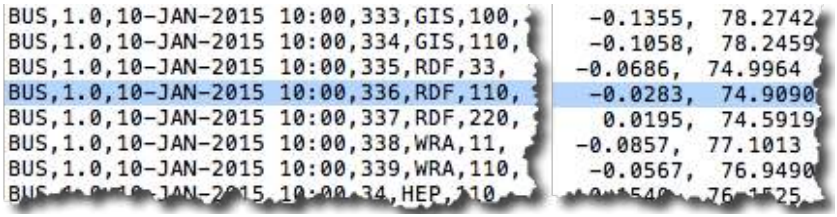


Figure 106: Scheduled bus price from BUS section of SPDSOLVED file (used as offer price for dummy gen)

Load data: from input data

For the 033 model the load data is from the MV90 column of the PNODEINT section of the PERIOD input file, as shown in Figure 107.

Figure 107: Load data from PNODEINT section of PERIOD file

Because we are obtaining our data from a Final Pricing schedule the load data represents metered load, i.e., load that was actually supplied. Hence, it is *required load* and therefore we will assign a very high bid price to ensure that it all clears.

Branch data

The branch limit data is from the BRANCHLIMIT section of the MSSMOD input file, as shown in Figure 108.

Figure 108: Branch limit from BRANCHLIMIT section of MSSMOD file

Branch susceptance and resistance are from the BRANCHBUS section of the period–specific MSSNET file, as shown in Figure 109.

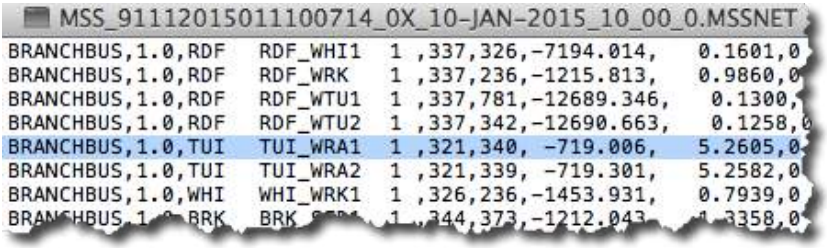


Figure 109: Branch susceptance and resistance from BRANCHBUS section of period-specific MSSNET file

There is one of these period–specific MSSNET files for each trading period in the schedule, not to be confused with the single MSSNET file that covers all trading periods (we don't use data from that file because it mostly contains info about the connectivity of the network model… we manually built our model based on the System Operator's network diagram, so we don't need the model from the MSSNET file).

Per cent per-unit

In the New Zealand electricity market the resistance and susceptance values in the case files are per-cent per-unit, i.e., 100 times the per-unit value. Because the app expects per-unit values, the resistance and susceptance values from the input files must be

divided by 100 to remove the per-cent factor before they are entered into the model.

Per-unit base

As discussed in the branch losses section, the app expects per-unit values that were calculated using a 100MVA base. The data in the input files was calculated using a 100MVA base, so we don't need to make any adjustments for that.

Result comparison

Figure 110 displays the result produced by the 033 sample model while Table 6 and Table 7 present the real-world results, extracted from the SPDSOLVED file.

Table 6: Bus results from SPDSOLVED file

ID_ST	ID_KV	LOAD	PRICE
FHL	33	32.995	75.7776
FHL	110	0	75.4302
GIS	50	25.791	78.3108
GIS	110	0	78.2459
GIS	11	0	78.2726
RDF	33	40.785	74.9964
RDF	110	0	74.909
TUI	11	0.291	76.8425
TUI	110	0	76.6103
WRA	11	5.966	77.1013
WRA	110	0	76.949

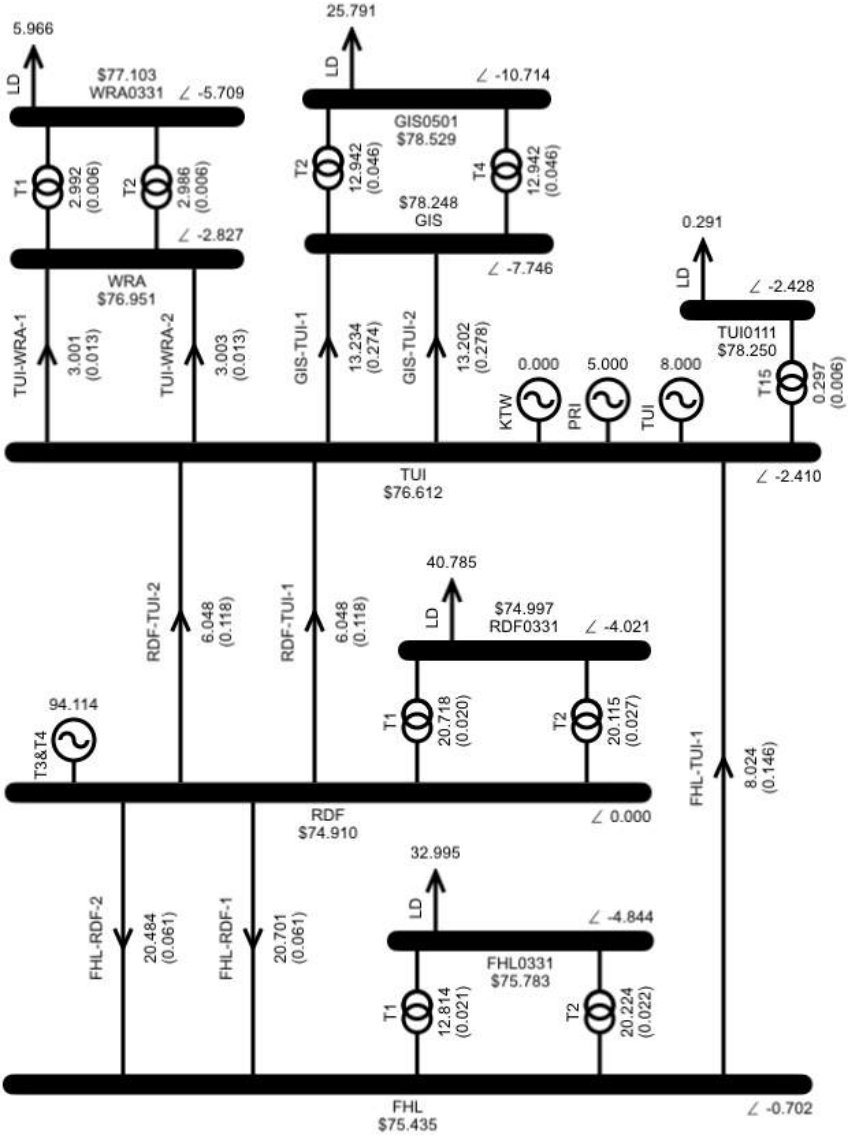


Figure 110: Result for Hawkes Bay 033 sample

Table 7: Branch results from SPDSOLVED file

BRANCHNAME	FIXED LOSS	FLOW	BRANCH LOSSES
FHL FHL_RDF1 1	0	-20.775	0.06202
FHL FHL_RDF2 1	0	-20.556	0.0609
FHL FHL_TUI1 1	0	8.056	0.14706
FHL T1 T1	0.0545	12.837	0.07543
FHL T2 T2	0.059	20.258	0.08147
GIS GIS_TUI1 1	0	-13.228	0.27438
GIS GIS_TUI2 1	0	-13.194	0.2779
GIS T2 T2	0.0286	12.92	0.03328
GIS T4 T4	0.0286	12.92	0.03328
RDF RDF_TUI1 1	0	6.066	0.11852
RDF RDF_TUI2 1	0	6.066	0.11852
RDF T1 T1	0.045	20.738	0.06541
RDF T2 T2	0.0326	20.133	0.05985
TUI T15 T15	0.0112	-0.297	0.0121
TUI TUI_WRA1 1	0	3.043	0.0134
TUI TUI_WRA2 1	0	3.037	0.01337
WRA T1 T1	0.0379	-3.01	0.0438
WRA T2 T2	0.0374	-3.005	0.04336

Differences between the app and the real-world results can be tracked down to the app not modelling fixed losses. Transformers have fixed losses, which are losses that occur when the transformer is energised, regardless of the flow. In the actual system the solver assigns the fixed losses 50-50 to the buses at each end of the transformer,

regardless of where the dynamic losses are assigned. The app does not model fixed losses.

Simplifying the model and improving the results

As mentioned in the introduction, we can simplify the model by using the scheduled in-flow of the supply transformers as the load value. This will remove the need to model the supply transformers and the supply bus. It will also allow us to account for the transformer fixed losses… because the actual system assigns fixed losses 50/50 to the buses at either end of the transformer, half of the fixed losses are already included in the scheduled flow on the transformer. We can model the other half of the fixed loss value by explicitly adding it to the 110kV load.

Load data: from scheduled results

The scheduled in-flow on the supply transformers is obtained from the BRANCH section of the SPDSOLVED file as shown in Figure 111. The fixed loss is also obtained from the branch results.

```
BRANCH,1.0,10-JAN-2015 10:00,FHL FHL_RDF1  -20.713, -20.775,
BRANCH,1.0,10-JAN-2015 10:00,FHL FHL_RDF2  -20.495, -20.556,
BRANCH,1.0,10-JAN-2015 10:00,FHL T1  T1,    12.837,  12.789,
BRANCH,1.0,10-JAN-2015 10:00,FHL T2  T2,    20.258,  20.206,
BRANCH,1.0,10-JAN-2015 10:00,FHL FHL_TUI1    8.056,   7.909,
BRANCH,1.0,10-JAN-2015 10:00,FHL FHL_WPW1    0.000,   0.000,
```

Figure 111: Load from BRANCH section of SPDSOLVED file

For example, to calculate the 110kV load value for FHL, we add the branch flows on FHL T1 and T2. This flow will include half of the fixed losses that the actual solver assigned to the supply bus, the other half of the fixed losses are included in our model by explicitly adding them to the load value that we calculate from the transformer branch flows.

The scheduled flow and fixed losses for FHL T1 and T2 are shown in Table 7. Hence the value we use for the 110kV load at FHL is the scheduled flow plus half the fixed losses:

$20.258 + 12.837 + (0.0545 + 0.059)/2 = 33.15175$

The results of the 110kV model are shown in Figure 112. These results agree with those from the actual system to within 2 decimal places.

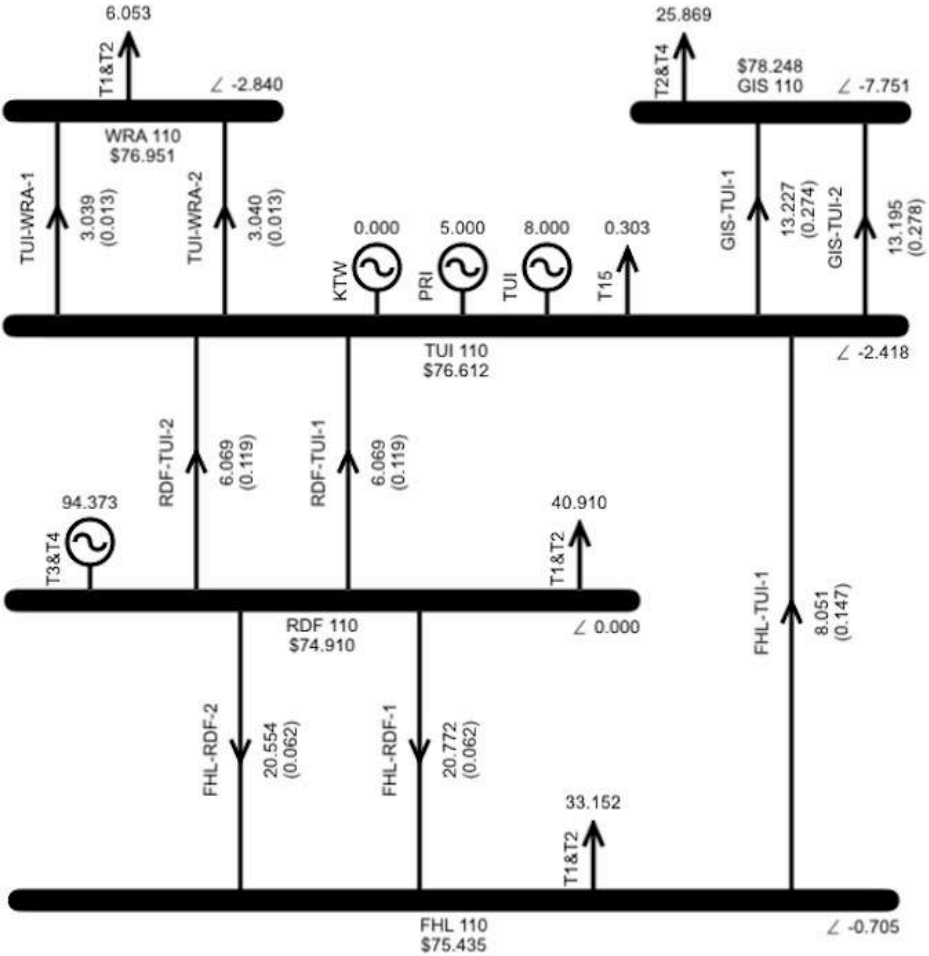


Figure 112: Result for Hawkes Bay 110 sample model

Further investigations

We can use the Hawkes Bay 110 model to conduct some further investigations as follows.

Changing the number of branch segments

Scroll through the loss results for the Hawkes Bay 110 result by using the forward and back arrows indicated in Figure 113. Notice that most of the scheduled branch flows are in the first flow-loss segment.

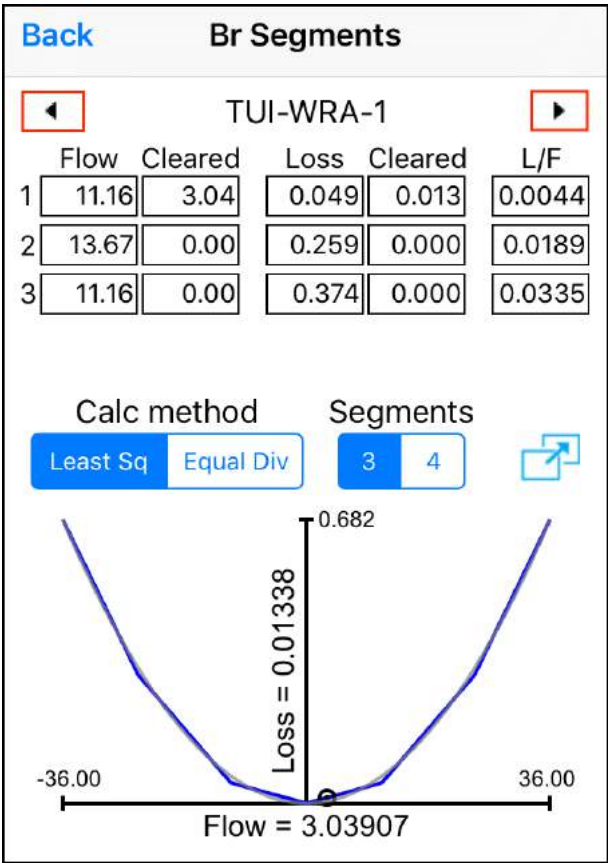


Figure 113: Use arrow buttons to scroll through loss results

As explained in Tutorial 4: Transmission Losses, it turns out that no loss rentals are incurred on branches where the scheduled flow is in the first flow-loss segment.

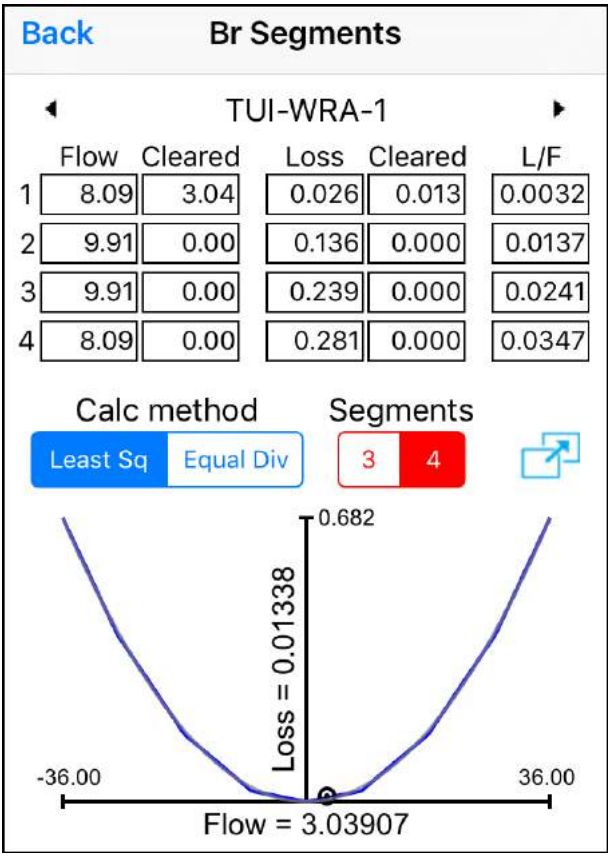


Figure 114: Changing number of segments from 3 to 4

If we increase the number of flow-loss segments from three to four then the flow on a number of the

branches will move into the second flow-loss segment, and we can see what happens.

To change the number of branch segments from 3 to 4, go to the Losses display for any branch and tap the button for 4 segments. The button colour will change to red to indicate that the value has changed, as shown in Figure 190.

With the number of segments changed, when you leave the display you will be alerted that the change will be applied to all branches… tap OK to apply the change.

After solving the model with four branch segments, the Results display is shown in Figure 115. Now that fewer branches have flow confined to the first flow-loss segment there has been an increase in the transmission rentals, i.e., the $Grid on the Results display, for the reasons discussed in Tutorial 4: Transmission Losses.

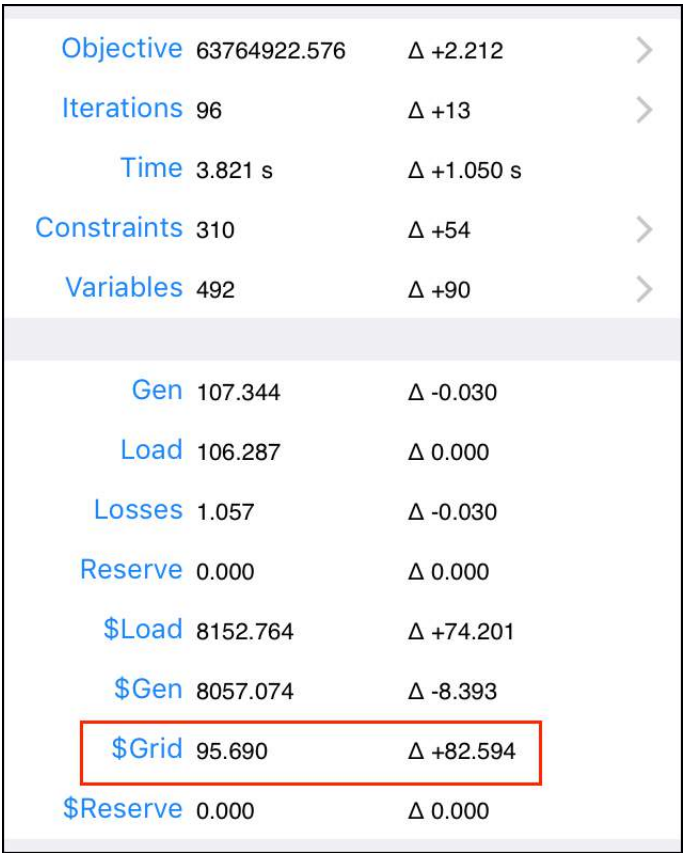


Figure 115: Result after increasing segments from 3 to 4

In contrast to the increase in transmission rentals, the losses have decreased slightly because the four-segment model has resulted in a lower loss estimate for the scheduled flows, due to the way that the segments have adjusted in the move from 3 to 4 segments.

Change the ordering of the constraints

Changing the order in which the components are supplied to the solver will change the order of the constraints in the initial tableau, and therefore change how the solve progresses.

Figure 116 indicates the Solve Setting that determines the sort order.

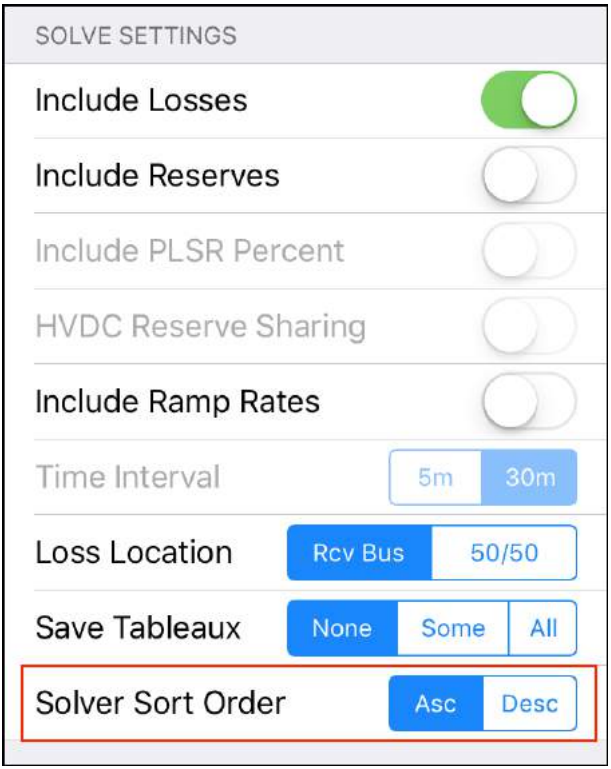


Figure 116: Changing the sort order of the constraints

Figure 117 and Figure 118 show the impact that different sort orders have on the progress of the simplex algorithm. These plots are accessed via Results – Iterations – Chart.

Note that although the path to the solution is different, the end result is the same. In Tutorial 9: Simplex Explained we will see why the sort order changes the path taken by the simplex algorithm.

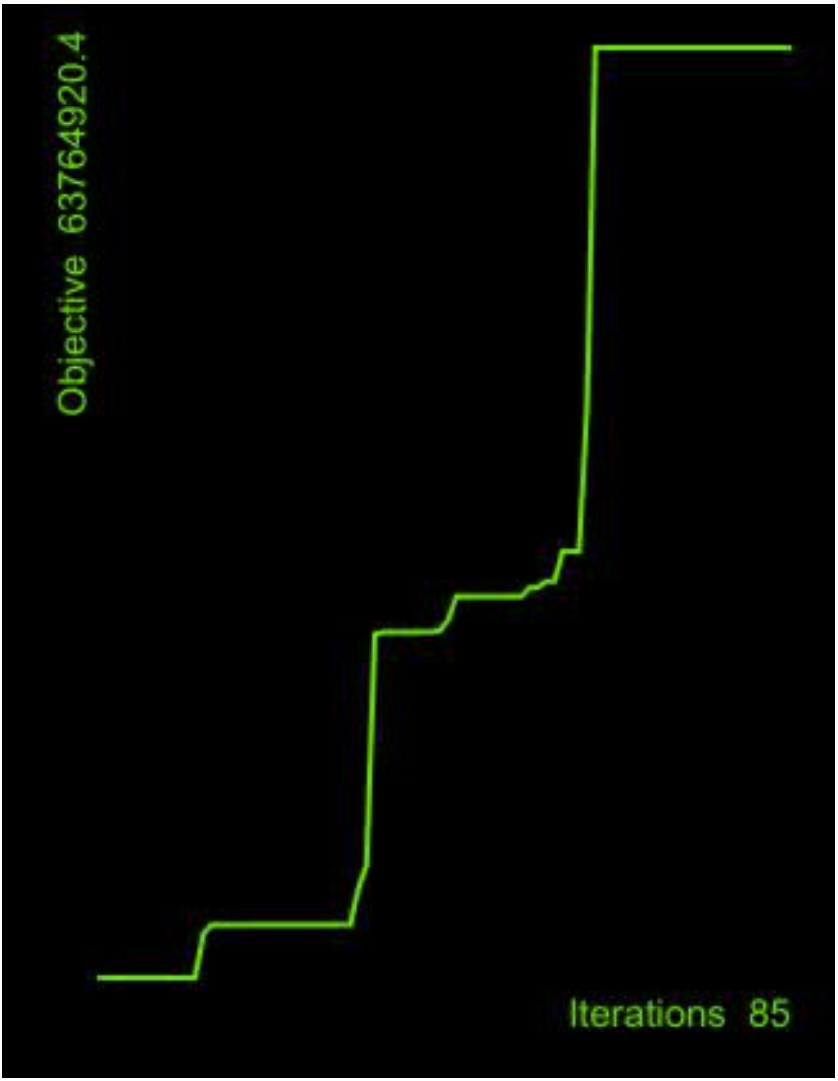


Figure 117: Objective value vs iteration count for the Hawkes Bay 110 model with ascending sort order

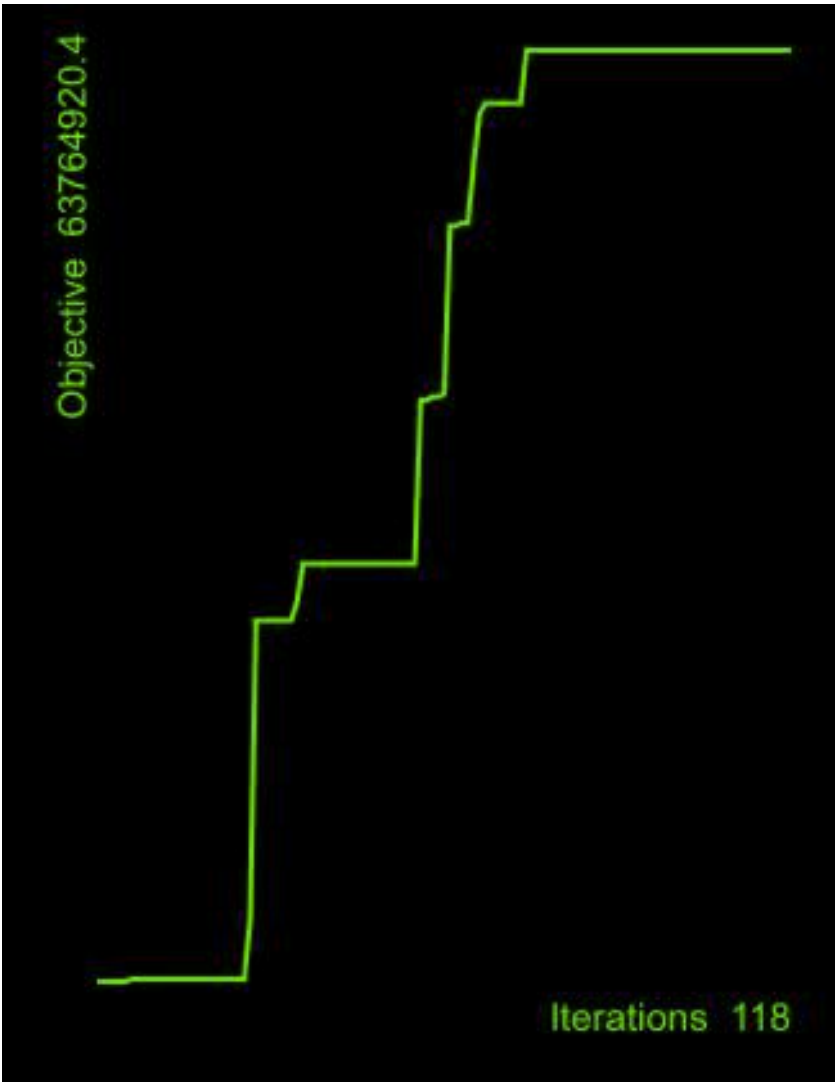


Figure 118: Objective value vs iteration count for the Hawkes Bay 110 model with descending sort order

Summary

In this tutorial we saw how to model a portion of the New Zealand electricity market by sourcing input data from the actual electricity market. The results produced by the app held up quite well when compared with the results from the actual market.

We used the sample model to investigate the impact that changing the number of branch segments has on transmission rentals, as explained in Tutorial 4: Transmission Losses.

We also investigated the impact that changing the sort order of the constraints has on the progress of the simplex algorithm. In the next section we will look at how the simplex algorithm solves the model, which will make it clear why the sort order has an impact.

Tutorial 9: Simplex Explained

"Does an 'explanation' make it any less impressive?"

— Ludwig Wittgenstein

This tutorial explains how the simplex algorithm solves the LP model, i.e., how the simplex algorithm takes us from the mathematical model of the electricity market to the resulting prices and quantities.

Demonstrating the simplex algorithm

Objective and Constraints

As discussed in Tutorial 1: Explaining Prices, the simplex algorithm solves a Linear Programming (LP) model, which is a mathematical problem specified in terms of:

- An *objective* equation
- *Constraint* equations that must be obeyed while achieving the objective.

The objective of the electricity market model is to maximize the objective value, which is calculated as the difference between the value of the load that is supplied, and the cost of the generation offers that must be cleared in order to supply the load.

The constraints ensure that the mathematical model behaves in the same way as the physical electricity network, e.g., there is a constraint that models how power flows along a transmission circuit.

Small electricity market model

The details of the simplex algorithm can be demonstrated by means of a small electricity market model consisting of a generator and a load connected via a bus, as shown in Figure 119. Build the model by tapping the Bus, Gen, Load buttons on the Build toolbar.

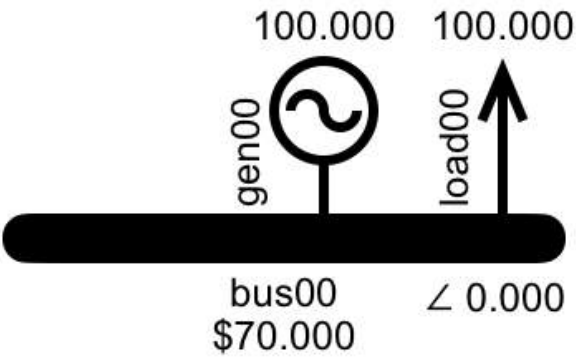


Figure 119: The small model

Constraints for the small model

The constraints that model the physical reality of the small model are listed in Table 8.

Table 8: Constraints in the small model

Component	Purpose of associated constraint
Bus00	Node balance constraint ensures that the quantity of electricity that enters the bus is equal to quantity that leaves
Gen00	Limit the cleared offer quantity to no more than the quantity that was offered
Load00	Limit the cleared bid quantity to no more than the quantity that was bid

The constraints listed in Table 8 are modelled by the equations shown in Equation 25, Equation 26 and Equation 27.

$$genOffer_{Cleared} - loadBid_{Cleared} = 0$$

Equation 25: Node balance equation

$$loadBid_{Cleared} \leq loadBid_{Max}$$

Equation 26: Max cleared bids

$$genOffer_{Cleared} \leq genOffer_{Max}$$

Equation 27: Max cleared offers

Objective function for the small model

The objective function for the small model is represented by Equation 28.

Maximize:

$$loadBid_{Cleared} \times loadBid_{Price}$$
$$-genOffer_{Cleared} \times genOffer_{Price}$$

Equation 28: Objective function

Simplex standard form

The simplex algorithm requires that the equations be expressed in a standard form:

- All equations expressed as $ax \leq b$ with $b \geq 0$, i.e., the RHS values are all ≥ 0
- Objective function expressed as a value to be maximized.

The electricity market model in standard form

Our objective function already has maximization as its requirement, which matches standard form.

All the constraints except for the node balance constraint meet the $ax \leq b$ requirement. The node balance constraint is an equality constraint, so we replace the = with a $\geq$ and a $\leq$.

The $\leq$ is already in standard form. The $\geq$ equation is not, so we multiply both sides by -1 to turn it into a $\leq$ equation. We can do this because the RHS is zero; hence after multiplying it by -1 the RHS is still zero and we still meet the $b \geq 0$ requirement.

After this adjustment the complete set of constraints for the model, expressed in the form required by the simplex algorithm, are as shown in Equation 29.

$$-loadBid_{Cleared} + genOffer_{Cleared} \leq 0$$

$$loadBid_{Cleared} - genOffer_{Cleared} \leq 0$$

$$loadBid_{Cleared} \leq loadBid_{Max}$$

$$genOffer_{Cleared} \leq genOffer_{Max}$$

Equation 29: Equation constraints for the small model, expressed in standard form

Viewing the constraints

To view the constraints for the small model, first you need to solve the model by tapping the Solve button, which takes you to the Solve Settings display, make sure all the options are selected OFF and then tap the Solve Now button.

When the solver has completed, tap the Results button, then tap the Constraints row, which takes you to the constraints display shown in Figure 120.

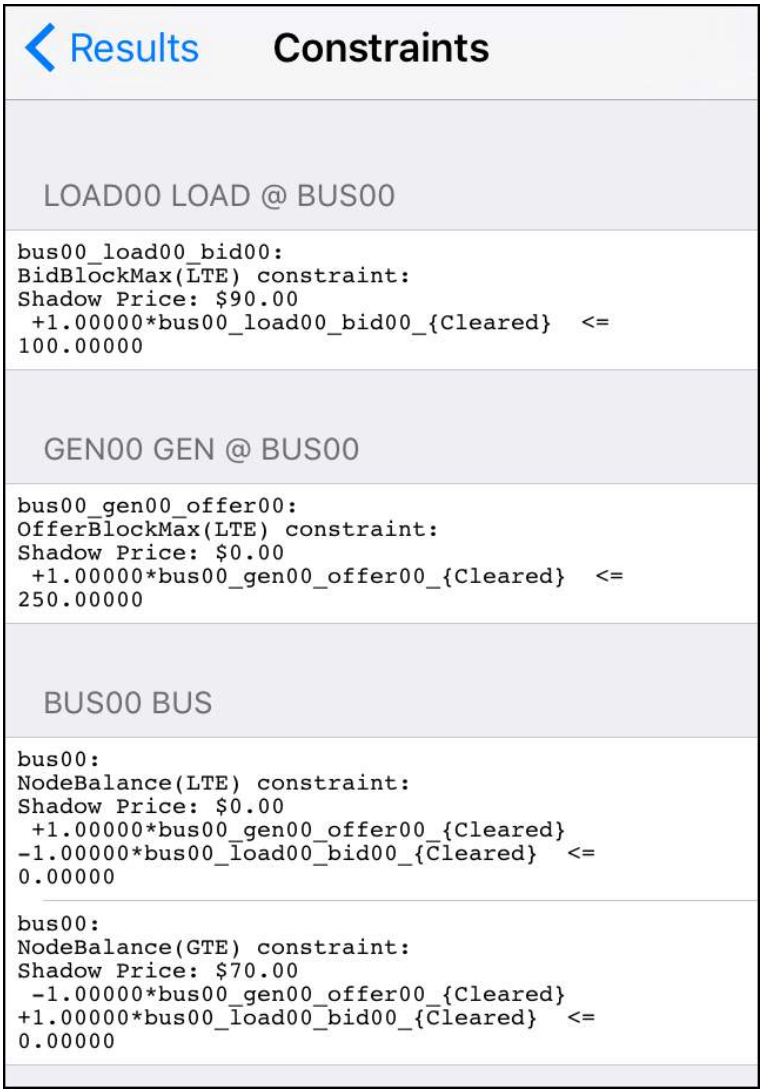


Figure 120: Constraint equations for the small model

Parameters and Variables

The elements of the constraint equations are *variables* and *parameters*.

Parameters are the fixed amounts:

$loadBid_{Max} = 100$

$genOffer_{Max} = 250$

Variables are the values that will be determined by the simplex algorithm:

$loadBid_{Cleared}$

$genOffer_{Cleared}$

Electricity market equations represented in tableau form

The simplex algorithm is easier to follow if the equations that form the mathematical model are represented as a tableau.

In the tableau the constraints are the rows, the variables are the columns, and the number at the intersection of the rows and columns represents the factor that the variable has in the constraint.

The following sections describe the actions that the simplex algorithm takes in order to maximize the objective value. You can track these actions via the tableaux.

The tableaux for the small model are shown in Figure 121 and Figure 122 (which are two halves of one spreadsheet, presented as two figures because it is too wide for this page).

###TABLEAU[0001]### col[1] will enter via row[2] and col[4] will leave			
Variables->	RHS	[1]bid00_{Cleared}	[2]offer00_{Cleared}
[1]_{NodeBalanceLTE} basic col[3]	0	-1	1
[2]_{NodeBalanceGTE} basic col[4]	0	1	-1
[3]bid00_{BidBlockMaxLTE} basic col[5]	100	1	0
[4]offer00_{OfferBlockMaxLTE} basic col[6]	250	0	1
[5]ObjectiveFn	0	-160	70
###TABLEAU[0002]### col[2] will enter via row[3] and col[5] will leave			
Variables->	RHS	[1]bid00_{Cleared}	[2]offer00_{Cleared}
[1]_{NodeBalanceLTE} basic col[3]	0	0	0
[2]_{NodeBalanceGTE} basic col[1]	0	1	-1
[3]bid00_{BidBlockMaxLTE} basic col[5]	100	0	1
[4]offer00_{OfferBlockMaxLTE} basic col[6]	250	0	1
[5]ObjectiveFn	0	0	-90
###TABLEAU[0003]### FINAL			
Variables->	RHS	[1]bid00_{Cleared}	[2]offer00_{Cleared}
[1]_{NodeBalanceLTE} basic col[3]	0	0	0
[2]_{NodeBalanceGTE} basic col[1]	100	1	0
[3]bid00_{BidBlockMaxLTE} basic col[2]	100	0	1
[4]offer00_{OfferBlockMaxLTE} basic col[6]	150	0	0
[5]ObjectiveFn	9000	0	0

Figure 121: Left side of the tableaux

[3]slack{NodeBalanceLTE}	[4]slack{NodeBalanceGTE}	[5]slack_bid00_{BidBlockMaxLTE}	[6]slack_offer00_{OfferBlockMaxLTE}
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
0	0	0	0
[3]slack{NodeBalanceLTE}	[4]slack{NodeBalanceGTE}	[5]slack_bid00_{BidBlockMaxLTE}	[6]slack_offer00_{OfferBlockMaxLTE}
1	1	0	0
0	1	0	0
0	-1	1	0
0	0	0	1
0	160	0	0
[3]slack{NodeBalanceLTE}	[4]slack{NodeBalanceGTE}	[5]slack_bid00_{BidBlockMaxLTE}	[6]slack_offer00_{OfferBlockMaxLTE}
1	1	0	0
0	0	1	0
0	-1	1	0
0	1	-1	1
0	70	90	0

Figure 122: Right side of the tableaux

Tracking the simplex solve

The app allows you to track the actions of the simplex algorithm by saving the tableaux that the simplex algorithm uses. After the simplex algorithm has finished solving, the tableaux can be exported to a csv file, e.g., as shown in Figure 121 and Figure 122.

We are going to export the tableaux for the small model. Before we click the Solve Now button the app needs to know that we want to save the tableaus. The tableaux are not saved by default, because for larger models the size of the tableaux grows quite quickly and writing them out will slow

down the time that it takes to solve the model (or if they are very large, the app will run out of memory).

Choosing to save tableaux

Tapping the Solve button takes you to the Solve Options display, which includes the "Save Tableaux" option shown in Figure 123. The "Some" option will save the first, second and last tableaux. Select the "All" option, then solve.

Figure 123: "Save Tableaux" on the Solve Settings display

Accessing the saved tableaux

To view the saved tableaux, after the solve has completed you will need to export them as a csv file via email. Go to the Results display and then tap the "Import Export" button, indicated in Figure 124.

Figure 124: The Import Export button

The "Import Export" icon leads to the "Import Export" display shown in Figure 125. To export the tableaux, tap the "Email Results" button.

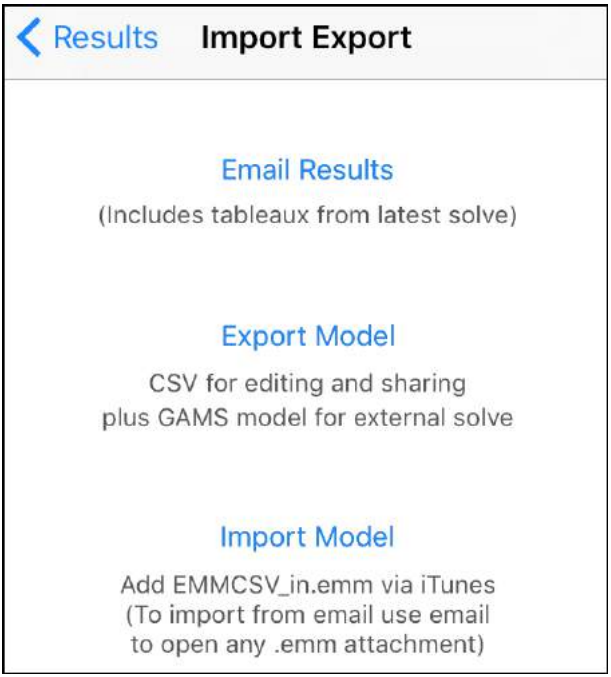


Figure 125: The Import Export display… use Email Results to export the tableaux

The "Email Results" email will contain several attachments… a screenshot of the model, a csv file containing the results of the latest solve and (assuming that there are saved tableaux) a csv file containing the saved tableaux from the solve. You can open the tableaux csv in a spreadsheet.

Layout of the tableau

The heading for each tableau describes the actions that the algorithm will apply to that tableau in order

to improve the objective value and thereby arrive at the next tableau, e.g., for Tableau00 the heading is "col[1] will enter via row[2] and col[4] will leave". What this means is explained below, as we track the progress of the solve.

Each variable has its own column in the tableau. Each constraint has its own row. The label for each row includes the name of the constraint, the row number, and the column number of the *basic variable* (explained below) for that constraint.

Initial Tableau

The simplex algorithm works to maximise the objective value while meeting the requirements of the constraints. To do this the simplex algorithm converts the equations, which are supplied to the algorithm in the standard form described above, into a set of *equality* constraints that represent a feasible solution. These are the equations represented by the tableau.

Slack variables

Each tableau represents a valid solution that meets the requirements of the constraints. To create the initial tableau the first thing that the simplex algorithm does is to convert the <= constraints to equality constraints (constraints that have an =

sign) by adding slack variables. Starting with the inequality constraint, e.g.,

$$-loadBid_{Cleared} + genOffer_{Cleared} \leq 0$$

…the simplex algorithm adds a slack variable, s_1. The slack variable can take any non-negative value and this allows the inequality $\leq$ in the original equation to be replaced with an equality, to create a new constraint:

$$-loadBid_{Cleared} + genOffer_{Cleared} + s_1 = 0$$

This meets the requirements of the original constraint, provided that the RHS is non-negative and all variables are ≥ 0. These requirements will be enforced by the rules of the algorithm.

Adding slack variables

Adding slack variables produces an equivalent set of equality constraints:

$$-loadBid_{Cleared} + genOffer_{Cleared} + s_1 = 0$$

$$loadBid_{Cleared} - genOffer_{Cleared} + s_2 = 0$$

$$loadBid_{Cleared} + s_3 = 100$$

$$genOffer_{Cleared} + s_4 = 250$$

Basic Feasible Solution (BFS)

From the equations shown above, the simplex algorithm creates the initial feasible solution by

setting all the slack variables to the RHS value and all other variables to zero:

$$s_1 = 0$$

$$s_2 = 0$$

$$\text{loadBid}_{\text{Cleared}} = 0$$

$$s_3 = 100$$

$$genOffer_{Cleared} = 0$$

$$s_4 = 250$$

This is represented by the initial tableau from the csv, i.e., Tableau01, reproduced in Table 9.

Table 9: Constraints in the initial tableau

Basic	$loadBid_{Clrd}$ [1]	$genOffer_{Clrd}$ [2]	s_1 [3]	s_2 [4]	s_3 [5]	s_4 [6]	RHS
[3]	-1	1	1				0
[4]	1	-1		1			0
[5]	1				1		100
[6]		1				1	250

Basic variables

In this first Basic Feasible Solution (BFS) all the variables have their *value* set to zero except for the slack variables. The solution is feasible because

each constraint only has one non-zero variable, which is the slack variable.

The variables that are set to zero are referred to as the non-basic variables. The remaining variables, currently represented by the slack variables, constitute the basis of the solution and are referred to as the basic variables.

One basic variable per constraint

The basic variables are the variables that can be non-zero. Each constraint has its own basic variable, this basic variable has a factor of 1.0 in the constraint where it is the basic variable, and a factor of zero in all other constraints.

In the tableaux file, each row records the column that contains the basic variable for that row's constraint, e.g., for row [1] the basic column is column [3].

Note that while the non-basic variables are assigned a *value* of zero, they can still have a non-zero *factor*. For example, the LoadBidCleared variable has non-zero factors in three of the equations, but it is non-basic so the LoadBidCleared variable has a *value* of zero by definition of it being non-basic.

The objective function including slack variables

While this initial solution is feasible, the objective value as calculated by Equation 30 is zero because it was defined in terms of the original variables, which are currently all non-basic and therefore have values of zero. The slack variables have been added to the objective value, with factors of zero.

Equation 30: The objective, incorporating the slack variables

Maximize:

$$150 \times loadBid_{Cleared}$$
$$-100 \times genOffer_{Cleared}$$
$$+ 0 \times s_1 + 0 \times S_2 + 0 \times s_3 + 0 \times S_4$$

Improving the objective value

Starting with the initial Basic Feasible Solution (BFS), the simplex algorithm moves to another BFS that improves the objective value. Improving the objective value is achieved by identifying the non-basic variable whose increase would most improve the objective value, and then making it a basic variable, by exchanging it with one of the existing basic variables.

Keeping track of the objective value

To keep track of the objective value, the objective function is re-written as a constraint. Using z to represent the objective value, the objective function as a constraint is shown in Equation 31.

$$z - loadBid_{Cleared} \times loadBid_{Price} + genOffer_{Cleared} \times genOffer_{Price} = 0$$

Equation 31: Writing the objective function as a constraint with the objective value as variable z

Table 10 shows the initial tableau with the objective function added as a constraint.

Table 10: Constraints and objective as a tableau

z	$loadBid_{Clrd}$	$genOffer_{Clrd}$	s_1	s_2	s_3	s_4	RHS
	-1	1				1	0
	1	-1			1		0
	1			1			100
		1	1				250
1	-160	70					0

The algorithm progresses by making non-basic variables basic to improve the objective, while obeying the rule that there is one basic variable per constraint and the basic variable has a factor of zero in all other rows. This is achieved by a series of row

operations. This will maintain the relationship between the variables as enforced by the original set of constraints. The row operations will also update the objective.

It turns out that we don't need to display the z column, because other than z there are no other values in this column so row operations won't change the z factor, and we won't be scaling the objective row, hence the factor of z will always be one.

The initial tableau can now be presented as shown in Table 11, which lines up with the csv export shown in Figure 121 and Figure 122.

Table 11: The initial tableau, incorporating the objective

Basic	$loadBid_{Clrd}$ [1]	$genOffer_{Clrd}$ [2]	s_1 [3]	s_2 [4]	s_3 [5]	s_4 [6]	RHS
[3]	-1	1	1				0
[4]	1	-1		1			0
[5]	1				1		100
[6]		1				1	250
	-160	70					0

The basic variables will also have factors of zero in the objective function constraint. As the other variables in the objective constraint are non-basic

and have *values* of zero, the only variable that is not set to zero and has a non-zero factor is the objective value z, which is therefore equal to the RHS.

Currently the RHS of the objective row is zero and therefore the objective value is zero. The algorithm will now work to improve the objective value.

Reduced costs

The only variables in the objective function constraint that have non-zero *factors* are the non-basic variables.

The non-basic variables have *values* of zero. If one of these variables became non-zero then the objective value z, (hidden off to the left with its factor always one) would need to change, in order to keep the constraint balanced.

For example, if the $loadBid_{Clrd}$ variable, which has a factor of -150 in the objective function constraint, were to take on a value of 1.0 then the objective value RHS (and therefore z) would need to take on a value of 150 in order to keep the objective constraint balanced.

Likewise, if the $genOffer_{Clrd}$ variable with a factor of 70 were to take on a value of 1.0 then the objective value would need to be -70.

Hence, the *factors* in the objective function constraint indicate which variable is the best to increase in order to improve the objective value. These factors are referred to as the *reduced costs*.

In this case only the $loadBid_{Clrd}$ variable can improve the objective value. Because it has a reduced cost of -150, each unit of increase by the $loadBid_{Clrd}$ variable improves the objective value by 150.

The entering variable and the leaving variable

To increase the objective value, the simplex algorithm chooses the variable with the most negative reduced cost, in this case $loadBid_{Clrd}$, and then increases the value of that variable as much as possible. Because the only variables with non-zero reduced costs are the non-basic variables (assigned a value of zero), in order for the value of this variable to be increased it needs to become a basic variable.

However, the tableau is feasible by virtue of the fact that there is only one basic variable for each constraint, so in order for the non-basic variable to become basic, i.e., to *enter the basis*, one of the basic variables will need to become non-basic, i.e., *leave the basis*.

For the non-basic variable to enter, the tableau will need to be manipulated so that canonical form is maintained, i.e., so that the newly basic variable has a factor of 1.0 in the entering constraint and zero in all other constraints, including the objective function constraint.

Determining the entering constraint

The entering constraint is selected by finding the row that allows the factor of the entering variable to be set to 1.0 in that row and zero in all other rows.

What drives this selection is not what happens in the entering row, because there it is a simple division to obtain a factor of one, but rather what happens to the other rows in order to set the entering variable's factor to zero in those rows.

The first step to making the entering variable basic is to divide the entering row by the factor of the entering variable in that row, so that its factor becomes 1.0. Then the row operations will subtract a multiple of the entering row from the other rows so that the entering variable's factor becomes zero in those rows.

Row operations example on a dummy tableau

To demonstrate how and why the algorithm determines the entering row, we will use the

dummy tableau shown in Table 12. We are using a dummy tableau because the numbers in our actual model don't lend themselves to explaining the issue.

The entering row is dictated by the requirement that variables must be non-negative (remember that this requirement arose when the original inequality constraints were replaced with slack variables and equality constraints).

The algorithm cannot select an entering row that would cause any of the basic variables to be assigned a negative value (the *non*-basic variables have values of zero regardless of the equations, so we don't have to worry about them).

Table 12: Dummy tableau to demonstrate entering and leaving variables

	Col#1	2	3	4	5	RHS
	var1	*var2*	*var3*	*var4*	*var5*	
Row#1	-7	1				1
2	5		1			20
3	10			1		1
4	40		1		1	8
5	-150	70				0

In the dummy example the entering variable is in column one (because it has the most negative reduced cost). When selecting the entering row we obviously can't use the row with the -7 factor because to make the factor 1 would result in a RHS of -1/7, and then the value of the basic variable would be -1/7, which cannot happen because all variables must be >= 0.

If row 2 was selected, then in order for the entering variable to take on a factor of 1.0 in that row, the row would be divided by 5.0, and the RHS becomes 4.0. Row 2 and 3 would then be as follows:

	Col#1	2	3	4	5	RHS
Row#2	1		1/5			4
Row#3	10			1		1

Row 2 is OK, but now to give the entering variable a factor of zero in row 3 we need to subtract 10 x row 2 from row 3. This will result in a RHS value of -39 for row 3 and then the basic variable for row 3 would become negative… so we can't do this.

Therefore, in order to prevent a negative RHS for any of the other rows, the entering row must be the row with the smallest positive ratio of RHS to entering variable factor… so that when any other row subtracts the entering row (in order to zero the

entering variable in that row), the RHS of that row will remain positive.

In our dummy example, the row with the smallest positive ratio of RHS to entering factor is row 3, where the ration is 1/10. After the entering variable has entered via row 3, the tableau is shown in Table 13.

Table 13: Dummy tableau after pivot

	Col#1	2	3	4	5	RHS
Row#1	0	1		0.7		1.7
#2	0		1	-0.5		19.5
#3	1			0.1		0.1
#4	0		1	-4	1	4
Obj	0	70		15		15

The new tableau after the pivot

Back to our small model, the column with the smallest ratio of RHS to entering factor is row 2, with a ratio of zero. This is the same as the ratio for row 1… for consistency if more than one row has the same ratio then we always use the last row that we find, i.e., row 2 in this case (it does not matter how we make the decision as long as we are consistent).

After row operations have been performed to allow $loadBid_{Clrd}$ to become a basic variable, the tableau appears as shown in Table 15.

The entering variable is $loadBid_{Clrd}$. The leaving variable is the variable that was previously basic in the entering row. In this case the leaving variable is the slack variable *s2*, which becomes a non -basic variable.

Now that $loadBid_{Clrd}$ is a basic variable it is no longer confined to a value of zero. However, the equation where $loadBid_{Clrd}$is basic has a RHS of zero, therefore the $loadBid_{Clrd}$ variable is assigned a value of zero by the equation.

We can also see from the RHS of the objective constraint that the objective value is still zero.

Table 14: Initial tableau (again)

Basic	$loadBid_{Clrd}$ [1]	$genOffer_{Clrd}$ [2]	s_1 [3]	s_2 [4]	s_3 [5]	s_4 [6]	RHS
[3]	-1	1	1				0
[4]	1	-1		1			0
[5]	1				1		100
[6]		1				1	250
	-160	70					0

Table 15: Second tableau

Basic	$loadBid_{Clrd}$ [1]	$genOffer_{Clrd}$ [2]	s_1 [3]	s_2 [4]	s_3 [5]	s_4 [6]	RHS
[3]			1	1			0
[1]	1	-1		1			0
[5]		1		-1	1		100
[6]		1				1	250
		-90		160			0

Updating the objective value

Bringing the entering variable into the basis included a row operation to set its reduced cost to zero. The reduced cost of zero indicates that the variable has been increased as much as possible and cannot improve the objective value any further. The row update that set the reduced cost to zero also updated the objective value.

The reduced cost of the entering variable was -150, hence to zero the reduced cost, the row operation added 150 times the entering row to the objective row. This added 150 times the RHS of the entering row to the RHS of the objective row. Because the RHS of the entering row *is* the value of the entering variable, this has the effect of adding the entering

variable to the objective value… which was the intent of the iteration in the first place.

Adjusting the reduced costs

The first step of the row operations was to scale the entering row so that the entering variable has a factor of one. This also scaled all other factors in that row.

Apart from the entering variable, the only other variables with non-zero factors in the entering row are non-basic variables. After scaling, the entering variable has a factor of one and the factor of each non-basic variable in the row now indicates how much a change in their value would force the entering variable to change (if they were to become non-zero).

After we scaled the entering row to make the entering factor one, we added that row, multiplied by the reduced cost of the entering variable, to the objective function, this was to zero the reduced cost of the entering variable. This updated the reduced cost of all the non-basic variables based on how much freedom they can provide the entering variable, and the value of the entering variable.

Looking at tableau 2 in Figure 126 and Figure 127 (the tableaux csv again) we can see that the update

of the objective function has changed the reduced cost of the cleared offer from positive to negative, because its potential to allow the cleared bid to increase has overcome the cost of the offer.

###TABLEAU[0001]### col[1] will enter via row[2] and col[4] will leave			
Variables->	RHS	[1]bid00_{Cleared}	[2]offer00_{Cleared}
[1]_{NodeBalanceLTE} basic col[3]	0	-1	1
[2]_{NodeBalanceGTE} basic col[4]	0	1	-1
[3]bid00_{BidBlockMaxLTE} basic col[5]	100	1	0
[4]offer00_{OfferBlockMaxLTE} basic col[6]	250	0	1
[5]ObjectiveFn	0	-160	70
###TABLEAU[0002]### col[2] will enter via row[3] and col[5] will leave			
Variables->	RHS	[1]bid00_{Cleared}	[2]offer00_{Cleared}
[1]_{NodeBalanceLTE} basic col[3]	0	0	0
[2]_{NodeBalanceGTE} basic col[1]	0	1	-1
[3]bid00_{BidBlockMaxLTE} basic col[5]	100	0	1
[4]offer00_{OfferBlockMaxLTE} basic col[6]	250	0	1
[5]ObjectiveFn	0	0	-90
###TABLEAU[0003]### FINAL			
Variables->	RHS	[1]bid00_{Cleared}	[2]offer00_{Cleared}
[1]_{NodeBalanceLTE} basic col[3]	0	0	0
[2]_{NodeBalanceGTE} basic col[1]	100	1	0
[3]bid00_{BidBlockMaxLTE} basic col[2]	100	0	1
[4]offer00_{OfferBlockMaxLTE} basic col[6]	150	0	0
[5]ObjectiveFn	9000	0	0

Figure 126: Left side of the tableaux csv

[3]slack{NodeBalanceLTE}	[4]slack{NodeBalanceGTE}	[5]slack_bid00_{BidBlockMaxLTE}	[6]slack_offer00_{OfferBlockMaxLTE}
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
0	0	0	0

[3]slack{NodeBalanceLTE}	[4]slack{NodeBalanceGTE}	[5]slack_bid00_{BidBlockMaxLTE}	[6]slack_offer00_{OfferBlockMaxLTE}
1	1	0	0
0	1	0	0
0	-1	1	0
0	0	0	1
0	160	0	0

[3]slack{NodeBalanceLTE}	[4]slack{NodeBalanceGTE}	[5]slack_bid00_{BidBlockMaxLTE}	[6]slack_offer00_{OfferBlockMaxLTE}
1	1	0	0
0	0	1	0
0	-1	1	0
0	1	-1	1
0	70	90	0

Figure 127: Right side of the tableaux csv

Why the objective value is still zero

Now that the cleared bid is a basic variable it can take a non-zero value but because the RHS of its basic constraint is zero the cleared bid is constrained to zero, and therefore the objective value is still zero.

The next entering variable

Now that the cleared bid variable is basic it is constrained by the RHS of its constraint. The cleared bid can only increase if enabled to do so by another variable in this constraint. The other variable would need to have a negative factor in the constraint… increasing the value of this other

variable would allow the cleared bid variable to increase.

We know the contents of the node balance constraint in this model, i.e., cleared bids out of the bus must be matched by cleared offers in, so it is no surprise that the variable that will allow the cleared bid variable to increase is the cleared offer variable.

The reduced cost for cleared offers is now -$90, i.e., the difference between the $70 cost of the cleared offer (i.e., its original reduced cost) and the $160/MW benefit resulting from the increase in cleared bids that the cleared offer would enable.

This difference was calculated when the row operations zeroed the reduced cost of the cleared bid, passing the per MW benefit of the cleared bid along to the cleared offer, offsetting the existing cost of the offer.

Now that the cleared offer variable is the variable with the most negative reduced cost, it will be the next entering variable. The next (and, in this small model, final) step is for the cleared offer variable to enter the basis. This will follow the same procedure as above… find the entering row as the row which will allow the necessary row operations without making any basic variables negative, and then perform row operations in order to allow the

entering variable to become basic (by meeting the requirement that it has a factor of 1 in the entering row and zero in all others).

A closer look at the entering row

The entering variable becomes the basic variable in the row with the lowest ratio of RHS to factor, because choosing any other row would create a negative RHS.

Looking at the numbers, we can relate this to the reality of our model. One of the constraints in tableau02 is the offer max constraint, which in the initial tableau prevents the offer from clearing more than its offered quantity. After the first pivot, this constraint now acts to limit the cleared offer to be no more than the cleared bid.

The row operations of the pivot effectively moved the relationship between the bids and offers from the node balance constraint to the bid max constraint.

This "new" constraint, which limits the cleared offer to be no more that the cleared bid (which is the bid max), becomes the entering constraint for the cleared offer. We can see that if we had allowed the cleared offer to enter via the other constraint, i.e., the offer max constraint, then the cleared offer

quantity would be more than the cleared bid quantity, breaking the node balance.

The algorithm can't see this, but it selects the bid max row as the entering row for the cleared offer because it is the constraint with the lowest ratio of factor to RHS, which means the same thing as our observation… it is the constraint that will bind first when the entering variable is increased that determines the entering row.

Solve complete

When the cleared offer enters the basis, the RHS of the bid max constraint sets the value for the cleared offer. Because there are now no negative reduced costs there is no way to further improve the objective, hence the solve is complete.

"The tremendous power of the simplex method is a constant surprise to me" - George Dantzig (inventor of the simplex algorithm)

Extracting the results from the final tableau

Quantities from RHS values

In the final tableau shown in Table 16 the cleared offer and cleared bid variables are both basic, i.e., able to be non-zero.

The cleared bid (column 1) is basic in row 2, with a RHS of 100, the cleared offer (column 2) is basic in row 3, also with a RHS of 100. These RHS values provide the values of the basic variables.

Table 16: The final tableau

Basic	$loadBid_{Clrd}$ [1]	$genOffer_{Clrd}$ [2]	s_1 [3]	s_2 [4]	s_3 [5]	s_4 [6]	RHS
[3]			1	1			
[1]	1				1		100
[2]		1		-1	1		100
[6]				1	-1	1	150
				70	90		9000

Shadow price of constraints

As discussed in Tutorial 1: Explaining Prices, the shadow price of a constraint is the rate of change of the objective value due to relaxing that constraint. The shadow price therefore indicates the value of the quantity that the constraint is constraining.

The bus price, i.e., the value of power at a bus, is the shadow price of the node balance constraint... the node balance constraint constrains the flow of power into and out of the bus, therefore the value of relaxing the node balance constraint tells us the value of electrical power at the bus.

Shadow price from final tableau

The shadow prices can be extracted from the final tableau.

The values in the final tableau represent the optimum objective value and also meet the requirements of the original constraints. For example, given the original set of constraints (shown in Equation 32) we can plug in the values from the final tableau and the equations will solve.

Equation 32: Constraints of the linear programme (again)

$$-loadBid_{Cleared} + genOffer_{Cleared} + s_1 = 0$$

$$loadBid_{Cleared} - genOffer_{Cleared} + s_2 = 0$$

$$loadBid_{Cleared} + s_3 = 10$$

$$genOffer_{Cleared} + s_4 = 20$$

Using the values from the final tableau, the second node balance constraint is shown in Equation 33.

$$\begin{array}{ccccccc} loadBid_{Cleared} & - & genOffer_{Cleared} & + & s_2 & = & 0 \\ 100 & - & 100 & + & 0 & = & 0 \end{array}$$

Equation 33: Node balance with final tableau values

We can see from Equation 33 that decreasing the value of slack variable s_2 relaxes the node balance constraint, and therefore the impact on the

objective value of decreasing s_2 in the final tableau will provide the shadow price of the node balance constraint in the final tableau, and therefore the bus price.

To find the impact on the objective value of decreasing s_2 remember that the progress of the algorithm was driven by looking for negative reduced costs because they indicate the rate at which the objective value will improve if we increase the associated variable.

From this it follows that a positive reduced cost indicates the rate at which the objective will *decrease* if we increased the variable. And also, the observation that helps us here, a positive reduced cost indicates the rate at which the objective value will increase if we could *decrease* the variable.

The positive reduced cost of s_2 provides the rate of objective value increase that would result from decreasing s_2. Decreasing s_2 relaxes the node balance constraint that originally contained s_2. Therefore, the positive reduced cost of s_2 provides the improvement to the objective value due to relaxing the node balance constraint.

From the final tableau we can see that the reduced cost of s_2 is 70, which is therefore the shadow price of the node balance constraint.

The above is true of any of the constraints. The positive reduced cost in the final tableau of the constraint's original slack variable will provide the shadow price of the constraint.

Viewing the simplex iterations

As well as exporting the tableaux to see what the simplex algorithm did, you can also view details of the simplex solve from within the app.

Iteration details

On the Results display select the Iterations row, this will take you to the iteration details as displayed in Figure 128 (these are the iterations from the small model that we have just studied).

Figure 128: Iteration details display

Graphical view of iterations

The simplex iterations can also be viewed as a chart by tapping the Chart button on the Iterations display (this chart is also displayed while the solver is solving). The chart shows iterations vs objective, as displayed in Figure 129. This chart shown is from the Hawkes Bay 033 sample case which was presented in Tutorial 8: Actual Market Data.

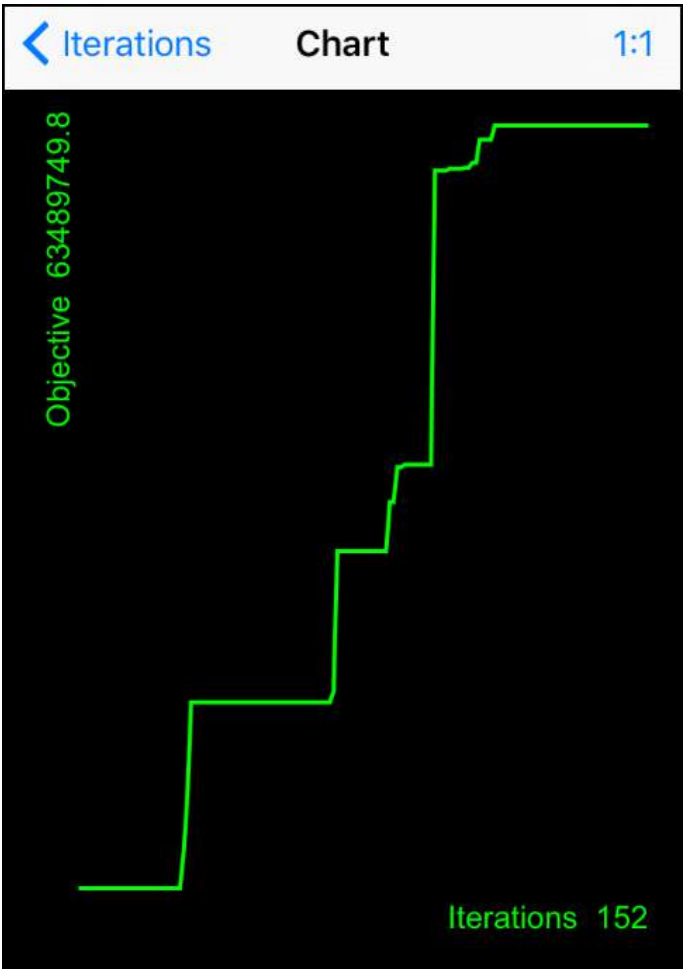


Figure 129: Iterations vs objective (shown here for the Hawkes Bay 033 model)

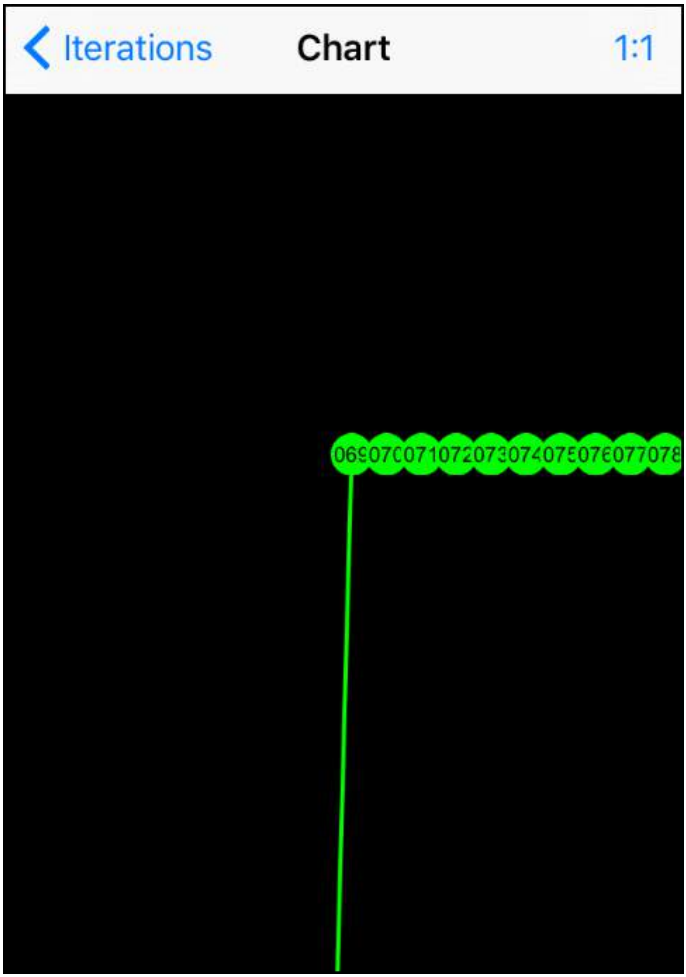


Figure 130: Zoom in to view individual iterations

Use the pinch gesture to zoom in and view the individual iteration points, as shown in Figure 130. Tapping on an iteration point will present the

simplex actions that produced this result, see the example in Figure 131.

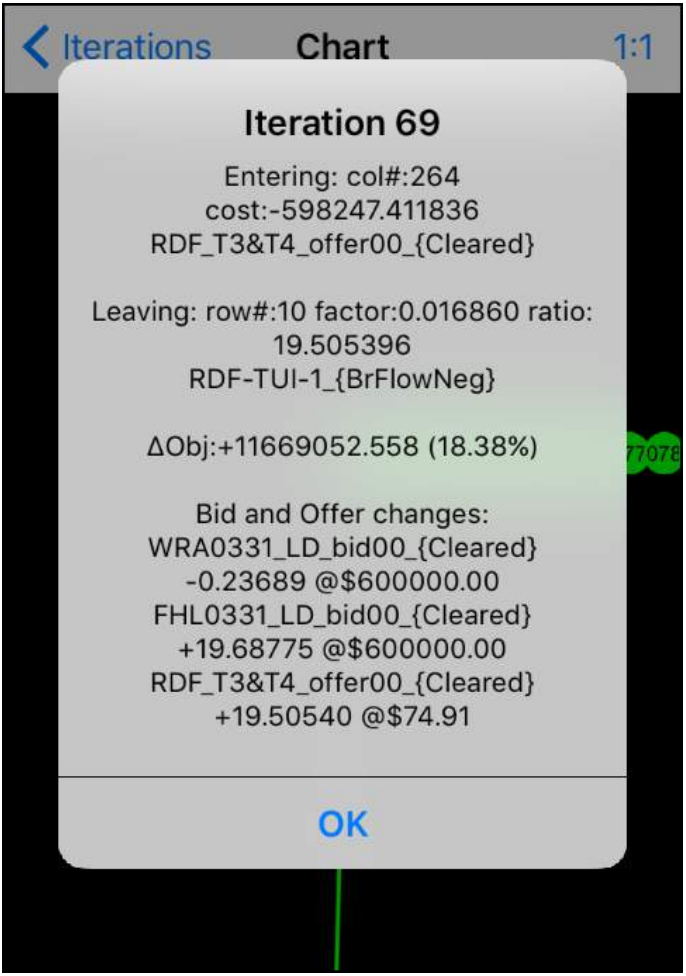


Figure 131: Tap iteration point to view simplex actions

Summary

In this section we built and solved a small electricity model then exported its tableaux and tracked in detail how the simplex algorithm arrived at the result. Along the way we explained how and why the simplex algorithm makes the decisions that it does, and why it works.

When the result was complete, we saw how the prices and quantities are extracted from the final tableau.

We also saw how to view the simplex iterations in the app, either via a description of the iterations, or graphically as a chart of objective value vs iteration count.

Using the Simplex Nodal App

This section walks you through how to use the Simplex Nodal app to build and solve electricity market models.

Build an electricity market model

Starting a new model

The "new model" button shown in Figure 132 removes the existing model (if any) from the display.

Figure 132: The "new model" button

If the existing model has not been saved, then it will be lost. Hence, if there is a model currently displayed then the alert shown in Figure 133 is raised.

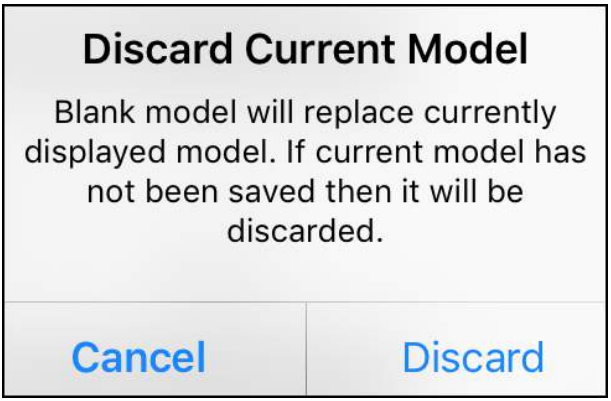


Figure 133: Warning when "new model" is actioned

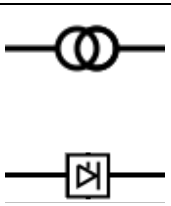
How to save and load a model is covered in the Controls and Displays section.

Components of the network model

The first step in creating a new model is to build the electricity network. This network is built using the components shown in Table 17.

Table 17: Components of the electricity network model

Button	Visual	Description
Bus		A busbar that provides a common connection point at a substation… a node on the network.
Branch		Transmits electricity between buses. Represents either a transmission line, a

		cable, a transformer, or an HVDC link. Display options: line/cable, transformer, or HVDC link
Gen		An electricity generator
Load		An electricity consumer

Toolbars

Use the toolbar buttons to add components to the network model. The toolbar looks like this:

On the iPhone, because it is smaller, there are two toolbars; the Main toolbar:

…and the Build toolbar:

On the iPhone the Main toolbar includes a button that takes you to the Build toolbar and vice-versa.

Adding components

To build the model shown in Figure 134 tap the toolbar buttons: Bus-Bus-Gen-Load-Branch.

The default placement of the components automatically builds this model. Components can be moved and re-sized as necessary to build more complex models.

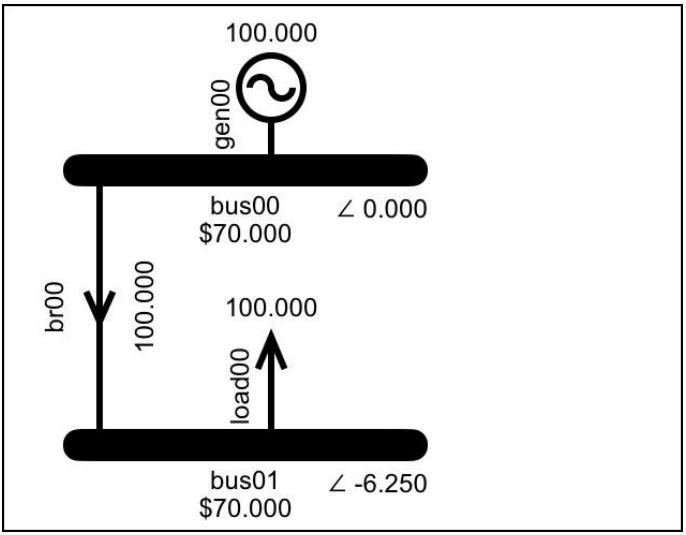


Figure 134: Simple model incorporating all component types

Viable components

Components that are connected together create electrical islands. Components that are *not* capable of making a valid contribution to the solution are

assigned to island 0, coloured green and excluded from the solve.

A component cannot only contribute if it is fully connected, and this means different things for different component types. A generator or load is only fully connected when connected to a bus, while a branch is only fully connected if both ends are connected.

A bus is only fully connected if it has two or more different components connected to it, or two branches connected to it, or it is connected to another bus that is fully connected. A bus that has only a load or only a generator connected to it is not capable of making a meaningful contribution to the result.

Figure 135 shows examples of viable components and non-viable (island 0) components. In the example where the bus has two branches that connect to another bus that also only has the same two branches connected, this is not actually a viable island, but it is not marked as island 0… the assessment is not intended to be foolproof, it is a screening; the important thing is that it removes most non-viable components, and it won't accidentally exclude viable components.

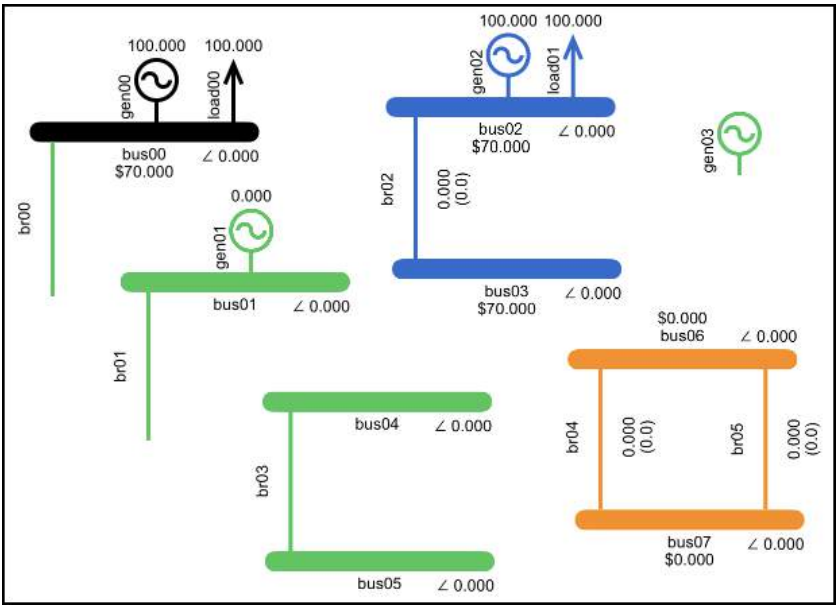


Figure 135: Island 0 components are coloured green and are excluded from the solve

Note also that the term "island 0" is somewhat misleading. Viable islands consist of components that are connected together, and in a viable island all the components in the island have the same island number. Components that are assigned to island 0 are not necessarily connected… the only thing they have in common is that they are not able to make a meaningful contribution to the result.

Parameters

Once a component is added to the network model you can double-tap it to edit its parameters. The components and their parameters are listed in Table 18.

Table 18: The components and their parameters

Component	Parameter(s)
Bus	Has an editable parameter that indicates if it is the reference bus. (Each electrical island has one reference bus, which by default is the first bus added, but can be changed)
Branch	1) Resistance, susceptance and maximum flow. Reactance can also be entered instead of susceptance in which case the susceptance is calculated from the resistance and reactance. 2) An option to display the branch as a circuit, a transformer, or an HVDC link. 3) An associated Losses display has parameters that determine how many loss segments there are and how the segments are calculated.

Gen	1) Energy offers, which have a price and quantity. 2) An associated Reserves display has reserve offers that have a price and quantity. The Reserves display also has parameters for total capacity, PLSR% and whether or not this gen can be a risk setter. 3) An associated Ramp display has an initial MW parameter and an up-ramp-rate parameter.
Load	Energy bids, which have a price and quantity. There is also the option to use the bid price entered for block 1 as the bid price for all other block 1 bids in the model.

Default parameters

The parameters can be edited, but the first few tutorials use the default parameters.

The default parameters can be changed by tapping the Data Display's eyedropper button, indicated in Figure 136, and then tapping OK on the alert shown in Figure 137.

The default values for reserves and ramp-rates cannot be changed.

Figure 136: Eyedropper button updates the default parameters

Figure 137: Alert to confirm changing the defaults

Solving the model

Having built the model shown in Figure 134, tap the Solve button to open the Solve Settings display. Select the solver options shown in Figure 138 and then tap the Solve Now button.

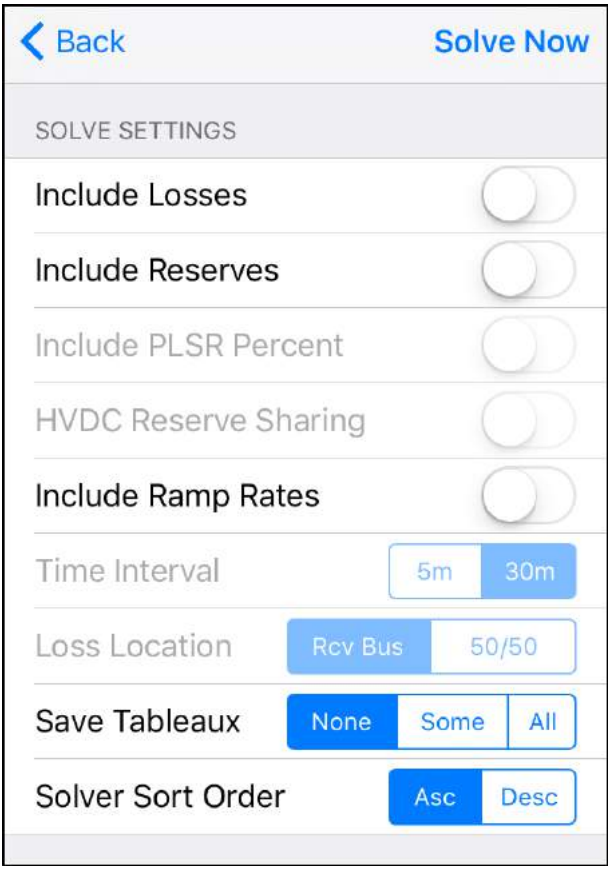


Figure 138: The Solve Settings display

Solver progress

As the model is solved, the simplex algorithm performs iterations that improve the objective value. The solve is complete when the objective value cannot be improved any further; this is

explained in Tutorial 1: Explaining Prices, and also in Tutorial 9: Simplex Explained.

A plot of the objective value vs the iteration count is displayed as an overlay while the model is solving. For a large model such as the Hawkes Bay sample (see Tutorial 8: Actual Market Data) the solve takes a noticeable amount of time.

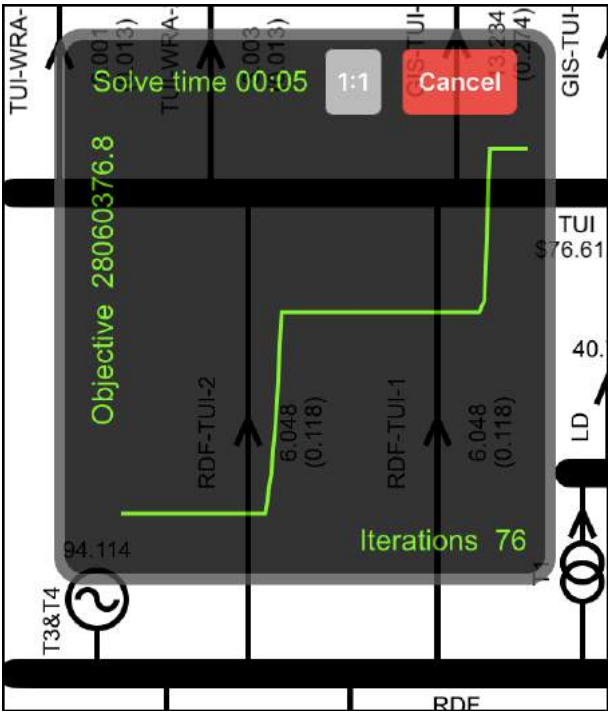


Figure 139: Objective vs iterations as overlay during solve

While the solve is in progress the chart displays a "Cancel" button, as shown in Figure 139. When the

solve is complete this changes to a “Done” button, as shown in Figure 140. The Done button is used to dismiss the chart.

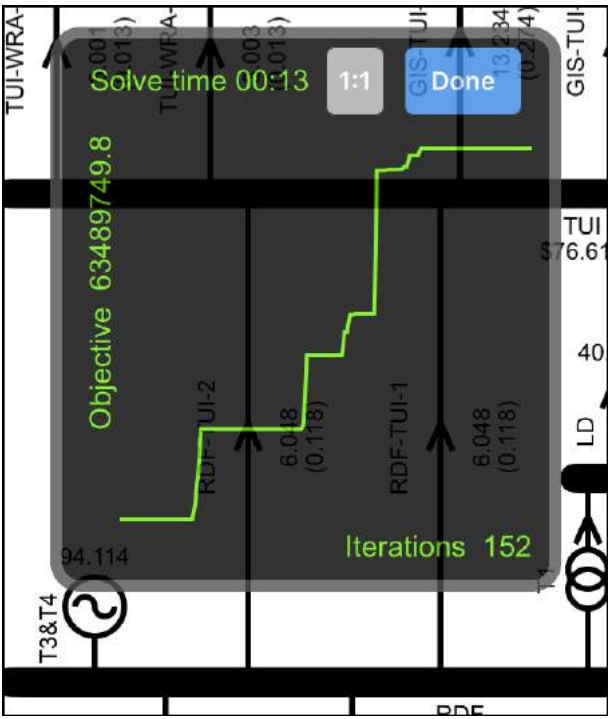


Figure 140: Objective vs iteration count as overlay during solve

For a small model the chart will be far less interesting, as shown in Figure 141, and the solve will be completed almost immediately.

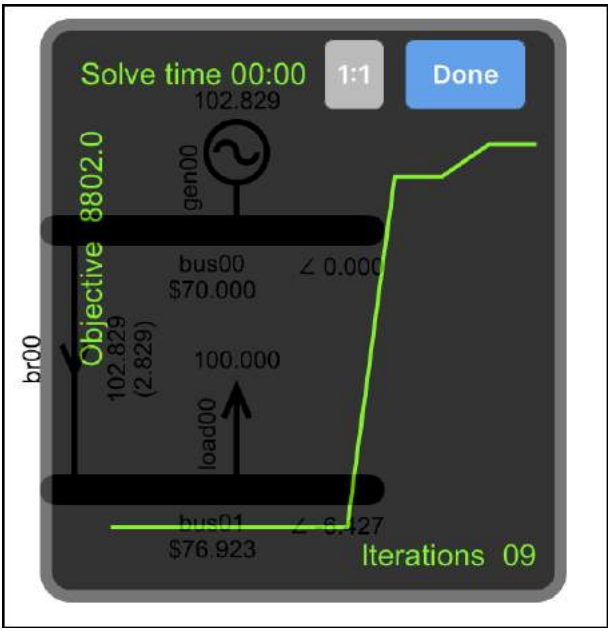


Figure 141: Objective vs iteration count for a smaller model

You can zoom into the plot to see the individual iteration points and tap them to see their details. The 1:1 button is available to make it easy to zoom back to 1:1 scale.

Viewing the results

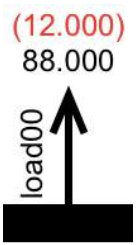
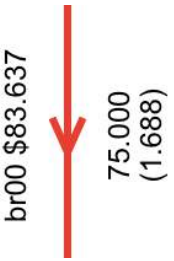
Results are displayed on the network model alongside the associated component. As shown in Figure 134 the cleared generation is displayed above the generator, the branch flow is displayed beside the branch, etc. More detailed results can be viewed via a component's Data Display, which is

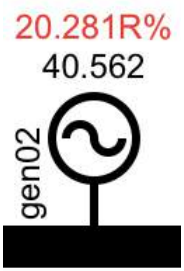
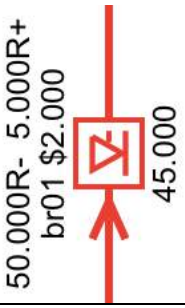
accessed by double-tapping the component. System level results can be viewed via the Results display.

Results on the network model

The network model display includes colours that identify binding constraints, as shown in Table 19.

Table 19: Colours used to indicate results

	Load bid is not fully cleared
	Branch flow is at max, i.e., binding branch
	Generator is binding on capacity limit

	Generator has reserve constrained by PLSR%
	Generator is constrained by a binding up-ramp-rate
	Generator is risk-setter
	HVDC link is binding on capacity (energy + reserves) constraint. Capacity is 50MW… 5MW of energy transfer and 45MW of reserve shared in same direction, 50MW of reserve in the opposite direction.

	HVDC link is risk setter

Results on the Results display

The Results display is accessed via the toolbar button labelled "Results". As shown in Figure 142, the Results display is divided into two sections; metrics relating to the simplex algorithm's solving of the model, and quantities relating to the system being modelled.

All results include a △ column, which shows the change from the previous solve.

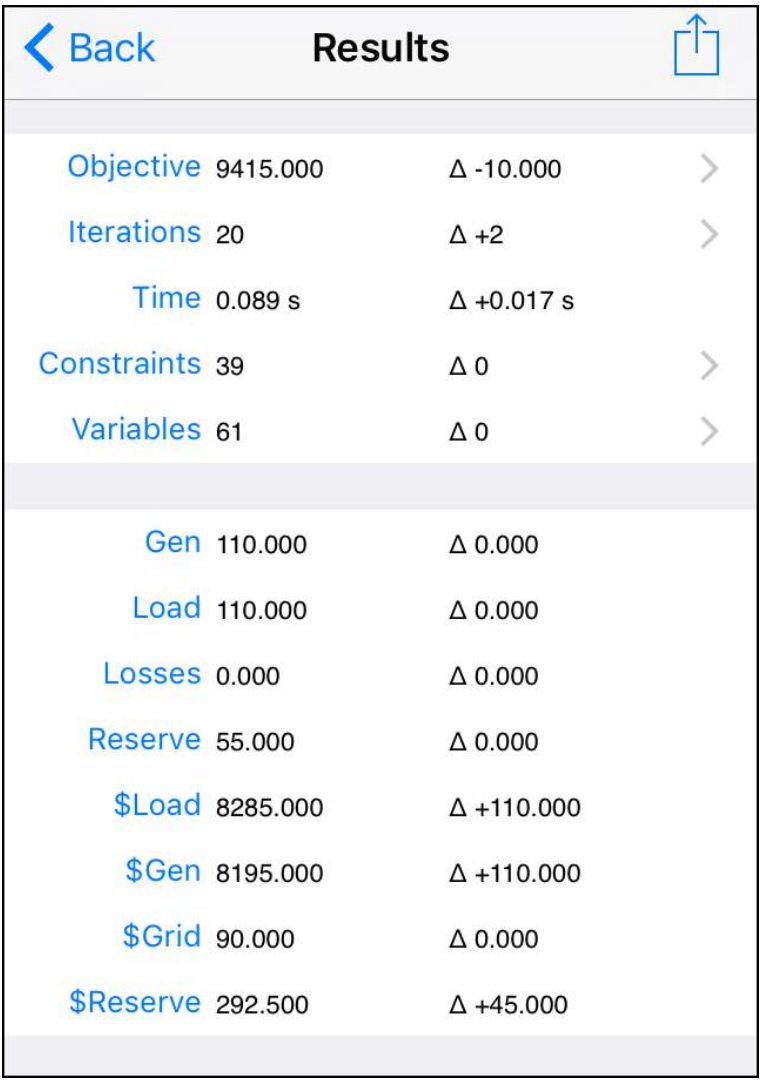


Figure 142: The Results display

If a row has the detail icon > then tapping that row leads to a display showing the details of the result, as described in Table 20.

Table 20: Rows that lead to displays showing detailed results

Results Row	Leads to detail…
Objective	How the objective value was calculated
Iterations	Details of all the actions taken by the simplex algorithm and a link to a plot of objective value vs iteration count
Constraints	All the constraints in the model and their shadow price
Variables	All the variables in the model, their value and whether they are basic or non-basic

Shadow prices are explained in Tutorial 1: Explaining Prices. Basic and non-basic variables are explained in Tutorial 9: Simplex Explained.

The Import/Export button on the Results display

The toolbar on the Results display includes the import/export button indicated in Figure 143.

Figure 143: The Import/Export button

The import/export button leads to the Import Export display, which allows for the export of the following data via email… the simplex tableaux as a csv file, the results as a csv file, a screenshot of the network model, the complete model as a csv file (which can also be imported to the app via email or iTunes), and the complete model as a GAMS file. See the Controls and Displays section for more details on exporting and importing.

Adjusting components

When we built the model shown in Figure 134 we left the components where they landed. However, components can be moved, re-sized or deleted.

The selected component

Before a component can be adjusted, i.e., moved or re-sized, it must be selected. A component is selected by tapping it once.

There is only one selected component at a time. The selected component has a red or green box around it; the box is green if the component is fully connected, red if it is not.

Gen and load components are fully connected if they are connected to a bus; a branch is only fully connected if both ends are connected to a bus, a bus is always fully connected. Figure 144 shows an

example of a branch that is the selected component but not fully connected.

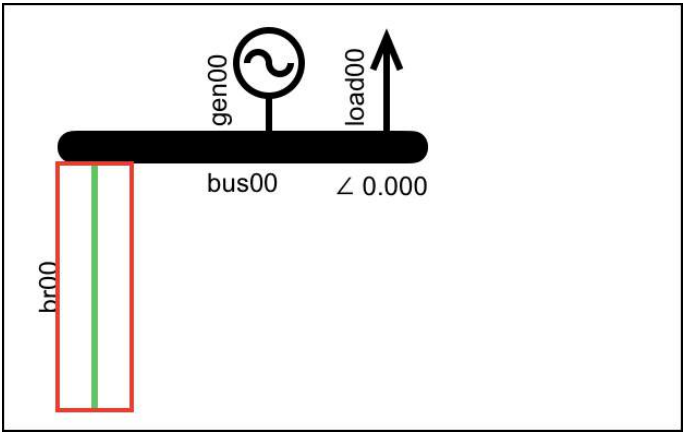


Figure 144: Branch that is selected but not fully connected

Moving or resizing a component

The selected component is moved by dragging its centre. Buses and branches can also be re-sized, by dragging from the end. Figure 145 shows where to grab buses and branches in order to re-size or move them (these delineations are not normally visible).

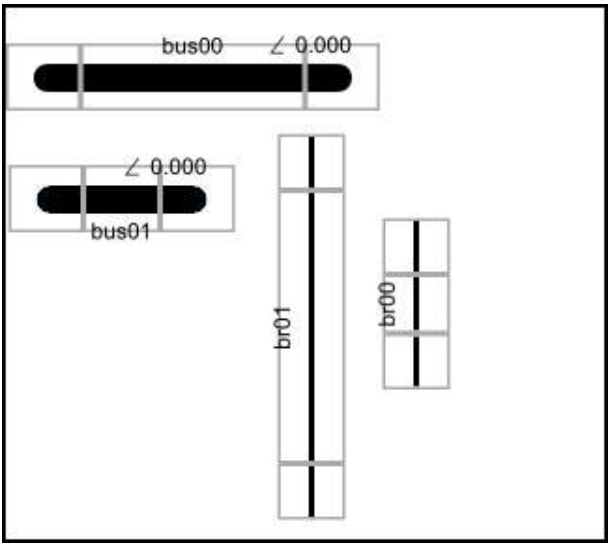


Figure 145: Drag from the middle to move, drag from the end to resize

When a bus is moved, the gen, load and branch components that are connected to the bus move with it.

Deleting a component

To delete a component, press and hold it… a cross will appear, as shown in Figure 146, and the component will wobble. Tap the component to delete it. Tap anywhere else to cancel the delete.

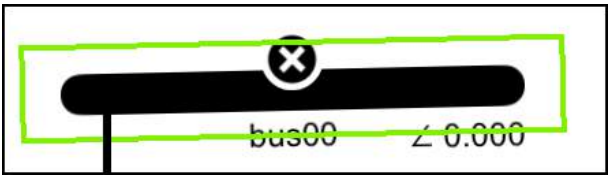


Figure 146: Select a component for deletion by press and hold

Zoom and scroll

Zoom and scroll enable the iPhone to edit an iPad-sized model. The extents of the large model are accessed by scrolling to move around, or by zooming in and out using the pinch gesture.

For models that fit on the iPhone screen without zooming, it is easier to work if zoom/scroll is switched off. Zoom/scroll is switched on and off via the Settings menu.

On the iPad there is no zoom, but scrolling is available when you operate in landscape mode.

Colours

Viable islands

Components that are connected to a bus will be in the same electrical island as that bus.

If the bus does not have enough components connected to it in order to form a viable result then it will be in island 0, as described above.

When there are enough connected components then a viable island is formed. The first viable island is assigned island number 1. In most of the worked examples we will only have one island. Multiple islands appear in Tutorial 7: HVDC Link.

Reference Bus

The first bus that forms a viable island becomes the reference bus for that island. When the model is solved, the reference bus has its phase angle set to zero, while all other phase angles in the island are variables that the solver can modify. Phase angles are explained in Tutorial 2: Modelling Transmission.

Connectivity colours

All components that are connected together are in the same island and have the same colour. The exception to this are the non-viable components, which are all in island 0 even if they are not connected together. Components in island 0 are always green.

Changing colours

Island colours, except for island 0, can be changed via the Settings display, see the Controls and Displays section for details.

Controls and Displays

This section covers the controls and displays of the Simplex Nodal app.

Data Display common controls

Double-tap a component to access its Data Display. The controls indicated in Figure 147 are common to a number of the Data Displays. These controls are described in the following sections.

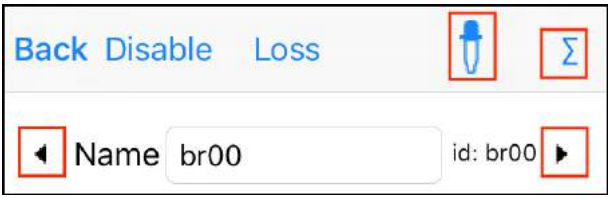


Figure 147: Controls common to a number of Data Displays

Navigation between displays

Common to all components:

◀ and ▶	Navigate to the Data Display for the previous/next component of the same type, e.g., for a branch navigate to the previous/next branch

Setting default parameters

Common to the Data Displays for Branch, Gen and Load components:

	Set the parameter values currently displayed as the default values for all new components of this type. Default values can be reset to the original defaults via Settings – Reset Default Values

Viewing variables and constraints

Common to the Data Displays for Branch, Gen and Load components:

	Displays the component's variables and constraints, as created by the latest solve. Variables and constraints can also be viewed via the Results display.

Naming a component

When components are added they are automatically assigned an i.d. and a name. The name is used for display purposes and is initially the same as the i.d. The i.d. cannot be changed, but the name can be changed via the Name field on the Data Display, as shown in Figure 148.

Figure 148: Name data entry for bus

Note that for the branch component there is also an auto-naming option available, this is described in the Settings section below.

Data Display for a Bus component

Shows whether or not the bus is the reference bus for the island. Includes the option to "Set as reference bus" by tapping the button indicated in Figure 149.

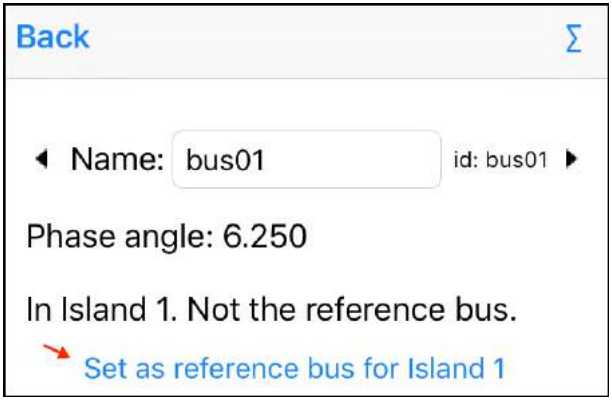


Figure 149: Button to make this bus the reference bus

Also shows the phase angle from the latest solve, useful if you choose to not show the phase angle on the main display.

Data Display for a Gen component

Energy Offers

Figure 150 shows the Data Display for a Gen component, where you can edit the name of the component and enter its energy offers. The lower half of the display shows the cleared offer quantities from the latest solve.

Figure 150: Data Display for Gen component

The toolbar has the "Reserve" button that leads to the Reserve display and the "Ramp" button for the Ramp Rate display.

Figure 151: Reserve Data Display for Gen component

Reserve display

The Reserve display shown in Figure 151 is where you enter the reserve offers, the generator capacity, the PLSR%, enable/disable the PLSR% constraint,

and define the generator as a risk unit. Note that reserves are only included in the model if Reserve is enabled via the Solve Settings display.

The interaction between Risk, Reserve Offers, Capacity and PLSR% are described in Tutorial 5: Risk and Reserve.

The lower half of the display shows the cleared reserve quantities from the latest solve.

Figure 152: Non-zero reserve offers and zero Capacity

The "∑" button sets the Capacity value to the sum of the energy offers. The button is only displayed if the Capacity is not already equal to the sum of the energy offers.

The capacity constraint is only applied if the generator has non-zero reserve offers and the solve

settings has Reserve enabled. When applied, the capacity constraint will limit the sum of cleared energy and reserve offers to be no more than the capacity of the generator.

If there are non-zero reserve offers but the capacity limit is zero then the Capacity will be highlighted red as shown in Figure 152, to alert that if the value is not changed and Reserve is enabled then the sum of the cleared energy and reserves will be limited to zero.

If you leave the Reserves display with non-zero reserve offers but a capacity of zero, then the warning shown in Figure 153 is displayed.

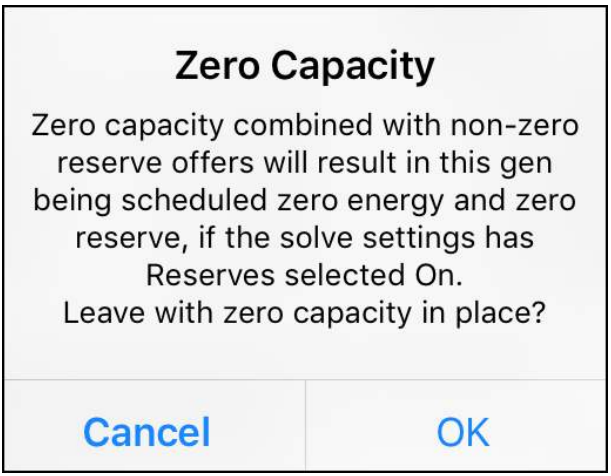


Figure 153: Warning when leaving Reserves display

Ramp Rate display

The ramp rate constraint restricts the generation based on how far the Ramp Rate Up value allows the generation to move from the Initial MW, as explained in Tutorial 6: Ramp Rates.

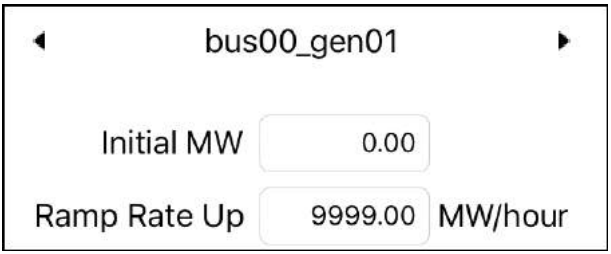


Figure 154: Ramp Rate Data Display

The Initial MW and the Ramp Rate Up are entered via the Ramp Rate display shown in Figure 154. The Ramp Rate Up has a default value of 9999, which ensures that for any realistically sized generator the ramp rate constraint will not bind unless the default value is edited.

Ramp rate constraints are only included in the model if Ramp Rate is enabled on the Solve Settings display. Ramp Rate constraints are not created if the ramp rate limit exceeds the sum of the offers (because the ramp rate limit would have no impact).

Data Display for a Load component

The Data Display for a Load component is shown in Figure 155. This is where you can change the load's name and enter the energy bids.

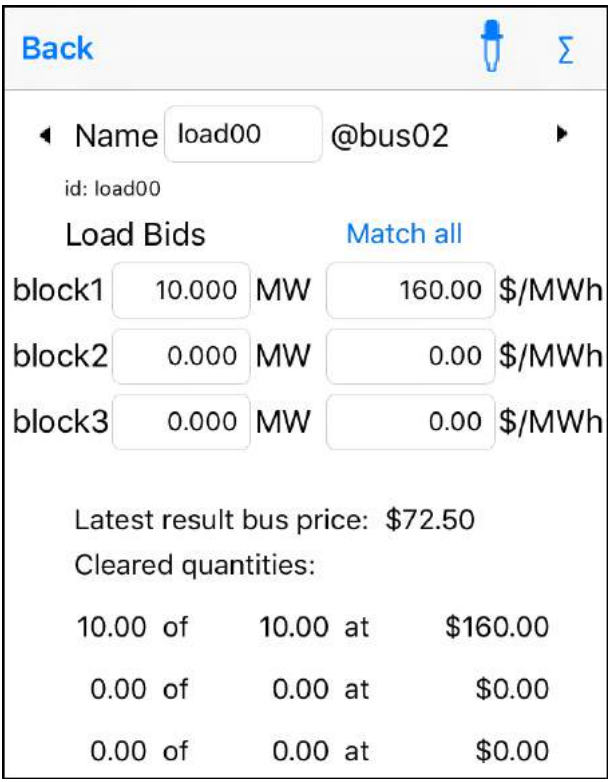


Figure 155: Data Display for Load component

The "Match all" button will update the block 1 bid price for all existing Load components to match the block 1 bid price of the currently displayed Load. This is useful if you want to treat the block 1 bid

quantity as a required load and use the block 1 bid price as the penalty cost for all required loads. Before "match-all" is applied the alert shown in Figure 156 is raised.

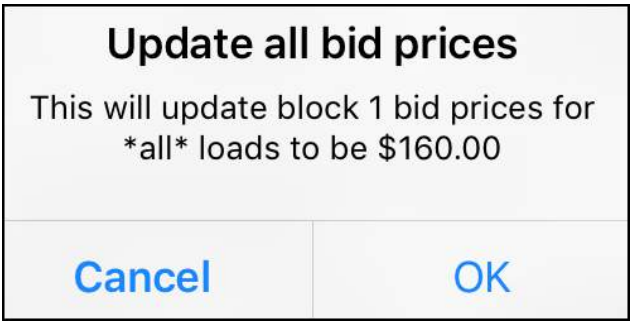


Figure 156: Alert raised after "Match all" button is tapped

Data Display for a Branch component

Branch parameters

Figure 157 shows the Data Display for a branch component. This is where you enter the branch parameters of maximum flow, resistance, and reactance. There is also the option to enter the susceptance instead of the reactance, by tapping the "View Susceptance" button. The relationship between resistance, reactance and susceptance is described in Tutorial 2: Modelling Transmission.

The display includes a switch to mark the branch as a transformer; the only thing that this changes is how the branch appears on the display.

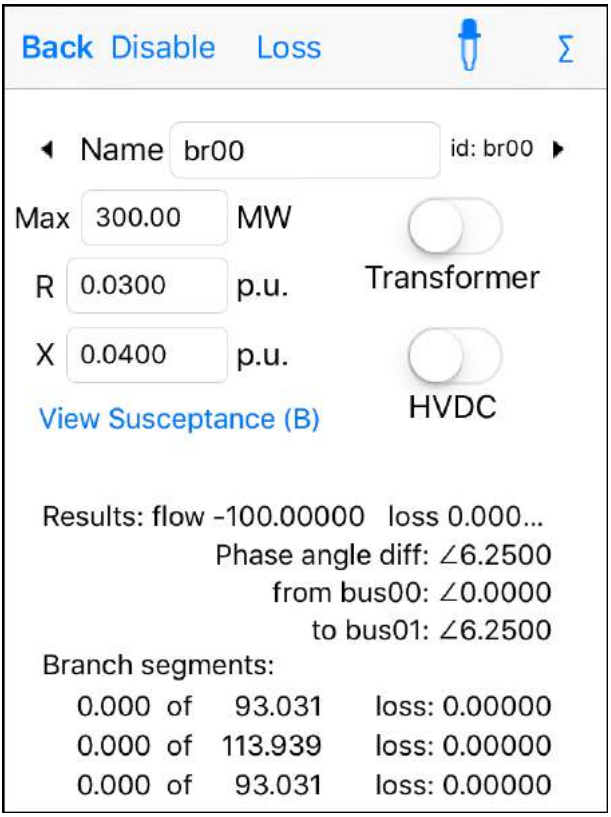


Figure 157: Data Display for a Branch component

The Data Display also includes a switch to define a branch as an HVDC link. The functional differences between an HVDC link and a normal branch are covered in Tutorial 7: HVDC Link.

The toolbar includes the Disable button, which removes the branch from the model without deleting the branch. After a branch is disabled the

Disable button changes to Enable as shown in Figure 158.

Figure 158: Disabled branch has an Enable button

Figure 159 shows the display differences for the various branch options.

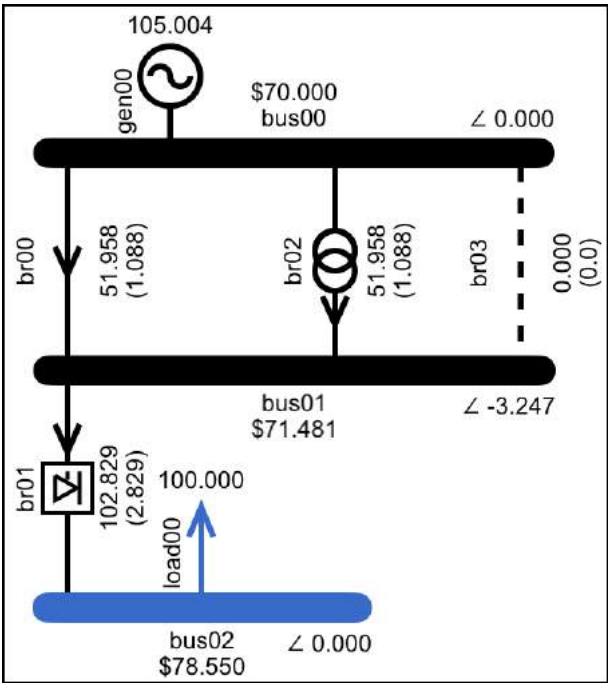


Figure 159: br00 is the default (a circuit), br01 is an HVDC link, br02 is a transformer, br03 is removed from the model

Branch losses

The Losses button on the Branch Data Display links to the Br Segments display shown in Figure 160, which shows the data for the individual segments that model the flow-loss parabola.

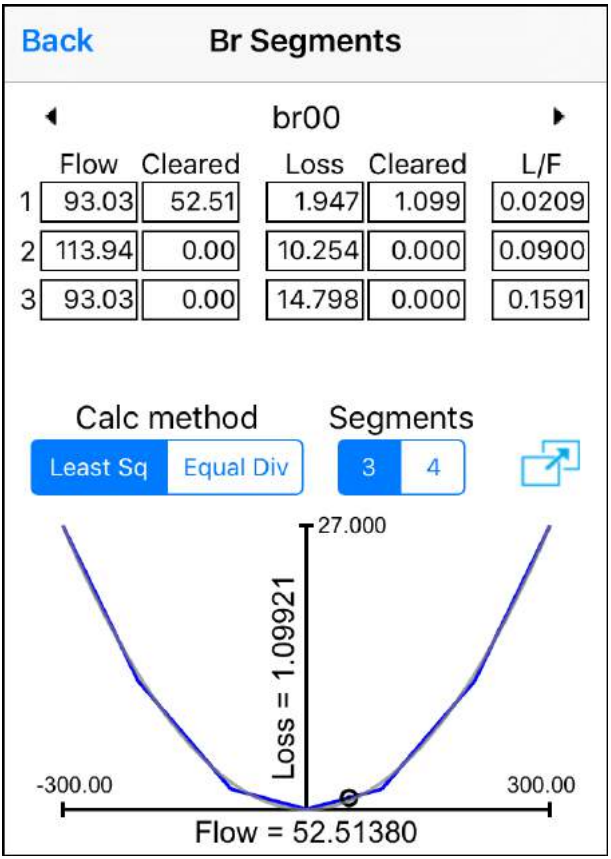


Figure 160: Branch segments display, i.e., branch flow-loss

The branch segments display includes options to change the number of segments, and to change the way that the segments are calculated. For a full discussion of these settings see Tutorial 4: Transmission Losses.

When a change is made to the settings, the segment data is re-calculated and the display is updated. Altered settings are highlighted red as shown in Figure 161. For the altered settings to be saved they must be applied to all branches in the model, which will happen automatically when you leave the display. Before this automatic update occurs, you will be prompted to confirm the update via the alert shown in Figure 162.

Figure 161: Altered loss segment settings are highlighted red

Figure 162: Segment changes will be applied to all branches

If you don't want to apply the change then select the Cancel option and undo the methodology change before leaving the display.

Full screen plot of flow-loss

The flow-loss plot is superimposed over the curve that represents the actual flow-loss parabola. You can use the pinch gesture to zoom in for a closer look at the flow-loss curve, as shown in Figure 163.

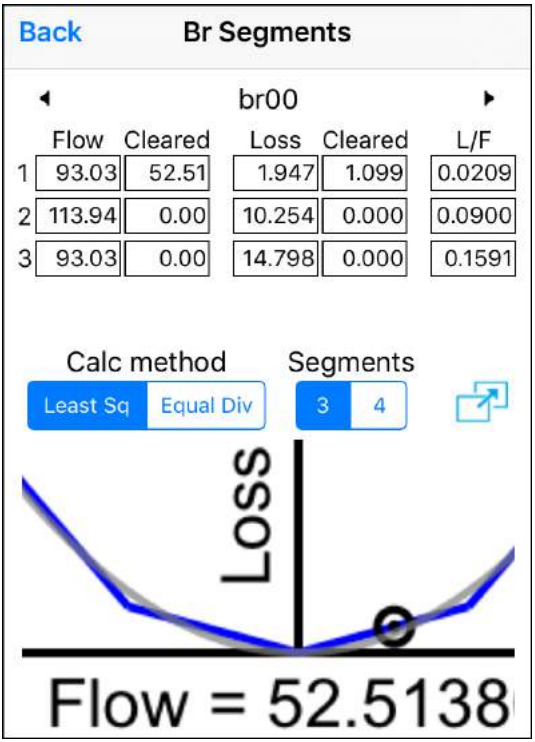


Figure 163: Flow-loss plot zoomed in

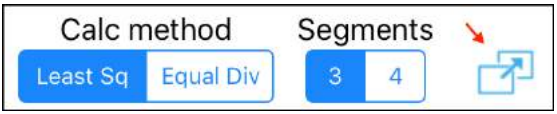


Figure 164: Button to view flow-loss curve full screen

The button indicated in Figure 164 leads to a full-screen plot of the flow-loss curve, as shown in Figure 165.

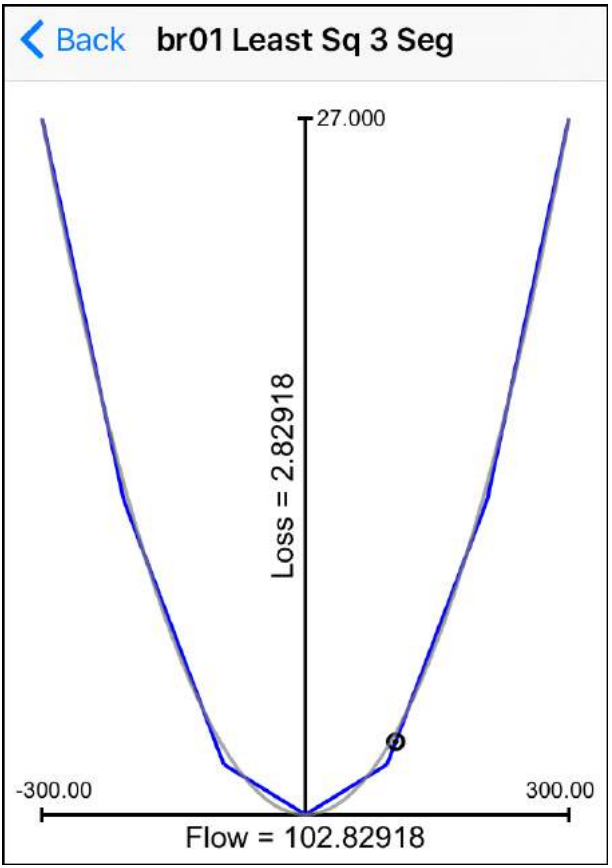


Figure 165: Full-screen view of flow-loss curve

Layout of bus labels

A bus's name label and phase angle label are situated in locations where they can get in the way, as shown in Figure 166. For this reason, every time a component is moved an automatic check takes place to see whether it would be better if the name and phase angle labels were positioned above or below the bus.

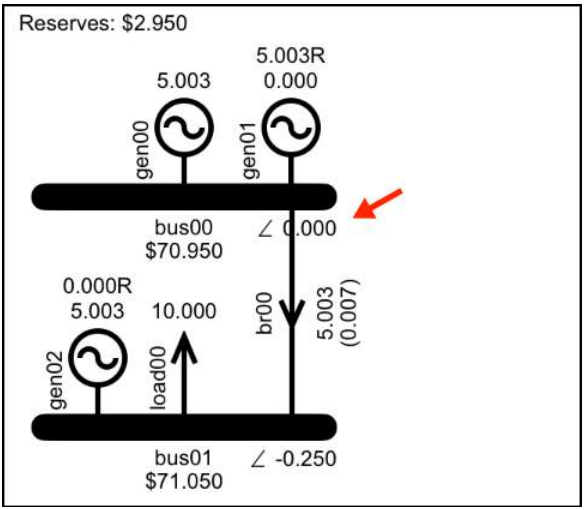


Figure 166: Phase angle label can get in the way

Figure 167 shows that if gen01 is moved then the location of the phase angle label for bus00 will be automatically adjusted.

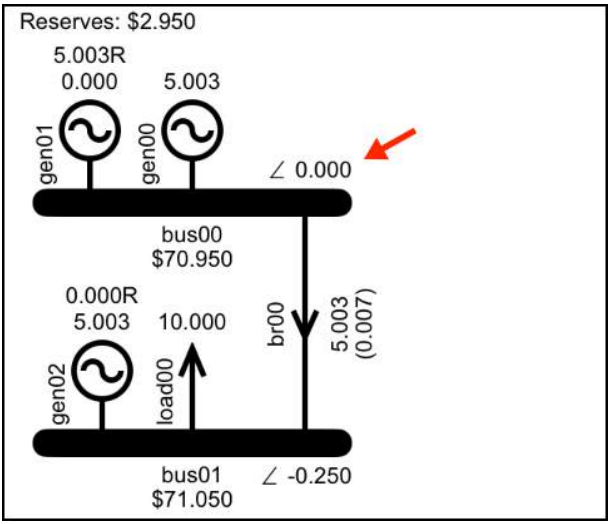


Figure 167: Automatic check moves label to better location

The automatic placement is not perfect because it is the size of the text box that determines whether or not there is an overlap, regardless of whether the text box is full. On the odd occasion when this is a problem you can get the placement to do what you want by disconnecting and re-connecting the troublesome component. If and when you save the model, the final placement of the labels is saved with the model.

Zoom and scroll

The tutorial examples can all be viewed full size on an iPhone screen, with the exception of the large sample model in Tutorial 8: Actual Market Data,

which can be viewed full size on an iPad screen as shown in Figure 168.

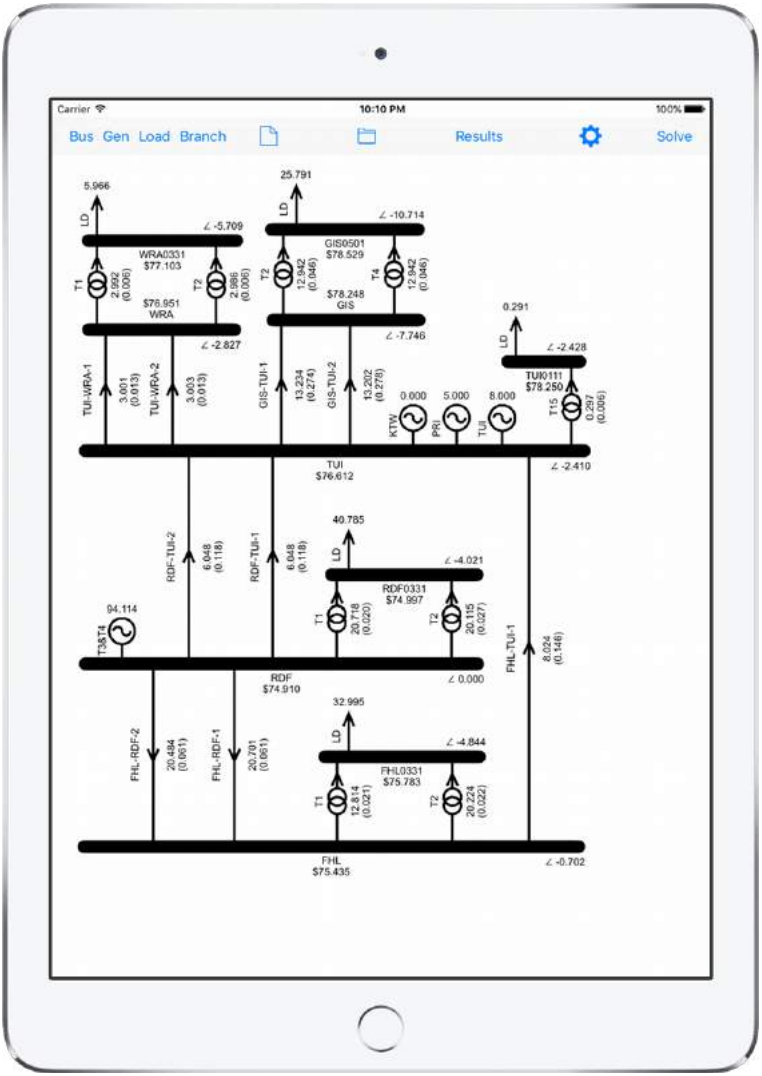


Figure 168: Large sample model on an iPad screen

The large model can be viewed on an iPhone screen by using zoom and scroll. When this model is first loaded onto the iPhone you will only be able to see part of it, as shown in Figure 169.

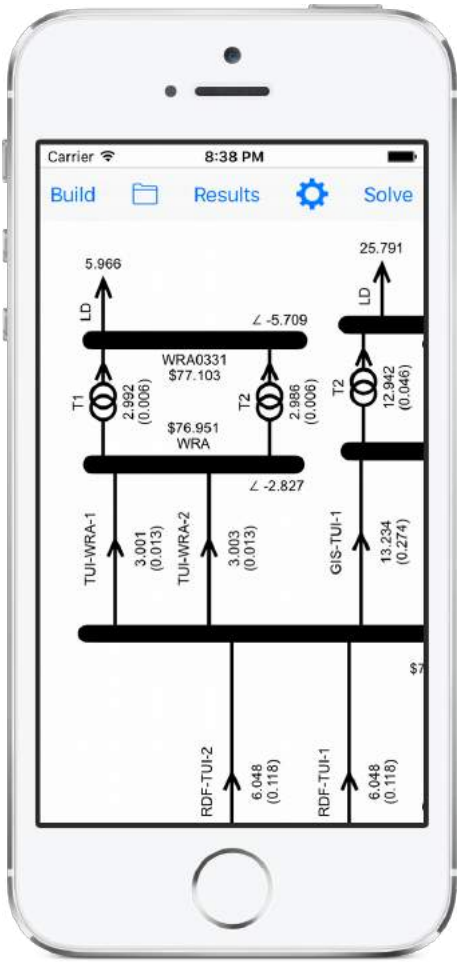


Figure 169: Large sample model at full size on an iPhone

With zoom/scroll enabled you can drag the background to scroll or use the pinch gesture to zoom out and view the full model as in Figure 170.

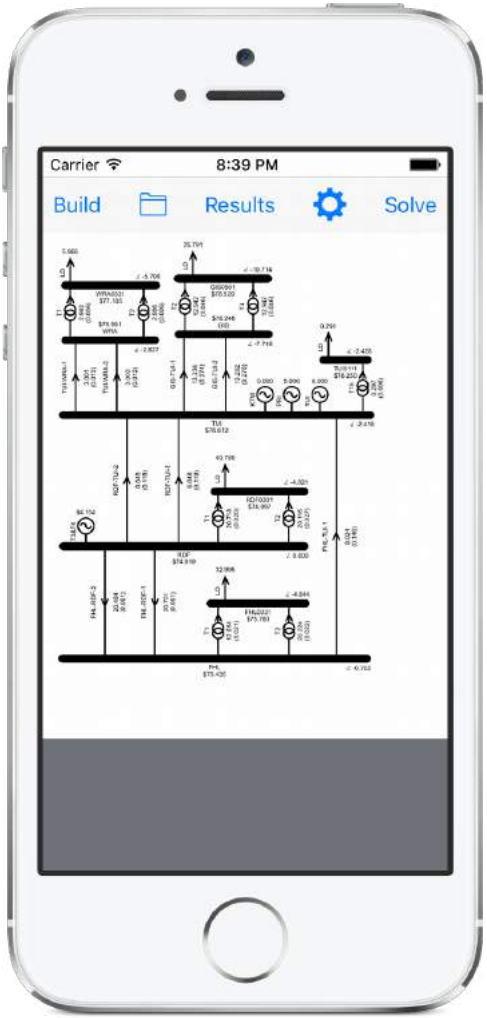


Figure 170: Large sample model zoomed out on iPhone

The iPad does not need zoom because the full size of the model is the size of the iPad screen. Scroll is available on the iPad because you may need to scroll if you operate in landscape mode.

Zoom and scroll for the iPhone is enabled/disabled via the Settings display.

Settings

The Settings display is accessed via the Settings button indicated in Figure 171.

Figure 171: The Settings button

The Settings display is shown in Figure 172, the individual options are described in the following subsections.

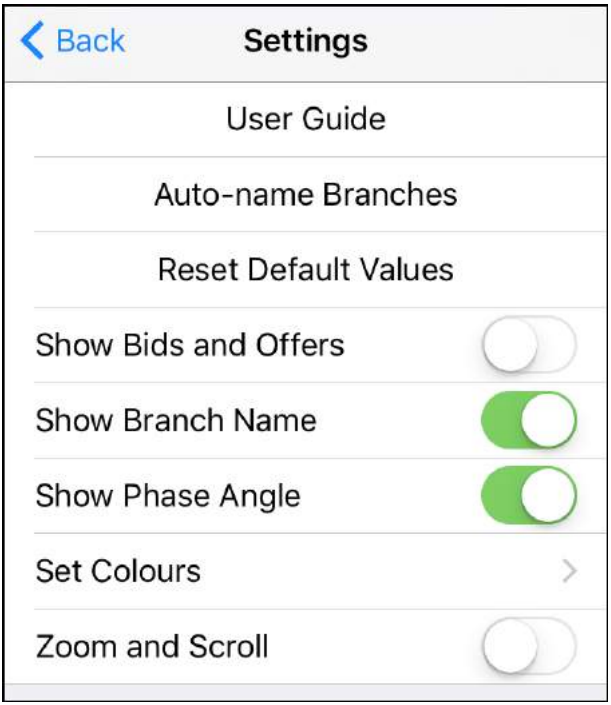


Figure 172: The Settings display

Settings: User Guide

The user guide display, shown in Figure 173, provides a brief guide to user actions that are not immediately obvious. It also provides a link to the location of this document, and an email link for the user to provide suggestions, feedback, or bug reporting.

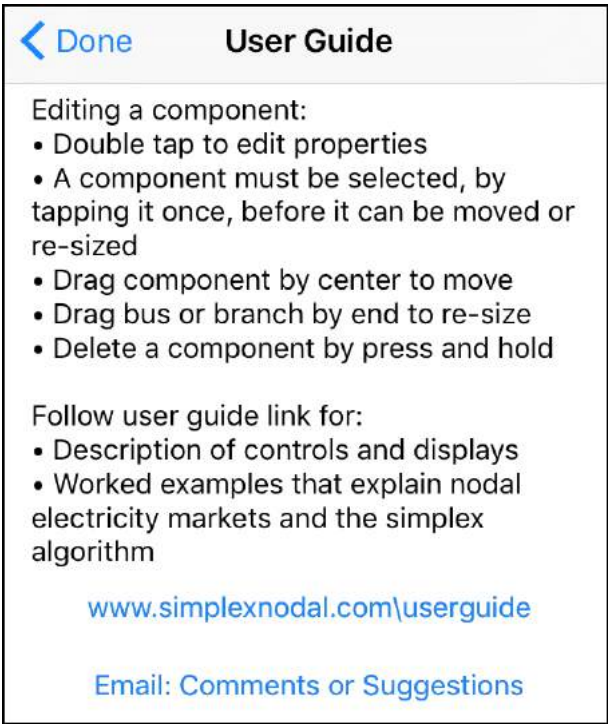


Figure 173: User Guide display

Settings: Auto-name Branches

Branches can be auto-named, following the convention of "fromBus-toBus-*number*", where:

- fromBus and toBus are the buses at either end of the branch, and fromBus is the bus whose name is alphabetically before the name of toBus

- *number* is used to differentiate between branches when more than one branch connects the same two buses
- the delimiter can be selected to be either a dash or an underscore

After tapping the "Auto-name Branches" button the alert shown in Figure 174 is raised to let you know that any existing branch names will be over-written. The prompt provides the option of delimiting the names using either underscores or dashes.

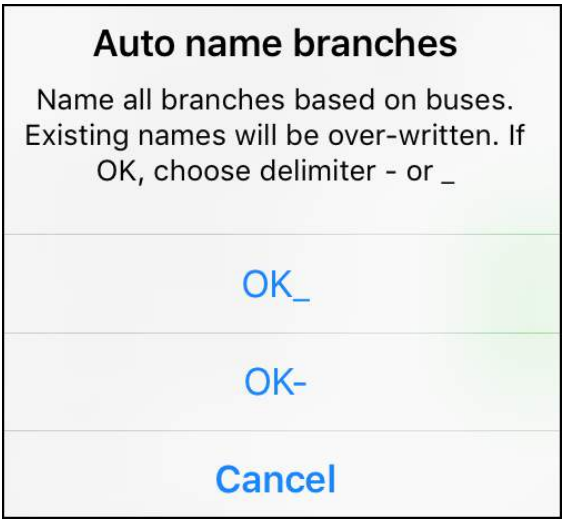


Figure 174: Alert to confirm auto-naming of branches

Settings: Reset Default Values

When a component is added to the model its parameters are populated using default values. As

described previously, some of these defaults can be updated using the eye-dropper button on the Data Displays.

The worked examples in this document mostly use the default values that were in place when the app was run for the very first time. If you have changed the default values and you want to reset them to the original default values (to make it easier to follow the worked examples) then use the "Reset Default Values" button.

Settings: Show bids and offers

The details of the bids and offers (price, quantity and cleared quantity) are viewed via the Data Displays of the Gen and Load components. The details can also be displayed on the network model, as demonstrated in Figure 175, by selecting the "Show Bids and Offers" option to ON.

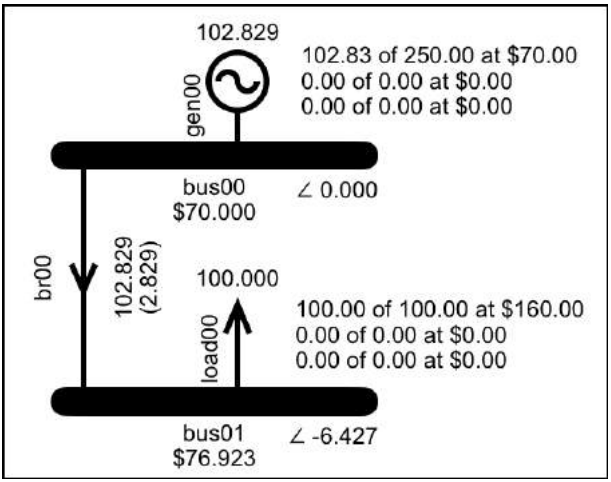


Figure 175: The model with bid and offer details displayed

The advantage of this is that it makes it easy to explain the results without needing to go to other displays, the disadvantage is that it takes up a lot of screen space.

Settings: Show Branch Name

You can save screen space by hiding the branch name, see Figure 176 (branch name displayed) compared with Figure 177 (branch name hidden).

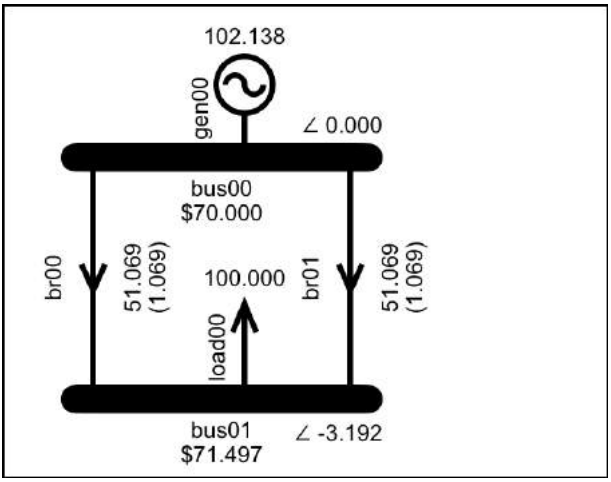


Figure 176: Branch name shown

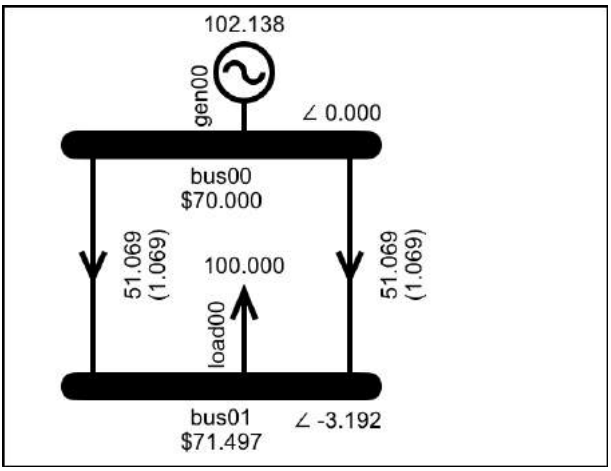


Figure 177: Branch name not shown

Settings: Show Phase Angle

Another way of freeing up screen real-estate is by removing the bus phase angle from the display, see Figure 178 compared with Figure 179.

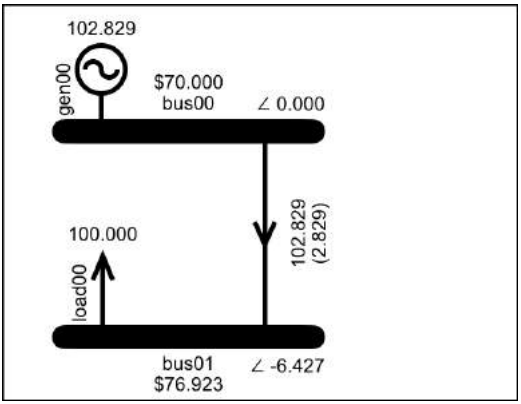


Figure 178: Show bus phase angle = ON

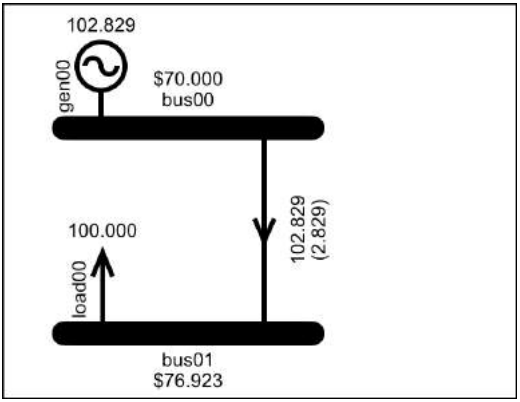


Figure 179: Show bus phase angle = OFF

The bus phase angle can still be viewed on the bus Data Display, and on the branch Data Display which shows the phase angle of the branch's from-bus and to-bus.

Settings: Set Colours

The Settings display also includes the Set Colours option that leads to the Set Colours display shown in Figure 180 where you can set the background colour and the island colours.

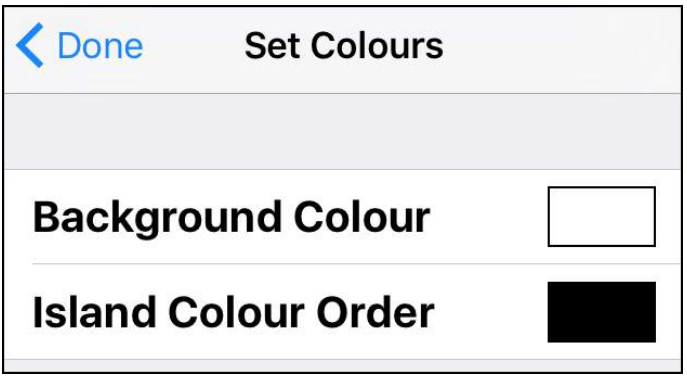


Figure 180: The Set Colours display

The "Background Colour" option leads to the display shown in Figure 181. An example of what happens if you change the background colour is shown in Figure 182.

Figure 181: Background Colour options

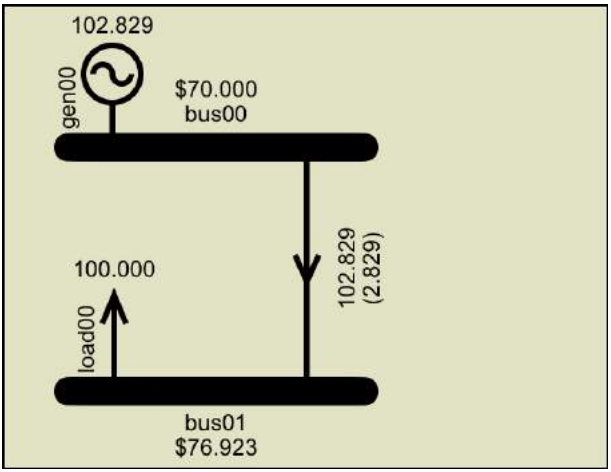


Figure 182: Example of changing the background colour

The "Island Colour Order" option takes you to the display shown in Figure 183 where you can drag the rows by their right hand edge to set the order in which the island colours are assigned. Note that the colour for Island 0 cannot be re-ordered.

Figure 183: Drag the row by its RHS to re-order the island colours. Note that Island 0 cannot be re-ordered.

Settings: Zoom and Scroll

On the iPad the model is always full size so there is no zoom, while scroll is available in landscape mode. Hence, the iPad does not have "Zoom and Scroll" as an option.

On the *iPhone* if "Zoom and Scroll" is on then you can zoom using the pinch gesture, and scroll by dragging the background. If a model will fit on the screen without zooming then it is easier to work with "Zoom and Scroll" selected off.

Saving and Loading a Model

Figure 184: The Save/Load button

The display for saving and loading models is accessed via the folder button indicated in Figure 184. This links to the Models display shown in Figure 185, which lists the names of the models that you have saved. It also has a link to the Save models display (to save the current model) and, in the lower toolbar, a link to the Sample models display (to view and load sample models).

The list of models can be sorted by either by name or by date, using the sort-order button in the lower toolbar.

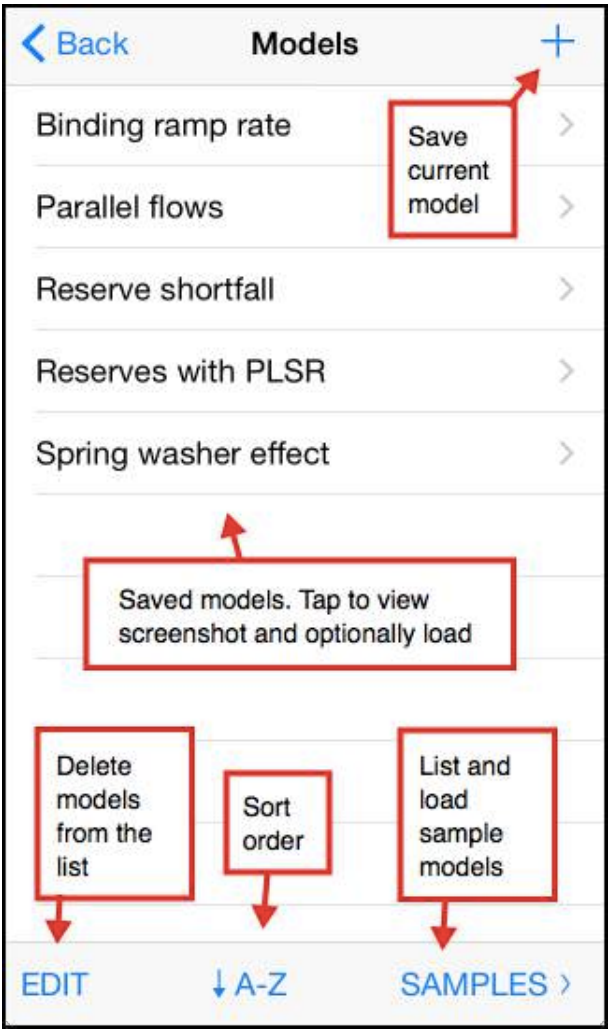


Figure 185: The save/load Models display

Load a model

Tapping a model's name on the Models display takes you to the model details display in Figure 186.

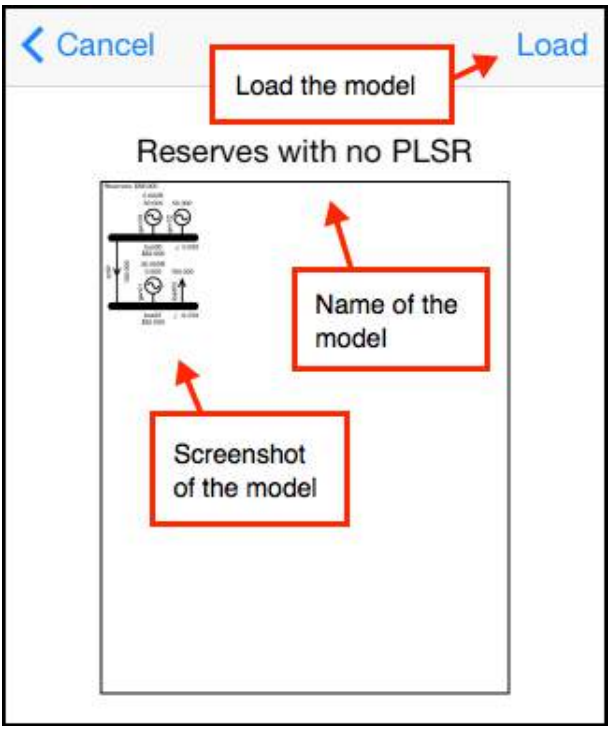


Figure 186: View and Load a model

Tapping the "Load" button will load the model as the current model. If there is an existing current model then the alert shown in Figure 187 will be raised to warn you that loading the model will over-write the existing current model.

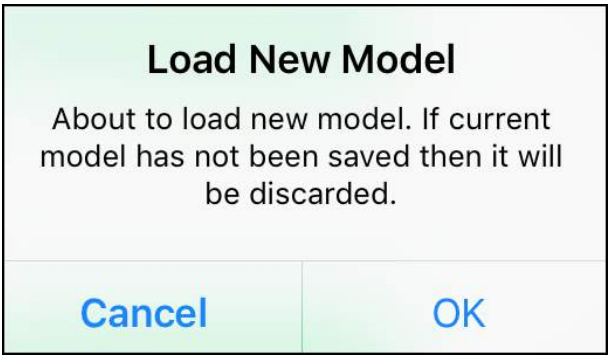


Figure 187: Alert raised if current model exists

Save a model

To save the current model, tap the "+" button on the Models display. This will take you to the Save display shown in Figure 188 where you name and save the model.

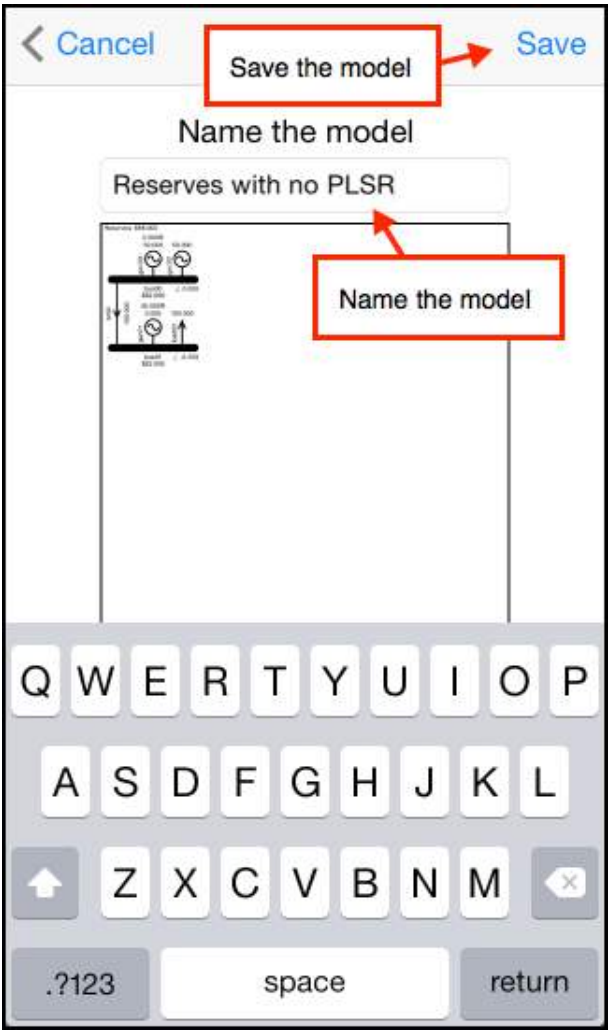


Figure 188: Name and save the model

Delete a model

To delete a model, tap the Edit button on the bottom toolbar of the Models display. This will provide the option of deleting any of the saved models.

Load a sample model

The app includes pre-built sample models. The samples are accessed via the Samples button on the lower toolbar.

The Import Export display

The current model can be exported as a text file. This text file representation of the model can also be imported into the app either as an email attachment or via iTunes.

The Import Export display is accessed via the Import Export button on the Results toolbar, as indicated in Figure 189.

Figure 189: The "Import Export" button on the Results toolbar

The Import Export display is shown in Figure 190. The available options are described in the following sections.

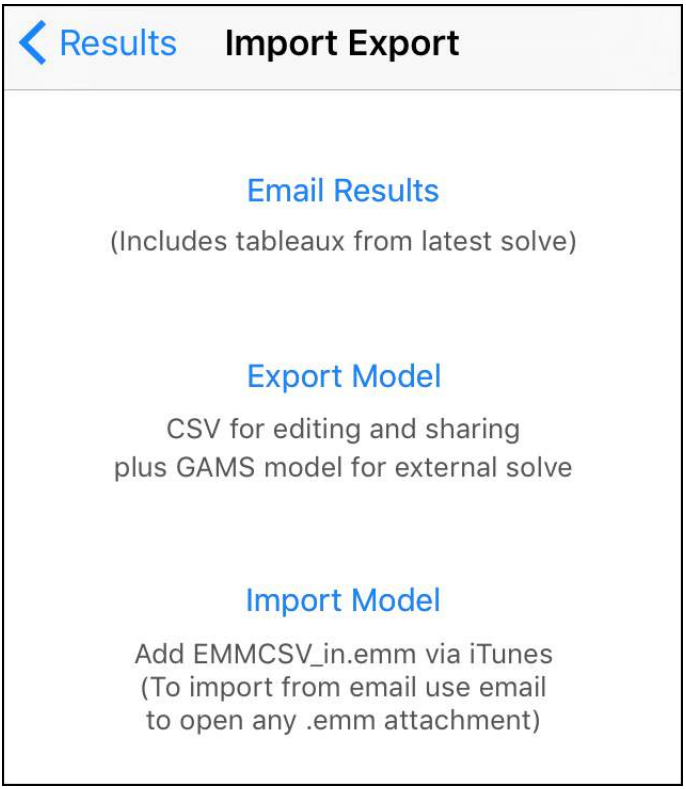


Figure 190: The "Import Export" display

Email Results

Creates an email with the following attachments:

- A screenshot of the current model
- The results of the latest solve, in csv format
- The simplex tableaux from the latest solve, in csv format

Note that the tableaux are only saved if the option to "Save Tableaux" was selected at solve time. If no tableaux were saved then the Email Results button will include the caption "Latest solve did not save tableaux".

As you build bigger models, the associated simplex tableaux get larger quite quickly. If the tableaux are saved then this will slow the solve and strain the memory, so it is best not to save tableaux unless you want to study the simplex solve in detail.

The simplex tableaux are explained in Tutorial 9: Simplex Explained.

Export Model

Creates an email with the following attachments:

- A screenshot of the current model
- A csv representation of the model. This file has a ".emm" extension (electricity market model), and can be imported back into the app (see below for details)
- A GAMS representation of the model. This file has a ".gms" extension and can be solved using the GAMS solver

Solving the model using GAMS

You can download GAMS from…

http://www.gams.com/download/

…install it on your computer and then use it to solve the GAMS model that you exported.

At the time of writing, the free version of GAMS will allow you to solve a model that has up to 300 constraints and variables. This is enough to recreate the results from any of the worked examples in the tutorials, except for the sample models of an actual market which are too large.

Note that HVDC links were added in version 2 of the app and have not yet been added to the GAMS model (if there is any interest in this then it can be suggested as an enhancement).

Confirming the GAMS result

Confirm the GAMS result by loading and solving the sample model "Losses Reserves Ramp". This model exercises most of the market system features implemented by the app.

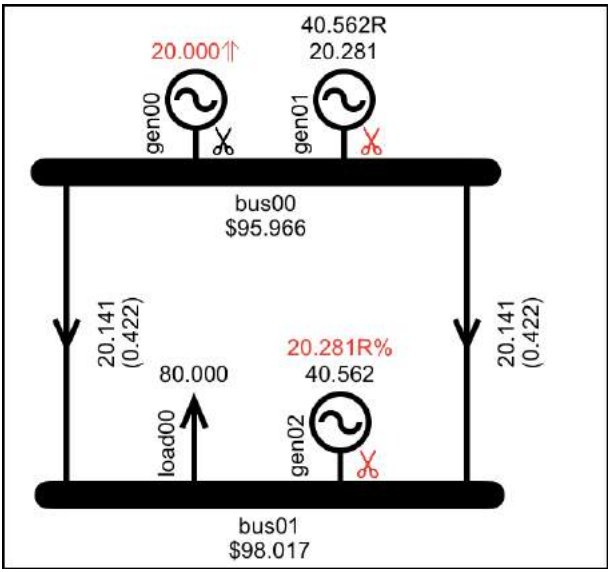


Figure 191: Sample model "Losses Reserves Ramp"

Use the Results - Export Model option to email a copy of the gms file to your computer. Solve the gms file in GAMS.

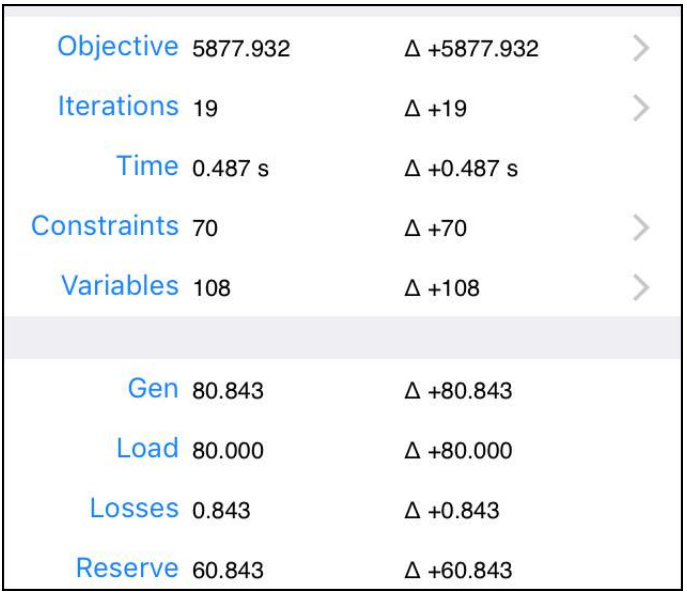
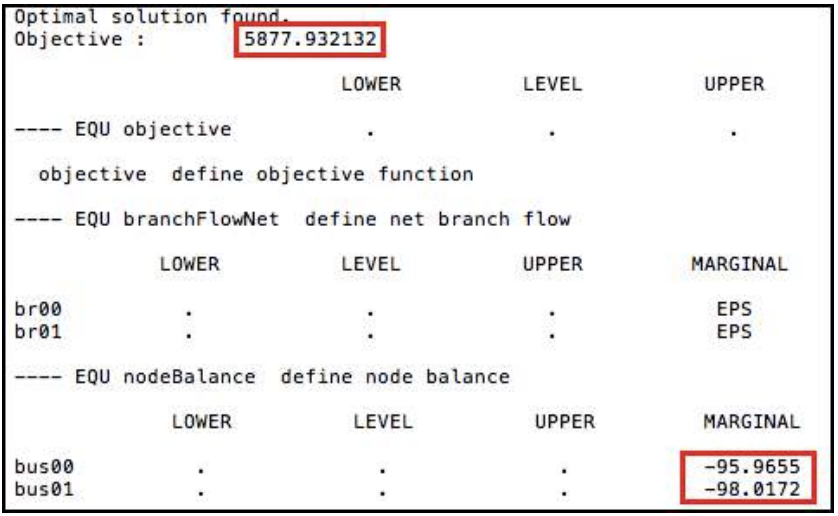


Figure 192: Results for "Losses Reserves Ramp" model

Confirm that the GAMS output, a snippet of which is shown in Figure 193, matches the app results in Figure 191 and Figure 192.



```
Optimal solution found.
Objective :        5877.932132

                            LOWER          LEVEL          UPPER

---- EQU objective             .              .              .

  objective  define objective function

---- EQU branchFlowNet  define net branch flow

            LOWER          LEVEL          UPPER         MARGINAL

br00          .              .              .             EPS
br01          .              .              .             EPS

---- EQU nodeBalance  define node balance

            LOWER          LEVEL          UPPER         MARGINAL

bus00         .              .              .           -95.9655
bus01         .              .              .           -98.0172
```

Figure 193: Snippet of GAMS result for "Losses Reserves Ramp"

Importing a model

Once the model has been exported as a csv file, you can edit the model via the csv, e.g., to easily update multiple inputs, or share it with someone else who has the app. The csv model file can be re-imported by the following two methods.

If the model is sent as an email attachment with the ".emm" extension then the email app on the iPhone/iPod/iPad will recognize the file type and provide the option of loading the model into the Simplex Nodal app.

The model can also be imported using iTunes. In iTunes, select your iPhone/iPod/iPad. In the

Settings side panel select Apps, then in the right-hand side panel scroll down to the File Sharing section, shown in Figure 194

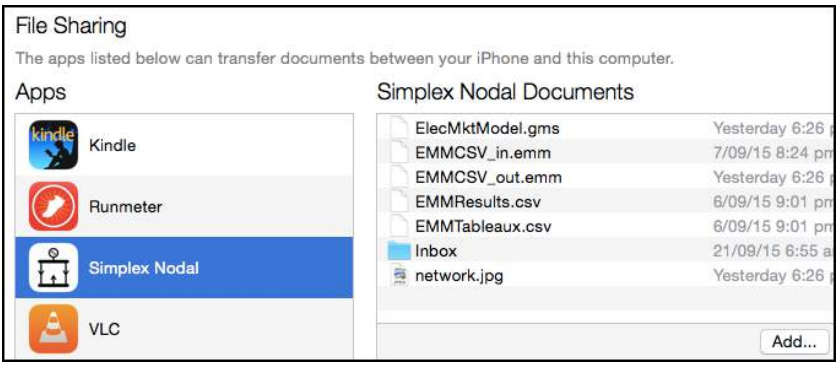


Figure 194: Importing a model using iTunes

Select the "Simplex Nodal" app and click the "Add" button then select the model file from your computer's file system. The file must be named *EMMCSV_in.emm* This file name is also specified on the Import Export display.

Once the file has been added via iTunes, it can be imported to the app via the "Import Model" button on the Import/Export display.

When a model is imported, if the app has a model currently loaded then an alert will be raised.

Solve Settings

The Solve Settings display shown in Figure 196 is accessed via the Solve button indicated in Figure 195.

Figure 195: The Solve button accesses the Solve Settings display

The Solve Settings display is where the solve settings are selected and also where the solve is initiated, via the "Solve Now" button.

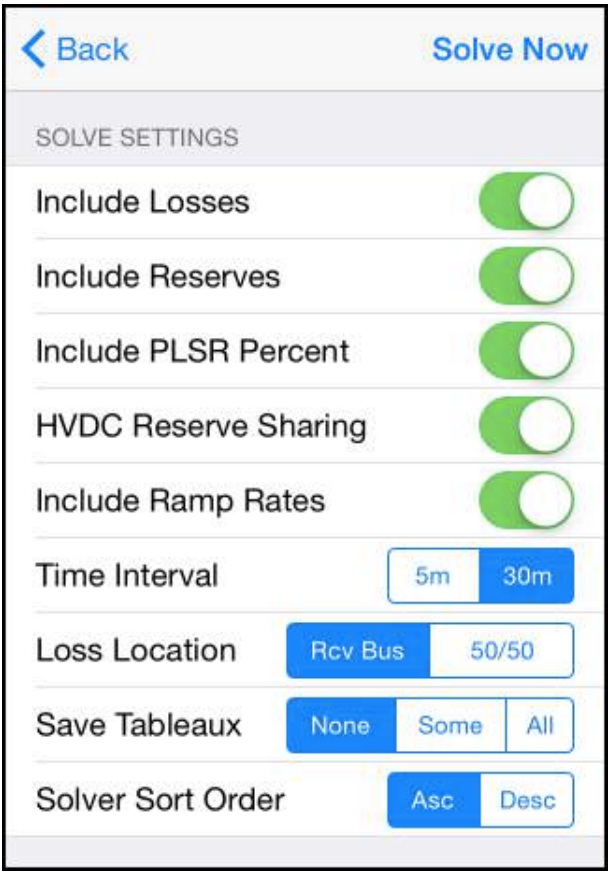


Figure 196: The Solve Settings display

The Solve Settings are as follows:

Setting	Determines
Include Losses	Whether or not the model includes branch loss constraints. See Tutorial 4: Transmission Losses.

Include Reserves	Whether or not the model includes reserve constraints. See Tutorial 5: Risk and Reserve.
Include PLSR%	Whether or not the model includes PLSR% constraints. Only available if the model includes reserves. See Tutorial 5: Risk and Reserve.
HVDC Reserve Sharing	Whether or not HVDC links are capable of sharing reserve between islands. Only available if the model includes reserves. See Tutorial 7: HVDC Link
Include Ramp Rates	Whether or not the model includes ramp rate constraints. See Tutorial 6: Ramp Rates.
Time Interval	The time interval used by the ramp rate constraints. Only available if the model includes ramp rate constraints. See Tutorial 6: Ramp Rates.
Loss Location	Determines whether dynamic line losses are assigned to the bus at the receiving end of the branch, or half of the losses are assigned to the buses at each end of the branch. Only available if the model includes line losses. The New Zealand electricity market assigns dynamic losses to the receiving end of the branch, while

	the Singapore electricity market assigns them half at each end. The receiving end option is the value assumed by the tutorial models, if the 50/50 option is selected then the selection is highlighted orange to indicate it is not the usual option.
Save Tableaux	Whether or not the solver saves the simplex tableaux. Options are "None", "Some" or "All". The "Some" option will save the first, second, and last tableaux. If the model is anything other than very small then the "Some" or "All" options will cause a warning to be raised due to the potential size of the tableaux (and its impact on memory and CPU), and the selection will be highlighted orange. See Tutorial 9: Simplex Explained for an explanation of the simplex tableaux.
Solver Sort Order	Determines the sort order of the constraints and variables in the solver's initial tableau. The initial tableau is explained in Tutorial 9: Simplex Explained. There is a demonstration of the impact of the sort order in Tutorial 8: Actual Market Data.

The Results display

The Results display, shown in Figure 198, is accessed via the Results button shown in Figure 197.

Figure 197: The Results button accesses the Results display

The toolbar on the Results display includes the Import Export button, for importing and exporting model data and results, as described in the Import Export section.

The Results display itself was described in the Using the Simplex Nodal App section. Here we will briefly describe the displays that are accessed via the Results display.

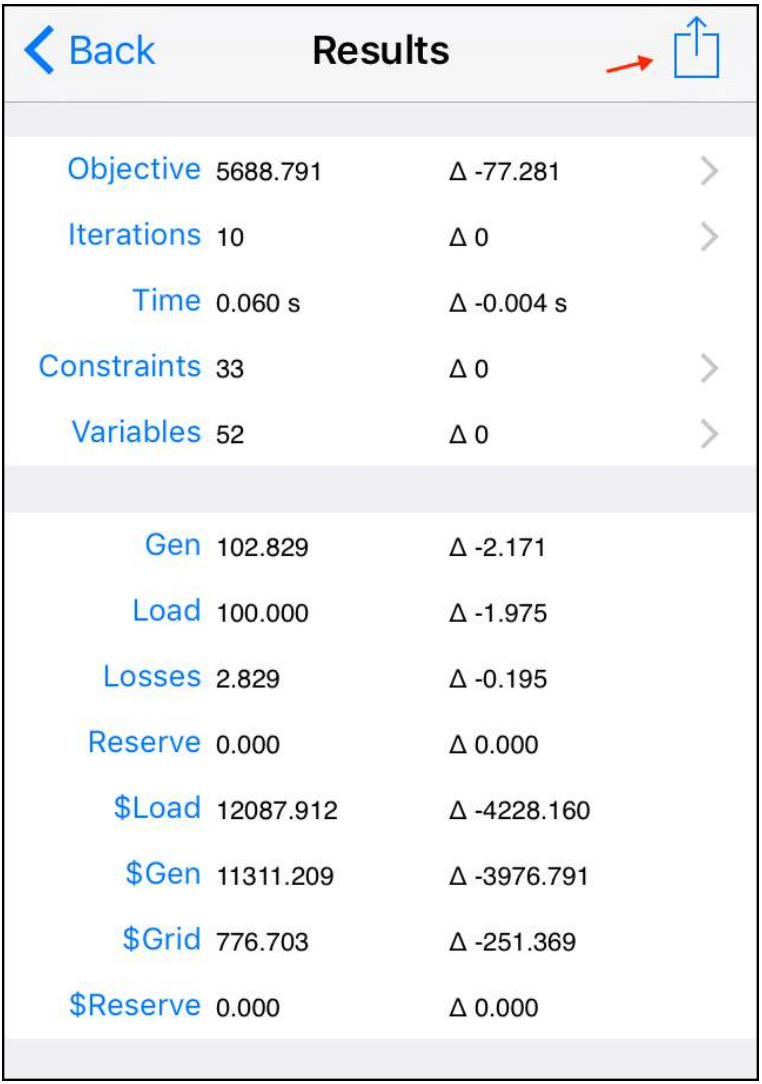


Figure 198: The Results display, with Import/Export button indicated

Objective display

The Objective display in Figure 199 is accessed via the Objective row of the Results display.

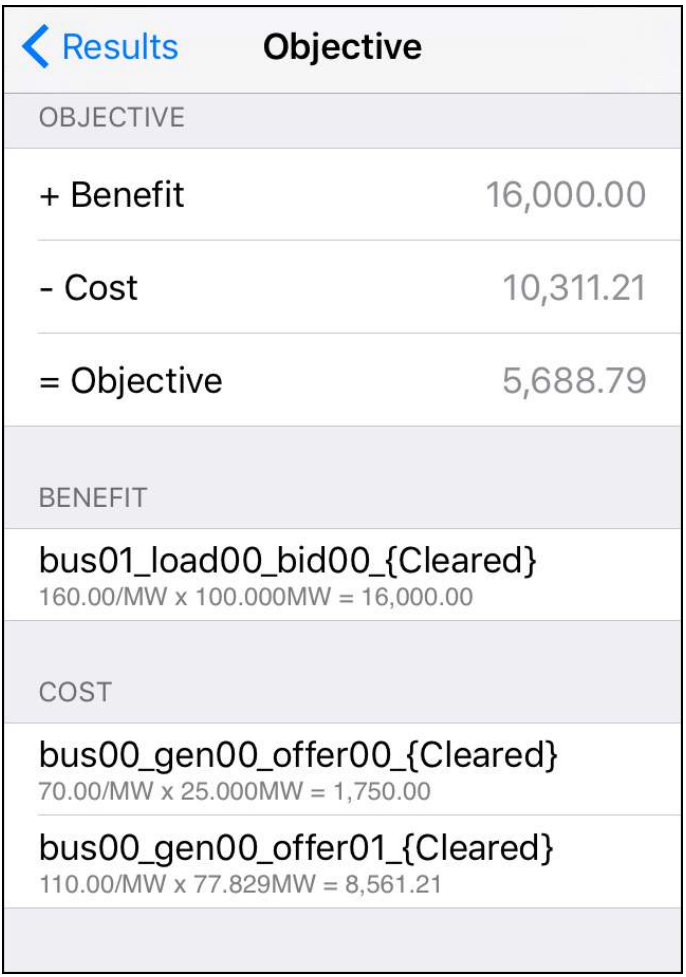


Figure 199: The Objective display

The Objective display shows the objective value calculation, i.e., Benefit – Cost, followed by the details of the individual benefits (cleared bids) and costs (cleared offers).

Iterations display

The Iterations display, shown in Figure 200, is accessed via the Iterations row of the Results display. It provides details of each of the iterations that the simplex algorithm performed.

The details of how and why the simplex algorithm iterates are explained in Tutorial 9: Simplex Explained.

In the list of iterations, iteration 1 describes the step from the first simplex tableau to the second, and so on. Hence, if the simplex algorithm uses three tableaux there will be two iterations. The tableaux themselves can be exported as a csv file, as described in the Import Export section.

The toolbar of the iterations display includes a Chart button which links to the iterations chart described below.

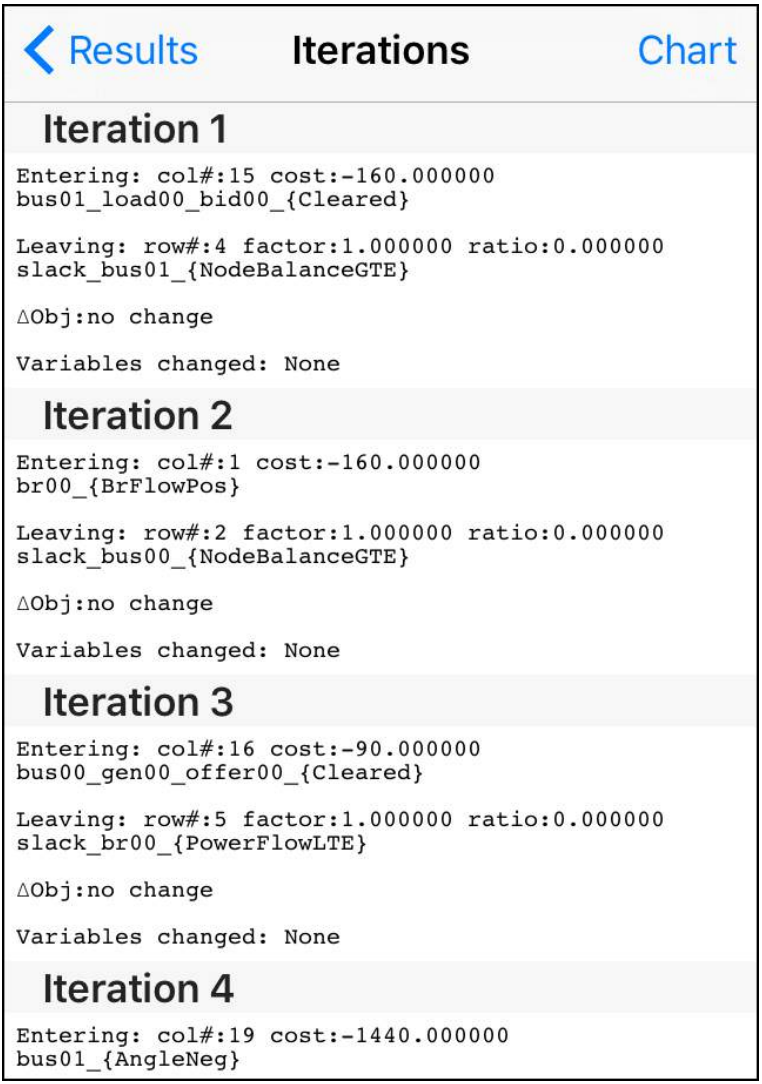


Figure 200: The iterations display

Iterations Chart

The Iterations Chart shown in Figure 201 is accessed via the Chart button on the Iterations display. The chart displays the individual objective values that the simplex algorithm passed through as it iterated towards the optimal objective value.

You can zoom in on the chart using the pinch gesture to see the points that represent the individual iterations, as shown in Figure 202. Tapping on a point will display the details of the iteration, as shown in Figure 203.

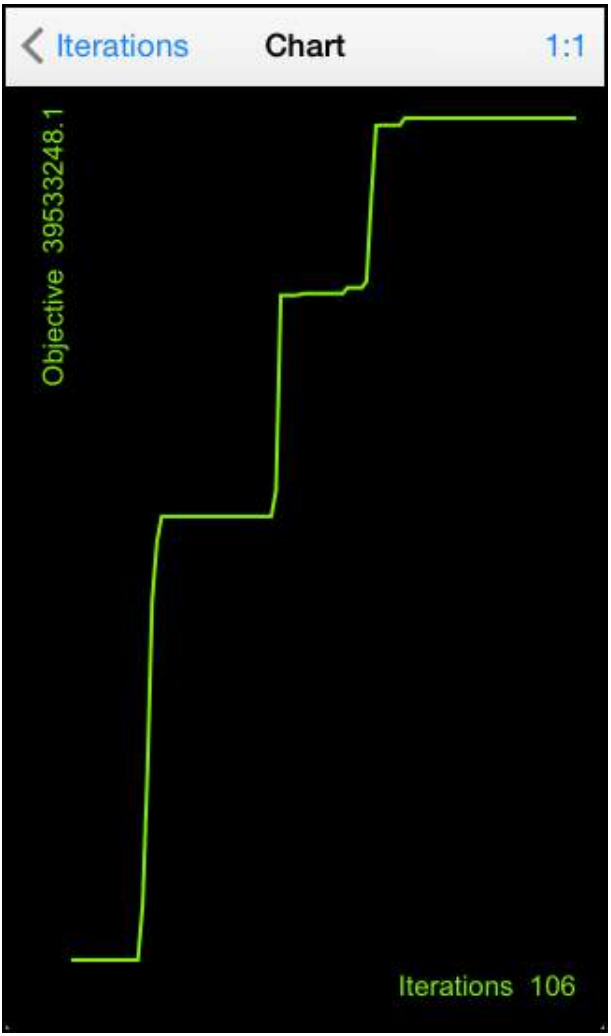


Figure 201: The iterations chart at 1:1 zoom

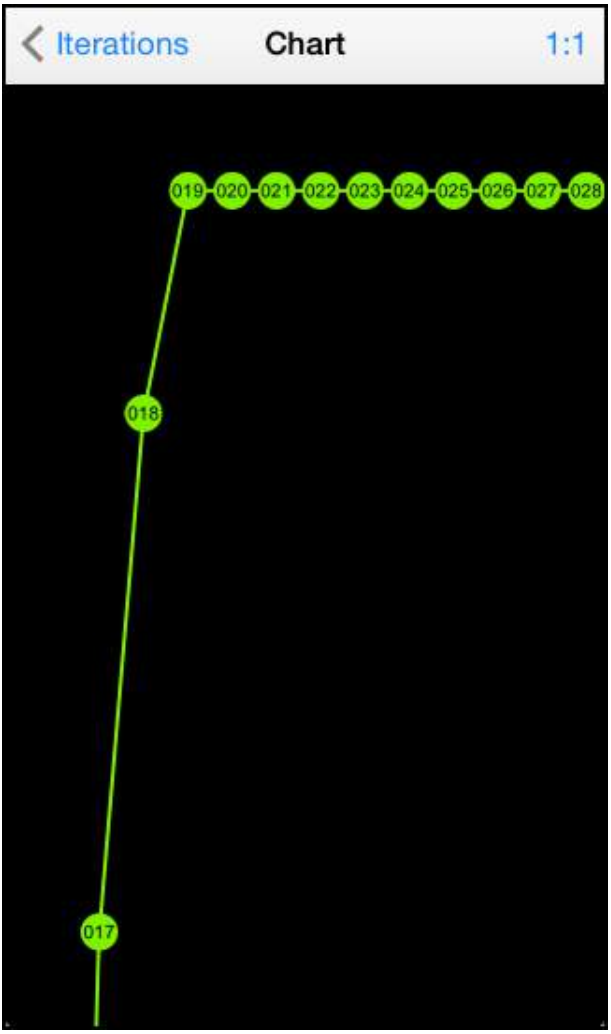


Figure 202: Zoom in to see individual points

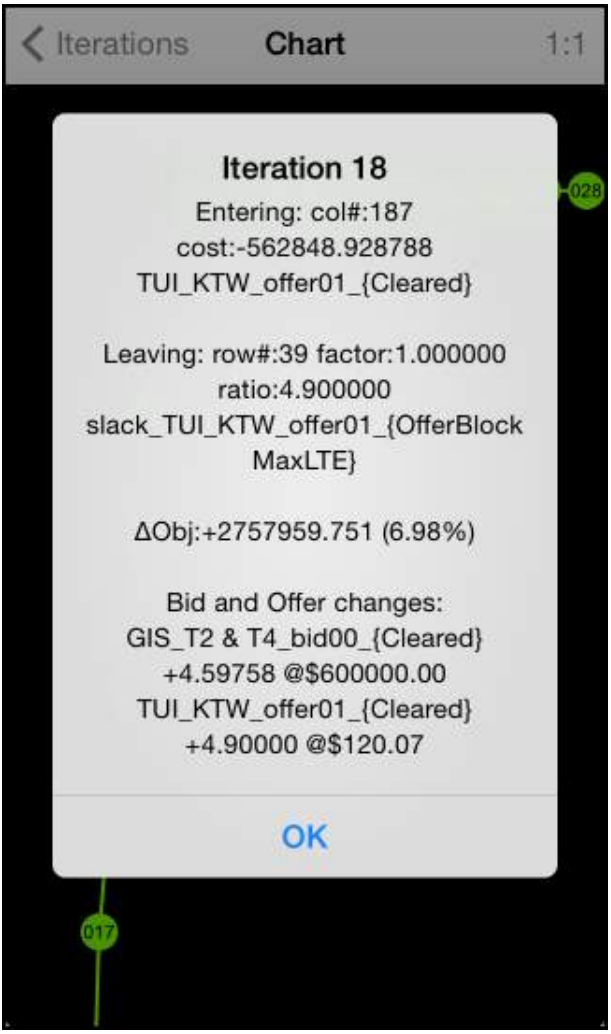


Figure 203: Tap an individual point to view the iteration details

Constraints display

The Constraints display in Figure 204 is accessed via the Constraints row of the Results display.

Figure 204: The Constraints display

The Constraints display lists the constraints for all the components in the model, together with the associated shadow price.

The shadow price is introduced in Tutorial 1: Explaining Prices and explained in detail in Tutorial 9: Simplex Explained.

(The constraints for an individual component can also be viewed via the ∑ button on the component's Data Display).

Variables display

The Variables display, shown in Figure 205, is accessed via the Variables row of the Results display. The Variables display lists all the variables from the latest solve and their associated values. It also displays whether the variable is basic or non-basic. The difference between basic and non-basic variables is explained in Tutorial 9: Simplex Explained.

(The variables for an individual component can also be viewed via the ∑ button on the component's Data Display).

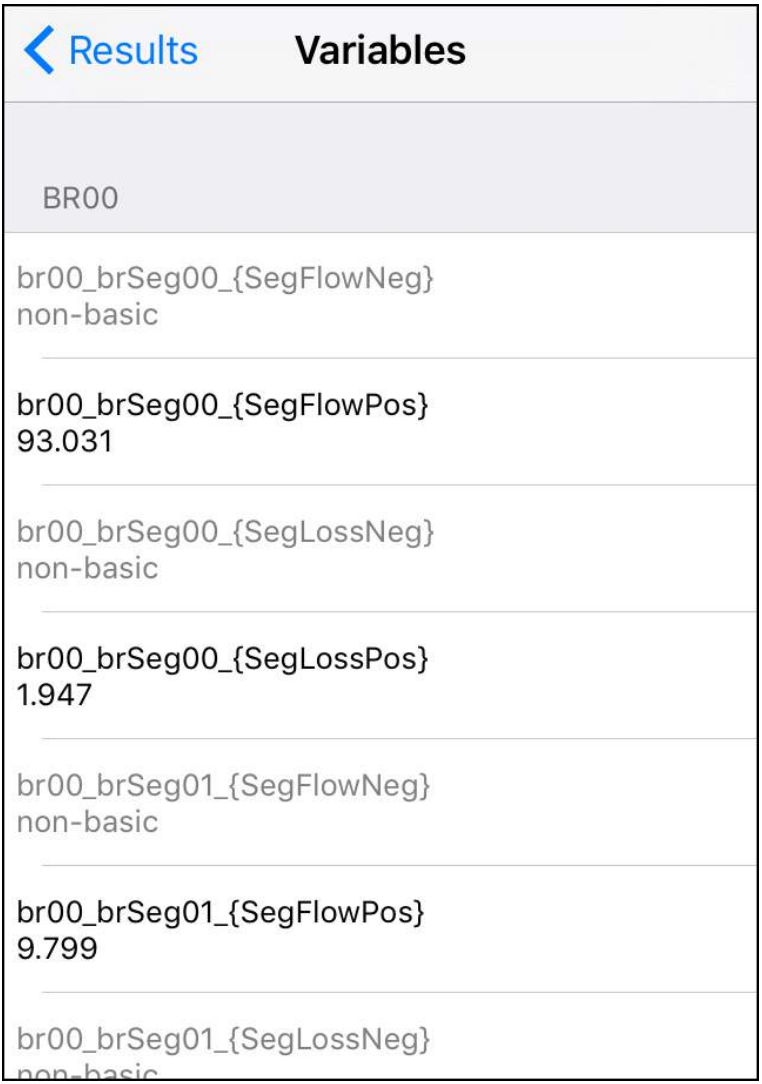


Figure 205: The Variables display