An introduction of Scientific Computing

Institute of Mathematics Modelling and Scientific Computing

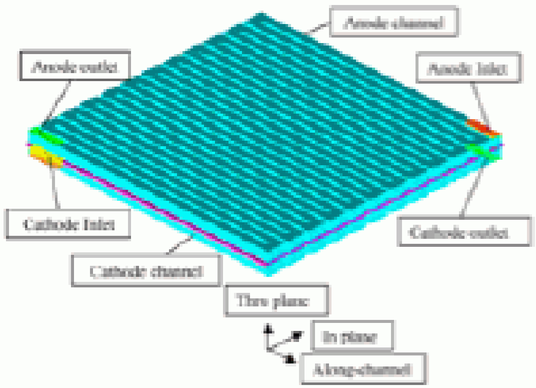
National Chiao-Tung University

Chin-Tien Wu

Why *scientific computing*?

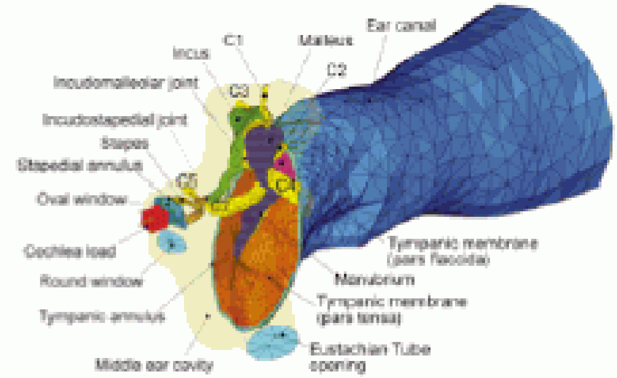
- Simulation of natural phenomena
- Virtual prototyping of engineering designs
- More insights from unknown phenomenon
- Much more

Large-scale Simulation of Polymer Electrolyte Fuel Cells by parallel Computing
(Hua Meng and Chao-Yang Wang, 2004)

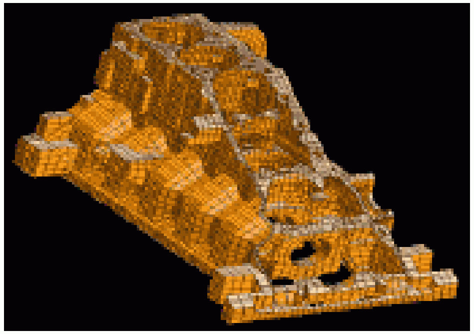


FEM model with O(10^6) nodes

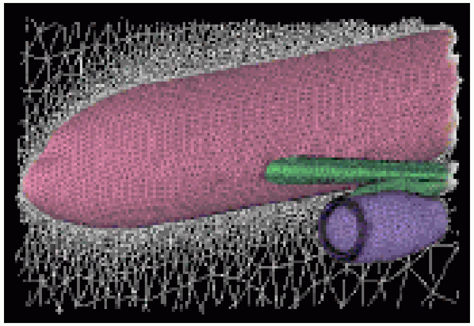
Three-Dimensional Finite Element Modeling of Human Ear for Sound Transmission
(R. Z. Gan, B. Feng and Q. Sun, 2004)



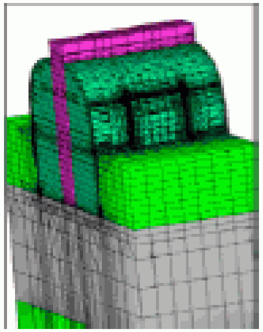
FEM model with $10^5 \sim 10^6$ nodes



Car engine with O(10^5) nodes



Commercial Aircraft: 10^7 nodes



FINFET transistor: 10^5 nodes

Figure 1 Composite Space Reflector

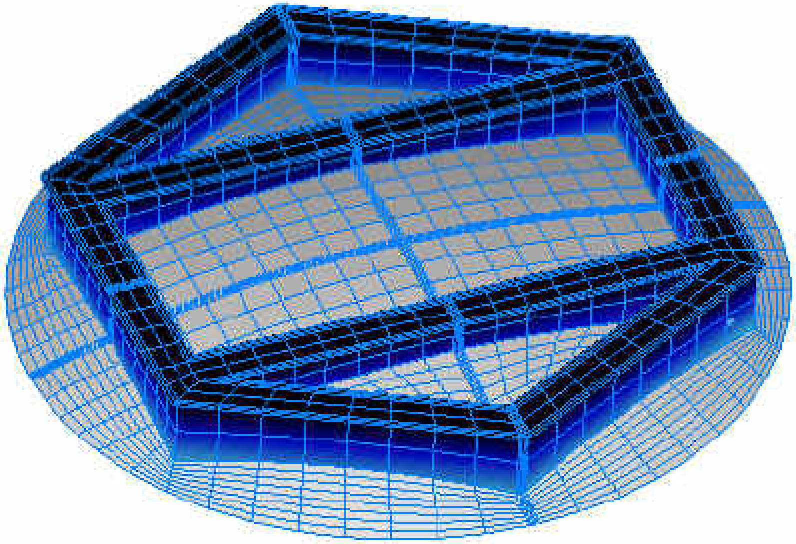


Figure 3 Finite Element Model for the Reflector

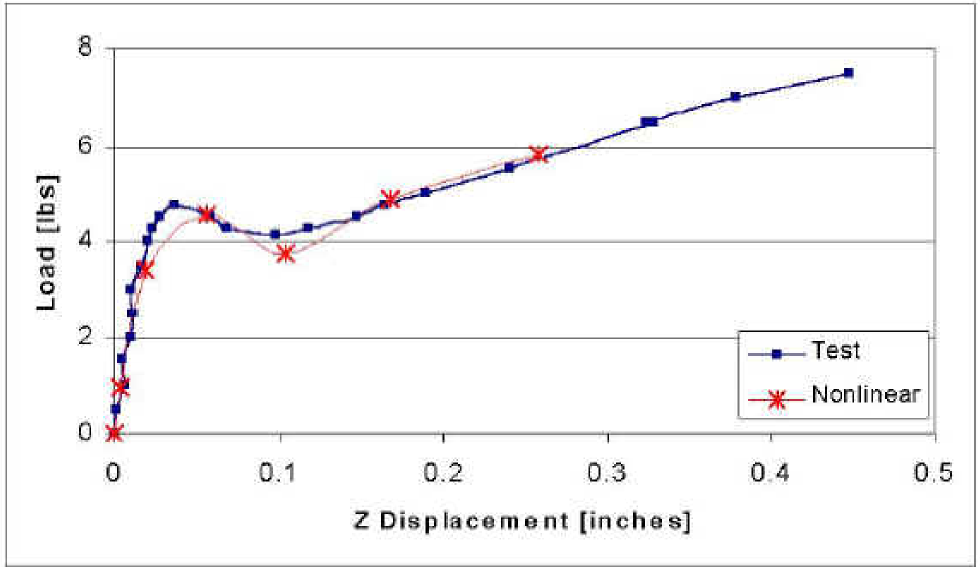


Figure 21 Load vs. Displacement at the Load Point

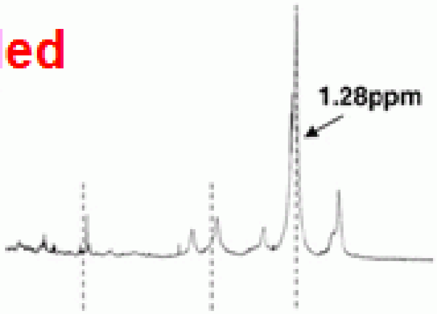
Mathematics makes magnetic resonance better

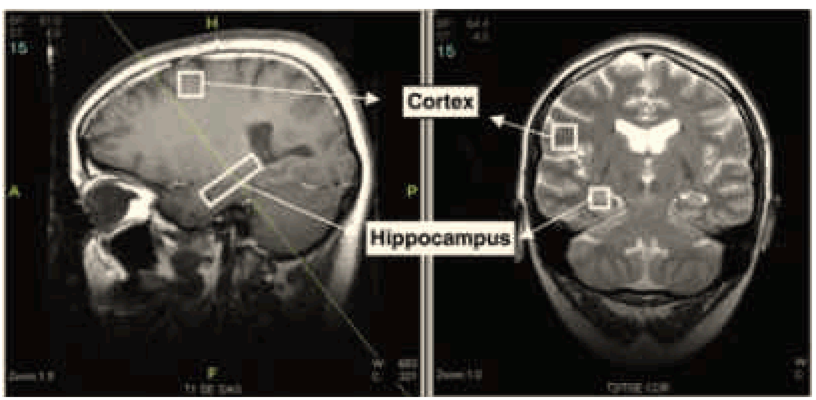
Science **318**, 980 (2007)

Magnetic Resonance Spectroscopy Identifies Neural Progenitor Cells in the Live Human Brain

Louis N. Manganas,[1,3] Xueying Zhang,[1] Yao Li,[1] Raphael D. Hazel,[1,2] S. David Smith,[2] Mark E. Wagshul,[1] Fritz Henn,[2] Helene Benveniste,[1,2] Petar M. Djurić,[1] Grigori Enikolopov,[3*] Mirjana Maletić-Savatić[1,3*]

In the paper, a mathematical method called the singular value decomposition (SVD) is used to decompose the signals so as to identify neural stem cells





What is *scientific computing*?

Design and analysis of algorithms for numerically solving mathematical problems in science and engineering. Traditionally called *numerical analysis*

Distinguishing features of *scientific* computing

Approximate *continuous* quantities by discrete quantities. Considers effects of approximations including error and sensitivity

What should be cared in scientific computing?

- Problem is *well-posed* if solution
  - exists
  - is unique
  - depends continuously on problem data

  Otherwise, problem is *ill-posed*

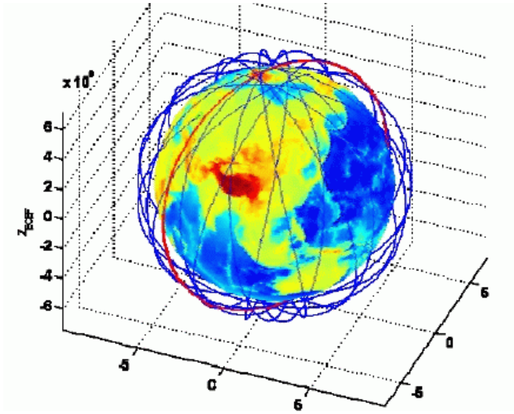
- Even if problem is well posed, solution may still be *sensitive* to input data

- Computational algorithm should not make sensitivity worse

Sources of approximation

- Before computation
  - modeling
  - empirical measurements
  - previous computations
- During computation
  - truncation or discretization
  - rounding
- Uncertainty in input may be amplified by problem
- Perturbations during computation may be amplified by algorithm

Example 1



- Computing surface area of Earth using formula $A = 4\pi r^2$ involves several approximations
  - Earth is modeled as sphere, idealizing its true shape
  - Value for radius is based on empirical measurements and previous computations
  - Value for π requires truncating infinite process
  - Values for input data and results of arithmetic operations are rounded in computer

Error measurement

- *Absolute error*: approximate value − true value
- *Relative error*: $\dfrac{\text{absolute error}}{\text{true value}}$
- Equivalently, approx value $=$ (true value) $\times$ (1 $+$ rel error)
- True value usually unknown, so we *estimate* or *bound* error rather than compute it exactly
- Relative error often taken relative to approximate value, rather than (unknown) true value

Where numerical errors come from?

- *Truncation error* : difference between true result (for actual input) and result produced by given algorithm using exact arithmetic
  - Due to approximations such as truncating infinite series or terminating iterative sequence before convergence
- *Rounding error* : difference between result produced by given algorithm using exact arithmetic and result produced by same algorithm using limited precision arithmetic
  - Due to inexact representation of real numbers and arithmetic operations upon them

More about error (I)

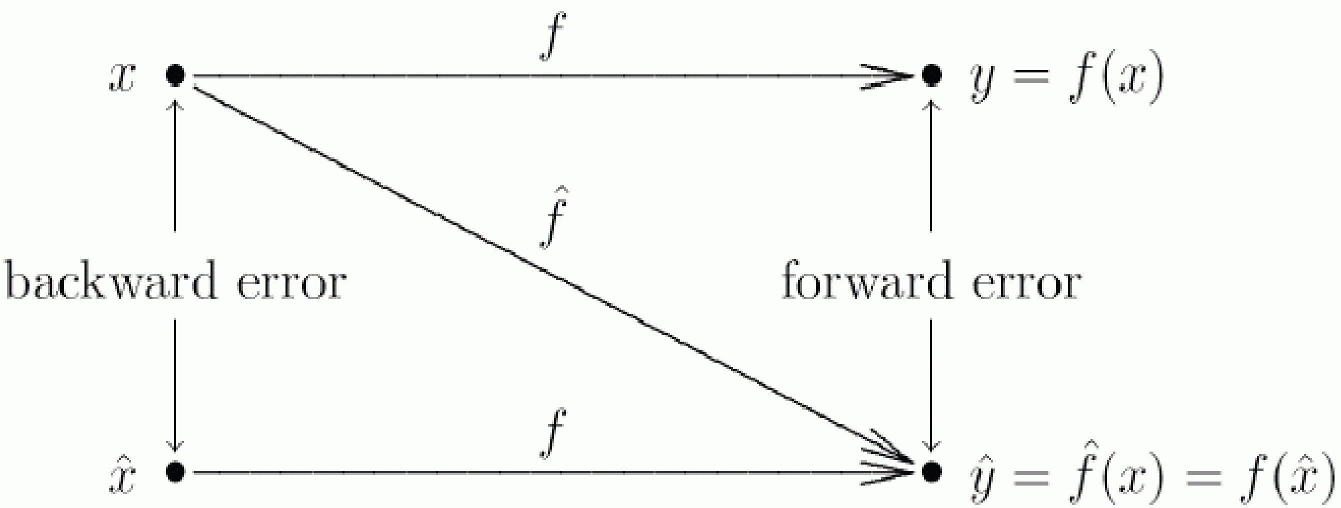
- Typical problem: compute value of function $f : \mathbb{R} \to \mathbb{R}$ for given argument
  - $x =$ true value of input
  - $f(x) =$ desired result
  - $\hat{x} =$ approximate (inexact) input
  - $\hat{f} =$ approximate function actually computed
- Total error: $\hat{f}(\hat{x}) - f(x) =$

$$\hat{f}(\hat{x}) - f(\hat{x}) \quad + \quad f(\hat{x}) - f(x)$$

computational error + propagated data error

- Algorithm has no effect on propagated data error

More about the error (II)

- Suppose we want to compute $y = f(x)$, where $f: \mathbb{R} \to \mathbb{R}$, but obtain approximate value $\hat{y}$
- *Forward error*: $\Delta y = \hat{y} - y$
- *Backward error*: $\Delta x = \hat{x} - x$, where $f(\hat{x}) = \hat{y}$
- 

Condition number of a function f at variable value x is defined as:

$$cond_x(f) = \sup_{\hat{x}} \frac{\left| f(\hat{x}) - f(x) \right| / \left| f(x) \right|}{\left| \hat{x} - x \right| / \left| x \right|}$$

Relative propogated error (rounding error) can be estimated by:

$$\left| f(\hat{x}) - f(x) \right| / \left| f(x) \right| \leq cond_x(f) \cdot \underbrace{\left| \hat{x} - x \right| / \left| x \right|}_{\text{relative input error}}$$

Relative computational error (truncation error) can be estimated by:

$$\left| \hat{f}(\hat{x}) - f(\hat{x}) \right| / \left| f(x) \right| = \left| \hat{f}(\hat{x}) - \hat{f}(\tilde{x}) \right| / \left| f(x) \right|$$

$$\leq cond_x(\hat{f}) \cdot \frac{\left| \hat{f}(\hat{x}) \right|}{\left| f(x) \right|} \cdot \underbrace{\left| \tilde{x} - \hat{x} \right| / \left| \hat{x} \right|}_{\text{relative backward error}}$$

Evaluate the forward error, backward error and the condition number:

Evaluate function f for approximate input $\hat{x} = x + \Delta x$ instead of the true input x, we have

Absolute forward error = $\left| f\left(x+\Delta x\right) - f\left(x\right) \right| \approx \left| f'\left(x\right) \right| \left| \Delta x \right|$

Relative forward error = $\dfrac{\left| f\left(x+\Delta x\right) - f\left(x\right) \right|}{\left| f\left(x\right) \right|} \approx \left| \dfrac{f'\left(x\right)}{f\left(x\right)} \right| \left| \Delta x \right|$

Condition number $= \sup\limits_{\Delta x} \dfrac{\left| f\left(x+\Delta x\right) - f\left(x\right) \right| / \left| f\left(x\right) \right|}{\left| \Delta x \right| / \left| x \right|} \approx \left| \dfrac{f'\left(x\right)}{f\left(x\right)} \right| \left| x \right|$

In what condition the backward error can be controlled?

Consider: $\left\| \hat{f}(x) - f(x) \right\| = \left\| f(\tilde{x}) - f(x) \right\|$

$$= \left\| f'(x)(\tilde{x} - x) + \frac{1}{2} f''(\bar{x})(\tilde{x} - x)^2 \right\|$$

$$\Rightarrow \left\| \hat{f}(x) - f(x) \right\| \approx \left\| f(x) \right\| \left\| cond(f) \right\| \cdot \frac{\left\| \tilde{x} - x \right\|}{\left\| x \right\|} \text{ when } \tilde{x} \sim x.$$

Clearly, when $\begin{cases} cond(f) < \infty \\ \left\| f(x) \right\| < \infty \end{cases}$, $\hat{f} \to f \Leftrightarrow \dfrac{\left\| \tilde{x} - x \right\|}{\left\| x \right\|} \to 0$

Moreover, the backward error can be estimated by

$$\text{Relative backward} \approx \frac{\text{Relative forward error}}{\text{Condition number}}$$

A problem f is said to be well-posed if or well-conditioned if

$$cond_x(f) << \infty \text{ for all } x$$

An algorithm $\hat{f}$ is said to be well-posed if or well-conditioned if

$$cond_x(\hat{f}) << \infty \text{ for all } x$$

⇨ Relative error in the solution (output) is insensitive to the relative small change in the input.

A problem f or an algorithm $\hat{f}$ is said to be stable if the relative backward error is small

⇨ Accurate numerical solution can be obtained only when a problem and computational algorithm are well-conditioned and stable

example2

- Approximating cosine function $f(x) = \cos(x)$ by truncating Taylor series after two terms gives

$$\hat{y} = \hat{f}(x) = 1 - x^2/2$$

- Forward error is given by

$$\Delta y = \hat{y} - y = \hat{f}(x) - f(x) = 1 - x^2/2 - \cos(x)$$

- To determine backward error, need value $\hat{x}$ such that $f(\hat{x}) = \hat{f}(x)$
- For cosine function, $\hat{x} = \arccos(\hat{f}(x)) = \arccos(\hat{y})$

- For $x = 1$,

$$\begin{aligned} y &= f(1) = \cos(1) \approx 0.5403 \\ \hat{y} &= \hat{f}(1) = 1 - 1^2/2 = 0.5 \\ \hat{x} &= \arccos(\hat{y}) = \arccos(0.5) \approx 1.0472 \end{aligned}$$

- Forward error: $\Delta y = \hat{y} - y \approx 0.5 - 0.5403 = -0.0403$
- Backward error: $\Delta x = \hat{x} - x \approx 1.0472 - 1 = 0.0472$

example3

- Tangent function is sensitive for arguments near $\pi/2$
  - $\tan(1.57079) \approx 1.58058 \times 10^5$
  - $\tan(1.57078) \approx 6.12490 \times 10^4$
- Relative change in output is quarter million times greater than relative change in input
  - For $x = 1.57079$, cond $\approx 2.48275 \times 10^5$

The reason is as following:

$$cond_{\frac{\pi}{2}-\varepsilon}(\tan) \approx \left|\frac{\pi}{2}-\varepsilon\right|\left|\sec^2\left(\frac{\pi}{2}-\varepsilon\right)\right| \Big/ \left|\tan\left(\frac{\pi}{2}-\varepsilon\right)\right| \approx \left|\frac{\pi}{2}-\varepsilon\right| \Big/ \varepsilon$$

Example 4

Consider approximating $f(x) = e^{-x^2}$ by $\hat{f}(x) = 1 - x^2$

For $x = \sqrt{1-\varepsilon}$, we have $\hat{f}(x) = \varepsilon = f(\hat{x})$, this implies $\hat{x} = \sqrt{-\ln(\varepsilon)}$.

Clearly, the backward error at x=1 is $\lim_{\varepsilon \to 0} \frac{\left|\sqrt{1-\varepsilon} - \sqrt{-\ln(\varepsilon)}\right|}{\left|\sqrt{1-\varepsilon}\right|} = \infty$

The approximation $\hat{f}$ to f is unstable near x=1.

Exercise: Estimate the backward error by the formula given at p.11.
Could you explain why the estimation is nothing close to the real error?