Optimally applying latency insertion method in large system models

Yiwei Zhang¹, Rod Leonard², Jijun Tang¹, Roger Dougal²
¹Department of Computer Science and Engineering
²Department of Electrical Engineering
University of South Carolina
{zhang33, leonard, jtang, dougal}@cec.sc.edu

Keywords: Latency insertion method, optimal partition, large system model

Abstract
Latency Insertion Method (LIM) can be applied to divide a large system model into a multiplicity of smaller models to reduce simulation execution time. In this paper we have developed an algorithm for optimal placement of LIM connectors into a large system model so that the large model can be partitioned into many smaller models to achieve maximum speed up. Testing results showed that for reference system models having structures similar to electric power distribution systems, simulation speed was increased by a factor of more than three while keeping the error rate within 1%.

1. INTRODUCTION
The conservation of energy and momentum are laws of nature with which models can be physically connected to one another. Certain techniques of electrical circuit theory rely on these conservation equations for the mathematical solution of networks. Numerous other relationships, including mechanical, fluid, thermal and rotational conservation techniques, are analogous to electrical circuit theory. Therefore, many natural models can be developed using the Resistive Companion (RC) [1] technique and a system that is formed by the interconnection of these natural components can be characterized by a single large conductance matrix. The simulation of the system is carried out by solving the conductance matrix.

It has always been difficult to simulate a large system for the following reasons. First of all, the conductance matrix corresponding to the system model usually has a large dimension. Moreover, if the system contains non-linear components, the conductance matrix becomes time-dependent which means the matrix should be calculated and updated during each time step. Second, small time steps are required to compute accurate results which further demands longer simulation time. In conclusion we can see that simulation is extremely time-consuming for large systems.

Recently, an explicit simulation approach called Latency Insertion Method (LIM) [2] has been proposed to divide a large system model, characterized by a single large conductance matrix, into a multiplicity of smaller models. Originally LIM was only applied to circuits system, but later its characteristics have been exploited on other natural models. So it is possible to apply LIM to solve other kind of natural systems. Lalgudi [3] describe how LIM could be applied for on-chip power-grid simulation, but in his work he manually applied the LIM method to break the system at every node. Meanwhile, Moreira [4], working with electric power systems, manually inserted LIM objects at hand-picked locations. This approach was not amenable to automation and it is very difficult for a user to find optimal places in a large system without any assistance. Our work moves beyond these two by introducing a method that automatically exploits actual latencies in a large network model where LIM method can be applied. First we employ a method that automatically checks the system to see where LIM can be applied to partition the system. Then we introduce a method to estimate the speed up and make the optimal partition that has the maximum speed up.

We have applied this method, in the VTB (Virtual Test Bed) environment, to the simulation of several large-sized systems with linear and non-linear components. Depending on the actual structure of the system, the speed up varies, and we will show the simulation result on a large ship electrical power system.

2. PARTITIONING LARGE SYSTEM

2.1. Latency Insertion Method
The LIM is developed to for time-domain simulation of large scale networks. In this method, a finite formulation is used to update branch currents and node voltages in a leapfrog manner. The LIM is readily enabled by insertion of latency or exploration of latency in the networks. A network has latency if a node in it has a shunt capacitance to ground and each branch in it has a series inductance. In fact, such parasitic elements exist in every network, but many times they have vanishingly small manifestations. In other cases, such as circuits that involve transmission lines, these
parasitic impedances have values that are entirely appropriate for exploitation via latency methods.

In Figure 1, a piece of a circuit is shown for which the LIM can be applied. The symbols in Figure 1 mean the following: $i$ and $j$ refer to the nodes; $R_{ij}$ and $L_{ij}$ are the series resistance and inductance of the branch between nodes $i$ and $j$, respectively; $C_{ii}$ and $C_{jj}$ refers to the shunt capacitance from node to ideal ground; $V_i(t)$ and $V_j(t)$ refers to the voltage at node $i$ and $j$; $i_{ij}(t)$ refers to the current in the branch between nodes $i$ and $j$ and $i_{ij}(t)$ refers to the current from node $i$ through capacitor $C_{ii}$ to ground. The leapfrog scheme is a second-order integration method to solve differential equations. This scheme relies on staggering the voltages and the currents by half a time step. The leapfrog scheme is referred to as a semi-implicit scheme in [5].

![Figure 1. Typical equivalent circuit to enable LIM](image)

The newly calculated branch current will be used for the computation of next time step and the above operations will be iterated.

In the LIM, the transient simulation is accomplished by updating the node voltages and the branch currents at each time step. The conceptual equivalent circuit looks as shown in Figure 2. The original circuit now can be viewed as been separated. At time $n\Delta t$, the sub-systems connected at each side can be solved individually. At time $(n+1/2)\Delta t$, voltage at node $i$ is given by:

$$v_i^{n+1/2} = v_i^{n-1/2} + \frac{\Delta t}{C_{ii}}i_{ii}^n$$

(1)

And then the branch current at time $(n+1)\Delta t$ can be derived as following that is based on previous calculation:

$$i_{ij}^{n+1} = \frac{1 - \frac{R_{ij}\Delta t}{2L_{ij}}}{1 + \frac{R_{ij}\Delta t}{2L_{ij}}}i_{ij}^n + \frac{\Delta t}{L_{ij}}(v_{ij}^{n+1/2} - v_{ij}^{n-1/2})$$

(2)

The newly calculated branch current will be used for solving the sub-systems at $(n+1)$th time step.

![Figure 2. Conceptual equivalent circuit with LIM](image)

### 2.2. Method to partition the system

#### 2.2.1. Make a LIM connector

We use VTB (Virtual Test Bed) system to perform real time simulation of systems. A system is made up of a series of nodes connected by components. Since the simulation is performed based on the solution of resistive companion equations, VTB creates the conductance matrix in terms of the conductance stamps of each component in the system. At each time step, it solves the resistive companion equations corresponding to the conductance matrix.

In the first step, we developed a subsystem connector that applies the LIM. The LIM subsystem connector is a two-port element having a set of natural ports on each side. But unlike other two-port elements having natural ports, each of the two ports can appear in separate subsystems that are independently solvable per time step.

Since some existing component models, like RLC cable, are appropriate for exploitation via latency methods, we implement an additional decoupling interface for such components so that we can identify these components during partition. If we finally find that LIM method shall be applied for the component, we shall replace the component with a LIM connector.

If the system is successfully partitioned into several sub-systems, then each LIM connector will be assigned the initial capacitor voltages and branch currents. During each time step of the simulation, VTB will solve each sub-system separately with the branch current of the LIM connectors. After all the sub-systems are solved at $n$th time step, each LIM connector will re-calculate the half step voltages of the capacitors according to (1), update its branch current based on the half step capacitor voltage and branch current at $n$th time step according to (2). The updated branch current shall be used for solving the sub-systems at $(n+1)$th time step.

#### 2.2.2. Automatically partition the system

We introduce a method that employs a Breadth-First-Search algorithm that is familiar in graph theory. It examines the whole system and checks all paths of connectivity between adjacent nodes. If two nodes are connected only by components which have LIM equivalents, then a LIM connector is applied to substitute the original component and these two nodes now belong to...
different sub-systems. Otherwise they are grouped together. This algorithm will automatically identify places that LIM can be applied and decouple the original system into sub-systems. The procedure is as follows:

- Initialize each node as unmarked
- For each node in the system
  - If the node is not marked
    - Mark the node
    - Create a sub-system and put the node into it
    - Get the first node in the sub-system
- While current node is not NULL do
  - For each node connected to current node
    - If the nodes are suitable for LIM connection
      - Apply LIM connector
    - Else
      - Add the node to the sub-system
      - Mark the node
  - End of if
  - End of for
- End of while
- End of if
- End of for

When the algorithm finishes, all the nodes in the system will be divided into several sub-systems that are connected by LIM connectors. If the ports of the LIM connector are in the same sub-system, then this LIM connector will be abandoned since it cannot break the system.

If a sub-system contains any non-linear components, then the conductance matrix of that subsystem will vary with time and the matrix inverse will need to be computed at each time step. But if all the components within a sub-system are linear, then the conductance matrix needs to be inverted only once, before the first time step, so there cannot be any appreciable increase of performance by partitioning linear subsystems. Therefore, if two adjacent sub-systems connected with LIM are both linear, our algorithm merges these two sub-systems together.

2.2.3. Optimally select the LIM connectors

Mathematical stability requires that the time step is no larger than specified by (3), in terms of the inductance and capacitance of the LIM connector:

$$\Delta t \leq \sqrt{\frac{1}{LC}}$$

(3)

For a single rate solver, the global minimum time step must satisfy this criterion for all the LIM connectors. If this upper bound of time step is smaller than the time step that could be used in the un-partitioned system, then the overall simulation of the partitioned system might actually take longer than simulation of the un-partitioned system. So we shall keep the LIM connectors that can achieve maximum speed up.

The majority of the computation time is used for matrix inversion. During each time step, the inversion of the system conductance matrix is calculated first. If the system contains non-linear components, then a loop is performed to iteratively solve the system until convergence is achieved. So the matrix inversion needs to be performed each time the conductance matrix getting updated during the iteration. The speed up can be estimated as following:

$$\text{SpeedUp} = \frac{N^k + M \cdot N^k}{\sum N_i^k + \sum M \cdot N_i^k}$$

(4)

Where \(N\) is the number of nodes of the original system and \(N_i\) is the number of nodes of each non-linear sub-system. \(M\) can be estimated by the maximum number of iterations needed to reach convergence. \(N^k\) stands for the inversion time of the conductance matrix. In our simulation system inversion of a sparse matrix has the complexity of \(O(N^{2.3})\) so the value of \(k\) is about 2.3.

If the \(\sqrt{LC}\) value of certain LIM connector is smaller than the original simulation time step, then the estimation of speed up shall be modified. Let \(\Delta t\) be the original simulation time step, \(\Delta t\) be the minimum time step of LIM connector’s \(\sqrt{LC}\) in the system. The actual speed up can be calculated as follows:

$$\text{SpeedUp} = \frac{N^k + M \cdot N^k}{\sum N_i^k + \sum M \cdot N_i^k} \cdot \frac{\Delta t}{\Delta t'}$$

(5)

For each LIM connector in the system whose value of \(\sqrt{LC}\) is smaller than the original simulation time step, we can choose to keep it or abandon it which would lead to different speed up. The following algorithm gives a fast way to optimally select the LIM connectors to get maximum speed up:

- Calculate the initial minimum time step needed
- Calculate initial speed up according to (5)
- Loop until the minimum time step needed is larger than actual simulation time step
- Remove the LIM connectors with least time step
- Merge the sub-systems connected by these LIM
- Recalculate the minimum time step needed
- Recalculate current speed up
- If current speed up is larger than previous speed up
  - Record current LIM connectors
  - Record current speed up
- End of if
- End of Loop
- Keep the LIM connectors that produce maximum speed up
3. SIMULATION EXPERIMENT

We use a notional ship electric power system to conduct the simulation experiments. This system contains 648 nodes and 1574 components. Most of the components are linear, but it contains some complicated non-linear components such as generators and motors.

As mentioned above, currently the only methods to apply LIM is either manually finding set of nodes to insert LIM connector or inserting LIM connector between every possible set of nodes. Apart from the above two approaches, there are no other automatic decoupling methods to insert LIM connectors. So we compare the performance of our automatic decoupling method with these two methods.

Figure 3. The schematic of a ship electric power system

3.1. System decoupling result

The system contains many power cables that are represented as a series combination of a resistance and an inductance. These are possible places to substitute with LIM connectors because cables already contain latency. Also a cable is usually the only component that connects set of nodes, so applying LIM here might be able to decouple the original system.

The partition made by our algorithm produced 7 sub-systems. In Figure 3, the solid rectangles show the decoupling points selected by our method. Table 1 shows the information about the decoupled sub-systems.

Table 1. The statistics of the automatically decoupled sub-systems

<table>
<thead>
<tr>
<th>Sub-system</th>
<th>Number of nodes</th>
<th>Minimum time step</th>
<th>Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18</td>
<td>0.00015s</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>0.00015s</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>524</td>
<td>0.0001s</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>0.0001s</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>0.0001s</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>0.0001s</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>22</td>
<td>0.0001s</td>
<td>No</td>
</tr>
</tbody>
</table>

As comparison to our algorithm, we asked a power systems expert to hand place decoupling points according to his experience. He separated all the subsystem loads from the main grid. This is a logical breaking point that people might try. In Figure 3, the dashed rectangles show the manually selected decoupling points. The manual decoupling produces 8 sub-systems and Table 2 shows the information about manually decoupled sub-systems.

Table 2. The statistics of the manually decoupled sub-systems

<table>
<thead>
<tr>
<th>Sub-system</th>
<th>Number of nodes</th>
<th>Minimum time step</th>
<th>Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>209</td>
<td>0.00018s</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>117</td>
<td>0.0001s</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>117</td>
<td>0.0001s</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>117</td>
<td>0.0001s</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>0.0001s</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>0.0001s</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>22</td>
<td>0.0001s</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>0.0001s</td>
<td>No</td>
</tr>
</tbody>
</table>

The difference between the automatic method and manual method is that, the automatic method keeps the linear section of all subsystem loads connected together so that the size of non-linear subsystem can become as small as possible. On the other hand it is not easy for a user to find out all the linear components in a large system and combine them together as much as together.

A third method is to apply decoupling at every possible cable, and we get a total of 140 sub-systems. Most of these sub-systems are linear and they are actually merged together...
in our automatic method. Then we shall compare the performances of all these decoupling methods.

3.2. Simulation time comparison

The original simulation time step was 0.001 second and the total simulation interval was 1 second. First we ran the simulation without applying our method. Then we ran the simulation with our method and other two methods. Table 3 shows the simulation time cost of each method.

Table 3. Simulation time comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>Time cost (s)</th>
<th>Speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without LIM</td>
<td>560</td>
<td></td>
</tr>
<tr>
<td>Automatic decoupling</td>
<td>165</td>
<td>3.4</td>
</tr>
<tr>
<td>Manually decoupling</td>
<td>428</td>
<td>1.3</td>
</tr>
<tr>
<td>Decoupling at every cable</td>
<td>243</td>
<td>2.3</td>
</tr>
</tbody>
</table>

Although the minimum time step needed by our automatic partition is 0.0001 second, we can still get nearly 3 times speed up according to equation (5). However, larger sizes of non-linear sub-systems are generated based on manual decoupling, so manual method produces smaller speed up compared with automatic method.

Ideally decoupling at every cable should have the same speed up as automatic method because they output same number and sizes of non-linear sub-systems. But decoupling at every node would generate too many small linear systems. After each time step, there are post processes for each sub-system to update and exchange some information between each other, so decoupling the original system into too many small sub-systems would increase the extra time of post processing. As a result, the actual speed up of this method is not as good as the automatic decoupling method which outputs a very small number of sub-systems.

3.3. Simulation accuracy

The error rate is based on the relative root mean square error (RMSE) of the voltages at decoupling point. $V_1$ is the voltage of the un-decoupled system and $V_2$ is the voltage of the decoupled system. $N$ is the total number of time steps.

$$\text{ErrorRate} = \sqrt{\frac{\sum_{i=1}^{N} (V_1(i) - V_2(i))^2}{\sum_{i=1}^{N} V_1(i)^2}}$$

(5)

Table 4 shows the average error rate of all the decoupling point under different decoupling method. We can see that automatic decoupling method produces quite small error rate. Since the error rate is very low at every decoupling point, we are confident that the average error rate at all the other nodes should also be very low which ensures high accuracy of automatic decoupling method. Also we can see that the accuracy of manually decoupling method is very close to our method. On the other hand, decoupling at every cable produces much higher error rate than the automatic method. It is clear that applying LIM method would introduce small error at the decoupling point. So adding number of LIM connectors would also increase the error rate which means inserting LIM connectors at every possible node would have a larger error compared with our method that inserts much smaller number of LIM connectors.

Table 4. Simulation error rate

<table>
<thead>
<tr>
<th>Method</th>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic decoupling</td>
<td>0.86%</td>
</tr>
<tr>
<td>Manually decoupling</td>
<td>0.87%</td>
</tr>
<tr>
<td>Decoupling at every cable</td>
<td>4.51%</td>
</tr>
</tbody>
</table>

To illustrate the accuracy, we also compared the voltage waveform of the motor in the sub-system near decoupling point. Figure 4 shows the position of the component.

Figure 4. The arrow points to the generator near decoupling point whose current is compared

The upper part of Figure 5 shows the comparison of the plotted waveform of the original system and of the decoupled system at every cable. We can see that there is relatively large difference between them. The error rate is actually about 3%. The lower part shows the waveform comparison of the original system and of the system automatically decoupled by our method. We can see that the simulated waveform of our method is very consistent with the original simulation result. The error rate is actually less than 0.9%.
4. **CONCLUSION**

This paper presents an algorithm for optimal placement of LIM decouplers to partition a large system automatically. Experiment results show that high speed up can be achieved when using this method while the accuracy is still satisfying.

Our method focuses on finding components in the system that can apply LIM and satisfy the overall speed up demand. So besides the circuits system, our automatic method can also be applied to decouple other types of natural systems to achieve maximum speed up if user has created LIM equivalent for other type of components.

5. **REFERENCES**


**ACKNOWLEDGEMENTS**

This work was supported by the US Office of Naval Research under contract N00014-07-1-0686.