Using Python/Pyomo to Undertake CGE Modelling: An Application to Forward-looking Expectations and Electricity Market Policy Changes

Bruce Layman

Chief Economist

Economic Regulation Authority of Western Australia[*]

Abstract

Python is an open-source, object-orientated interpreted programming language, while Pyomo is a Python optimisation modelling package that can be used to solve Computable General Equilibrium (CGE) models in the tradition of solving GTAP with GAMS software. This paper finds that while Python/Pyomo lacks the ease of use of GEMPACK for CGE modelling, its flexibility allows the implementation of assumptions different to standard CGE treatments. In particular it uses Python and Pyomo to construct: a simple stylised recursive-dynamic CGE model with forward-looking expectations in response to a known but staged policy change; and a simple stylised short-run comparative-static CGE model with an integrated electricity market trading-interval dispatch model. Its preliminary results indicate that forward-looking expectations lead to a more immediate investment response to a known staged policy change than comparative-static expectations, while the economic impact of increasing the supply of intermittent non-storable renewable energy production depends on how strongly this production correlates with electricity demand.

*The views presented herein are those of the author and should not be taken as reflecting the views of the Authority, individual members of the Authority, the Authority's Secretariat, or members of other organisations. The Author acknowledges the time and support from the Authority to undertake this paper.

Contents

Introduction **1**

Python/Pyomo **2**

Python 2

Pyomo 5

Mathematical Optimisation Techniques **6**

Linear Programming 6

CGE Modelling using LP Techniques 10

Application 1: A Simple Dynamic CGE **11**

Model description 11

Dynamic CGE Modelling and Expectations 12

Policy Shock and Results 15

Application 2: An Integrated CGE Model and a Detailed Electricity Generation Model **17**

Background 17

The Models 18

Link between Models and the Policy Shock 20

Results 22

Discussion **23**

References **25**

Introduction

Since its introduction in 1984, GEMPACK (Horridge and Pearson, 2011) has facilitated some of Australia’s most influential policy advice through the Centre of Policy Studies (CoPS) suite of models. It is also used in over 400 other locations in 60 countries. This is likely to continue in the future and for good reason.

GEMPACK allows modellers to solve large systems of simultaneous equations and, once some basic intuitive GEMPACK coding language is learned, it frees modellers to concentrate on the economics of a problem without worrying about computational issues or onerous data input and output considerations.

Additionally, the CoPS suite of models from ORANI (Horridge, 2014) to the Victoria University Regional Model (VURM, formerly MMRF, Peter, Horridge, Meagher, Naqvi and Parmenter, 1996) has allowed researcher to ask policy questions, perhaps with some slight modifications to the base model, without having to construct a Computable General Equilibrium (CGE) model and database from scratch. The author has used MMRF in particular to analyse a number of policy questions related to the Western Australian economy.

This study is conducted to undertake the somewhat eccentric task of CGE modelling outside of the CoPS/GEMPACK paradigm, although the basis for the modelling is firmly rooted in the CoPS linearised CGE tradition.

It is motivated by a desire to test the Python programming language and its Pyomo package, which can already be used across a range of uses, to another purpose. The attraction of a language with a wide range of uses is that, even if it is not the best for any single purpose, it potentially enables researchers to learn how to use only one language to undertake a range of tasks. This reduces the fixed cost of learning the background language for each task.

Another potential benefit from using Python for CGE modelling is that there is flexibility to go beyond the boundaries of GEMPACK’s limits, although this increased flexibility comes with increased risk of model failure and data handling errors. Hence this study attempts to solve CGE models in ways that in the author’s knowledge have not been used before.

This paper constructs simple stylised models to illustrate: a recursive dynamic model with forward looking expectations to deal with a policy change that is phased in over time; and a comparative-static model that is directly integrated with an electricity market dispatch model in the same Python file.

Both of these simulations are based on stylised or toy models with fictitious databases. This paper does not claim that the results hold beyond the models presented, either in terms of Python/Pyomo solving large scale CGE models or that the results presented have any practical policy implications. They do, however, suggest that further research in these areas could be useful.

This paper has been prepared by Bruce Layman, Chief Economist, in the Economic Regulation Authority (ERA) of Western Australia Secretariat. The views presented herein are those of the author and should not be taken as reflecting the views of the Authority, individual members of the Authority, the Authority’s Secretariat, or members of other organisations.

Python/Pyomo

Python

Description

Python (Chan, 2014) is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.[1] It has been used for many different software products and applications in the field of science and finance.[2]

Python is open source, meaning that it is free to use, even for commercial applications. It is managed by the Python Software Foundation[3]. However, there are commercial products leveraged off Python, while additional commercial software is sometimes required to run Python packages.

It is an object-orientated programming language, meaning that it focuses on the data or objects being analysed, rather than on the logic required to manipulate them as in procedural languages. Fortran, on which GEMPACK is based (Horridge and Pearson, 2011), is a procedural language.

It is an interpreted language, meaning its instructions are implemented directly rather than having to compile functions into machine-language instructions. This means that the program is slower to run, but allows the code to be more intuitive than for a compiled language. This is increasingly an advantage in a world where computer processing power is increasing and becoming cheaper every year.

Python is run on a text editor or inside an 'Integrated Development Environment' (IDE), which provides additional facilities for programmers. There are many popular Python IDEs. The calculations in this paper were constructed in the Anaconda[4] environment, which has many mathematical and scientific packages preinstalled.

Python may be run interactively in an IDE, or saved as a Python (.py) file, from which large blocks of code can be run.

Python has a large number of additional 'packages' that can be installed and called in a Python script. Such packages are sometimes difficult to install, but allow users to call intuitive functions to perform common and many less common tasks. Consequently, users do not need to be programmers with an extensive knowledge of Python programming concepts, but simply informed users of the packages that can be called to perform complex data manipulation tasks.

Python packages must be explicitly imported into Python scripts for which they are needed. For example, to call the 'pandas' package, which organises large datasets into dataframes (an organised spreadsheet-like table), the following Python script is run:

[1] Python for Unix/C Programmers Copyright 1993 Guido van Rossum 1 (1993), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.2023 .

[2] A list of companies that use Python and products written with Python is available at: http://brochure.getpython.info/media/releases/psf-python-brochure-vol.-i-final-download.pdf/view .

[3] Information on the Python software Foundation can be found at: https://www.python.org/psf/ .

[4] Details of the Anaconda Environment can be found at: https://www.continuum.io/downloads .

import pandas as pd

If the user requires a new dataframe using simple data directly inputted their keyboard, the following command could be used:

df = pd.DataFrame({'a': [1,2,3], 'b': [2,3,4]}).

The 'pd' is required to call the package, while 'DataFrame' is a function from within the package. This creates the following dataframe:

	a	b
0	1	2
1	2	3
2	3	4

The data are contained in columns 'a' and 'b', which the leftmost column of cells is the (yet unnamed) index.

In reality, base input data are more likely to be imported from spreadsheets, csv files or database products such as SQL. For example, it is straightforward for Python scripts to access information in a SQL database using the 'pymssql' package. This package executes SQL code written inside the Python script to access the SQL data.

As Python is open source, there are many discussion boards and websites dedicated to Python education and answering programmer questions. In all likelihood, if a user has a problem, some else has solved it and posted it online.

For example, the popular 'Stack Overflow' question and answer website for programmers[5] has over 600,000 questions directly tagged as 'Python', as well as many tens of thousands of questions each dedicated to individual Python topics.

Data Manipulation Example

As an example of data manipulation in Python, the following code retrieves 2015 Western Australian Wholesale Electricity Market (WEM) data of the Australian Energy Market Operator (AEMO) website, creates a dataframe containing only trading interval-by-trading interval system demand in Megawatts (MW)[6], and then charts this information.

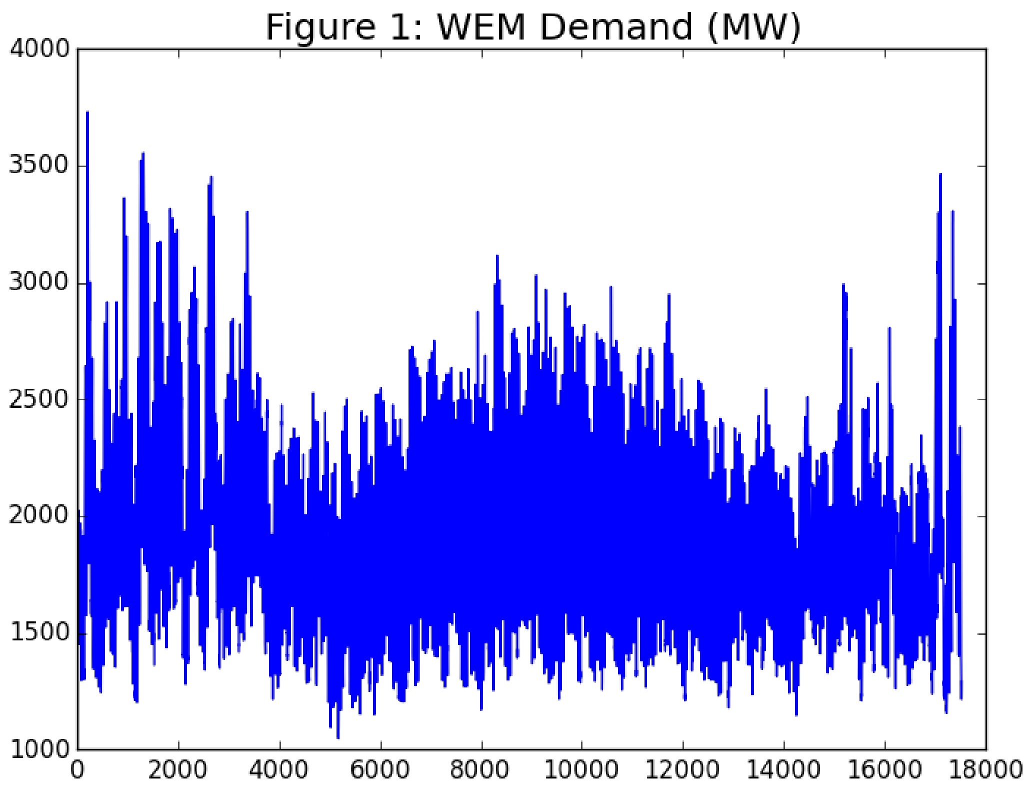
```
import pandas as pd

import matplotlib.pyplot as plt

url =http://data.wa.aemo.com.au/datafiles/balancing-summary/balancing-summary-
2015.csv

table = pd.read_csv(url)

df1=pd.DataFrame(table)
```

[5] This can be found at: http://stackoverflow.com/ . The data are current as at 1 August 2016.

[6] Megawatts (MW) are units of power. Megawatt hours (MWh) are a measure of energy, or work multiplied by time.

```
df2=pd.DataFrame(df1['Scheduled Generation (MW)'])

df2.columns=['MW']

plt.plot(df2['MW'])

plt.title("Figure 1: WEM Demand (MW)", size=18)

plt.ylim(1000, 4000)
```

This generates Figure 1 below.



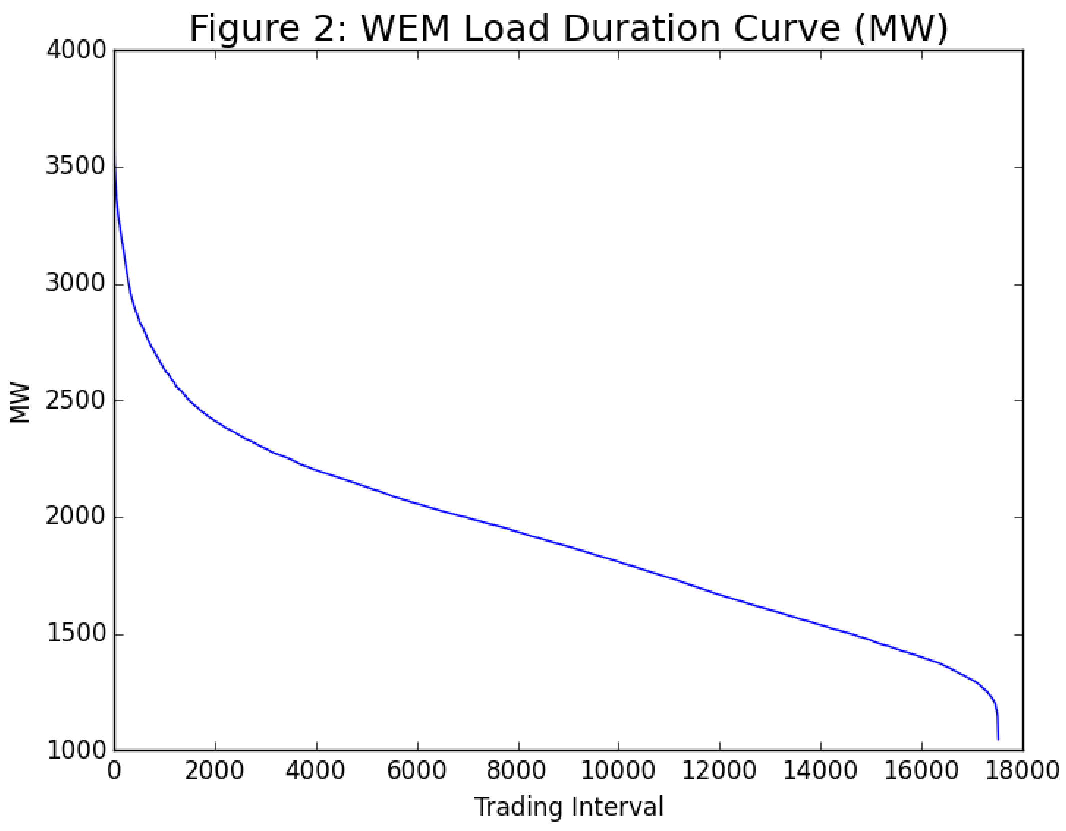
However, if the user were interested in the WEM's load duration curve, with demand ordered from the highest quantity to the lowest quality[7], they could create a new dataframe and re-order these data. Alternatively, if they wished to view the reordered data in a chart but leave the df2 dataframe untouched, and add a chart title and axes labels, the following plot command could be used:

```
plt.plot(df2.sort(['MW'], ascending=[False]))

plt.title("WEM Load Duration Curve (MW)", size=18)

plt.xlabel('Trading Interval', fontsize=12)
```

[7] Examining load duration curves are useful for power system or individual generator planning purposes.

```
        plt.ylabel('MW', fontsize=12)
```

This code creates Figure 2 below.



Pyomo

Summary

Python Optimising Modelling Objects (Pyomo, Hart, Laird, Watson and Woodruff, 2012) is a Python package that 'supports the formulation and analysis of complex optimisation applications'. A simple optimisation problem is described in the next section.

While Python is an object-orientated language[8] and Fortran is a procedural language, in practice there is very little difference to a modeller in terms of how GEMPACK and Pyomo operate. The type of model used in the main simulations in this paper is an abstract model, which means that the code constructed can be used on different datasets.

Sets, parameters and variables have a similar meaning to the same terms in GEMPACK. Constraints are Pyomo optimisation functions that play a similar role as GEMPACK Equations. Constraints are further explained in the next section. Calculating initial

[8] In object-orientated programming terms, the Pyomo model type 'AbstractModel' is a Python class and a model constructed with that code is an instance of that class. Hence the term 'model= AbstractModel()' is included in the code near the beginning of the script, and all variables, the objective function and constraints are defined in terms of that instance. For example, in Pyomo a variable 'X" is defined as 'model.X'.

parameters, as is done with formulas in GEMPACK, is done with Parameter Initialisations in Pyomo.

As an example of Pyomo versus GEMPACK code, consider the standard equation for industry demand for labour taken from the Mini-USAGE model (Dixon and Rimmer, 2005). In GEMPACK the equation is:

```
Equation E_x1lab # Industry demands for labour # (All,i,IND)
x1lab(i) - a1lab(i) = x1prim(i) - SIGMA1PRIM(i) * [p1lab(i) - p1prim(i)];
```

In contrast, the Pyomo Constraint is as follows:

```
def E_x1lab (model,i):
return model.x1lab[i] == model.x1prim[i] -
model.SIGMA1PRIM[i]*(model.p1lab[i]-model.p1prim[i])
model. E_x1lab = Constraint(model.IND, rule= E_x1lab)
```

While there are differences in Syntax, the basic structure of the relationship is clear in both expressions.

Pyomo requires an optimisation solver program to run. The solver's required capabilities, such as linear versus no-linear programming or whether it supports mixed-integer programming or not, depends on the problem to be solved. Solvers range from free open source programs to very expensive commercial applications.

The models in this paper solve quickly using both the free CBC solver[9] and the commercial CPLEX[10] solver, but the models slow dramatically once additional complexity is included in the models. This is especially the case for the electricity dispatch model described later, which slows dramatically on both solvers once additional constraints are included.

Mathematical Optimisation Techniques

Linear Programming

Basics

Optimisation modelling seeks to solve for an 'objective function' relative to a set of constraints. For example, the models in this paper use a subset of optimisation modelling in Linear Programming (LP).

Briefly, LP is an optimisation procedure for linear objectives and constraints. Formally, for a vector of variables of interest x, vectors of parameters C and b and a matric of parameters A, an LP problem can be specified as:

[9] For more information on the CBC solver see: https://projects.coin-or.org/Cbc .

[10] For more information on the CPLEX solver see: https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/ .

Maximise $C^T x$

Subject to: $Ax <= b$

$x >= 0$

$C^T x$ is the objective function, while Ax <= b are the model constraints. The remaining terms sets the lower bound for the solution.

To illustrate the LP problem, consider a simple example for two variables x1 and x2, with the parameters set to their numerical values:

Maximize: model.x1 + 2*model.x2

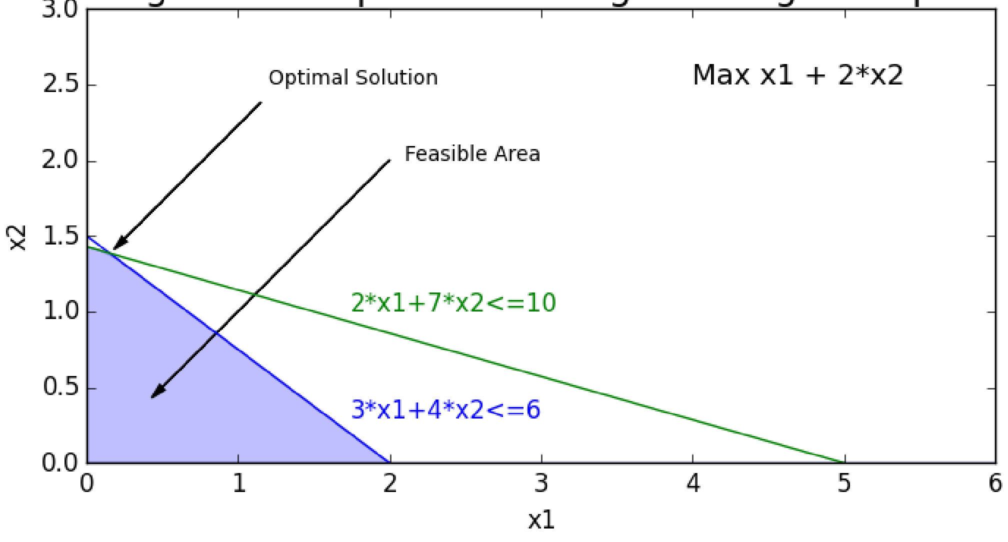
Subject to: x1*3 + x2*4 <= 6

x1*2 + x2*7 <= 10

x1, x2=> 0[11]

This problem is shown graphically in Figure 3 below. The shaded area is the area below both constraint lines and, given both constraint inequalities are less-than-or-equal-to, is the area for which a solution is feasible. The optimal solution in this case is the intersection of the two curves in 0.15384615 for x1 and 1.3846154 for x2.

Figure 3: Simple Linear Programming Example



[11] The solution for this problem is 0.15384615 for C1 and 1.3846154 for C2.

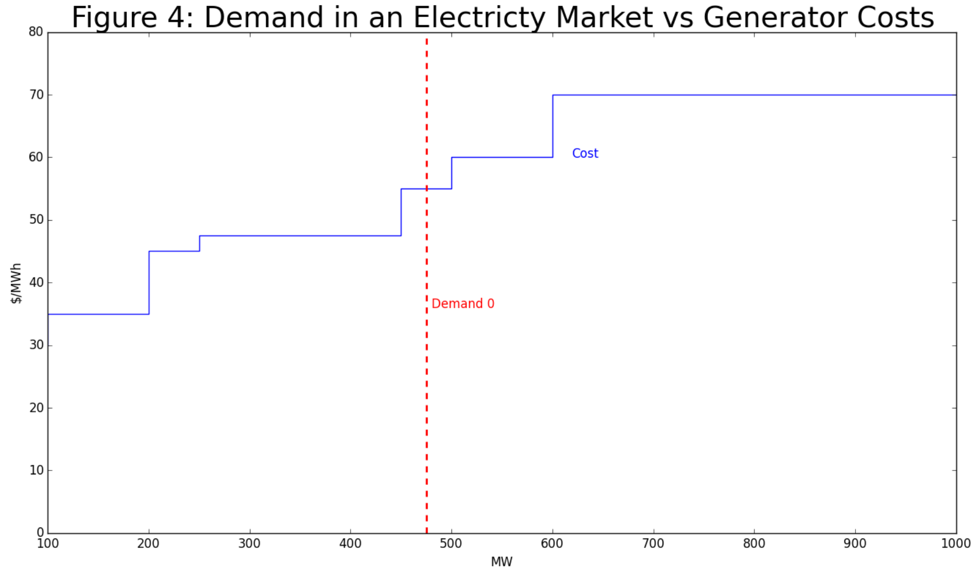
Practical Linear Programming Example

For a more practical example of linear programming, such models are often used to solve electricity market dispatch problems for a central planner or a market with centralised dispatch. Dispatch quantities from each generator in each time interval (variables) are optimised to minimise the total generation cost for the period of interest (the objective), given a known set of generators with known marginal costs and fixed total demand (parameters). This is a short-run process that minimises variable costs.

Figure 4 below shows the cost of electricity for a system as the power production increases. It is a step function, with the cheapest generator able to produce at $30/MWh until it reaches its maximum level of production of 100 MW, before the next cheapest generator produces at $35/MW for the next 100 MW, and so on.

If demand is initially 475 MW, the marginal generator's cost is $55/MWh. If the system were a market, this generator would set the market price. To the left of 475 MW, generator constraints are binding because generators cannot exceed their maximum levels of production. To the right of 475 MW however, higher cost generators are not needed and so the constraints are not binding.

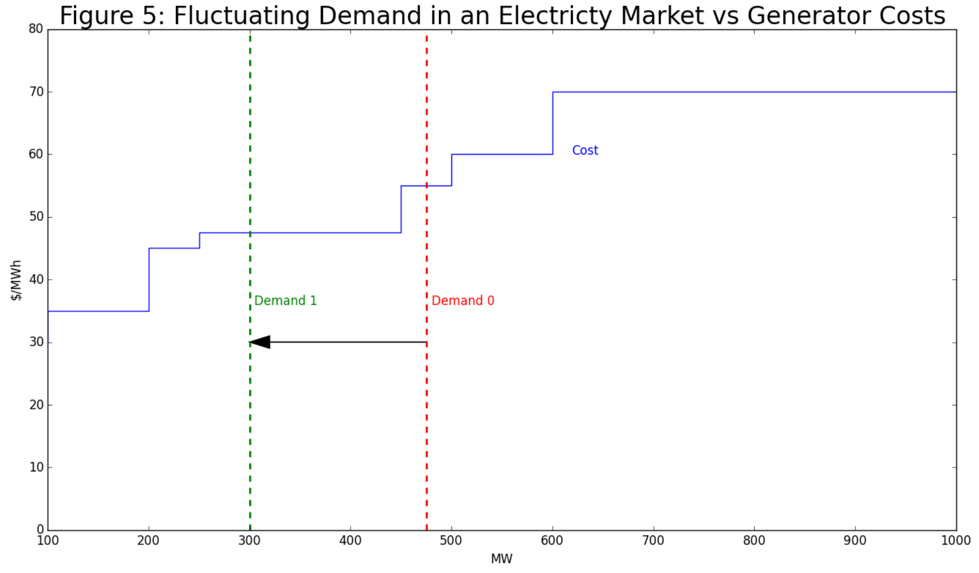
Figure 4: Demand in an Electricty Market vs Generator Costs



The natural way to solve this type of problem of cost minimisation is by LP. The non-binding constraints are solvable because generators can produce less than or equal to their maximum production, rather than only at a set level of production.

While the example explained above is trivial, real-world electricity market problems are not. For example, if demand fell to 300 MW as in Figure 5 below, a different set of constraints become binding.

Figure 5: Fluctuating Demand in an Electricty Market vs Generator Costs



Additionally, successive trading interval cost minimisation problems are not independent because:

- Generators also have minimum production levels, meaning that plants might be shutdown even though notionally there is some demand for their output;

- Generators cost (often substantial amounts) of money to start up and shut down, meaning they may keep running even though the market price is less than their marginal cost, or may not start up if they are needed for only a few trading intervals;

- Technical requirements of each generator may mean that they have a minimum number of trading intervals that they must be run for and a minimum number of trading intervals for which they must be rested once shutdown. These are called minimum-up and minimum down times; and

- Plants have technically safe minimum changes in output over time, or ramp rates.

Finally, the short-run marginal cost of many traditional electricity generators actually slope downwards, as they become more efficient as they move from their minimum generation level to their maximum generation. Consequently, over small changes in demand, costs can go up when demand falls.

These factors mean that, rather than a succession of independent cost minimisation problems, electricity dispatch over a time period is one single problem.[12] The overall time period may be divided up into multiple time periods, but care needs to be taken that initial and terminal conditions for each sub-period do not lead to unrealistic results[13].

[12] This model can be solved by the Mixed-Integer Linear Programming (MILP) technique, which allows binary variables. This is explained later in the electricity dispatch model details.

[13] These can occur in situations where dispatch during trading intervals outside of the sub-sample affects dispatch inside the sub-sample. For example, it might make sense to shut-down a generator towards the end of a sub-sample if only the sub-sample was considered, but it might not make sense if this generator were required to be dispatched early in the next sub-sample.

CGE Modelling using LP Techniques

CGE modelling using optimisation techniques has been successfully conducted through the GAMS optimisation or the MPSGE GAMS subsystem software (Horridge and Pearson, 2011). GAMS has usually been used to solve nonlinear CGE problems such as those in various versions of GTAP (GTAP in GAMS, Rutherford, 2006)

The method used for GAMS is to set the objective function to zero and specify all the constraints as equalities rather than inequalities (Horridge and Pearson, 2011). This is in contrast to GEMPACK which solves an exactly identified set of linear equations.

As an example, consider the previous simple linear programming example. If the constraints are specified as equalities and the objective function is replaced with zero, the problem becomes:

Maximize: 0

Subject to: x1*3 + x2*4 == 6

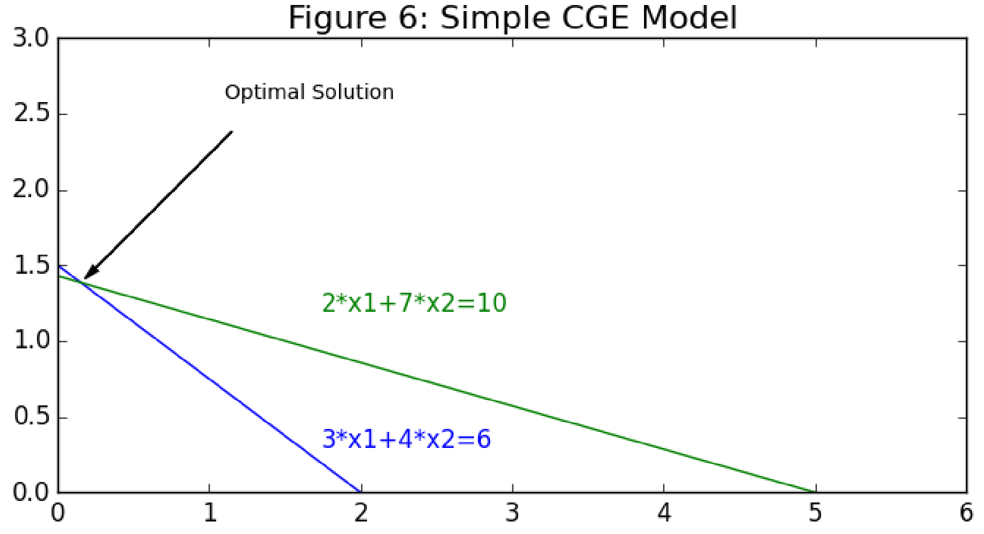
x1*2 + x2*7 == 10

x1, x2=> 0[14]

In this case the solution to the problem is exactly the same as the LP optimisation above. However, rather than this solution being the maximisation within a feasible region, in a CGE model is the only possible solution.

[14] The solution for this problem is 0.15384615 for C1 and 1.3846154 for C2.

Figure 6: Simple CGE Model



The ability for Pyomo to solve system of equations that is not exactly identified can lead to difficulty in determining whether a CGE model has a valid closure. If the variable count is 1-2 variables over or under-identified, the model will still solve but will not give the 'correct' result. More than 1-2 variables from identification will lead to trivial solutions such as all price changes being equal to zero, and at some point the model will not solve.

Application 1: A Simple Dynamic CGE

Model description

To illustrate the concept of forward looking expectations in a CGE model, a simple stylised model is created. The model has the following characteristics:

- It has linearised economic relationships;
- It has one good and one industry;
- It has labour and capital inputs, but no intermediate demands;
- It has one single price for the good, which is the model numeraire. This is discussed further below;
- It has an instantaneous short-run labour market closure;
- It has linearised CES production technology for labour and capital inputs;

- It has no margins, taxes or government expenditure;
- It is a closed economy with no exports or imports. Income in each year is spent on either consumption or investment with no net debt created; and
- It has a standard recursive-dynamic modification, specifically modelled on mini-USAGE model (Dixon and Rimmer, 2005)

The model is calibrated so that it is in a long-term steady-state, with investment in each year exactly equal to depreciation in the database. This removes the need to do separate base and base-rerun simulations, as the alternative to the policy simulation results is zero.

Unlike a model with an export/import module with an associated exchange rate, a simple closed-economy model does not specifically require the creation of an exogenous model price numeraire. However, a numeraire is used in this model as it facilitates the model's steady state.

If the single price in this model is specified as an exogenous numeraire, the model's consumption function is not required as the model's commodity market clearing constraint is equivalent to a nominal income constraint. This reduces the number of equations by one, which accommodates the additional exogenous variable.

Dynamic CGE Modelling and Expectations

Almost all dynamic modelling in the CoPs family of CGE modelling has been of the form of recursive-dynamic modelling, where a succession of comparative-static simulations are run with linking equations for stocks of capital and the database for each year is affected by the updated database from the previous year.

Dixon, Pearson, Picton and Rimmer (2003) note that recursive dynamic modelling has advantages over a fully specified dynamic modelling, where all years are solved simultaneously, as: the ability to solve larger and more detailed models with recursive-dynamic modelling; and higher transparency to the reader. With regard to the second point, even a percentage change model becomes non-linear under a simultaneous model, as the usual model coefficients become variables of the model rather than as input parameters.

Dixon et al (2003) note that the Monash Model static expectations formula is given by:

$$EROR(i,t)^1 = EROR\_SE(i,t)^1$$

Dixon and Rimmer (2005) show that in practical terms, the static expectations rate of return calculation is given by:

$$100 * del\_ror = \left[\left(\frac{1}{1+RINT}\right) * \left[\frac{V1CAP(i)}{VCAP(i)}\right] * (p1cap(i) - p2tot(i))\right]$$

Where:

EROR(i,t) is the rate of return used in the model for industry I in year t;

EROR_SE(i,t) is the static rate of return for industry I in year t. It is equal to 100*del_ror(i) in mini-USAGE;

del_ror is the linear change in the rate of return for industry I;

RINT is the real interest rate;

V1CAP(i) are the gross returns to capital for industry i

VCAP(i) is the value of the capital stock in industry I;

p1cap(i) is the price of capital for industry I; and

p2tot(i) is the price of investment goods to industry 1.

The interpretation of the static returns expectations relationship is that the rate of return considered by investors in year t is the rate of return for year t discounted by one year to reflect the fact that returns from investments in year t will not be received until year t+1.

However, almost all capital created in year t+1 will have a life beyond that year. For example, if the same gross return was received in year t+2 as in t and t+1, it would need to be discounted by the real interest rate twice. Additionally, depreciation from year t+2 onwards would mean that the return would be on successively lower capital unless offsetting capital expenditure was made[15].

Dixon et al. (2003) proposed a method to incorporate rational expectations into recursive dynamic models by iteratively ensuring that expected rate of return in each industry in year t is equal to the actual rate of return in year t+1, plus the residual value of the new capital at the end of year t+1.

Dixon et al. (2003) replace static expectations with one-year forward looking expectations, which they define as ‘rational expectations’.

$$EROR(j,t)^2 = ROR\_ACT(j,t)^2$$

In this equation ROR_ACT is the actual return plus residual value of the capital in year t+1.

$$ROR\_ACT(j,t) \quad = -1 + \left[\frac{1}{1+INT}\right] * \left[\frac{Q(j,t+1)}{PI(j,t)}\right] + \left[\frac{1-D(j)}{1+INT}\right] * \left[\frac{PI(j,t+1)}{PI(j,t)}\right]$$

Firstly, the model is run under static expectations. Then the rate of return in year t is replaced by ROR_ACT(i,t) (the actual return in year t+1) and the model runs again. If investment in year t changes relative to the initial comparative-static simulation, ROR_ACT(i,t) will also change. As such it is required to loop the model until:

$$EROR(j,t)^N = ROR\_ACT(j,t)^N$$

The Dixon et al. (2003) method assumes that the residual value of new capital at the end of year t+1 is reflects only the change in the price of investment between year t and year t+1. Hence the investment is reversible at the end of period t+1. This might be a reasonable expectation for many or most industries, but it might not be appropriate for a specific class of policy shocks where the change is phased in over time and there is reasonable certainty of the path of the change.

Consider an import tariff change or the imposition of a carbon tax that a government announces is to be phased in over time. For example, a government might think that the social cost of carbon is $100/t CO2, but that immediate imposition of such a tax would cause

[15] The rate of return flows directly through to capital growth, not investment.

major costs on the economy. It therefore decides to start the tax at \$20/t in year one, rising by \$5/t per annum until it reaches \$100/t in year 16.

Under static expectations, investors will make decisions about their capital stock in year t+1 onwards based on a carbon tax of \$20/t, while under the Dixon et al method rational expectations method, they will consider \$25/t plus residual value at the end of year t+1.

Both of these methods are likely to understate the impact on investment decisions on (say) a coal-fired power station with an effective life of at least 30 years. Assuming a tax of \$20-\$25/t in an investment calculation is clearly too low for the life of the asset, while returns from year t+2 onwards are likely to be less than the cost-based residual asset value at the end of year t+1. This could have major implications for the pattern of investment in a CGE model policy shock.

Additionally, economic cost of such action on carbon will depend critically on a race between the investment impact of the increase in the carbon tax and the improvements in the cost and practicality of low-carbon technologies. CGE models are likely to underestimate the economic cost of a phased-in carbon tax if their investment responses lag those likely to occur.

This paper uses a similar method to Dixon et al (2003) to calculate the rate of return on an investment from year t until the terminal year of the analysis. This formulation is designed to be comparable to static expectations, except that it allows for deviations in the present value of returns from year t+1 to the terminal year. There is only one industry in the model, so the formula is:

$$\mathrm{ROR\_ACT(t)} = \left[\left(\frac{1}{1+RINT}\right) * \frac{V1CAP(t)}{VCAP(t)} * \frac{\left[\sum_{u=t+1}^{T}\left(Q(u)\right)^{u}\right]}{(T-(t+1))}\right]$$

The variables in this equation are the same as above. For example, if the static expectations rate of return increases from 5% in year t+1 to 6% in year t+2, EROR_ACT(t) would be 5.5% discounted back from year t+1 to year t.

Neither the Dixon el al. (2003) method nor the method suggested in this paper is true rational expectations in terms of maximising consumption over time. Investment is based on an annual approximation the response of investment to changes in rates of returns. True rational expectations require moving to a single solution nonlinear model.

Additionally, the formula is greatly simplified by the adoption of only a single price in the model which is treated as an exogenous numeraire. This means that the formula does not have to account for changing investment prices.

Finally, the formula does not at this stage include a residual asset value for capital remaining at the end of the terminal year of the simulation. This could be improved in any future iterations of this method. Otherwise, the model could simply be run for a number of years after the period of interest ends, so that the present value of capital created during the period of interest is mostly depreciated.

The process to iterate expected and actual returns is as follows:

1. The model is run with static expectations, where EROR(t) = EROR_SE(t)

2. The actual rate of return for each year t to the terminal year is calculated as above (EROR_ACT(t)^0).

3. A vector of coefficients, DIFF, is created. This is the difference between the imposed expected rate of return in the model, versus the actual return once the model is run with investment based on these expectations.

4. A variable, tolerance, is created. This is the modeller accepted maximum error

5. The model is looped using the Python ‘while’ function as follows.

```
While DIFF(t) > tolerance(t):
        Run model with EROR(t) = $EROR\_ACT(t)^1$
        Calculate $EROR\_ACT(t)^2$
        DIFF = abs($EROR\_ACT(t)^1$ - $EROR\_ACT(t)^2$)
        $EROR\_ACT(t)^1$ = $EROR\_ACT(t)^2$
```

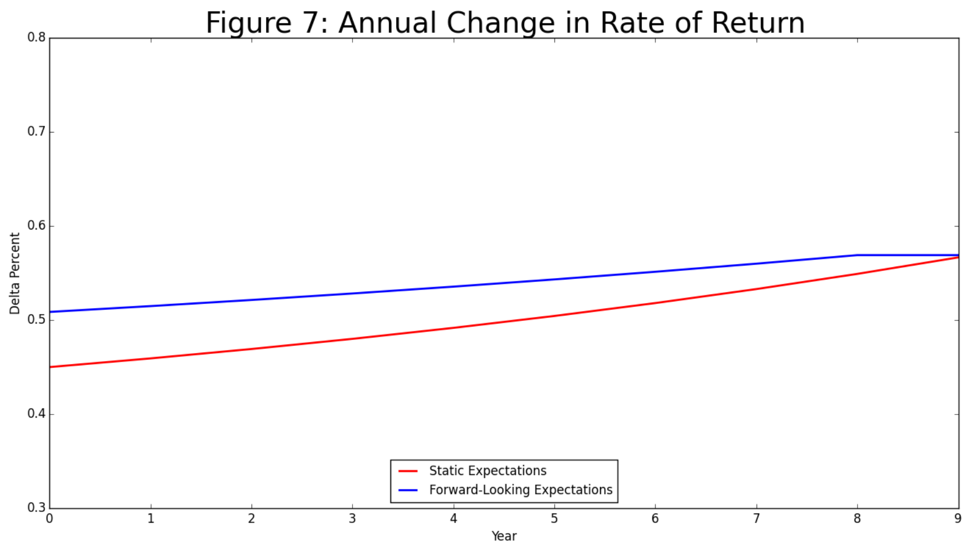
If the absolute value of DIFF(t) is less than the tolerance, then the process ends. If the absolute value of DIFF(t) is greater than or equal to the tolerance, the model replaces $EROR\_ACT(t)^1$ with $EROR\_ACT(t)^2$ and starts the process again.

For the small model examined convergence is achieved relatively quickly (less 5 seconds) for a relatively small number of iterations (2-3 loops), negating the need for a partial adjustment mechanism to avoid result cycling (as described in Dixon et al., 2003). This paper does not examine whether this result holds for larger models.

Policy Shock and Results

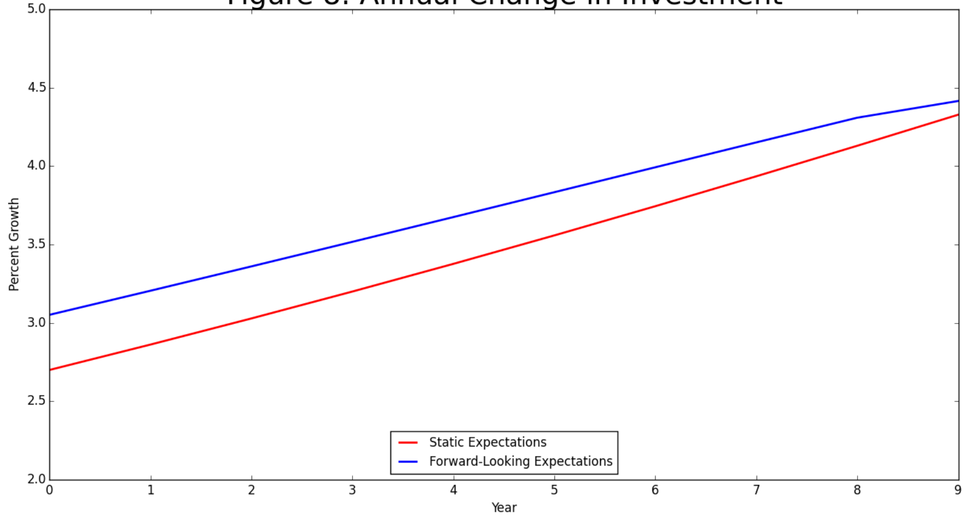
A 2.5% per annum labour saving technology is applied to the model in the form of a shock to an exogenous variable in the model’s labour demand equations.

As expected, the expected rate of return in the comparative-static simulation climbs gradually over time as the effect of the annual shocks compounds through the database. The forward-looking returns, however, are much larger at the start of the period as investors foresee the impact of the change on their returns from investment now in future years. This is shown in Figure 7 below.



Consistent with higher changes in rates of return in the early years of the model run, changes in investment starts much higher in the forward-looking expectation simulation that the comparative-static simulation and stays higher for the entire period. This is partially because investment early in the period causes the capital stock to grow rapidly, requiring greater investment to offset depreciation later. Investment for both simulations is shown in Figure 8 below.

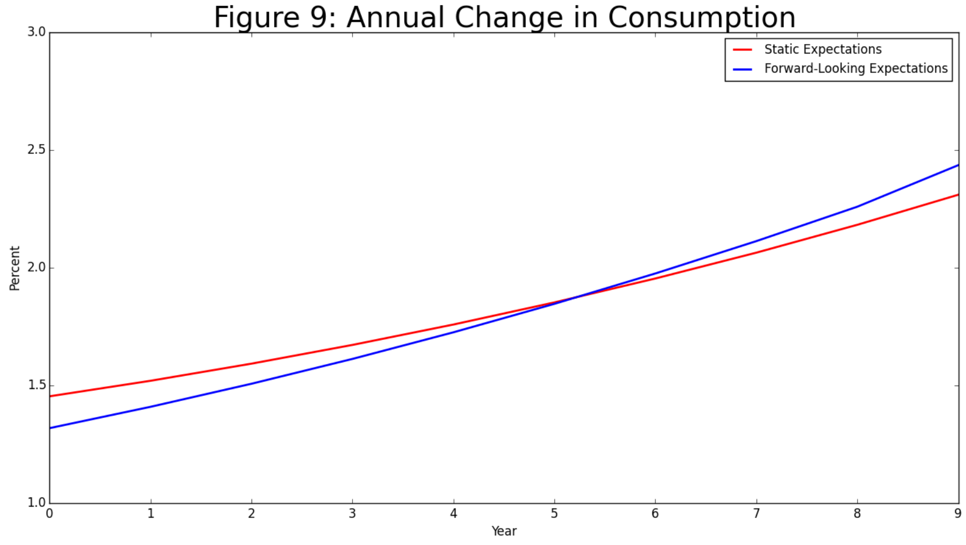
Figure 8: Annual Change in Investment



The capital stock in year 9 rises very slightly under the forward looking expectations scenario, from $1,016.1 million to $1,017.4 million or 0.13% higher. The growth in capital from the steady state capital stock of $1,000 million was 7.94% higher under forward looking expectations than under static expectations.

Figure 9 shows that real consumption in the forward-looking expectations simulation falls dramatically as the economy devotes more resources to investment rather than consumption in a closed system. In the later years of the forward-looking expectations simulation, consumption rises dramatically as the larger capital stock generates incomes and rates of return fall.

Figure 9: Annual Change in Consumption



As noted above, the model does not optimise consumption over time, so the results are dependent upon how well the approximation of the dynamic investment response accurately represents the actual response in the economy. This is not an ideal situation, but solving to optimise consumption would entail solving a complex large non-linear model for all years considered simultaneously.

Application 2: An Integrated CGE Model and a Detailed Electricity Generation Model

Background

The concept of combining the advantages of CGE models and electricity dispatch/market models is not new. For example in 2011 the MMRF CGE model and ROAM Consulting/SKA MMA electricity-sector models[16] were combined for Treasury's modelling of the Carbon Pollution Reduction Scheme (Treasury, 2011).

This process is described in Adams, Parmenter and Verikios (2016), where MMRF was combined with Frontier Economics' WHIRLYGIG model (Frontier Economics, 2009). The modelling process was:

- MMRF Provides information on fuel prices and other electricity-sector costs and on electricity demand;
- Given these assumptions, WHIRLYGIG generates detailed cost estimates for various generation types, which are then fed back into MMRF; and
- Information is passed back and forth between the models in a series of iterations until the retail price of electricity has stabilised in both models.

Adams et al. (2016) do not specify whether WHIRLYGIG is run with electricity demand specified in actual trading interval order (as in Figure 1 above) or as a load duration curve (Figure 2 above).

They note that using electricity market models has the advantage of increasing the technical detail, better simulate changes in generation capacity and increase in the policy detail that can be analysed.

Given the level of detail in each model, it makes sense to utilise the pre-existing detail in established model which has been built up over a many years, rather than start again with an integrated model.

Nevertheless, with improvements in computer processing power and the importance of the policy, this might not be the case indefinitely. This paper undertakes the first steps towards an integrated model, by linking still separate models communicating in a similar way to Adams et al. (2016), but combined in the same software file and with the looping process automated.

[16] And other industry specific models.

The Models

CGE Model

The CGE model used in this simulation is a very simple. It is a linearised comparative-static model set to a short-run closure, with capital in each industry fixed and wages set to an economy-wide fixed level.

It is a three industry, three good model. The industries are electricity generation, resources (which supplies fuel to electricity generation) and services. The database for the model is simulated but the electricity sector is approximately scaled to that in the electricity dispatch model below. All industries are then scaled to the equivalent to the equivalent industries in the Australian Input-Output Table Direct Requirements Coefficients (Australian Bureau of Statistics, 2016).

The model has Leontief technology at the top level for intermediate inputs and primary factors. There is linearised CES substitution between primary factors. The model is a closed economy with no investment in which all outputs are consumed.

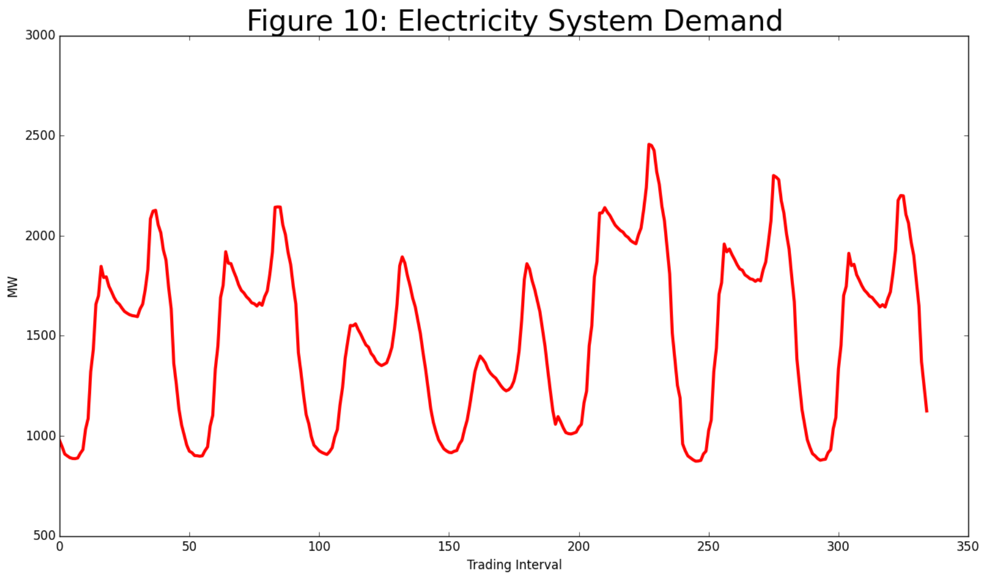
The model contains both a nominal income constraint and a commodity market clearing constraint, so has no general price numeraire as discussed above. The only fixed price is the economy-wide wage.

Electricity Markey Model

The electricity dispatch model used in this simulation is a straightforward standard but simplified model of electricity generator dispatch (modelled on a simplified version of Morales-Espańa, Latorre and Ramos, 2013). It is a Mixed-Integer Linear Program (MILP) in levels, which allows for binary variables to impose minimum generation constraints. In this model:

- There are 335 trading intervals, or approximately one week’s worth of half hour trading intervals. Cost is minimised for a fixed level of varying demand across the entire 335 intervals;

- Demand varies between 876 MW (or 438 MWh per trading interval) and 2,456 MW per trading interval, and varies by trading interval according to daily fluctuations in winter demand in the Western Australian Wholesale Electricity Market (WEM). This is shown in Figure 10 below;



- It is energy-only dispatch, with no specific allowance for capital costs, as occurs in the National Electricity Market (NEM);

- There is a fixed capacity of generators of 3,150 MW. Generators have constant marginal costs of production between minimum and maximum generation levels;

- A generator can be shut down, but if it is to be dispatched it must be at no less than its minimum generation level. For example, a generator with minimum generation of 100 MW and maximum generation of 200 MW can be dispatched at 0 MW, or anywhere between 100-200 MW;

- The generation sector comprises of :

  - Three baseload generators totalling 1,200 MW, with low marginal cost of production, but high start-up and shut-down costs;

  - Two mid merit plants totally 350 MW, with moderate startup and shutdown costs and higher marginal costs than the baseload units;

  - Six peaking plants, with very low startup and shutdown costs; and

  - One Generator with very flexible production levels, but very high ($1000/MWh) costs, this is to reflect that the system might run out of capacity for some trading intervals. This can be thought of as approximating a situation where generators offer higher than their marginal costs in the NEM during periods of high demand and limited supply/ Thist has not been fed back to the electricity industry in the CGE model as super-normal profits at this stage.

The model is highly simplified to accommodate fast solution times. For example, no generator ramp-rates or minimum up or down-times are contained in the model at this stage. Including generator ramp rates was trialled but this dramatically increased model run times.

Link between Models and the Policy Shock

The integration between the two models assumes that the 335 trading intervals equates to one year in the CGE model. In reality, there are 17,520 trading intervals during a non-leap year.

The electricity model is treated as a cost calculator in the CGE model, rather than as a market in its own right. It effectively calculates the fuel cost (or price of the resource industry commodity) for the CGE electricity industry.

The process is best explained by outlining the policy shock in this paper. The shock is in the form of a 200 MW renewable generator entering the system. It is a short-run simulation, so the model does not consider the investment cost of the new generator. Rather, it just appears in the electricity industry capital stock and is automatically valued at its rental value.

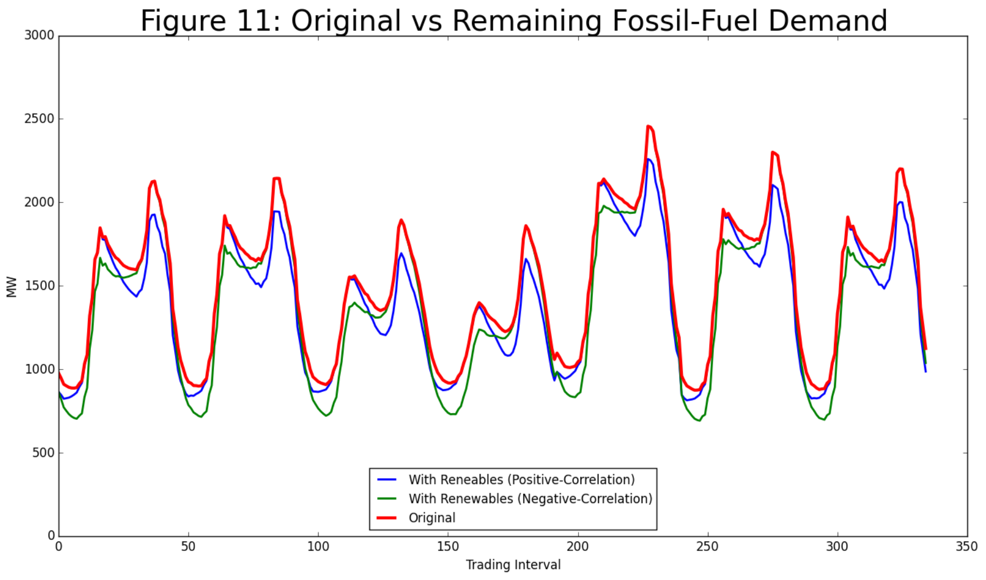
Electricity production for the renewable intermittent generator is simulated with production rising and falling according to a Sine Curve;

- Two scenarios are examined for renewable production. The first has production from the new generator correlating with demand, reducing the system's demand peaks and not affecting offpeak production.

- The second correlates negatively with demand, leaving the system peaks unaffected but the offpeak troughs lower than beforehand; and

- The renewable generator is said to have 'dispatch priority', where it is dispatched whenever the wind is blowing. This is consistent with arrangements in electricity markets around the world[17].

Demand for the initial fossil-fuel industries after the renewable generator has dispatched under both scenarios is shown in Figure 11 below.



[17] Can turn off if price too low.

The renewable generator's marginal cost is zero[18] when it is running, but its production is intermittent depending on the availability of the resource, with no energy storage possible. This might be because the wind does not blow or the sun does not shine all of the time.

The steps to implement this shock into the two models are numerous and outlined below.

- Calculate the original average price of electricity in the electricity industry/commodity by running the electricity market model separately. This is **AP0**.

- Run the CGE model with an increase in capital of 6.3% (200 MW divided by 3,150 MW) in the electricity industry;

- Calculate the change in the quantity of capital in the electricity generation industry and the change in fuel input price (from the resources industry) to the electricity generation industry.

- Loop the two models as follows:

  While C_DIFF>tol_C or Q_DIFF>tol_Q (see below for definitions):

  > Calculate remaining demand for traditional generators by subtracting renewable production from the original production. This reduction in traditional electricity demand is then increased by the increase in production from the CGE model in **QELEC_CGE^{1}**.
  >
  > The marginal cost for each generator is increased by the increase in the price of resources industry output used by the electricity industry from the initial CGE run. This is **PRES_CGE^{1}**.
  >
  > The electricity dispatch model is run with demand and generator costs affected by Z_CGE and A_CGE. The model calculates an average price for electricity for the remaining demand. This is then adjusted for the zero marginal cost energy from the renewable source to find **AP** and **dp**=(AP-AP0)/AP0.
  >
  > The value for dp is fed into the CGE model as an exogenous value for the cost of intermediate inputs purchased by the electricity industry and the CGE model is run again. Updated value for PRES_CGE^{2} and QELEC_CGE^{2} are calculated.
  >
  > **C_DIFF** and **Q_DIFF** are calculated as the difference between the original values for PRES_CGE and QELEC_CGE and the updated values.
  >
  > Specify tolerances for each of C_DIFF (**tol_C**) and Q_DIFF (**tol_Q**) that are acceptable to the modeller for convergence.
  >
  > If the either difference is larger than its respective tolerance, the updated values of Z_CGE and A_CGE are fed back into the electricity model and so on.

- The adjustment process in this loop is specified as a full adjustment process, where the exogenous inputs are imposed at less than the change from the alternative

[18] The vast majority of costs in currently-know renewable technologies are fixed capital costs.

model. For the shock implemented in this paper there is no value recycling where values do not converge, or explosive cycling where the values move away from convergence.

The speed of convergence depends on the power of computer and type of solver involved, but is usually less than 20 seconds for around 3-5 loops for convergence. This compares to Adams et al. (2016), which notes a typical count of three loops before MMRF and WHIRLYGIG results converge.

Results

The results of the from the stand-alone CGE simulation, the CGE model with positively correlated renewable energy and CGE scenario with negatively correlated renewable energy are shown in Table 1. The levels of various electricity prices from each scenario are also shown.

The simulation has added capital for no investment cost, which is an old consultant's trick to curry favour with governments to support their client's projects, so is expected show some aggregate economic benefit. The initial CGE simulation shows that this is the case.

Table 1: CGE/Electricity Market Model Results from the Addition of a 200 MW Renewable Generator (% Change unless otherwise stated)

	Original (no electricity model) CGE Results	CGE Results with Electricity Model Positive Correlation	CGE Results with Electricity Model Negative Correlation
Production of Electricity Industry	1.85	02.41	1.60
Aggregate Prices	-0.027	-3.72	1.50
Aggregate Consumption	1.84	5.43	2.00
Price of Electricity	-0.597	-27.70	8.95
Price of Electricity Fuel Input	4.99	-31.80	17.90
Average Marginal Fuel Cost of Fossil Fuel Generation	na	$56.92/MWh	$102.34/MWh
Average Marginal Fuel Cost of All Generation	na	$54.42/MWh	$97.11/MWh

Note: the original Electricity Price from the initial electricity market simulation (AP0) was $81/MWh[19]

However, when the electricity model is integrated to the CGE model, several things happen:

- In the stand-alone electricity model, electricity prices fall faster than the CGE model predicts in the for the positive correlation scenario, but rise very slightly in the negative correlation scenario;
    - This is because the additional supply during low demand periods in the negative correlation scenario forces base load generators into costly start-ups and shut-downs and to run more expensive mid-merit generators during offpeak periods[20];
- In both scenarios, the rise in economic activity in the intial CGE simulation causes the prices of the resources industry, which supplies the electricity industry's main commodity input, to rise;
- This increase in input prices feeds through to increasing electricity prices, which loops through the model back to increasing fuel input prices;
- The final electricity prices in both scenarios are higher than where the stand alone CGE model and stand-alone electricity model originally predicted, although the price in the positive correlation simulation is much lower than the original or base price; and
- The price of electricity in the negative correlation simulation is almost twice that in the positive correlation scenario.

These results, while very preliminary, show that the shape of the renewable production curve relative to the system demand curve is critical to calculating the economic impact of adding such technologies to an electricity system. The currently topical 'duck curve' in California (California Independent System Operator, 2016) shows the difficulty a system might experience if too much electricity is produced at low demand time.

For a model of this type to be truly useful it needs a theory of investment, capital growth and capital retirement in the electricity industry that integrates into both the CGE and electricity models. Additionally, commercially viable electricity storage may be available in the future, which will need to be incorporated into the electricity model.

Discussion

The examples presented in this paper are specific to the precise models, databases and policy shocks presented. This paper does not examine whether the results are applicable to more general and realistic models and the number areas for development before they could be considered useful models is virtually endless.

Nevertheless, for the specific cases considered, this paper shows that:

[19] The prices in the stand-alone electricity market model, with original demand minus renewable supply, were $49.84/MWh and $82.74/MWh.

[20] This does not include additional ancillary services costs (e.g. additional load following costs) that renewable impose on the system.

- Small-scale comparative-static and recursive-dynamic CGE models can be built in the Python Pyomo package;
- Static expectations may lead to a smaller investment shock than is warranted in response to a known but phased-in policy shock;
- Construction of an integrated CGE model and electricity model is possible; and
- The economic impact of new non-storable renewable energy generation entering the energy sector depends critically on how production from this new industry correlates with electricity demand.

Given the development of substantial development of GEMPACK over many years it is unlikely that the first bullet point is little more than a novelty. Additionally, while not certain, modification of the Dixon et al. (2003) algorithm to look forward past year t+1 would probably be able to be accommodated with GEMPACK and its associated dynamic modelling software.

Undertaking an integrated CGE and electricity model would be a substantial undertaking and no claim is made regarding whether this is a worthwhile exercise. However, exact specification of the stationary energy sector is important to policy questions that are likely to become more important over the next decade. It is not clear whether this could be accommodated directly into GEMPACK.

Practical implementation of the concepts examined in this paper would require substantial computing power. However, computing power continues to improve enormously. For example, computers are much more powerful in 2016 than then when Dixon el al. (2003) noted that solving a fully dynamic model was computationally intensive.[21]

Additionally, the pioneering CGE modellers in Australia were not deterred by daunting computing tasks. The original ORANI model was run on the external CSIRO's CYBER 76 computing system, with charges of $50 (in 1977 prices) to run the model (Sutton, 1977). In the early 1980s ORANI was still solved on the CSIRO mainframe computer where modellers would submit data cards and wait overnight for the model to run (Horridge, Meeraus, Pearson and Rutherford, 2013). They did so, presumably, because they thought the results were vital enough to wait for. The question is still whether the results are worth the wait.

[21] The difficulty of interpreting non-linear equations still remains.

References

Adams, P.D., B.R. Parmenter and G. Verikios (2016), An Emissions Trading Scheme for Australia: National and Regional Impacts, Centre of Policy Studies Working Paper G-243 Melbourne.

Australian Bureau of Statistics (2016), Australian Input-Output Tables 2013-14, Canberra, Table 9, Direct Requirement Coefficients (Indirect Allocation of Imports).

California Independent System Operator (2016), *What the Duck Curve Tells Us About Managing a Green Grid, Fast Facts*, Folsom, https://www.caiso.com/Documents/FlexibleResourcesHelpRenewables_FastFacts.pdf .

Chan, J. (2014), *Learn Python in One Day and Learn it Well*, LCF Publishing, Kindle Edition.

Dixon, P.B., K.R. Pearson, M.R. Picton and M.T. Rimmer (2003), Rational Expectations for Large Models: a Practical Algorithm and a Policy Application, Centre of Policy Studies Working Paper ip-81, Melbourne.

Dixon, P.B and M.T Rimmer (2005), *Mini-USAGE: reducing Barriers to Entry in CGE Modelling*, Paper prepared for the 8th Annual Conference on Global Economic Analysis, Lűbeck, Germany, June 9-11 2005.

Frontier Economics (2009), Modelling *Methodology and Assumptions: A Report for IPART*, Melbourne.

Hart, W.E., C. Laird, J. Watson and D.L. Woodruff (2012), *Pyomo - Optimisation Modelling in Python*, Springer, New York, Kindle Edition.

Horridge, M. (2014), *ORANI_G: A Generic Single-Country Computable General Equilibrium Model*, Centre of Policy Studies and Impact Project, Victoria University, Australia.

Horridge, M. and K.R. Pearson (2011), *Solution Software for CGE Modelling*, Centre of Policy Studies Working Paper G-214, Melbourne.

Horridge, M and Pearson, K and Meeraus, A and Rutherford, T (2013), “Solution Software for Computable General Equilibrium Modeling”. In: *Handbook of Computable General Equilibrium Modelling*. Dixon, PB and Jorgenson, D, eds. Elsevier, Amsterdam, 1331 - 1381. ISBN 9780444595560

Independent Market Operator (2012), *Wholesale Electricity Market Design Summary*, http://www.aemo.com.au/Electricity/-/media/F12B82DEB2484DD0848FD5C277DB5CA8.ashx , accessed 7 August 2016.

Morales-Espańa, G., J.M. Latorre and A. Ramos, (2013), “Tight and Compact MILP Formulation of Start-Up and Shut-Down Ramping in Unit Commitment”, *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1288-1296, May 2013.

Peter, M.W., M. Horridge, G.A.Meagher, F. Naqvi and B.R. Parmenter (1996), *The Theoretical Structure of MONASH-MRF*, Centre of Policy Studies Working Paper Number OP-85, Melbourne.

Rutherford, T.F. (2006), *GTAP6 in GAMS*, http://www.mpsge.org/gtap6/ .

Sutton, J. (1977), Computing Manual for the ORANI Model - First Edition, Centre of Policy Studies Computing Paper C1-01, Melbourne.

Treasury (2011), *Strong Growth, Low Pollution: Modelling a Carbon Price*, Commonwealth of Australia, Canberra, http://carbonpricemodelling.treasury.gov.au/content/report.asp .