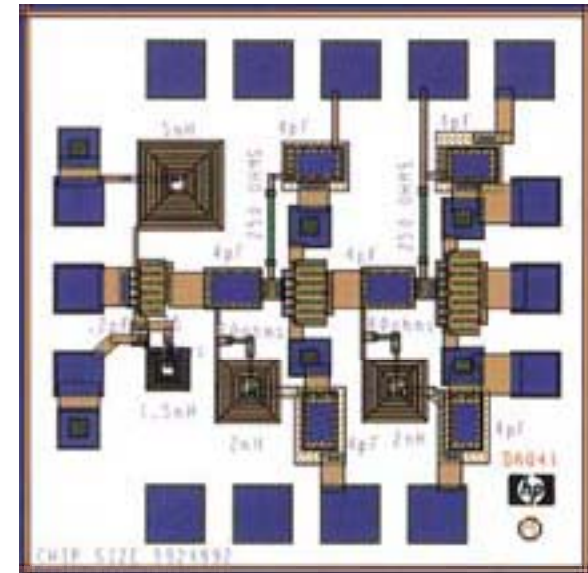
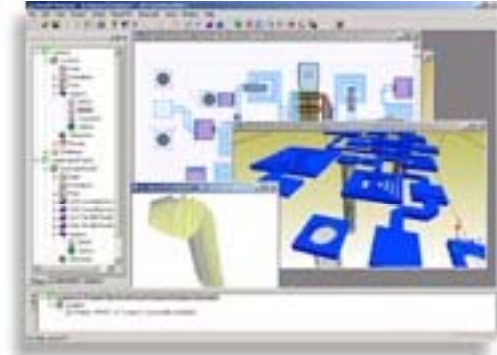




Why Computational Electromagnetics?

- Simulate Physical Phenomena of Electrical (Electromagnetic) signals
 - Stimulate new ideas/new products
- Engineering Design and Analysis
 - Physical prototyping is expensive
 - Accurate CAD design and analysis greatly reduces design cycle time and cost



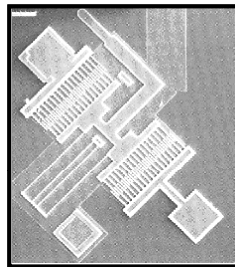
A Variety of Applications ...

Quasi-Static

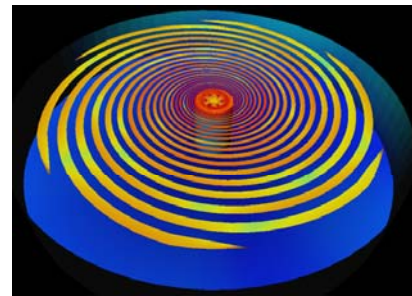
Full-Wave

Quasi-Optical

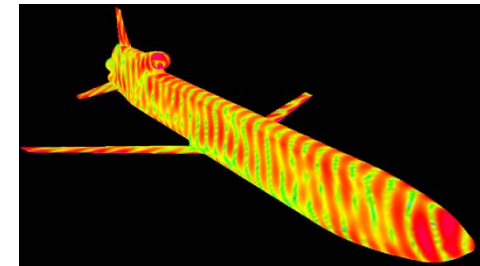
Asymptotic



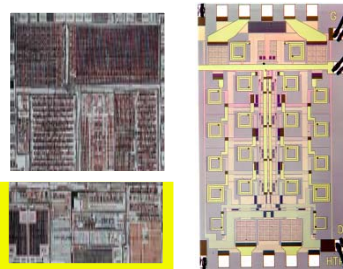
MEMs devices



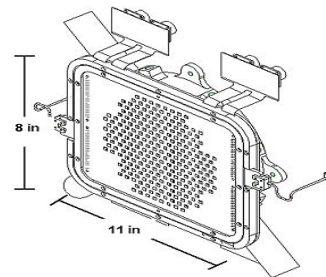
Broadband Antennas



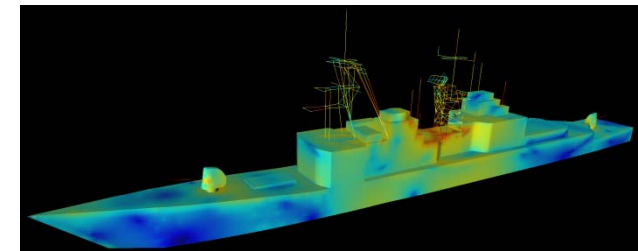
EM Scattering / Imaging



Mixed-Signal Circuits



Phased Array Antennas

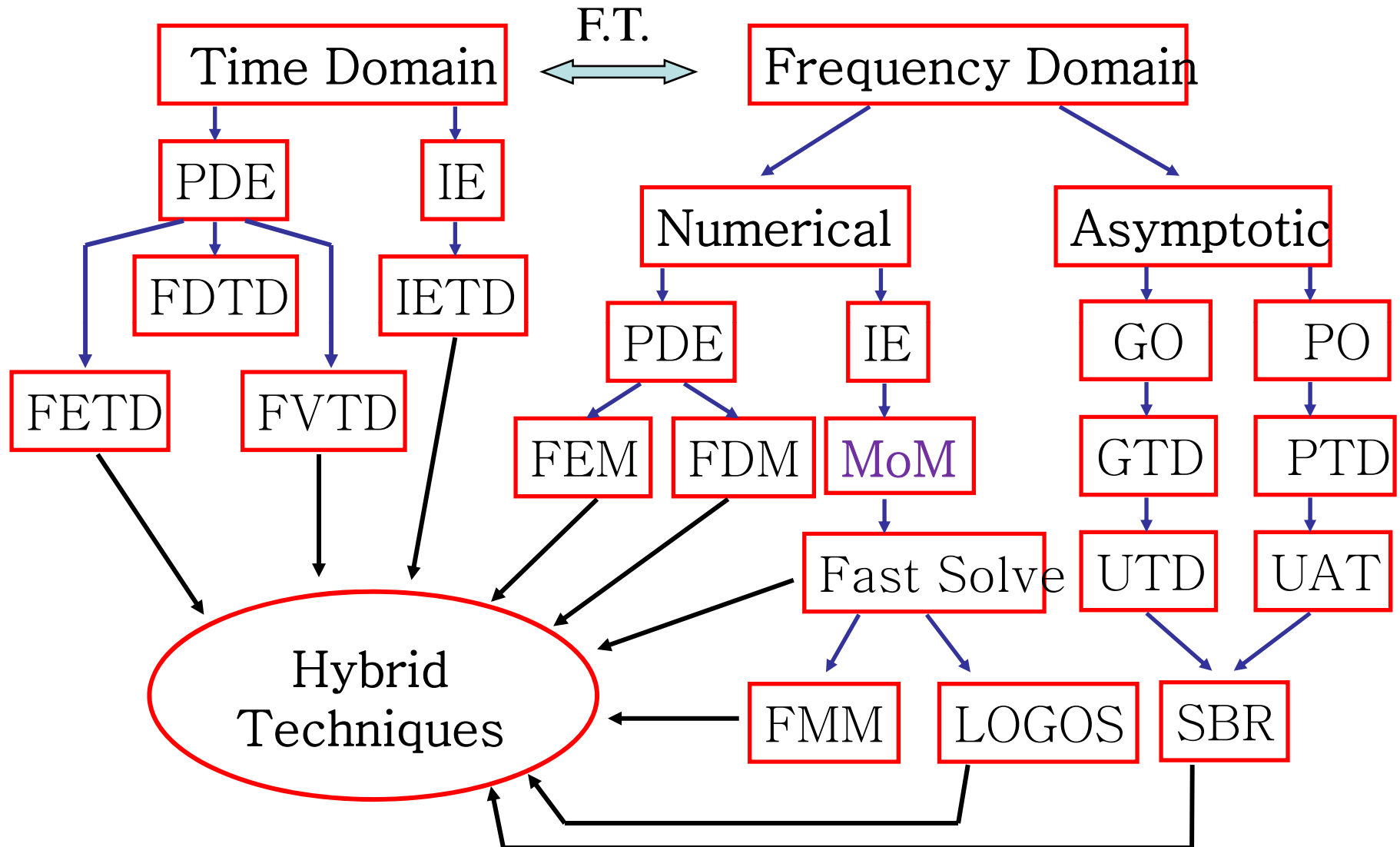


Complex Systems

f = 0 Hz

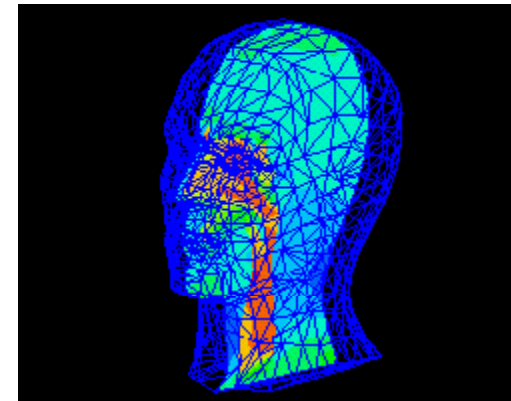
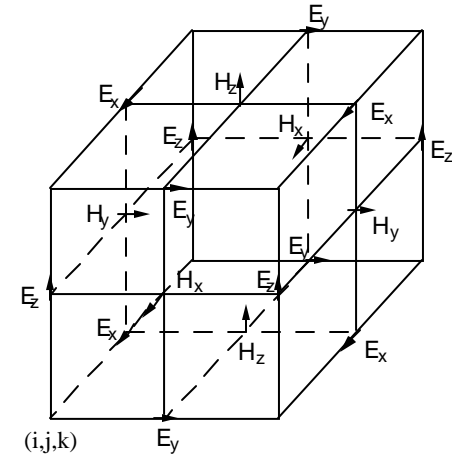
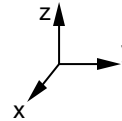


Computational Methods



Partial Differential Equation (PDE) Methods

- Finite-Difference Methods
 - Finite-Difference Time-Domain (FDTD)
 - Structured Volume Discretization
 - Difference approximation of spatial/time derivatives
 - Solution obtained through explicit time-marching scheme (recursive solution)
- Finite-Element Methods (FEM)
 - General Volume Discretization into finite elements
 - Weak form of the wave equation is typically formulated via a variational technique
 - Method of weighted residuals used to solve for the Fields
- Properties:
 - Require solution of large sparse linear system of equations
 - Unbounded problems require radiation boundary conditions
 - Fields are discretized throughout the volume. Discretizations typically must be finely done throughout the entire space since both magnitude & phase are modeled
- Error
 - Error is typically dispersive (phase-error) and anisotropic.
 - Error can grow with the problem size as it is accumulated over distance



Integral Equation (IE) Methods

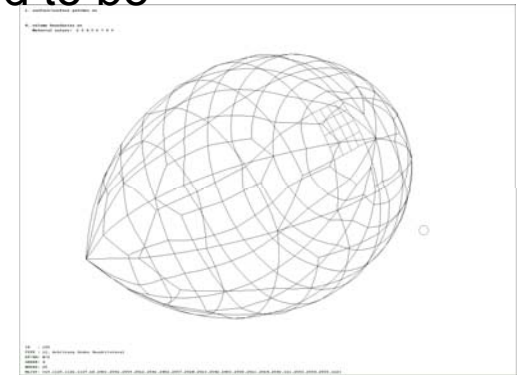
- Integral equations are derived from known solutions to the PDE
 - Green's function
 - For most applications, the Green's function is known and is derived from the solution to the PDE driven by a point source
- Boundary constraints lead to well-posed integral equation formulations
- Unknowns typically bound to the surface
 - Surface current density, charge density
- Surface is discretized using “patches” or “elements”
 - Used to model the geometric surface
 - Functions distributed over the patches represent the unknown surface current
- Method of weighted residuals (“Method of Moments”) is used to solve for unknown current (charge) density
 - Leads to large dense linear system of equations
 - Since the unknowns are restricted to surfaces, unknowns are typically \ll difference methods
 - The matrices are typically fully populated requiring full N^2 storage and N^3 solution time
 - Fast solution techniques can reduce this to N storage and $N \log N$ solution time.
 - Generally the linear systems are better conditioned than PDE methods
 - Radiation condition is automatically satisfied by the Green's function

EE525

- This class focuses on the introduction to formulating solutions to Electromagnetic Interaction Problems via “Integral Equation Solution Methods”
 - The “Method of Moments”
- EE525 lays necessary framework for basic numerical analysis
 - Solution to dense linear systems of equations
 - Numerical integration
 - Green’s functions, equivalence principals and integral equation formulations
 - Method of Moment solutions of integral equations derived for 2D problems, 3D problems, and for thin wire antenna problems
 - Design and Creation of computer software based on IE solution methods

Sources of Approximation

- The integral equations derived provide can provide an *exact* mathematical representation
- The computation of the solution for the surface current density and other desired quantities encounter errors. Such errors need to be understood and controlled
- Sources of Error:
 - Discretization Error (representing the surface geometry)
 - Interpolation Error
 - Empirical Formula error (i.e., impedance boundary condition)
 - Numerical Error:
 - Truncation, rounding
 - Errors can be amplified during the computation (poor conditioning)
 - Sources of error and the amount of error must be understood to maintain confidence in the final solution



ERROR

- Absolute Error

absolute error = approximate solution – true solution

- Relative Error

relative error = (absolute error)/(true solution)

- Observation:

approximation solution = (1+relative error)*(true solution)

– Since the true solution is typically unknown, thus the best we can do is to estimate the error or to derive an error bound

- Typically, error is estimated using the currently known best approximation to the solution

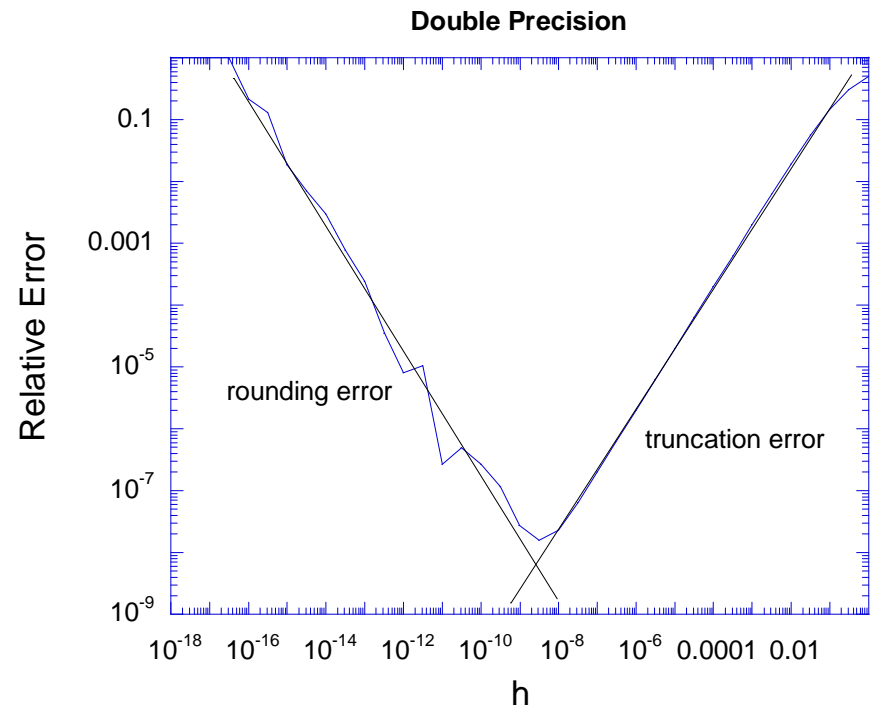
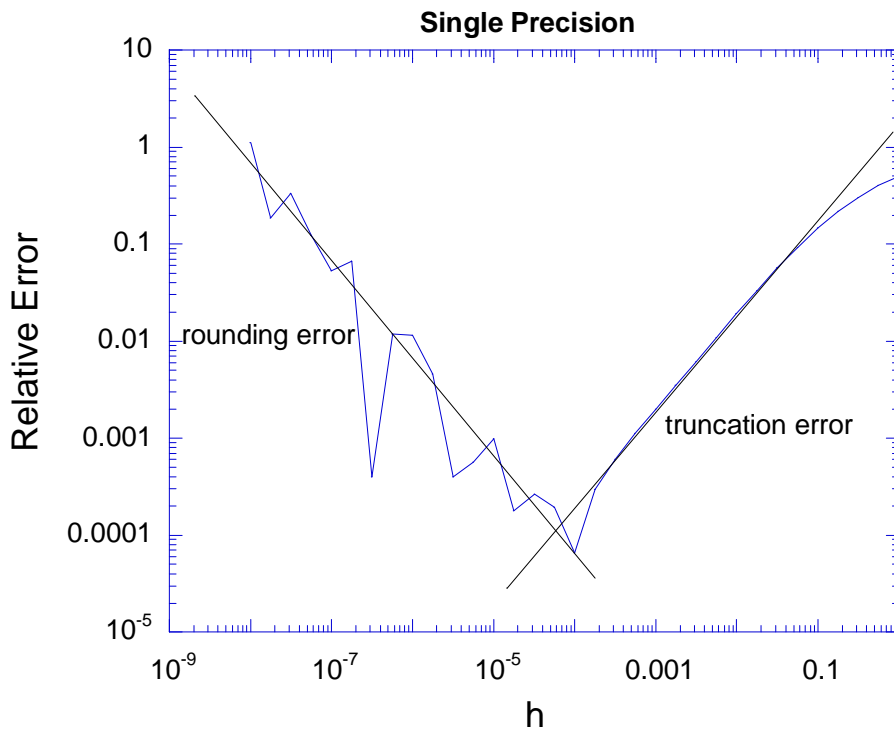
Types of Error

- Computational Error vs. Data Error
 - You wish to evaluate a function f with argument x .
 - True value of x leads to desired result for $f(x)$
 - \underline{x} is the approximate value for x
 - \underline{f} is the approximate function computed
 - Thus
 - $\text{Error} = \underline{f(\underline{x})} - f(x) = (\underline{f(\underline{x})} - \underline{f(x)}) + (\underline{f(x)} - f(x))$
 $= \text{computational error} + \text{propagated data error}$
- Rounding error
 - Difference between the result computed using finite precision arithmetic and the result produced using exact arithmetic.
 - Errors arise due to inexact representation of real numbers and the arithmetic operations on them.
- Truncation Error (Discretization Error)
 - Arises when continuous or infinite operations (such as an integration or summation) are truncated to discrete operations

Example

- Finite difference example
 - Approximate a derivative of a function with a backward difference approximation

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$



Floating Point Numbers

- A floating point number is characterized by a computer using:

B - base or radix

p - precision

$[L,U]$ - exponent range

$d_0d_1d_2d_3\cdots d_{p-1}$ - *mantissa* (d_i are digits)

E - the exponent

- A floating point number is represented as:

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

Typical Floating Point Characteristics

- Almost all computers using binary arithmetic ($B = 2$)
- IEEE Standards:

Standard	B	P	L	U
IEEE SP	2	24	-126	127
IEEE DP	2	53	-1022	1023

- Underflow – the smallest positive number a system can use

$$UF = B^L \quad (2^{-126} = 1.1754943E - 38)$$

$$UF = B^L \quad (2^{-1022} = 2.2250738585072016E - 308)$$

- Overflow – the largest floating point number

$$OF = B^{U+1} (1 - B^{-P}) \quad (IEEE\ SP = 3.4028235E + 38)$$

$$OF = B^{U+1} (1 - B^{-P}) \quad (IEEE\ DP = 8.988465674311579E + 307)$$

Rounding

- Due to finite precision of a discrete numbering system, *not all real numbers are exactly representable*.
 - *Machine numbers* – discrete set of numbers (finite) exactly representable by a computer
- Floating point numbers must be “rounded” to the nearest machine number
 - IEEE standard rounds to nearest machine number
 - Other non-standard systems will truncate or chop to the nearest real
 - Rounding is more accurate
- Machine Precision:
 - Smallest number added to “1” that can be distinguished (single digit)

$$\varepsilon_{mach} = \frac{1}{2} B^{1-p}$$

$$IEEE - SP : \varepsilon_{mach} = 2^{-24} \approx 1E - 7$$

$$IEEE - DP : \varepsilon_{mach} = 2^{-53} \approx 1E - 16$$

Floating-Point Arithmetic

- Addition or Subtraction
 - Mantissa is shifted to make exponents match
 - Precision is determined by the difference in the exponents
- Multiplication
 - Product of two p -digit mantissas contain $2p$ digits. Result is truncated.
 - Precision is determined by mantissa rather than exponent
- Division
 - Quotient of two p -digit mantissas. May contain more than p digits.
 - Result is rounded

Examples

- Addition/Subtraction:
 - Consider: $(1+a) - (1-a) = 2a$
 - Let $a = \varepsilon_{\text{mach}}$
 - Result: $= 2 \varepsilon_{\text{mach}}$. However, the result has at most 1 digit of precision!
 - Let $a < \varepsilon_{\text{mach}}$
 - Result: $= 0$. The result has zero digits of precision
 - Reorder operation: $(1-1) + (a+a) = 2a$
 - p digits of precision maintained
 - Lesson Learned:
 - It is up to the programmer to control the order of operations in a way that minimizes rounding errors

Condition Number

- The conditioning of an operator or set of operations is based on the relative change in the error in the final result given error in the input.
- *Well Conditioned Operator*
 - An operator is well conditioned if small changes in the input lead to small changes in the output (relative to the input)
 - The operator is fairly insensitive to small changes
- *Ill-Conditioned Operator*
 - An operator is ill conditioned if small changes in the input lead to large changes in the output (relative to the input)

- *Condition Number*
$$\sigma = \frac{|\text{relative change in output}|}{|\text{relative change in input}|}$$
$$= \frac{|[L(x + \Delta x) - L(x)] / L(x)|}{|\Delta x / x|}$$

- The larger the condition number, the more ill-conditioned the operator becomes