Real Time Hybrid Simulation on a Parallel Machine

Gregory Bunting, Payton Lindsay

Computational Solid and Structural Mechanics Lab
School of Civil Engineering, Purdue University

Overview

- Introduction to RTHS
- RTHS Example
- My requirements for Parallel Code
- Structure of RTHS Code
- Parallel Transformations
- Execution time and Speedups
- Conclusions

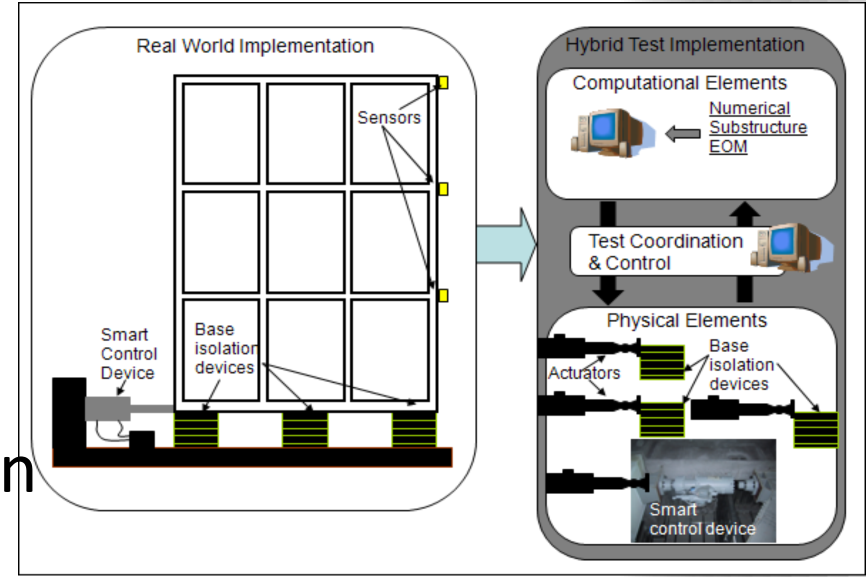
Real Time Hybrid Simulation

- Full scale tests can be expensive

- Hybrid Simulation:

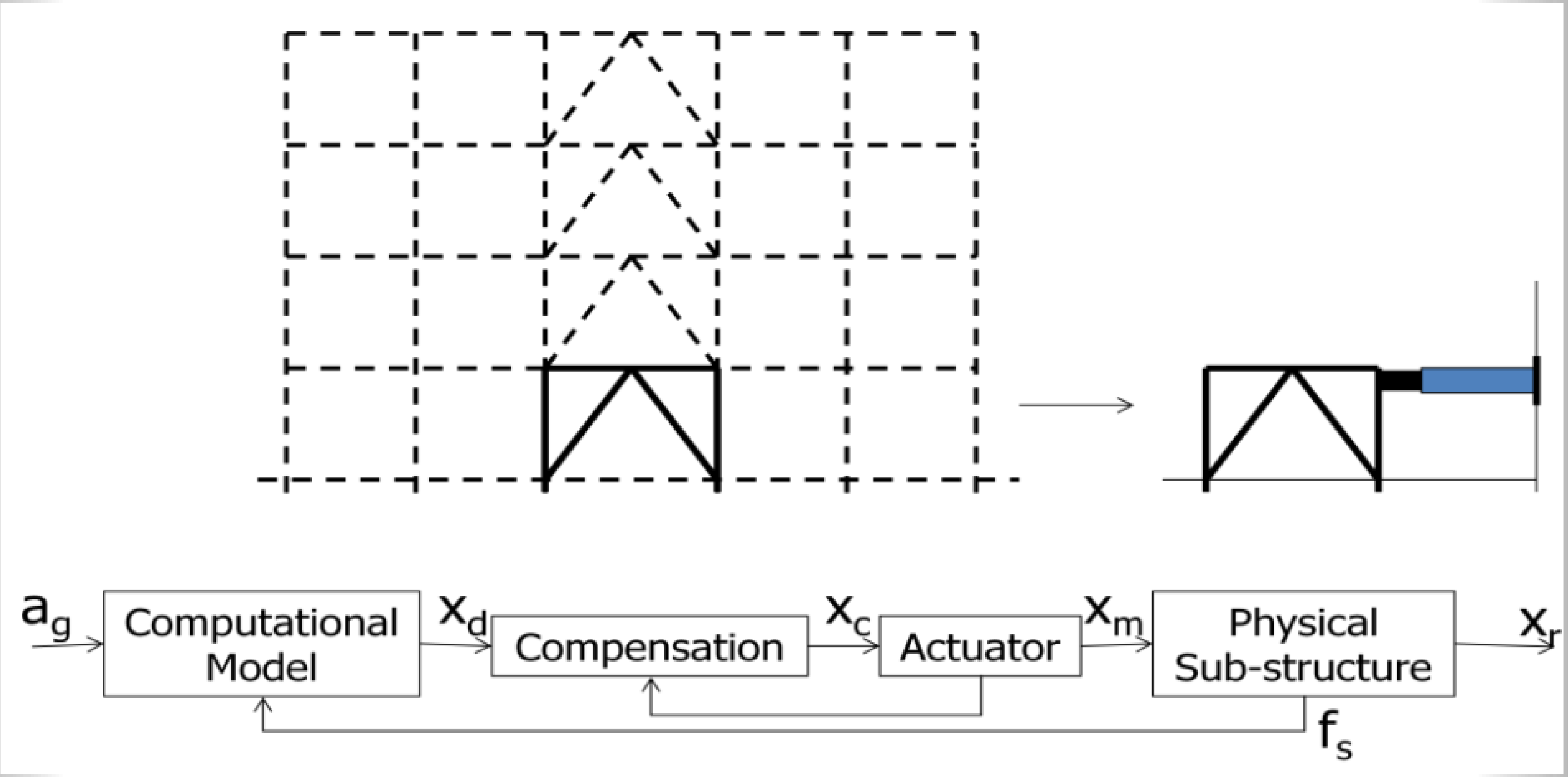
  Physical + Numerical

- Real Time Hybrid Simulation
  - Accurate dynamics
  - Typically performed at 1024 Hz
  - Constraints on numerical model size



Tidwell, Gao, Huang, Lu, Dyke, Gill, 2009

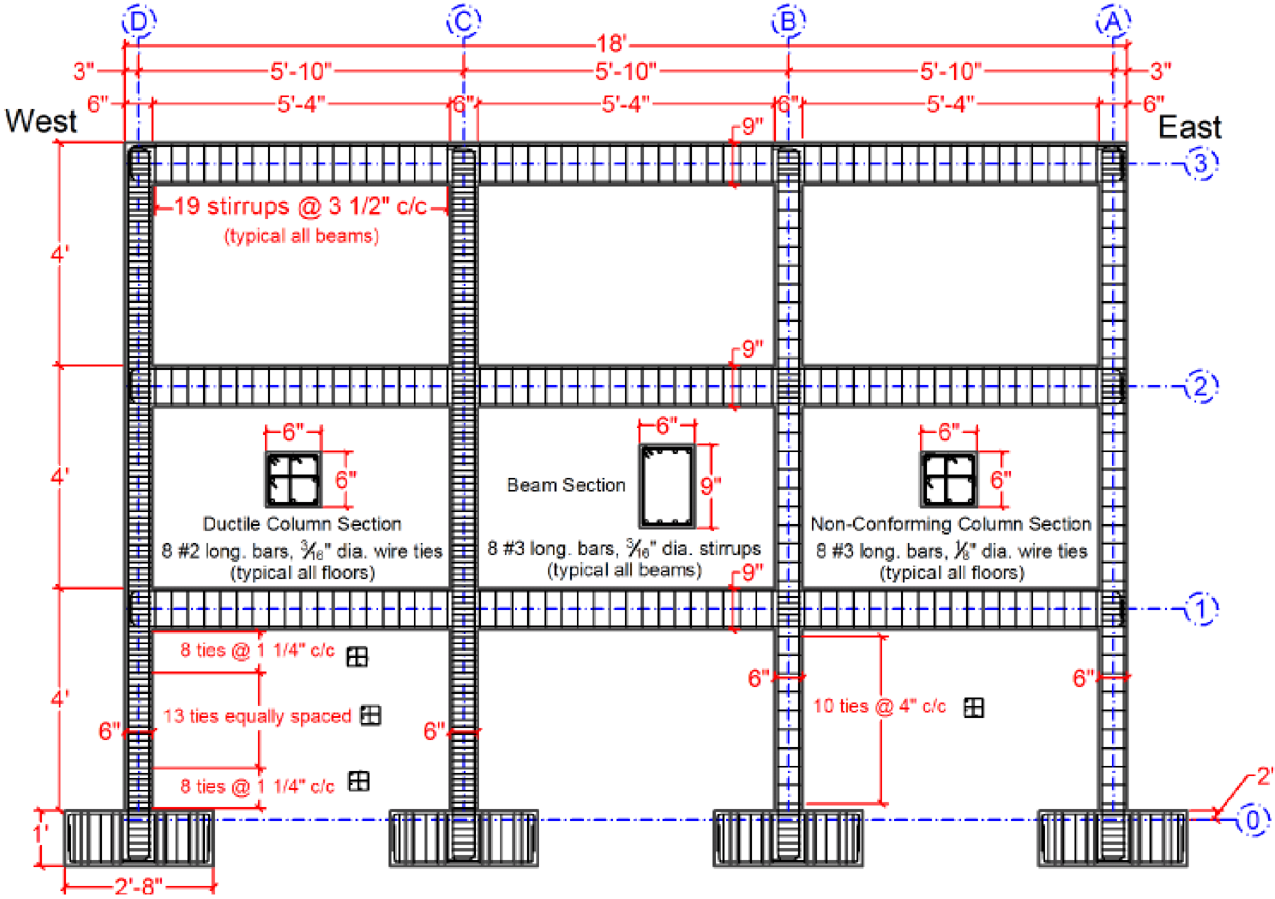
Real Time Hybrid Simulation



RTHS Example

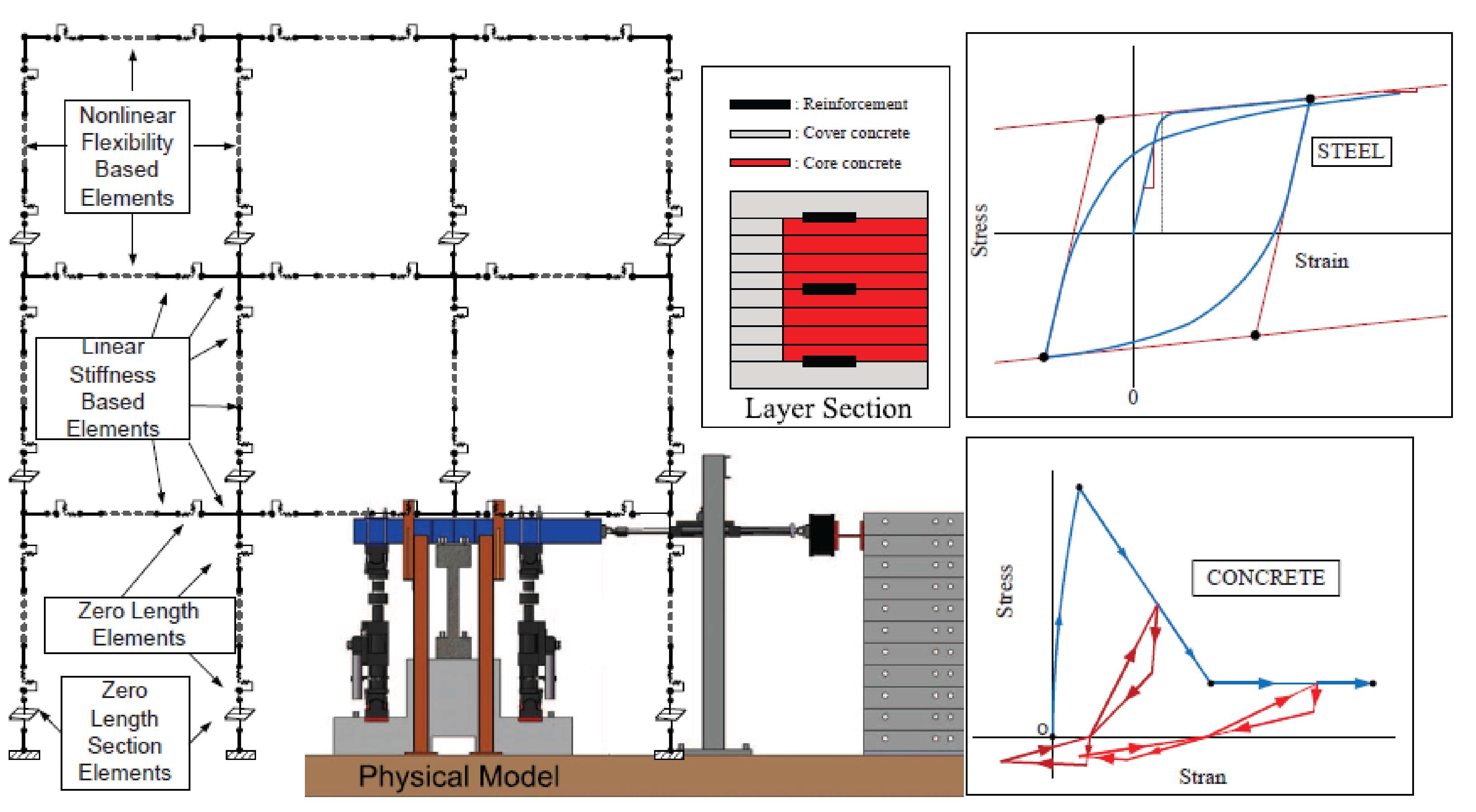
- **Ghannoum and Moehle (2012) UC Berkeley**

RTHS Example



- **Ghannoum and Moehle (2012) UC Berkeley**

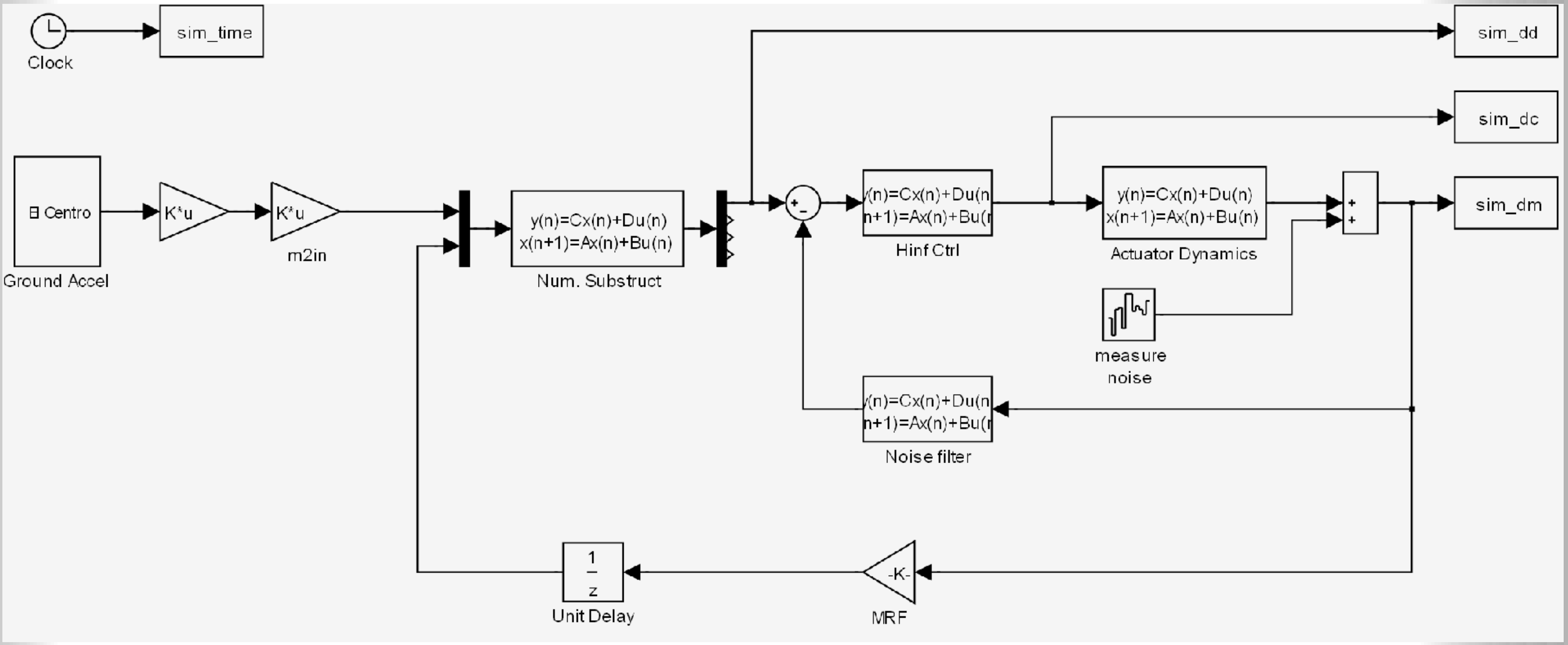
RTHS Example



- **Saoumo et al 2013 UC Boulder**

My Requirements for Parallel Code

- Need to meet 1024 Hz requirement
- Size of numerical model is unknown – the bigger numerical model we can handle, the more interesting tests we can run
- Will eventually run on real time machine
- Will run as part of a physical test
  - Shake Table
  - Actuators
  - Accelerometers


Clock
sim_time
sim_dd
sim_dc
sim_dm
El Centro
Ground Accel
K*u
K*u
m2in
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Num. Substruct
+
-
y(n)=Cx(n)+Du(n
n+1)=Ax(n)+Bu(n
Hinf Ctrl
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Actuator Dynamics
+
+
measure noise
y(n)=Cx(n)+Du(n
n+1)=Ax(n)+Bu(n
Noise filter
1
z
Unit Delay
-K-
MRF

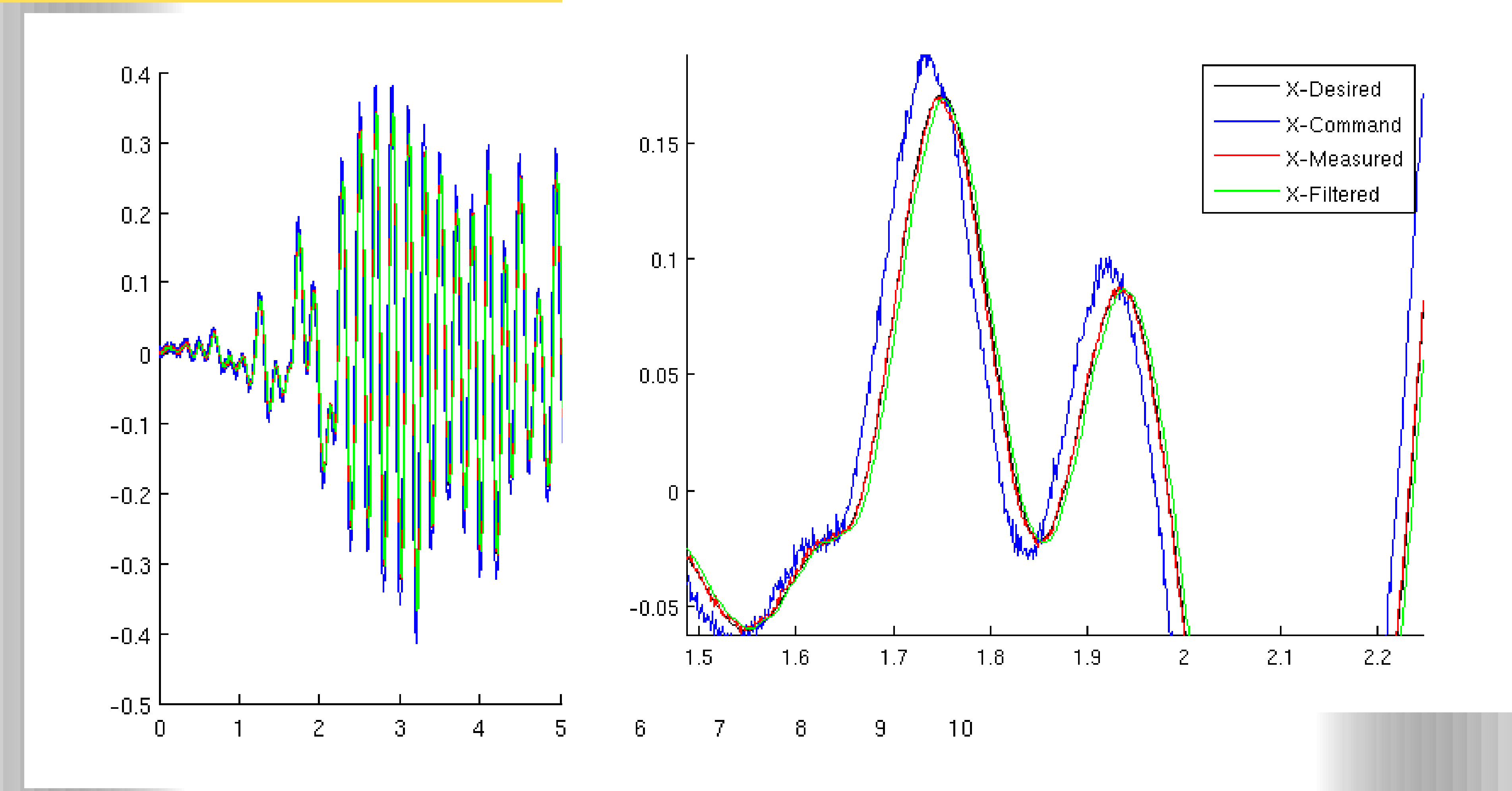
Structure of RTHS Code

- Earthquake input (El Centro, 60s)
  - Gains to adjust units
- Numerical Substructure
  - State Space or FEA Representation
- Compensation / Control
- Actuator Dynamics
- Randomly Generated Noise Values
- Noise Filter
- Physical Substructure (Pure Stiffness)

Structure of Code

- Preprocessing – building of state space matrices – generally ignored because this can be done before Real Time section starts
- Large Loop – Time stepping
  - Not parallelizable (Need previous results)
  - Must be solved in less than .976 milliseconds
  - Bulk of computation will be in NS
- Postprocessing – graphing results, etc. generally ignored because this can be done after Real Time Section

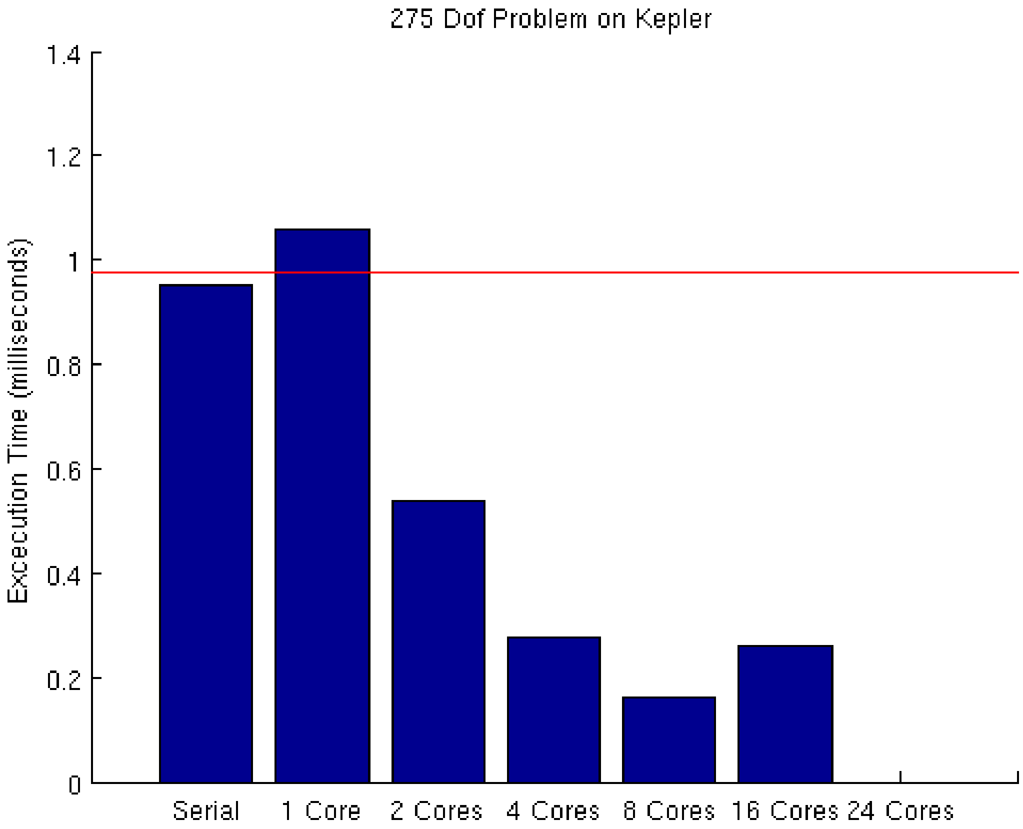
Structure of Code



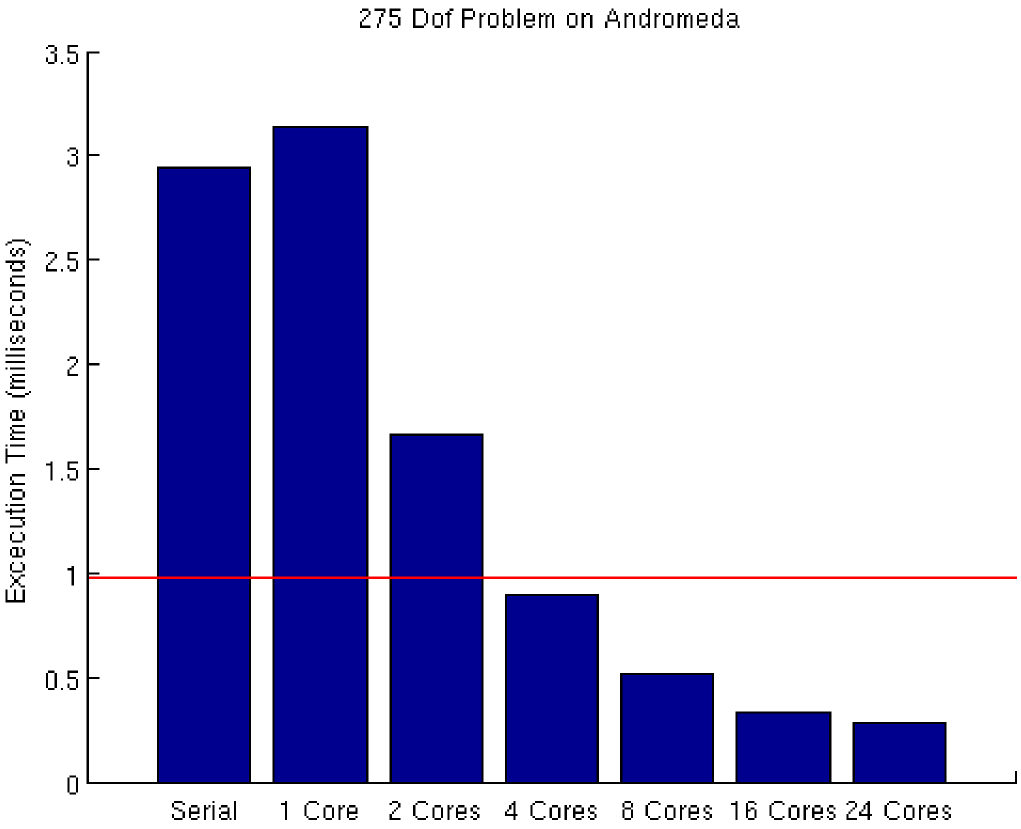
Parallel Transformations

- Parallel for loops and loop collapsing– just on NS solve
- Parallel for loops on other solves had high overhead
- Parallel sections – splitting NS from the rest of the time stepping code and adding a unit delay
  - High overhead for this problem
  - May use if we have more complicated control techniques or other calculations to do

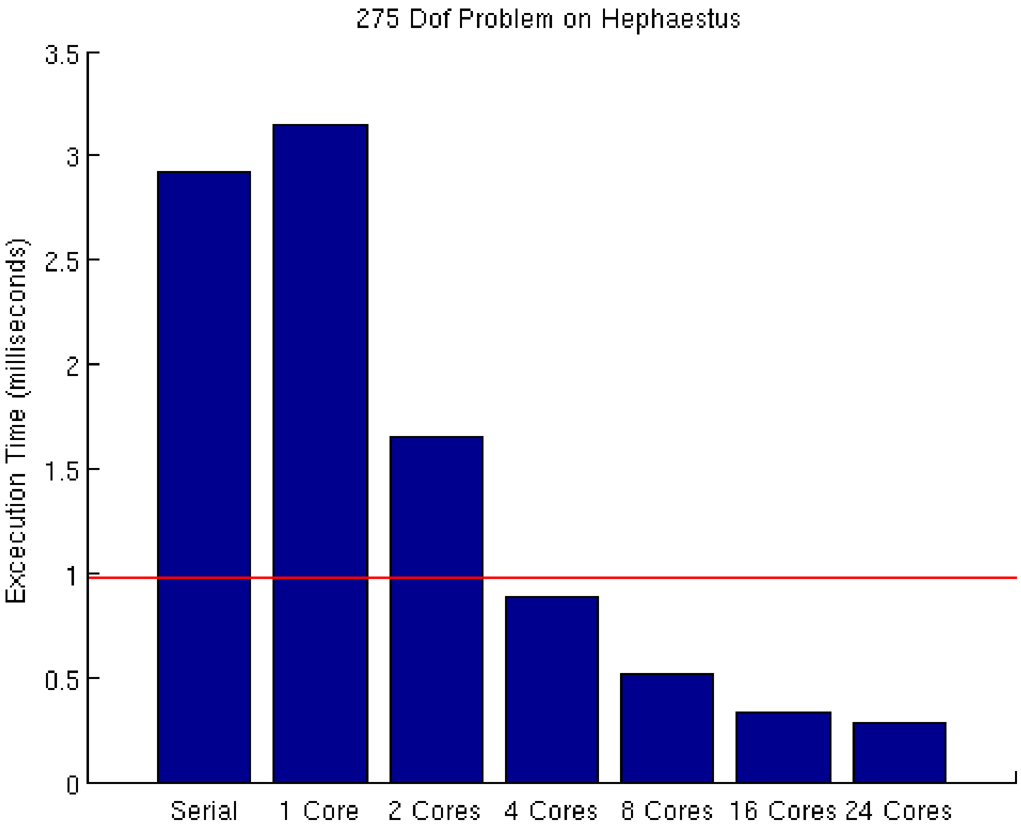
Timing Data


275 Dof Problem on Kepler
Execution Time (milliseconds)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Serial
1 Core
2 Cores
4 Cores
8 Cores
16 Cores
24 Cores

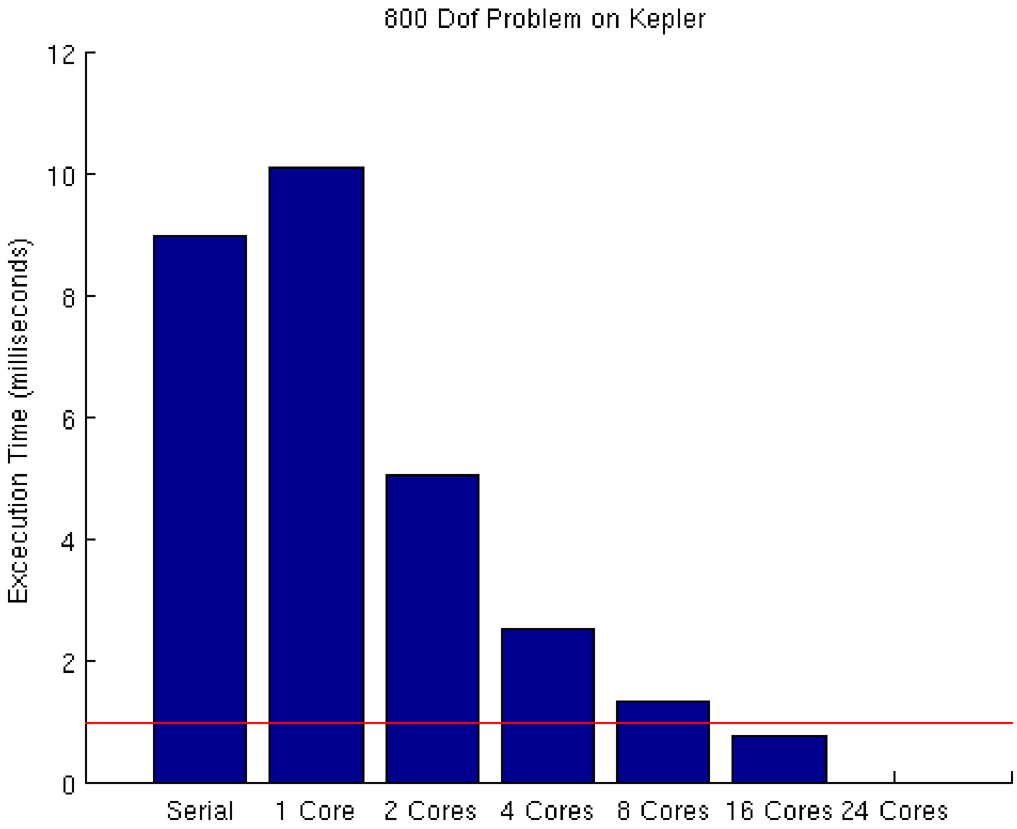
Timing Data


275 Dof Problem on Andromeda
Excecution Time (milliseconds)
3.5
3
2.5
2
1.5
1
0.5
0
Serial
1 Core
2 Cores
4 Cores
8 Cores
16 Cores
24 Cores

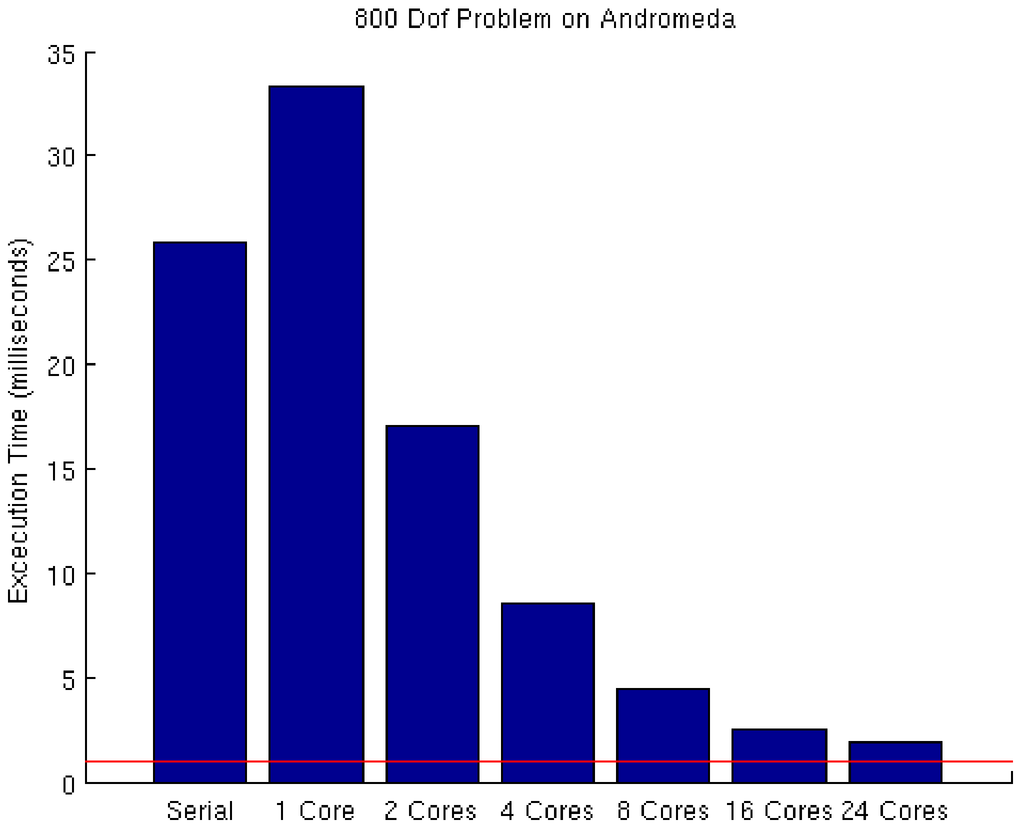
Timing Data


275 Dof Problem on Hephaestus
Execution Time (milliseconds)
3.5
3
2.5
2
1.5
1
0.5
0
Serial
1 Core
2 Cores
4 Cores
8 Cores
16 Cores
24 Cores

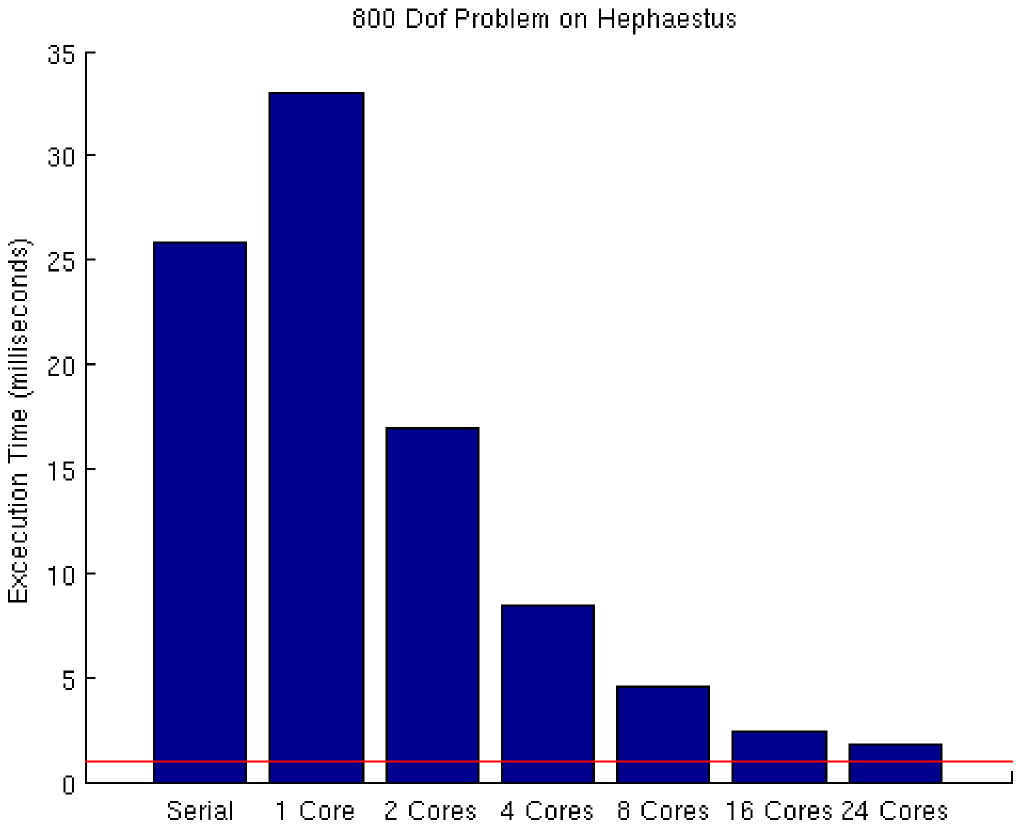
Timing Data


800 Dof Problem on Kepler
Execution Time (milliseconds)
0
2
4
6
8
10
12
Serial
1 Core
2 Cores
4 Cores
8 Cores
16 Cores
24 Cores

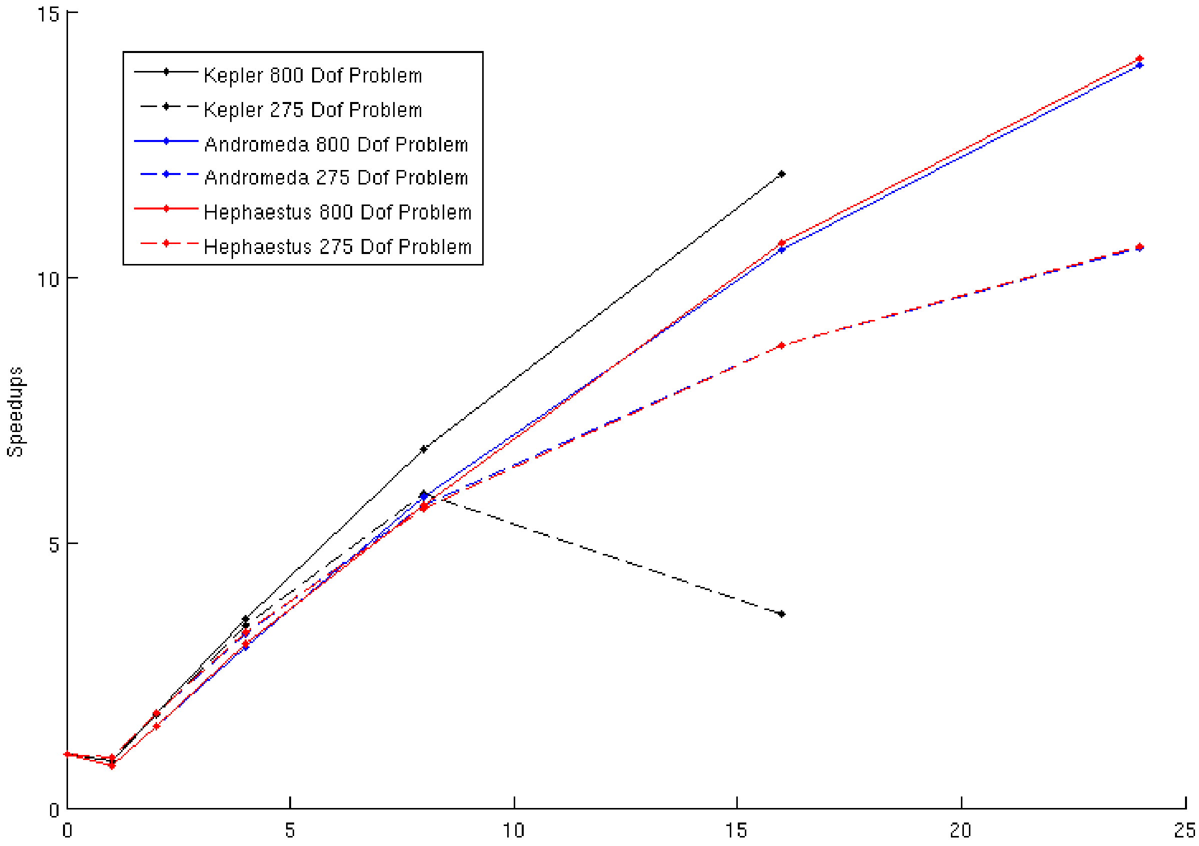
Timing Data


800 Dof Problem on Andromeda
Excecution Time (milliseconds)
0
5
10
15
20
25
30
35
Serial
1 Core
2 Cores
4 Cores
8 Cores
16 Cores
24 Cores

Timing Data


800 Dof Problem on Hephaestus
Execution Time (milliseconds)
35
30
25
20
15
10
5
0
Serial
1 Core
2 Cores
4 Cores
8 Cores
16 Cores
24 Cores

Speedups



15
10
5
0
Speedups
0
5
10
15
20
25
Kepler 800 Dof Problem
Kepler 275 Dof Problem
Andromeda 800 Dof Problem
Andromeda 275 Dof Problem
Hephaestus 800 Dof Problem
Hephaestus 275 Dof Problem

Conclusions

- Parallel Code Meets our requirements – increases the size of the NS that we are able to solve
- This code will be used to run a RTHS in the future
- Even though our speedups are not great, it works well for our application

Future Work

- Implement code in RT framework on a RT kernal
- Add ability to interact with sensors and control devices
- Possibly use parallel sections if we have another large computation part
- Run a RTHS with this setup