



Advanced Computation:
Computational Electromagnetics

Introduction to CEM



1

Outline

- What is CEM?
- CEM wisdom
- General concepts in CEM
- Classification of methods

2

What is CEM?

Slide 3

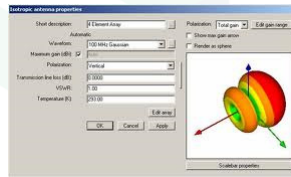
3

Computational Electromagnetics

Definition

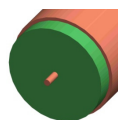
Computational electromagnetics (CEM) is the procedure we must follow to model and simulate the behavior of electromagnetic fields in devices or around structures.

Most often, CEM implies using numerical techniques to solve Maxwell's equations instead of obtaining analytical solutions.

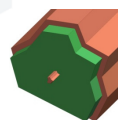


Why is this needed?

Very often, exact analytical solutions, or even good approximate solutions, are not available. Using a numerical technique offers the ability to solve virtually any electromagnetic problem of interest.



$$Z_c = \frac{\sqrt{\mu/\epsilon}}{2\pi} \cosh^{-1} \left(\frac{r_{\text{out}}}{r_{\text{in}}} \right)$$



$$Z_c = ?$$

Slide 4

4

Popular Numerical Techniques

- **Transfer matrix method**
- Scattering matrix method
- **Finite-difference frequency-domain**
- Finite-difference time-domain
- Transmission line modeling method
- Beam propagation method
- Method of lines
- **Rigorous coupled-wave analysis**
- **Plane wave expansion method**
- Slice absorption method
- Finite element analysis
- Method of moments
- Boundary element method
- Spectral domain method
- Discontinuous Galerkin method

5

CEM Wisdom

6

The Key to Computation is Visualization

Is there anything wrong? If so, what is it?

$$\frac{E_1^{(1,2)} - E_1^{(2,1)}}{\Delta y'} = \frac{E_1^{(1,1)} - E_1^{(2,2)}}{\Delta z'} = \mu_0^{(1,2)} \tilde{H}_1^{(1,2)}$$

$$+ \frac{\mu_0^{(1,2)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(1,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,2)} \tilde{H}_1^{(1,2)}}{4}$$

$$+ \frac{\mu_0^{(1,2)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(1,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,2)} \tilde{H}_1^{(1,2)}}{4}$$

$$\frac{E_1^{(1,2)} - E_1^{(2,1)}}{\Delta z'} = \frac{E_1^{(1,1)} - E_1^{(2,2)}}{\Delta y'} = \frac{\mu_0^{(1,2)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(1,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,2)} \tilde{H}_1^{(1,2)}}{4}$$

$$+ \frac{\mu_0^{(1,2)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(1,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,2)} \tilde{H}_1^{(1,2)}}{4}$$

$$\frac{E_1^{(1,2)} - E_1^{(2,1)}}{\Delta y'} = \frac{E_1^{(1,1)} - E_1^{(2,2)}}{\Delta y'} = \frac{\mu_0^{(1,2)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(1,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,2)} \tilde{H}_1^{(1,2)}}{4}$$

$$+ \frac{\mu_0^{(1,2)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(1,1)} \tilde{H}_1^{(1,2)} + \mu_0^{(2,2)} \tilde{H}_1^{(1,2)}}{4}$$

$$\frac{\tilde{H}_1^{(1,2)} - \tilde{H}_1^{(2,1)}}{\Delta y'} = \frac{\tilde{H}_1^{(1,1)} - \tilde{H}_1^{(2,2)}}{\Delta z'} = \epsilon_0^{(1,2)} E_1^{(1,2)}$$

$$+ \frac{\epsilon_0^{(1,2)} E_1^{(1,2)} + \epsilon_0^{(2,1)} E_1^{(1,2)} + \epsilon_0^{(1,1)} E_1^{(1,2)} + \epsilon_0^{(2,2)} E_1^{(1,2)}}{4}$$

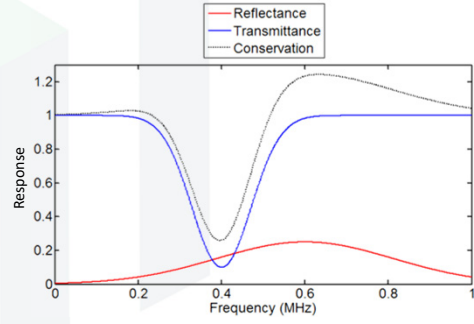
$$+ \frac{\epsilon_0^{(1,2)} E_1^{(1,2)} + \epsilon_0^{(2,1)} E_1^{(1,2)} + \epsilon_0^{(1,1)} E_1^{(1,2)} + \epsilon_0^{(2,2)} E_1^{(1,2)}}{4}$$

$$\frac{\tilde{H}_1^{(1,2)} - \tilde{H}_1^{(2,1)}}{\Delta z'} = \frac{\tilde{H}_1^{(1,1)} - \tilde{H}_1^{(2,2)}}{\Delta y'} = \frac{\epsilon_0^{(1,2)} E_1^{(1,2)} + \epsilon_0^{(2,1)} E_1^{(1,2)} + \epsilon_0^{(1,1)} E_1^{(1,2)} + \epsilon_0^{(2,2)} E_1^{(1,2)}}{4}$$

$$+ \frac{\epsilon_0^{(1,2)} E_1^{(1,2)} + \epsilon_0^{(2,1)} E_1^{(1,2)} + \epsilon_0^{(1,1)} E_1^{(1,2)} + \epsilon_0^{(2,2)} E_1^{(1,2)}}{4}$$

$$\frac{\tilde{H}_1^{(1,2)} - \tilde{H}_1^{(2,1)}}{\Delta y'} = \frac{\tilde{H}_1^{(1,1)} - \tilde{H}_1^{(2,2)}}{\Delta y'} = \frac{\epsilon_0^{(1,2)} E_1^{(1,2)} + \epsilon_0^{(2,1)} E_1^{(1,2)} + \epsilon_0^{(1,1)} E_1^{(1,2)} + \epsilon_0^{(2,2)} E_1^{(1,2)}}{4}$$

$$+ \frac{\epsilon_0^{(1,2)} E_1^{(1,2)} + \epsilon_0^{(2,1)} E_1^{(1,2)} + \epsilon_0^{(1,1)} E_1^{(1,2)} + \epsilon_0^{(2,2)} E_1^{(1,2)}}{4}$$



7

Golden Rule #1

All numbers should equal 1.

Why?

(1.234567...) + (0.0123456...) = Lost two digits of accuracy!!

Solution: NORMALIZE EVERYTHING!!!

$$\lambda'_0 = \frac{\lambda_0}{1 \mu\text{m}}$$

$$\tilde{\tilde{E}} = \sqrt{\frac{\epsilon_0}{\mu_0}} \tilde{E}$$

or

$$\tilde{\tilde{H}} = \sqrt{\frac{\mu_0}{\epsilon_0}} \tilde{H}$$

$$x' = k_0 x$$

$$y' = k_0 y$$

$$z' = k_0 z$$

8

Golden Rule #2

Never perform calculations.**Why?**

1. Golden Rule #1.
2. Finite floating point precision introduces round-off errors.

Solution: MINIMIZE NUMBER OF COMPUTATIONS!!!

1. Take problems as far analytically as possible.
2. Avoid unnecessary computations.

$$\begin{array}{ccc}
 \cancel{r = \sqrt{x^2 + y^2}} & \rightarrow & R = x^2 + y^2 \\
 \cancel{g(r) = \exp\left(-\frac{r^2}{\sigma^2}\right)} & & g(R) = \exp\left(-\frac{R}{\sigma^2}\right)
 \end{array}$$

Golden Rule #3

Write clean code.

- Well organized
- Well commented
- Compact
- No junk code

Why?

1. It will run faster and more reliably.
2. Easier to catch mistakes.
3. Easier to troubleshoot.
4. Easier to pick up again at a later date.
5. Easier to modify.

Solution

1. Outline your code before writing it.
2. Delete obsolete code.
3. Comment every step.
4. Use meaningful variable names.

The CEM Process



There is a rhythm to computational electromagnetics and it repeats itself constantly.

Starts with Maxwell's equations and derives all the necessary equations to implement the algorithm in MATLAB.

- Equations everywhere! Only a few are needed.
- Implementation does not resemble the formulation.

Organizes the equations derived in the formulation and implements the equations in a computer code.

- Consider all numerical best practices.
- Should end with a detailed block diagram and a computer program.

Tests the code and practices the user by matching results of known devices and from the literature.

- Start simple and work toward more complex devices.
- Always simulate the closest known device.

Practice makes perfect!

CEM

Formulation

Implementation

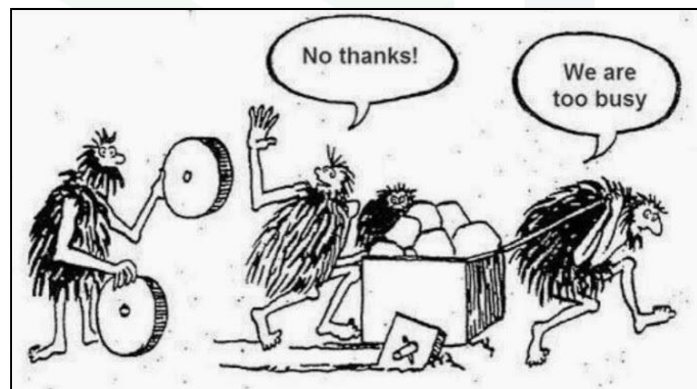
Benchmark

Practice

11

Don't Be Lazy

A little extra time making your program more efficient or simulating a device in a more intelligent manner can save you lots of time, energy, and aggravation in the long term.



12

Formulation Wisdom

- Derive equations as far and as simple as possible.
- Build big/complicated matrices from small/simple matrices.
 - This usually requires converting to matrix form early in the formulation.
- Make your formulation documents very detailed.
- Keep your formulation consistent with your code.
- A good understanding of the formulation gives you the ability to modify your algorithm or to add/subtract features.

Implementation Wisdom

- **Make a detailed block diagram!**
- In the block diagram, include only the equations you will incorporate into your code.
- Add all other steps to your block diagram.
 - Sources
 - Building devices
 - Extracting information
 - Post-processing data
 - Etc.

Coding Wisdom

- Work hard to write clean, simple, and well commented code.
 - Indent code inside loops, if statements, etc.
 - Let linear algebra do the work for you.
 - Use lots of comments.
- Match your code to your formulation.
 - Try to use the same variables.
- Do not fix your code with incorrect equations.
 - Changing signs arbitrarily is a common way to make things work, but you are hiding a problem and possibly creating more.
- Do not ever hard-code any numbers.

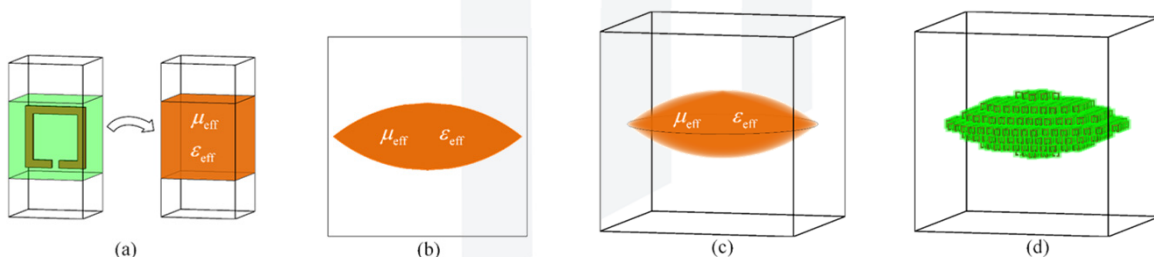
Simulation Wisdom #1

Simulate devices in multiple steps using models of increasing complexity.

Avoid the temptation to jump straight to the big, bad, and ugly 3D simulation in all of its glorious complexity.

Model your device with slowly increasing levels of complexity.

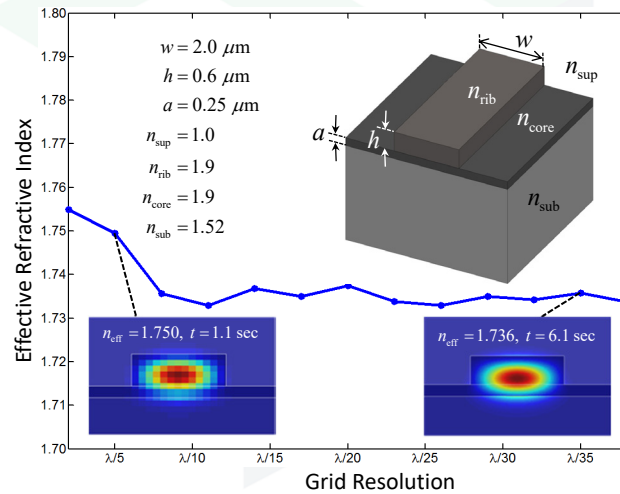
You will get to your final answer much faster and much easier this way!



R. C. Rumpf, "Engineering the dispersion and anisotropy in periodic electromagnetic structures," Solid State Physics 66, 2015.

Simulation Wisdom #2

It must be standard practice to ensure your results are converged.



17

Simulation Wisdom #3



Those who simulate the most,
trust the simulations the least.

*Never trust your code or your results.
Benchmark. Benchmark. Benchmark.*

18

Final Word of Wisdom

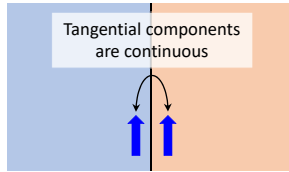
Do not EVER share your codes!
Only bad things will happen.

The best thing that can happen
is that you become useless.

Instead, offer to simulate devices for
people and make yourself a collaborator.

General Concepts in Computational EM

Physical Vs. Numerical Boundary Conditions

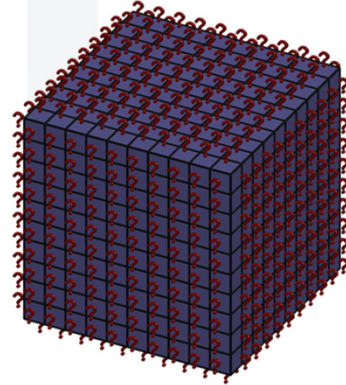


Physical Boundary Conditions

Physical boundary conditions refer to the conditions that must be satisfied at the boundary between two materials. These are derived from the integral form of Maxwell's equations.

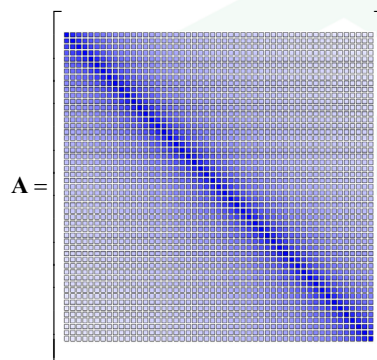
Numerical Boundary Conditions

Numerical boundary conditions refer to the what is done at the edge of a grid or mesh and how fields outside the grid are estimated.



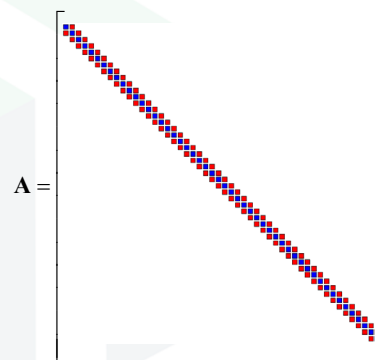
Full Vs. Sparse Matrices

Full Matrices



Full matrices have all non-zero elements.
They tend to look banded with the largest numbers running down the main diagonal.

Sparse Matrices



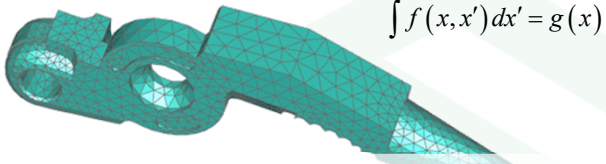
Sparse matrices have most of their elements equal to zero. They are often more than 99% sparse.

It is most memory efficient to store only the non-zero elements in memory.

They tend to "banded" matrices with the largest numbers running down the main diagonal.

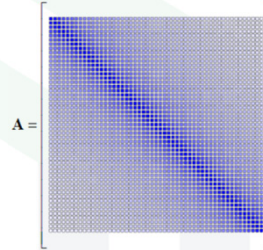
Integral Vs. Differential Equations (1 of 2)

Integral Equations



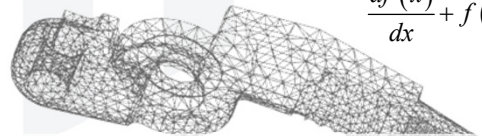
$$\int f(x, x') dx' = g(x)$$

Integral equations calculate a quantity at a specific point using information from the entire domain. They are usually written around boundaries and lead to formulations with full matrices. They do not require boundary conditions.

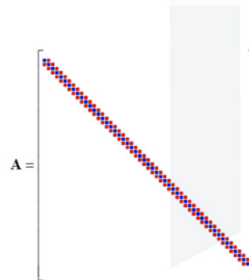


Differential Equations

$$\frac{df(x)}{dx} + f(x) = g(x)$$



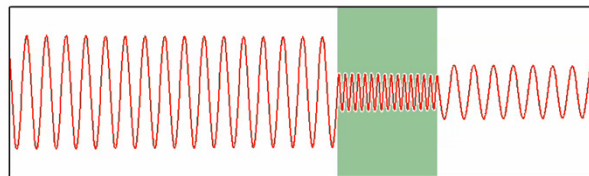
Differential equations calculate a quantity at a specific point using only information from the local vicinity. They are usually written for points distributed throughout a volume and lead to formulations with sparse matrices. They require boundary conditions.



Frequency-Domain Vs. Time-Domain

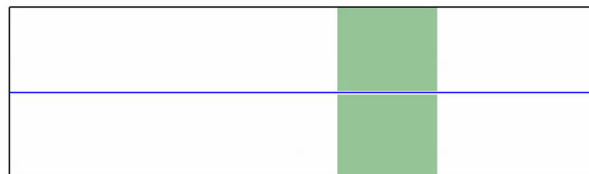
STEADY-STATE RESPONSE

This is what a frequency-domain code calculates.



TRANSIENT RESPONSE

This is what a time-domain code calculates.



Frequency-domain solutions are at a single frequency. Time-domain solutions look different because there are inherently a broad range of frequencies involved.

Definition of “Convergence”

Virtually all numerical methods have some sort of “resolution” parameter that when taken to infinity solves Maxwell’s equations exactly. In practice, we cannot do this arbitrarily far because a computer will run out of memory and simulations will take prohibitively long to run.

There are no equations to calculate what “resolution” is needed to obtain “accurate” results. Instead, the user must look for convergence. There are, however, some good rules of thumb to make an initial guess at resolution.

Convergence is the tendency of a calculated parameter to asymptotically approach some fixed value as the resolution of the model is increased. **A converged solution does not imply an accurate solution!!!**

Tips About Convergence

- Make checking for convergence a habit that you always perform.
- When checking a parameter for convergence, ensure that it is the only thing about the simulation that is changing.
- Simulations do not get more “accurate” as resolution is increased. They only get more “converged.”

Conservation of Power

When electromagnetic wave is applied to a device, it can be absorbed (i.e. converted to another form of energy), reflected and/or transmitted. Without a nuclear reaction, nothing else can happen.

$$A + R + T = 1$$

Reflectance, R

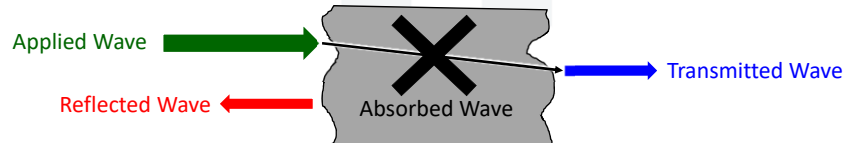
Fraction of power from the applied wave that is reflected from the device.

Transmittance, T

Fraction of power from the applied wave that is transmitted through the device.

Absorptance, A

Fraction of power from the applied wave that is absorbed by the device.



27

How Do You Know if Your Model Works?

In many cases, you may not know. ☹

1. BENCHMARK, BENCHMARK, BENCHMARK
2. Check for power conservation
3. CONVERGENCE, CONVERGENCE, CONVERGENCE

Common Sense – Check your model for simple things like conservation of power, magnitude of the numbers, consistency with the physics, etc.

Benchmark – You can verify your code is working by modeling a device with a known response. Does your model predict that response?

Convergence – Your models will have certain parameters that you can adjust to improve “accuracy” usually at the cost of computer memory and run time. Keep increasing “accuracy” until your answer does not change much any more.

When modeling a new device, benchmark your model using as similar of a device as you can find which has a known response. Compare your experimental results to the model. Do they agree? Reconcile any differences.

28

Tips for Matching Simulations to Experiments

1. Model the geometry of the device as accurately as possible.
2. Incorporate accurate material properties, including dispersion.
3. Ensure the source replicates the experimental source as closely as possible.
4. Ensure you detect power in your simulation in a way that is consistent with laboratory experiments.

Classification of Methods

Classification by Size Scale

Low Frequency Methods

$$\lambda_0 \sim a$$

Structural dimensions are on the order of the wavelength or smaller.

Polarization and the vector nature of the field is important.

- Finite-difference time-domain
- Finite-difference frequency-domain
- Finite element analysis
- Method of moments
- Rigorous coupled-wave analysis
- Method of lines
- Beam propagation method
- Boundary element method
- Spectral domain method
- Plane wave expansion method

High Frequency Methods

$$\lambda_0 \ll a$$

Structural dimensions much larger than the wavelength.

Fields can be accurately treated as scalar quantities.

- Ray tracing
- Geometric theory of diffraction
- Physical optics
- Physical theory of diffraction
- Shooting and bouncing rays



Classification by Approximations

Rigorous Methods

A method is **rigorous** if there exists a “resolution” parameter that when taken to infinity, finds an exact solution to Maxwell’s equations.

- Finite-difference time-domain
- Finite-difference frequency-domain
- Finite element method
- Rigorous coupled-wave analysis
- Method of lines

Full Wave Methods

A method is **full wave** if it accounts for the vector nature of the electromagnetic field. A full wave method is not necessarily rigorous.

- Method of moments
- Boundary element method
- Beam propagation method

Scalar Methods

A method is **scalar** if the vector nature of the field is not accounted for.

- Ray tracing

Comparison of Method Types

Frequency-Domain + resolves sharp resonances + handles oblique incidence + longitudinal periodicity + can be very fast - scales at best $M \log N$ - can miss sharp resonances - active & nonlinear devices	Time-Domain + wideband simulations + scales near linearly + active & nonlinear devices + easily locates resonances - longitudinal periodicity - sharp resonances - memory requirements - oblique incidence
Fully Numerical + better convergence + scales better than SA + complex device geometry - memory requirements - long uniform sections	Semi-Analytical + very fast & efficient + layered devices + less memory - convergence issues - scales poorly - complex device geometry
Real-Space + high index contrast + metals + resolving fine details + field visualization - slow for low index contrast	Fourier-Space + moderate index contrast + periodic problems + very fast and efficient - field visualization - formulation difficult - resolving fine details
Structured Grid + easy to implement + rectangular structures + easy for divergence free - less efficient - curved surfaces	Unstructured Grid + most efficient + handles larger structures + conforms to curved surfaces - difficult to implement - spurious solutions
Differential Based + sparse matrices + easier to formulate + easier to implement - volume mesh - spurious solutions	Integral Based + surface mesh + Very efficient for many structures - full matrices - more difficult to formulate - more difficult to implement

33

Multiphysics Simulations

A multiphysics simulation is one that accounts for multiple simultaneous physical mechanisms at the same time.

- Electromagnetic
- Thermal
- Fluids
- Motion
- Chemical
- Acoustic
- Optical

34

Any Method Can Do Anything

Any method can be made to do anything.

The real questions are:

- What devices and information is a particular method best suited for?
- How much of a “force fit” is it for that method?