Modeling Supply and Demand in Modelica

Michael M. Tiller[1]

[1]michael.tiller@gmail.com

Abstract

This paper demonstrates using component oriented modeling and acausal semantics to create a basic library of behavioral components to model supply and demand. The models presented are each steady state models. While some examples include shifting economic conditions that cause the equilibrium points to change during the simulation, none of the models feature dynamic states. The main purpose of this paper is to demonstrate to people unfamiliar with Modelica (Modelica Association 2017) how Modelica can be used to model non-engineering systems and how it makes such modeling faster, easier and less error-prone compared to other approaches (*e.g.,* using spreadsheets).

Keywords: Modelica, economics, supply and demand

1 Introduction

Modelica was designed from the outset to be domain neutral. The hope was that the foundations of Modelica were sufficiently complete that it could not only be used to model the wide range of engineering related systems that the designers were familiar with but that it was universal enough to model nearly any system from any domain, even those unfamiliar to the language designers. The wide range of domains that Modelica has been applied to over the last 20 years is a testament to the success of these design goals.

As I hope to demonstrate in this paper, the acausal semantics in Modelica are not only useful for describing the familiar conservation laws present across engineering domains. These semantics can be utilized whenever there is a need to ensure a proper accounting of many different quantities. In this particular case, we will focus on the movement of goods passing from producers to consumers.

In this paper, we will introduce the economic concepts of supply and demand. These concepts are often discussed only in qualitative terms. But if you characterize supply and demand quantitatively, you can use the features of Modelica to create a library of components models that can model not just sources of supply (production) and demand (consumers) but also model other economic effects such as taxation, complementary goods, exchange rates, *etc.*

The goal of this paper is not to provide a comprehensive collection of quantitative models of economic actors and effects. Instead, this paper attempts to achieve two primary goals. First, to demonstrate the applicability of Modelica to yet another domain. In this case, the non-engineering related subset of economics that involves modeling of supply and demand. The other goal of the paper is to describe the models in such a way that someone familiar with economics but unfamiliar with Modelica will appreciate how Modelica works and how it could be useful in the field of economics to create trusted and reusable libraries of components that are capable of performing all the necessary bookkeeping required for supply and demand systems and solve the underlying non-linear systems of equations better than other approaches (*e.g.,* using spreadsheets).

There are a number of online books (Taylor 2018; Hutchinson 2016; Posner and Tayari 2018) that cover the topics in this paper in much greater depth and I would encourage the reader to seek out these books to learn more about these topics from experts. Furthermore, there have been previous articles that used Modelica to model economic effects (Zimmer and Schlabe 2012; Casella, Miragliotta, and Uglietti 2005). However, these papers focused on specific types of markets and used slightly different approaches.

2 Mathematics of Supply and Demand

Before we discuss the details of the Modelica implementation, it will be useful to provide a basic discussion of the topic of supply and demand curves, how they are characterized and how we can use them to arrive at a supplied price and supplied volume.

Both supply and demand curves are expressed with price as the dependent variable and sales volume as the independent variable. This choice is unintuitive because price is the thing that is most controllable here and volume is simply a consequence of the chosen price. Nevertheless, this is the way supply and demand curves are typically represented and so this paper follows that convention as well.

2.1 Supply

As mentioned previously, the supply curve shows price as a function of sales volume. The curve visualizes what the per unit price would be for a given sales volume. A very simple supply curve is shown by the blue line in Figure 4. One important characteristic of a supply curve is that *it generally has a positive derivative*. At first, this seems counter-intuitive because most producers actually

discount their products if customers are willing to buy more of the product. But this is related to pricing strategy and is contingent on being able to scale up production.

But supply curves are generally based on availability of limited resources (*e.g.,* Uber's "surge pricing"). As such, as demand for a limited resource goes up, the price that a producer can command will go up. In the *long run*, the market may compensate by increasing production which, in turn, increases overall supply and lowers prices. But this is an example of how the supply curve itself adapts to market conditions over time.

Another reason for the supply curve to have a positive derivative is related to accessibility of raw materials. Even if the amount of the raw material doesn't have a finite limit, it may be the case that there are multiple sources of the raw material and that some are more expensive than others. In such a case, the shape of the supply curve is a reflection of the fact that the initial sales will rely on easily accessible (*i.e.,* cheaper) sources while larger sales will require less accessible (*i.e.,* more expensive) sources.

2.2 Demand

While supply curves represent the availability of a given good, the demand curve represents how much consumers are willing to pay for a good. A very simple demand curve is shown by the red line in Figure 4. Unlike the supply curve, the demand curve generally has *a negative first derivative*. The simplest way to understand this is to think about a demand curve as a *histogram*. Imagine the consumer who values this good the most. They define the maximum possible price (*i.e.,* the y-intercept on demand curve). If producers offer that product at that price, they can expect to sell only to the wealthiest or most enthusiastic consumers. However, if producers reduce their price, they can expect to attract even more buyers. As they continue to lower the price, they can reasonably expect to continue to attract more buyers. In this sense, the demand curve is a histogram showing how much consumers are willing to pay.

3 Interfaces

3.1 Connector

The cornerstone of any Modelica library is the `connector` definitions. This is because the `connectors` define the way in which components interact. So it is necessary to carefully design the `connectors` so they can represent all potential interactions.

In the case of modeling supply and demand, there are two fundamental quantities we are concerned with. The first is the *price* of goods. We will talk about how price impacts the behavior of both producers and consumers shortly. But for now, all we need to recognize is that price motivates transactions to occur.

The other fundamental quantity is *volume* of sales. This represents the number of goods either produced or consumed (by producers and consumers, respectively). Our systems will be formulated such that all goods must be accounted for. This means that all goods produced have to go *somewhere*. It might be into a warehouse, it might be purchased by a consumer, it might be transported to a geographically remote market. But it must be accounted for.

As such, the volume of sales will be the `flow` variable in our system. In this way, the acausal semantics of Modelica, normally used to account for conserved quantities in engineering domains, will ensure our constraint that all goods are accounted for. Since the volume of sales is the `flow` variable, we will adopt the price as our across/potential variable.

Before we define the `connector`, let us first introduce two types:

```
type Price = Real(min=0, quantity="Price");
type SalesVolume = Real(quantity="Units");
```

With these two types defined, we can now define our connector as follows:

```
connector Market "Market interaction"
  Types.Price price(start=10);
  flow Types.SalesVolume volume;
end Market;
```

We establish a `min` and `start` value on the `Price` type to assist solvers in finding solutions for non-linear systems. The `min` attribute informs the solvers that negative values are not viable solutions. The `start` attribute provides an initial guess which helps the solver locate a solution and/or choose between multiple solutions. In the `Market` connector, we chose the rather arbitary value of 10 as an initial guess just to provide a positive initial guess. In specific models, thise `start` attribute can be overriden to provide a better problem specific initial guess. The non-linear solvers will also need good initial guesses for `volume`, but we cannot provide `min` and `start` values here because the sign will depend on the nature of the component so we will instead add those attributes on variables whose sign is known.

3.2 Partial Models

Our `connector` is defined in the `Interfaces` subpackage along with a few useful partial models.

3.2.1 Producer

The first of these partial models is a `Producer` model. The idea behind the `Producer` model is to define some protected variables associated with and employing the *sign convention* of a producer. Specifically, the normal Modelica sign convention is that flow of a conserved quantity (in this case, goods) is positive when flowing into a component. In the case of a producer, goods are always flowing *out*. As such, the `volume` field on the connector is always negative. However, the supply curve volume is always positive. For this reason, within the `Producer` model we define a local variable, `volume`, which represents the independent variable on the supply curve (*i.e.,*

DOI 10.3384/ecp19157365

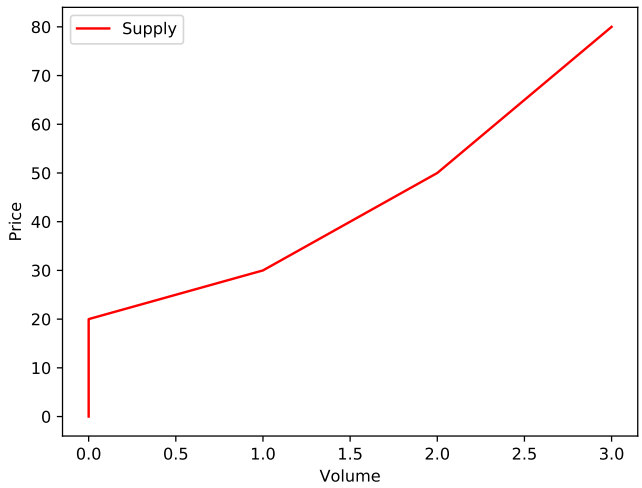

Figure 1. Sample S-parameterized Supply Curve

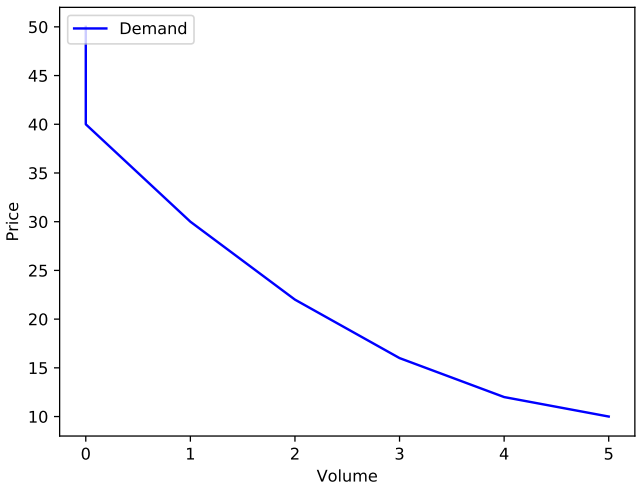

Figure 2. Sample S-parameterized Demand Curve

normally positive) and map that to the `volume` field on the `connector`, *i.e.*,

```
partial model Producer "Goods producer"
  Types.SalesVolume volume;
  Types.Price price;
  Interfaces.Market market(volume(start
    =-10));
protected
  Real s(start=-1);
equation
   if (s<0) then
     price = market.price;
     volume = -s;
   else
     price = market.price-s;
     volume = 0;
   end if;
  market.volume = -volume;
end Producer;
```

The first thing to notice in this model is the fact that it doesn't just define variables for `price` and `volume` but also a variable named `s`. Internally, the supply curve is not strictly represented as `price` as a function of `volume`. Instead, both `price` and `volume` are represented in terms of `s`. The resulting supply curve (parameterized in terms of `s`) is shown in Figure 1.

Doing the parameterization in this way allows us to define multiple potential prices for a given volume. This allows us to handle the case where the lowest possible production price is still above the highest price that consumers are willing to pay. Using this parameterization, we extend the supply curve to indicate that no goods will be produced (`volume=0`) for all prices below the lowest possible production costs. This allows us to solve for a supplied price and supplied volume in the case where a producer (or consumer) is priced out of the market.

3.2.2 Consumer

The `Consumer` model is very similar to the `Producer` model. It doesn't actually need to perform the sign change on `volume` but it does implement a similar s-parameterization of the demand curve except that in the case of the demand curve the `s` parameter extends the price *upward* rather than downward as shown in Figure 2.

```
partial model Consumer "Goods consumer"
  extends Curve;
  Types.SalesVolume volume;
  Types.Price price;
  Interfaces.Market market;
protected
  Real s(start=1) "Volume or price gap";
equation
  if (s<0) then
    price = market.price + s;
    volume = 0;
  else
    market.price = price;
    volume = s;
  end if;
  market.volume = volume;
end Consumer;
```

4 Supply and Demand

With the connectors and `partial` models defined, we can start defining various models for both supply and demand.

4.1 Linear Models

Many explanations of supply and demand use linear supply and demand curves to describe how to arrive at the supplied price and volume. So we'll start with such models and then transition into more realistic models of supply and demand shortly.

Consider a market where the highest price a consumer is willing to pay would be $12. But for every $1 that we reduce the cost of the good, we find 20 more customer. Let us further assume that the producer of these goods must charge at least $10 and for every $1 increase in price, 5 more goods can be supplied.

The supply curve would then be represented by:

$$p^s(v) = p_0^s + \beta * v = 10 + 0.2 * v$$

where β represents how much the production price would increase with each additional unit of goods produced. In the same way, the demand curve would be represented by:

$$p^d(v) = p_0^d - \alpha * v = 12 - 0.05 * v$$

where α represents how much the price would have to be reduced in order to sell each additional unit of goods.

These two curves are shown as the blue and red lines, respectively, in Figure 4. We want to find a combination of price and volume that are consistent with both the supply curve and the demand curve. In fact, what we are looking for is the *intersection* of these two curves. This is a price/volume point that satisfies both the consumers and the producer. As we can see in Figure 4, the price for goods at this point is called the *supplied price* and the volume of goods sold in that scenario is the *supplied volume*. The supplied volume is the volume, v_s, at which the price on the supply curve matched the price on the demand curve. In other words,

$$p^s(v_s) = p^d(v_s)$$

Note that because of the connection semantics of Modelica, this equation is automatically generated whenver we connect the `Market` connector of the consumer and the producers. This equation combined with the "conservation equation" generator by the connector which, in the case of a system containing only a consumer an producer as the effect of setting the `volume` values used by both to be equal, means that for this use case we can trivially determine that the supplied volume must be:

$$\frac{p_0^d - p_0^s}{\alpha + \beta} = \frac{12 - 10}{0.2 + 0.05} = \frac{2}{0.25} = 8$$

Plugging this supplied value in the supply (or demand) curve tells us that the supplied price must, therefore, be $10 + 0.2 * 8$ or \$11.6. To model this in Modelica, we can create the following two models to represent the supply and demand curves respectively:

```
model LinearProducer
  "Production with a minimum price and
      linear price increase"
  extends Interfaces.Producer;
  parameter Types.Price min_price "Minimum
      price to produce";
  parameter Types.PriceSensitivity beta "
      Price increase as a function of
      volume";
equation
  price = min_price + beta*volume;
end LinearProducer;

model LinearConsumer "Linear distribution
    of consumers"
  extends Interfaces.Consumer;
  parameter Types.Price max_price "The
      largest amount any consumer is
      willing to pay for this good";
  parameter Types.PriceSensitivity alpha "
      Rate of price drop as volume
      increases";
equation
  price = max_price - alpha*volume;
end LinearConsumer;
```

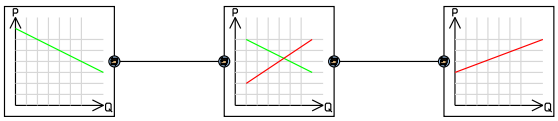

Figure 3. `LinearMarket` model

With these two models in hand, we can create a system market model in Modelica as follows:

```
model LinearMarket
  "Market where consumer and producer have
      linear relationships"

  Components.LinearConsumer consumer(
      max_price=12, alpha=0.05);
  Components.LinearProducer producer(
      min_price=10, beta=0.2);
  Components.MarketAnalysis market;
equation
  connect(market.producers, producer.market
      );
  connect(market.consumers, consumer.market
      );
end LinearMarket;
```

A diagram of our system model is shown in Figure 3. Note that to solve for the supplied price and supplied demand all we need to do is connect the `Market` connector of the consumer and the producer. But for this model we have introduced a special "intermediary" called the `MarketAnalysis` model in the center between the consumer and producer. This `MarketAnalysis` model enforces a market equilibrium condition (just as if we had directly connected the consumer to the producer) but only *at the start of the simulation*. This means that the price that the consumer is willing to pay has to match the price that the producer is willing to charge. Furthermore, the volume of goods that the producer produces must be equal to the volume of goods that consumers consume.

This solution is found at the start of the simulation. From that point (and over the following 1 second of simulation time), the `MarketAnalysis` model perturbs the system into non-equilibrium states. As a result of this process, it is possible to visualize the supply and demand curves parametrically. The results of the Modelica simulation are shown in Figure 4.

4.2 Exponential Models

Linear models work well to explain the concept of supply and demand as well as the idea of supplied price and supplied volume because it is straightforward to find a closed

DOI 10.3384/ecp19157365

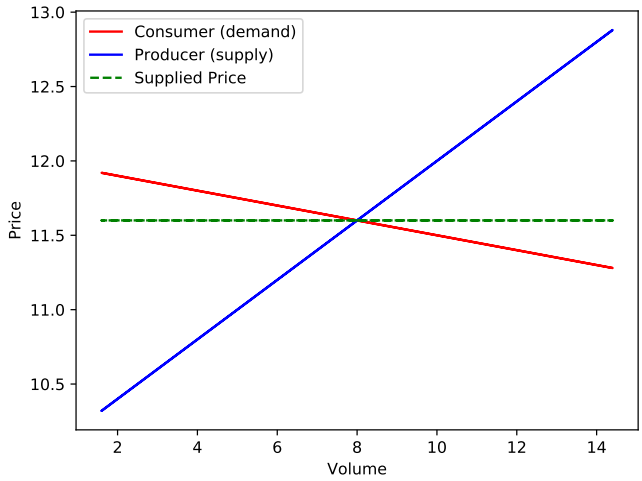

Figure 4. Simulation results for `LinearMarket`

form solution for two intersecting lines. But linear models are problematic because they don't make much sense. Very quickly the lines cross the axes and leave the first quadrant. For example, once the demand line crosses the x axis, the price goes negative. This reflects a situation where producers would have to pay consumers to take their products. While there are markets where this effect could be seen, it isn't a normal situation and you would generally have to have supplied many, many consumers at positive prices before this is likely to happen (something not usually reflected by a linear demand curve). Similar problems occur when the supply curve leaves the first quadrant.

A more realistic model is an exponential model. As with the linear models, this model also defines a price point on both the supply and demand curve associated with a sales volume of zero (*i.e.,* p_0^s and p_0^d). But instead of assuming a linear relationship, we introduce parameters representing an exponential price decay or growth.

Let's start with the demand curve. The equation for an exponential demand curve would be:

$$p^d(v) = p_0^d e^{-k_d v}$$

As before, p_0^d represents the maximum that consumers would be willing to pay. But with this model of demand, that price falls off exponentially with volume. An important characteristic of such a demand curve is that it never drops below zero. In other words, as the price approaches zero, the potential volume of sales approaches infinity.

The Modelica code for this model is:

```
model ExponentialConsumer "Exponential
    distribution of consumers"
  extends Interfaces.Consumer;
  parameter Types.Price max_price "The
      largest amount any consumer is
      willing to pay for this good";
  parameter Real decay(min=0) "Exponential
      decay rate as a function of volume";
equation
  price = max_price*exp(-decay*volume);
end ExponentialConsumer;
```

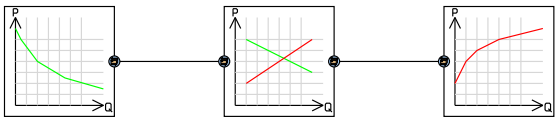

Figure 5. Exponential curves for consumer and producer

The supply curve is slightly different. It is characterized by the following equation:

$$p^s(v) = p_0^d e^{k_s v}$$

Again we see p_0^d, the minimum price that goods can be produced for. But now instead of a linear increase in price with respect to sales volume, we see an exponential curve. Whereas the demand curve tapers off with volume, the supply curve grows exponentially because as more and more of a finite resource is consumed, the cost of the resource sores.

In Modelica, this model can be expressed as:

```
model ExponentialProducer "Exponential
    pricing curve"
  extends Interfaces.Producer;
  parameter Types.Price min_price "Price at
      zero volume";
  parameter Real growth(min=0) "Exponential
      growth rate as a function of volume"
      ;
equation
  price = min_price*exp(growth*volume);
end ExponentialProducer;
```

We can combine an exponential models of supply and demand to create a simple system model as shown in the following Modelica model:

```
model ExponentialMarket
  "Market where consumer and producer have
      exponential relationships"

  Components.MarketAnalysis market(minScale
      =1, maxScale=2.5);
  Components.ExponentialConsumer consumer(
      max_price=12, decay=0.04);
  Components.ExponentialProducer producer(
      min_price=10, growth=0.06);
equation
  connect(consumer.market, market.consumers
      );
  connect(market.producers, producer.market
      );
end ExponentialMarket;
```

The diagram for this model is shown in Figure 5. Simulating this model we get the results shown by the thick lines in Figure 6. Note the slight curvature of the lines vs. the linear model. The curvature would be more pronounced if we considered a wider range of sales volumes.

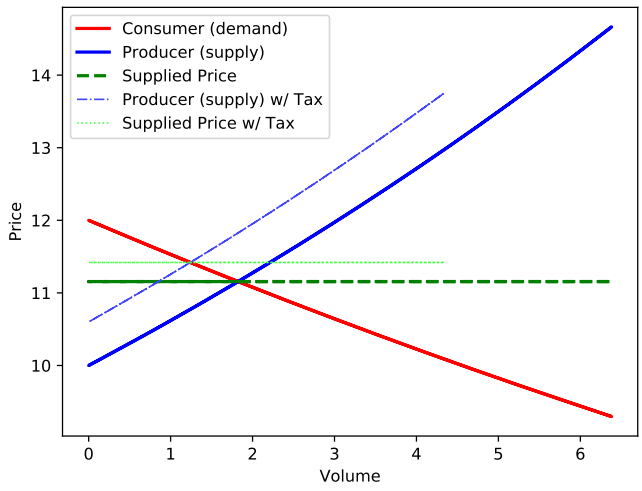

Figure 6. Simulation results for `ExponentialMarket`

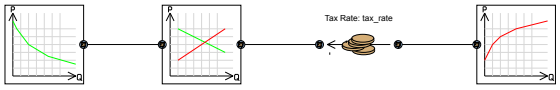

Figure 7. Adding taxation to `ExponentialMarket`

5 Scenarios

With these exponential models in place, a variety of interesting scenarios open up because we have the building blocks necessary to start modeling real markets. Because we have graphical component models we can now compose these scenarios simply by dragging and dropping these consumer and producer models down into a diagram and combining them with various other economic factors.

5.1 Taxation

A very simple adjustment we can make to the market is to introduce a tax and see how that impacts the supply and demand. In our previous example, `ExponentialMarket`, the supplied price was $11.16 and the supplied volume was 1.82. Now let us revise the model to include a model of taxation. The taxation model itself can be implemented as follows:

```
model Tax
  "Model a tax (increasing effective price
     to consumers)"
  parameter Types.TaxRate taxRate;
  output Types.Price taxRevenue;
  Interfaces.Market consumers;
  Interfaces.Market producers;
equation
   consumers.price = producers.price*(1+
      taxRate);
   taxRevenue = producers.price*taxRate*
      producers.volume;
   consumers.volume+producers.volume = 0;
end Tax;
```

Add this to our overall system, we then get:

```
model ExponentialMarketWithTaxes
  "Market where consumer and producer have
     exponential relationships"

  Components.MarketAnalysis market(minScale
     =1, maxScale=2.5);
  Components.ExponentialConsumer consumer(
     max_price=12, decay=0.04);
  Components.ExponentialProducer producer(
     min_price=10, growth=0.06);
  Effects.Tax tax(taxRate=0.06);
equation
  connect(consumer.market, market.consumers
     );
  connect(tax.producers, producer.market);
  connect(tax.consumers, market.producers);
end ExponentialMarketWithTaxes;
```

The diagram for this model is shown in Figure 7. Running this model, which includes the same supply and demand curves, we find that the supplied price has risen from $11.16 to $11.42 and the supplied volume has dropped from 1.82 to 1.24.

In the untaxed case, the consumers paid $20.30 for the goods and all that revenue went to the producers. In the taxed case, consumer spending dropped to $14.17 and, of that, only $13.36 went to the producer. The remaining $0.81 was collected as tax revenue. Note that this seems like a dramatic effect for a 6% sales tax. But please note that the supply and demand curves are completely arbitrary in this example.

5.2 Raw vs. Finished Goods

The next example involves manufacturing. Specifically, we have producers of two different *raw* materials and those are then manufactured into a finished good which is sold to consumers. This example demonstrates the concept of *complementary goods*. Two goods are complementary if demand for one drives of demand for the other because purchasers of one may want (or require) the other good as well.

In order to model our manufacturing system, we must introduce the following model of the `Manufacturer`:

```
model Manufacturer "Combines two types of
    goods to form a third"
  parameter Real markup;
  Interfaces.Market production;
  Interfaces.Market supply_A;
  Interfaces.Market supply_B;
equation
  supply_A.volume + production.volume = 0;
  supply_B.volume + production.volume = 0;
  production.price = (supply_A.price+
     supply_B.price)*(1+markup);
end Manufacturer;
```

This model acts as *both* a consumer and a producer. For each good it produces (to the `production` market), it consumes one good from `supply_A` and another from `supply_B`. In addition, the price that it offers its finished goods for is the price it must pay for the two raw goods plus some percentage markup.

DOI 10.3384/ecp19157365

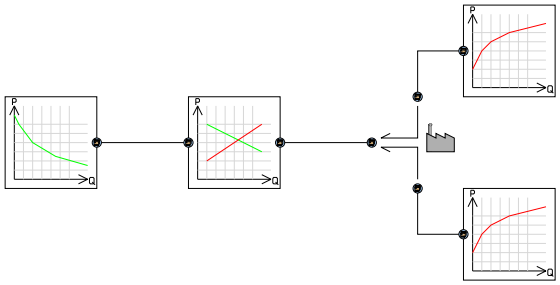

Figure 8. `SupplyChain` model

The final system model, shown in Figure 8, is expressed in Modelica as follows:

```
model SupplyChain "Model of manufacturing
    supply chain"
  Components.Manufacturer manufacturer(
      markup=0);
  Components.MarketAnalysis market;
  Components.ExponentialConsumer consumer(
      max_price=12, decay=0.04);
  Components.ExponentialProducer producer_A
      (growth=0.06, min_price=4);
  Components.ExponentialProducer producer_B
      (growth=0.06, min_price=6);
equation
  connect(market.producers,
      manufacturer.production);
  connect(consumer.market, market.consumers
      );
  connect(producer_A.market,
      manufacturer.supply_A);
  connect(producer_B.market,
      manufacturer.supply_B);
end SupplyChain;
```

For this case, the supplied price for the finished goods is $11.59 at a supplied volume of 0.87.

5.3 Competition for Resources

We can add an interesting twist to the previous model if we add additional consumers for one of the raw materials. This will drive up demand for that good and, as a consequence, increase the price for that particular raw material. Because our goods are complementary, the manufacturing process requires both and the price of the finished goods should rise as a result of the competition for the raw materials.

In this case, we do not need any new models. As seen in Figure 9, we can simply extend the `SupplyChain` model with another consumer for one of the raw materials, *e.g.*,

```
model ResourceCompetition
  extends SupplyChain;
  Components.ExponentialConsumer
      raw_consumer(decay=0.04, max_price=5)
      ;
equation
  connect(raw_consumer.market,
      producer_A.market);
```

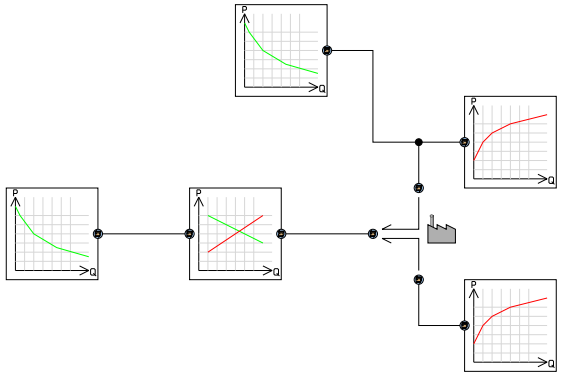

Figure 9. Adding competition for raw materials

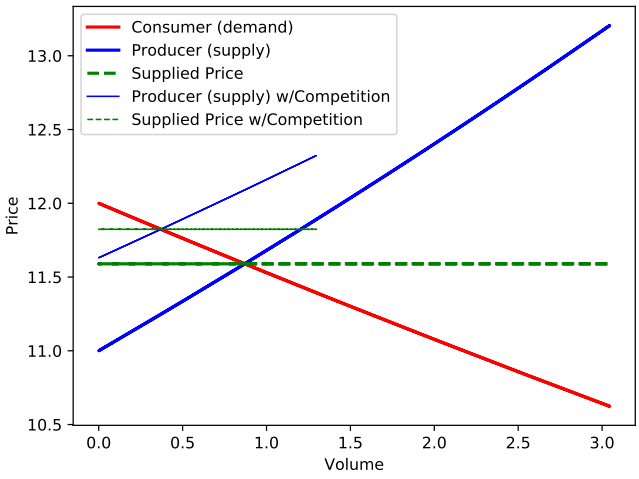

Figure 10. Comparison with and without resource competition

```
end ResourceCompetition;
```

We can see from the results, indicated by the thin lines in Figure 10, that the supplied price of the finished materials has risen to $11.82 and the supplied volume has been reduced to 0.37 because of this competition for the raw materials.

5.4 International Trade

One final example involves international trade. In this model, shown in Figure 11, we have two distinct consumers and producers. Each is located in their own geographical region. Left alone, each consumer would trade only with the producer in their own geographical region. But if we add the ability to ship goods between the geographies (with the associated transportation costs), then we create a global market. But there are other factors that impact global trade besides just transportation costs. Tariffs may be in place to limit the amount of global trade. Furthermore, currency exchange rates will also affect the price of goods.

For our final example, we include all these effects and the resulting Modelica model is:

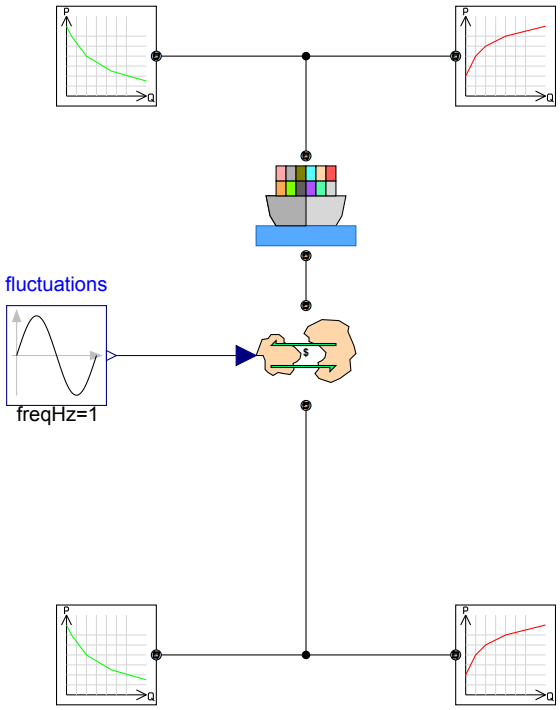

Figure 11. Model of international trade

```
model InternationalTrade
  Components.ExponentialProducer
      foreign_producer(min_price=40, growth
      =0.01);
  Components.ExponentialConsumer
      foreign_consumer(max_price=55, decay
      =0.001);
  Components.ExponentialConsumer
      domestic_consumer(max_price=80, decay
      =0.003);
  Components.ExponentialProducer
      domestic_producer(min_price=50,
      growth=0.02);
  Components.Trade trade(tariff_AB=0.05);
  Modelica.Blocks.Sources.Sine fluctuations
      (
    amplitude=0.05, freqHz=1, startTime
        =0.5,
    offset=1.2) "Fluctuation of currency
        exchange rates";
  Components.Shipping shipping(
      shipping_cost=1);
equation
  connect(foreign_consumer.market,
      foreign_producer.market);
  connect(domestic_consumer.market,
      domestic_producer.market);
  connect(trade.market_B,
      domestic_producer.market);
  connect(fluctuations.y, trade.xrate);
  connect(shipping.remote, trade.market_A);
  connect(shipping.local,
      foreign_producer.market);
```

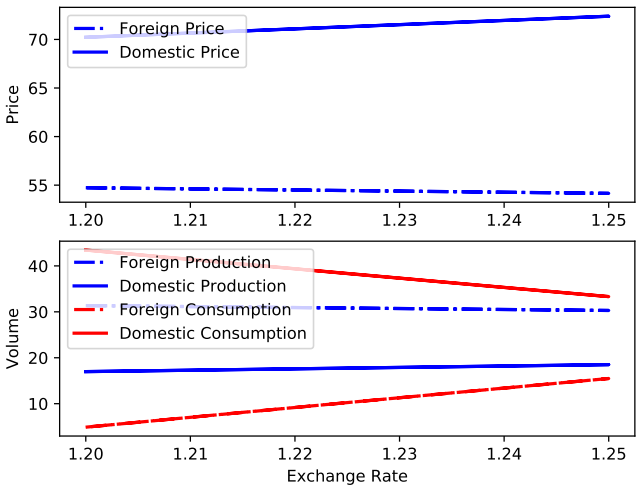

Figure 12. Price and Volume vs. Exchange Rate

```
end InternationalTrade;
```

In this case, the shipping model adds $1 to the cost of any good that moves between the markets. Similarly, the `trade` block handles the currency conversion and imposes a tariff of 5% on goods moving from the foreign market (top) to domestic market (bottom).

In this particular model, there is no `MarketAnalysis` block. Instead, the system is always in equilibrium. However, there is a time varying component to this model because the curreny rate fluctuates. This gives us a chance to see how the currency rate influences both consumers and producers in both geographies.

Figure 12 shows how consumers and producers respond to changes in exchange rate. Specifically, what we see is that as the exchange rate increases, the prices in the "domestic region" (at the bottom of Figure 11) rise. As a result, we can see that while domestic production rises only slightly and foreign production drops slightly (the blue lines in bottom plot of Figure 12), consumption of goods domestically drops while there is a corresponding rise in consumption by foreign consumers. In other words, as the exchange rate rises, the foreign produced goods become less attractive to the domestic market. Since fewer foreign goods are being shipped away from their home market, the effective demand drops and so does the price in the foreign markets which triggers foreign consumers to purchase more.

6 Future Directions

There are a number of additional effects it would have been interesting to model but time and space constraints prevented a deeper study of the topic. In particular, we did not look into market dynamics. There are no states in these models. While some conditions may change (*e.g.*, the exchange rate in the last example), the solutions are still strictly algebraic.

DOI 10.3384/ecp19157365

One could imagine adding dynamics in several ways. First, the output of producers could respond (with some lag) to increasing demand and the potential to increase revenue through higher volumes of sales (at cheaper market prices). Furthermore, we haven't examined the impact of stockpiling of resources when prices are relatively low for the purpose of reselling them when they are higher.

These models point out a few limitations in Modelica as well. The first is that we might accidentally mix different goods with an erroneous connection. It would be useful if we could somehow parameterize the various models in terms of the underlying types of goods so that a consumer of light bulbs couldn't accidentally be connected to a producer of turbine blades. It isn't clear how to formulate these models so that such mistakes could be statically detected. One approach would be to establish the type of good as a `parameter` on the `Market` connector. However, this approach would require setting these parameter values all over the place unless we adopted an approach similar to how fluid models leverage media models.

Another limitation is related to how supply and demand curves are expressed. Currently, these curves are instantiated once by each consumer and producer model. However, initial attempts to model market segmentation (*e.g.,* different classes of consumers with different price sensitivities) suggested the potential value of being able to export these curves so that other models could instantiate them for their own purposes. The closest physical analogy would be how some vehicle dynamics libraries express the elevation of a road surface such that each tire of the vehicle can independently query the road for its elevation. Such capabilities might allow the calculation of economic metrics like deadweight loss, *etc.*

Finally, more complex economic models will almost certainly require the need to express constraints and objectives along the lines of what is expressed in Modelica extensions like Optimica (Åkesson 2008). With such expressiveness a Modelica compiler may compile such economic models into general optimization problems or perhaps specialized linear programs.

7 Conclusion

In conclusion, this paper discusses how to formulate typical supply and demand curves as reusable component models. These component models can then be connected together to simulate the behavior of a market. Modelica semantics ensure that each entity in the market uses a consistent price and that all goods are properly accounted for. Additional components have be defined to model the effects of taxation, transportation, tariffs, manufacturing, *etc.*

Although the models here are not meant to be a complete or rigorous approach to modeling supply and demand systems, hopefully the discussions here will provide a reasonable starting point for further development. Furthermore, the content of this paper could assist those intereste in economic models but unfamiliar with Modelica in creating economic models in Modelica. The models described in this paper are provided under an MIT open source license and can be found at `https://github.com/mtiller/EconomicsLibrary`.

References

Åkesson, Johan (2008). "Optimica - An Extension of Modelica Supporting Dynamic Optimization". In: *Proceedings of the Modelica Conference, 2008*. URL: https://www.modelica.org/events/modelica2008/Proceedings/sessions/session1b3.pdf.

Casella, Francesco, Giovanni Miragliotta, and Luigi Uglietti (2005). "Analysis of Supply Chain Dynamics through Object Oriented Simulation". In: DOI: https://doi.org/10.1007/3-7908-1636-1_30.

Hutchinson, Emma (2016). *Principles of Microeconomics*. OpenStax College. URL: https://pressbooks.bccampus.ca/uvicecon103/.

Modelica Association (2017). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.4*. Tech. rep. Linköping: Modelica Association. URL: https://www.modelica.org/documents/ModelicaSpec34.pdf.

Posner, Barry and Farid Tayari (2018). *Introduction to Energy and Earth Sciences Economics*. Penn State. URL: https://www.e-education.psu.edu/ebf200.

Taylor, Timothy (2018). *Principles of Economics*. OpenStax College. DOI: 10.24926/8668.1601. URL: https://doi.org/10.24926/8668.1601.

Zimmer, Dirk and Daniel Schlabe (2012). "Implementation of a Modelica Library for Energy Management based on Economic Models". In: *Proceedings of the 9th International Modelica Conference*, pp. 133–142. DOI: 10.3384/ecp12076133. URL: http://www.ep.liu.se/ecp/076/012/ecp12076012.pdf.

DOI
10.3384/ecp19157365