Modelling a Dynamic Economy using Nodal Analysis

Shadow_slicer

March 12, 2008

Abstract

The current vegastrike economy is statically defined. Events in the universe do nothing to affect the economy. Additionally planets seem quite similar, so that all planets of a given type seem economically identical.

1 Introduction

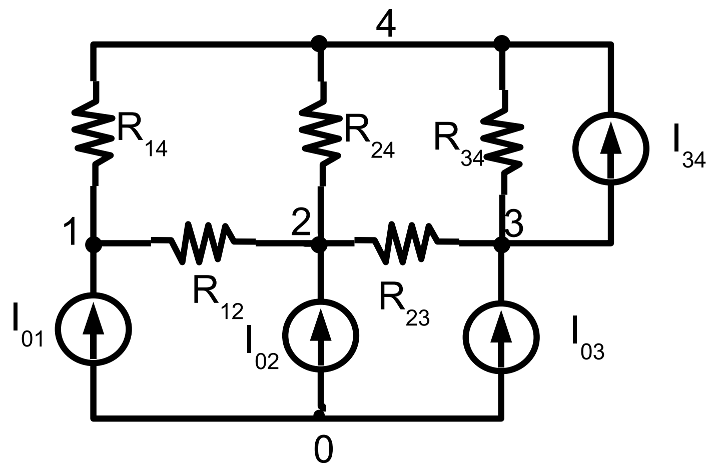
The current vegastrike economy is static. The basic characteristics of each type of planet is hardcoded, so all the planets regardless of their location, current factional ownership, presence or absence of sieging units forming a blockade, proximity to other planets or bases that produce or consume a commodity have the same price and amount distributions for every available commodity.

The only scalable solution to solve the economy problems is a dynamic economy. It is quite clear that the current static economy will have to suffice for the 0.5 release, but for the 0.6 release we could start planning a dynamic economy.

1.1 Nodal Analysis

Nodal analysis is a technique commonly used to solve circuits to find unknown voltages. The circuit is assumed to consist of resistors and current sources. The basis of this method is Kirchhoff's current law which states that at each node the current entering the node is equal to the current leaving the node. The current from the current sources is known. So a matrix equation is set up to solve for the voltages.

For example take the following circuit:



The matrix equation for the voltages in the circuit is below. R12(V1-V2) + R14(V1-V4)

$$\begin{bmatrix} \frac{1}{R_{12}} + \frac{1}{R_{14}} & -\frac{1}{R_{12}} & 0 & -\frac{1}{R_{14}} \\ -\frac{1}{R_{12}} & \frac{1}{R_{12}} + \frac{1}{R_{24}} + \frac{1}{R_{23}} & -\frac{1}{R_{23}} & -\frac{1}{R_{24}} \\ 0 & -\frac{1}{R_{23}} & \frac{1}{R_{23}} + \frac{1}{R_{34}} & -\frac{1}{R_{34}} \\ -\frac{1}{R_{14}} & -\frac{1}{R_{24}} & -\frac{1}{R_{34}} & \frac{1}{R_{14}} + \frac{1}{R_{24}} + \frac{1}{R_{34}} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} I_{01} \\ I_{02} \\ I_{03} - I_{34} \\ I_{34} \end{bmatrix}$$

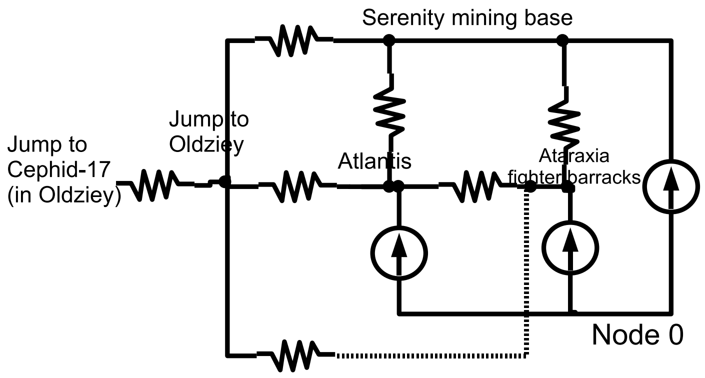
This matrix can be systematically generated based on the network. Basically if there is a resistor linking node i and node j, put $\frac{1}{R_{ij}}$ in position (i, i) and (j, j) in the resistance matrix, and put $-\frac{1}{R_{ij}}$ in (i, j) and (j, i). If the resistor connects to node 0, however, only put $\frac{1}{R_{i0}}$ in (i, i). Then for the corresponding current, sum the currents entering the node. For capacitors, you normally use the laplace transform and treat the equivalent resistance, $\frac{1}{sC}$, as if it were a resistance.

2 Modelling a Dynamic Economy using Nodal Analysis

The basic idea is to represent the supply and demand as the currents entering or leaving the node (from and to points unknown). Then the distances and conditions in between can determine the inter-node resistances. We can solve for the voltages which could be scaled to create reasonable prices. This will allow prices to consider global conditions (faction relations, presence of asteroids, sieges, or pirates, etc.). There are a few possible ways to represent the universe and solve for prices using nodal analysis.

2.1 Every base as a node without memory

Let every planet, base, and jumpoint in every system be a node. Nodes in the same system will all be completely connected (i.e. every node connected to every other node). The only links between systems would be a single resistance between the corresponding jumppoint nodes.



2.2 Every base as a node with memory

We will probably want the model account for the reserve supply of goods that planets and bases keep. Since in our current model the transport of goods is represented as a current, the storage of goods would be represented by a stored charge. This means the warehouses on planets and bases would be modeled as a capacitor between that base and node 0. Probably we should model planets as having a large capacitance since they presumably have significantly more storage space than orbiting bases.

The typical method of solving this problem as a circuit uses the Laplace transform which complicates the equations a little (adds 1/sC terms to the diagonal). That method allows continuous evaluation of the solution. In our case we don't need that, and only really require an estimate at certain times.

For this discrete time case we can alter the resistance matrix by adding a C_i/dt term to the diagonal. Then we can account for the current supply by adding Q/dt to the supply current. dt is the simulation step size. This should be the same as the time step for productions of commodities. Note that after taking the inverse of the resistance matrix, these update computations would simply involve a matrix multiplication since the only term that depends on the present stockpile of goods is in the I term.

2.3 Hierarchial simplification

If the cost of jumping is significantly larger than in-system travel, we can use an approximation to convert the global model into hierarchial one that scales significantly better. We divide the problem into two pieces.

In the first piece, every system is node whose input current is the sum of the supplies and demands inside the system (and whose reserves and capacitances are also equal to the sum). The resistances between them will need to be equal to the cost of jumping plus some cost representing the cost of traveling through the system (due to the presence of pirates, asteroids, fleets of hostile factions, etc). These systems could then be simulated as in the previous two methods.

For the current (and possibly adjacent) systems we can use one of the previous more detailed models, only with the jumpoints connected to current sources with a current calculated based on the previous method instead of the adjacent systems. This allows us to simulate the universe at a lower resolution while still having higher resolution nearby.

2.4 Splitting the problem

Even with the hierarchal simplification above, the matrices will still be quite large. It turns out it is not necessary to simulate the entire universe at a time. We can use the above method to simulate a region consisting of (some number of sectors) at a time. The links to and from sectors outside the simulated region can be replaced by current sources equal to the estimated commodity flow across these links. If we use overlapping regions, and simulate the entire network a sufficient number of times, we should achieve the same answer as in Section 2.3. It is not necssary to take the matrix inverse for each iteration, since the only change will be the currents existing on the border. Note that it is critical that the regions overlap. This ensures that the flow across every link will eventually be updated.

This approach may result in some numerical instability if the memoryless model listed in Section 2.1 is used.

3 How to solve the equation

These equations can be solved through several different methods. The 'Every base is a node' methods form nearly diagonal matrices so Gaussian elimination would be relatively quick. In spite of that, I would propose an iterative algorithm to be more effective. Note that the resistances matrices for each good could be slightly different if contraband is treated differently by different factions or other things that affect some goods but not others. This means that this equation will need to be solved for each class of goods. Since each of the resistance networks will be similar the inverse of the resistance matrix should also be similar. This means that if we use an iterative algorithm with the starting guess equal to the answer previously calculated for another good we will likely only need to do a few iterations before converging.

Also the matrix has a well defined structure, so there may be a specialized method that results in less computation.

4 When to solve the equation

The economy shouldn't change that rapidly, so it should be enough to simply do it every time the player docks. I expect it to finish rapidly, and vegastrike does a lot less work when the player is docked so I expect the impact will be low.

Or vegastrike to continuously update the economy. If an iterative algorithm is used, the economy could run in a separate thread, slowly improving its price estimates and reacting to changing conditions. Thought this is probably not necessary.

5 Implementation Issues

5.1 Ensuring Stability

Regardless of the model used, if the supply and demand do not exactly match, the system will become unstable and prices will fall to the minimum or rise to the maximum. In a real economic system there is some spoilage, (i.e. more goods are produced than consumed because some goods go to waste). In order to ensure stability a large resistance R_s could be added to each node to allow extra current to disappear.

5.2 Memory Usage

Note that we would need to solve this problem for each of the commodities in the network. Many commodities would use the same resistance network, so they could be grouped and the solution reused. Inspite of this reuse the matrices themselves can be quite large.

In vegastrike there are some 5700 star systems modeled in the universe. If each system has 5 bases/planets and 5 jumppoints there would be a total of 57000 nodes in the most complex model (from Section 2.2. This would mean this approach would require inverting a 57000x57000 matrix. If composed of 8 byte floats, this matrix would take up 25 gigabytes of storage. But most of this matrix would be 0. If we only store the non-zero values, it would take up about 4.6 megabytes. There is also a large amount of symmetry in the matrix, so we could reduce it to almost half of that. The inverse of the matrix could potentially take 25 gigabytes to store, but due to the structure of the problem it is likely to be about the same size as the resistance matrix.

If we use the simplified system model from Section 2.3, we would have only 5700 nodes, so the full matrix only takes 250 megabytes. If we only store the non-zero values, we will have 222 kilobytes. For each system we simulate in detail, we have 800 bytes for the full matrix.

If we use the progressive simulation mode from Section 2.4, the matrix size depends on the size of the simulated region. If the region contains 400 systems, the full matrix will take 1.22 megabytes. Only the nonzero values would make up 15.6 kilobytes. In order to simulate enough regions to cover the entire universe, we need to invert 42 of these matrices, and perform numerous iterations of matrix multiplications for the universe to converge to the same answer as in Section 2.3

6 Conclusion

A dynamic universe needs a dynamic economy. Modeling the economy as a network of resistors and solving with nodal analysis is a fairly simple yet generic approach that is capable of achieving all of the features desired in a dynamic economy. It is scalable to arbitrary sizes, and with the method described in Section 2.4 can be restricted to O(n) memory usage.