Physics-Based Modeling: Principles, Methods and Examples

Wesley N. Colley
Leslie A. Litten
Center for Modeling, Simulation and Analysis
University of Alabama in Huntsville

Purpose of Tutorial

- Motivate need for good physics in M&S
- Motivate idea that good physics can often be carried out efficiently
  - Familiarize audience with numerical techniques that radically enhance computational efficiency
  - Present physics-based examples that benefit from such techniques
  - Present an example where difficult physics has a simple mathematical solution

Quick Acknowledgements

- Much of this material can be found in *Numerical Recipes*
  - in C++, C and FORTRAN
  - Press, Teukolsky, Vetterling, Flannery
- Charts and much analysis prepared for this talk carried out in IDL
  - Interactive Data Language
  - see www.itt-vis.com
  - otherwise C++

Outline

- Intro to physics in M&S
- Quadrature (Integration of Functions)
- Integration of Differential Equations
  - orbits and trajectories
- Radiative Processes
  - atmospheric effects on visibility
- Fourier methods
  - image processing

How does physics play a role in M&S?

- Physics and M&S share a similar goal
  - Model the world around us
- Physics started when
  - Computers didn't exist
  - Questions were simple, like "why do arrows fly?"
- M&S and Physics meet when
  - Modeler: Accurate models of natural behavior are needed in my simulation
  - Physicist: Computers are necessary to handle the math in my physics problem

Strengths of Physics

- Physics (at some level) describes everything in the Universe
  - Sub-atomic interactions
    - Binding of quarks in proton
  - Cosmological scale interactions
    - Expansion and acceleration of the Universe
  - Everything in between
    - Atoms, molecules, baseballs, mountains, planets, stars, galaxies

What about a human thought?

Okay, smarty, no. The electrons in the neurons, though…

Macroscopic Stuff

- Basic mechanics
  - Flight of baseballs, pendula, springs, orbits
- Thermo-/Hydro-dynamics
  - Airframe modeling, mixing of airborne agents, dam engineering, rockets, explosives, heat pump
- Materials
  - Heat resistance, tensile strength, conductive properties, lightness
- Electricity and Magnetism
  - Optics, radar, compasses, electrical engineering
- Quantum Physics
  - Lasers, microchips, nuclear

The Weakness of Physics

- Physics tends to break down when very large numbers of physical entities are involved
  - Cannot compute bridge properties through quantum interactions (~10^{35} atoms in a bridge!)
- *Chemistry, Chem E*: rule sets approximating quantum mechanics
- *Biology, Materials Science*: rule sets approximating Chemistry and quantum
- *Astrophysics*: rule set approximating gravity, hydro and quantum
- *Engineering*: (often) use of physical properties of materials, gases, etc. for large systems

So physics is (often) useful…

- How do we model it?
- MATH
- Physics is very often a means of mapping reality into mathematics
  - Almost all macroscopic interactions are governed by a second-order partial differential equation
- Just math? Then is knowledge of Nature's apparent rules "deep?" Does $\mathbf{F} = m\mathbf{a}$
  - Tell us something fundamental about Nature
  - Or just provide a synopsis of our observations?

Math...

- The math physics generates is typically complicated
  - Very few realistic problems can be solved analyitically
- Answer: Computer
  - Use numerical mathematics

$$-\frac{\hbar^2}{2m}\nabla^2\psi(\mathbf{r})-\frac{ke^2}{r}\psi(\mathbf{r})=i\hbar\frac{\partial\psi(\mathbf{r})}{\partial t}$$

This equation governs a single electron in a Hydrogen atom!

The strategy

- When faced with a problem, identify the type of physics at its root
- Make approximations that simplify the problem
  - Air resistance is negligible on a falling coin
    - Not true from Empire State building
  - Moon is a point mass
    - Not true if concerned about tides on moon

The Strategy (cont.)

- Once you are working at the right level, begin looking at the physics involved
- Identify the mathematical issues the physics presents
- Choose the correct numerical methods for handling that math
- Model away!

Outline

- Intro to physics in M&S
- Quadrature (Integration of Functions)
- Integration of Differential Equations
  - orbits and trajectories
- Radiative Processes
  - atmospheric effects on visibility
- Fourier methods
  - image processing

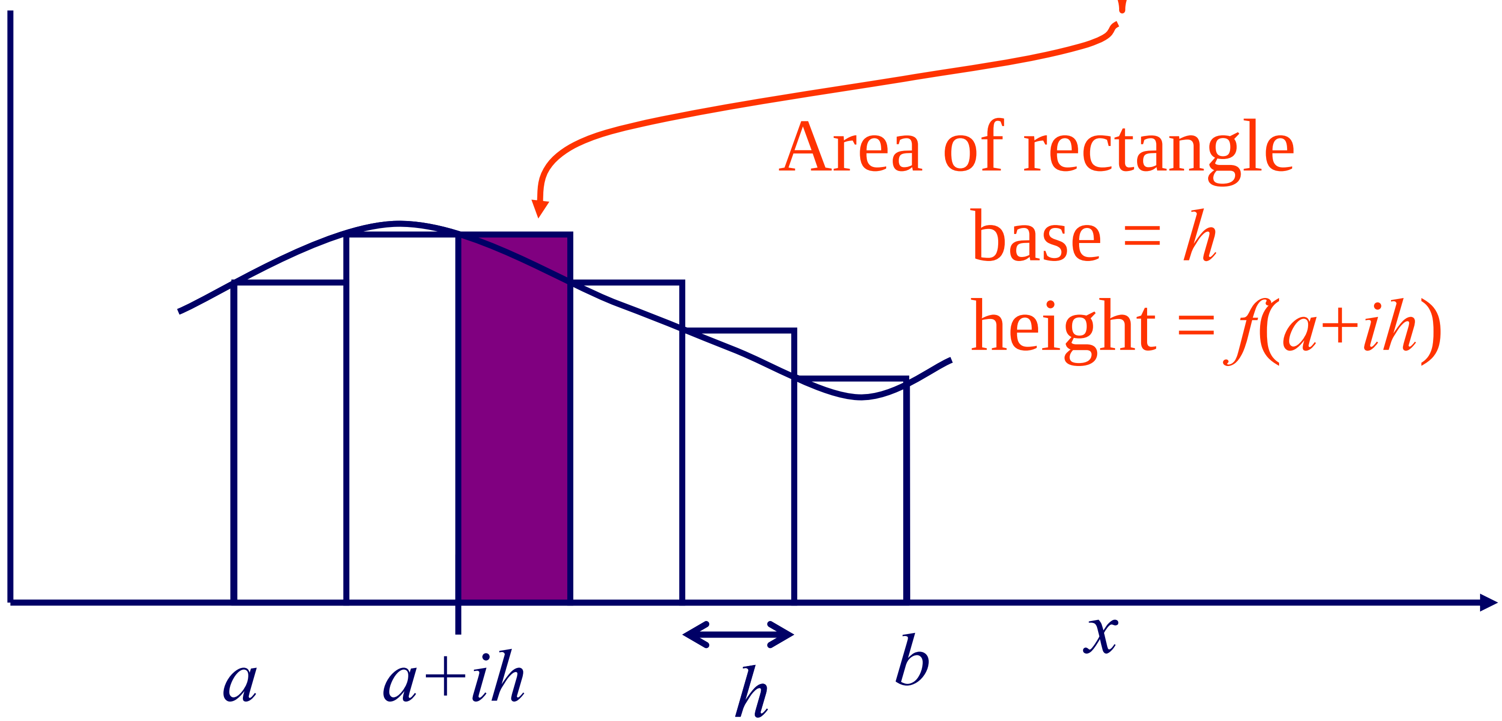
Quadrature Segue

- First things first
- Introduce a powerful mathematical technique that can be generalized into physics applications

- Simple math question:
  - How do I find the area under the function $f(x)$?

Riemann Sum: Simple Question
What is Area Under Curve $f(x)$?

$$F(a,b) = \int_a^b f(x)dx = \lim_{h \to 0} \sum_{i=0}^{(a-b)/h-1} \underbrace{f(a+ih)h}$$



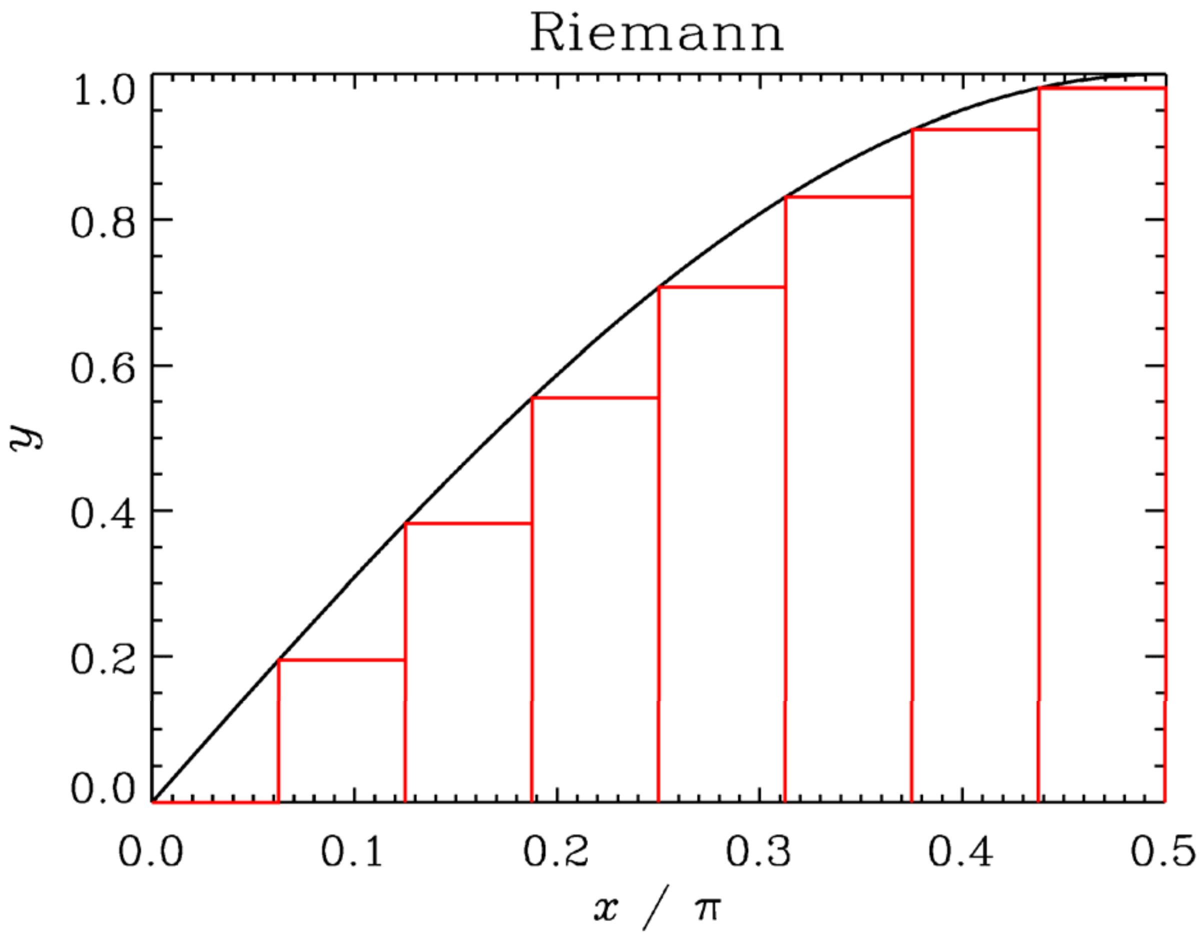
Riemann Sum Example

- $y = \sin(x)$
- $a = 0$
- $b = \pi/2$
- $n = 8$

$$\int_0^{\pi/2} \sin(x)\,dx = 1$$

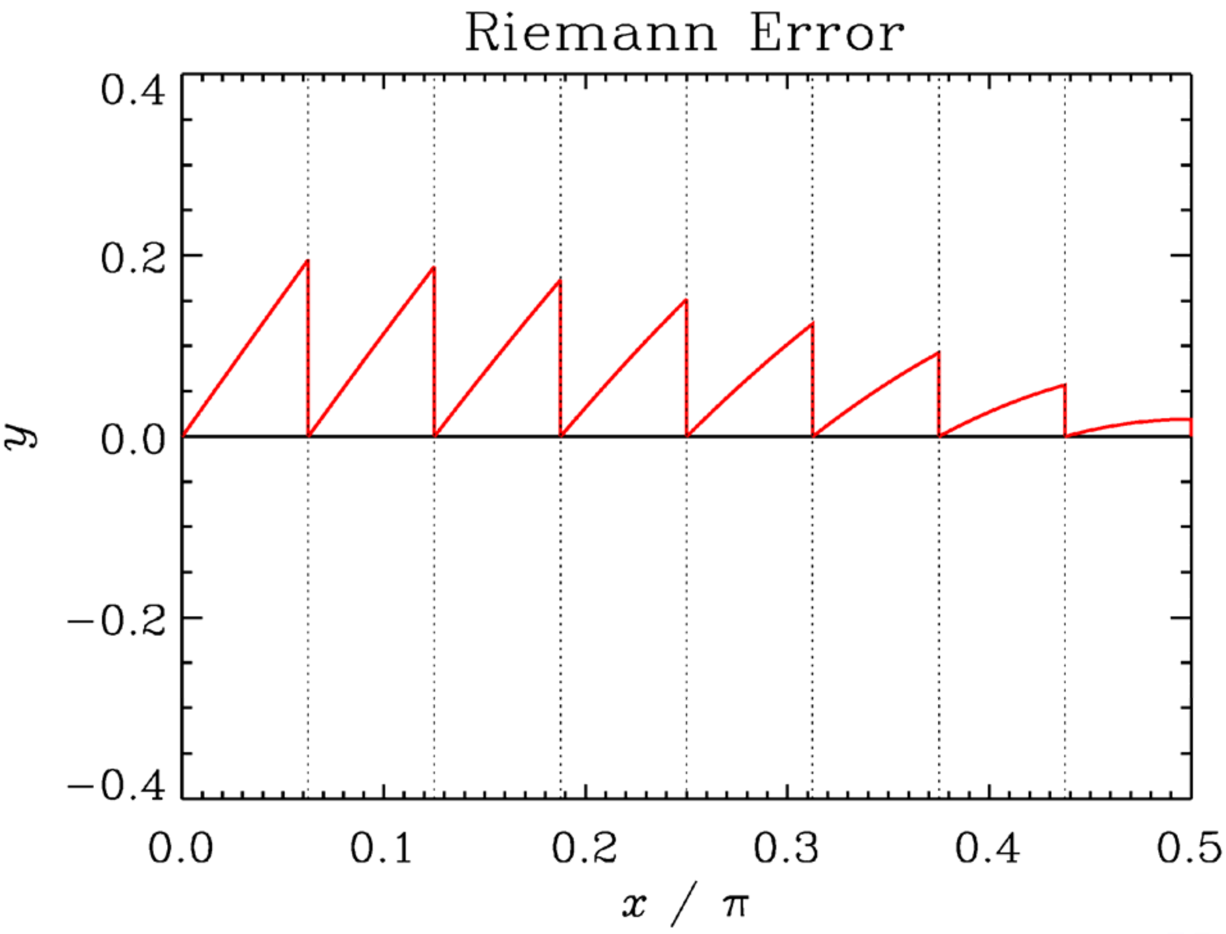
- Estimate: 0.89861040

so-so result

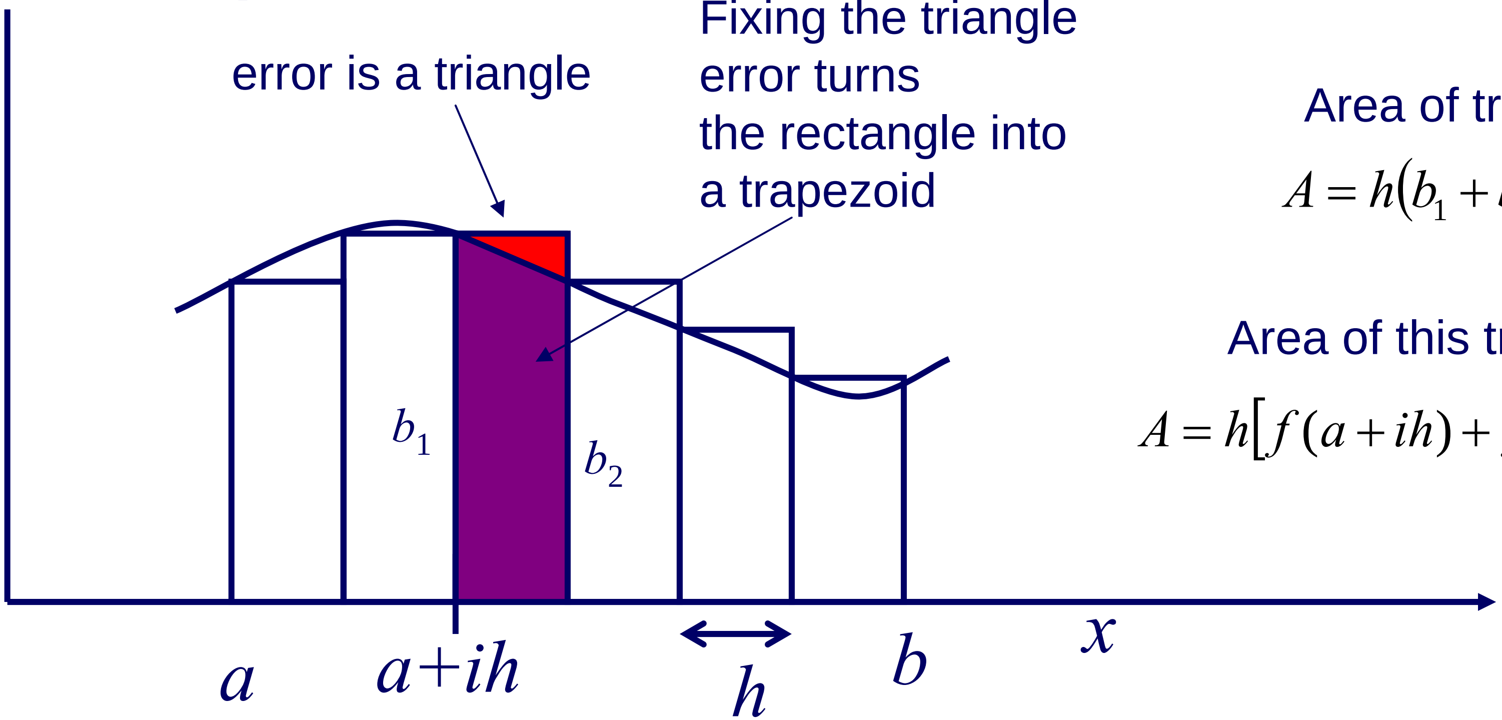


Improving

- Examine errors made
- Basically look like triangles
- Can we correct for that?



Trapezoid Rule



Area of trapezoid:

$$A = h(b_1 + b_2)/2$$

Area of this trapezoid:

$$A = h[f(a+ih) + f(a+ih+h)]/2$$

$$F(a,b) \approx \frac{h}{2} \sum_{i=0}^{(a-b)/h-1} f(a+ih) + f(a+ih+h)$$

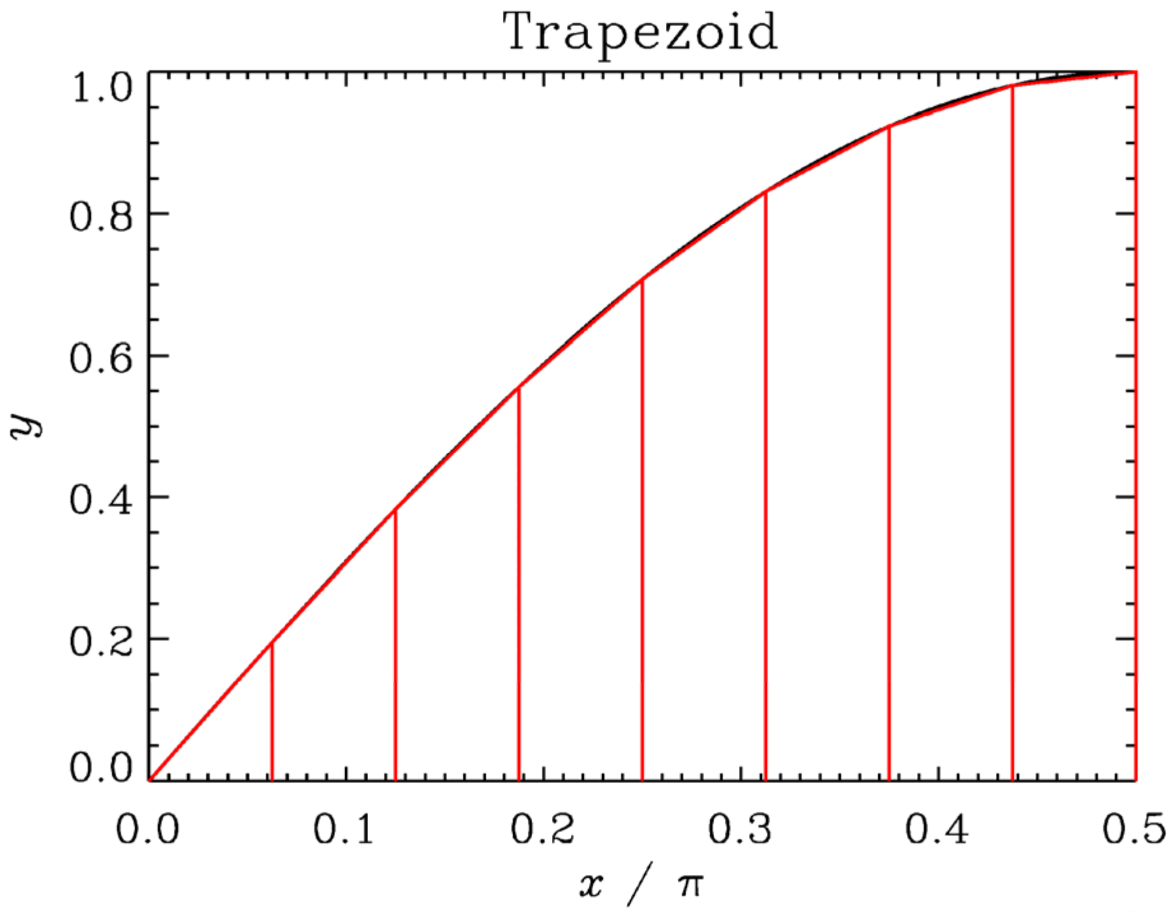
sum up trapezoids, not rectangles

$$= h \sum_{i=1}^{(a-b)/h-1} f(a+ih) + \frac{h}{2}[f(a) + f(b)]$$

simplify: same as Riemann, except endpoints… hmmm

Trapezoid Rule

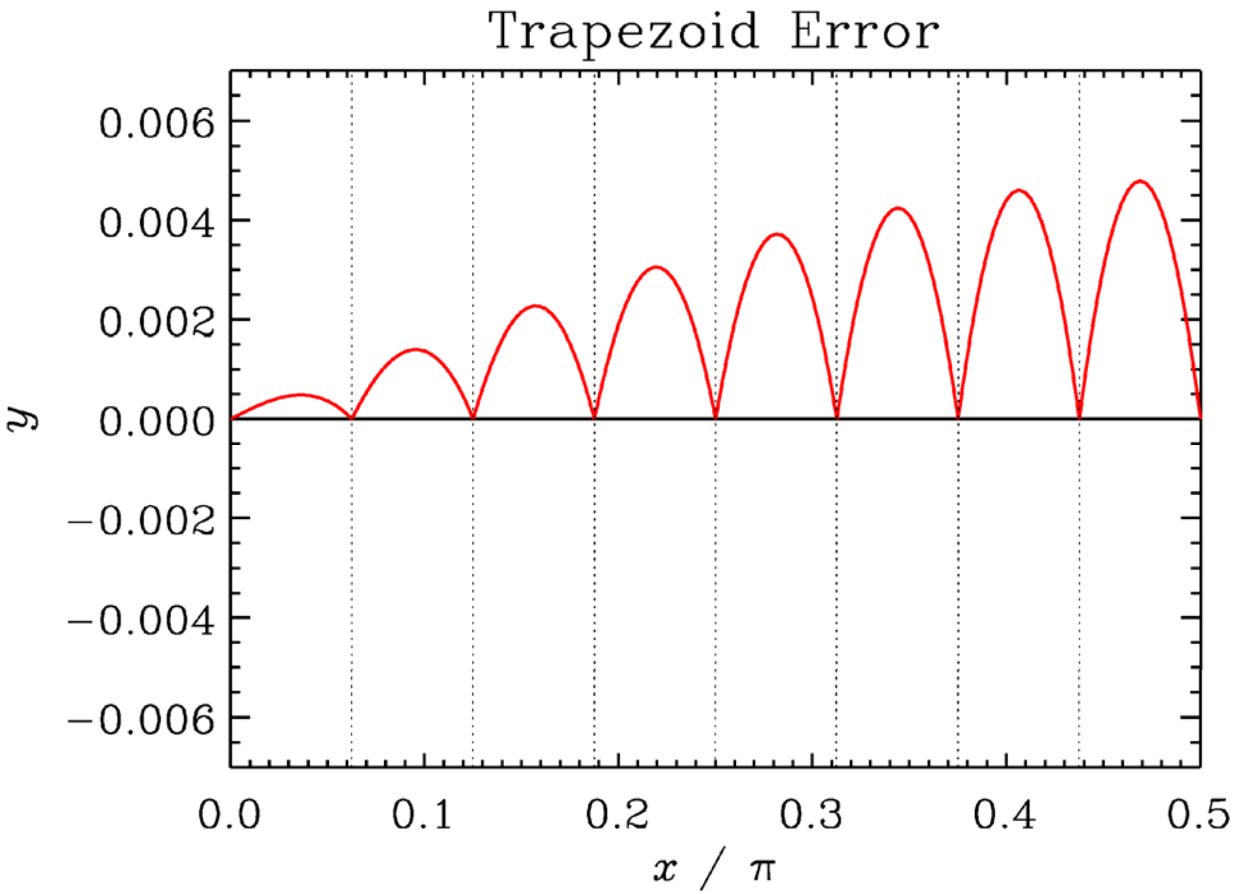
- Return to sine curve
- Much better looking
- Estimate: 0.99678517
- Much better!
- Same number of calls to derivative function!



Note: Need continous first derivative for it to work right…

Trapezoid Errors

- Now errors
- are much smaller
- They look like parabolas
- What next?



Simpson's Rule

- Fit parabolas to every three points
  - find area under each parabola
- Sounds complicated, but the area under the parabola is given by a simple linear formula
  - not quadratic as one might guess

For a parabola fitting the points

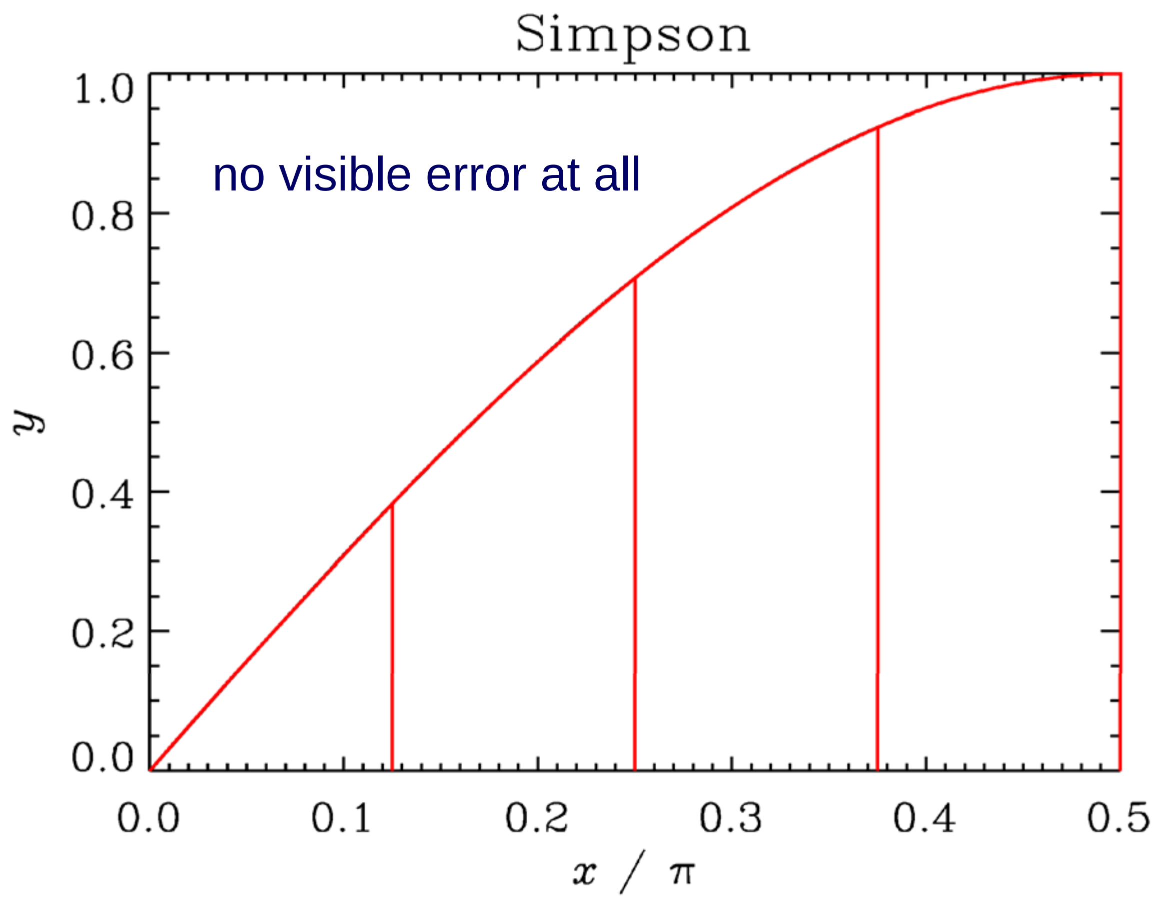
$$(x_0, y_0), (x_0 + h, y_1), (x_0 + 2h, y_2)$$

$$A = \frac{h}{3}\left(y_0 + 4y_1 + y_2\right)$$

Simply adjust weights in sum!

Simpson's Rule

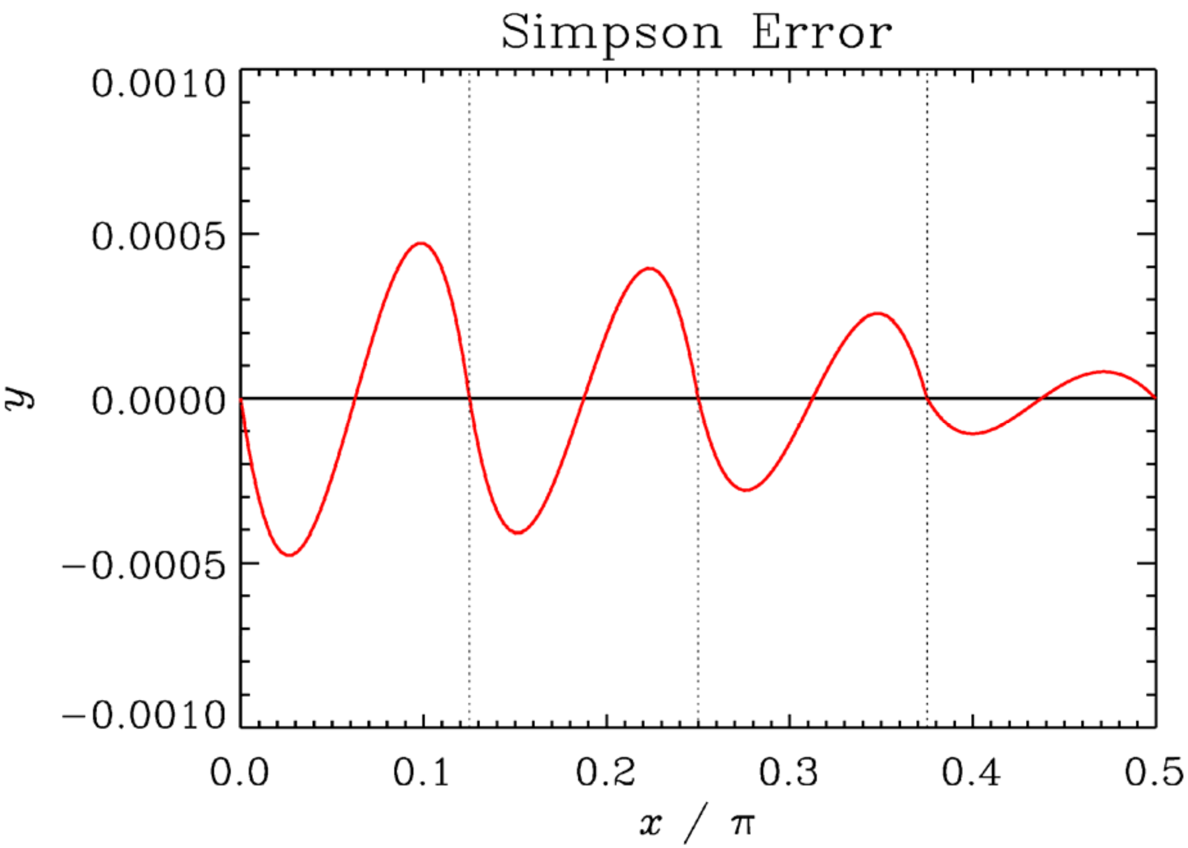
- Find area under parabolas in every interval of $2h$.
- Estimate: 1.0000083
- Very good, and still same number of calls.



Note: Need continous second derivative for it to work right…

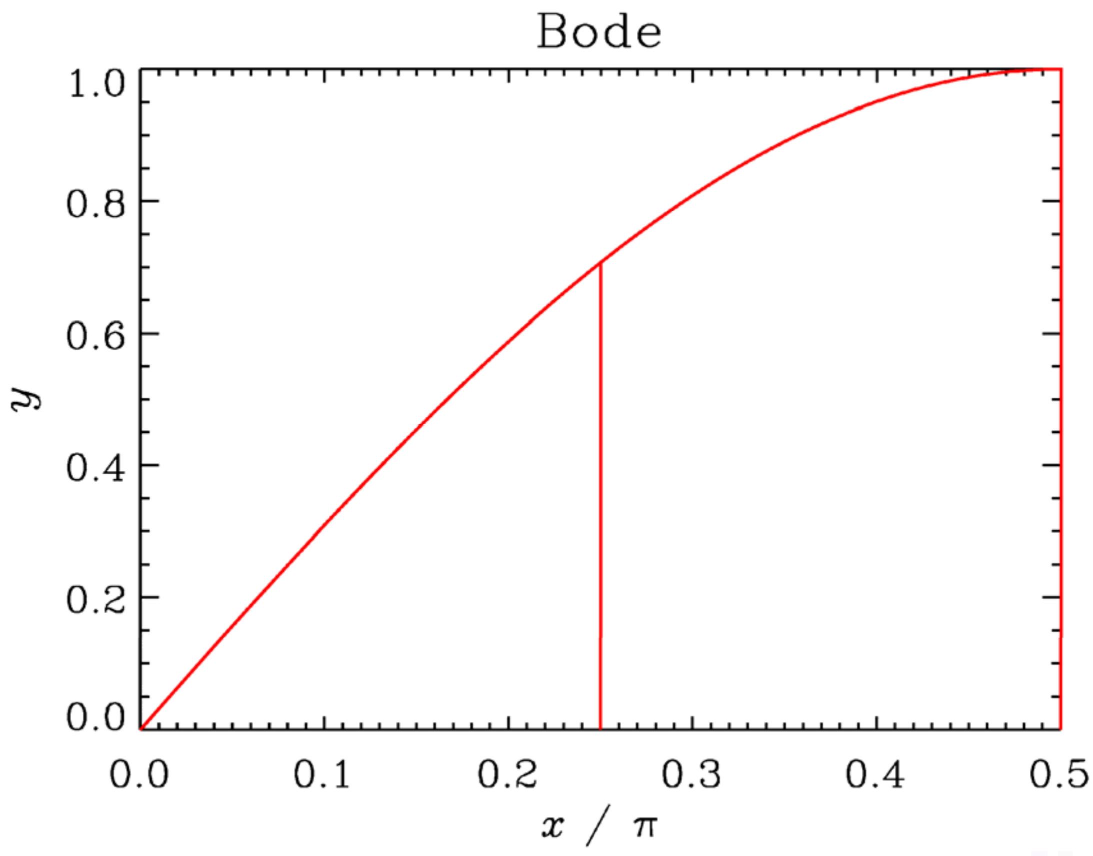
Simpson Errors

- Errors are now quite small
- Cubic in nautre
- Curiously, cubic terms cancel, leaving quartic errors



Bode's Rule

- Okay, fit quartics to each interval of $4h$
- Just different weights in sum again
- Estimate: 0.99999988
- Still better, still same number of calls

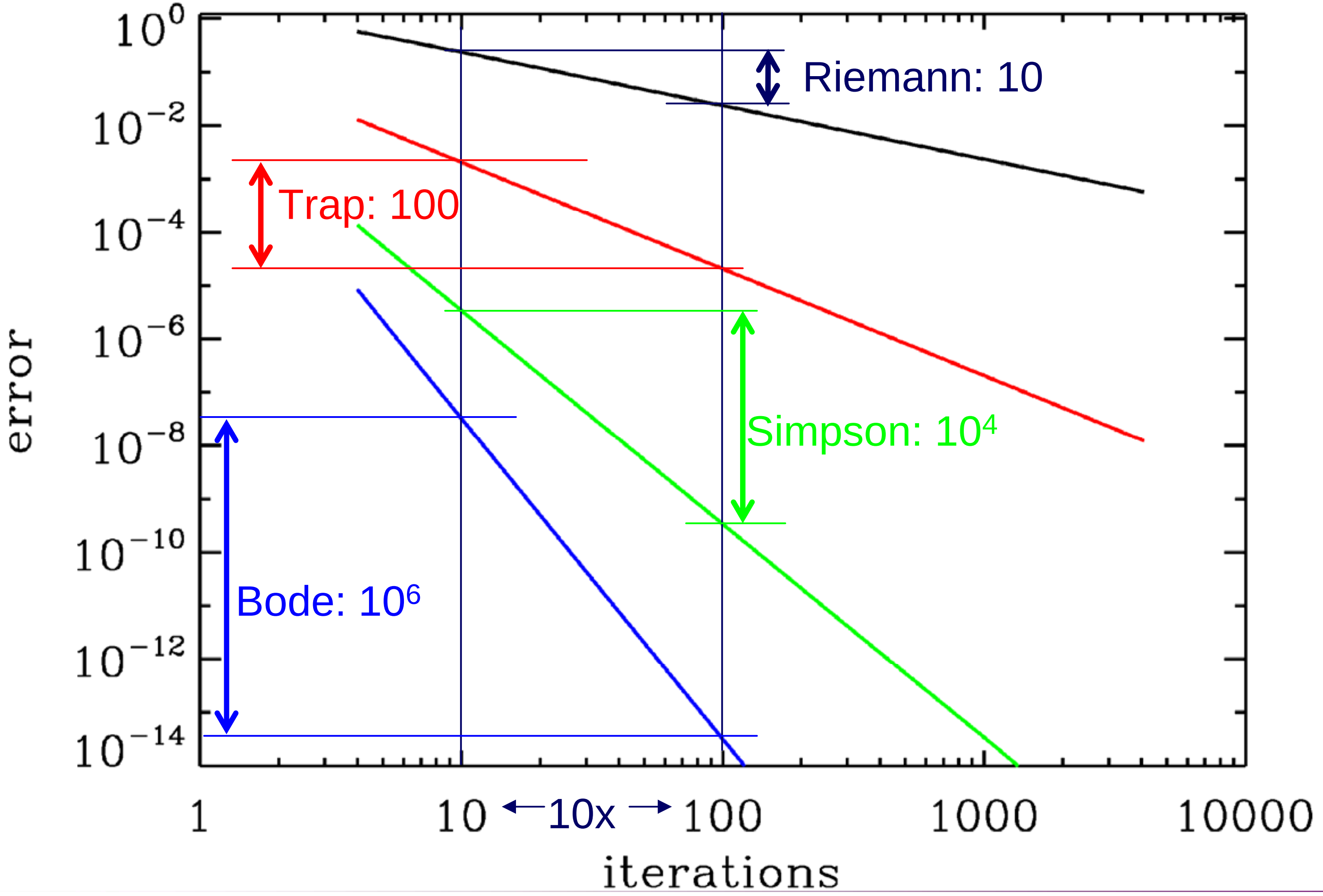


Note: Need continous **fourth** derivative for it to work right…

Convergence

- One can also improve estimate by "brute force"
  - Simply carry out more iterations
- How much do estimates improve as a function of number of iterations?

Convergence



Convergence

- Riemann sum improves linearly with increased iterations
- Trapezoid: quadratically
- Simpson’s Rule: quartically
- Bode’s Rule: 6^{th} order
  - a million times better with ten times the iterations!
- Why not keep going?

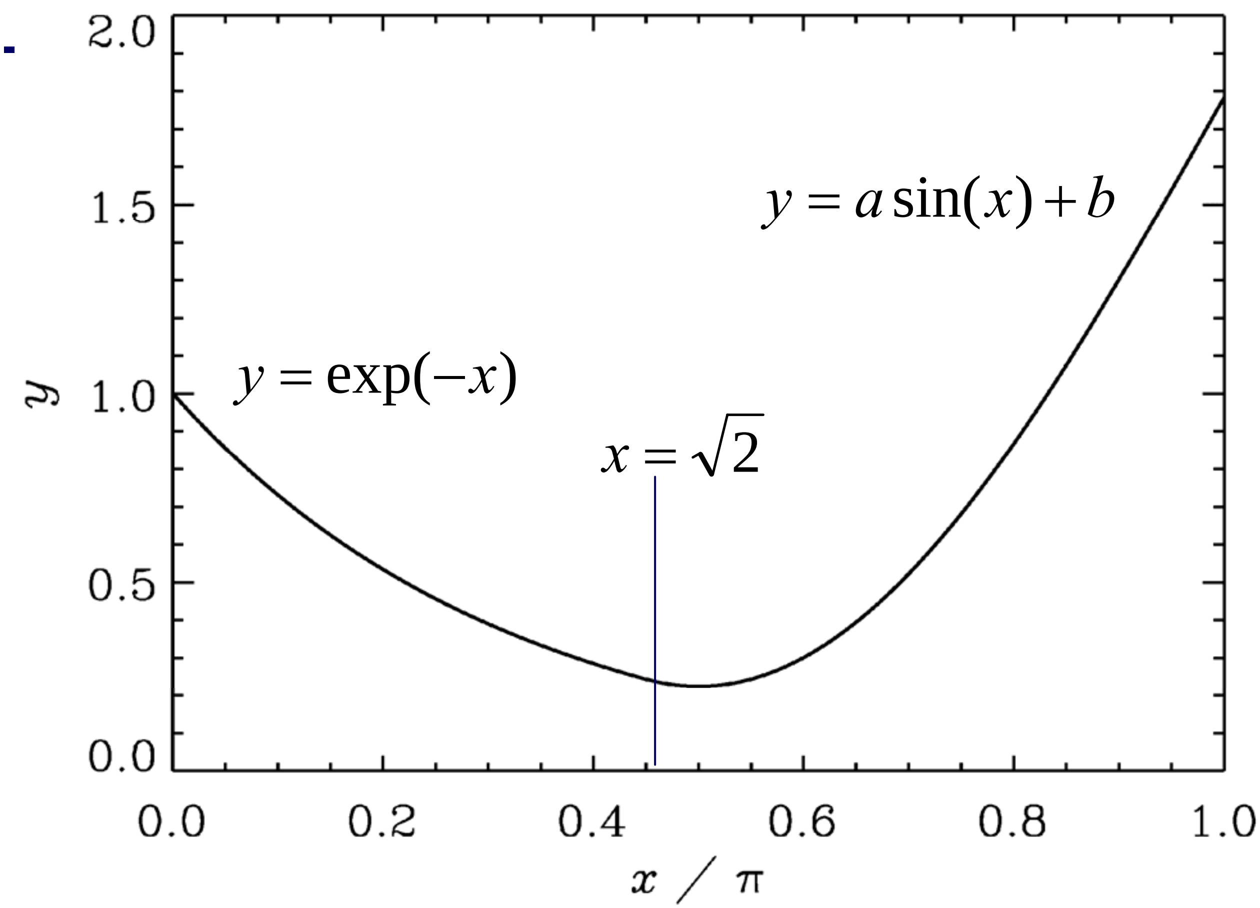
Getting silly

- One could keep fitting higher-order polynomials to improve the fit
  - and maintain computational load
- However, these high-order rules require increasingly well-behaved functions
  - Namely, functions must be continuously differentiable at the order of the polynomial
  - Not likely in real world too often
    - If it is, the integral is probably analytic or semi-analytic... Just look up the answer!

One Counter-Example

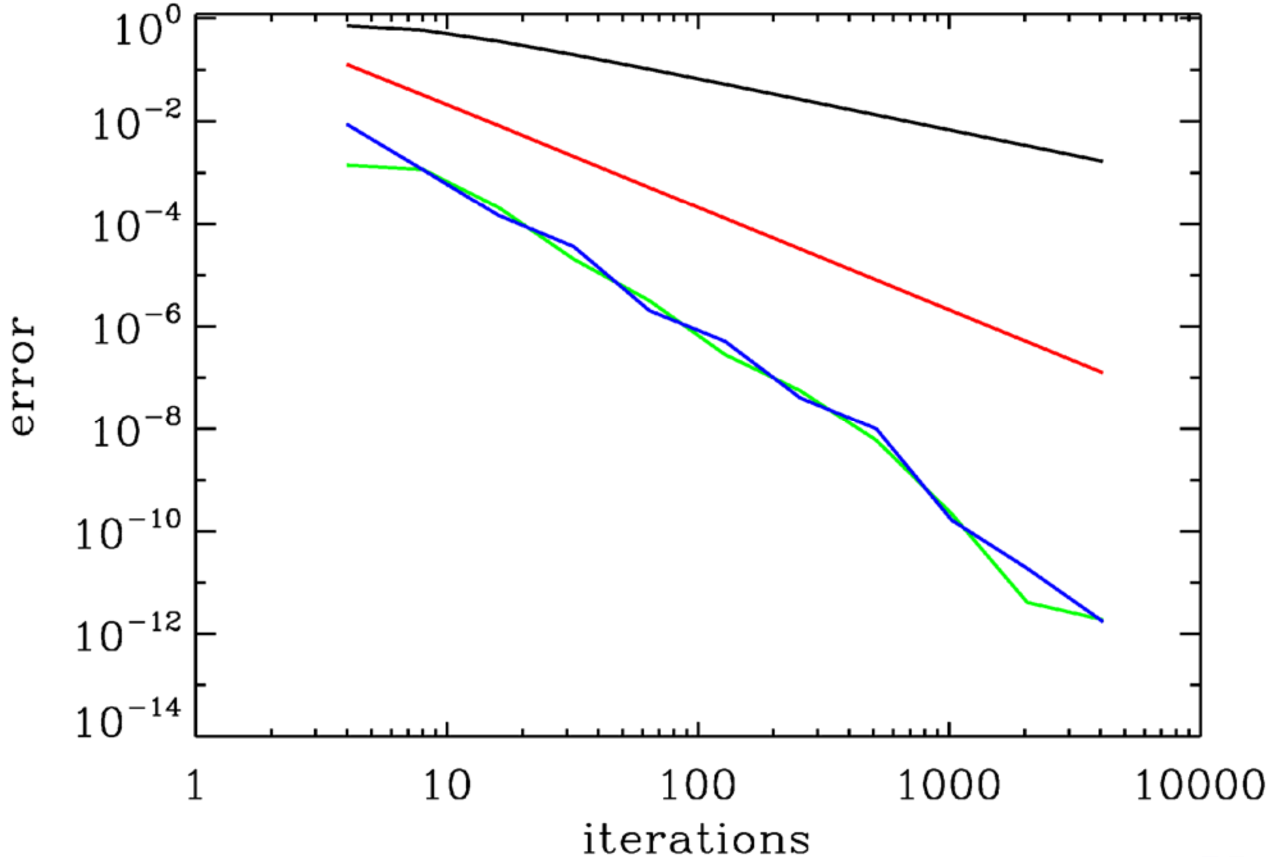
The derivative rules really do matter

- Continuous first-derivative
- discontinuous 2nd derivative
- Trapezoid and Riemann shouldn't notice
- Simpson and Bode should



Convergence for baddish function

- Riemann and trapezoid behave normally
- Simpson and Bode do not
  - improvement is essentially 2nd order, same as trapezoid



Quadrature Summary

- Several different methods use the exact same calls to the derivative function with vastly different results
  - higher order means better estimates AND
  - better convergence with more iterations
- But, beware the caveats of higher order methods
- My advice: try Simpson's Rule
- Advanced methods use extrapolation from results of different iteration numbers

Outline

- Intro to physics in M&S
- Quadrature (Integration of Functions)
- Integration of Differential Equations
  - orbits and trajectories
- Radiative Processes
  - atmospheric effects on visibility
- Fourier methods
  - image processing

Diff Eq Segue

- Similar methods to those of numerical integration carry over into ordinary differential equations
- A great many physical systems are governed by such equations
  - orbits
  - ballistics
  - analog circuits
  - springs, dampers
  - pendula

Tangential Integration

- Consider

$$\frac{dx}{dt} = f(t, x)$$

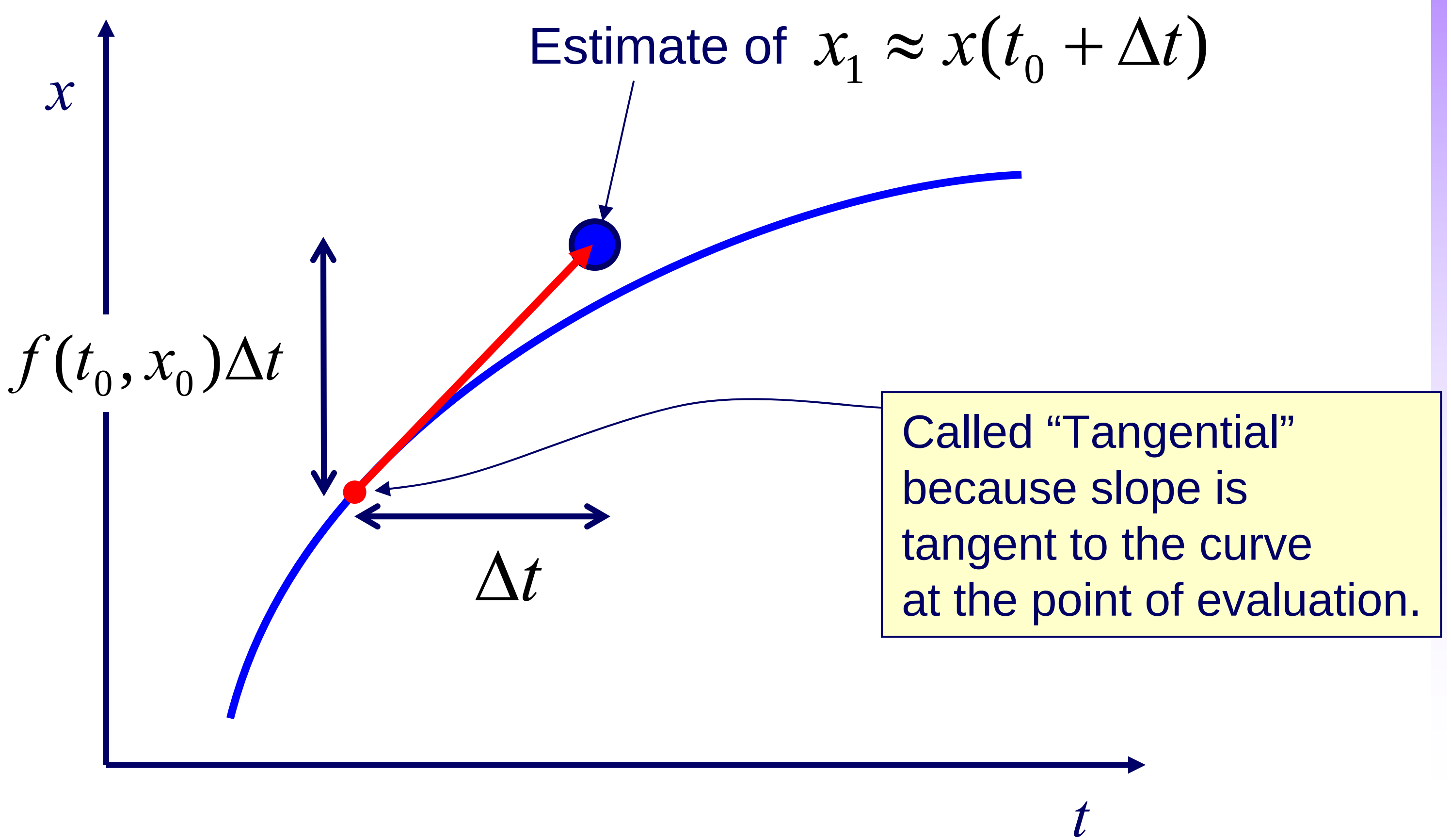
- One could integrate the solution:
  - start with an initial value
  - compute derivative
  - find next value

$$x(t_0) = x_0$$

$$x(t_0 + \Delta t) = x(t_0) + f(t_0, x_0)\Delta t$$

Graphically

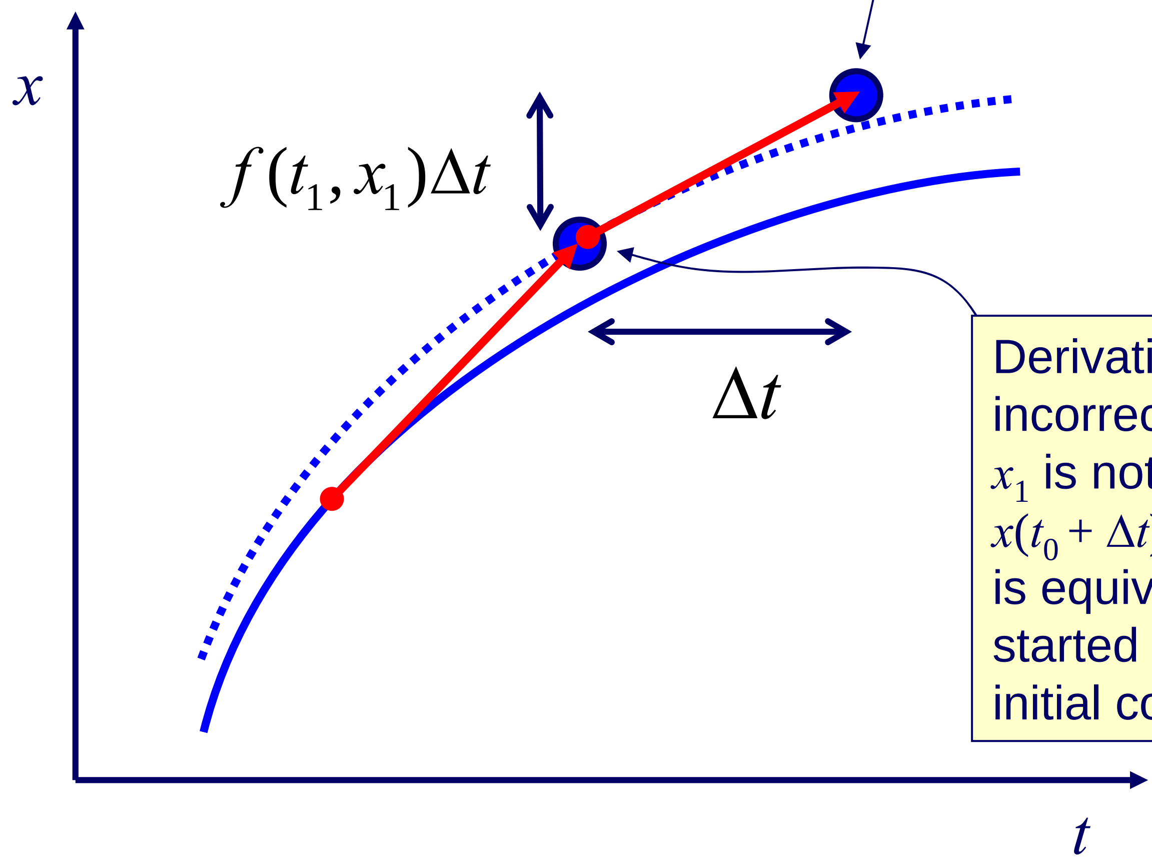
- Slope at $t_0, x_0 = f(t_0, x_0)$



Graphically

- Slope at $x_1,\ t_1 = f(t_1,\ x_1)$

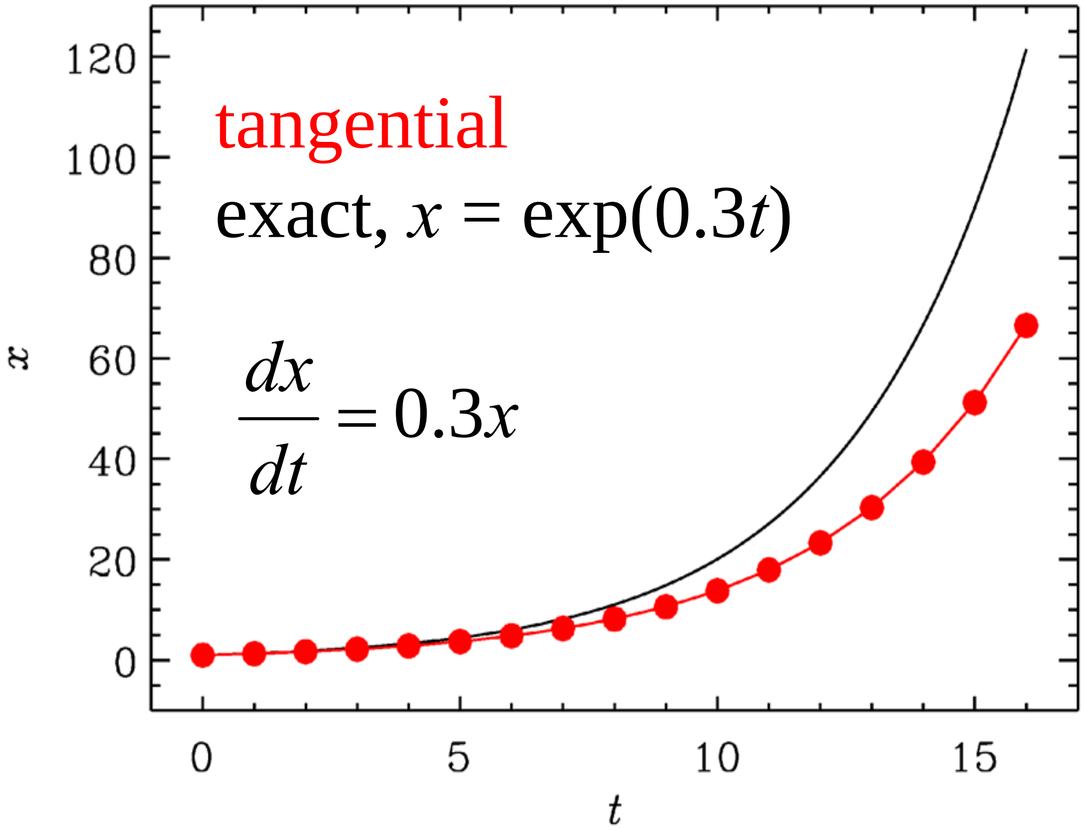
Estimate of $x_2 \approx x(t_0 + 2\Delta t)$



Derivative is now incorrect, because x_1 is not exactly $x(t_0 + \Delta t)$. The error is equivalent to having started with different initial conditions.

Errors Mount

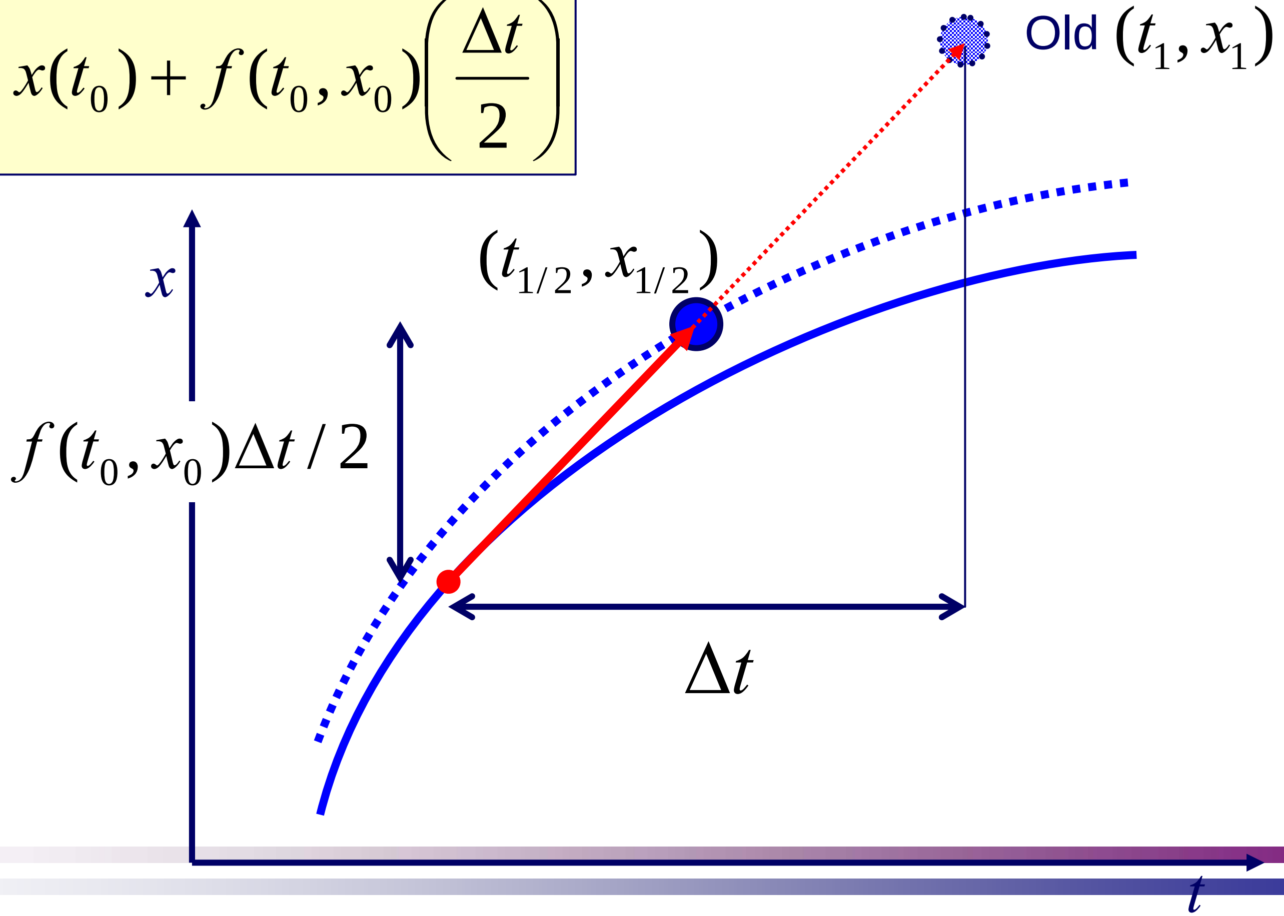
- Errors are worse than in integration of functions
  - With functions, derivative estimate is always correct
  - With diff eq's, derivative estimate becomes invetiably poorer as errors are made
  - Errors are **compounded**



Midpoint Method

- Find $x_{1/2}$ by using slope at x_0 but only moving half a time-step

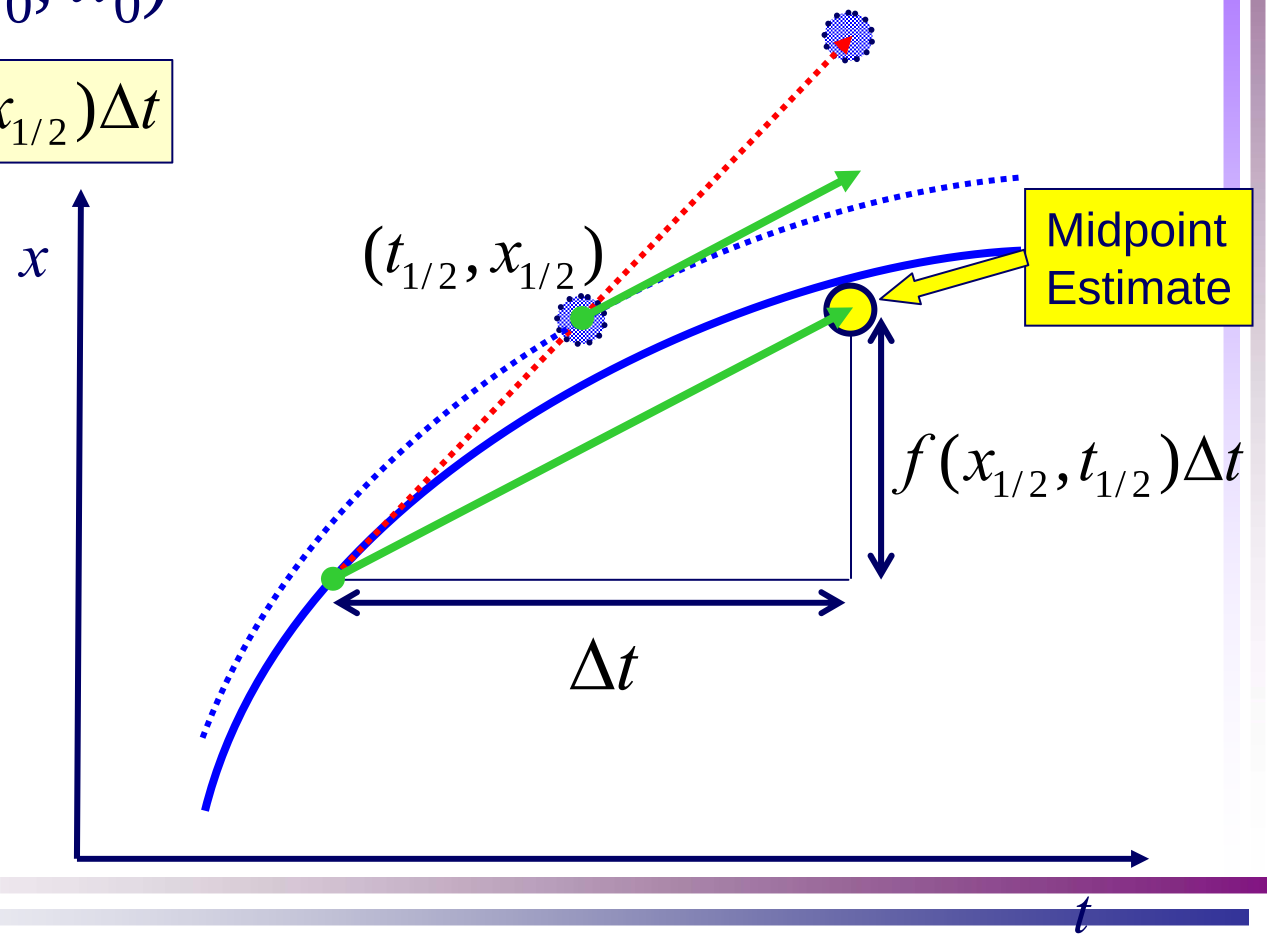
$$x_{1/2} = x(t_0) + f(t_0, x_0)\left(\frac{\Delta t}{2}\right)$$



Midpoint

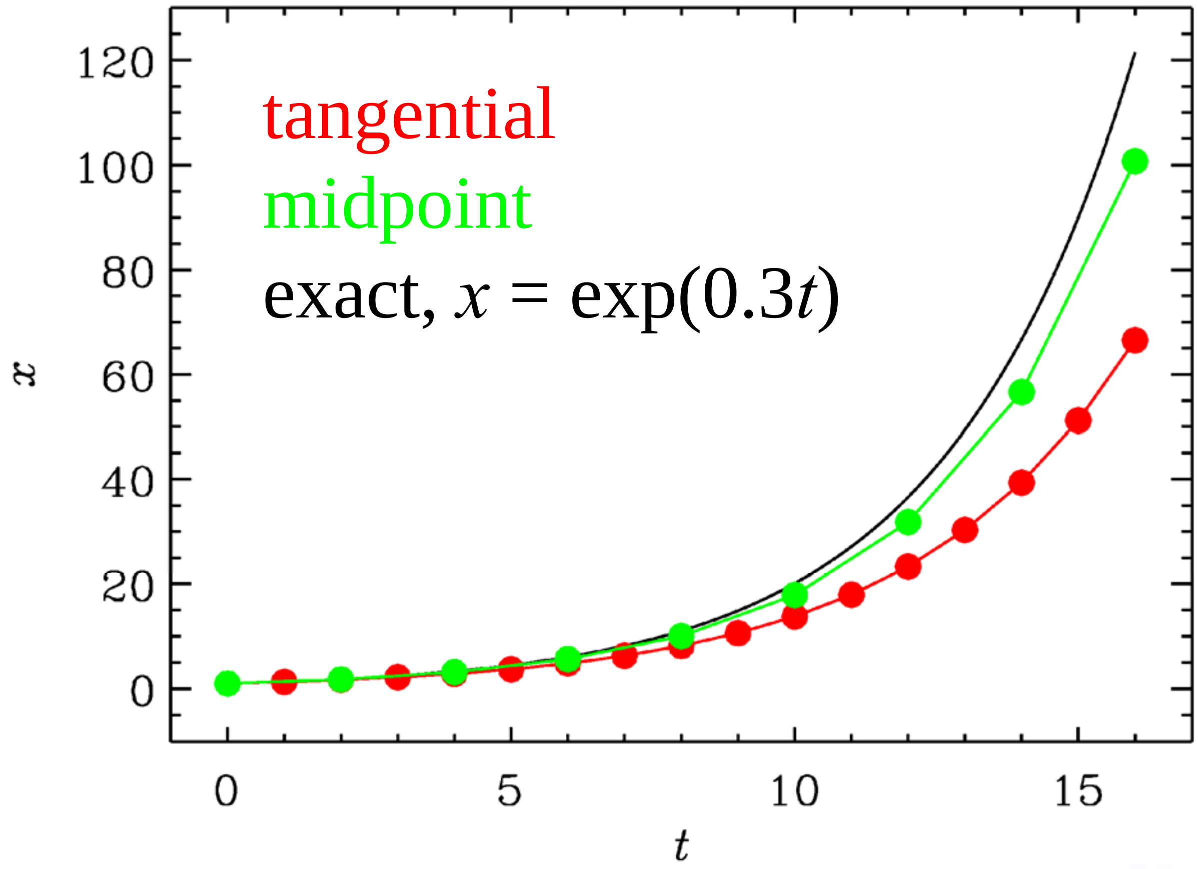
- Use slope at $(t_{1/2}, x_{1/2})$ to propagate full step from (t_0, x_0)

$$x_1 = x(t_0) + f(t_{1/2}, x_{1/2})\Delta t$$



Midpoint Formula

- Mitigates compounding errors significantly
- Allows for curvature during timestep

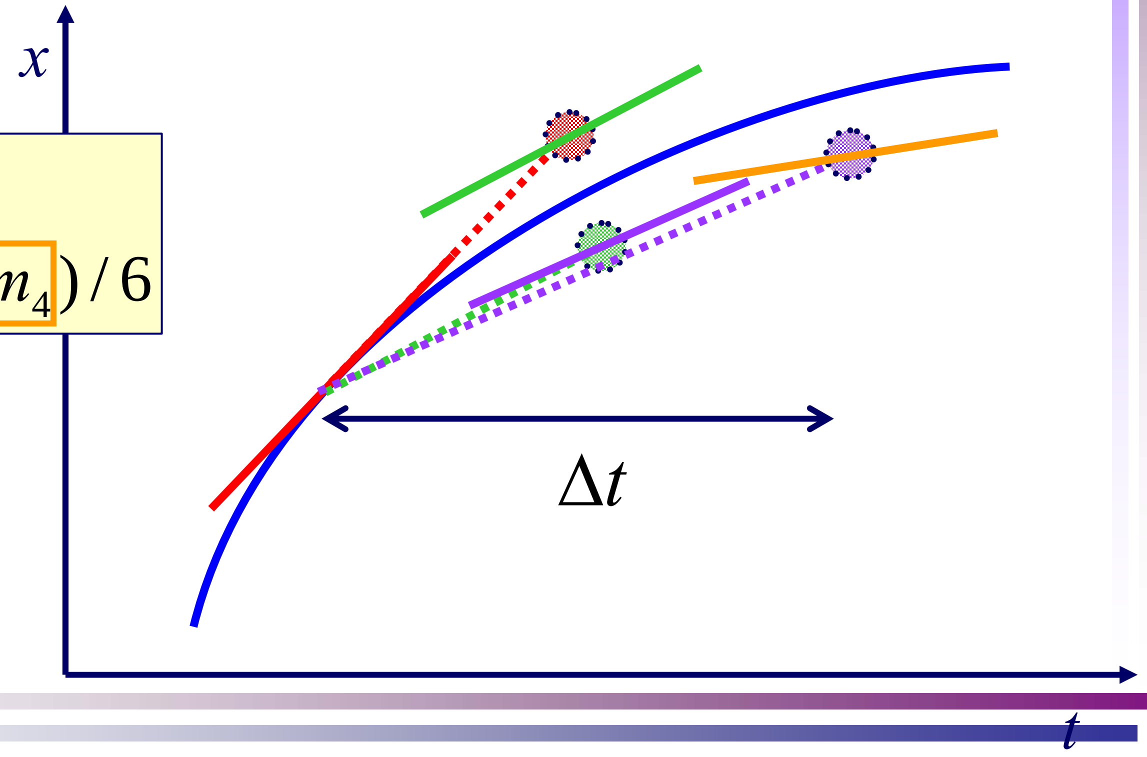


Note: 16 calls to derivative function for each

Runge-Kutta

- Similar idea to midpoint, but four points
- use slope at start to go to 1st midpoint
- use slope at 1st midpoint from start back to a 2nd midpoint
- use slope at 2nd midpoint to go to endpoint and obtain slope
- Add up slopes thus

$$x_1 = x_0 + \Delta t(\boxed{m_1} + 2\boxed{m_2} + 2\boxed{m_3} + \boxed{m_4})/6$$

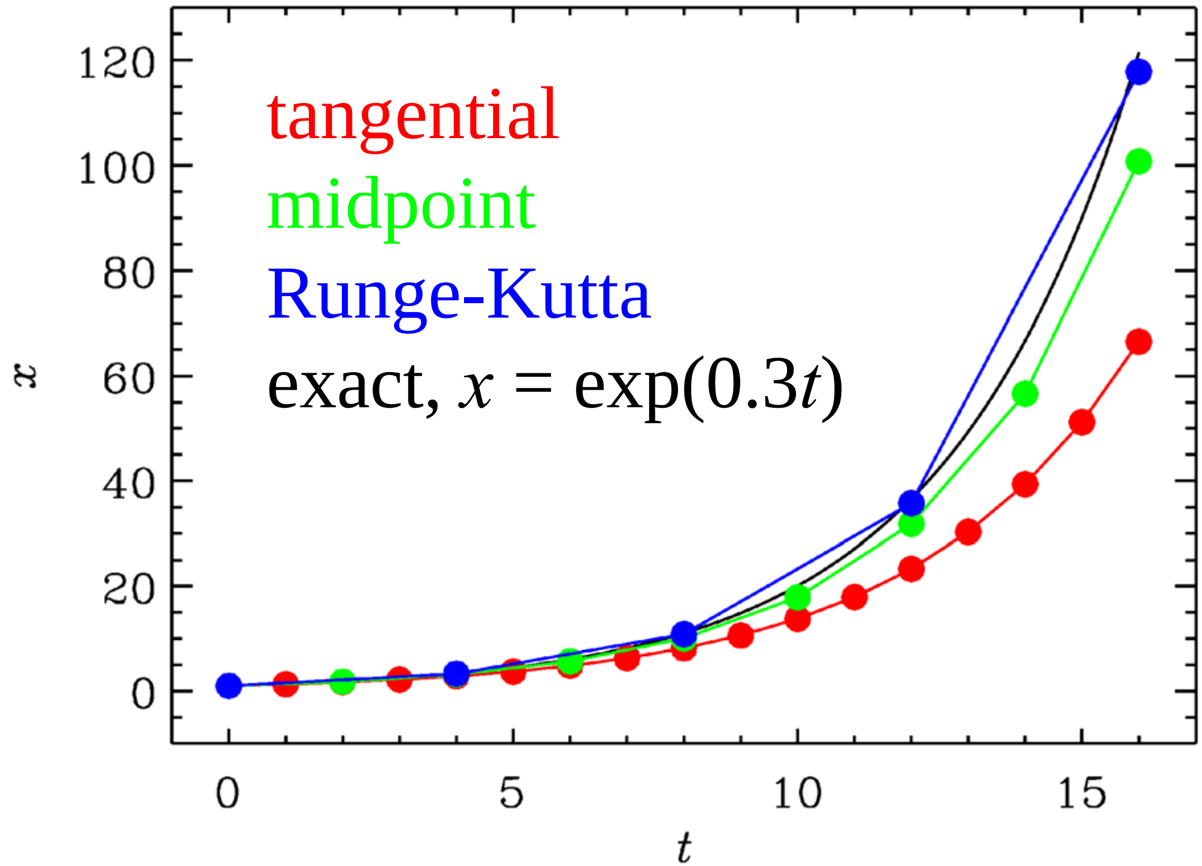


Runge-Kutta

Four-point method

- same principle as midpoint
- somewhat more complicated
  - two different midpoint evaluations
  - one endpoint evaluation
- still straight-forward to code and use

Much better behavior



Note: 16 calls to derivative function for each

Real World Example: Orbits

- Planetary orbits
  - Earthlike orbit (circular at earth distance from sun)
  - Elliptical orbit around sun
- Ballistic Missile trajectory
  - Siberian launch at Los Angeles
    - just Newtonian (Keplerian) gravity
    - no earth-rotation

$$\boxed{\mathbf{a} = -\frac{GM}{r^3}\mathbf{r}}$$

(only)

Simple Orbit Code

create 6-element state vector: x, y, z, v_x, v_y, v_z

```
const double GM = 1.33e23;
derivs(double* xv, double* dxvdt) {
    double r=sqrt(xv[0]*xv[0]+xv[1]*xv[1]+xv[2]*xv[2]);
    double r3=r*r*r;
    for (int i=0,i<3;i++) {
        dxvdt[i] = xv[i+3];
        dxvdt[i+3] = -GM*xv[i]/r3;
    }
}
```

$$\mathbf{a} = -\frac{GM}{r^3}\mathbf{r}$$

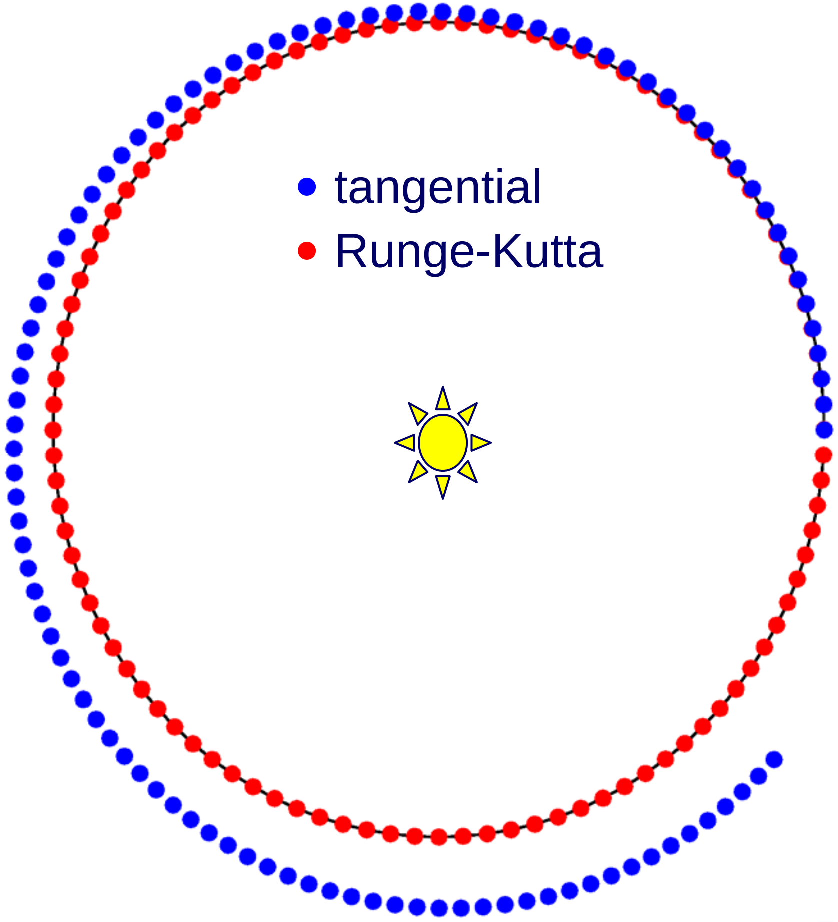
```
int main()
…
for (int i=0,i<niter;i++) {
    rk4(xv,dt,xvnew,derivs);
    xv = xvnew;
}
```

use canned RK integrator

Orbit Integration

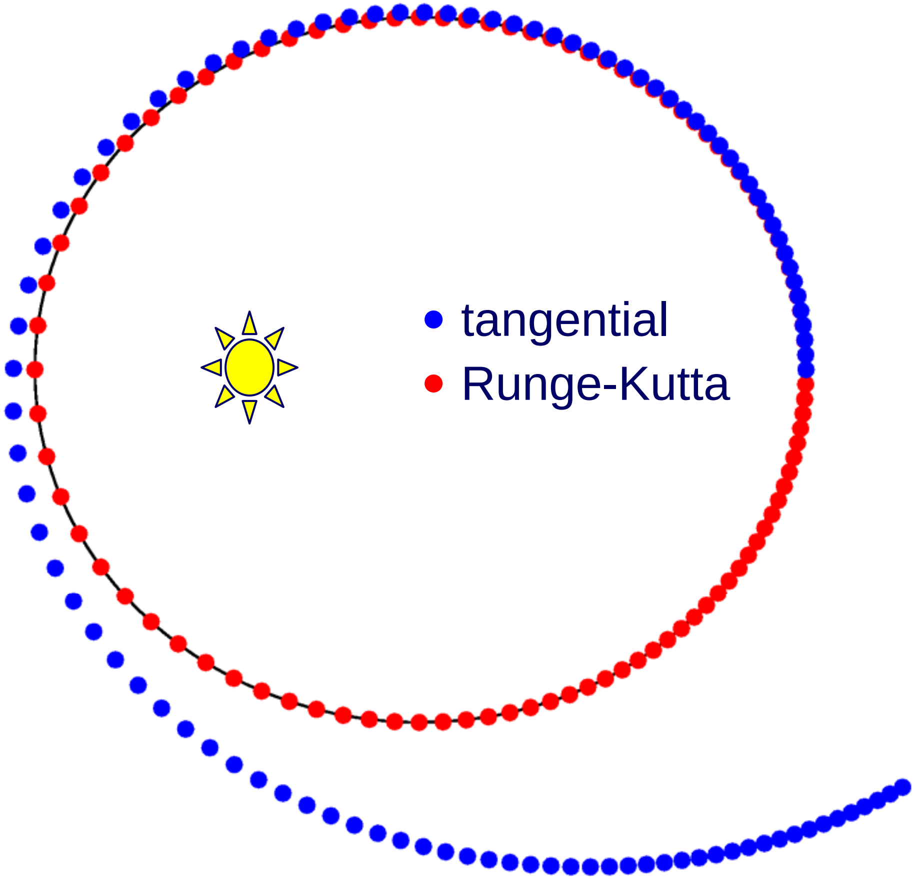
- Earthlike orbit
  - circular, 1 AU radius
- Runge-Kutta vs. Tangential (Eulerian)
  - 400 calls each to derivatives function
- Errors after one orbit

	RK	Tangential
Energy	0.000002%	14%
Position	0.00003%	82%



Orbit Integration

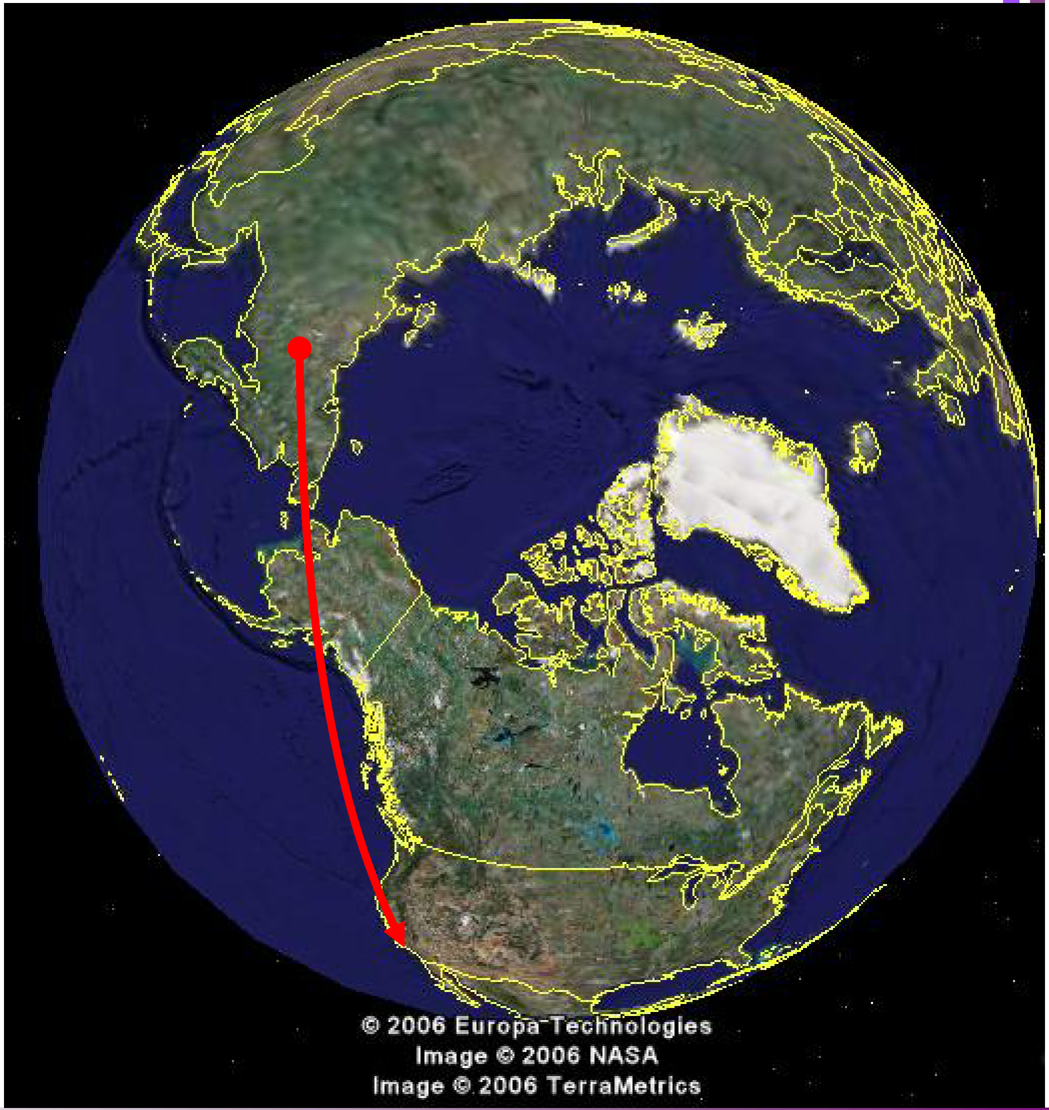
- Eccentric orbit
  - 1 AU radius
  - eccentricity = 0.5
  - b/a = 0.866
- Errors after one orbit



	RK	Tangential
Energy	0.003%	34%
Position	0.001%	70%

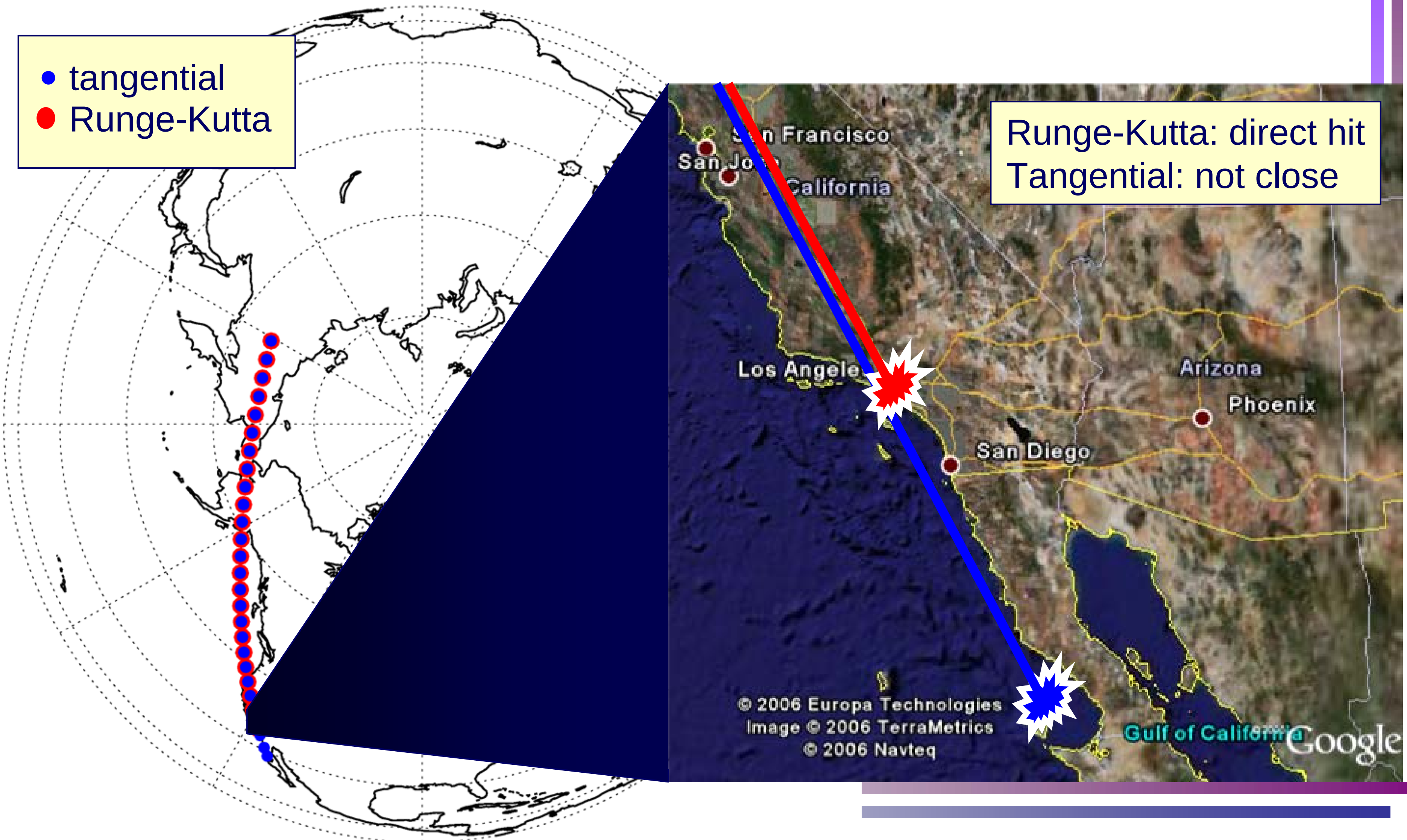
Ballistic Missile Flight

- Simple Keplerian gravity, no earth rotation
- Siberian launch, target Los Angeles
- Tangential vs. Runge-Kutta

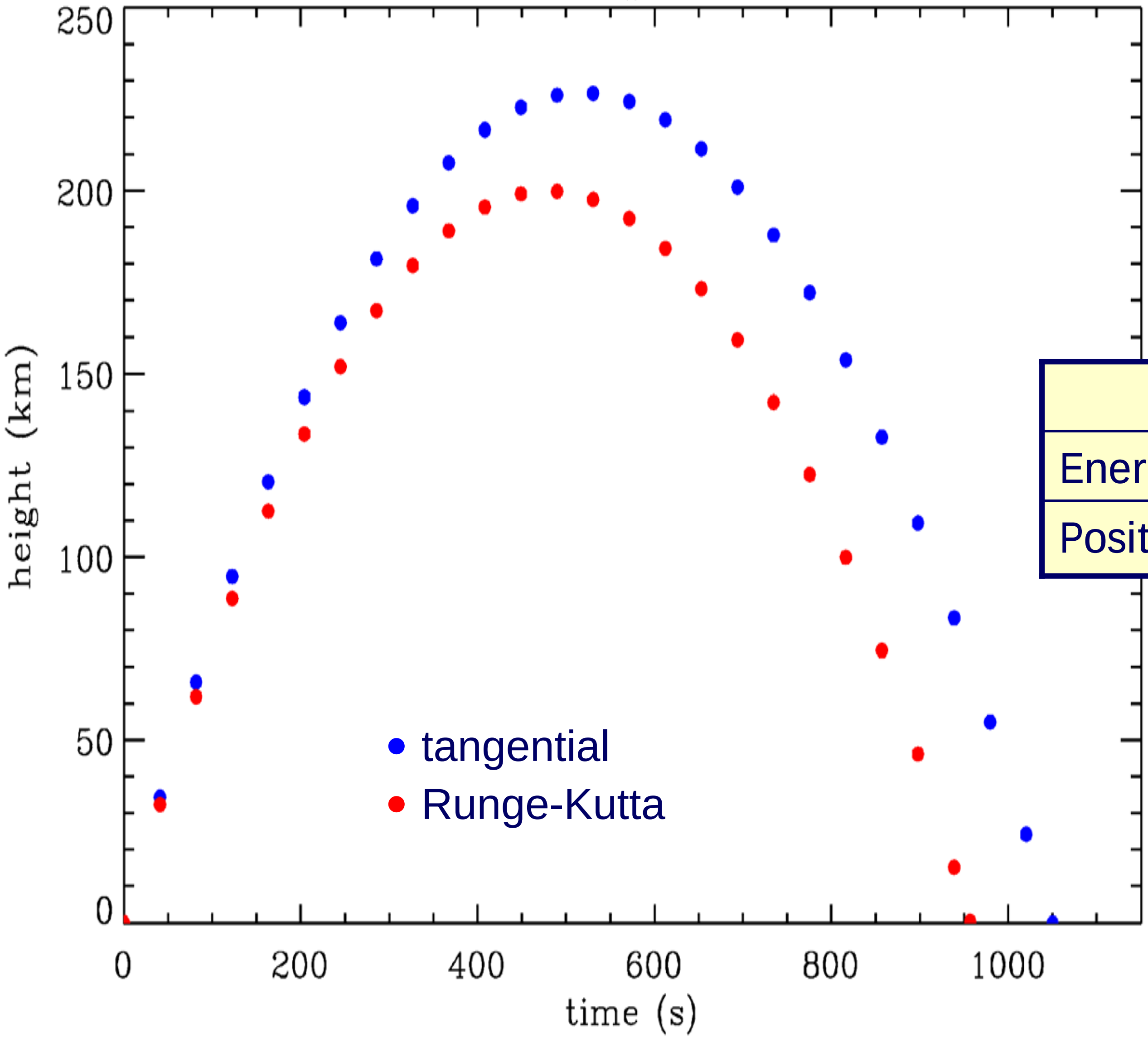


(Google)

Ballistic Missile Flight



Main Error is Height

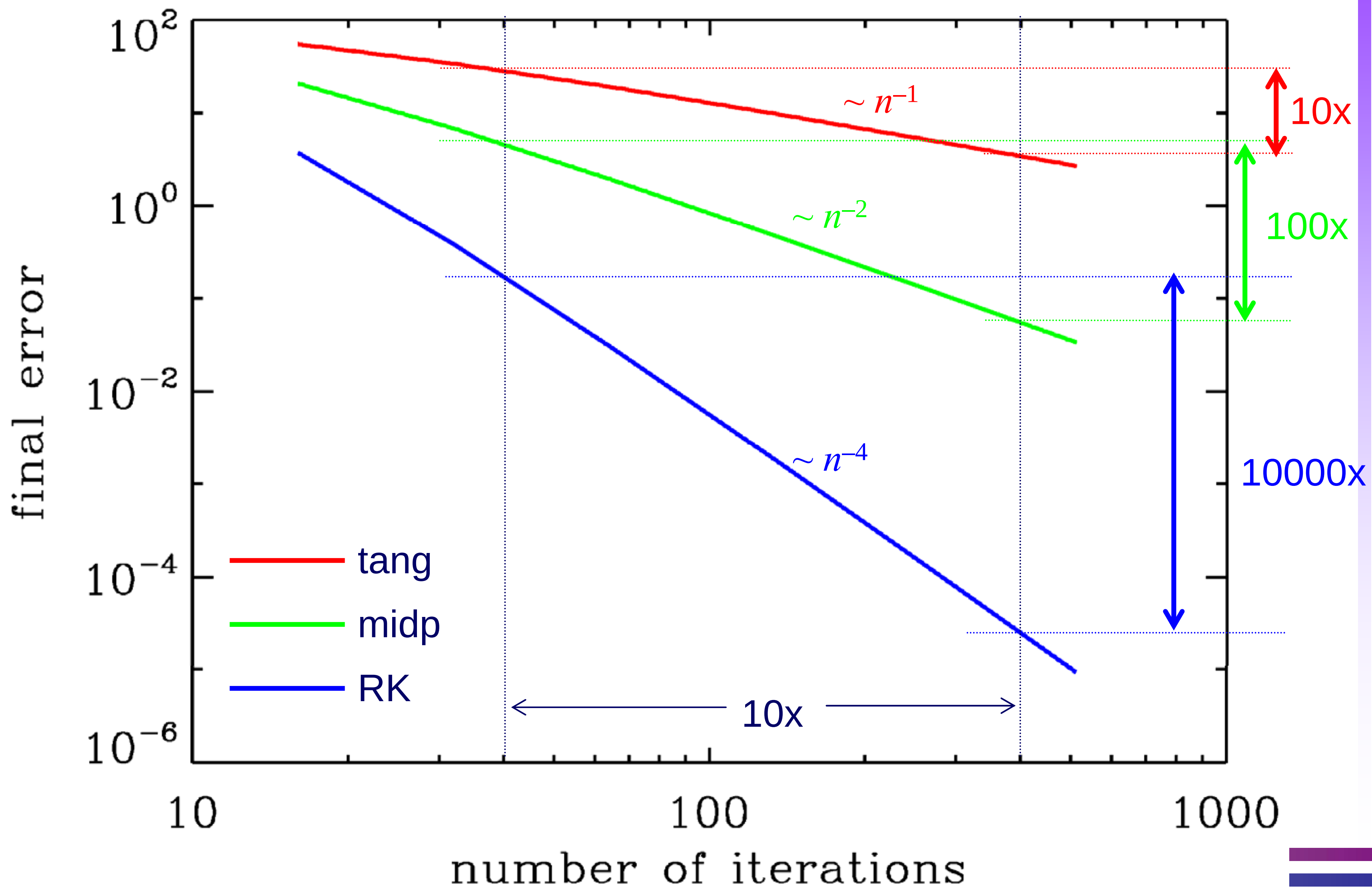


- Again, tangential overshoots
  - no curvature
- Error budget:

	RK	Tangential
Energy	$7\times10^{-7}\%$	2%
Position	3 km	660 km

Convergence

How rapidly does estimate improve with more iterations (CPU cycles)?

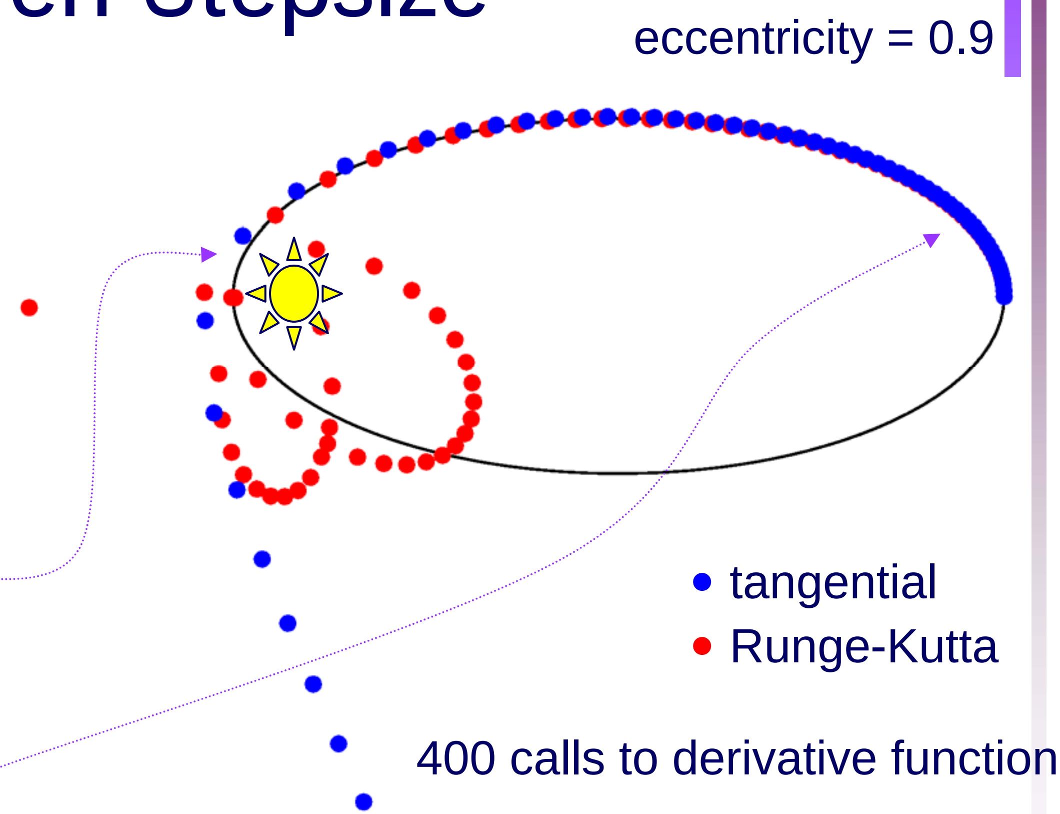


Convergence

- For same number of function calls
  - Tangential method improves linearly with increased iterations
  - Midpoint method improves quadratically with increased iterations
  - Runge-Kutta improves quartically with increased iterations
- Beware of choppy derivative functions that could screw this up

Problem with Even Stepsize

- Often the derivative function is highly variable
  - A high eccentricity orbit has much greater acceleration near the sun
- Even stepsize methods
  - far too little effor near sun (where planet zips around)
  - too much effort far from the sun (where planet moves slowly)
- Results
  - **DISASTROUS**



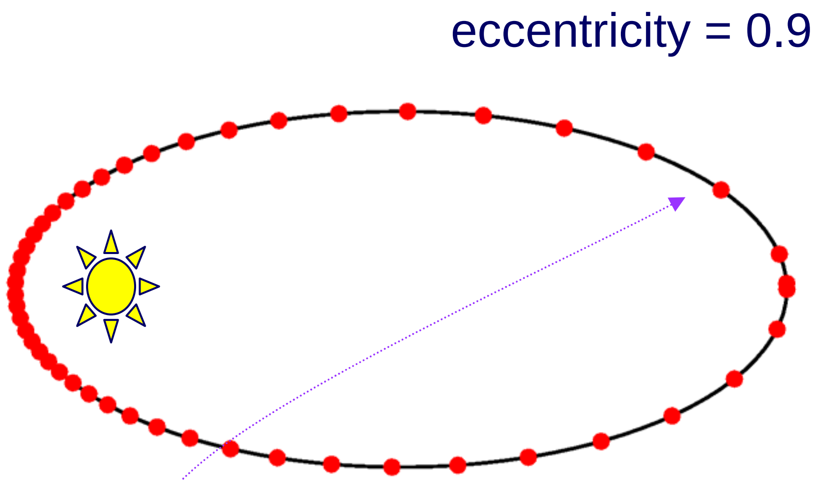
400 calls to derivative function

Errors:

	RK	Tangential
Energy	102%	540%
Position	350%	480%

Adaptive Stepsize

- Errors can be estimated along the way
  - estimates of different order with same derivative calls
- If error too large, stepsize shrinks
- If error too small, stepsize grows
- Results
  - Fine stepping near sun
  - Coarse stepping far from sun
  - **Efficient use of CPU!**



383 calls to derivative function
fewer calls!

Errors:

	Adaptive RK
Energy	0.001%
Position	0.002%

Differential Equations Summary

- Canned packages exist for Runge-Kutta
  - it's a good place to start
  - usually doesn't get you into too much trouble
- Consider adaptive stepsize
  - if derivative is known to vary a lot or suddenly
- Other methods: Bulirsch-Stoer, etc.
  - may offer radically fast performance, if derivatives are reasonably stable
  - often very similar calls can be made to multiple integrators, so play around!

Outline

- Intro to physics in M&S
- Quadrature (Integration of Functions)
- Integration of Differential Equations
  - orbits and trajectories
- Radiative Processes
  - atmospheric effects on visibility
- Fourier methods
  - image processing

Radiative Processes Segue

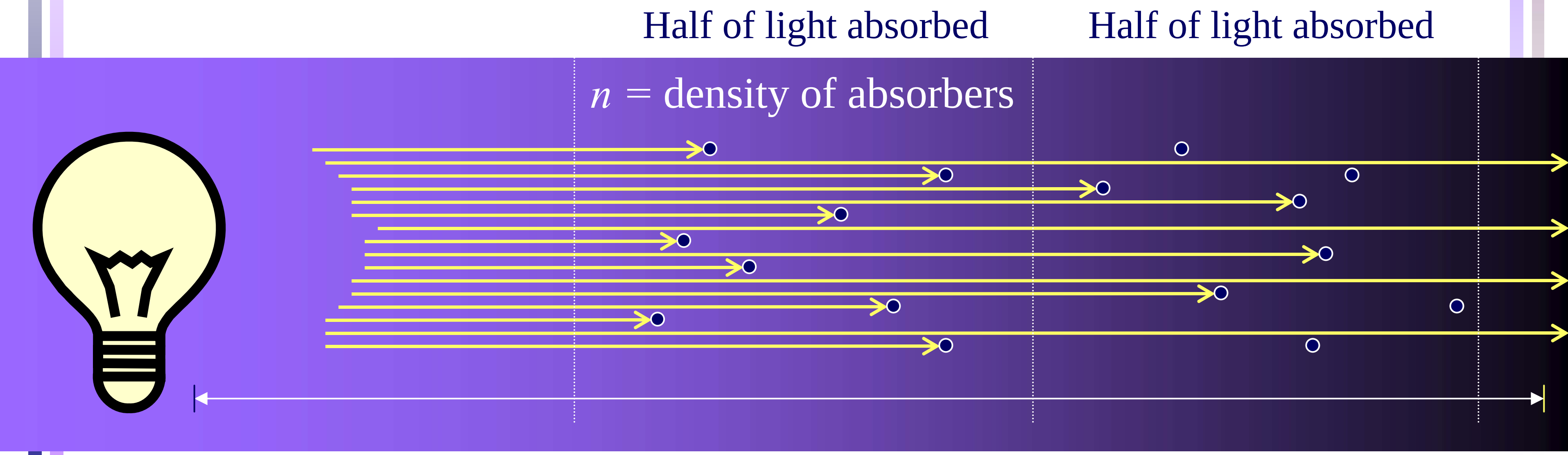
- Now an example of seemingly complicated physics
- But an extremely simple mathematical solution
  - can't get more efficient than that!

Radiative Processes

- Optical/IR detection depends not only on an obstruction-free line-of-sight, but also on atmospheric effects
- The atmosphere can basically do two things to light
  - absorb
  - scatter
- Fortunately, the math for these is straight-forward

Absorption

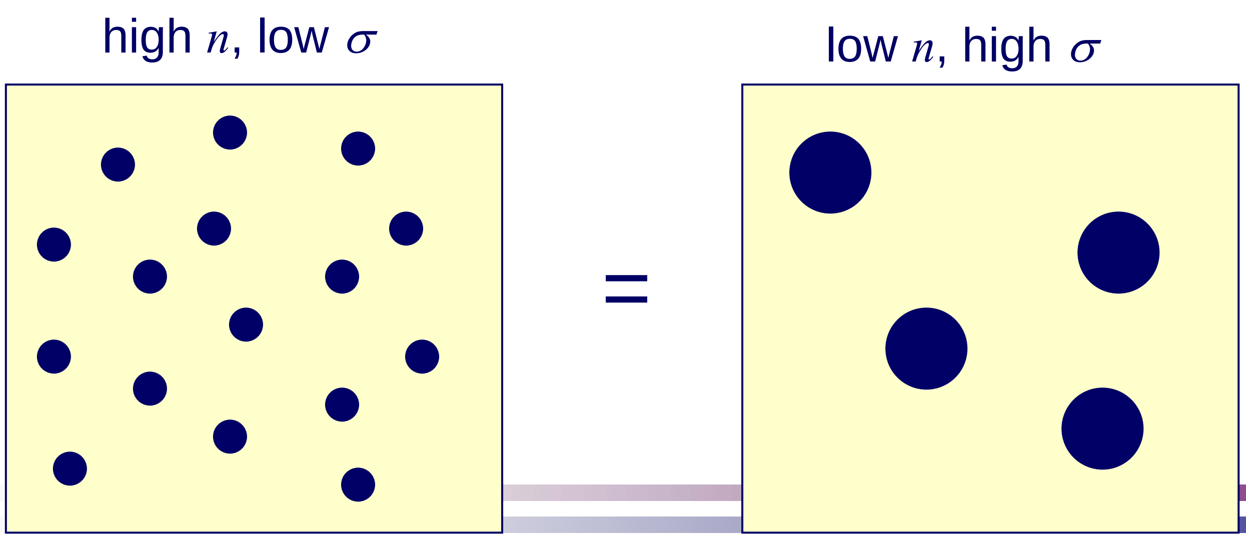
- Absoprtion attenuates light exponentially with distance.
  - If half of light is absorbed in the first meter, half of the remaining light is absorbed in the second
- Exponent proportional to density



Absorption Math

- Absorption is quantified in terms of an opacity κ, in units of m^{-1}
- Opacity is the product of the number density, n, and the cross-section, σ, of the absorbing particle

$$\kappa = n\sigma$$



Absorption Math

- Optical Depth, τ, is the product of κ and the distance to the object of interest
  - or integral over the distance
- The light received is simply

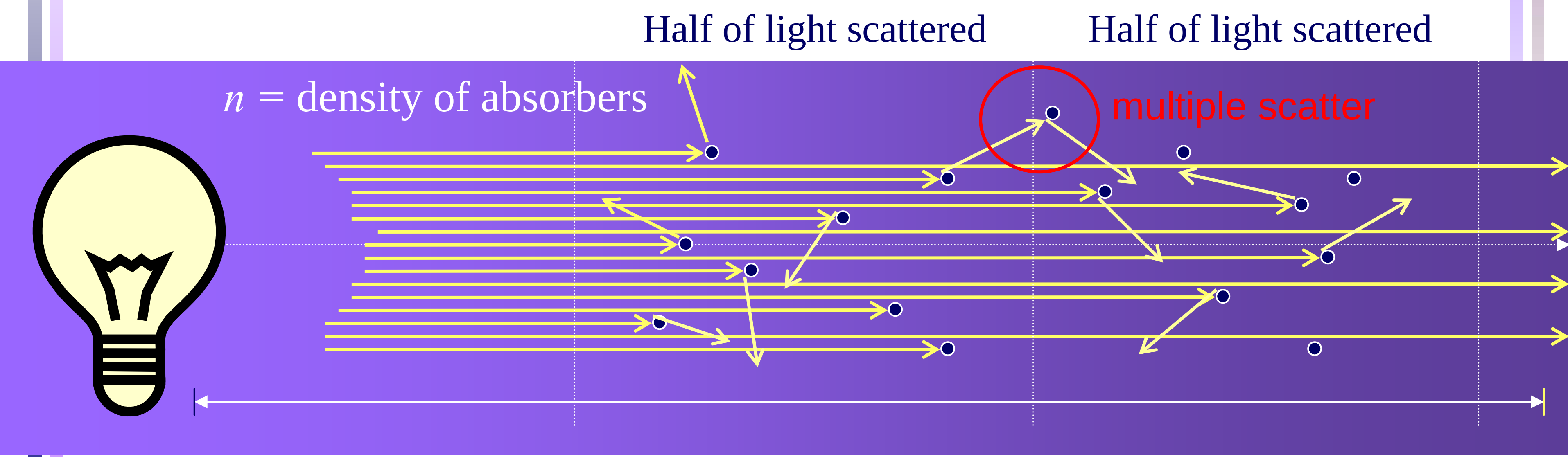
$$I = I_0 \exp(-\tau)$$

$$\tau = \int \kappa ds = \int n \sigma ds$$

Note that σ depends on quantum interaction probabilities, but tables are well-established for countless species.

Scattering

- Scattering features particles that bounce light in a random direction
  - light isn't attenuated by made more uniform in medium
    - smoke, fog, snow, rain
- Effect is again proportional to density



Intuitive Fog Example

- Demonstrate fog mathematics
- Problem, need 3-D
- Photograph selected for easy 3-D model
  - green pixels aren't Parris
  - ground is a simple plane
  - background trees treated as a plane

3-D Toy Model

- Dark = far
- light = close

- some minor errors

Toy Fog Model

- Fairly convincing to the eye

no fog, $\kappa = 0$

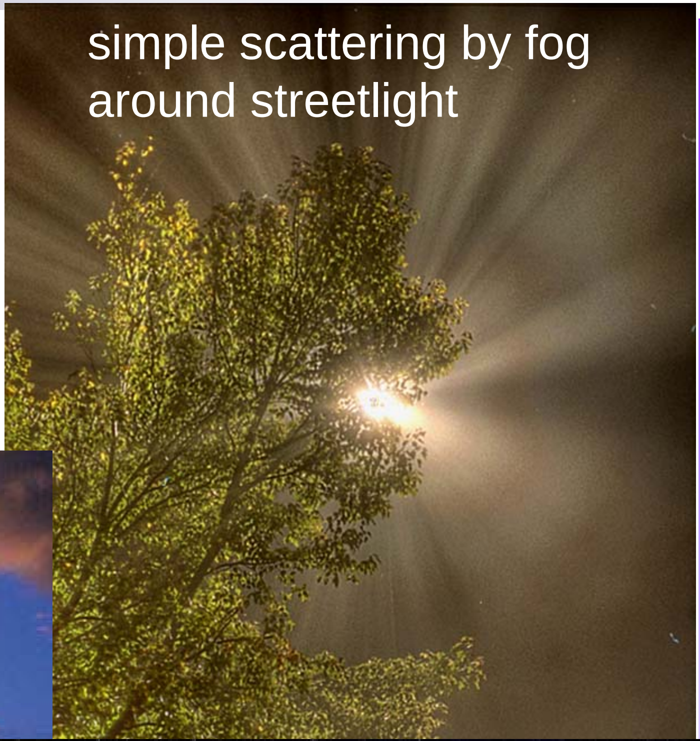
thin fog, $\kappa = 0.01$

thicker fog, $\kappa = 0.05$

strictly notional, but math is right

Real Scattering

Wavelength-dependent scattering (blue more scattered than red) = reddened sun, blue sky



simple scattering by fog around streetlight

more blue scattering

Toy Absorption

- Very fine coal dust?

$\kappa = 0$

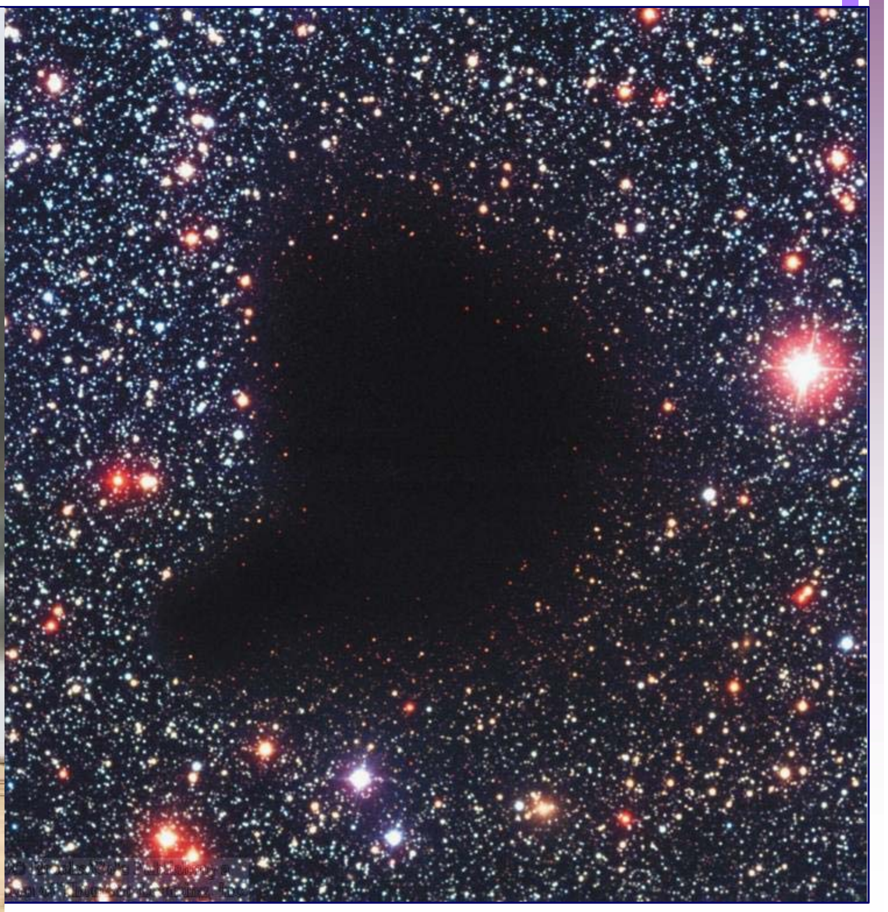
$\kappa = 0.01$

$\kappa = 0.05$

Real Absorption

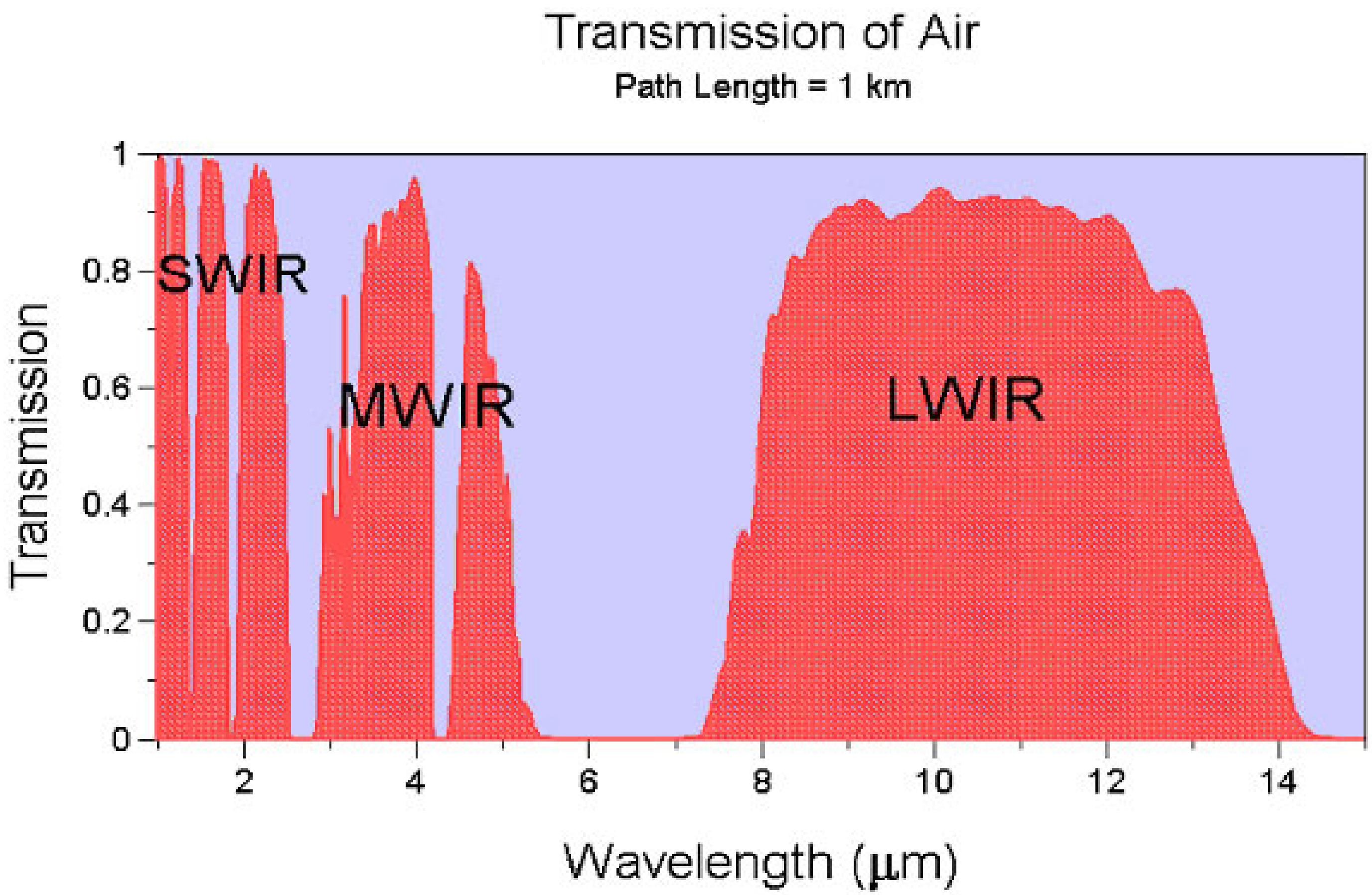
Black smoke from Iraqi oil fire

Dark interstellar dust



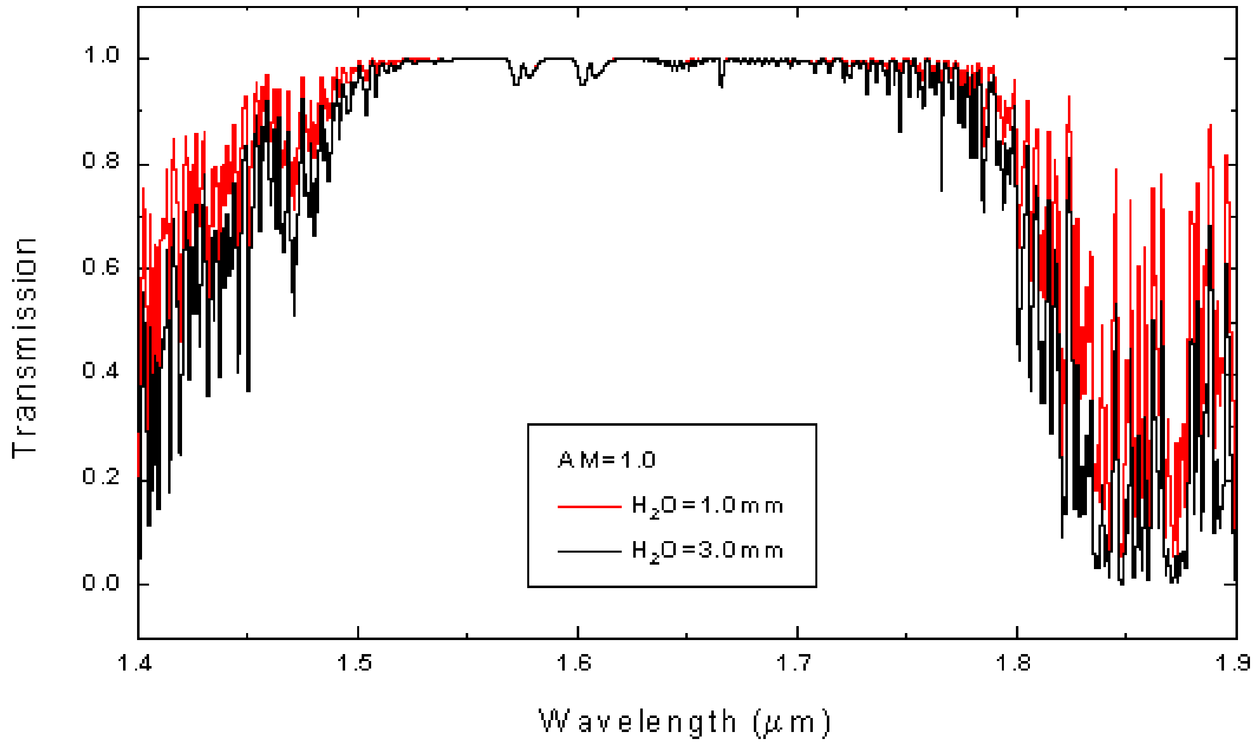
Infrared Absorption

- Water vapor (among other molecules) very effectively absorbs infrared radiation



Water Vapor and IR

- Spectral dependence is quite complex
  - water densities given in terms of mm
    - if I took all the water vapor in line of sight and made it liquid, how much water would I have?



Mike Skrutskie (UVa)

In practice

- Integrate the absorption spectra against the bandpass of your detector to get a simple function of total absoprtion vs. water column.

Outline

- Intro to physics in M&S
- Quadrature (Integration of Functions)
- Integration of Differential Equations
  - orbits and trajectories
- Radiative Processes
  - atmospheric effects on visibility
- Fourier methods
  - image processing

Fourier Transform Segue

- The purpose here is to present a very advanced computational technique with a great many applications

Intro

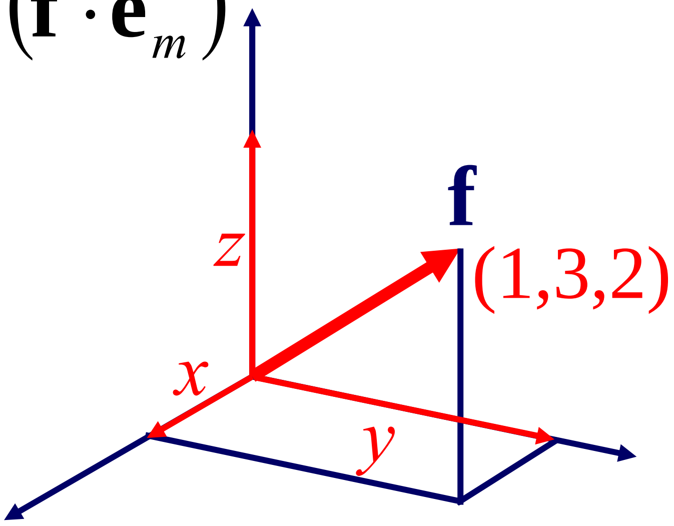
- The Fourier Transform
  - facilitates solution of partial differential equations
  - has applications in
    - compression
    - image processing
    - signal analysis
    - statistics
- The big advantage:
  - Allows many N^2 processes to be carried out in $N \log N$ time.
- First, the MATH

Basis Vectors

- Consider 3-vectors
- 3 coordinates are really projections of the vector onto the independent axes
- Each coordinate can be formed by taking the dot product of the vector with the axis's **basis vector**:

$$\mathbf{f} = \sum_{m=x,y,z} \hat{\mathbf{e}}_m \left(\mathbf{f} \cdot \hat{\mathbf{e}}_m\right)$$

$$x = \hat{\mathbf{e}}_x \cdot \mathbf{f} = (1,0,0) \cdot \mathbf{f} = 1$$
$$y = \hat{\mathbf{e}}_y \cdot \mathbf{f} = (0,1,0) \cdot \mathbf{f} = 3$$
$$z = \hat{\mathbf{e}}_z \cdot \mathbf{f} = (0,0,1) \cdot \mathbf{f} = 2$$



Find components by taking dot-product with basis vectors

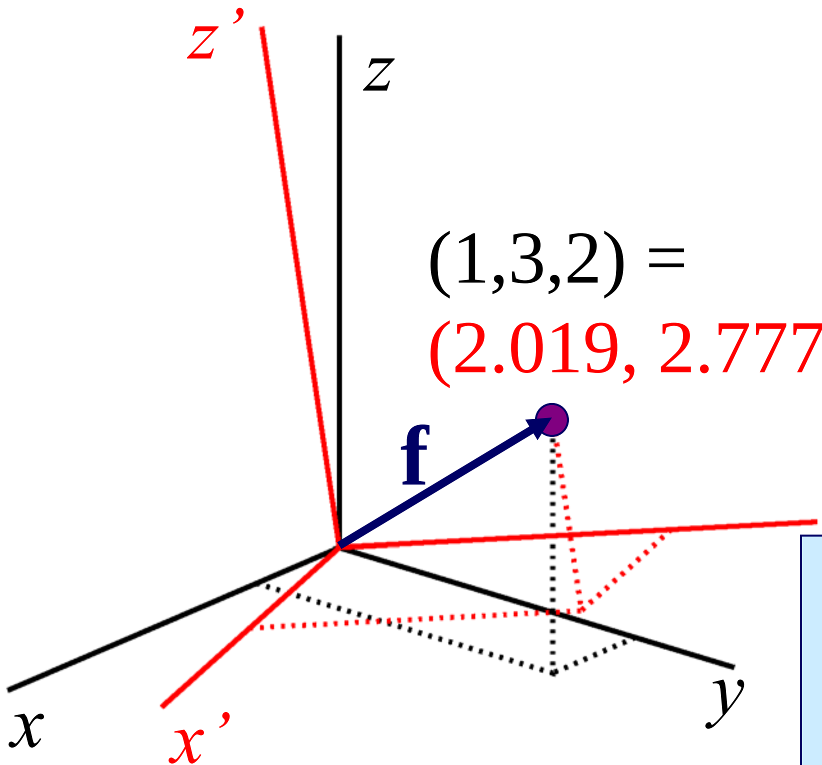
Different Basis Sets

- Example: rotated coordinate axes

$$x' = \hat{\mathbf{e}}_{x'} \cdot \mathbf{f} = 2.018725$$

$$y' = \hat{\mathbf{e}}_{y'} \cdot \mathbf{f} = 2.777474$$

$$z' = \hat{\mathbf{e}}_{z'} \cdot \mathbf{f} = 1.486739$$



(1,3,2) =
(2.019, 2.777, 1.487)'

$$\mathbf{f} = \sum_{m'} \hat{\mathbf{e}}_{m'} \left(\mathbf{f} \cdot \hat{\mathbf{e}}_{m'} \right)$$

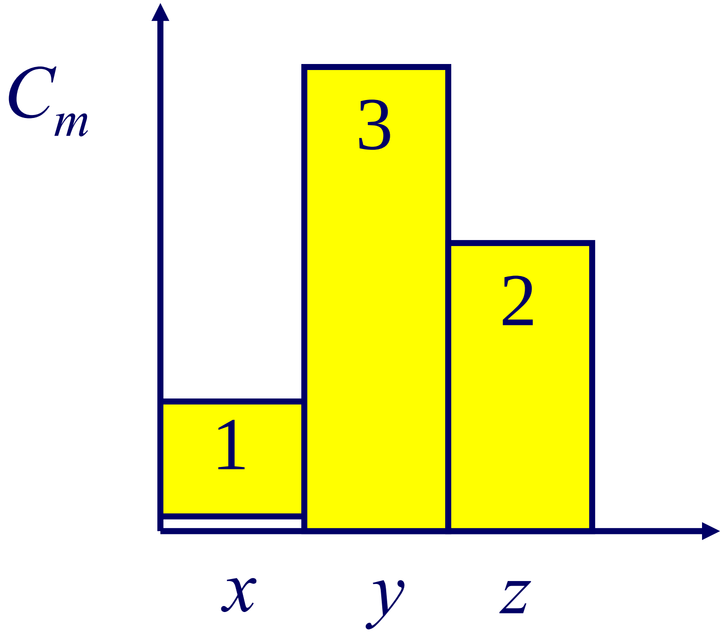
Same **f**, different representation. Still find components by taking dot-product with basis vectors

A different way to see it

- Regard x,y,z components as heights on a 3-bar histogram
  - exactly same information contained

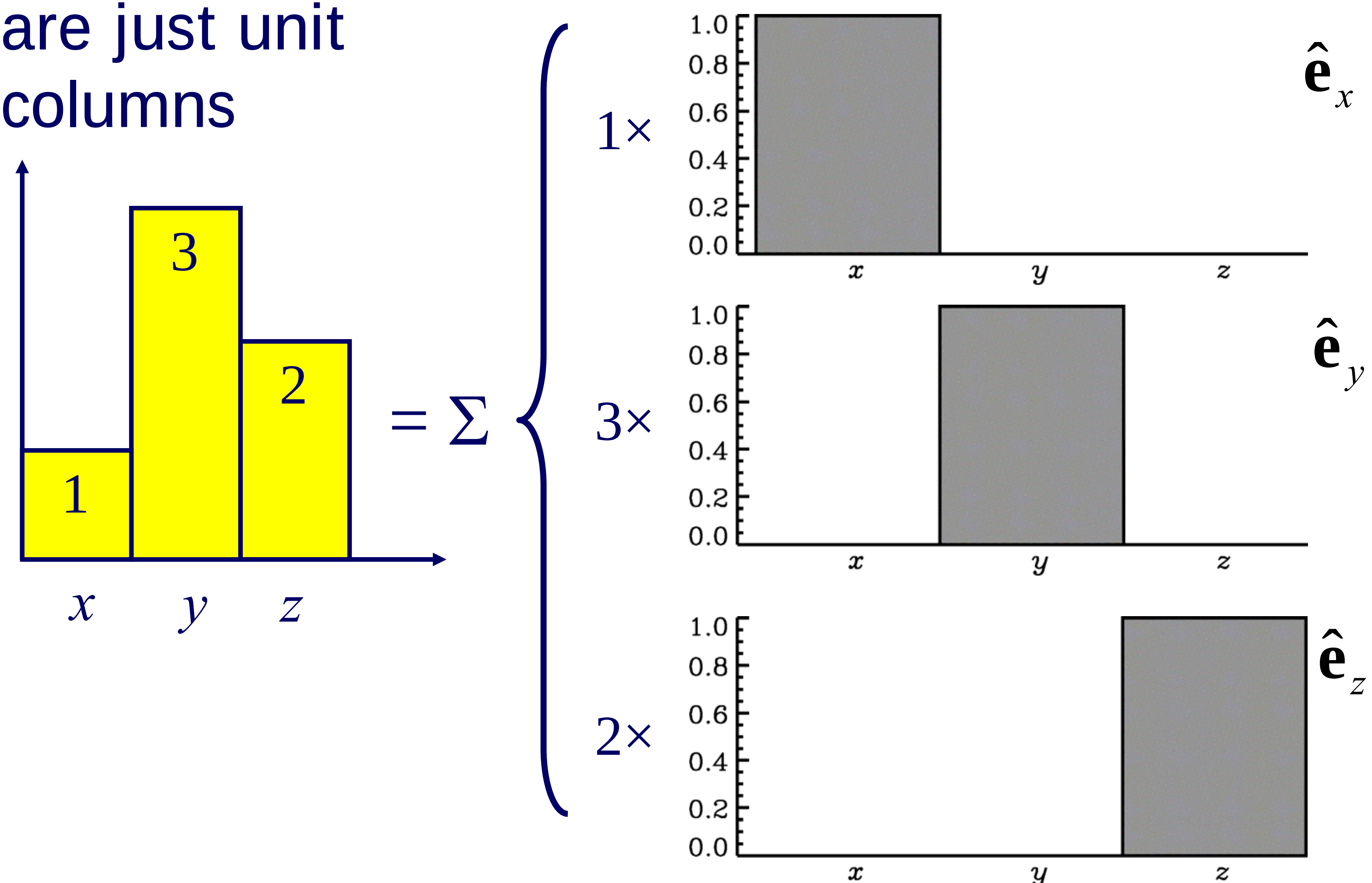
$$\text{height} = \mathbf{f}$$

$$\mathbf{f} = \sum_{m=x,y,z} C_m \hat{\mathbf{e}}_m$$

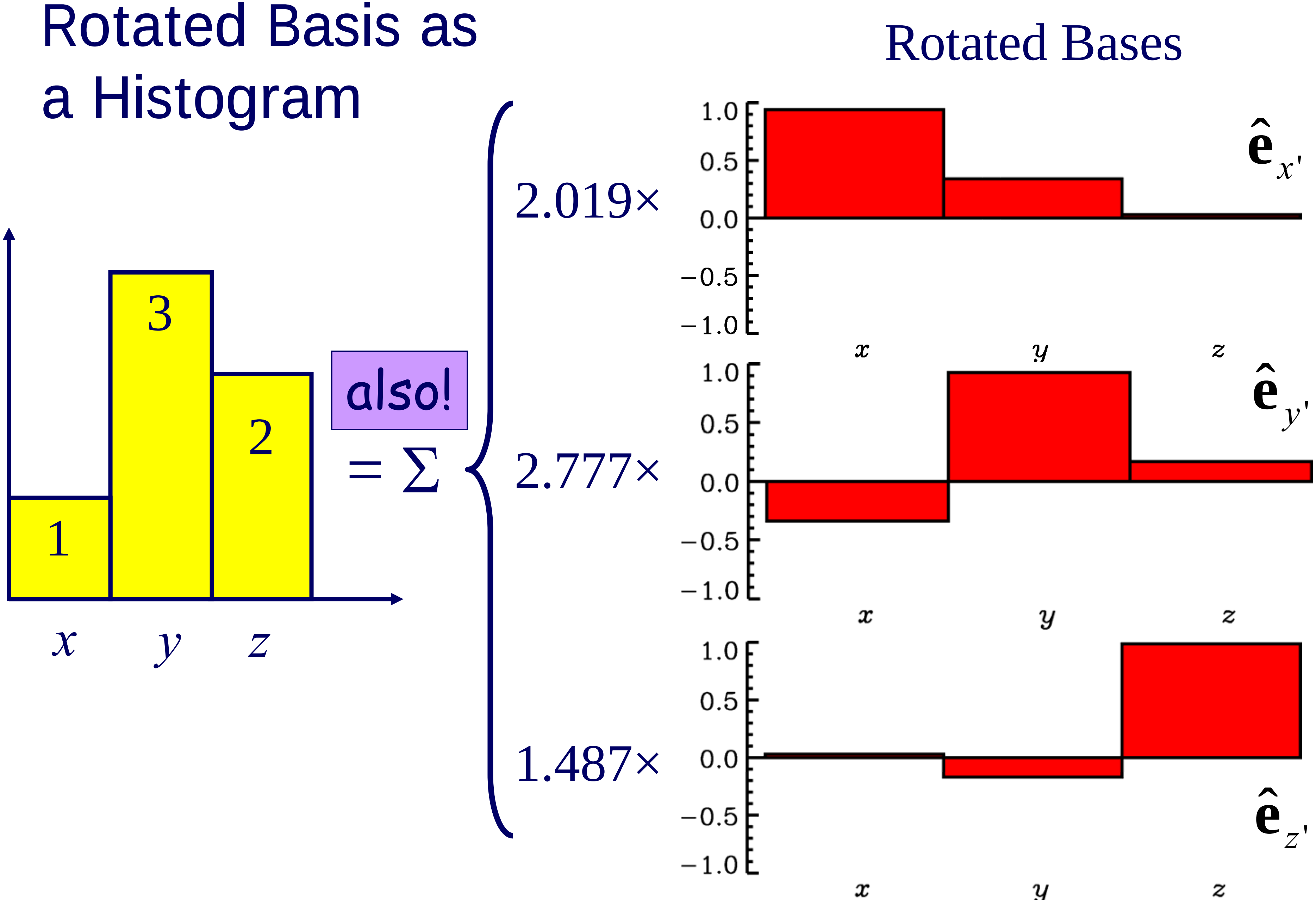


Now basis vectors are just unit columns

Cartesian Bases



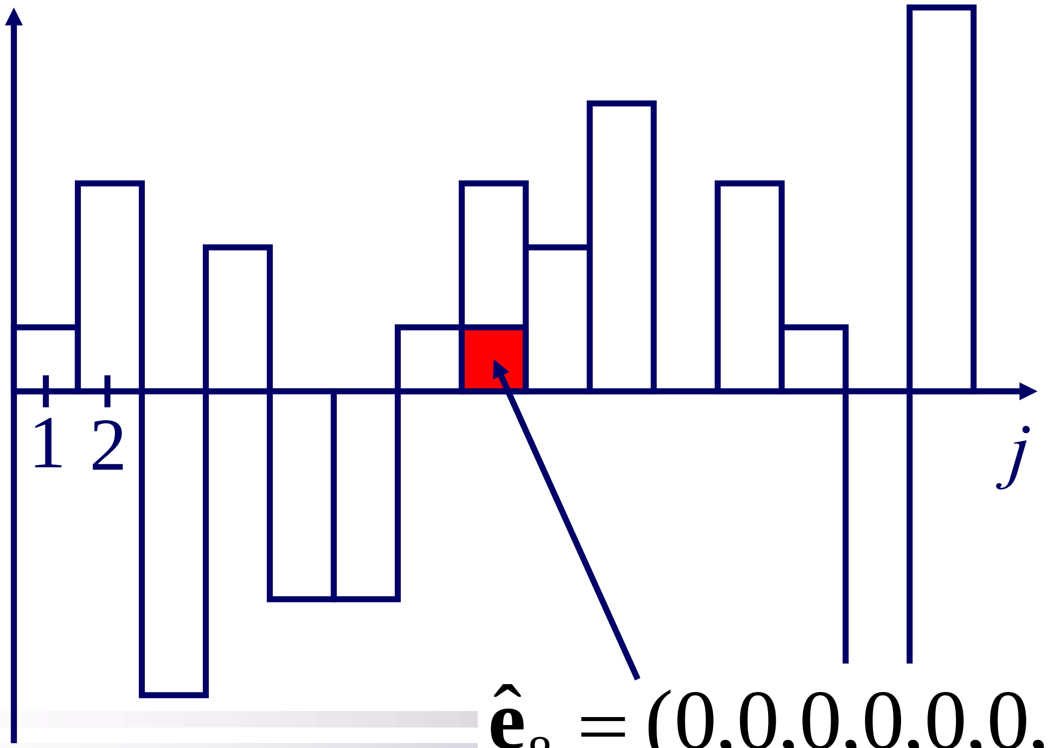
Rotated Basis as a Histogram



n -vectors

- Bar graph simply grows, with dimension along horizontal axis

$$\mathbf{f} = (1,3,-5,2,-3,-3,1,3,2,4,0,3,1,-5,6)$$



$$f_j = \hat{\mathbf{e}}_j \cdot \mathbf{f}$$

$$\hat{\mathbf{e}}_8 = (0,0,0,0,0,0,0,1,0,0,0,0,0,0,0)$$

Rule for basis vectors

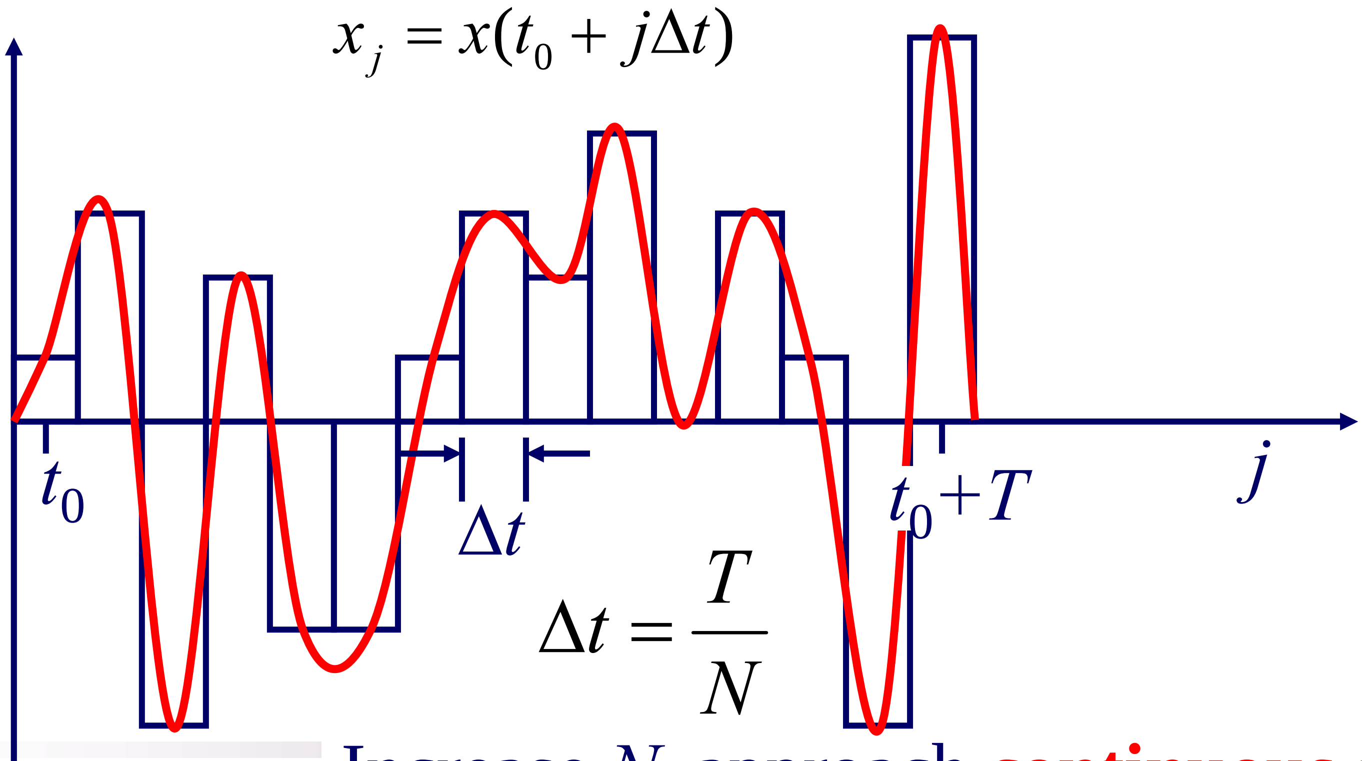
- Vectors must remain orthonormal

$$\hat{\mathbf{e}}_i \cdot \hat{\mathbf{e}}_j = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$$

- Simple enough
- lots of possibilities
  - we'll focus on one shortly

Adding more dimensions...

- With enough dimensions, the vector starts to look like a function



Increase N, approach **continuous** function

Basis of interest

- **Discrete Fourier Transform**, for a vector of length N

$$\left(\hat{\phi}_m\right)_j = \exp\left(\frac{2\pi i m}{N} j\right)$$

$$i = \sqrt{-1}$$

$$C_m = \mathbf{f} \cdot \hat{\phi}_m^* = \sum_j f_j \left(\hat{\phi}_m^*\right)_j$$

Asterisk is for complex conjugate

$$\mathbf{f} = \frac{1}{N} \sum_m C_m \hat{\phi}_m$$

Note:

$$\exp\left(\frac{2\pi i m}{N} j\right) = \cos\left(\frac{2\pi m}{N} j\right) + i \sin\left(\frac{2\pi m}{N} j\right)$$

Don't Panic! Tedious Math is Hidden...

- In IDL, for vector `f`

```
IDL> C = fft(f)
IDL> fsame = fft(C,/inverse)
```

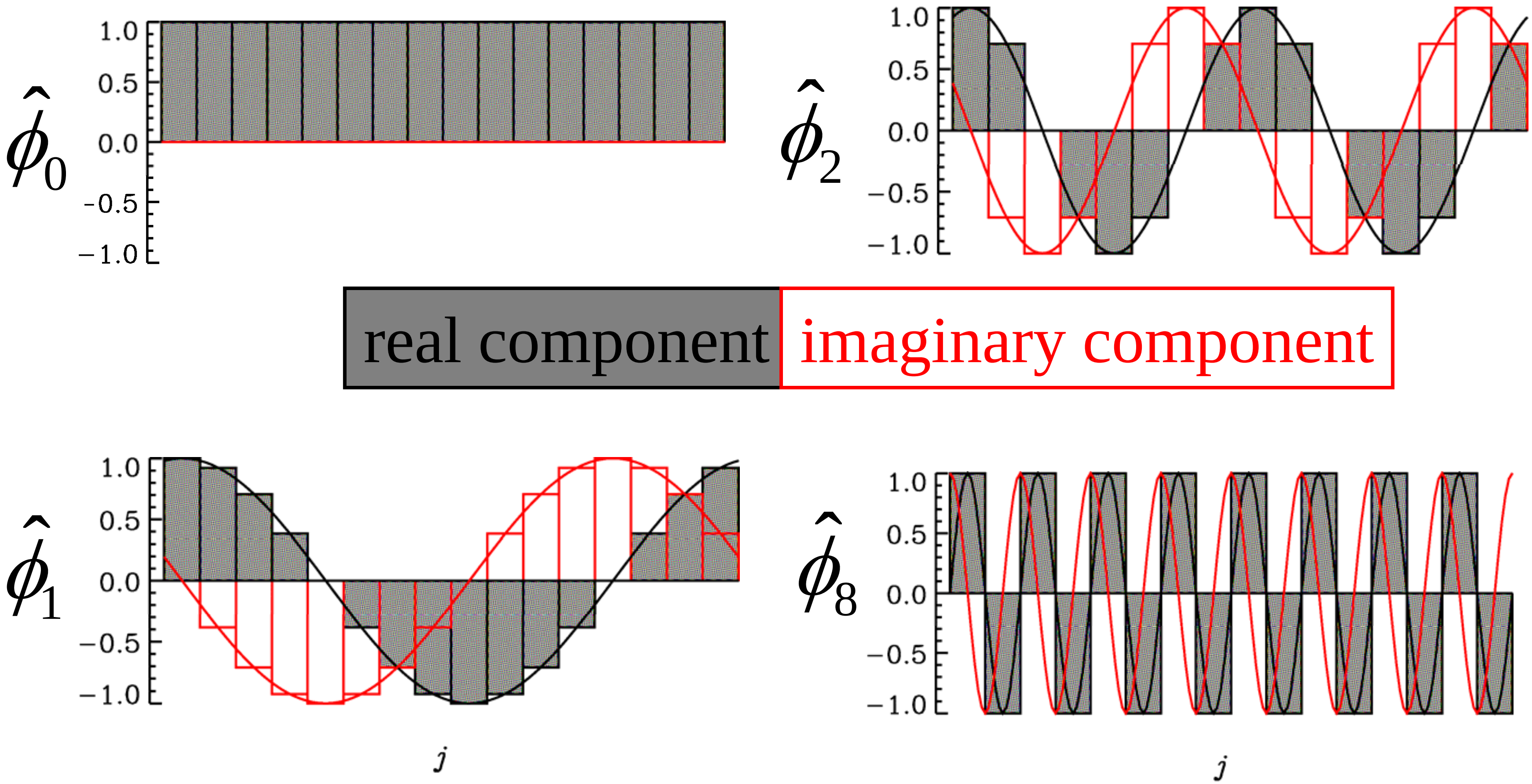
- Using Numerical Recipes in C++

```
int main() {
    …
    NR::four1(a,1)
    NR::four1(b,-1)
    …
}
```

(NR uses in-place storage)

Okay, what does this look like?

For $N = 16$

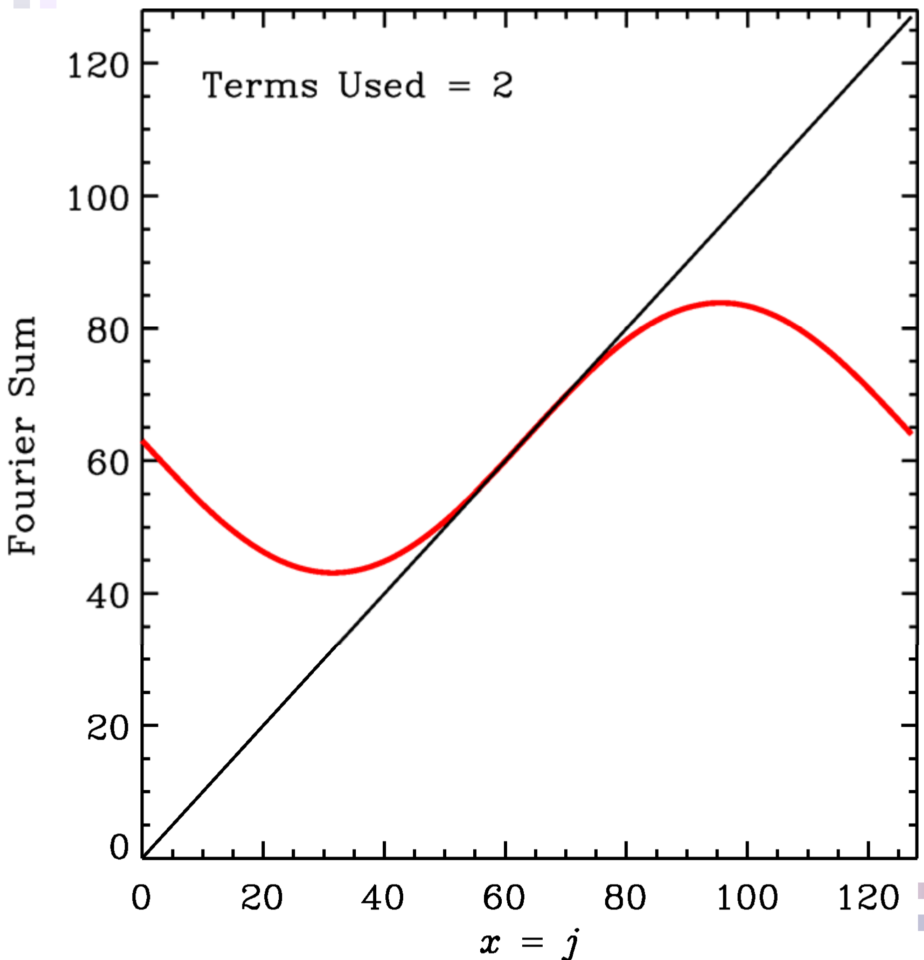


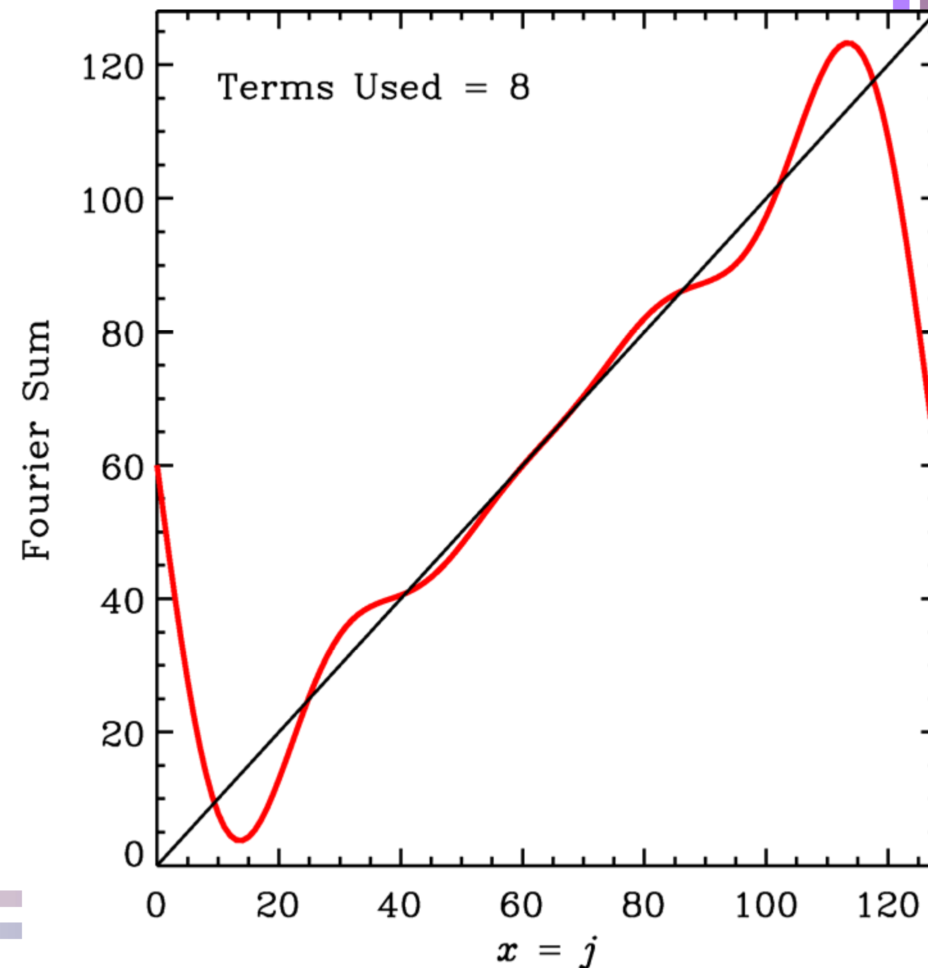
Example: $f_j = j, N = 128$

$$C_m = \mathbf{f} \cdot \hat{\phi}_m^*$$

$$\mathbf{f} = \frac{1}{N} \sum_m C_m \hat{\phi}_m$$

- Building up the sum



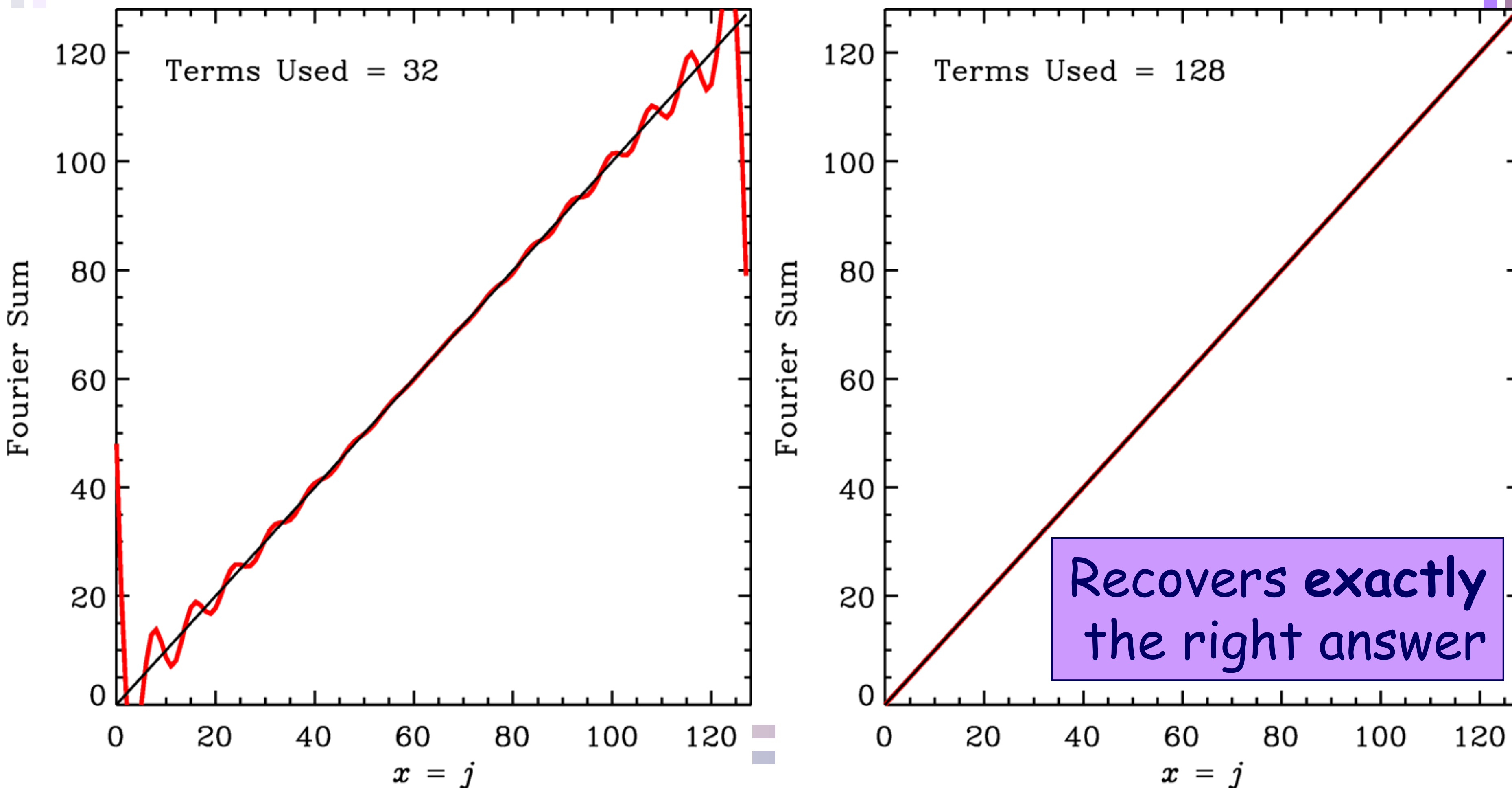


Example: $\mathbf{f} = j, N = 128$

$$C_m = \mathbf{f} \cdot \hat{\phi}_m^*$$

$$\mathbf{f} = \frac{1}{N} \sum_m C_m \hat{\phi}_m$$

- Building up the sum

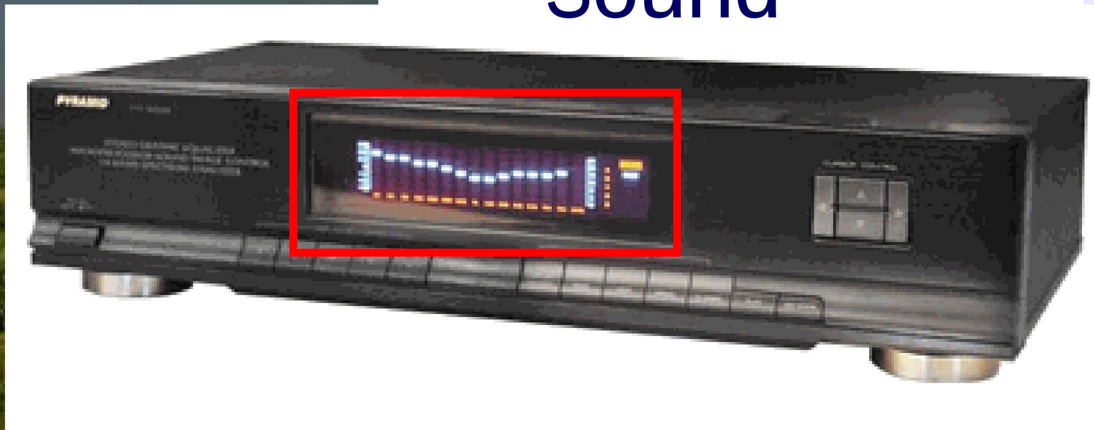


Why do this?!

- Spectrum of Light

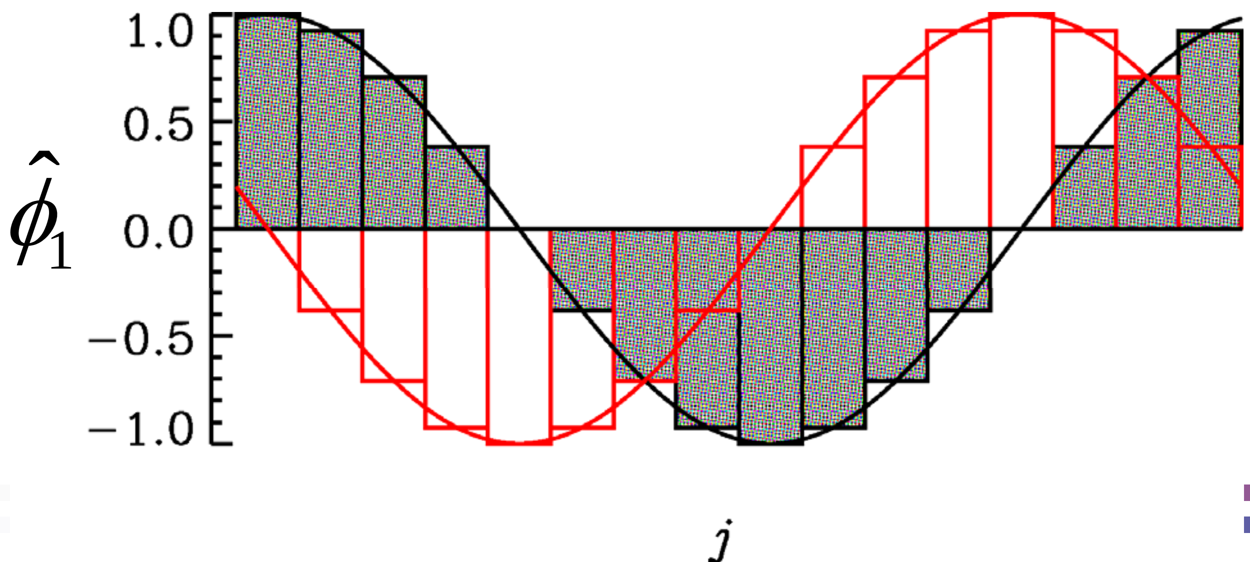
Fourier Transform Yields the *Spectrum*

- Spectrum of Sound



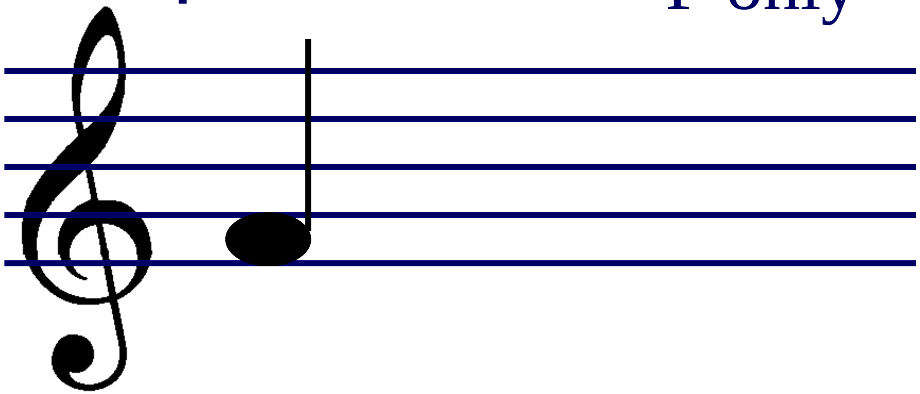
Why?

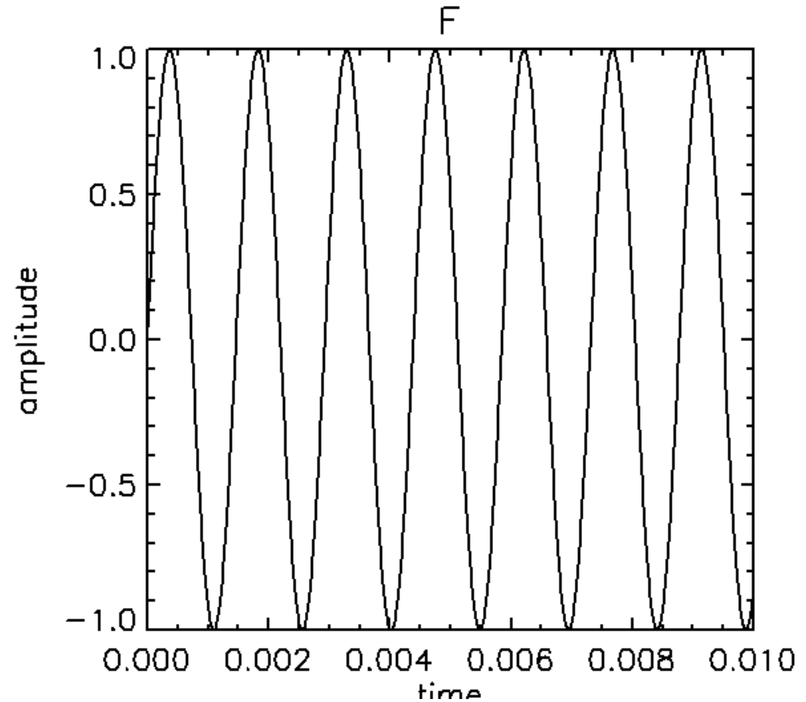
- Each Fourier Basis vector is a waveform of a different frequency
- Finding the components of frequency that make up a function is, by definition, taking its spectrum



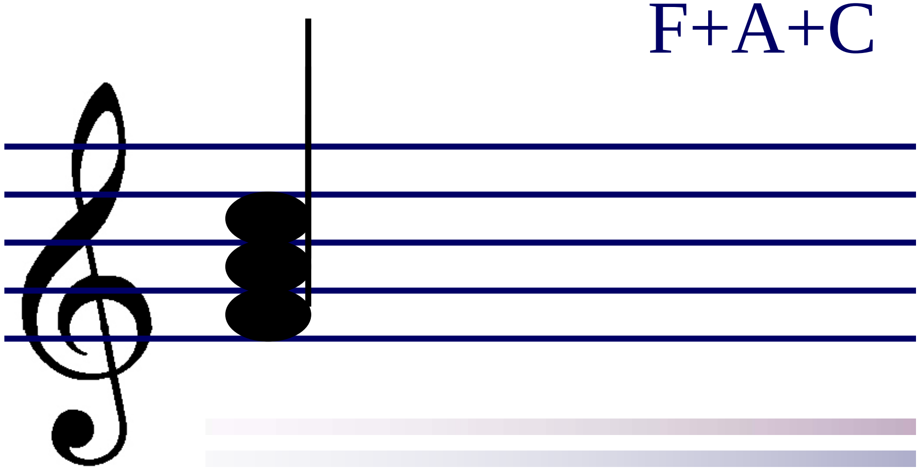
- Musical notes are really Fourier components

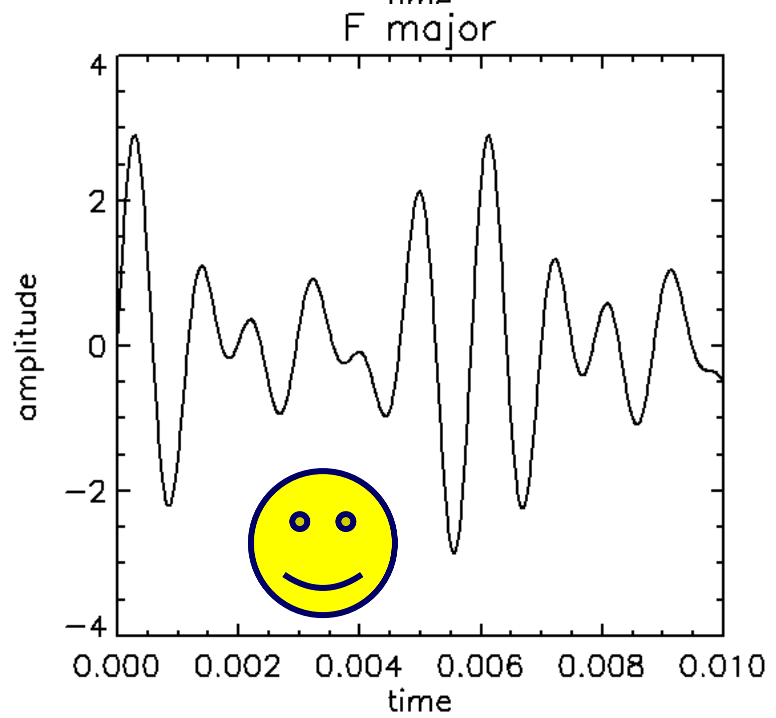
F only





F+A+C





Fourier Derivative

- Reconsider

$$\text{let } k_m = \frac{2\pi m}{N} \leftarrow$$ notational convenience

$$\hat{\phi}_m = \exp\left(\frac{2\pi i m}{N} j\right) = \exp(ik_m j)$$

$$C_m = \mathbf{f} \cdot \exp(-ik_m j)$$

$$f_j = \frac{1}{N}\sum_m C_m \exp(ik_m j)$$

- Let $x = j$, and take x derivative

$$f(x) = f_j = \frac{1}{N}\sum_m C_m \exp(ik_m x)$$

$$\frac{\partial \mathbf{f}}{\partial x} = \frac{\partial}{\partial x}\frac{1}{N}\sum_m C_m \exp(ik_m x)$$

$$= \frac{1}{N}\sum_m ik_m C_m \exp(ik_m x)$$

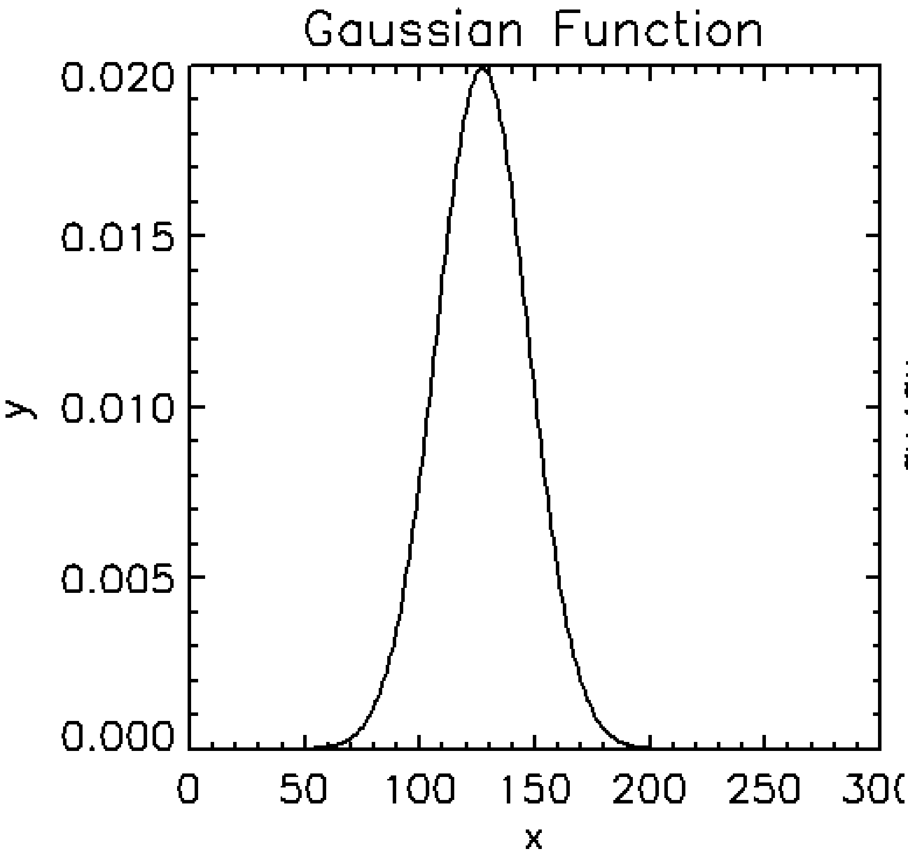
Derivative has become simple multiplication!

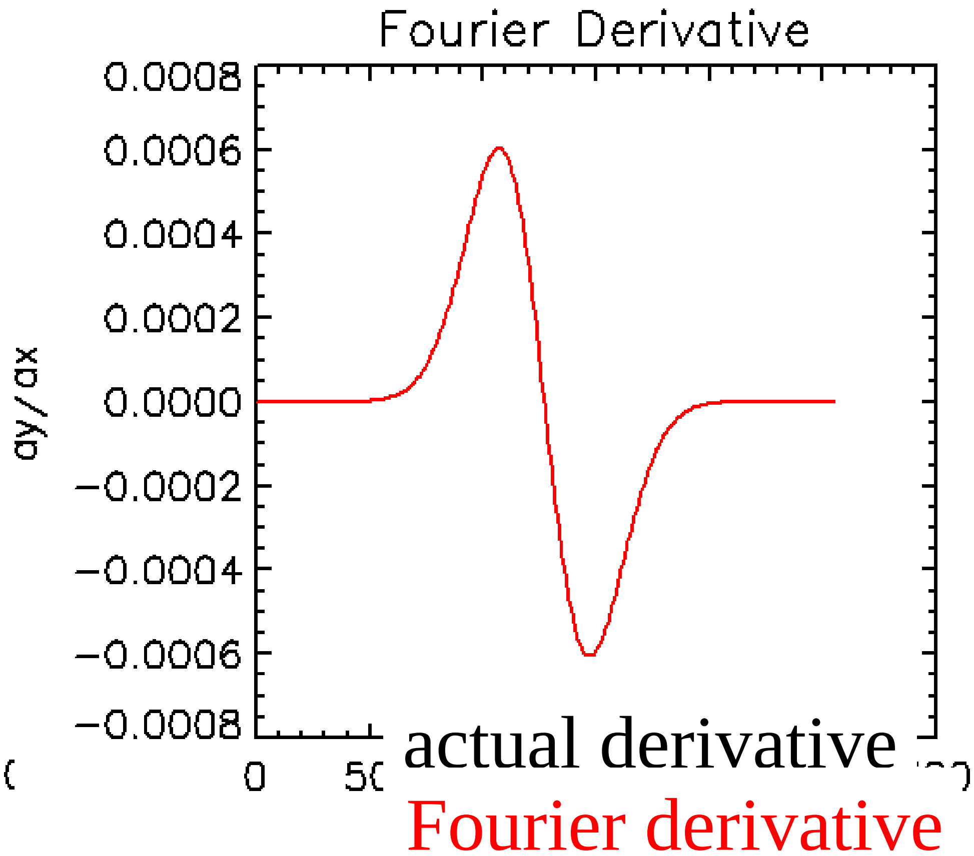
$$\boxed{C_m\left(\frac{\partial \mathbf{f}}{\partial x}\right) \rightarrow ik_m C_m(\mathbf{f})}$$

Fourier Differentiation of a Gaussian

$$y = \frac{1}{\sqrt{2\pi}(20)} \exp\left(-\frac{(x-127)^2}{2\cdot 20^2}\right)$$

$$y' = \frac{1}{\sqrt{2\pi}(20)} \left(-\frac{x}{20^2}\right) \exp\left(-\frac{(x-127)^2}{2\cdot 20^2}\right)$$



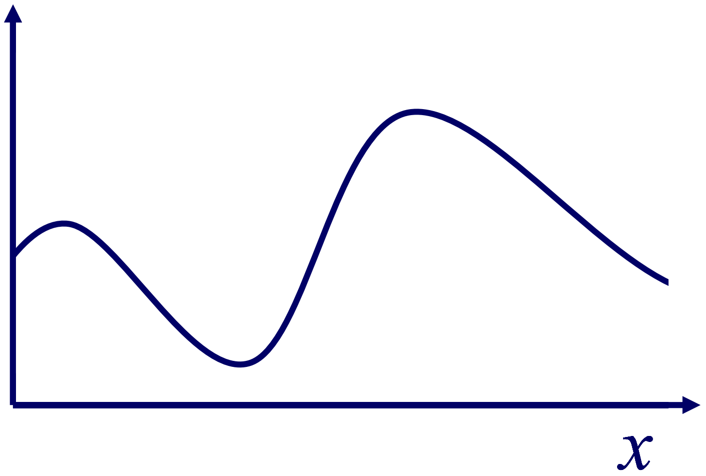


Fourier and Partial Diff Eqs

- A wave is a traveling function

$$F(x,t) = f(x - vt)$$

v is the velocity of the wave



Fourier Wave propagation

$$F(x,t) = f(x - vt)$$

$$\frac{\partial F}{\partial x} = -\frac{1}{v}\frac{\partial F}{\partial t}$$

- Consider coefficients as time-dependent

$$F = \frac{1}{N}\sum_{m=0}^{\infty} C_m(t)\exp(ik_m x)$$

$$\frac{\partial F}{\partial x} = \frac{1}{N}\sum_{m=0}^{\infty} ik_m C_m(t)\exp(ik_m x) = -\frac{1}{v}\frac{\partial F}{\partial t}$$

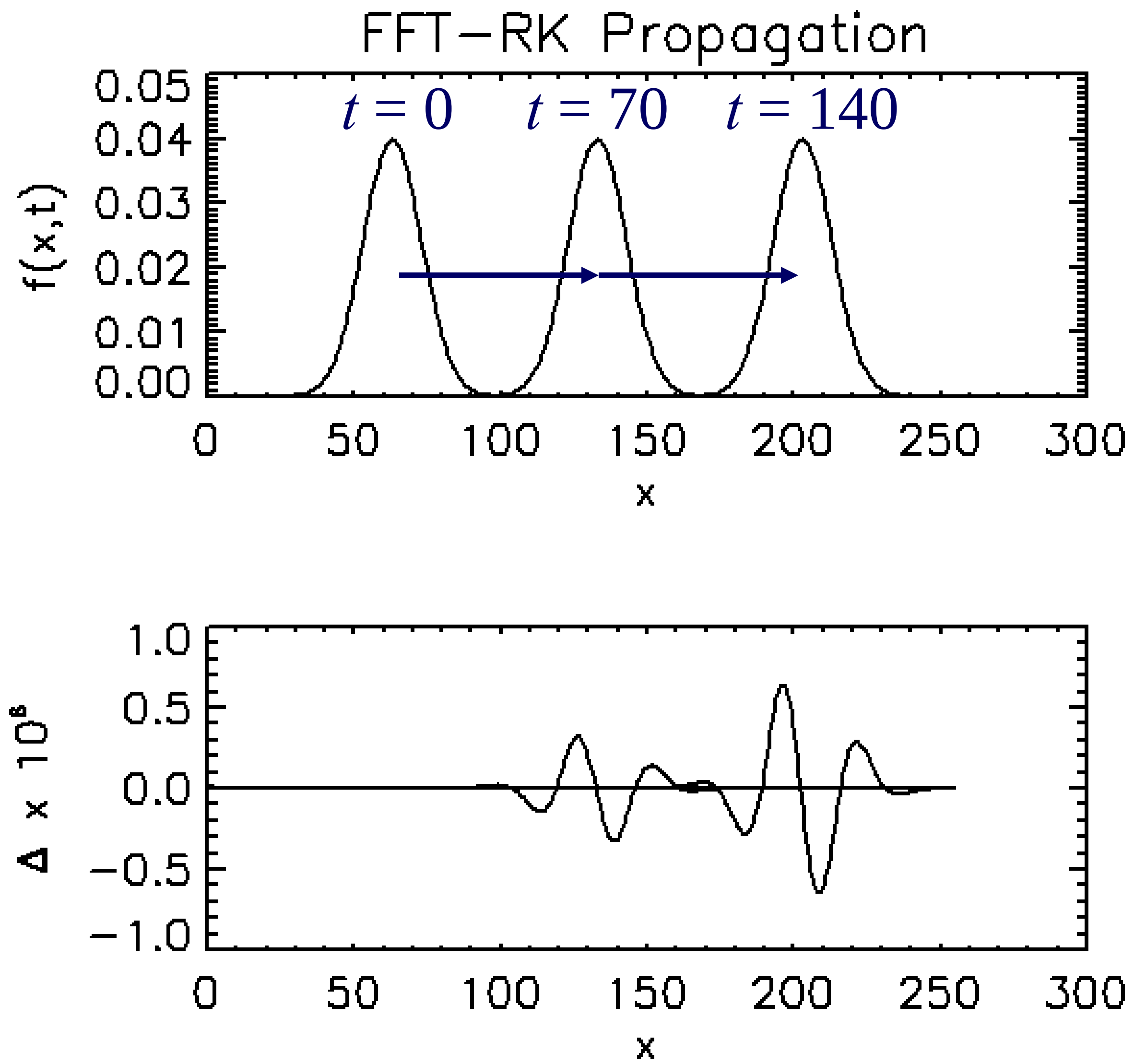
$$\boxed{\frac{\partial C_m}{\partial t} = -ik_m v C_m(t)}$$

Use ODE integrator to propagate C_m's

Gaussian wave, FFT propagation

- speed is 1
- Use RK4 steps (with FFT spatial derivatives)
- 200 iterations with Δt = 0.7sec

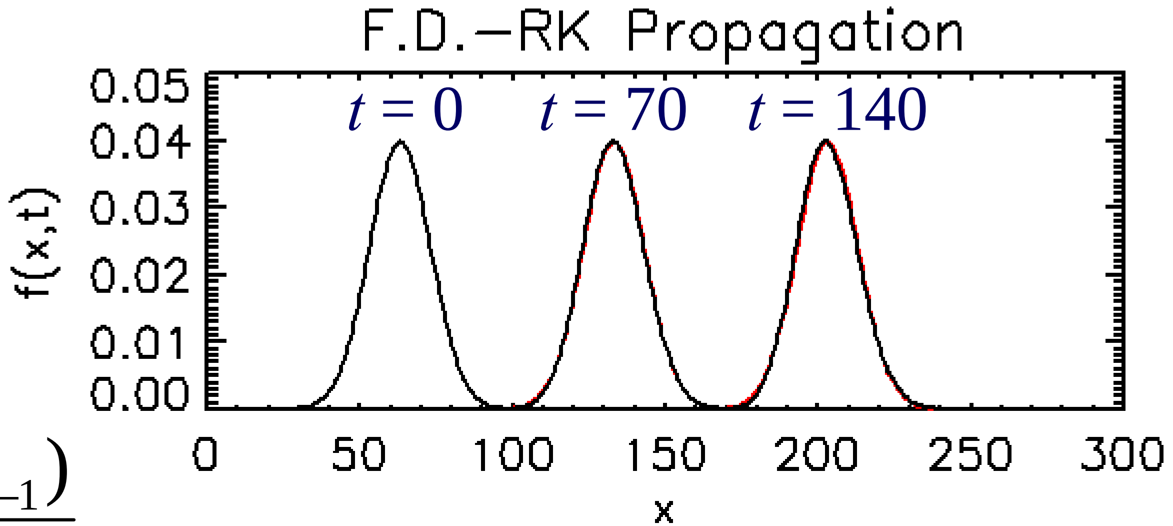
That's 10^6

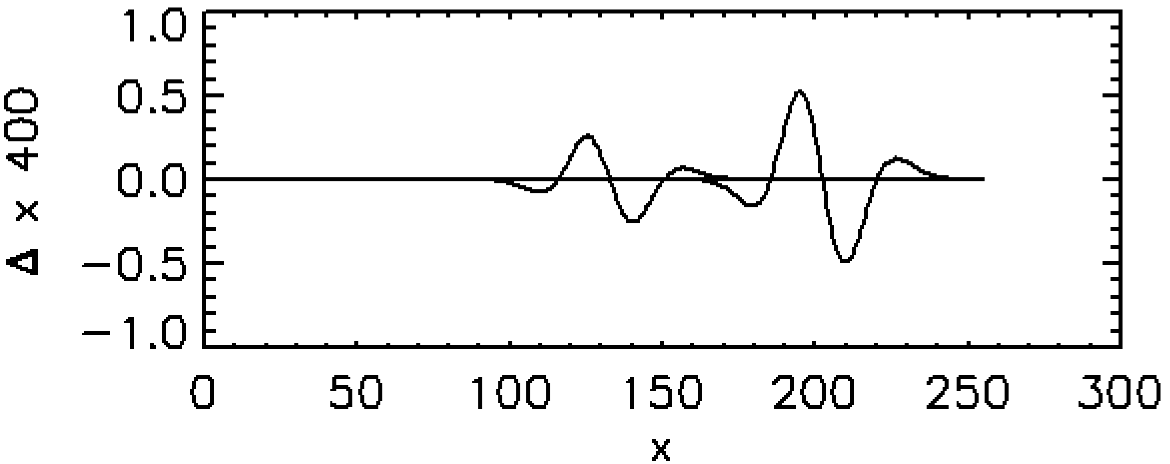


Gaussian wave, Finite Difference

- Use finite-difference steps

$$\left.\frac{\partial f}{\partial x}\right|_{x=x_i} = \frac{f(x_{i+1}) - f(x_{i-1})}{2\Delta x}$$

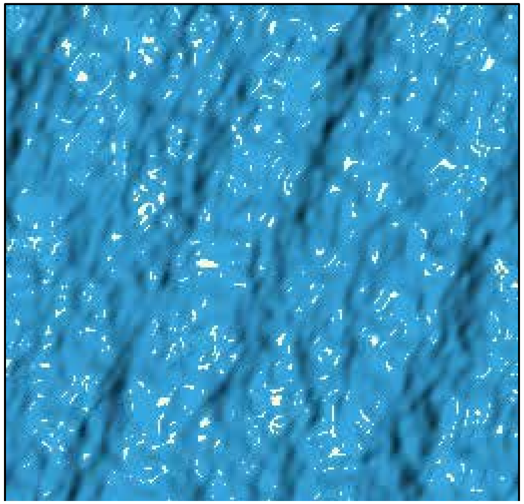




MUCH better performance by Fourier method

Cooler example

- Water waves
  - wave propagation speed proportional to square-root of wavelength = $2\pi/k$
  - all wave propagation in Fourier space
  - fancy shadows and reflections from basic geometry in real-space



movie runs about 3× calculation speed on PC…
FFTs are FAST!

Combined Wave and Diffusion

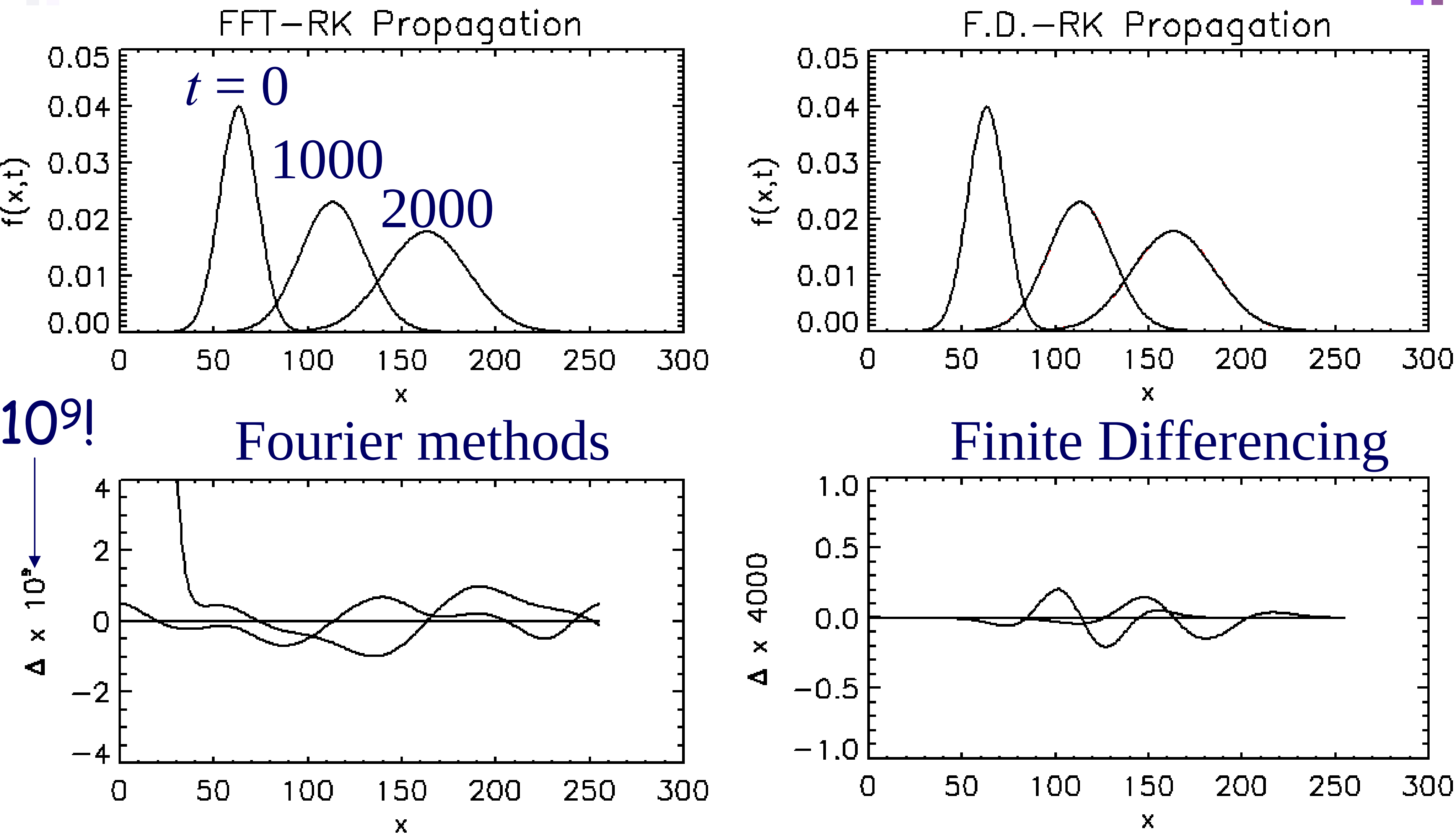
- Terms in first and second order spatial derivatives

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2} - v\frac{\partial f}{\partial x}$$

- In real media, waves typically diffuse due to friction or viscosity
  - Gaussian solution

$$f = \frac{1}{\sqrt{4\pi Dt}}\exp\left(-\frac{(x-vt)^2}{4Dt}\right)$$

Simulation results, v=0.1, D=0.2



Fourier in PDEs

- Often much more accurate that finite differencing
  - uses information from all points, not just two
- Still very fast
  - $N \log N$ operation

Convolution

- Convolution is usually a method of smoothing
  - can be used for filtering and unsmoothing
- Convolving $f(x)$ with $g(x)$ is accomplished thus

$$f(x) \otimes g(x) = \int_{y_0}^{y_1} g(y) f(x-y) dy$$

Typical Example

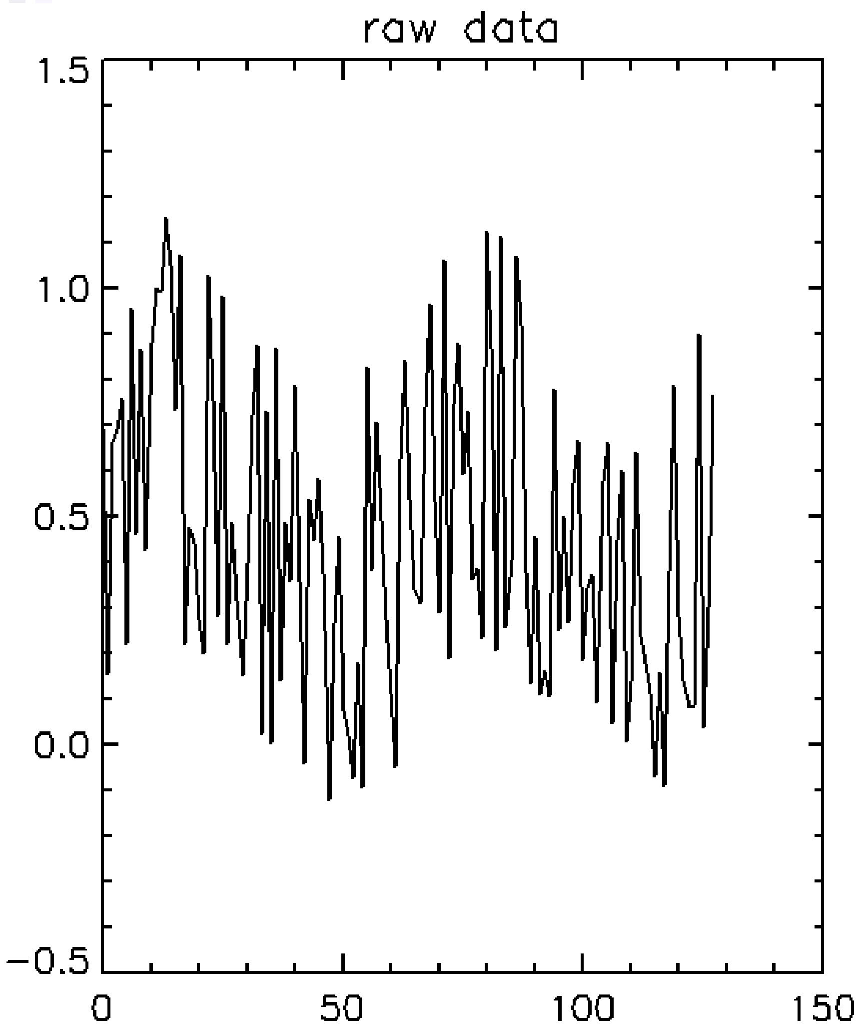
- Smooth with a Gaussian Function

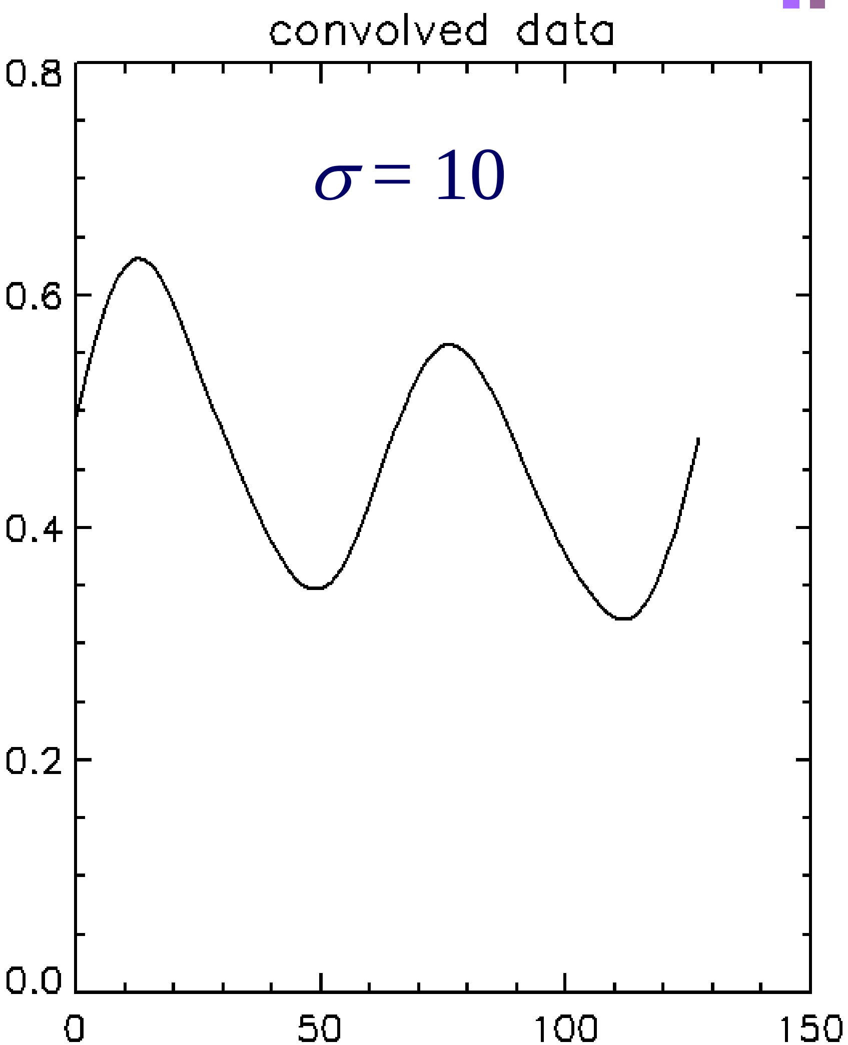
$$f(x) \otimes g(x) = \frac{1}{\sqrt{2\pi\sigma}} \int_{-\infty}^{\infty} \exp\left(-\frac{y^2}{2\sigma^2} \right) f(x-y)\,dy$$

- This smooths over features smaller than sigma, leaving only the long wavelength, smoother components

Typical Example

- Smooth with a Gaussian Function





Discrete Convolution

$$f(x) \otimes g(x) = \int_{y_0}^{y_1} g(y) f(x-y)\,dy$$

$$\left(f \otimes g\right)_j = \sum_{k=0}^{N-1} g_k f_{(j-k)}$$

- By definition, an N^2 process
  - each of N elements of the convolution requires a sum over N terms

$g(x)$ is the "convolution kernel"

Fourier Convolution

- Continuous limit: *N* becomes very large
  - Vectors become continuous functions
  - Dot products become integrals

$$\phi(k,x) = \exp(ikx)$$

$$\tilde{f}(k) = \int f(x)\exp(-ikx)dx$$

$$f(x) = \frac{1}{2\pi}\int \tilde{f}(k)\exp(ikx)dk$$

$$\frac{2\pi m}{N} = k_m \to k$$

$$j \to x$$

$$f_j \to f(x)$$

$$C_m \to \tilde{f}(k)$$

Fourier Convolution

- Convolution is simply multiplication in Fourier space, STILL $N \log N$!

$$
\begin{aligned}
f \otimes g &= \int_y g(y) f(x-y)\,dy \\
&= \int_y g(y) \int_k \tilde{f}(k) \exp\big(ik(x-y)\big)\,dk\,dy \\
&= \int_k \tilde{f}(k) \exp(ikx) \int_y g(y) \exp\big(-iky\big)\,dy\,dk \\
&= \int_k \tilde{f}(k) \exp(ikx)\, \tilde{g}(k)\,dk
\end{aligned}
$$

$$
\boxed{f \otimes g = \int_k \tilde{f}(k)\tilde{g}(k) \exp(ikx)\,dk}
$$

One Last Trick

(complete the square)

- Fourier Transform of a Gaussian is a Gaussian with $\sigma_k = 1/\sigma$

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2} \right)$$

$$\tilde{g}(k) = \frac{1}{\sqrt{2\pi}\sigma} \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp\left(-\frac{x^2}{2\sigma^2} \right) \exp(-ikx)\, dx$$

$$= \frac{1}{(2\pi)^{3/2}\sigma} \int_{-\infty}^{\infty} \exp\left(-\frac{x^2}{2\sigma^2} - ikx \right) dx$$

$$-\frac{x^2}{2\sigma^2} - ikx = -\frac{1}{2\sigma^2}\left(x^2 + 2\sigma^2 ikx \right)$$

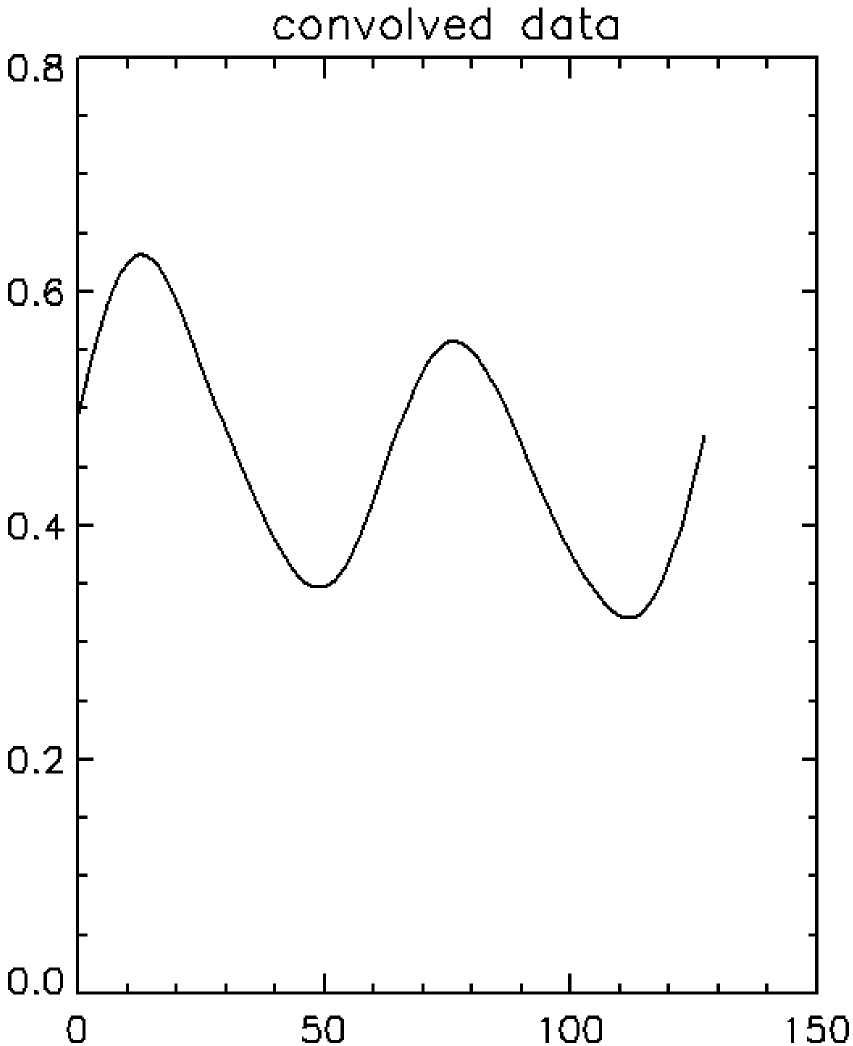
$$= -\frac{1}{2\sigma^2}\left[\left(x + \sigma^2 ik \right)^2 + \sigma^4 k^2 \right]$$

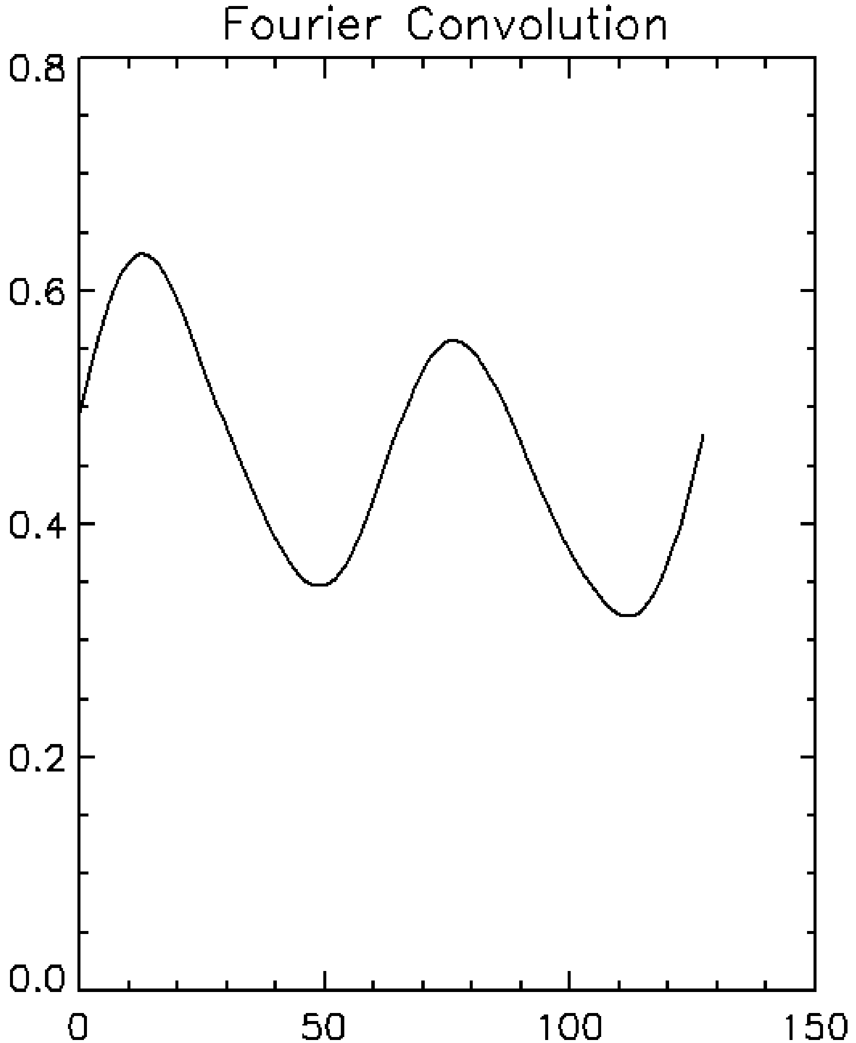
$$= -\frac{1}{2\sigma^2}\left[\left(x + \sigma^2 ik \right)^2 \right] - \frac{\sigma^2 k^2}{2}$$

$$\tilde{g}(k) = \frac{1}{(2\pi)^{3/2}\sigma} \exp\left(-\frac{\sigma^2 k^2}{2} \right) \int_{-\infty}^{\infty} \exp\left(-\frac{(x + \sigma^2 ik)^2}{2\sigma^2} \right) dx$$

$$= \text{constant} \times \exp\left(-\frac{\sigma^2 k^2}{2} \right)$$

Fourier Convolution





Again, Advantage Fourier

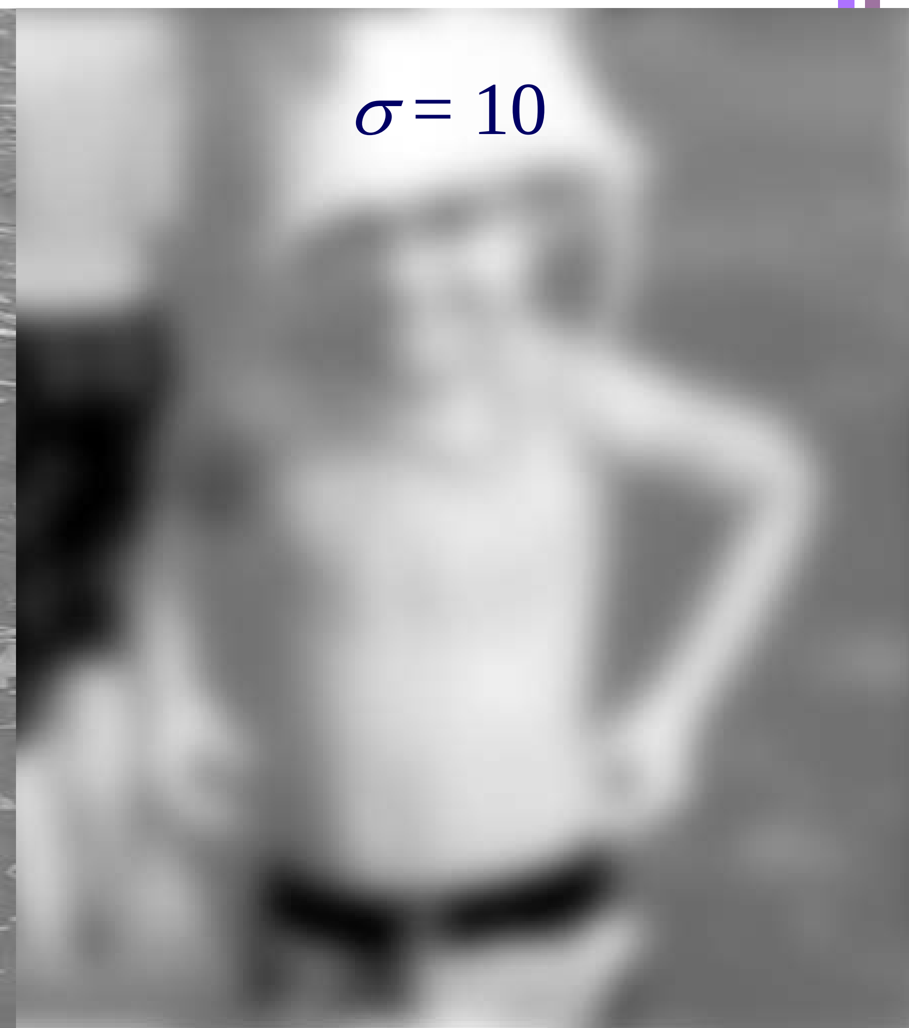
- Fourier Convolution happens in $N \log N$ time, not N^2 time.
- Becomes very important at large N.

2-D Convolution: Boxcar Smoothing

- Average all pixels in an *n* x *n* box

2-D Gaussian Smoothing

- Same math as in 1-D



Edge Detection

- Edges have large gradients
  - cliffs are steep
- Search for gradients by taking advantage of Fourier derivative = multiplication

$$\frac{df}{dx} \otimes g = \int \frac{d}{dx} \tilde{f}(k)\tilde{g}(k)\exp(ikx)dk$$

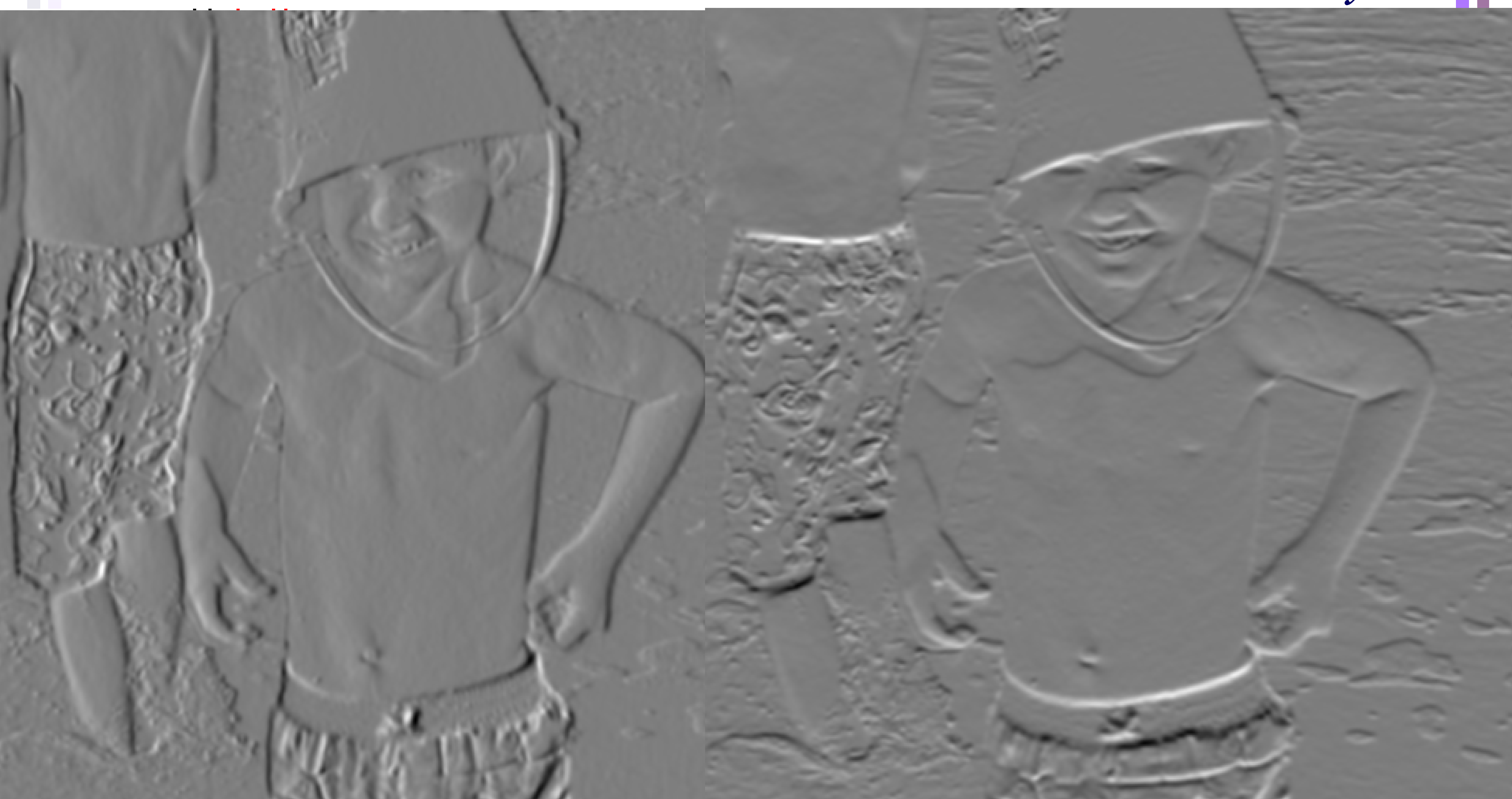
$$= \int ik\tilde{f}(k)\tilde{g}(k)\exp(ikx)dk$$

Fourier Derivative in continuous space: $ik_m \rightarrow ik$

Edge Detection

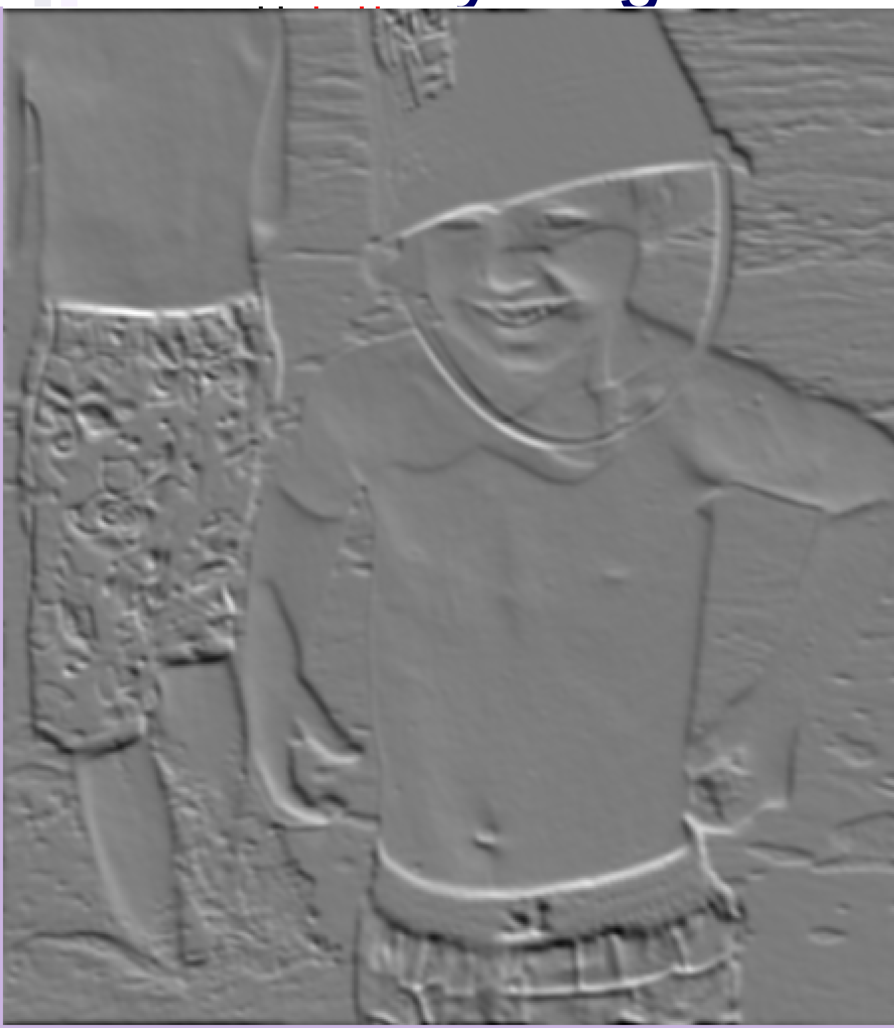
- Can look for gradients at any angle

x y



Edge Detection

- 60^o gradient edge, and some of squares of x & y edges

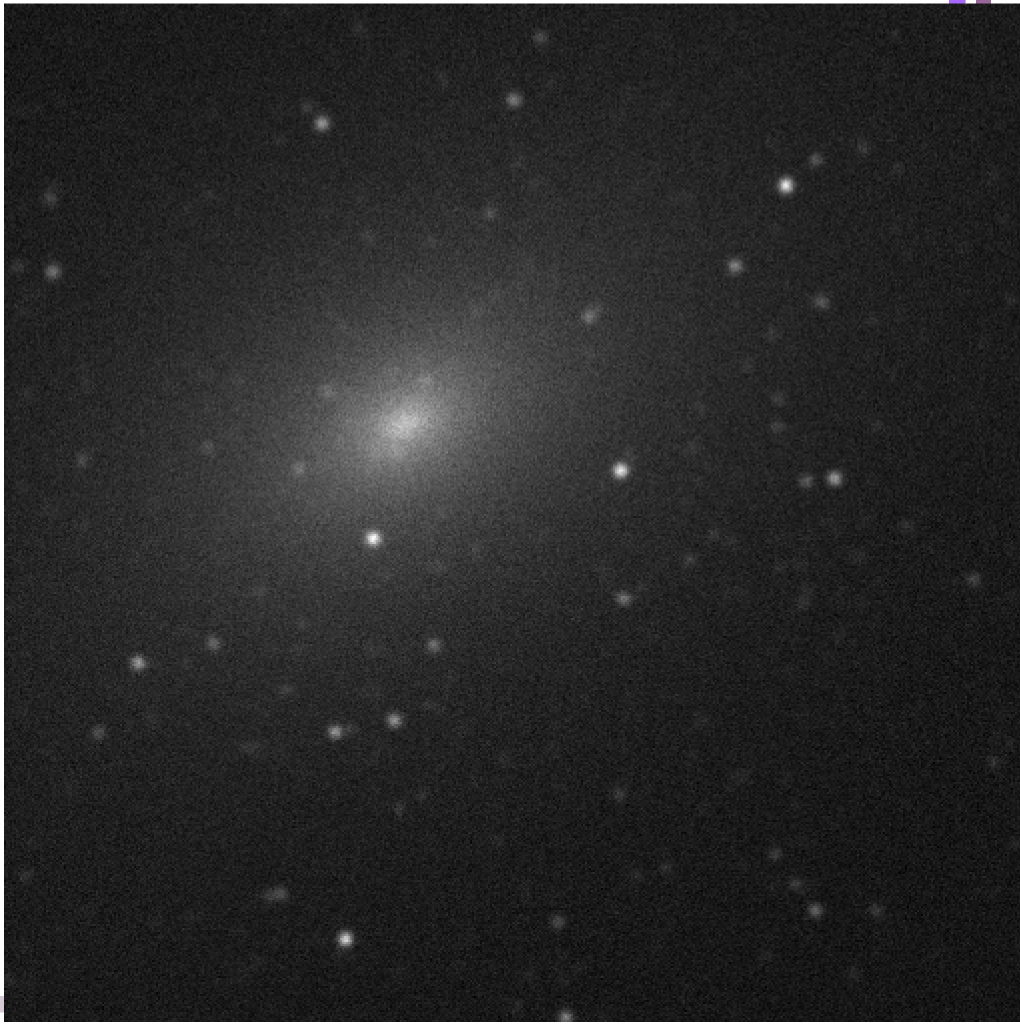


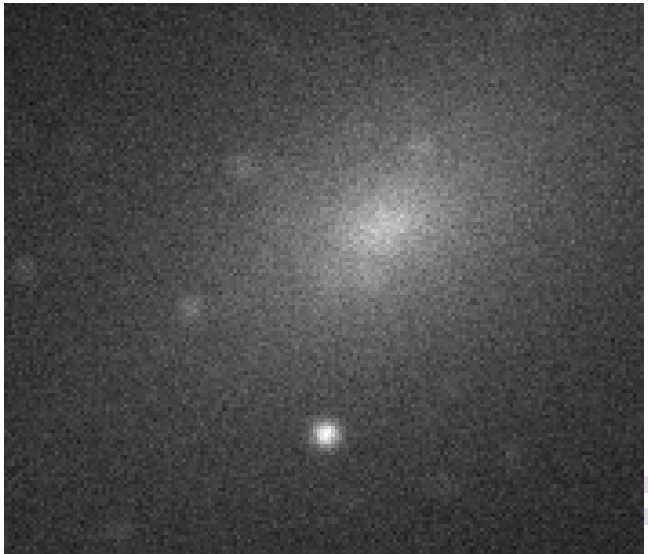
Matched Filtering

- Some applications require enhancement of modes only in a particular band = (attenuation of other bands)
  - high- and low- pass filter, like one column on a spectrum analyzer
- In image processing, source location is a biggie

Fake astronomical image

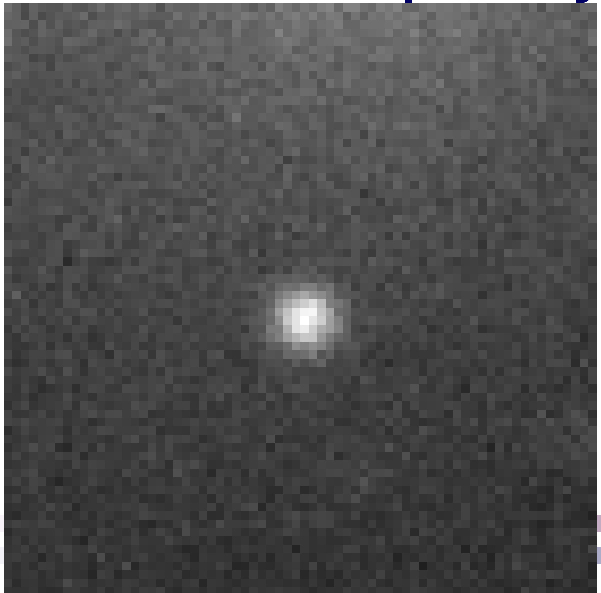
- Fairly typical galaxy in fairly typical star-field
- realistic noise added

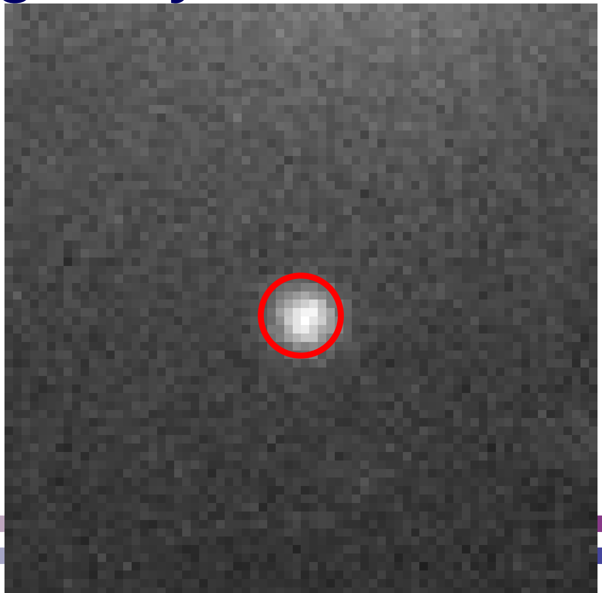




Want to find stars

- Use matched filter – select frequencies that correspond to the stars' "point spread function"
  - p.s.f. arises from blurring by atmosphere
  - remove high-frequency noise
  - remove low-frequency galaxy

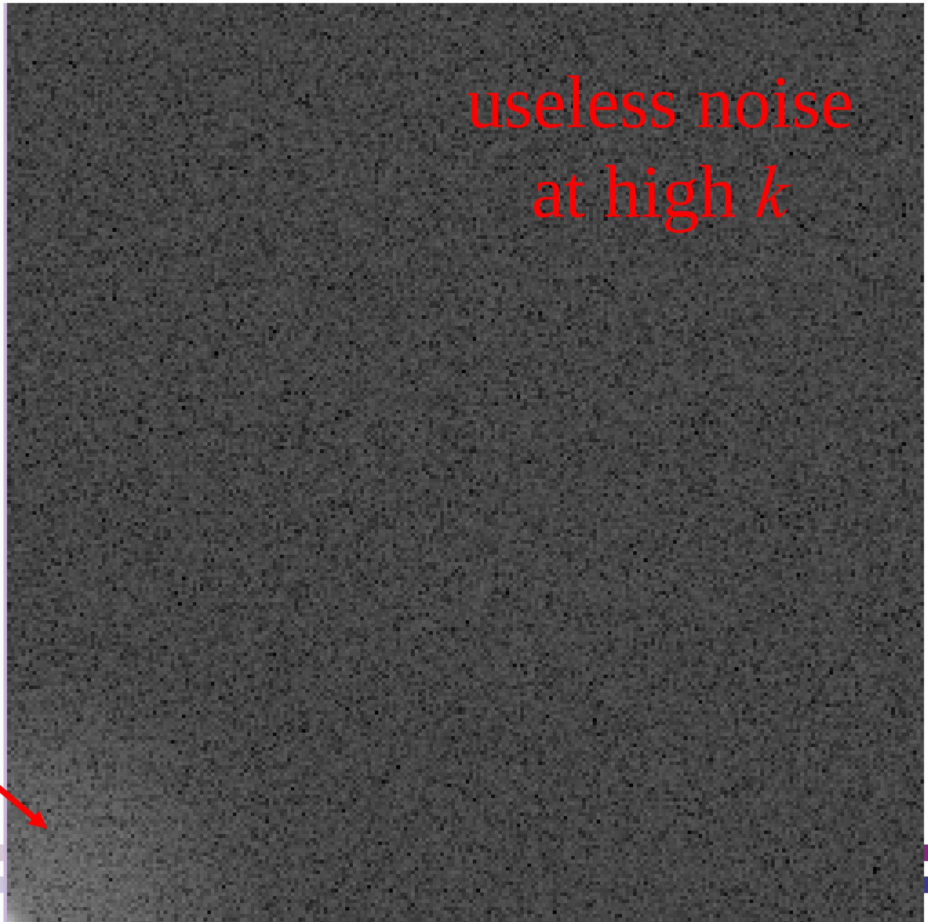




Power Spectrum of image

- The power spectrum show what needs to happen

stellar p.s.f.
at moderate *k*



supress high k

$$\tilde{g}(k) = \exp\left(-\frac{k^2\sigma_{\rm psf}^2}{2}\right)$$

First smoothing

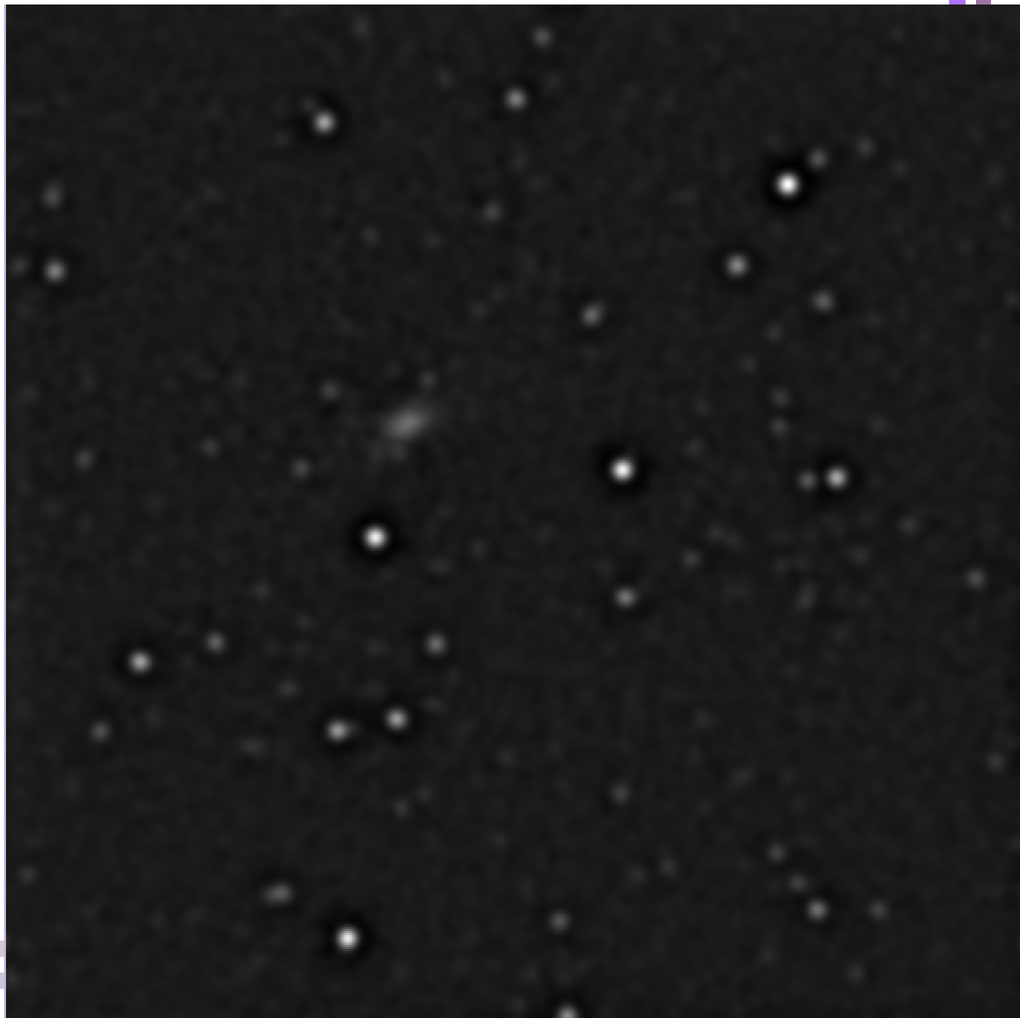
- Removes high-*k* noise
  - greatly enhances large, smooth galaxy
  - remember higher *k* modes are shorter waves = smaller, choppier structures
- enhances stars by removing noise, but not relative to galaxy

supress high k supress low k

$$\tilde{g}(k) = \exp\left(-\frac{k^2\sigma_{\mathrm{psf}}^2}{2}\right) - \exp\left(-\frac{9k^2\sigma_{\mathrm{psf}}^2}{2}\right)$$

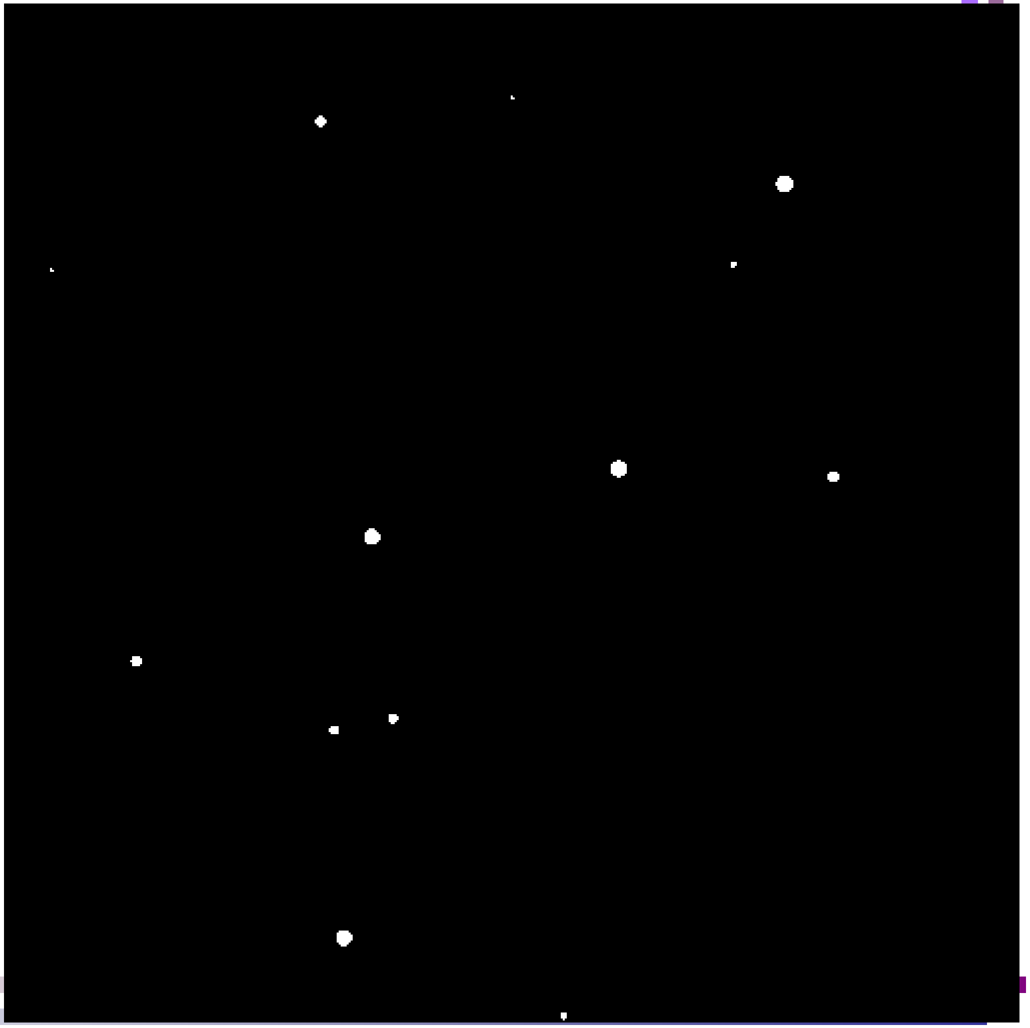
Matched filter

- Filter out very low *k* bands as well
  - low *k* is long wavelength = large, smooth structures
  - galaxy now largely removed
  - stars greatly enhanced



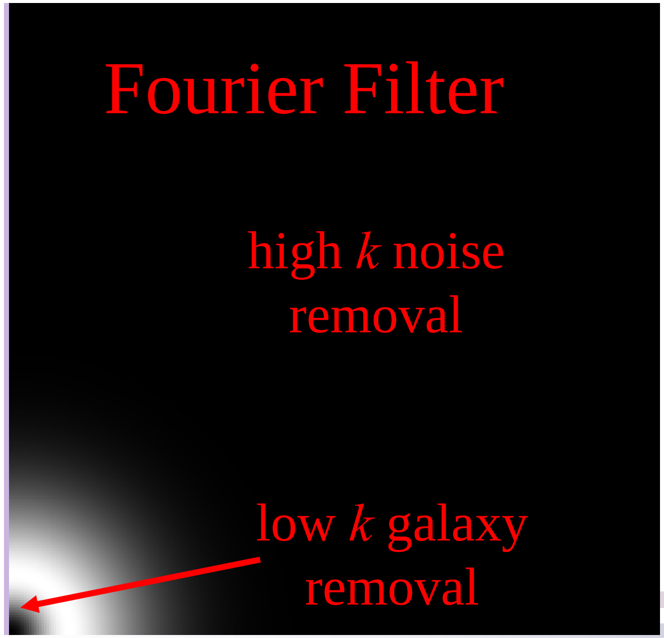
To find sources, use a threshold

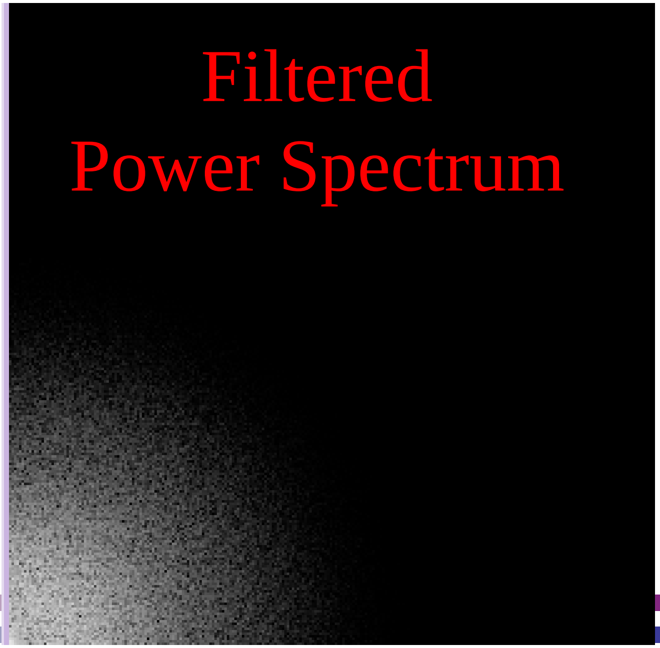
- Look at pixels only above a certain value
  - stars pop right out



What we did, in Fourier Space

- Removed high *k* noise by multiplying by Gaussian
- Removed low *k* structure by subtracting smaller Gaussian



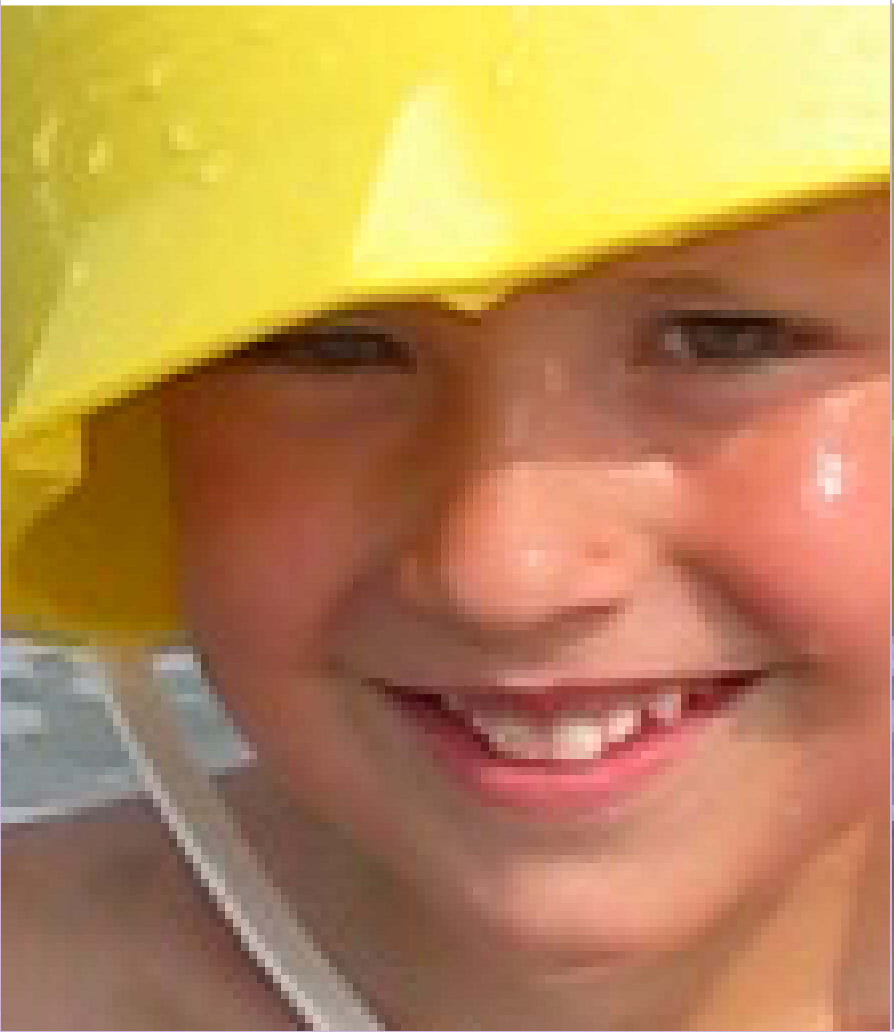


Fourier Image Compression

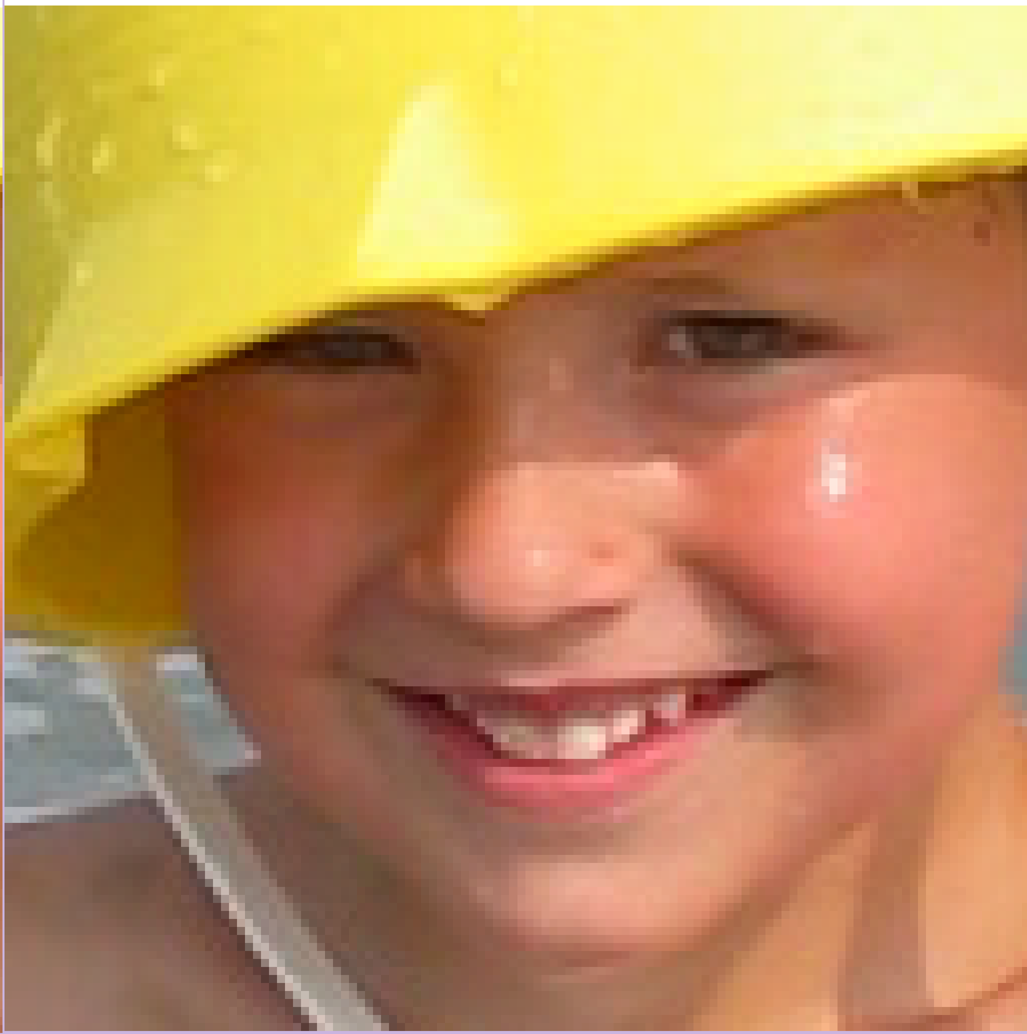
- Frequently real life images have very little power in most of the Fourier modes
  - Throw those modes out entirely, and use only the important ones
- Compute power spectrum
  - sort power spectrum; keep only some fraction of the most important modes

Fourier Compression

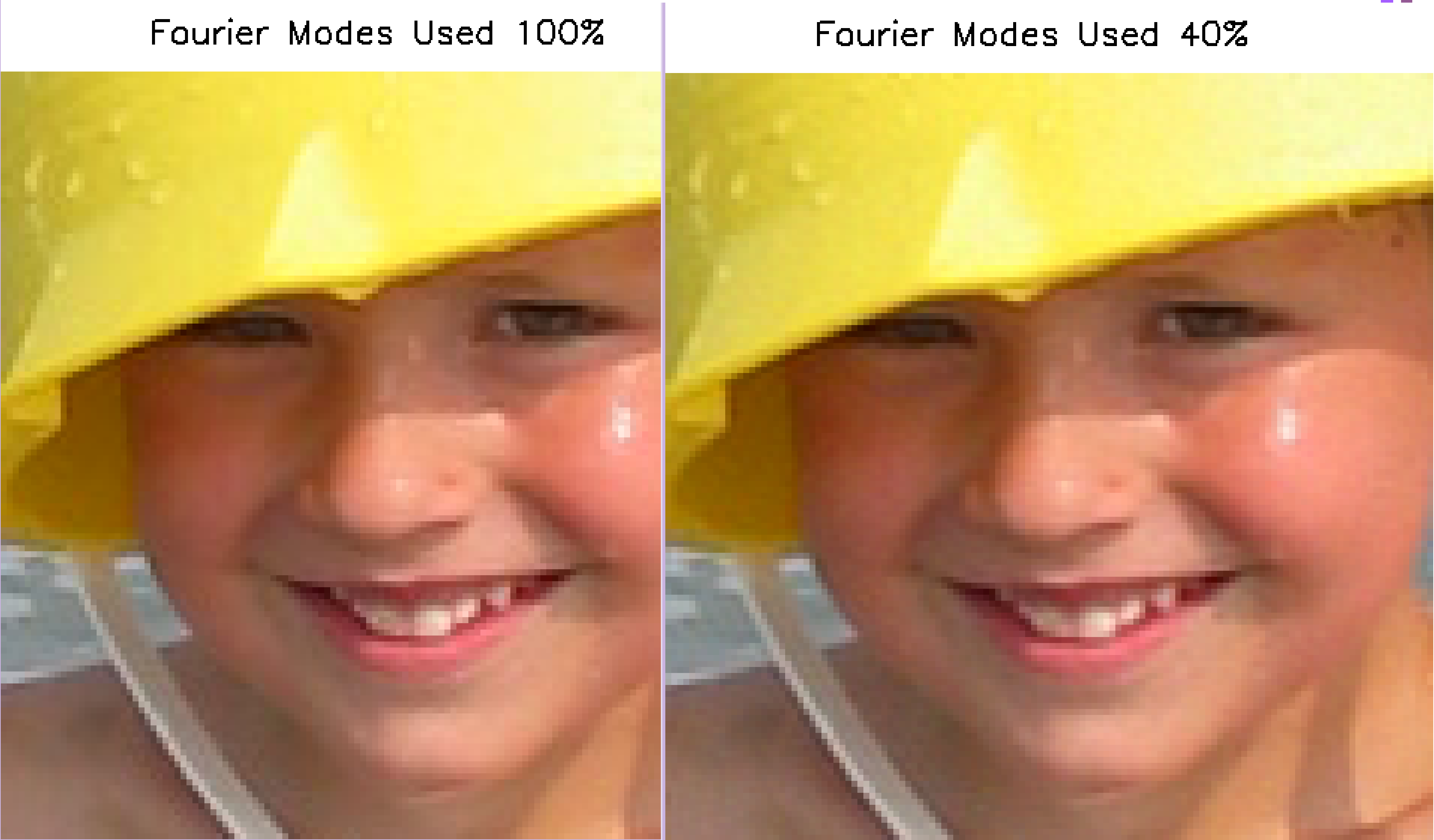
Fourier Modes Used 100%



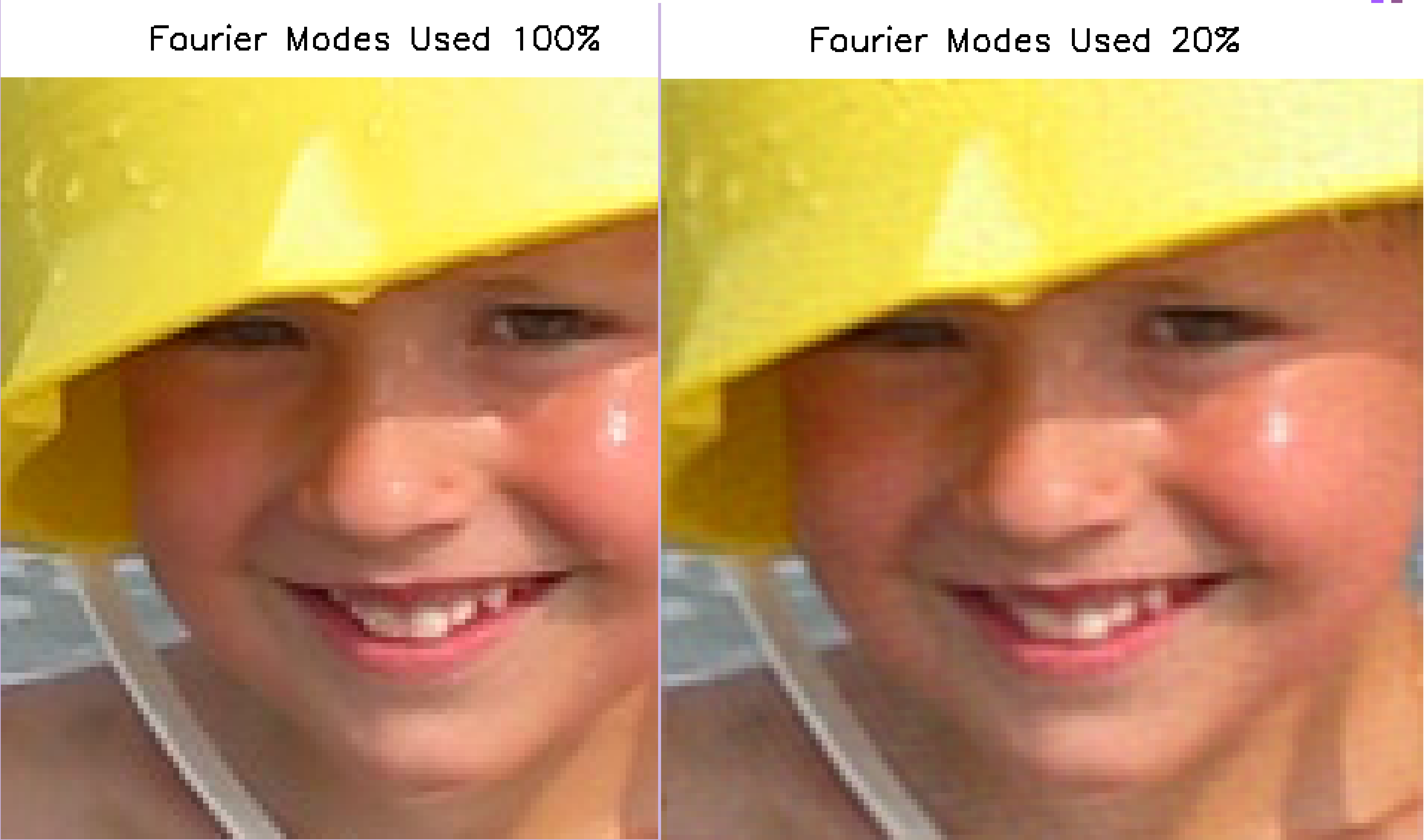
Fourier Modes Used 60%



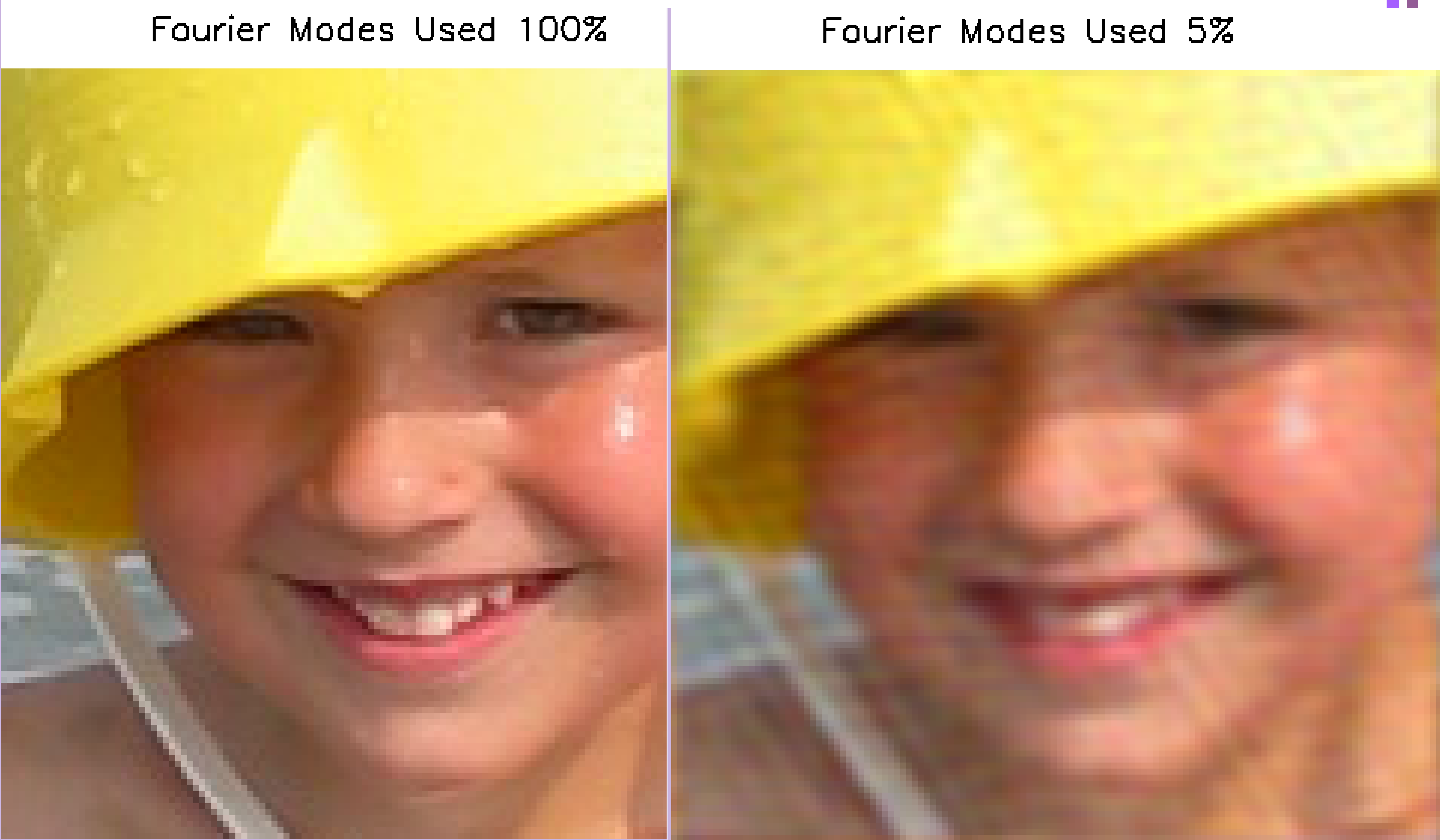
Fourier Compression – 40% acceptable



Fourier Compression – 20% marginal



Fourier Compression – 5% blurry



Fourier Compression

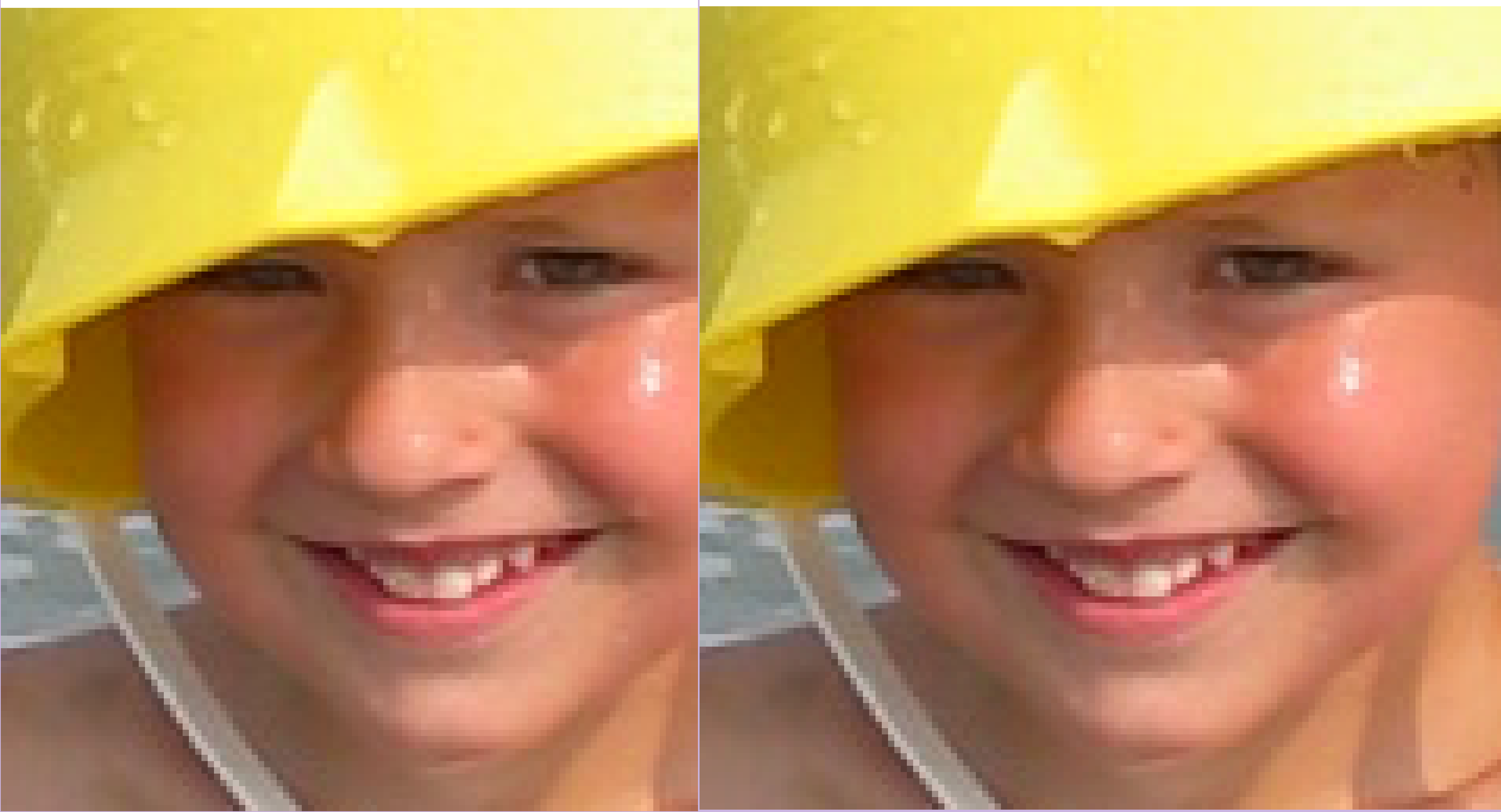
- Large compression factors can be used with acceptable maintenance of image quality
  - 20% is probably acceptable if not zoomed in so tightly
- Caveat
  - storage of WHICH Fourier modes to be used is a factor
  - For a 512x512 image, need at least 18 bits to locate the mode, plus the original 8 bits to give the coefficient of the mode
  - could play other games:
    - a one-bit image of which modes to use
    - run-length encoding of that (ZIP)

JPEG Compression

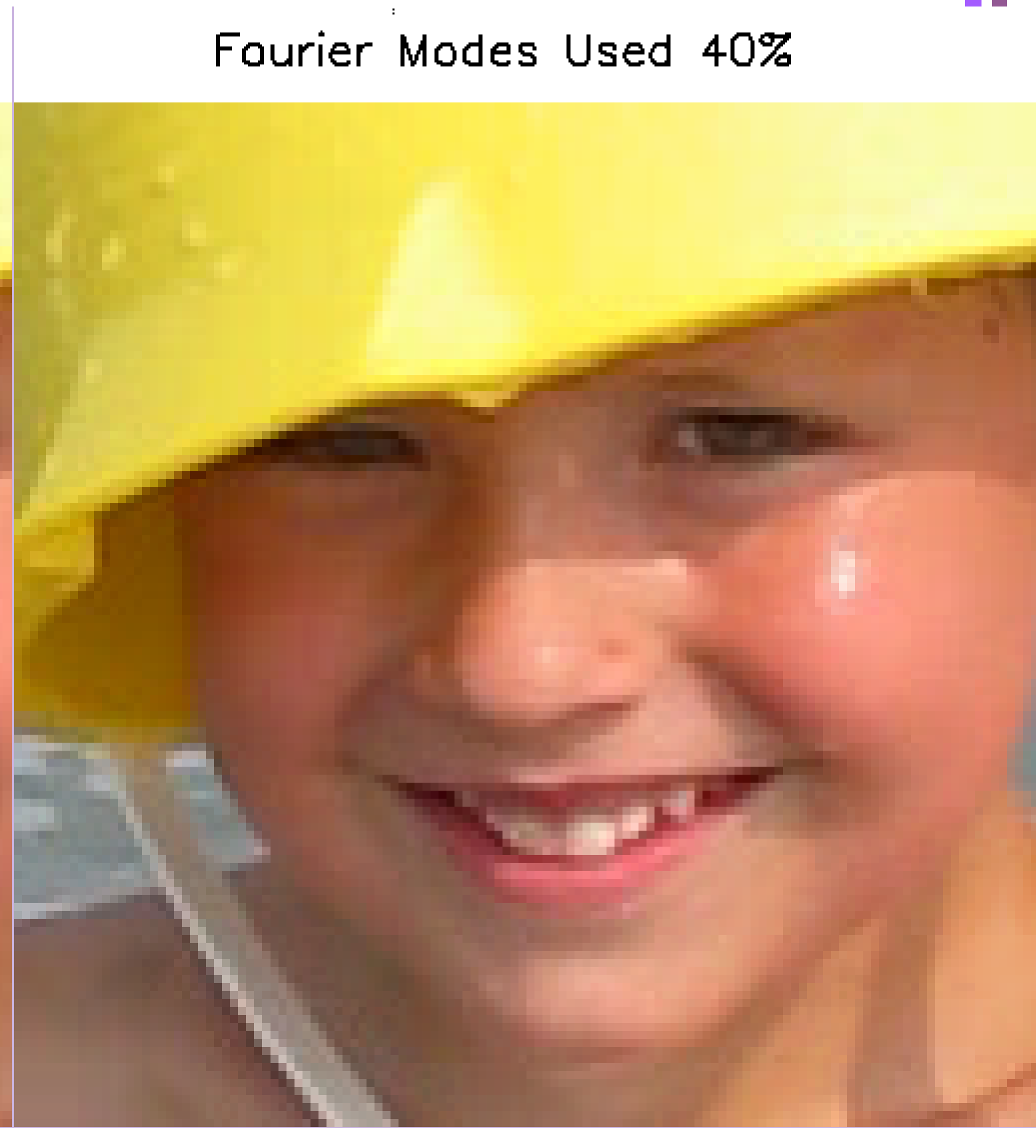
- JPEG is a localized Fourier compression
- 8 x 8 squares are carved out of the image, and Fourier compression is carried out within
  - For boring parts of the image, often only one mode is used (the constant mode)
  - In interesting areas, most modes are used

JPEG-like Compression

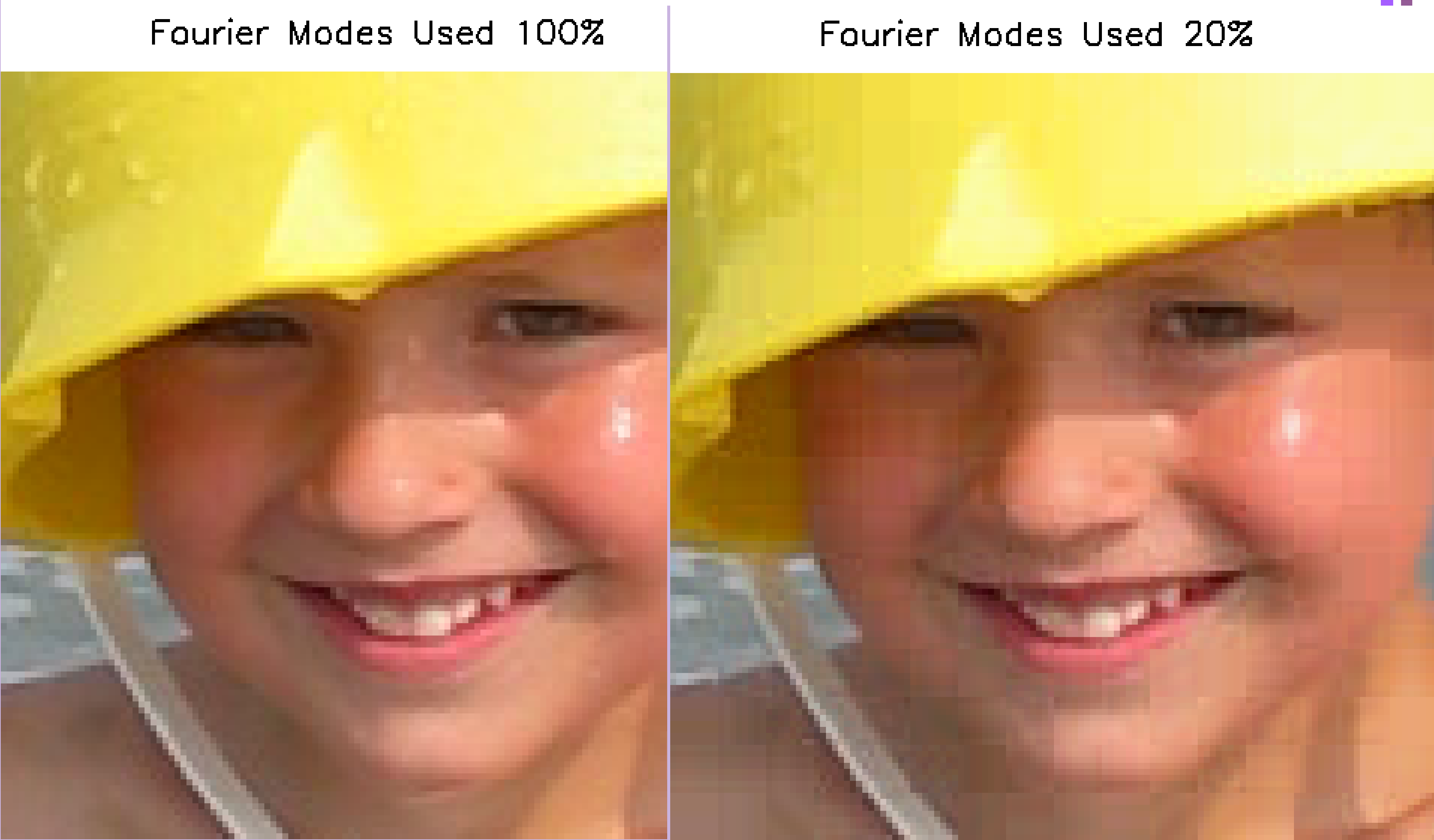
Fourier Modes Used 100%   Fourier Modes Used 60%



JPEG-like Compression



JPEG-like Compression

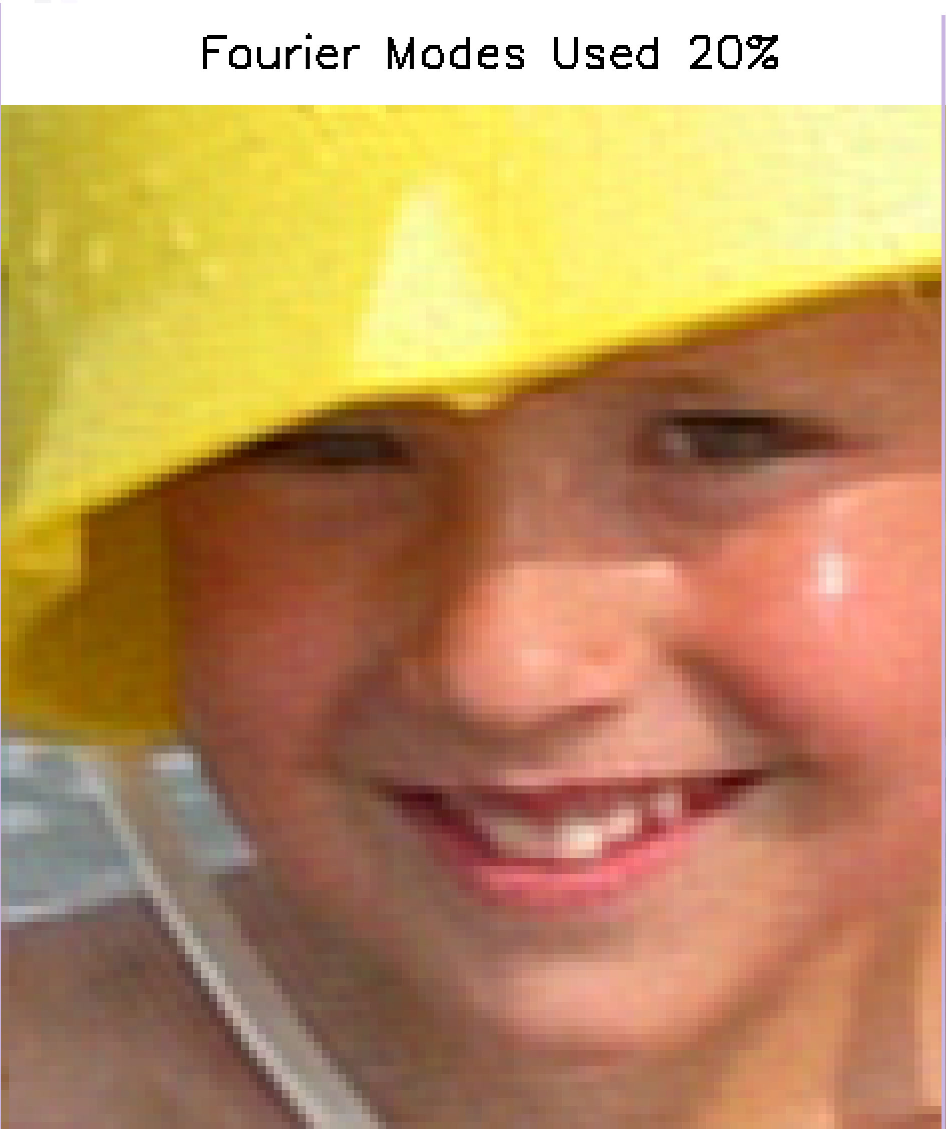


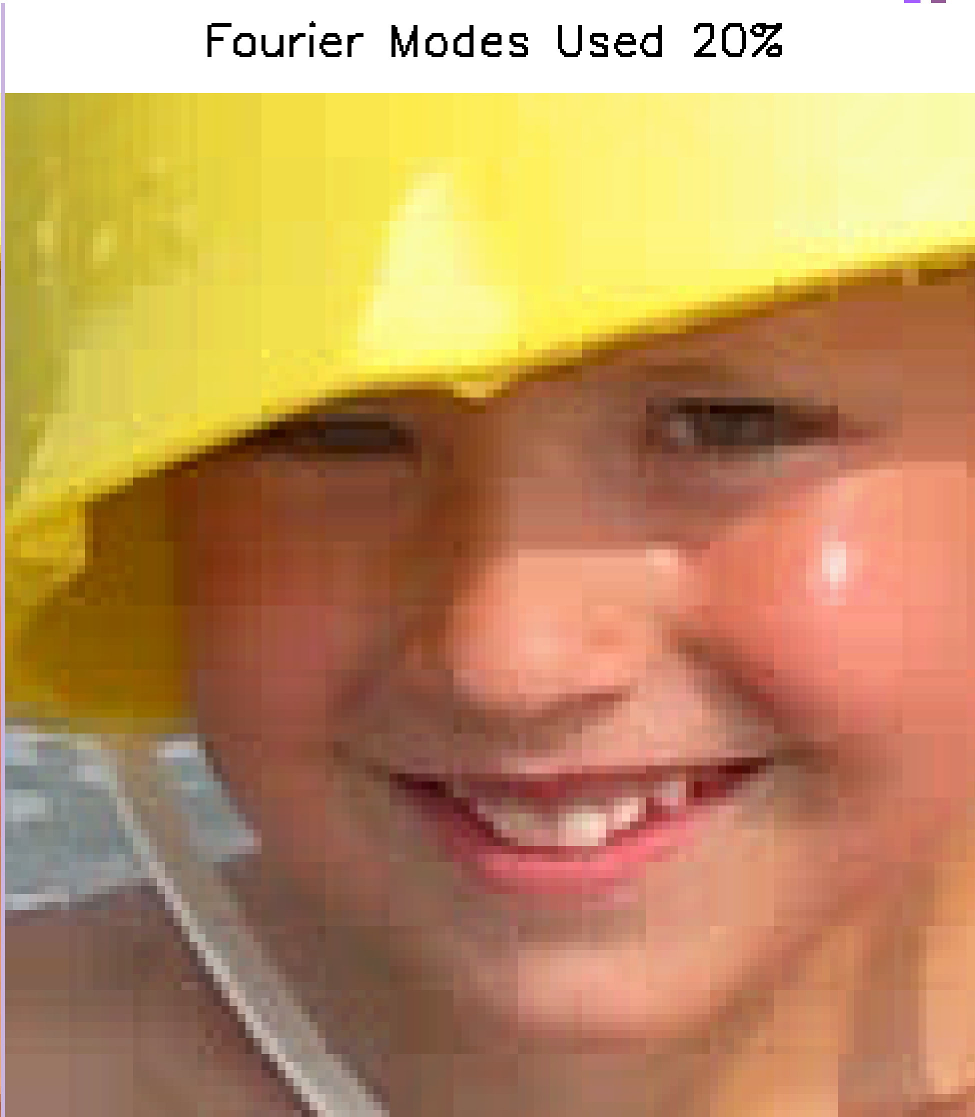
JPEG-like Compression

Fourier Modes Used 100%

Fourier Modes Used 5%

Fourier vs. JPEG





Fourier vs. JPEG

- JPEG preserves more locally sharp features, though pesky square edges start showing up
- Fourier compression loses sharp information, but essentially looks like a smoothed version of original
- Both have some granularity, though JPEG's is confined to particular squares
  - advantage JPEG: annoying behavior confined locally
- Take your pick!

Conclusions

- Physics-based modeling has countless applications in DoD M&S
  - Saw today: missile trajectories, atmospheric effects on sensors
- Using efficient mathematical techniques dramatically enhances compute power
  - Efficient integration algorithms
  - Analytical results
  - Fourier techniques (with spillover into compression and image processing)
- Bottom Line:
  - THINK about the physics
  - Take the time to use appropriate efficient algorithms, and your computer will thank you!