Challenges in Multi-physics Software Design: The Large Application Project Perspective

Andrew Siegel
Argonne National Laboratory

Contributions: Paul Fischer, Dinesh Kaushik, Tim Tautges (ANL)
Kord Smith (MIT)
Hank Childs (LBNL)
Fundamental question

- Gap/lag exists between new ideas coming out of applied math community and what one finds in typical production science applications.
  - Latter mostly implemented in prototype applications

- What are the practical reasons for this slow adoption and how might it better be overcome, particularly
  - from perspective of large, multiphysics apps project
  - given realities of software engineering ??
  - can help to be aware of broader perspectives and constraints beyond differential equations
Examples taken from SHARP

- **SHARP** – Simulation-based High Efficiency Advanced Reactor Prototyping.

- Since 2005
  - Joint ANL Nuclear Engineering/MCS + collaborators
  - Focus on advanced thermo-hydraulics/neutronics depletion modeling with coupling

- Multiple funding sources ~ 5-8M/year
  - DOE-NE AMSO Office
  - DOE-NE NERI, VHTR
  - DOE-SC ASCR (CESAR project)
Software Design Cycles (Notional)
Predictability is at the top of the heap for application codes

Useful Predictions

- frameworks
- Better software engineering
- better numerics
- better data

Lots of solutions based on heuristics but with long track record
- Reynolds stress closures
- non-linear acceleration techniques
- boiling models
- multi-group cross section collapse, homogenization

Inverse of Keyes’ “guilty until proven innocent”. High burden of proof to demonstrate that mathematically better is necessary
Results Presented by NEA/OECD Benchmark Organizers, CFD4NRS 9/15/10

- 29 entries, resolution \( n = 1 \) M to 70 M gridpoints
- ANL SHARP group submission w/ Nek5000 ranked 1\(^{st}\) in thermocouple prediction, 6\(^{th}\) in velocity prediction

### Table 9. Submissions Ranked by Comparison to Thermocouple Data

<table>
<thead>
<tr>
<th>Submission</th>
<th>TC Score</th>
<th>Code</th>
<th>Turbulence</th>
<th>Volumes</th>
</tr>
</thead>
<tbody>
<tr>
<td>S21</td>
<td>36</td>
<td><strong>Nek5000</strong></td>
<td>LES, spectral damping</td>
<td>21M</td>
</tr>
<tr>
<td>S16</td>
<td>45</td>
<td>Fluent 12</td>
<td>LES-WALE</td>
<td>7.7M</td>
</tr>
<tr>
<td>S8</td>
<td>48</td>
<td>STAR-CCM=3.06.005</td>
<td>LES-Wale SGS</td>
<td>13.2 M</td>
</tr>
<tr>
<td>S4</td>
<td>57</td>
<td>Fluent 6.3.26</td>
<td>LES, Dyn. Smagorinsky</td>
<td>13.1M</td>
</tr>
<tr>
<td>S22</td>
<td>72</td>
<td>Star-CCM+</td>
<td>LES</td>
<td>4.4 M</td>
</tr>
<tr>
<td>S23</td>
<td>78</td>
<td>CFX</td>
<td>LES-WALE</td>
<td>1.9 M</td>
</tr>
<tr>
<td>S5</td>
<td>81</td>
<td>Satmce</td>
<td>LES, Dyn. Smagorinsky</td>
<td>6.2M</td>
</tr>
<tr>
<td>S3</td>
<td>82</td>
<td>FLUENT 12.1</td>
<td>LES, Dyn. Smagorinsky</td>
<td>34M</td>
</tr>
<tr>
<td>S20</td>
<td>82</td>
<td>FLUENT</td>
<td>LES, Dyn. Smagorinsky</td>
<td>70.3 M</td>
</tr>
<tr>
<td>S19</td>
<td>83</td>
<td>FLUENT 12.1</td>
<td>SST-kos</td>
<td>11M</td>
</tr>
<tr>
<td>S14</td>
<td>88</td>
<td>CABARET</td>
<td>LES</td>
<td>1.8 M</td>
</tr>
<tr>
<td>S25</td>
<td>88</td>
<td>TransAT</td>
<td>LES-WALE</td>
<td>2.5 M</td>
</tr>
<tr>
<td>S18</td>
<td>93</td>
<td>CFX</td>
<td>DES-SST</td>
<td>2.4 M</td>
</tr>
<tr>
<td>S10</td>
<td>105</td>
<td>Fluent 12</td>
<td>LES-Smagorinsky-Lilly</td>
<td>0.92M</td>
</tr>
<tr>
<td>S26</td>
<td>105</td>
<td>FLUENT v12.1</td>
<td>LES - Dynamic KE SGS</td>
<td>7.2 M</td>
</tr>
<tr>
<td>S8</td>
<td>110</td>
<td>STAR-CCM=3.06.005</td>
<td>LES-Wale</td>
<td>9.3 M</td>
</tr>
<tr>
<td>S17</td>
<td>121</td>
<td>OpenFOAM 1.6</td>
<td>LES, legn. Dyn. eddy</td>
<td>0.28 M</td>
</tr>
<tr>
<td>S7</td>
<td>124</td>
<td>CFX</td>
<td>SAS-SST</td>
<td>5.0 M</td>
</tr>
<tr>
<td>S11</td>
<td>138</td>
<td>CFX</td>
<td>LES-Wale</td>
<td>3.4 M</td>
</tr>
<tr>
<td>S1</td>
<td>139</td>
<td>Fluent</td>
<td>LES</td>
<td>4.5 M</td>
</tr>
<tr>
<td>S34</td>
<td>151</td>
<td>OpenFOAM 1.6</td>
<td>LES, Dyn. Smagorinsky</td>
<td>8 M</td>
</tr>
<tr>
<td>S9</td>
<td>164</td>
<td>AdvanceFLOW/rev v4</td>
<td>LES-Smagorinsky</td>
<td>4.1 M</td>
</tr>
<tr>
<td>S15</td>
<td>166</td>
<td>CFX</td>
<td>SAS-SST</td>
<td>2.3 M</td>
</tr>
<tr>
<td>S13</td>
<td>186</td>
<td>CFX 12.0</td>
<td>SAS-SST</td>
<td>1.1 M</td>
</tr>
<tr>
<td>S37</td>
<td>188</td>
<td>Star-CCM+</td>
<td>Vf</td>
<td>0.02 M</td>
</tr>
<tr>
<td>S20</td>
<td>197</td>
<td>CFX</td>
<td>SAS-SST</td>
<td>1.0 M</td>
</tr>
<tr>
<td>S3</td>
<td>201</td>
<td>CFX 12.0</td>
<td>LES, WALE</td>
<td>0.97 M</td>
</tr>
<tr>
<td>S12</td>
<td>224</td>
<td>MODTURC_CLAS-I2ST</td>
<td>k-epsilon/RNG</td>
<td>0.89M</td>
</tr>
<tr>
<td>S28</td>
<td>232</td>
<td>STAR-CCM+</td>
<td>Dynamic Vreman</td>
<td>3.7 M</td>
</tr>
</tbody>
</table>

### Table 8. Submissions Ranked by Comparisons to PIV Data

<table>
<thead>
<tr>
<th>Submission</th>
<th>Velocity Score</th>
<th>Code</th>
<th>Turbulence</th>
<th>Volumes</th>
</tr>
</thead>
<tbody>
<tr>
<td>S20</td>
<td>75</td>
<td>FLUENT</td>
<td>LES, Dyn. Smagorinsky</td>
<td>70.3 M</td>
</tr>
<tr>
<td>S2</td>
<td>106</td>
<td>Fluent 12</td>
<td>LES, Dyn. Smagorinsky</td>
<td>34 M</td>
</tr>
<tr>
<td>S8</td>
<td>178</td>
<td>STAR-CCM=3.06.006</td>
<td>LES-Wale SGS</td>
<td>13.2 M</td>
</tr>
<tr>
<td>S4</td>
<td>184</td>
<td>Fluent 6.3.26</td>
<td>LES, Dyn. Smagorinsky</td>
<td>5.8 M</td>
</tr>
<tr>
<td>S24</td>
<td>212</td>
<td>OpenFOAM 1.6</td>
<td>LES, Dyn. Smagorinsky</td>
<td>8 M</td>
</tr>
<tr>
<td>S21</td>
<td>247</td>
<td>Nek5000</td>
<td>LES, spectral damping</td>
<td>21M</td>
</tr>
<tr>
<td>S16</td>
<td>270</td>
<td>Fluent 12</td>
<td>LES-WALE</td>
<td>7.7 M</td>
</tr>
<tr>
<td>S3</td>
<td>311</td>
<td>CFX3 v12</td>
<td>LES</td>
<td>0.97 M</td>
</tr>
<tr>
<td>S11</td>
<td>312</td>
<td>CFX</td>
<td>LES-Wale</td>
<td>3.4 M</td>
</tr>
<tr>
<td>S19</td>
<td>322</td>
<td>FLUENT 12.1</td>
<td>SST-kos</td>
<td>11 M</td>
</tr>
<tr>
<td>S18</td>
<td>349</td>
<td>CFX</td>
<td>LES</td>
<td>1.8 M</td>
</tr>
<tr>
<td>S14</td>
<td>358</td>
<td>CABARET</td>
<td>LES-Wale</td>
<td>3.4 M</td>
</tr>
<tr>
<td>S17</td>
<td>374</td>
<td>OpenFOAM 1.6</td>
<td>LES, legn. Dyn. eddy</td>
<td>0.28 M</td>
</tr>
<tr>
<td>S9</td>
<td>375</td>
<td>AdvanceFLOW/rev v4</td>
<td>LES-Smagorinsky</td>
<td>4.1 M</td>
</tr>
<tr>
<td>S22</td>
<td>404</td>
<td>STAR-CCM=3.06.006</td>
<td>LES-Wale</td>
<td>9.3 M</td>
</tr>
<tr>
<td>S5</td>
<td>408</td>
<td>FLUENT v12.1</td>
<td>LES - Dynamic KE SGS</td>
<td>7.2 M</td>
</tr>
<tr>
<td>S26</td>
<td>432</td>
<td>STAR-CCM</td>
<td>LES-Wale</td>
<td>9.3 M</td>
</tr>
<tr>
<td>S27</td>
<td>446</td>
<td>CFX</td>
<td>LES</td>
<td>4.5 M</td>
</tr>
<tr>
<td>S23</td>
<td>458</td>
<td>Fluent 12</td>
<td>LES-Wale</td>
<td>9.3 M</td>
</tr>
<tr>
<td>S10</td>
<td>477</td>
<td>Fluent 12</td>
<td>LES</td>
<td>4.5 M</td>
</tr>
<tr>
<td>S5</td>
<td>531</td>
<td>TransAT</td>
<td>LES</td>
<td>4.5 M</td>
</tr>
<tr>
<td>S25</td>
<td>585</td>
<td>Fluent</td>
<td>LES</td>
<td>4.5 M</td>
</tr>
<tr>
<td>S1</td>
<td>589</td>
<td>CFX</td>
<td>SAS-SST</td>
<td>5.0 M</td>
</tr>
<tr>
<td>S7</td>
<td>604</td>
<td>CFX</td>
<td>SAS-SST</td>
<td>2.3 M</td>
</tr>
<tr>
<td>S13</td>
<td>706</td>
<td>CFX 12.0</td>
<td>SAS-SST</td>
<td>1.1 M</td>
</tr>
<tr>
<td>S12</td>
<td>712</td>
<td>MODTURC_CLAS-I2ST</td>
<td>k-epsilon/RNG</td>
<td>0.89 M</td>
</tr>
<tr>
<td>S29</td>
<td>719</td>
<td>CFX</td>
<td>SAS-SST</td>
<td>1.0 M</td>
</tr>
</tbody>
</table>
Predicting important thermal fluid phenomena in the design of a sodium cooled fast reactor (SFR)

Can we predict temperature gradients to ensure integrity of structures?

Can we pressure drop across steam generator for variety of designs?

Can we predict peak local temperatures to ensure safety (no local boiling, melting, etc.)?

What is basic theory that enables these predictions?
Nuscale Small Modular Reactor

Pressure drop across helical Steam generators?

Peak temperatures reached during incipient free convection at startup?
Breed-burn concepts

Terrapower’s Traveling Wave Reactor

- New regime for neutronics codes – power profiles change with time
- Highly detailed neutronics necessary with careful treatment of uncertainties in breeding zone

A. Coolant Pumps
B. Expansion area for fission gases
C. Fuel – depleted uranium inside hexagonal pillars
D. Fission wave (red)
E. Breeding wave (yellow)
F. Liquid sodium coolant
“Typical” multiphysics system and basis for many examples

LWR: Entire Plant Modeled for Safety Analysis
Model Many-meter to Sub-millimeter Scales
Challenges:
Pellet/Clad Modeling and Simulation

- Cladding oxidation
- Corrosion deposition
- Pellet cracking/relocation
- Fission gas release
- Stress corrosion cracking
- Solid-solid conductance
- Pellet clad interaction
- Imperfection modeling
- Energetic fracturing
General form of continuous model equations

\[
\frac{\partial u}{\partial t} + u \cdot \nabla u = - \frac{1}{\rho} \Delta P + g \frac{T'}{T_0} \bar{k} + \nu \nabla^2 u
\]
\[
\nabla \cdot u = 0
\]

\[
\frac{\partial T}{\partial t} + u \cdot \nabla T = Pe^{-1} \nabla^2 T + \dot{S}(\Psi)
\]

\[
\left[ \hat{\Omega} \cdot \Delta + \sigma(r,E) \right] \Psi(r,\hat{\Omega},E) = \int dE' \int d\hat{\Omega}' \sigma^S(r,E' \rightarrow E,\hat{\Omega}' \cdot \hat{\Omega}) \Psi(r,\hat{\Omega}',E')
\]
\[
+ \frac{X(E)}{4\pi K_{eff}} \int dE' \nu \sigma^f(r,E') \int d\hat{\Omega}' \Psi(r,\hat{\Omega}',E')
\]

• Cross sections depend on temperature and density of fluid and fuel isotopics
• Fuel isotopics evolve slowly according to Batemen equations – no real coupling
• Structures can deform on slow time scales; FIV common also
• Detailed material composition of nuclear fuel, gap conductance, etc. all potential issues

\[\Psi: \text{ neutron density }\]
\[T: \text{ Temperature}\]
\[u: \text{ fluid velocity}\]

Becomes a study in reductionism as much as building a coupling matrix!
Figure 2.1: The \((n,\gamma)\) cross section of \(^{238}\text{U}\) and the fission cross sections of \(^{235}\text{U}\), \(^{238}\text{U}\), and \(^{239}\text{Pu}\) (ENDF/B-VI, 300 K data).
Evolution of Nodal LWR Computational Tools

• 1980 - developed as cutting-edge technology
  • Too slow for any production analysis on world’s best mainframe computers

• 1984 - first applications on PC-AT (6 MHz)
  • 12 hour execution per statepoint

• 1986 - widespread applications on high-end mainframes
  • Overnight execution of single-cycle depletions

• 1990 - cycle core design/steady-state safety analysis
  • Many day executions

• 1995 - first transient safety analysis
  • Many hour executions per transient

• 1998 - first core loading optimizations
  • Many day executions per cycle

• 2010 - full spectrum safety analysis on desktop PCs
  • Many full-core coupled-physics state points per second

• 30 years from initial demonstration to “complete” deployment
Reduction of continuous equations happens prior to mathematical formulation

• Confusion between code and problem
  – Most physics codes solve more than one problem
  – Nature of couplings and associated methods differ significantly from problem to problem for a given code.
  – Real world projects must accrete capability as they respond to new challenge problems!

• Can in principle formulate a “general model” based on all fundamental physics equations
  – Neutron transport (linear Boltzmann)
  – Conjugate heat transfer (CFD + non-linear diffusion)
  – Isotopics (Batemen equations)
  – Structural mechanics (including complex nuclear fuel)

• However, beyond a certain degree this is highly impractical. Relevant phenomena can range from tracking fission gas release to fuel reloading.

• Code usually assumes range of target scales that are within its capability and starts thinking of variables as homogenized parameters.
  – However these boundaries can change as new opportunities arise.
For a given simulation, usually 1 or 2 physical processes dominate

• Structural Mechanics/CFD:
  – FIV: strongly coupled
  – Rod bowing: weakly coupled
  – Assume everything else is constant

• Neutronics/CFD
  – Rod rupture: strongly coupled
  – Evolution to steady stage: weakly coupled
  – Assume everything else constant

• Neutronics/isotopics
  – Traveling wave: strongly coupled
  – Normal operation: weakly coupled (succession of steady states)
• When developing a community code to address these coupled problems (and beyond)

What are the key software challenges?
Software Issues not significant until dealing with production codes

• In software design performance and abstraction typically compete.

• None of the solvers discussed in this workshop are difficult to implement for prototype problems.

• In fact, it wouldn’t be too hard to develop high level languages that could automatically discretize and solve a broad class of coupled PDEs.

• Not particularly difficult in the larger scheme of things to retrofit a legacy component e.g. into a JNFLK framework (return a residual vector).

• SE challenges become most relevant for production codes
  – Production typically means living on the cutting edge in terms of HPC resources
  – Managing complexity for full applications also comes to forefront
  – Performance is justified partly by the science, and partly by the investments in large machines whose advantages must be maximally exploited
  – Funding cycles also dictate a very modest lead time from project inception to significant science/engineering, particularly in applied fields
Componentization of physics addresses legacy code reuse and complexity management

- Need to also understand broader motivations for preferring to split physics into standalone codes

- Code could certainly in principle be written as a single coupled system with a e.g. a Newton iteration loop.

- We have identified leveraging single physics “legacy codes” as a major motivator

- Not clear we wouldn’t want to componentize even if writing from scratch – managing complexity key, domain experts can experiment more easily with state of the art methods, etc.

- Need to retain performance of single physics -- In reactor analysis e.g. 80% + of our coupled calculations probably don’t require any coupling at all.
Coupling methods are very important but must keep in perspective vis-a-vis broader issues

- Also, when considering application goals of a code project, accuracy in coupling is only one in a very long list of potential sources of error.
  - Subgrid modeling for turbulent flows, resolution, parameter homogenization, uncertainties e.g. in nuclear cross section data, geometric uncertainties, mesh quality, single physics instabilities, sampling errors, bifurcations, assessment of predictive capability, ease of incorporation of new modules, etc.
  - Vast number of issues between solution to discretized PDE and answer of relevance

- Given finite resources, tendency is to attack biggest sources of error first and favor existing approaches to others with low cost of implementation – both time and performance wise – until demonstration that something more sophisticated and with better properties is needed.

- Currently different forms of operator splitting with fixed point iteration have a lower threshold to adoption.

- This does not necessarily have to be the case ...

- Incentive system is well understood – framework adoption will increase when it accelerates progress according to existing reward system (can’t wait for latter to change)
Self-contained applications will have increasing difficulty competing in coming decade

- At the same time, any production multi-physics code requires non-trivial infrastructure that ideally each application would not have to replicate.

- This is compounded when one considers the lifecycle of an application, from meshing to i/o to solvers to coupling to visualization.

- Incompatible solutions for each beg for some universal standards/tools to ease the burden on the application programmer.

- Need a lower SE threshold (risk and effort) to get from existing capability to relevant prototype solution.
Application frameworks

• *Framework*: For my purposes any software that does a piece of the end to end application (though probably something more entrenched than a pre- or post processing tool).

• We became wise to the Framework vs. framework distinction in the past ten years after many failed projects who shared the “plug your physics in here, we’ll do the rest” approach.

• Significant progress in past ten years on merging interests of software engineering, applied math, CS with application needs

• Visualization, meshing, and solver tools have been the most broadly adopted for production codes.

• Coupling, workflow, embedded UQ, or any tool that too tightly constrains programming environment – provides too many rules and too much functionality → too much dependency.
Characteristics of existing multiphysics frameworks vary greatly

• In general, different multiphysics frameworks provide services to help with different aspects of the problem and endeavor to be physics neutral (for PDEs at least)

• This significantly adds to the confusion of evaluating huge zoo of apparent alternatives.

• These fall into several classes:
  – Mesh interpolation
    • General
    • Vendor specific
  – Orchestration
  – Mesh representation (e.g. without interpolation)
  – Large implicit solves (e.g. JNFK)
  – Other services – solvers, i/o, unit testing, matrix assembly
General Coupling Model

The fully coupled set of physics applications can now be expressed with the following set of equations

\[ \hat{f}(\hat{x}, \hat{x}, \hat{p}, t) = 0, \]  

(2.8)

where

\[ \hat{x} = [\dot{x}_0, \ldots, \dot{x}_i, \ldots, \dot{x}_{N_f-1}] , \]
\[ \hat{x} = [x_0, \ldots, x_i, \ldots, x_{N_f-1}] , \]
\[ \hat{p} = [p_{0,0}, \ldots, p_{0,N_{p0}-1}, \ldots, p_{i,0}, \ldots, p_{i,N_{pi}-1}, \ldots, p_{N_f-1,0}, \ldots, p_{N_f-1,N_{pN_f-1}}] , \]
\[ \hat{f} = \begin{bmatrix} f_0(\dot{x}_0, x_0, \{r_{0,k}\{\{x_m\}, \{p_{m,n}\}\}}, \{p_{0,l}\}, t)) \\
\vdots \\
\vdots \\
f_i(\dot{x}_i, x_i, \{r_{i,k}\{\{x_m\}, \{p_{m,n}\}\}}, \{p_{i,l}\}, t)) \\
\vdots \\
f_{N_f-1}(\dot{x}_{N_f-1}, x_{N_f-1}, \{r_{N_f-1,k}\{\{x_m\}, \{p_{m,n}\}\}}, \{p_{N_f-1,l}\}, t)) \end{bmatrix} . \]  

(2.9)

What are issues in moving from theoretical description to software?

Significant functionality
- Difficult to distinguish based on PowerPoint
- Often takes a few months to find out limitations

The PowerPoint view of software realizations of these systems
random examples
Key issues not addressed in the high level view

• What interface must a component expose?

• What concept does the framework have of “data on a mesh”?

• What is the data structure representing this protocol?

• Does framework run on
  – Disjoint set of processors
  – Overlap physics processor partition

• What potential portability issues arise?

• What is the communication mechanism?

• Complete list of requirements to be a compliant module
Are coupling frameworks, etc really necessary?

• Coupling is not so difficult – certainly a reasonable choice
  – More control over performance, fewer dependencies

• Larger engineering issues in efficiency of data marshaling/unmarshaling
  – Mesh tool -> Disk -> app -> coupler -> viz tool
  – TSTT e.g. addresses this issue
  – Visit and CUBIT plug-ins
MpCCI

- Very high level semantics
- Built-in physics codes
- Standardized physics names
- Standardized variable names
- Not scalable

Write driver to specify orchestration of coupling

MpcCI's modular...
Other practical critical issues are multiplied with multiphysics

• Three most important topics beyond what has been discussed already
  
  – Versioning/Provenance: Adopting code into multiphysics framework that is managed/developed by another group

  – Testing

  – Early portability to new leadership class machines
General concluding observations on frameworks

- Accrete capability through continuous work with diverse application groups – solve real problems early.

- Supply tools that can be used piecemeal and incrementally integrated into an application – collection of “lightweight components”.

- Give considerable attention to practical considerations like ease of build, portability – minimize required dependencies

- Find the intellectual merit in some level of support/collaboration

- Understand the memory constraints of typically HPC applications and put considerable work into efficient parallel layouts

- Don’t provide too much

- Provide Fortran interfaces

- Document copiously with many examples

- Be aware of and honest about existing limitations