# 18.337 Final Report
# Parallel Implementation of a Multi-Length Scale Finite Element Method

Trevor Tippetts

May 8, 2003

## 1 Introduction

For years the finite element method (FEM) has held a dominant position in the field of computational structural analysis. As finite element users have demonstrated an insatiable appetite for computational resources, many software developers have tried various approaches to parallelize finite element simulations. Most have focused on distributing the system assembly and/or solution algorithm across multiple processors.

This usually leads to a situation in which the finite element model and the parallel algorithms for solving it place constraints on each other that are not necessary for achieving an acceptable solution. For example, for two points in the physical structure that are separated by large distance, small length scale field behavior near the two points typically has little interaction. Additionally, in many cases a representation of a structure at different length scales seems inherent to the problem, as is the case with composite materials.

However, if a finite element fomulation is developed for serial computation and later solved with a parallel algorithm, interprocessor communication representing this small-scale interaction is required. This interprocessor communication is potentially much more expensive than necessary to accurately model the physical structure.

In contrast, this attempt takes a step back to the derivation of the finite element governing equations. The variational equilibrium equations are recast with an awareness of the subsequent parallelization. The result will be a finite element model that is more amenable to parallel computation.

## 2 Multi-length scale finite element method

The multi-length scale finite element method (MLSFEM) is a method for deriving a system of finite element equations with degrees of freedom and interpolation functions on more than one length scale. This multi-length scale representation of the fields can give great computational advantages. Of particular importance for parallel compution is the potential for greatly reduced system bandwidth, which decreases the need for interprocessor communication. This is possible because only large-scale field information is passed between processors. With MLSFEM, some accuracy is lost in comparison to a monolithic fine mesh. However, for many simulations this lost accuracy is minimal, and the improved computational efficiency more than makes up for this disadvantage.

A second, perhaps more subtle, advantage is the capability to use different integration schemes in time for dynamic problems. For example, the large scale degrees of freedom can be integrated in time with an explicit integration method, while the small scale degrees of freedom would be integrated implicitly. This would allow the maximum time step to be limited by the Courant condition on the coarser large scale mesh rather than on the small scale mesh, greatly decreasing the number of required time steps.

## 2.1 MLSFEM formulation

A typical displacement-based finite element formulation is derived from a potential functional, $\Pi$, which is a functional of the diplacement fields. The displacement fields, $u$, are determined by the shape functions, $n_i$, and the degrees of freedom, $q_i$.

$$u = n_i q_i \tag{1}$$

This functional is minimized with respect to the degrees of freedom.

$$\delta\Pi = \frac{\partial \Pi}{\partial q_i}\delta q_i = 0 \tag{2}$$

Because the variations $\delta q_i$ are arbitrary, a set of $i$ equilibrium equations must be satisfied.

$$\frac{\partial \Pi}{\partial q_i} = 0 \tag{3}$$

In the two-scale MLSFEM formulation, the displacement fields are determined by both global ($Q_j$) and local ($q_i$) degrees of freedom and their corresponding shape functions as simply the sum of the two fields.

$$u = N_j Q_j + n_i q_i \tag{4}$$

The equilibrium equations are therefore

$$\frac{\partial \Pi}{\partial Q_j} = \frac{\partial \Pi}{\partial q_i} = 0. \tag{5}$$

A constraint must be applied to make the fields for the two scales independent. Otherwise, the system matrices would be singular.

$$\int (N_j Q_j)(n_i q_i) dV = 0 \tag{6}$$

The implementation of this finite element formulation, is greatly simplified by defining a field on the small length scale as the sum of the two fields.

$$\tilde{u} = U + u = \tilde{n}_k \tilde{q}_k \tag{7}$$

The equilibrium equations then allow the problem to be solved as a single-scale finite element model in the new degrees of freedom.

$$\frac{\partial \Pi}{\partial \tilde{q}_k} = 0 \tag{8}$$

The local fields are coupled to each other through the constraint (Equation 6), which is expressed with the new degrees of freedom.

$$\int (N_j Q_j) \left( (\tilde{n}_k \tilde{q}_k) - (N_j Q_j) \right) dV = 0 \tag{9}$$

It is convenient for both accuracy and computational efficiency to linearize Equation 9 with respect to $Q_j$. This yeilds a linear system of equations that can be implemented in finite element code as multiple point constraints.

$$A_{mj} Q_j = B_{mk} \tilde{q}_k \tag{10}$$

$$A_{mj} = \int N_m N_j dV \quad B_{mk} = \int N_m \tilde{n}_k dV \tag{11}$$

In this way, a model with two length scales can be solved as two single-scale models. The interaction between the two is accomplished with multiple point constraints. This allows a MLSFEM developer to use existing finite element source code to speed the software development.

## 2.2 Implementation

The MLSFEM is implemented in the finite element program CalculiX[1]. CalculiX is an open source serial FE code with which the author is already familiar. The MPI library is used for interprocessor communication between multiple identical intstances of the program.

I implemented module in the finite element code that computes the constraint matrices $A$ and $B$. The code takes a model definition of the superelement and the subelements in the same format as a finite element simulation and performs the integrations of the shape functions over the volumes as shown in Equation 11. It then outputs the elements of the constraint matrices in the form of coefficients in multiple point constraints that can be read in an input file by the finite element code. An example of a multiple point constraint equation is shown in Listing 1. The format of the listing will be recognized by users of the finite element codes ABAQUS$^{\text{TM}}$or CalculiX. The listing means $0.0138888889 q_{11}^1 + 0.0370370370 q_1^1 + \ldots = 0$, where $q_{11}^1$ is the displacement in direction 1 at node 11, etc.

[1]See http://www.calculix.de/ and http://www.dhondt.de/

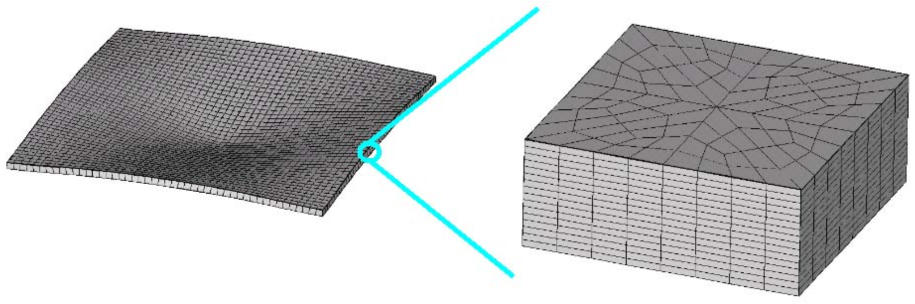


Figure 1: Schematic of MLSFEM. Each superelement is associated with a set of subelements.

**Algorithm 1** Example multiple point constraint equation.

```
*EQUATION
40
11,1, 0.0138888889, 1,1, 0.0370370370, 2,1, 0.0185185185, 3,1, 0.0092592593,
4,1, 0.0185185185, 5,1, 0.0185185185, 6,1, 0.0092592593, 7,1, 0.0046296296,
8,1, 0.0092592593, 12,1, 0.0127314815, 13,1, 0.0092592593, 14,1, 0.0127314815,
15,1, 0.0185185185, 16,1, 0.0162037037, 17,1, 0.0115740741, 18,1, 0.0162037037,
19,1, -0.0231481481, 20,1, -0.0115740741, 21,1, -0.0115740741, 22,1, -
0.0231481481,
23,1, -0.0277777778, 24,1, -0.0138888889, 25,1, -0.0138888889, 26,1, -
0.0277777778,
27,1, -0.0277777778, 28,1, -0.0138888889, 29,1, -0.0069444444, 30,1, -
0.0138888889,
35,1, 0.0046296296, 36,1, 0.0034722222, 37,1, 0.0023148148, 38,1, 0.0034722222,
43,1, -0.0046296296, 44,1, -0.0023148148, 45,1, -0.0023148148, 46,1, -
0.0046296296,
47,1, -0.0092592593, 48,1, -0.0046296296, 49,1, -0.0023148148, 50,1, -
0.0046296296,
```

As a MLSFEM simulation is running under MPI, the root process is set up to run the simulation of the assembly of superelements. Each of the other processes runs a simulation for a group of subelements, each with its own set of constraint equations. At each time step, the root process sends the nodal displacements at the current time step to the other processes, which use this information to take their own time steps. At the end of the simulation, each process writes an output file for the elements it contains. These output files may then be analysed separately or merged into a single file with information from all super- and subelements.

## 3 Results

### 3.1 Implementation test case

In order to test the code as it is being developed, a small model was created with one superelement and two subelements. The superelement is an eight-node cube, which gives a linear global displacement field. The subelements are twenty-node parallelepipeds (each half the height of the superelement) which give quadratic local fields within each subelement.

Figure 2 shows the (amplified) displacements in the subelements in response to a force applied at the corner node of the superelement. There are no boundary conditions imposed on the subelements directly. Instead, resulting displacement fields in the subelements are due only to the boundary conditions on the superelement degrees of freedom, connected to the subelement degrees of freedom through the constraint equations.

At each step, the root process passes its state vector of degrees of freedom ($Q$) to the processes simulating the subelements. Each of these processes then uses the received degrees of freedom in the constraint equation, 10.

It is clear from Figure 2 that the global fields are being transferred effectively to the local fields in an average sense. But it can also be seen that the increased number of degrees of freedom allow for small scale refinement of the field. This is most easily seen along the edges of the deformed cube in Figure 2. If the edges are compared to the undeformed wire frame edges, it is clear that the deformed shape is not quite linear. This local refinement of the displacement (from the linear global field) is exactly what the MLSFEM is intended to do.

### 3.2 Laminated composite plate

Building on the apparent success of the single-superelement test case, more complex model was simulated with the MLSFEM software. This example uses 2304 elements for the global field, assembled to form a plate. This plate is then subjected to impact loading.

Figures 3 and 4 show snapshots of the plate and of a group of superelements offset from the center of the plate, along one of the planes of symmetry. Figure 3 corresponds to the displacements at 0.00025 seconds after impact; Figure 4

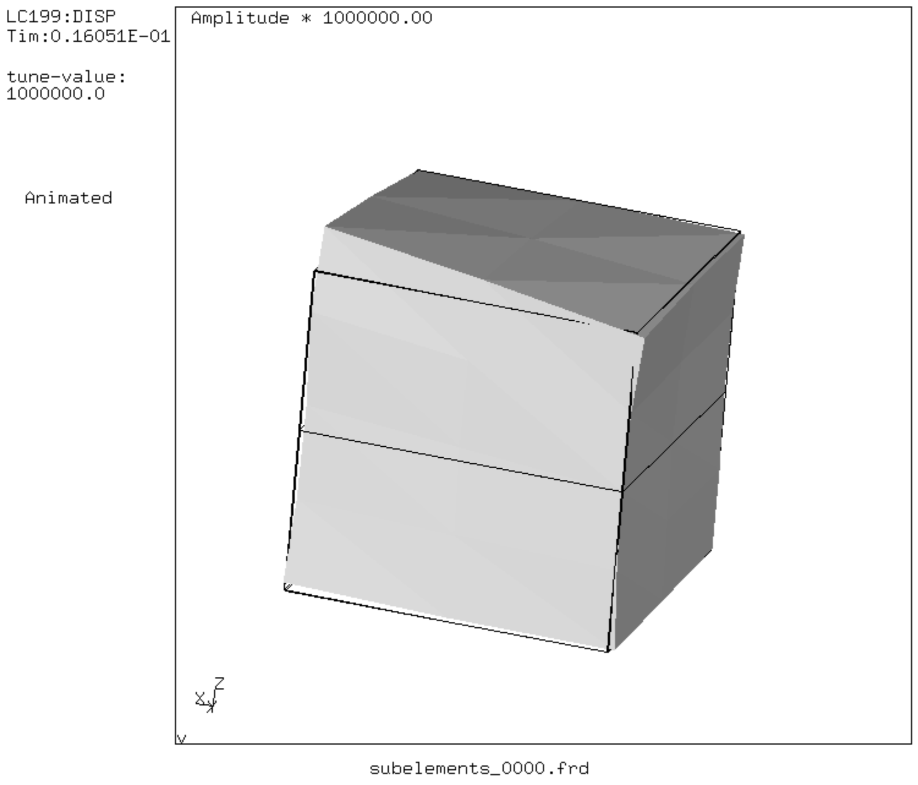
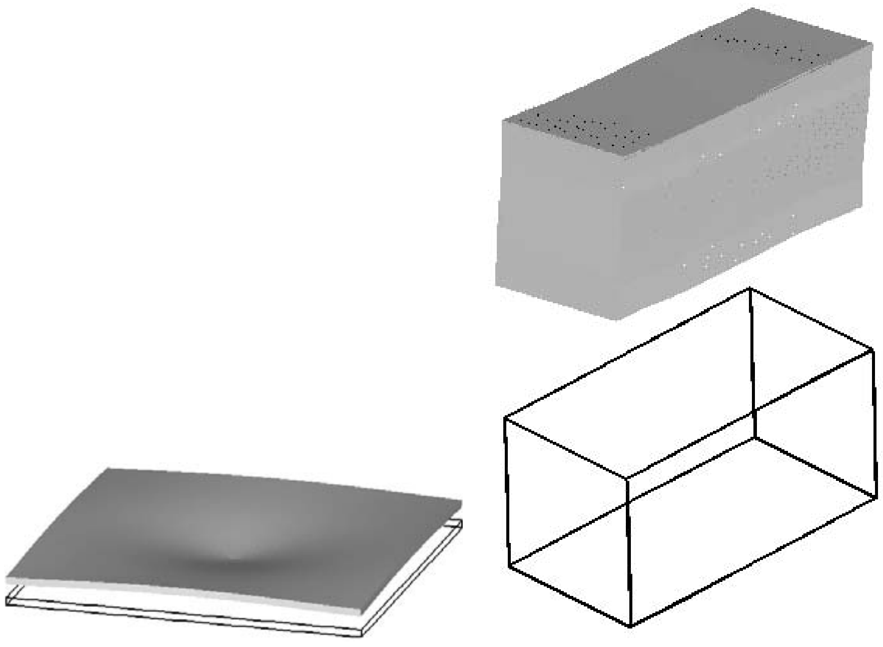
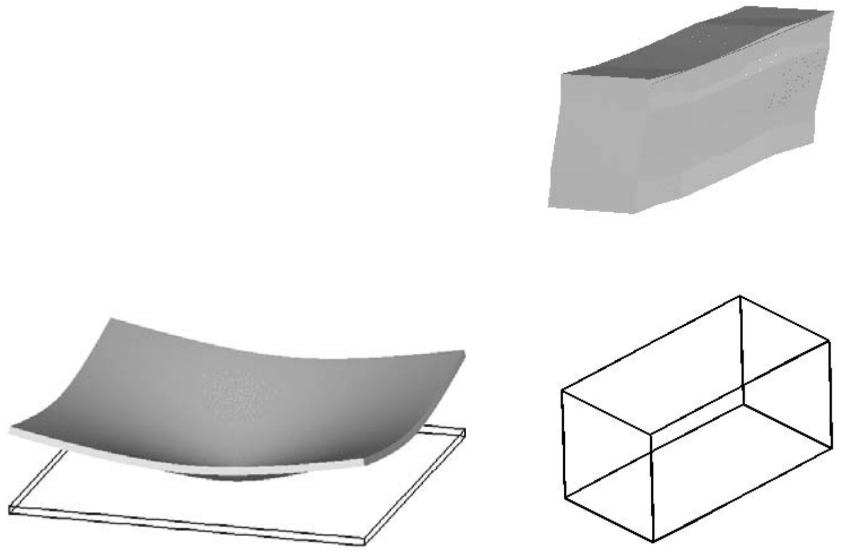


Figure 2: Snapshot of deformed subelements.



(a) (b)

Figure 3: The plate (a) and subelements (b), at time = 0.00025 seconds



(a) (b)

Figure 4: The plate (a) and subelements (b), at time = 0.00049 seconds

corresponds to the displacements at 0.00049 seconds after impact. As in the case of the test case presented in Section 3.1, no boundary conditions were applied to the subelements that make up the superelements. All displacements in the subelements are due to the constraint equations and the vector of superelement displacements that is passed to the process simulating the subelements at each time step.

## 4 Conclusions

A parallel implementation of the Multiple Length Scale Finite Element Method was performed using the MPI library. The implementation was tested using a simple test case with a single superelement and with a plate assembly. The subelements were shown to successfully represent the global fields in an average sense while also providing for local refinement of the displacement fields.