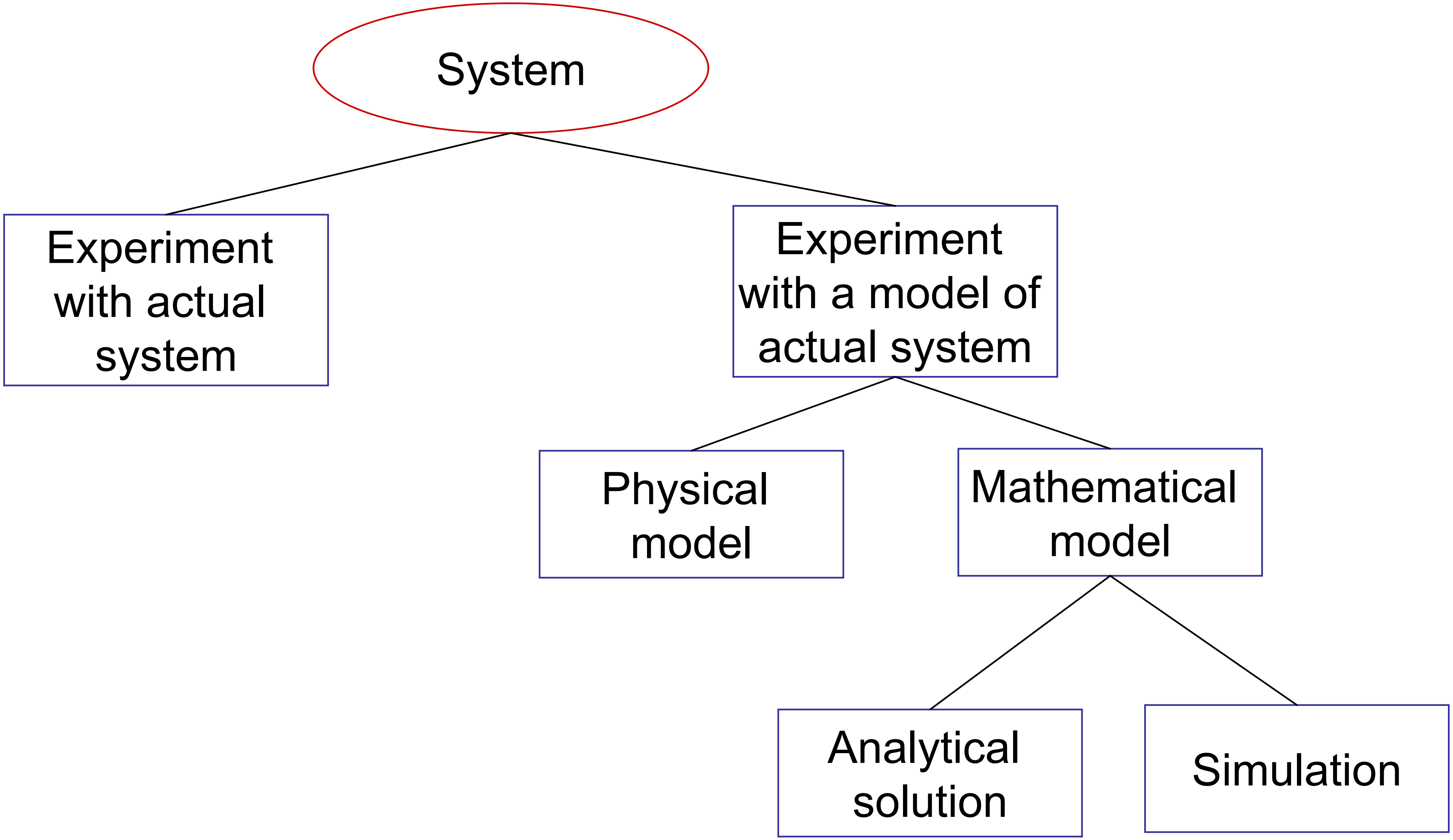
Uncertainties in Modeling & Simulation

Prof. Yan Wang
Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA 30332, U.S.A.
yan.wang@me.gatech.edu

Learning Objectives

- To understand fundamental concepts of Modeling & Simulation (M&S)
- To understand the major sources of epistemic uncertainty in M&S

How to study a system



What is Modeling?

Why mathematical modeling?

- Advantages

- Disadvantages

An example of modeling

- Free fall model

$$\frac{dv}{dt} = g = \frac{F_D}{m}$$

$$\frac{dv}{dt} = \frac{F_D + F_U}{m} = \frac{F_D - cv}{m} = g - \frac{c}{m}v$$

Resistance/
Drag force

$$\frac{dv}{dt} = \frac{F_D + F_U}{m} = \frac{F_D - c_1 v + c_2 \frac{dv}{dt}}{m}$$

White noise

$$\frac{dv}{dt} = \frac{F_D + F_U}{m} = \frac{F_D - \left(c_0 + \gamma\right) v + c_2 \frac{dv}{dt}}{m}$$

Mathematical model

- *Dependent variable=f(Independent variable)*

$$y = f(x)$$

- High dimensional

$$y = f(x_1, x_2, \ldots)$$

- Parametric systems

$$y = f(x, u)$$

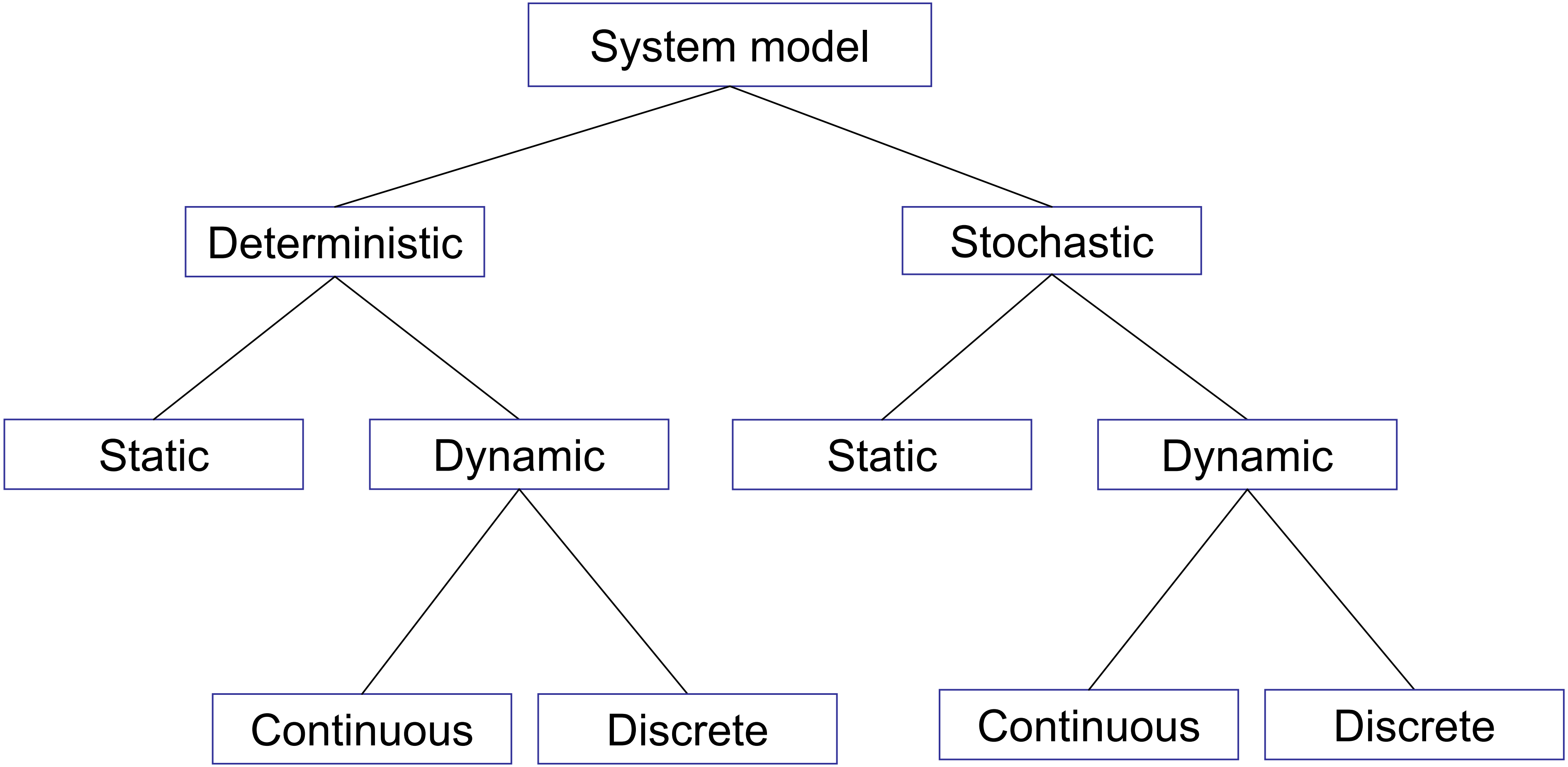
- "Noisy" systems

$$y = f(x, u, \gamma)$$

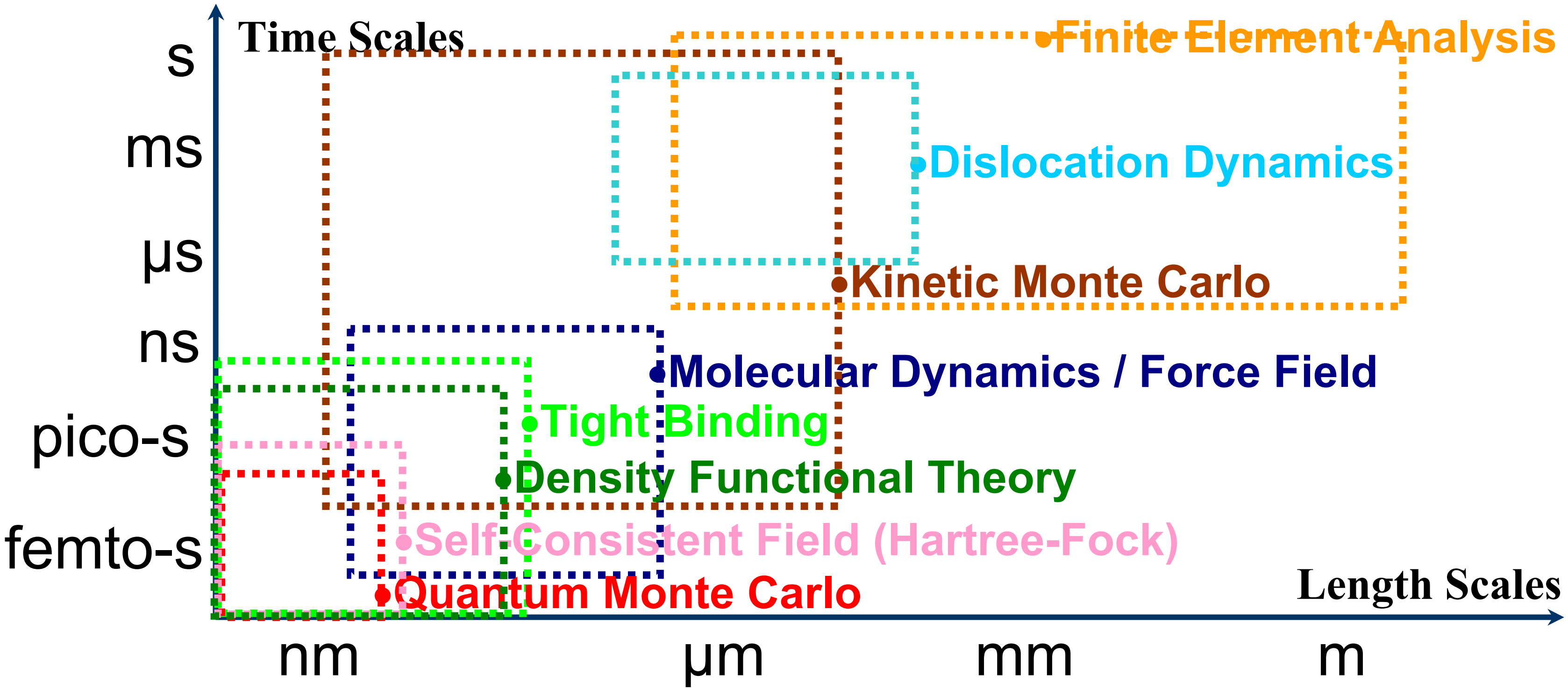
Complexity of Mathematical Models

Simple	Linear	Algebraic Equation / Closed-form	Static
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
Complex	Nonlinear	Differential Equation	Dynamic

Model Taxonomy

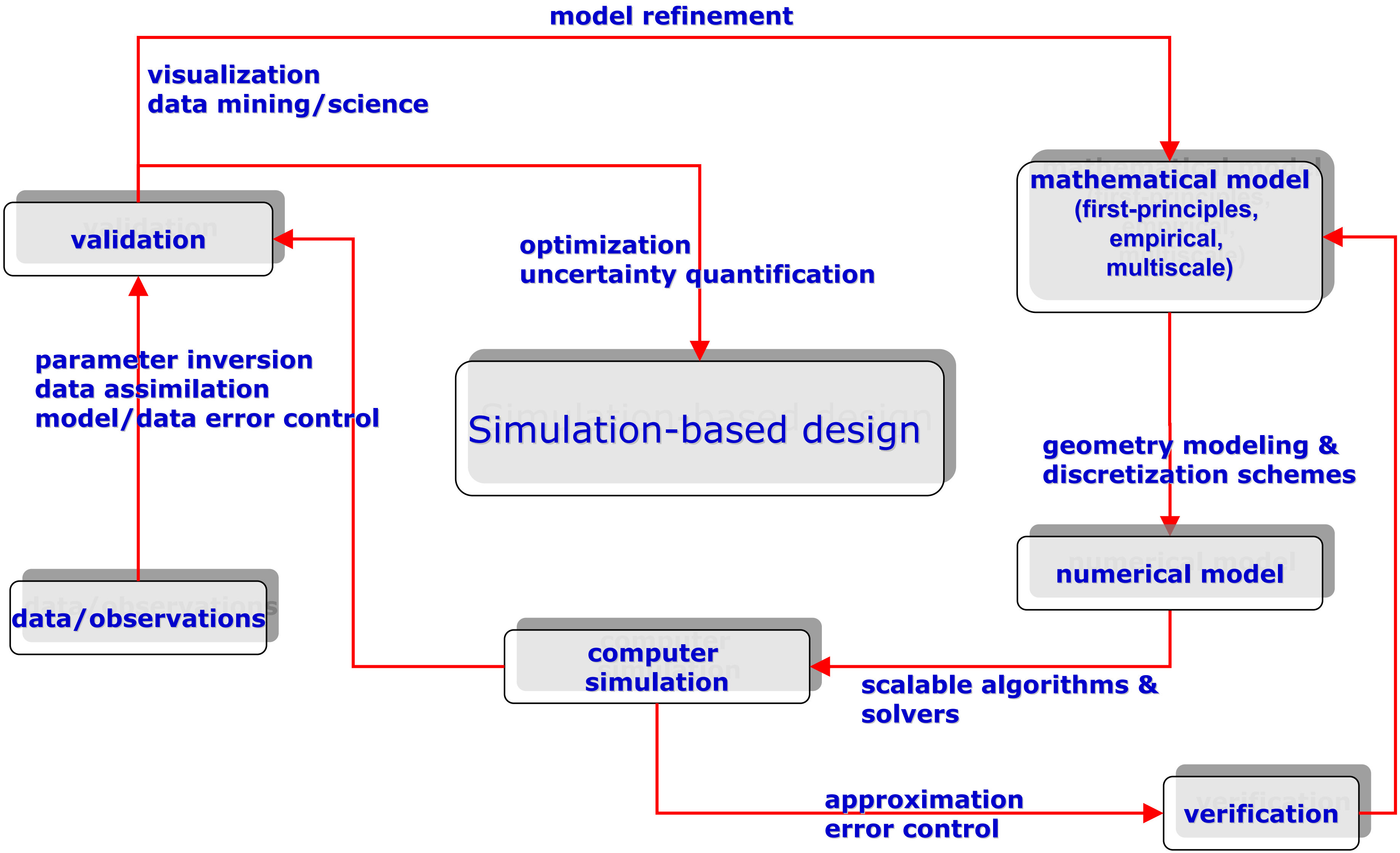


Modeling & Simulation at Multiple Scales



Various methods used to simulate at different length and time scales

Simulation-based Design

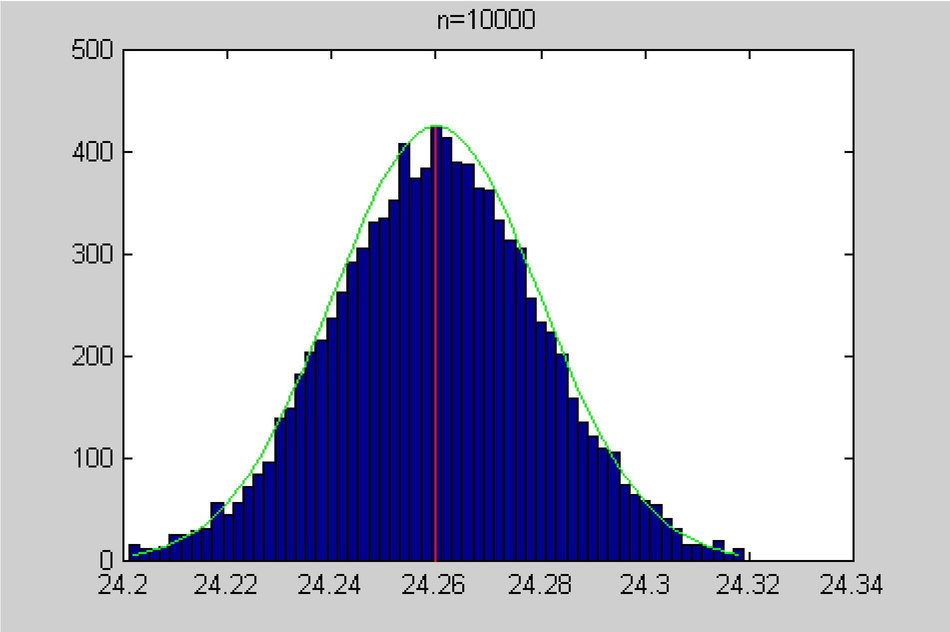


Two types of "uncertainties"

- Aleatory uncertainty (**variability**, irreducible uncertainty, random error)
  - inherently associated with the randomness/fluctuation (e.g. environmental stochasticity, inhomogeneity of materials, fluctuation of measuring instruments)
  - can only be reduced by taking average of multiple measurements.
- Epistemic uncertainty (incertitude, reducible uncertainty, systematic error)
  - imprecision comes from scientific ignorance, inobservability, lack of knowledge, etc.
  - can be reduced by additional empirical effort (such as calibration).

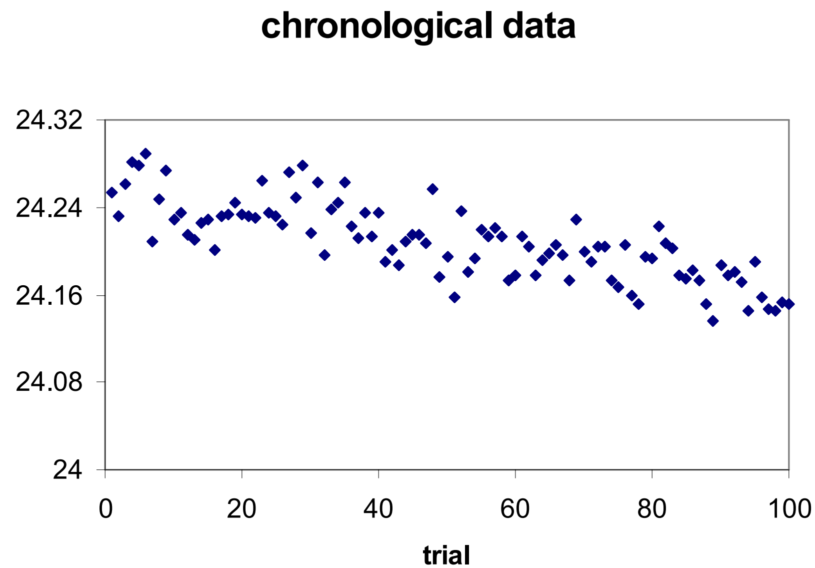
Random Error

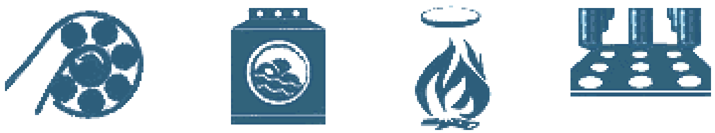
- Determines the *precision* of any measurement
- *Always* present in every physical measurement
  - Better apparatus
  - Better procedure
  - Repeat
- Estimate



Systematic Error

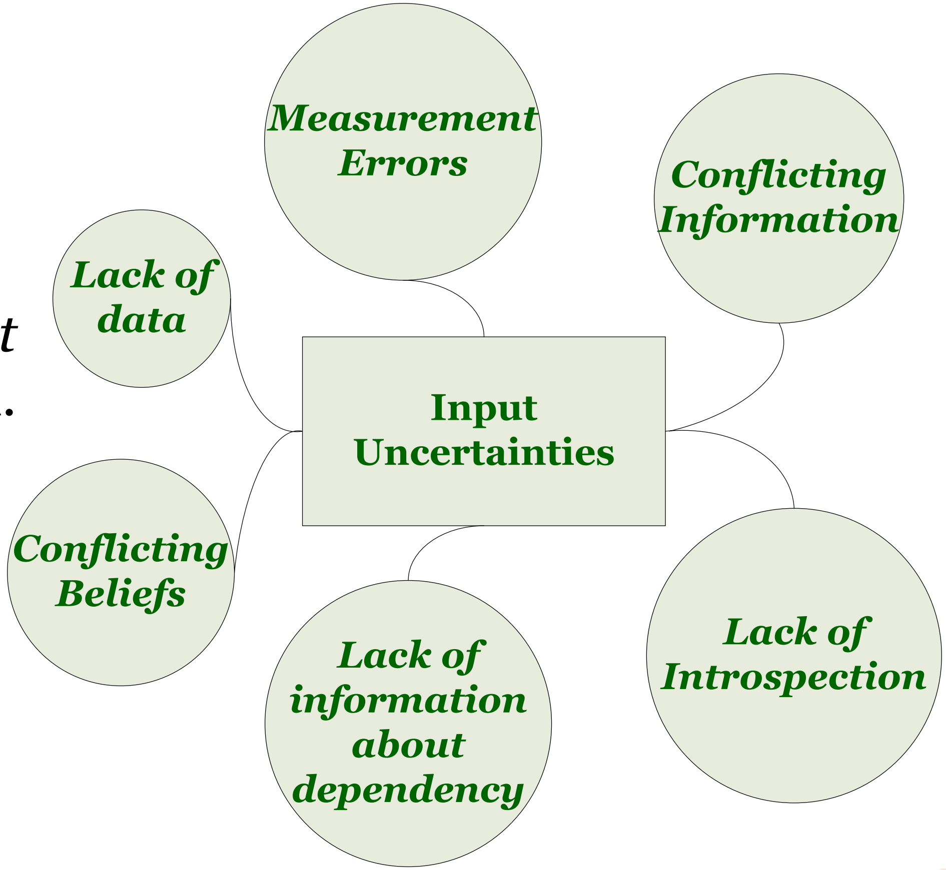
- Determines the *accuracy* of any measurement
- *May* be present in every physical measurement
  - calibration
  - uniform or controlled conditions (e.g., avoid systematic changes in temperature, light intensity, air currents, etc.)
- Identify & eliminate or reduce





Uncertainties in Modeling & Simulation

- ***Aleatory Uncertainty:***
  - *inherent randomness* in the system. Also known as:
    - stochastic uncertainty
    - variability
    - irreducible uncertainty
- ***Epistemic Uncertainty:***
  - due to *lack of perfect knowledge* about the system. Also known as:
    - Incertitude
    - system error
    - reducible uncertainty



Approximations in simulation

- From mathematical models to numerical models
  - Taylor series
  - Functional analysis
- From numerical models to computer codes
  - Discretization (differentiation, integration)
  - Searching algorithms (solving equations, optimization)

Mathematical models → Numerical models
Approximation in Taylor Series

- Truncation

$$
\begin{aligned}
f(x) &= f(x_0) + f'(x_0)(x - x_0) + O(h^2) \\
&= f(x_0) + f'(x_0)h + O(h^2)
\end{aligned}
$$

$$
f(x) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + O(h^3)
$$

Mathematical models → Numerical models Functional Analysis

- Convert complex functions into simple and computable ones by transformation in vector spaces
  - Fourier analysis
  - Wavelet transform
  - Polynomial chaos expansion
  - Spectral methods
  - Mesh-free methods
  - …

Mathematical models → Numerical models
Functional Analysis

- Approximate the original $f(x)$ by linear combinations of basis functions $\psi_i(x)$'s as

$$f(x) \approx \sum_{i=0}^{N} c_i \psi_i(x)$$

- In a vector space (e.g. Hilbert space) with an infinite number of dimensions

$$f(x) = \sum_{i=0}^{\infty} c_i \psi_i(x)$$

Mathematical models → Numerical models
Functional Analysis

- An *inner product* $\left\langle f, g \right\rangle$ is defined as a "projection" in the vector space, such as

$$\left\langle f, g \right\rangle := \int_{-\infty}^{\infty} f(x)g(x)W(x)dx$$

- Typically we choose orthogonal basis functions $\psi_i(x)$'s such that

$$\left\langle \psi_i, \psi_j \right\rangle = \int_{-\infty}^{\infty} \psi_i(x)\psi_j(x)W(x)dx = \begin{cases} Constant & (\forall i, j, i = j) \\ 0 & (i \neq j) \end{cases}$$

for orthonormal basis functions

$$\left\langle \psi_i, \psi_j \right\rangle = \int_{-\infty}^{\infty} \psi_i(x)\psi_j(x)W(x)dx = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}$$

Mathematical models → Numerical models
Functional Analysis

- The coefficients c_i's are computed by

$$c_i = \frac{\langle f, \psi_i \rangle}{\langle \psi_i, \psi_i \rangle}$$

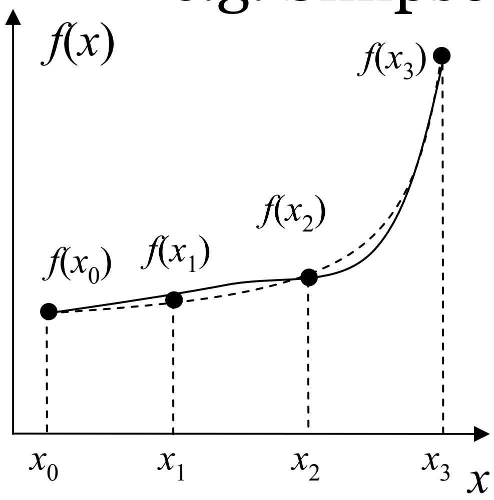
- The computable function is

$$f(x) \approx \sum_{i=0}^{N} c_i \psi_i(x)$$

with ***truncation***!

Numerical Model → Computer Code
Compute integrals

- Quadrature
  - Approximate the integrand function by a polynomial of certain degree
  - Approximate the integral by the weighted sum of regularly sampled functional values
    - e.g. Simpson's 3/8 rule



$$I \approx \int_{x_0}^{x_3} f^{(3)}(x)\,dx = \frac{3h}{8}\left[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)\right]$$

Numerical Model → Computer Code
Compute integrals

- Monte Carlo simulation
  - Let $p(u)$ denote uniform density function over $[a,\ b]$
  - Let U_i denote i th uniform random variable generated by Monte Carlo according to the density $p(u)$
  - Then, for "large" N

$$\int_a^b f(x)dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(U_i)$$

  - Variance reduction (importance sampling) to improve efficiency

Numerical Model → Computer Code
Compute derivatives

- Finite-divided-difference methods
  - Approximated derivatives come from Taylor series
    - e.g. forward-finite-difference

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + O(h^2) = f(x_i) + f'(x_i)h + O(h^2)$$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h)$$

Floating-Point Representation

- How does computer represent numbers?

Perfect world

Imperfect world

Ariane 5

- Exploded 37 seconds after liftoff
- Cargo worth $500 million

- Why
  - Computed horizontal velocity as floating point number
  - Converted to 16-bit integer
  - Worked OK for Ariane 4
  - Overflowed for Ariane 5
    - Used same software

Do you trust your computer?

Rump's function:

$$f(x,y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y}$$

$$f(x = 77617, y = 33096) = ?$$

- Single precision: f = 1.172603...
- Double precision: f = 1.1726039400531...
- Extended precision: f = 1.172603940053178...
- Correct one is: f = -0.827396059946821**3**

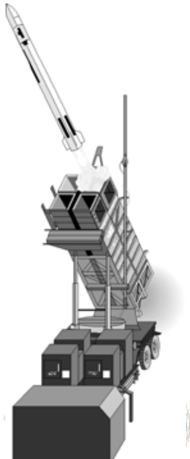
Bug?

Defect?

Another Story

- On February 25, 1991
- A Patriot missile battery assigned to protect a military installation at Dhahran, Saudi Arabia
- But … failed to intercept a Scud missile
- 28 soldiers died
- … an error in computer arithmetic

$$0.1 \times 10 \neq 1$$



IEEE 574 Standard

IEEE Floating Point Representation

s	exponent E	mantissa T
1 bit	w bits 8 bits	23 bits $t = p - 1$ bits

IEEE Double Precision Floating Point Representation

1 bit	11 bits	52 bits
s	exponent	mantissa

- If $E = 2^w - 1$ and $T \neq 0$, then v is **NaN** regardless of S.
- If $E = 2^w - 1$ and $T = 0$, then $v = (-1)^S \times \infty$.
- If $1 \leq E \leq 2^w - 2$, then $v = (-1)^S \times 2^{E-bias} \times (1 + 2^{1-p} \times T)$;

normalized numbers have an implicit leading significand bit of 1.

- If $E = 0$ and $T \neq 0$, $v = (-1)^S \times 2^{emin} \times (0 + 2^{1-p} \times T)$;

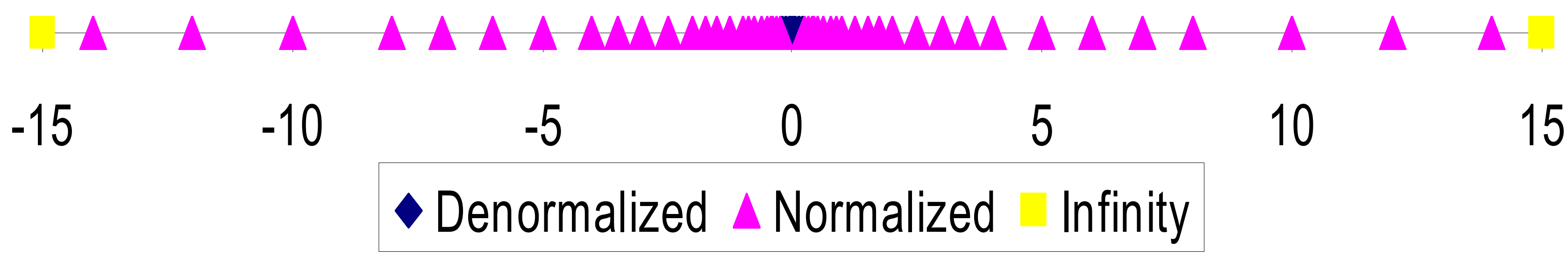
denormalized numbers have an implicit leading significand bit of 0.

- If $E = 0$ and $T = 0$, then $v = (-1)^S \times 0$ (signed zero)

where $bias = 2^{w-1} - 1$ and $emin = 2 - 2^{w-1} = 1 - bias$

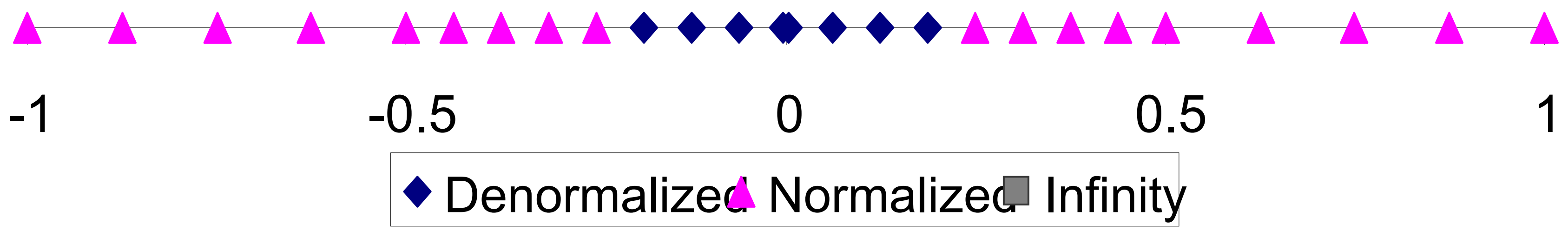
Distribution of Values

- 6-bit IEEE-like format
  - w = 3 exponent bits
  - t = 2 fraction/mantissa bits
  - bias = 3

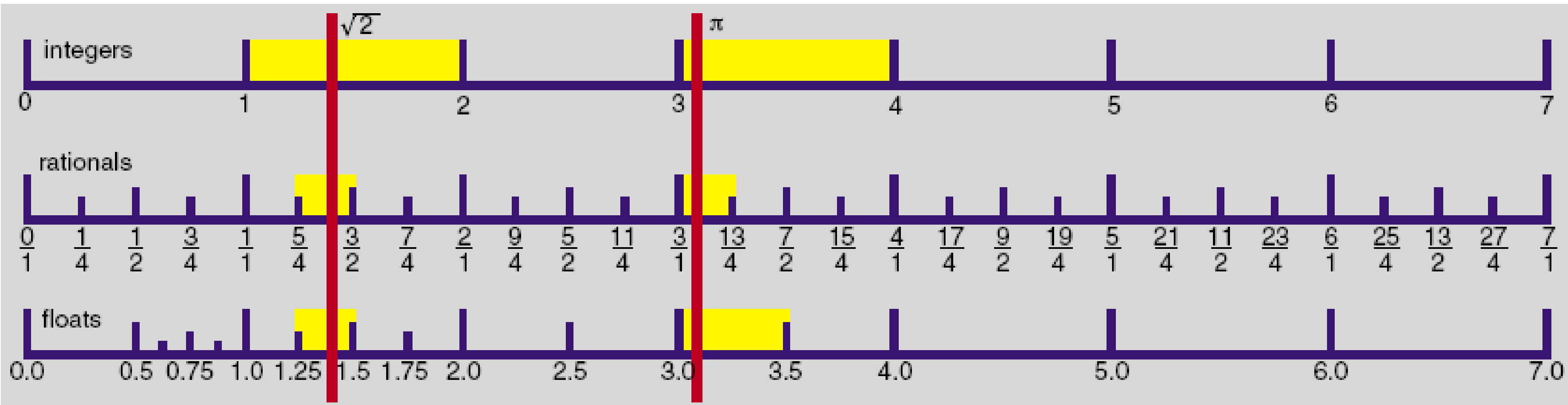


Distribution of Values (zoom-in view)

- 6-bit IEEE-like format
  - w = 3 exponent bits
  - t = 2 fraction/mantissa bits
  - bias = 3



Round-Off Errors



- Overflow error – “not large enough”
- Underflow error – “not small enough”
- Rounding error – “chopping”

- http://www.cs.utah.edu/~zachary/isp/applets/FP/FP.html

Rounding

$$D \times \beta^E = \pm\left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_{p-1}}{\beta^{p-1}}\right)\beta^E$$

- Round by chopping (round toward zero)
  - Truncate base expansion after (p-1)st digit
  - Machine epsilon $\varepsilon_{machine} = \beta^{1-p}$
- Round to nearest (round to even)
  - Last digit is even in case of tie
  - Machine epsilon $\varepsilon_{machine} = \frac{1}{2}\beta^{1-p}$

$$\left|\frac{fl(x) - x}{x}\right| \le \varepsilon_{machine}$$

Cancellation

- Subtraction between two *p*-digit numbers having the same sign and similar magnitudes yields result with fewer than *p* digits.
- Significant digits of two numbers cancel.
- Despite exactness of result, cancellation often implies serious loss of information.
- Relative uncertainty in difference is largely due to previous rounding errors.

$$(1+\varepsilon)-(1-\varepsilon)=1-1=0$$

Special Numbers

Expression	Result
0.0 / 0.0	NaN
1.0 / 0.0	Infinity
-1.0 / 0.0	-Infinity
NaN + 1.0	NaN
Infinity + 1.0	Infinity
Infinity + Infinity	Infinity
NaN > 1. 0	false
NaN == 1.0	false
NaN < 1.0	false
NaN == NaN	false
0.0 == -0.0	true

- standard range of values permitted by the encoding (from 1.4e-45 to 3.4028235e+38 for float)

Floating point Hazards

This expression	does NOT equal to this expression	when
0.0 - f	-f	f is 0
f < g	! (f >= g)	f or g is NaN
f == f	true	f is NaN
f + g – g	f	g is infinity or NaN

```
double s=0;
for (int i=0; i<26; i++) s += 0.1;
System.out.println(s);
```

- The result is
2.6000000000000001

```
double d = 29.0 * 0.01;
System.out.println(d);
System.out.println((int) (d * 100));
```

- The result is
0.29
28

Comparing Floating Point Numbers

- Try to avoid floating point comparison directly
- Testing if a floating number is greater than or less than zero is even risky.
- *Instead*, you should compare the absolute value of the difference of two floating numbers with some pre-chosen epsilon value, and test if they are "close enough“
- If the scale of the underlying measurements is unknown, the test “abs(a/b - 1) < epsilon” is more robust.
- Don’t use floating point numbers for exact values

Uncertainties in M&S

- Model errors due to approximations in truncation or sampling
  - Taylor approximation
  - Functional analysis
- Numerical errors due to floating-point representation
  - Round-off errors

Summary

- Modeling is *abstraction*
- M&S ***always*** has *approximations* involved, which are important sources of epistemic uncertainty.
- Computer tricks us

Abstraction
Approximation
Round-off

Further Readings

- Goldberg, D. (1991) "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys*, **23**(1), 5-48