The Challenges of Large-Scale Simulation

Science at Extreme Scales: Where Big Data Meets Large-Scale Computing
Tutorials

CASC
Center for Applied Scientific Computing

13 Sep 2018

Jeffrey A. F. Hittinger
Director

LLNL-PRES-758287

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

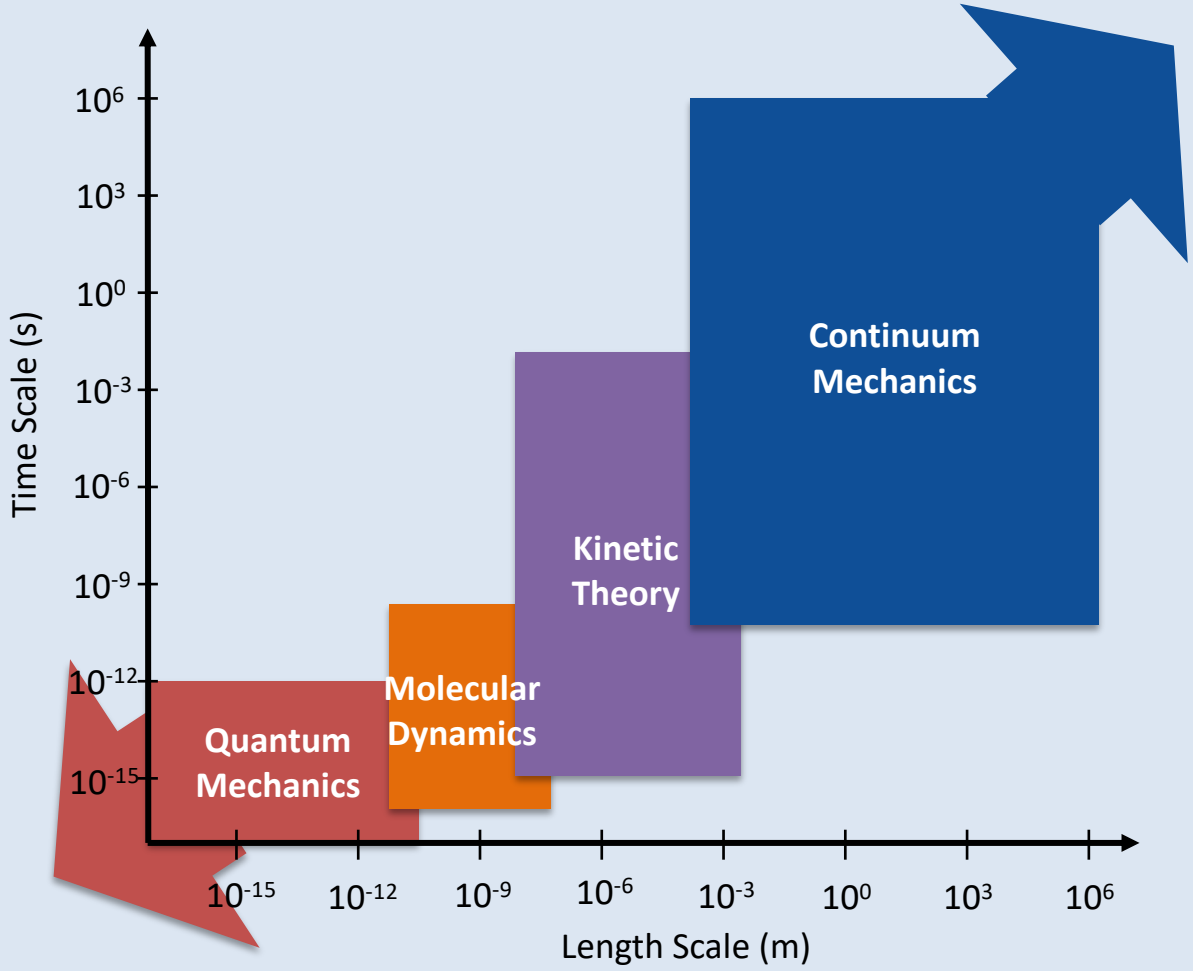
Thanks to

- Lois Curfman McInnes, ANL
- Lori Diachin, LLNL
- And others…

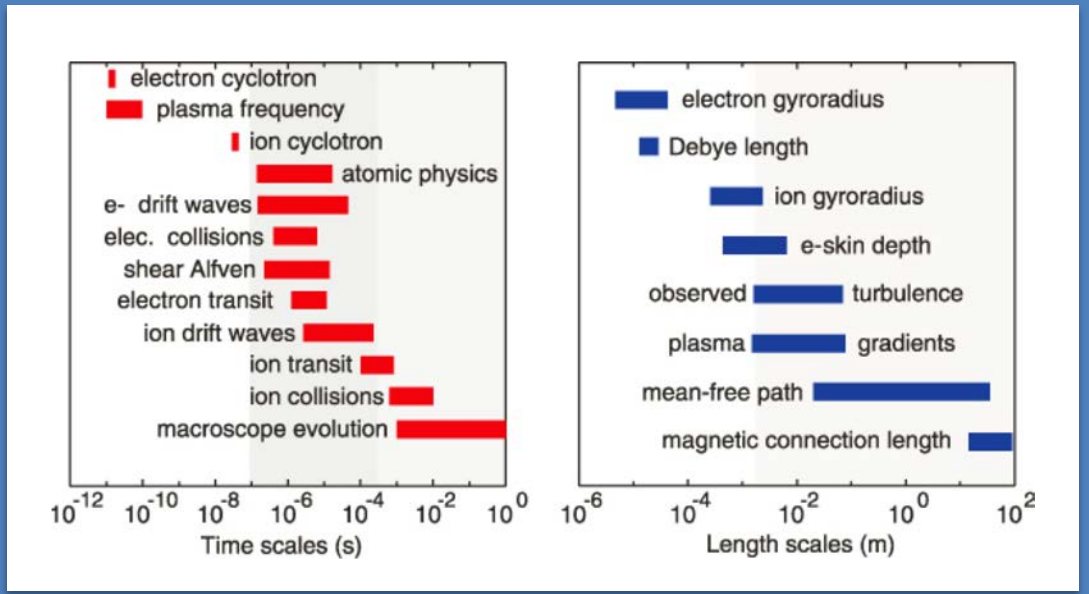
Large-Scale Physical Simulation: Why?

- **We can't do the math!**
  - We want to understand models
  - Lots of mathematical tools to analyze and solve small systems of *linear* equations
  - Most problems of interest are described by *nonlinear* models and/or large systems of equations
- **We can't do the experiments!**
  - Too difficult
  - Too dangerous
  - Too expensive
  - Too slow

Many physical problems have relevant physics across wide ranges of scales

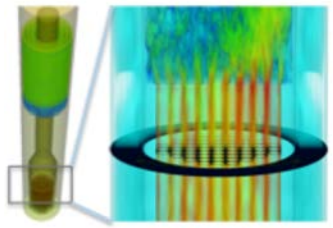


Length Scales in the DIII-D Tokomak Edge Plasma



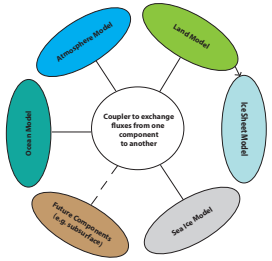
Multiphysics is a primary motivator for large-scale simulation

Multiphysics: More than one component governed by its own principle(s) for evolution or equilibrium

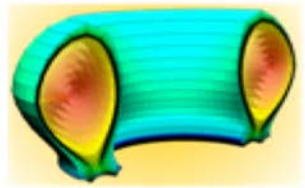


Nuclear reactors
A. Siegel, ANL

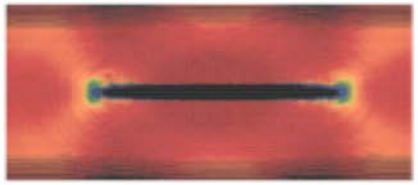
Particle accelerators
K. Lee, SLAC



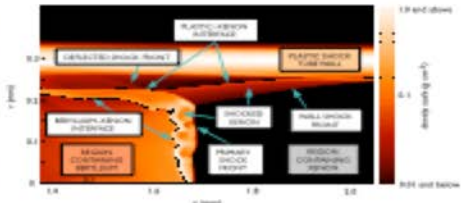
Climate
K. Evans, ORNL



Fusion
A. Hakim, PPPL



Crack propagation
E. Kaxiras, Harvard



Radiation hydrodynamics
E. Myra, Univ. of Michigan

IJHPCA, Feb 2013
Vol 27, Issue 1, pp. 4-83

THE INTERNATIONAL JOURNAL of HIGH PERFORMANCE COMPUTING APPLICATIONS

Multiphysics simulations: Challenges and opportunities

The International Journal of High Performance Computing Applications
27(1) 4–83

DOI: 10.1177/1094342012468181
hpc.sagepub.com
SAGE

David E Keyes[1,2], Lois C McInnes[3], Carol Woodward[4], William Gropp[5], Eric Myra[6], Michael Pernice[7], John Bell[8], Jed Brown[3], Alain Clo[1], Jeffrey Connors[4], Emil Constantinescu[3], Don Estep[9], Kate Evans[10], Charbel Farhat[11], Ammar Hakim[12], Glenn Hammond[13], Glen Hansen[14], Judith Hill[10], Tobin Isaac[15], Xiangmin Jiao[16], Kirk Jordan[17], Dinesh Kaushik[3], Efthimios Kaxiras[18], Alice Koniges[8], Kihwan Lee[19], Aaron Lott[4], Qiming Lu[20], John Magerlein[17], Reed Maxwell[21], Michael McCourt[22], Miriam Mehl[23], Roger Pawlowski[14], Amanda P Randles[18], Daniel Reynolds[24], Beatrice Rivière[25], Ulrich Rüde[26], Tim Scheibe[13], John Shadid[14], Brendan Sheehan[9], Mark Shephard[27], Andrew Siegel[3], Barry Smith[3], Xianzhu Tang[28], Cian Wilson[2] and Barbara Wohlmuth[23]

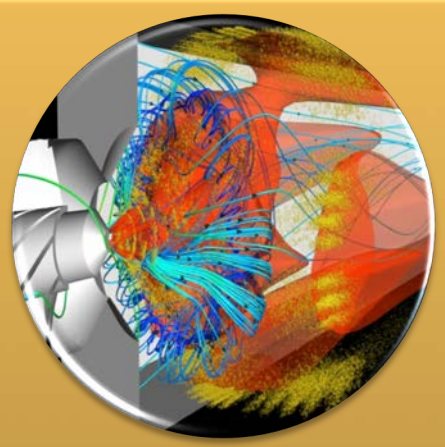
doi:10.1177/1094342012468181

Multiphysics challenges … the study of ‘and’

“We often think that when we have completed our study of one we know all about two, because ‘two’ is ‘one and one.’ We forget that we still have to make a study of ‘and.’ ”

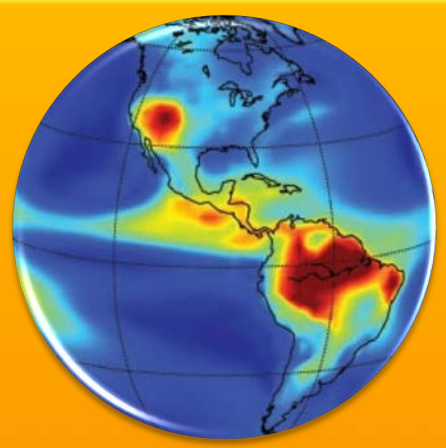
– Sir Arthur Stanley Eddington (1892–1944), British astrophysicist

We lack the computing power to tackle Grand Challenge Science problems



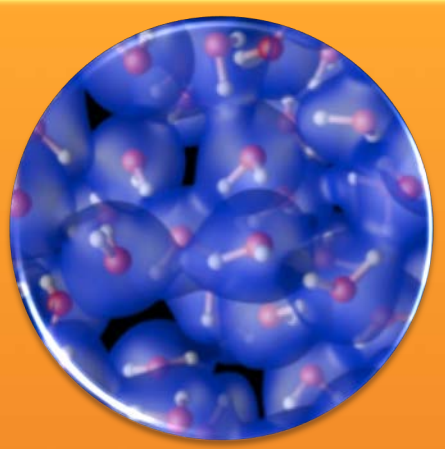
Combustion

- High-pressure, turbulent reacting flow
- Complex moving geometry
- Multiphase: fuel injection and soot
- Stochasticity
- Optimal engine design



Climate

- Coupling atmosphere, oceans, ice sheets, land mass, biosphere
- Global to microscopic
- Catastrophic rare events
- Extreme weather patterns
- Assessments for policy



Materials

- Transient mesoscale behavior of new materials
- Search for novel, optimal materials
- Model from nanometers to microns, femtoseconds to minutes

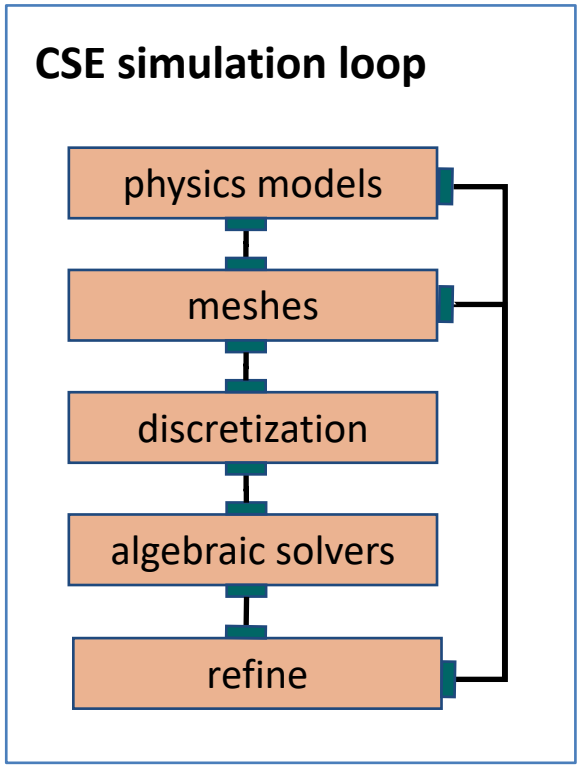
Need (at least) exascale computing resources

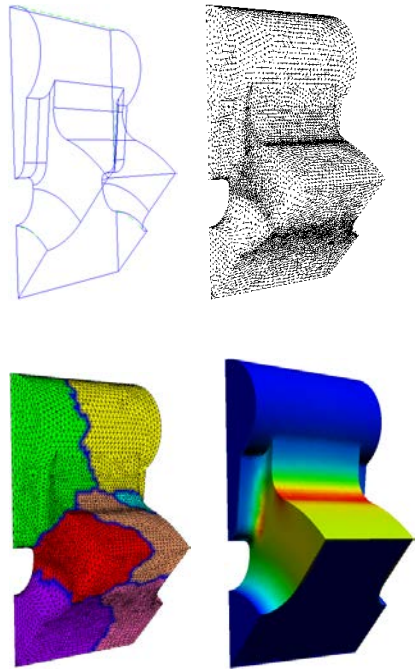
Large-Scale Simulation: How?

- **Emphasis on methods for differential and integral models**
- **Will not discuss in detail**
  - **Discrete Event Simulation**
  - **Meshless methods**
  - **Some types of particle simulations**
  - **Monte Carlo simulation**

CSE simulation starts with forward simulations that capture the physical phenomenon of interest

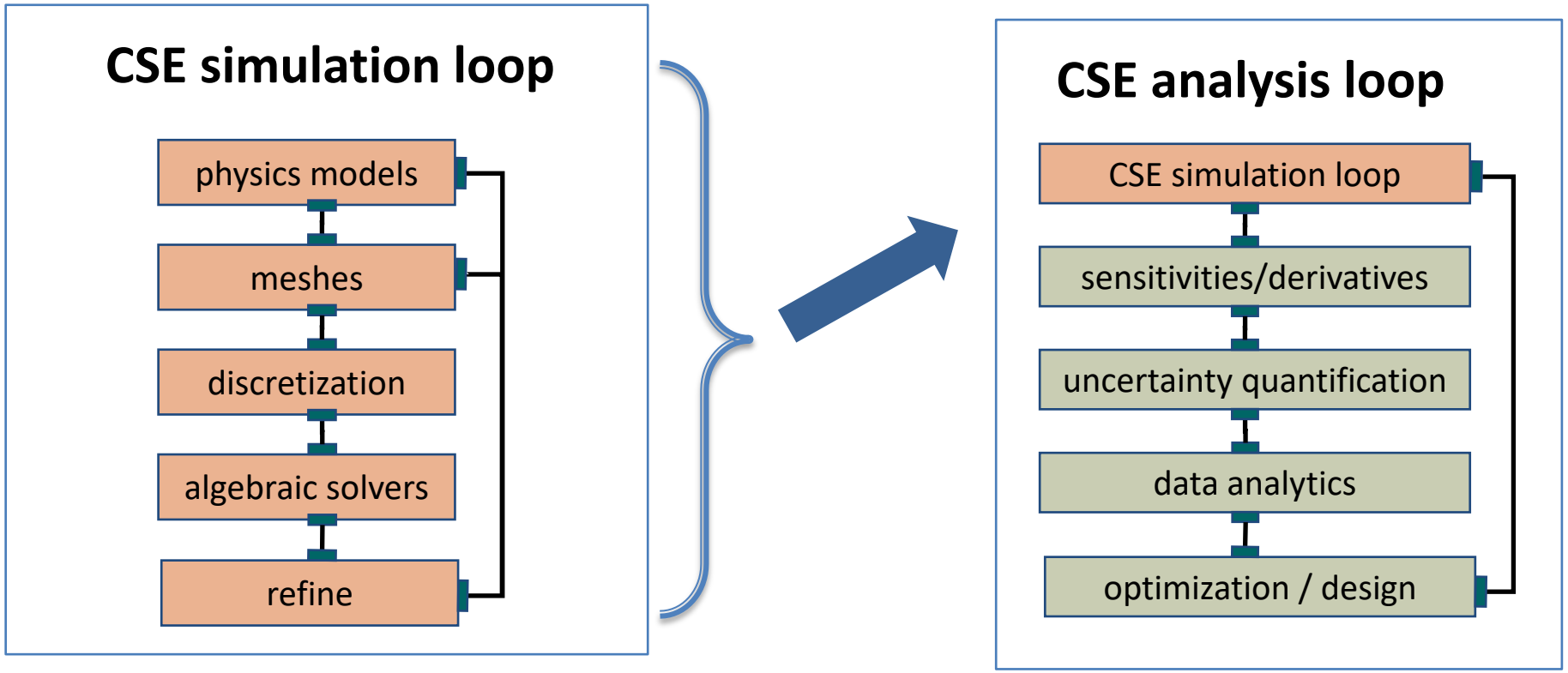
- Develop a mathematical model of the phenomenon of interest
- Approximate the model using a discrete representation
- Solve the discrete representation
- Adapt and refine the mesh or model
- Incorporate different physics, scales





Requires: mesh generation, partitioning, load balancing, high-order discretization, time integration, linear and nonlinear solvers, eigensolvers, mesh refinement, multiscale/multiphysics coupling methods, etc.

CSE analysis builds on the CSE simulation loop … and relies on even more numerical algorithms and software



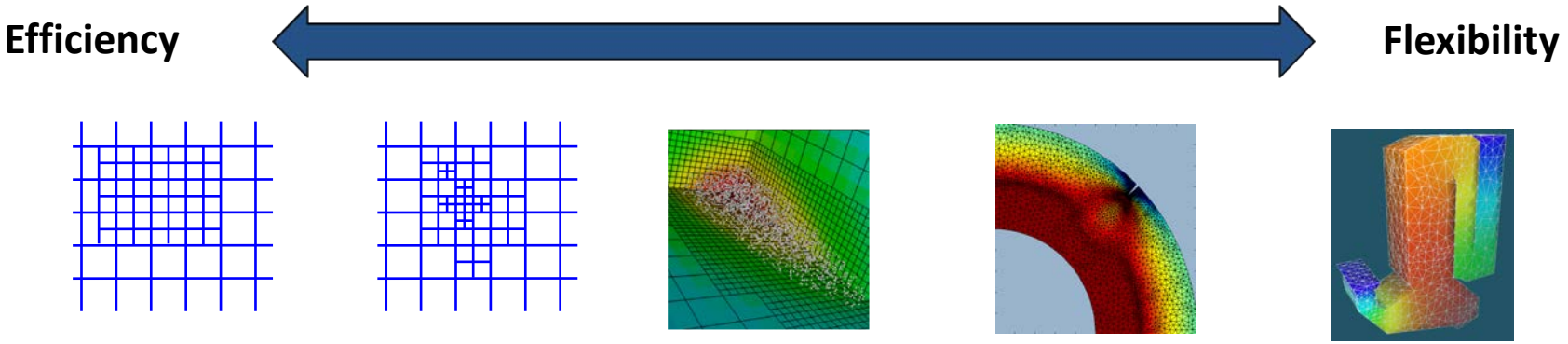
Beyond interpretive simulations …

working toward predictive science

Requires: adjoints, sensitivities, algorithmic differentiation, sampling, ensemble simulations, uncertainty quantification, data analytics, optimization (derivative free and derivative based), inverse problems, etc.

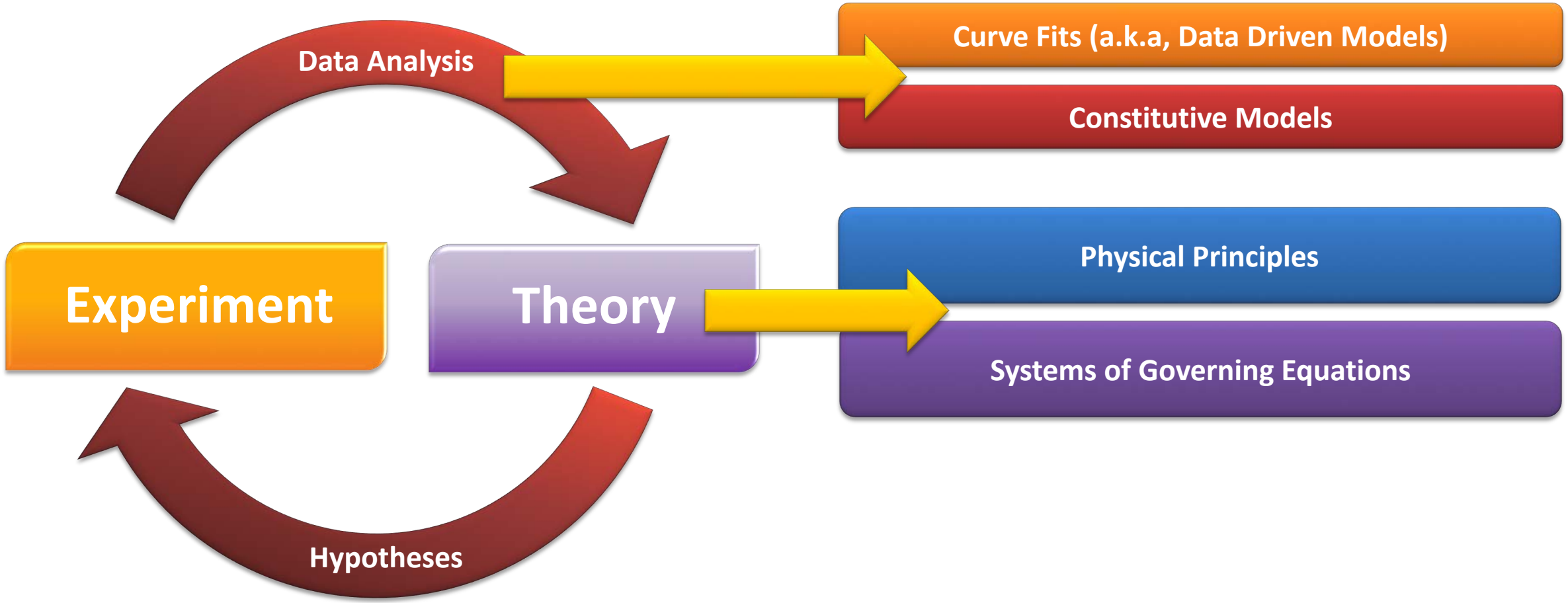
As problems get more complicated, so do the steps in the process

- **Different discretization strategies exist for differing needs**



- **Most problems are time dependent and nonlinear**
  - Need higher algorithmic levels than linear solvers
- **Increasingly combining multiple physical processes**
  - Interactions require careful handling
- Goal-oriented problem solving requires optimization, uncertainty quantification

Our models of reality come from experiment and observation



From a physical principle to a mathematical model using a constitutive relationship

Conservation of Energy: [Physical Principle]

$$\frac{dE}{dt} = 0 = \frac{d}{dt}\iiint_V T\ d^3x + \iint_S q(T)\cdot n\ d^2x$$

By the Divergence Theorem

$$= \iiint_V \left[\frac{\partial T}{\partial t} + \nabla\cdot q(T)\right]\ d^3x$$

Fourier's Law: [Constitutive Relationship]

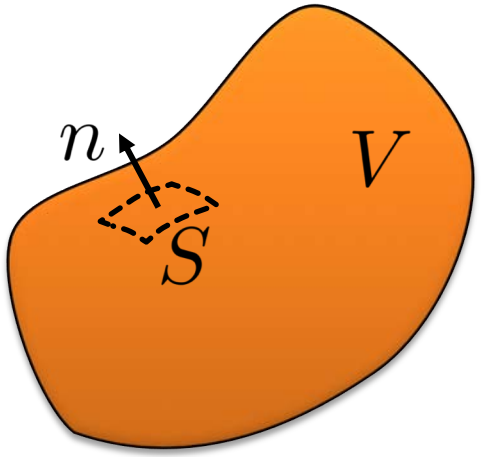
$$q(T) = -\mu\nabla T \implies$$

Heat Equation

$$\frac{\partial T}{\partial t} = \mu\nabla^2 T$$

Assumptions:

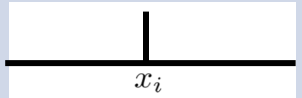
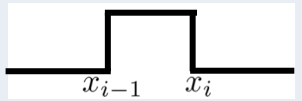
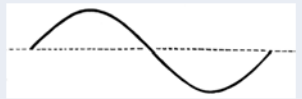
- Solid material
- No work done to volume
- $E \propto T$



Discretizations map infinite dimensional function spaces to finite dimensional vector spaces

- Different choices of basis lead to different types of approximation

$$u(x) = \sum_{j=1}^{N} \alpha_j \phi_j(x)$$

Type	$\phi_j(x)$		Formulation
Finite Difference	$\delta(x - x_j)$		Taylor Series
Finite Volume	$H(x - x_{i-1}) - H(x - x_i)$		Integration and Taylor Series
Finite Element	Local polynomials		Integration by parts
Spectral	$\exp(i2\pi jx)$		Basis orthogonality

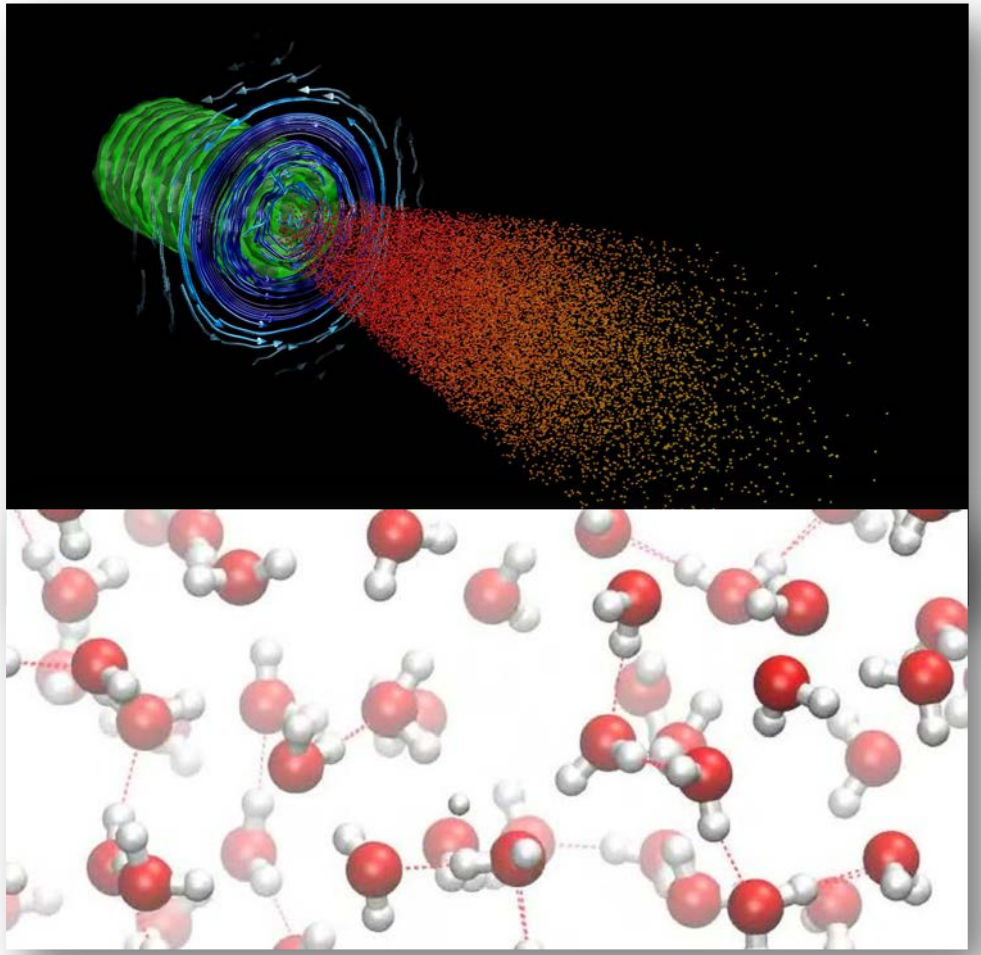
Mesh size (h, Δx), number of elements, number of modes are measures of resolution

Particle-based methods are a special case: they are already discrete spatially

- Particles move under basic physical laws

$$F = ma = m\frac{dv}{dt}$$

- Each particle interacts with other particles and/or external fields
  - Particles may be literal or lumped representations
  - Interactions (forces) alter the particle's trajectory
- Calculations are often done as an ensemble to produce statistical behavior



Let's consider discretizing the spatial variable of the 1D heat equation

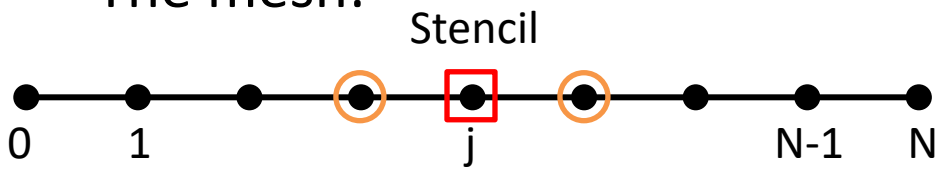
- The problem:

$$\frac{\partial T}{\partial t} = \mu \frac{\partial^2 T}{\partial x^2}$$

$$T(0,t) = T(1,t) = 0$$

$$T(x,0) = \sin(\pi x)$$

- The mesh:



$$x_j = jh, \quad 0 \le j \le N$$

- The spatial discretization:

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{j+1} - 2T_j + T_{j-1}}{h^2} + O(h^2)$$

- The semi-discrete problem

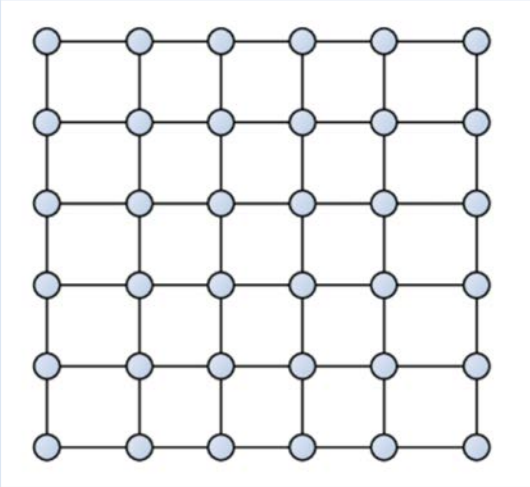
$$\frac{\partial T_j}{\partial t} = \mu \frac{T_{j+1} - 2T_j + T_{j-1}}{h^2}$$

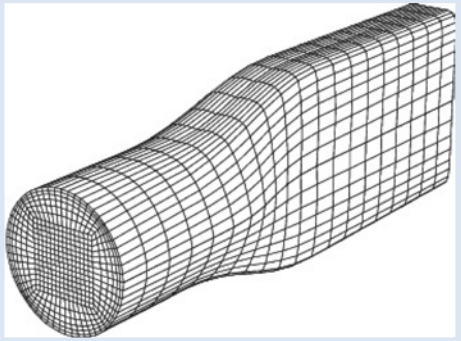
$$T_0(t) = T_N(t) = 0$$

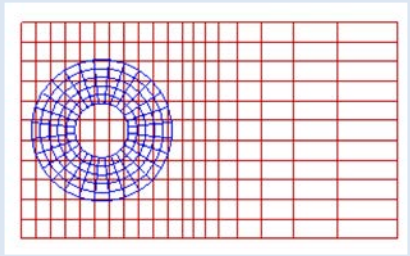
$$T_j(0) = \sin(\pi x_j)$$

$$T_j(t) = T(x_j, t)$$

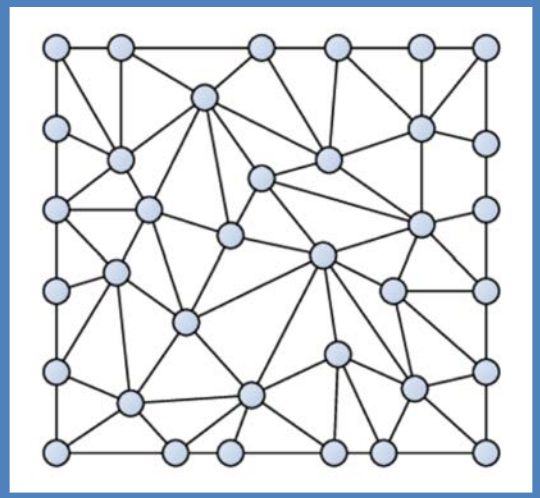
For grid-based approaches, there are two main types of meshes

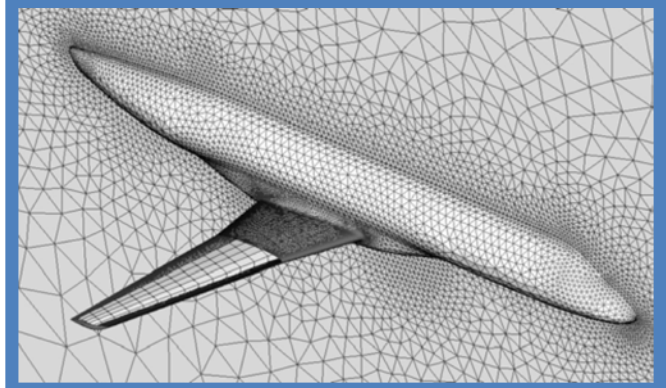






- **Structured Meshes**
  - Regular access patterns
  - Mapped
  - Multiblock
  - Chimera (overlapped)





- **Unstructured meshes**
  - Extra connectivity data
  - Mesh quality
  - Parallel decomposition / load balancing

The time variable also need to be discretized

$$\frac{du}{dt} = f(u,t) \rightarrow u^{n+1} = u^n + \int_{t^n}^{t^{n+1}} f(u,t)dt$$

- **Two main types of temporal discretization:**
  - Explicit – only use known prior values in approximation of integral
  - Implicit – use both known and unknown values in approximation of integral
- **Explicit methods are simpler, but less stable**
  - Do not generally require an explicit matrix inversion
  - Suffer from (sometimes severe) stability constraints on time step size, e.g., CFL number
- **Implicit methods typically are more stable, but more difficult**
  - Require matrix inversion at each time step
  - Allow for larger time step sizes; size is limited by accuracy, not stability

Separately discretizing space and time leads to a Method of Lines formulation

- **Spatial discretization leads to a system of coupled ODEs**
- **Standard ODE integration techniques (Runge-Kutta, Adams, BDF) can be used**

$$T_j^{n+1} = T_j^n + \frac{\mu \Delta t}{h^2}\left[T_{j+1}^n - 2T_j^n + T_{j-1}^n\right]$$

Forward Euler (explicit)

$$T_j^{n+1} - \frac{\mu \Delta t}{h^2}\left[T_{j+1}^{n+1} - 2T_j^{n+1} + T_{j-1}^{n+1}\right] = T_j^n$$

Backwards Euler (implicit)

- **Alternative: Space-time schemes where truncation error of temporal and spatial terms is balanced to achieve higher order with narrower stencils**

Now we have a finite dimensional nonlinear system of algebraic equations that must be solved

$$T_j^{n+1} - \frac{\Delta t}{h^2}\mu(T^{n+1})\left[T_{j+1}^{n+1} - 2T_j^{n+1} + T_{j-1}^{n+1}\right] = T_j^n$$

- To solve this nonlinear problem, we linearize

$$F(T) = 0 \quad \rightarrow \quad T^{(k+1)} = T^{(k)} - [F'(T^{(k)})]^{-1} F(T^{(k)})$$

  and iterate to convergence

- This is an example of Newton's method
- Other methods include fixed-point iteration, secant method, etc.

This is starting to look expensive. How does one solve such a system of equations distributed across a parallel computer?

Within each nonlinear iteration, we need to solve a linear algebra problem

$$F'(T^{(k)})[T^{(k+1)} - T^{(k)}] = F(T^{(k)}) \quad \rightarrow \quad Ax = b$$

- **Direct methods:**
  - Gaussian elimination
  - LU decomposition
  - Not practical for large problems

- **Iterative methods**
  - Jacobi/Gauss Seidel/SOR
  - Krylov Subspace Methods:
    - Conjugate Gradient (CG)
    - Generalized Minimum Residual (GMRES)
    - Biconjugate Gradient Stabilized (BiCGSTAB)
  - Multigrid

We finally have the problem in the form computers can solve: a finite-dimensional, linear algebraic system requiring only the basic operations of addition, subtraction, multiplication, and division

For our heat equation example, we have the following system

$$A = I - \frac{\Delta t}{h^2} \begin{pmatrix} -2\mu'_1 & \mu'_2 & 0 & 0 & \dots & 0 \\ \mu'_1 & -2\mu'_2 & \mu'_3 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & \mu'_{N-2} & -2\mu'_{N-1} & \mu'_N \\ 0 & \dots & 0 & 0 & \mu'_{N-1} & -2\mu'_N \end{pmatrix}$$

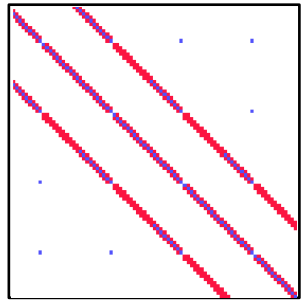
$$x_j = T_j^{(k+1)} - T_j^{(k)}$$

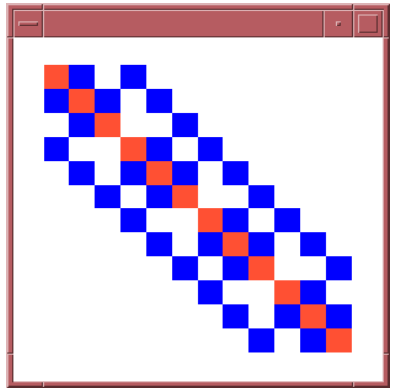
$$b_j = T_j^{(k)} - T_j^n - \frac{\Delta t}{h^2} \mu_j^{(k)} \left[T_{j+1}^{(k)} - 2T_j^{(k)} + T_{j-1}^{(k)} \right]$$

Note that there is structure to the matrix…and a lot of zeros

As problems grow in size, so do corresponding discrete systems

- **Targeting applications with billions grid points and unknowns**
- **Most linear systems resulting from these techniques are LARGE and sparse**
- **Often most expensive solution step**
- **Solvers:**
  - Iterative methods (e.g., Krylov Methods)
    - Preconditioning is typically critical
    - Mesh quality affects convergence rate
- **Many software tools deliver this functionality as numerical libraries: hypre, PETSc, SuperLU, Trilinos, etc.**





Preconditioning is done to accelerate convergence of iterative methods

- Condition number of a matrix

$$\kappa(A) = ||A^{-1}|| \cdot ||A|| = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$$

- If the condition number is large (relative to 1), convergence is slow
- Precondition to reduce the condition number:

$$PAx = Pb$$

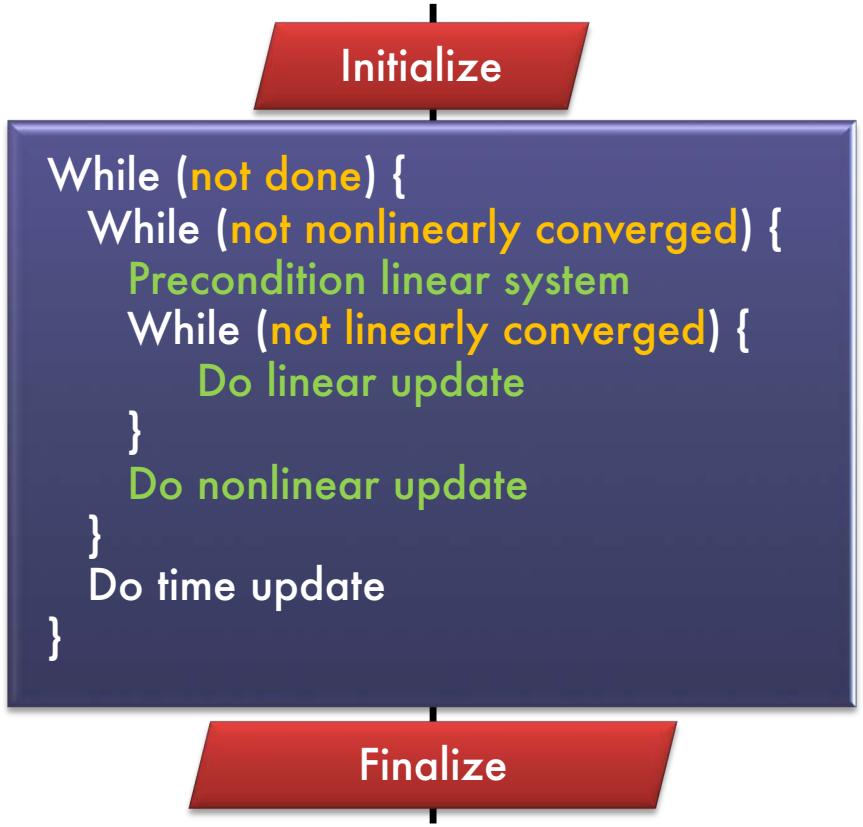
- Ideal preconditioner is A^{-1}
- Many strategies to form P: multigrid, domain decomposition, ILU, physics-based, low-order surrogates, etc.

What keeps computational mathematicians up at night?

- **Your problem should be well-posed (Hadamard):**
  - A solution exists
  - The solution is unique
  - The solution's behavior changes continuously with the initial conditions
- **Your discretization should be consistent:**
  - The discrete approximation applied to any smooth function approaches the original differential equation applied to that same smooth function in the limit of vanishing mesh size
- **Your discretization should be stable:**
  - The discrete approximate solution grows no faster than the solution of the DE, i.e., it doesn't grow unbounded (blow up) unless the solution of the DE does
- **Consistency and stability are necessary, but not sufficient conditions to ensure convergence to the solution of the DE**

Generically, the simulation codes are one to three nested loops

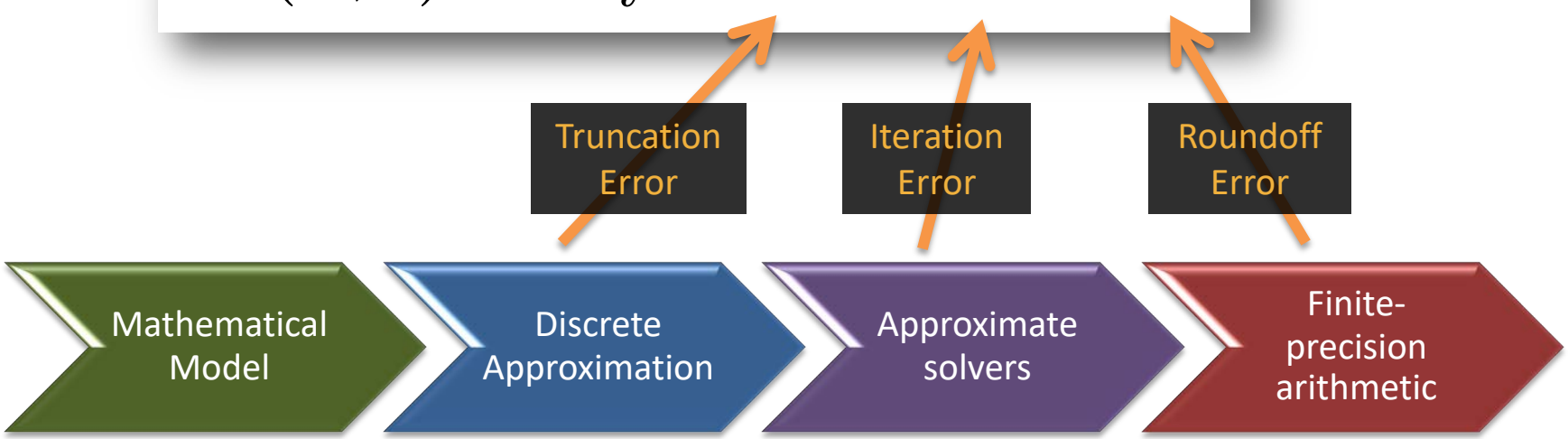
Forward Simulation – " Inner loop"



- Time advancement loop
  - Not present for equilibrium or eigenvalue problems
- Nonlinear iteration loop
  - Not needed for linear problems
  - Typically needed for implicit time advance
- Linear iteration loop
  - Not needed if an explicit time advance or direct solver is used

Simulations produce solutions with many kinds of approximation error

$$u(x,t) = u_i^n + \epsilon_{\mathrm{T}} + \epsilon_{\mathrm{I}} + \epsilon_{\mathrm{R}}$$



Numerical analysis is really about understanding and controlling these approximation errors

CASC

Center for Applied
Scientific Computing

Lawrence Livermore
National Laboratory