

Optical tomography reconstruction: inversion based on adjoint differentiation

Kenneth M. Hanson

Los Alamos National Laboratory

This presentation available under <http://www.lanl.gov/home/kmh/>

Overview of presentation

- General problem of inversion of complex simulations
- Introduction to optical tomography
- Propagation of IR photons in tissue, a diffusion process
- Simulation of diffusion process by finite-difference method - *“the forward problem”*
- Reconstruction of optical properties using adjoint differentiation - *“the inverse problem”*
- Examples of IR tomographic reconstructions
- Applications & more details about adjoint differentiation

Acknowledgements

- Optical tomography
 - Suhail Saquib, Greg Cunningham
 - Andreas Hielscher, Alexander Klose, David Catarious
- Bayes Inference Engine
 - Greg Cunningham
 - Xavier Battle, Bob McKee
- Applications to hydrodynamics
 - Rudy Henninger, Maria Rightley
- General discussions
 - C. Thacker, J. Skilling, S. Gull, R. Silver, J. Gee, R. Giering, P. Maudlin, L. Margolin, B. Travis, R. Errico, R. Fovell

Inversion of complex simulations

- There are BIG problems that
 - require complex numerical simulations to describe phenomena
 - are nonlinear in nature
 - one would like to fit to data, that is, solve the inverse problem
- Approximations are typically made in forward simulation to facilitate the solution of the inverse problem
 - perturbation methods (Born approximation)
 - truncated basis-function expansion
 - linearization of the problem
 - degradation of the spatial resolution
- Advanced methods are needed to invert large numerical simulations

Inversion of complex simulations

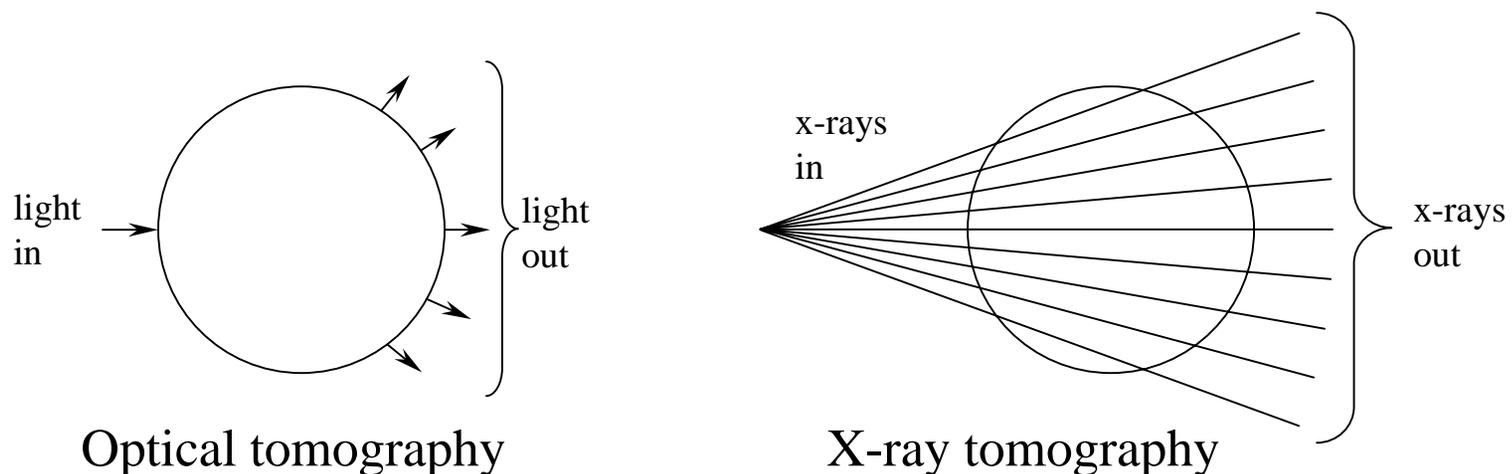
- Advanced techniques are required to cope with large data structures and numerous parameters
 - Optimization
 - gradient-based quasi-Newton methods (e.g., CG, BFGS)
 - adjoint differentiation for efficient calculation of gradients
 - multiscale methods for controlling optimization process
 - Bayesian methods
 - overcome ill posedness of inversion through use of prior knowledge
 - Markov chain Monte Carlo to characterize uncertainties
 - Appropriate higher-order models
 - Markov random fields
 - deformable geometrical models
 - but also consider lowest order, elemental representations

General uses of adjoint differentiation

- Reconstruction: imaging through refractive media
 - seismology, medical and NDE ultrasound, ...
- Matching large-scale simulations to data:
 - atmosphere and ocean models, fluid dynamics, hydrodynamic
- Optimization in large engineering design problems:
 - optical lens systems, geometry of integrated circuits, aerodynamic shape, engines
- Uncertainty analysis
 - sensitivity of uncertainty variance to each contributing cause
- Markov Chain Monte Carlo
 - generation of random samples from a prob. dens. function

Optical tomography - general idea

- Shine light on tissue sample; measure light out

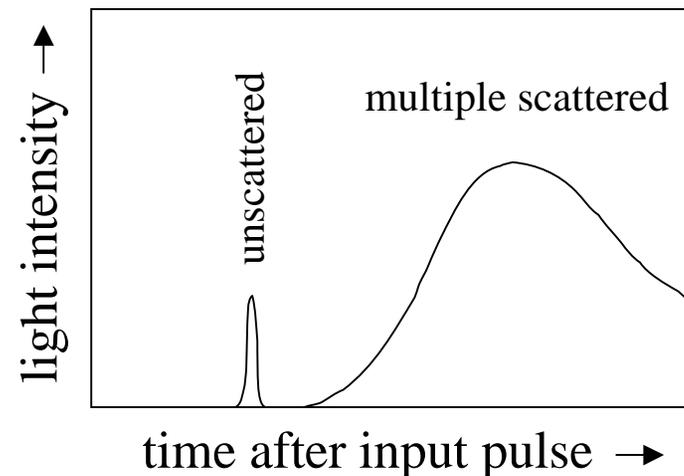
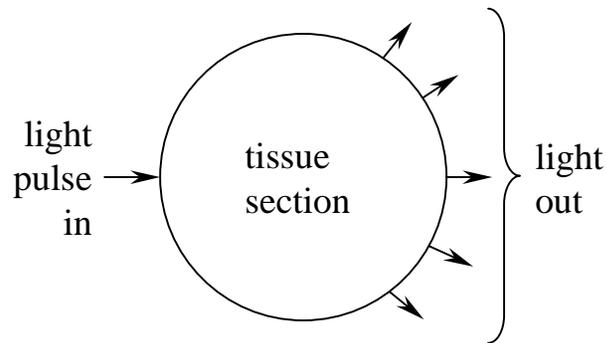


- Analogous to x-ray computed tomography, so:
 - Can one actually do optical tomography?
 - What are best operating conditions?
 - What are imaging properties and diagnostic uses?

Physics of propagation of light in tissue

- Basic processes are scattering and absorption of photons
 - absorption in tissue is minimal in *infrared* range
 - IR photons can actually pass through bone
 - for soft tissue, $\mu_{\text{scat}} \approx 1\text{-}10 \text{ cm}^{-1}$, $\mu_{\text{abs}} \approx 0.1 \text{ cm}^{-1}$
- Transport equation generally applies
- Diffusion equation often good approximation
 - valid when scattering is isotropic and without energy loss

Proposed experimental scheme

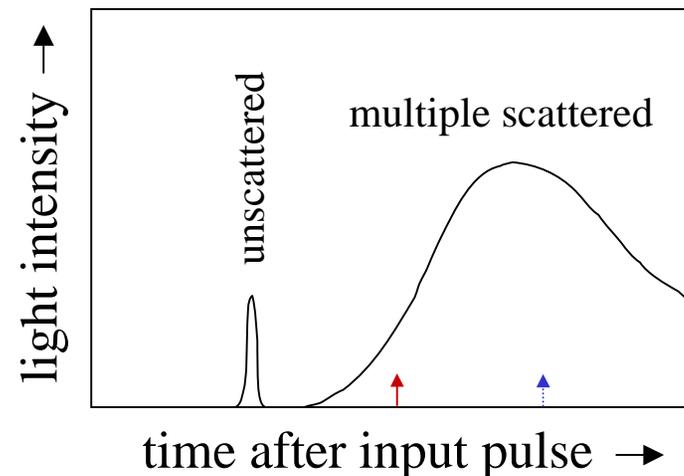
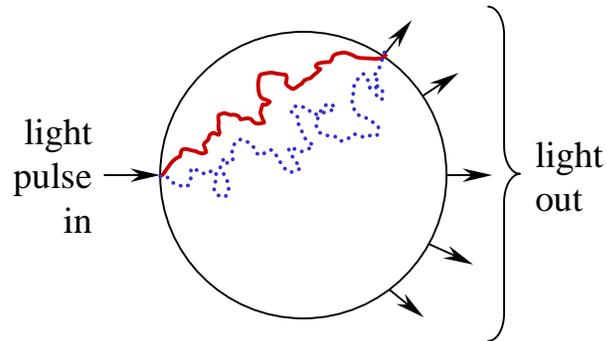


- Shine IR light pulse on tissue sample at several positions
- For each input pulse, measure at several output positions the light intensity vs. time with time resolution $\ll 1$ ns.
 - prompt, unscattered photons; few survive thick sections
 - multiply scattered photons; meander or diffuse through section

Alternative experimental schemes

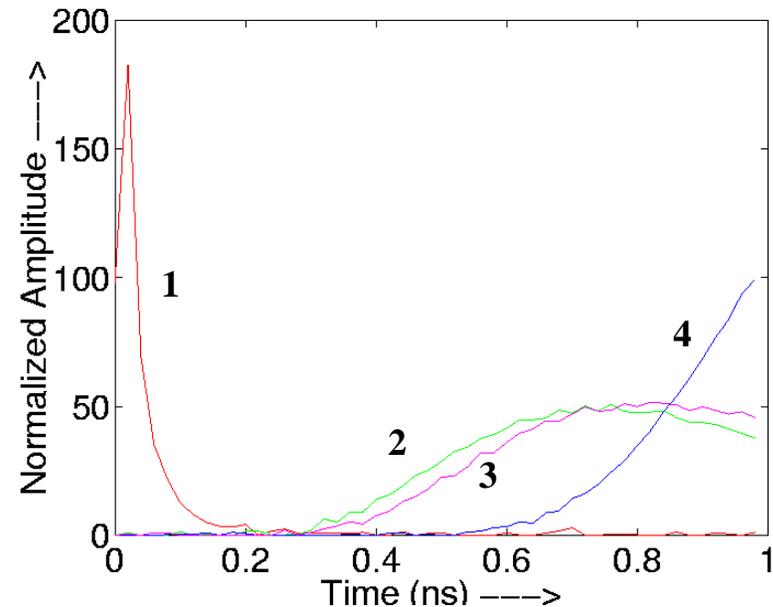
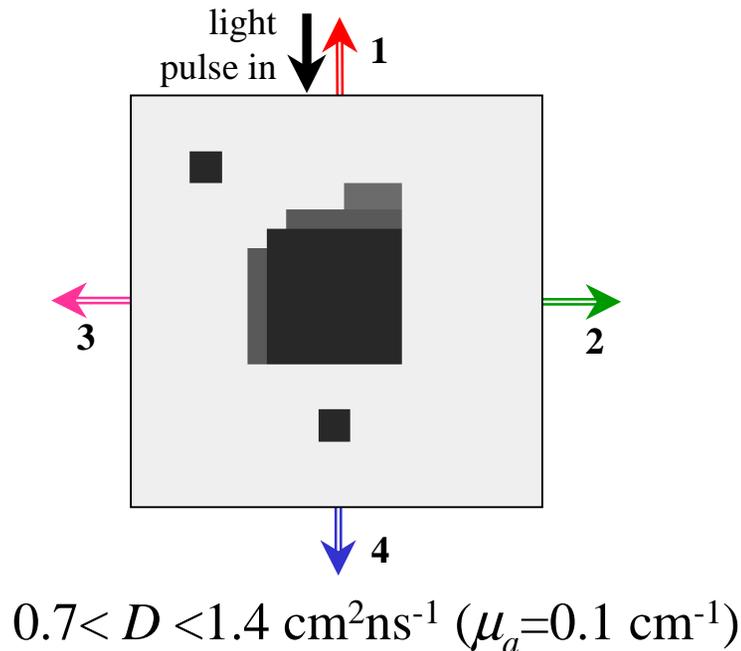
- Numerous types of measurements of the light transmitted through tissue sample are possible:
 - pulsed input; measure full time dependence distribution (delta-function response)
 - pulsed input; measure average time $\langle t \rangle$ (first moment of time distribution)
 - modulated input; measure amplitude and phase of modulated output intensity (Fourier transform of delta-function response)
 - constant input; measure amplitude of output (integral of delta-function response)

Modeling of process



- IR light photons in broad, retarded peak literally “diffuse” by multiple scattering from source to detector
 - time is equivalent to distance traveled
 - diffusion equation models these multiply-scattered photons
 - these photons do not follow straight lines

Simulation of light diffusion in tissue

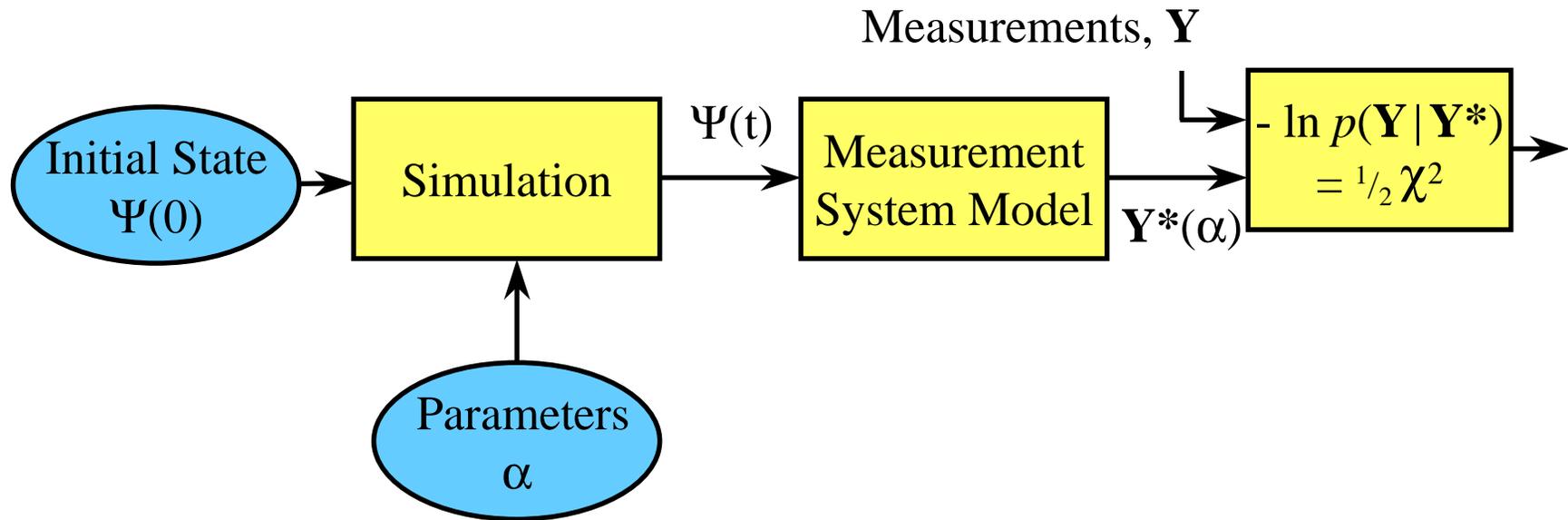


- for assumed distribution of diffusion coefficients (left)
- predict time-dependent output at four locations (right)
- reconstruction problem - determine image on left from data on right

Reconstruction problem

- Determine tissue properties from measurements
 - diffusion coefficient $D(x,y)$ and absorption coefficient $\mu_a(x,y)$, as a function of position - therefore, many unknowns
- Many problems must be overcome
 - photon paths depend on properties to be reconstructed
 - hence, inverse problem is nonlinear and difficult
 - measurements can only be calculated numerically; no analytic expression for measurements in terms of D and μ_a
 - gradients are desired for speedy gradient-based optimization
 - analytic gradients are not available
 - numerical gradients by perturbation too time consuming
 - **adjoint differentiation** crucial to success

Parameter estimation by fitting data



- Diagram describes general approach (analytical and computational)
- Find parameters (vector α) that minimize $-\ln p(Y|Y^*(\alpha))$
- Result is **maximum likelihood estimate** for α
 - also known as minimum-chi-squared or least-squares solution
- Optimization process is accelerated by using **gradient-based algorithms**; therefore need gradients of simulation and measurement processes

Diffusion equation

- Infrared light diffuses through tissue and bone
- Partial differential equation describes diffusion process
 - $U(x,y,t)$ is intensity of diffused light (no angular dependence)

$$\frac{\partial U}{\partial t} = \frac{\partial}{\partial x} \left[D \frac{\partial U}{\partial x} \right] + \frac{\partial}{\partial y} \left[D \frac{\partial U}{\partial y} \right] - c\mu_{abs}U + S$$

- where $D(x,y)$ is position-dependent diffusion coefficient,

$\mu_{abs}(x,y)$ is the linear absorption coefficient,

c is the speed of light,

$S(x,y,t)$ is a source term;

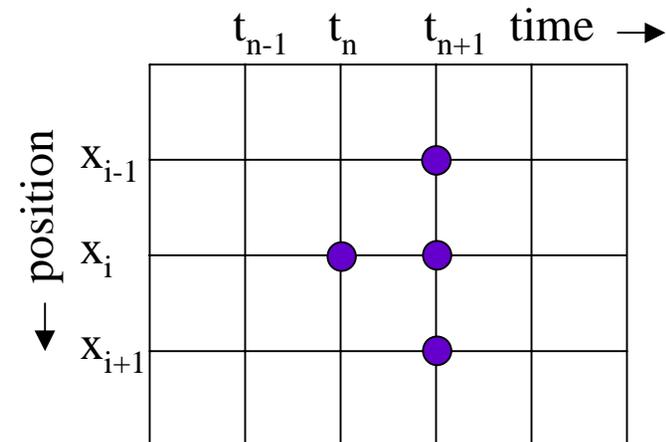
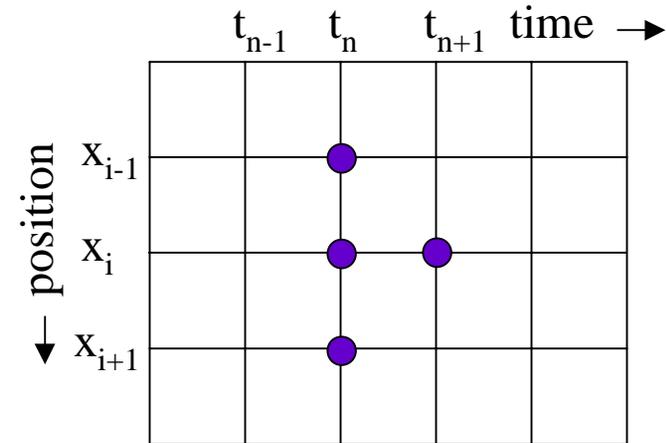
$$D = c[3(\mu_{abs} + \mu'_{scat})]^{-1} \quad (\mu'_{scat} = \text{effective scattering coefficient})$$

Method of finite differences

- Approximate derivatives by finite differences
 - wrt time: $\frac{\partial U}{\partial t} \Rightarrow \frac{\Delta U}{\Delta t} = \frac{U_{i,n+1} - U_{i,n}}{\Delta t}$
 - wrt position: $\frac{\partial^2 U}{\partial x^2} \Rightarrow \frac{U_{i+1,n} - 2U_{i,n} + U_{i-1,n}}{(\Delta x)^2}$
- Differential equation then leads to a set of linear equations to be solved to obtain time-step update
 - calculate time evolution, starting with initial conditions
- Question: at what time should second derivative wrt position be calculated, n or $n+1$?

Calculation of finite differences

- For diffusion equation, need
 - temporal first order derivative
 - spatial second order derivative
- Explicit technique
 - for step from time n to $n+1$, evaluate spatial derivative at n
 - **unstable** for moderate time steps
- Implicit technique
 - for step from time n to $n+1$, evaluate spatial derivative at $n+1$
 - inherently **stable**



Explicit method

- Evaluating position derivatives at n

$$\frac{U_{i,n+1} - U_{i,n}}{\Delta t} = D_i \frac{U_{i+1,n} - 2U_{i,n} + U_{i-1,n}}{(\Delta x)^2} - c\mu_i U_{i,n} + S_{i,n}$$

- for clarity, ignore position dependence of D and y coord.
- yields set of linear equations:

$$\boxed{\mathbf{U}_{n+1} = \mathbf{B}\mathbf{U}_n + b\mathbf{S}_n} \quad (b \text{ is a scalar constant})$$

- \mathbf{U}_{n+1} at new time $n+1$ is given **explicitly** in terms of state at previous \mathbf{U}_n
- Easy to calculate time steps (just matrix multiplication)
- Unfortunately, inherently **unstable** for moderate Δt

Implicit method

- Evaluating position derivatives at time $n+1$

$$\frac{U_{i,n+1} - U_{i,n}}{\Delta t} = D_i \frac{U_{i+1,n+1} - 2U_{i,n+1} + U_{i-1,n+1}}{(\Delta x)^2} - c\mu_i U_{i,n+1} + \frac{1}{2}(S_{i,n+1} + S_{i,n})$$

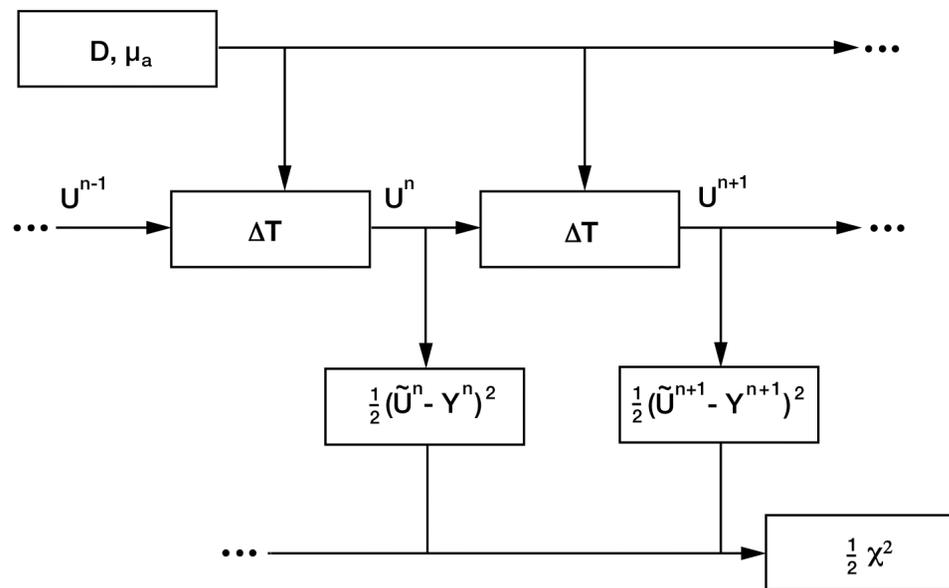
- for clarity, ignore position dependence of D and y coord.
- yields set of linear equations:

$$\boxed{\mathbf{A}\mathbf{U}_{n+1} = \mathbf{U}_n + a\mathbf{S}_n} \quad (a \text{ is a scalar constant})$$

- \mathbf{U}_{n+1} at new time $n+1$ is given **implicitly** in terms of state at previous time \mathbf{U}_n
- Must solve set of linear eqs. to calculate time steps
- Inherently **stable** for moderate Δt

Finite-difference calculation

- Data-flow diagram shows calculation of time-dependent measurements by finite-difference simulation
- Calculation marches through time steps Δt
 - new state \mathbf{U}_{n+1} depends only on previous state \mathbf{U}_n



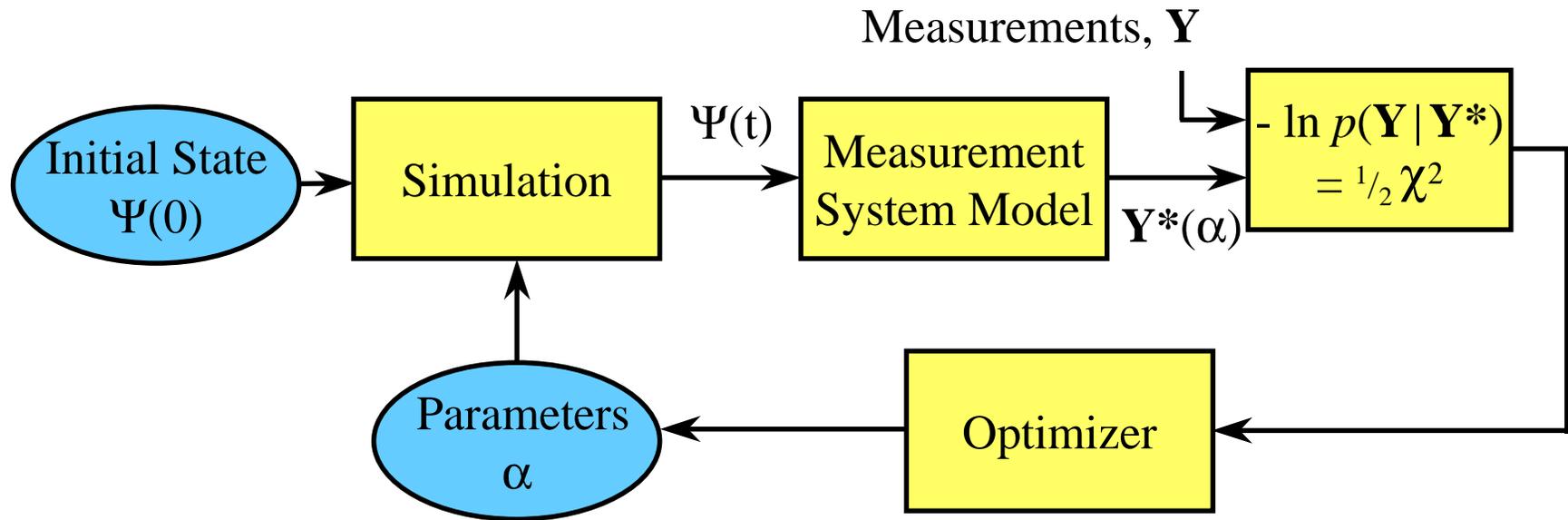
Inversion of forward calculation

- To find parameters $\alpha = (D, \mu_a)$, minimize minus-log-likelihood of data:

$$\phi(\alpha) = -\ln p(\mathbf{Y} | \alpha) = \frac{1}{2} \sum_m \frac{(Y_m - Y_m^*)^2}{\sigma_m^2} = \frac{1}{2} \chi^2$$

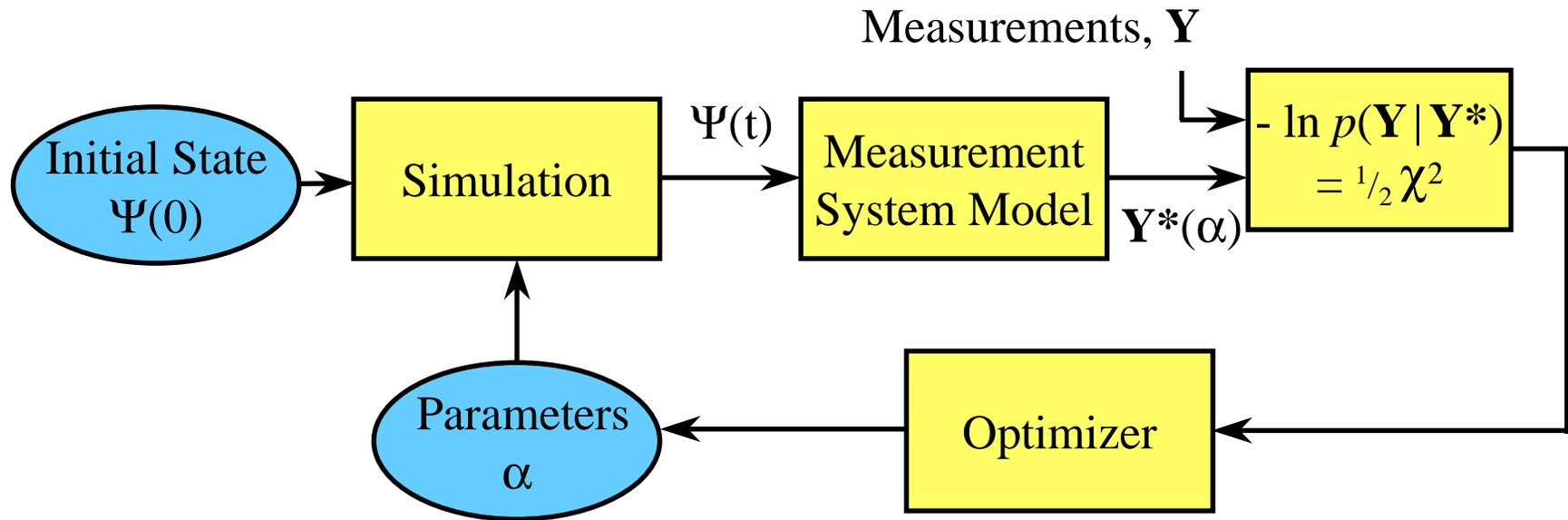
- where Y_m is the m th measurement,
 Y_m^* its predicted value ($= U_{s,n}$ at appropriate s and n),
 σ_m is rms noise in measurement
- measurements are at fixed position, but at all times
- Problems for inverting diffusion process
 - inversion may be ill posed, a theoretical issue
 - have only numerical solution of forward simulation, so calculation of gradient poses practical problem

Parameter estimation by fitting data



- Find parameters (vector α) that minimize $-\ln p(\mathbf{Y} | \mathbf{Y}^*(\alpha))$
- Result is **maximum likelihood estimate** for α
 - also known as minimum-chi-squared or least-squares solution
- Prior information can be used to overcome ill-posedness
- Bayesian approach

Maximum likelihood estimation by optimization



- Find minimum in $-\ln p(\mathbf{Y} | \mathbf{Y}^*(\alpha)) = \frac{1}{2} \chi^2$ by iteration over parameters α
- Optimization process is accelerated by using **gradient-based algorithms**; therefore need gradients of simulation and measurement processes
- **Adjoint differentiation** facilitates efficient calculation of gradients, i.e. derivative of scalar output ($\frac{1}{2} \chi^2$) wrt parameters α

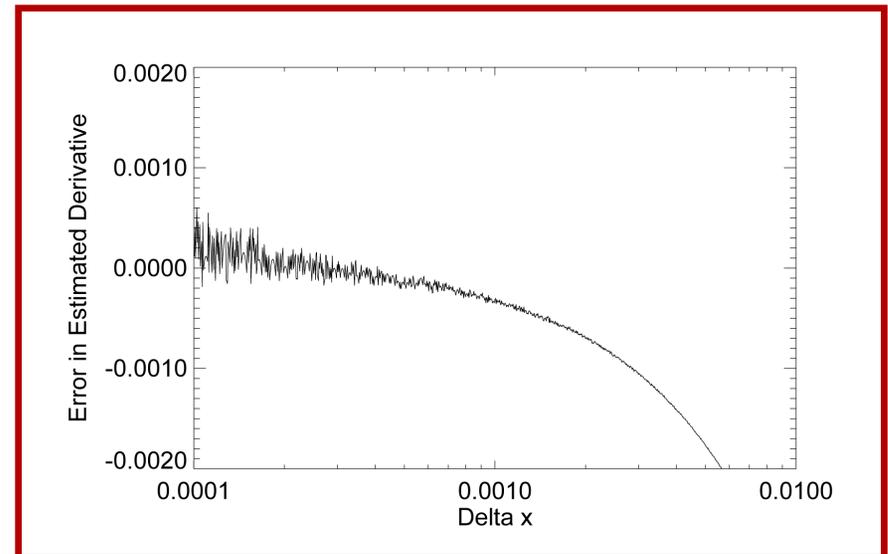
Derivative calculation by finite differences

- Derivative for function defined as limit of ratio of finite differences:

$$\left. \frac{df}{dx} \right|_{x_1} = \lim_{\Delta x \rightarrow 0} \frac{f(x_1 + \Delta x) - f(x_1)}{\Delta x}$$

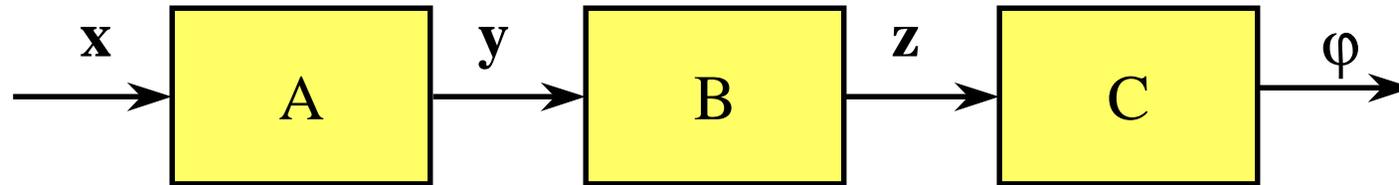
Wish to estimate derivatives of calculated function for which there is no analytic relation between outputs and inputs

- Numerical estimation based on finite differences is problematical:
 - difficult to choose perturbation Δx
 - # function evaluation \sim # variables
- Estimation based on functionality implied by computer code is more reliable



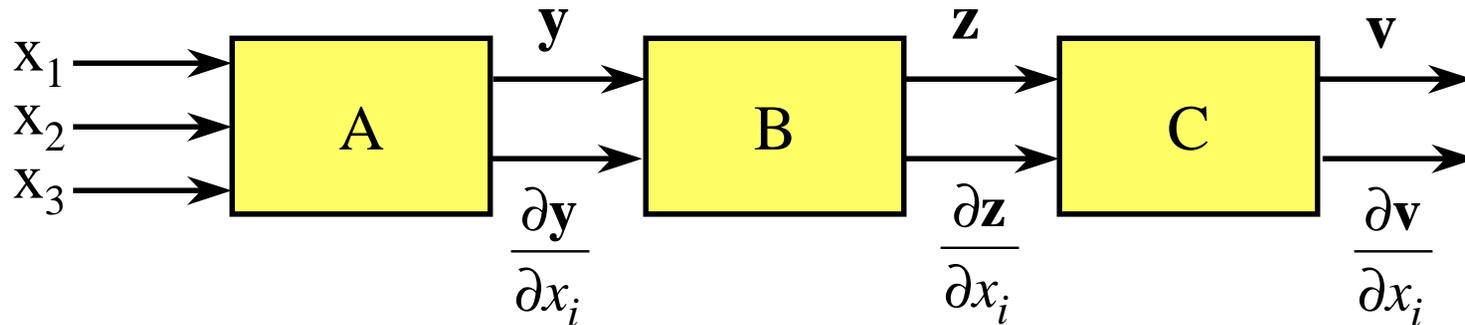
**Error in derivative of
 $\sin(x)$ vs. Δx at $x = \pi/4$**

Differentiation of sequence of transformations



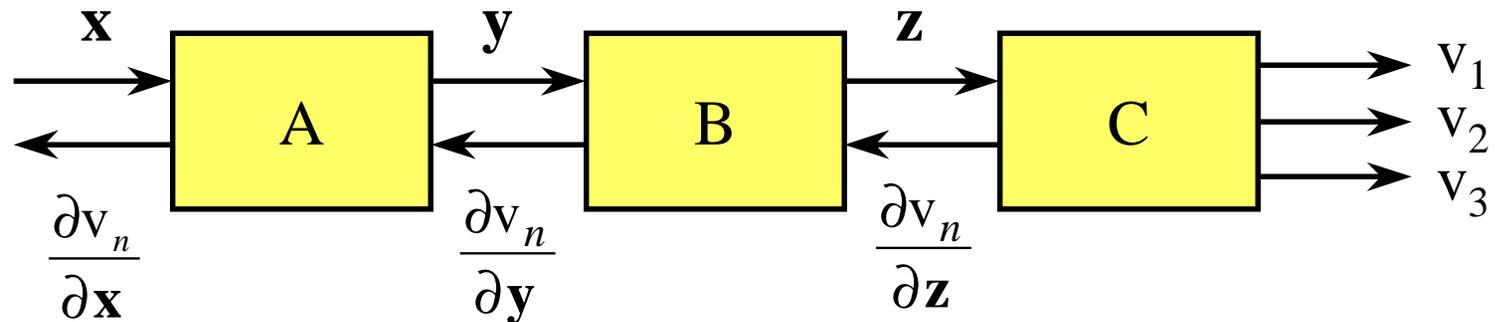
- Data-flow diagram shows sequence of transformations A->B->C that converts data structures \mathbf{x} to \mathbf{y} to \mathbf{z} and to scalar ϕ (**forward calculation**)
- Desire derivatives of ϕ wrt all components of \mathbf{x} , *assuming* ϕ is *differentiable*
- Chain rule applies:
$$\frac{\partial \phi}{\partial x_i} = \sum_{j,k} \frac{\partial y_j}{\partial x_i} \frac{\partial z_k}{\partial y_j} \frac{\partial \phi}{\partial z_k}$$
- Two choices for summation order:
 - doing j before k means derivatives follow data flow (forward calculation)
 - doing k before j means derivatives flow in reverse (adjoint) direction

Derivatives for few variables in, many out



- Derivatives of a few input variables wrt many output variables
- Most efficient approach is to propagate derivatives wrt input variables in direction of forward calculation
- For m inputs:
 - computation takes about m times longer than forward calculation
 - intermediate storage required is m times larger than for forward calculation
- One obtains derivatives of all variables in forward calculation wrt each input

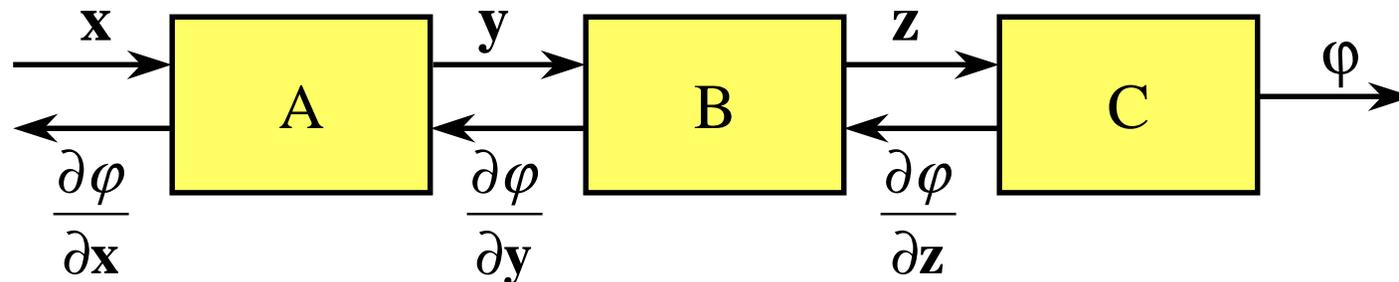
Derivatives for many variables in, few out



- Derivatives of many outputs wrt just a few output variables
- Most efficient approach is to propagate derivatives wrt output variables in reverse direction (also called **adjoint direction**)
- For n outputs:
 - computation takes on order of n times longer than forward calculation
 - intermediate storage required is n times larger than for forward calculation
- One obtains derivatives of each output variable wrt all variables in forward calculation

Adjoint Differentiation In Code Technique

ADICT



- For sequence of transformations that converts data structure \mathbf{x} to scalar ϕ
- Derivatives $\frac{\partial \phi}{\partial \mathbf{x}}$ are efficiently calculated in the reverse (adjoint) direction
- **Code-based approach:** logic of adjoint code is based explicitly on the forward code or on derivatives of the forward algorithm
- **Not** based on the theoretical eqs., which forward calc. only approximates
- Only assumption is that ϕ is a **differentiable function** of \mathbf{x}

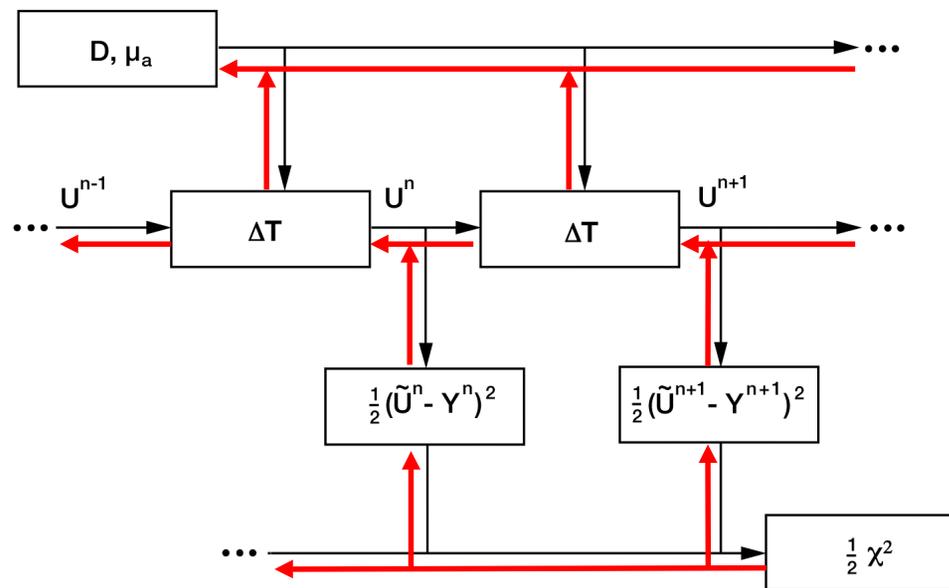
All derivatives computed in time comparable to forward calculation

Level of abstraction of implementation

- One can choose to differentiate forward code at various levels of abstraction - from coarse to fine
 - analytic-model based
 - e.g., differentiate partial differential equations and solve
 - not advised because forward codes only approximates model
 - algorithm based
 - differentiate each algorithm (e.g., Bayes Inference Engine)
 - code based
 - interpret programming code (FORTRAN, C, etc.)
 - automatic differentiation utilities
 - instruction based
 - reverse sequence of CPU instructions

Adjoint differentiation in diffusion calculation

- Adjoint differentiation calculation **precisely reverses** direction of forward calculation
- Each forward data structure has an associated derivative
 - where \mathbf{U}_n propagates forward, $\frac{\partial \phi}{\partial \mathbf{U}_n}$ goes backward ($\phi = \frac{1}{2} \chi^2$)



Adjoint differentiation of forward calculation

- Differentiate objective function, minus-log-likelihood:

$$\varphi(\alpha) = -\ln p(\mathbf{Y} | \alpha) = \frac{1}{2} \sum_m \frac{(Y_m - U_{s,n})^2}{\sigma_m^2} = \frac{1}{2} \chi^2$$

– where $U_{s,n}$ is at the position s and time n corresponding to the m th measurement

- Derivative wrt each $U_{s,n}$ that contributes to the m th measurement is

$$\frac{\partial \varphi}{\partial U_{s,n}} = -\frac{Y_m - U_{s,n}}{\sigma_m^2}$$

Adjoint differentiation of forward calculation

- Sensitivity of $\varphi = \frac{1}{2} \chi^2$ wrt parameters $\alpha_i = (D \text{ or } \mu_a)_i$

$$\frac{d\varphi}{d\alpha_k} = \sum_{i,n} \frac{d\varphi}{dU_{i,n}} \frac{\partial U_{i,n}}{\partial \alpha_k}$$

- Get second factor from update formula

$$\mathbf{A}\mathbf{U}_n = \mathbf{U}_{n-1} + a \mathbf{S}_n$$

- For explicit α_k sensitivity, differentiate wrt α_k , taking \mathbf{U}_{n-1} as constant:

$$\frac{\partial \mathbf{A}}{\partial \alpha_k} \mathbf{U}_n + \mathbf{A} \frac{\partial \mathbf{U}_n}{\partial \alpha_k} = 0$$

which leads to

$$\frac{\partial \mathbf{U}_n}{\partial \alpha_k} = -\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \alpha_k} \mathbf{U}_n$$

Adjoint differentiation of forward calculation

- Dependence of \mathbf{U}_{n+1} on \mathbf{U}_n comes from update formula

$$\mathbf{A}\mathbf{U}_{n+1} = \mathbf{U}_n + a \mathbf{S}_n$$

- Differentiate wrt \mathbf{U}_{n+1} : $\frac{\partial \mathbf{U}_{n+1}}{\partial \mathbf{U}_n} = \mathbf{A}^{-1}$

- Total derivative of φ wrt \mathbf{U}_n yields propagation rule

$$\frac{d\varphi}{d\mathbf{U}_n} = \left[\frac{d\mathbf{U}_{n+1}}{d\mathbf{U}_n} \right]^T \frac{d\varphi}{d\mathbf{U}_{n+1}} + \frac{\partial \varphi}{\partial \mathbf{U}_n} = [\mathbf{A}^{-1}]^T \frac{d\varphi}{d\mathbf{U}_{n+1}} + \frac{\partial \varphi}{\partial \mathbf{U}_n}$$

- second term is derivative of φ wrt \mathbf{U}_n when all other parameters are held constant
- first term for \mathbf{U}_n variation arising from other parameters

Comments about diffusion problem

- Algorithm used to solve forward problem was chosen without regard to inversion process
 - adjoint differentiation typically places minimal requirement on simulation method
- Simplifying aspects of diffusion problem:
 - update operation depends only on parameters $D(x,y)$ and $\mu_{abs}(x,y)$ (time independent)

Bayesian approach to inversion

- Inverse problems are often ill posed, meaning there is no unique solution
- Bayesian formalism overcomes ill posedness by introducing prior information through Bayes law:

$$\ln p(\alpha | \mathbf{Y}) = \ln p(\mathbf{Y} | \alpha) + \ln p(\alpha) + C$$

- where $p(\alpha|\mathbf{Y})$ = posterior probability of the parameters α ,
 $p(\mathbf{Y}|\alpha)$ = likelihood of the data,
 $p(\alpha)$ = prior probability of the parameters α
- Bayesian posterior $p(\alpha|\mathbf{Y})$ describes uncertainty in inferred parameters

Prior based on Markov Random Field model

- MRF can control local behavior of an intensity field
- Minus-log-prior given by (considering only D)

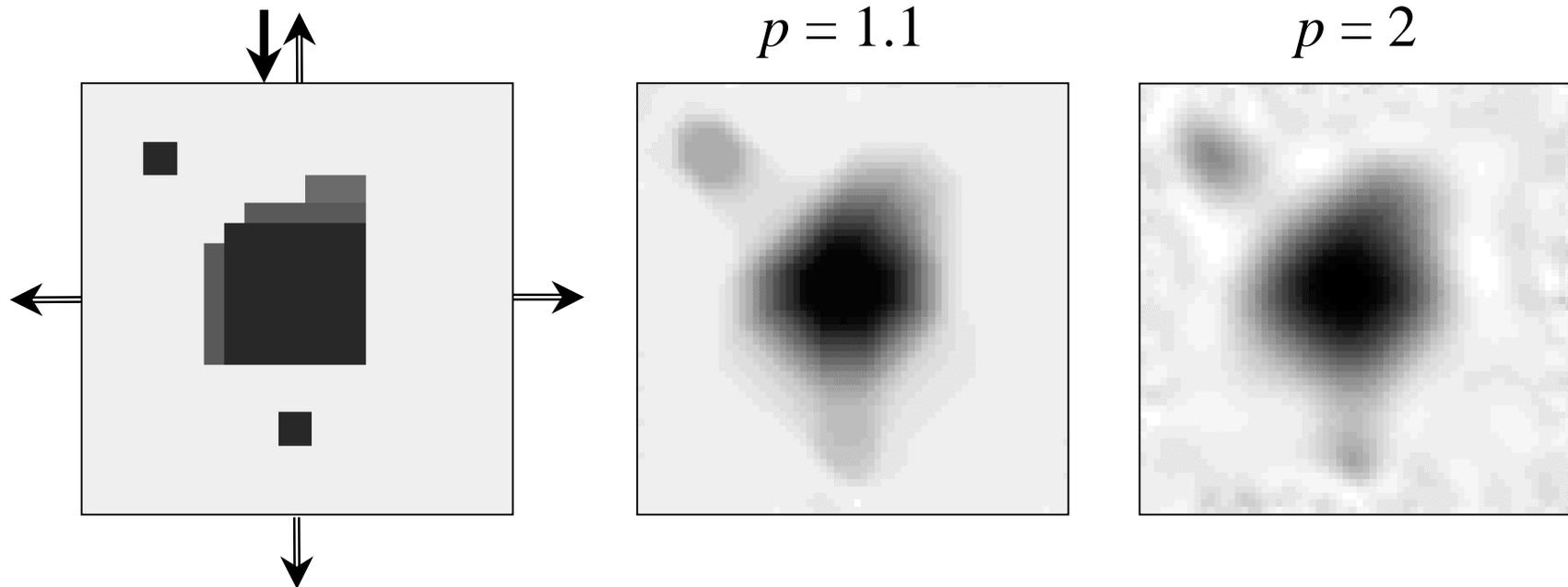
$$-\ln p(\mathbf{D}) = \beta \sum_i |D_i - \bar{D}_i|^p$$

- where D_i = diffusion coefficient at i th pixel,
 $\bar{D}_i = D$ averaged over a neighborhood of i th pixel
- this is added to minus-log likelihood ($\chi^2/2$)
- The exponent p controls shape of penalty function
 - $p = 2$ (standard) excessively penalizes large fluctuations
 - $p \cong 1$ results in better reconstructions
- Parameter β conveniently determined for MRF model

Examples of OT reconstructions

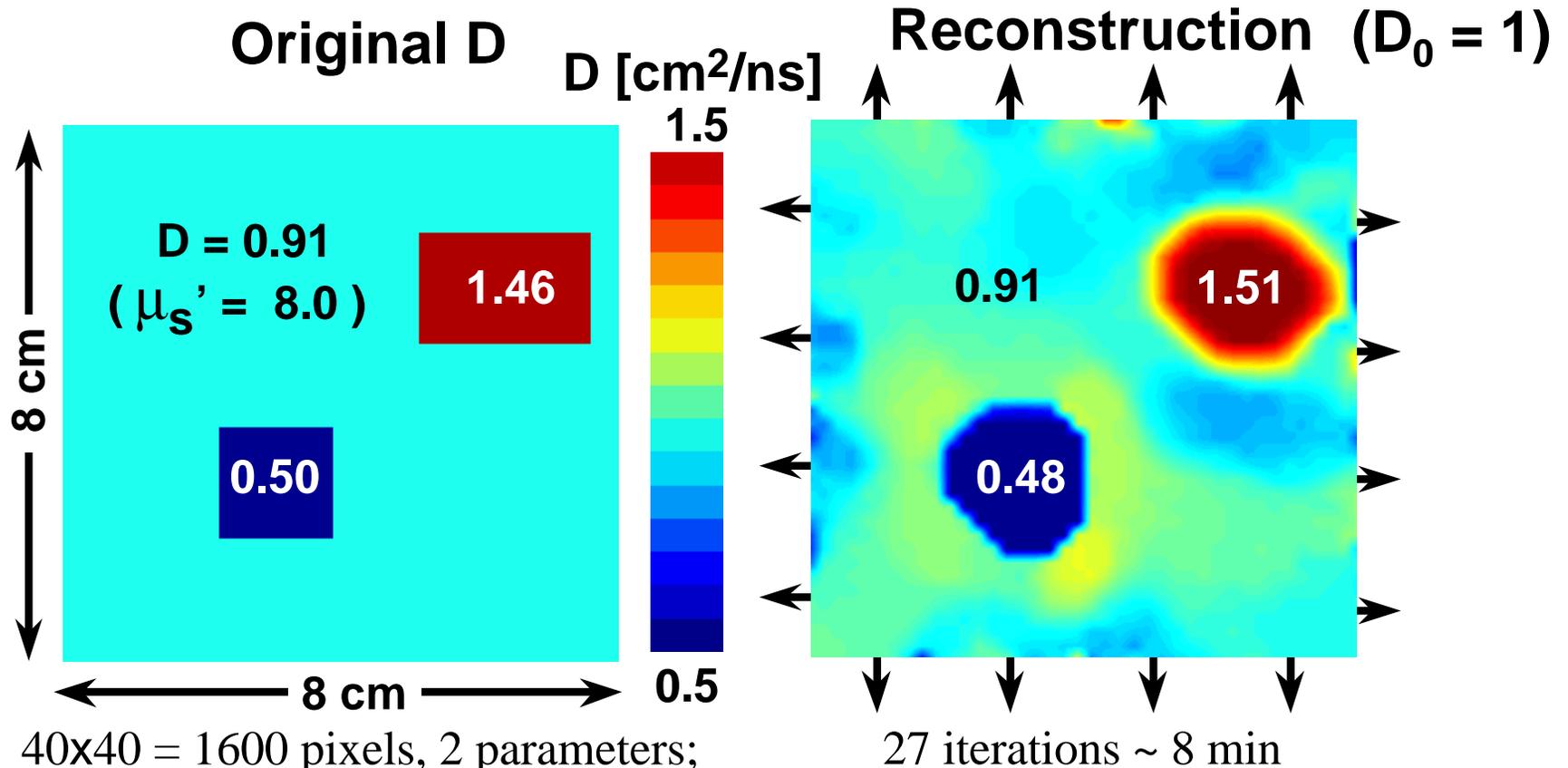
- Initial project - reconstruct $D(x,y)$ for simple phantom
 - Saquib, Hanson, Cunningham (LANL)
- Extension to simultaneously obtain $D(x,y)$ and $\mu_a(x,y)$; simulations relevant to human tissue
 - Hielscher, Klose, Catarious, Hanson (LANL)
- 3D reconstruction; applications to hypothetical diagnostic cases
 - brain, ventricular bleeding, and arthritis in finger joints
 - Hielscher, Klose (SUNY - Brooklyn), Hanson (LANL), Beuthan (FU Berlin)

Reconstruction of simple phantom



- Measurements
 - section is $(6.4\text{cm})^2$, $0.7 < D < 1.4 \text{ cm}^2\text{ns}^{-1}$ ($\mu_{\text{abs}} = 0.1 \text{ cm}^{-1}$)
 - 4 input pulse locations (middle of each side)
 - 4 detector locations; intensity measured every 50 ps for 1 ns
- Reconstructions on 64×64 grid from noisy data (rmsn = 3%)
- Conjugate-gradient optimization algorithm

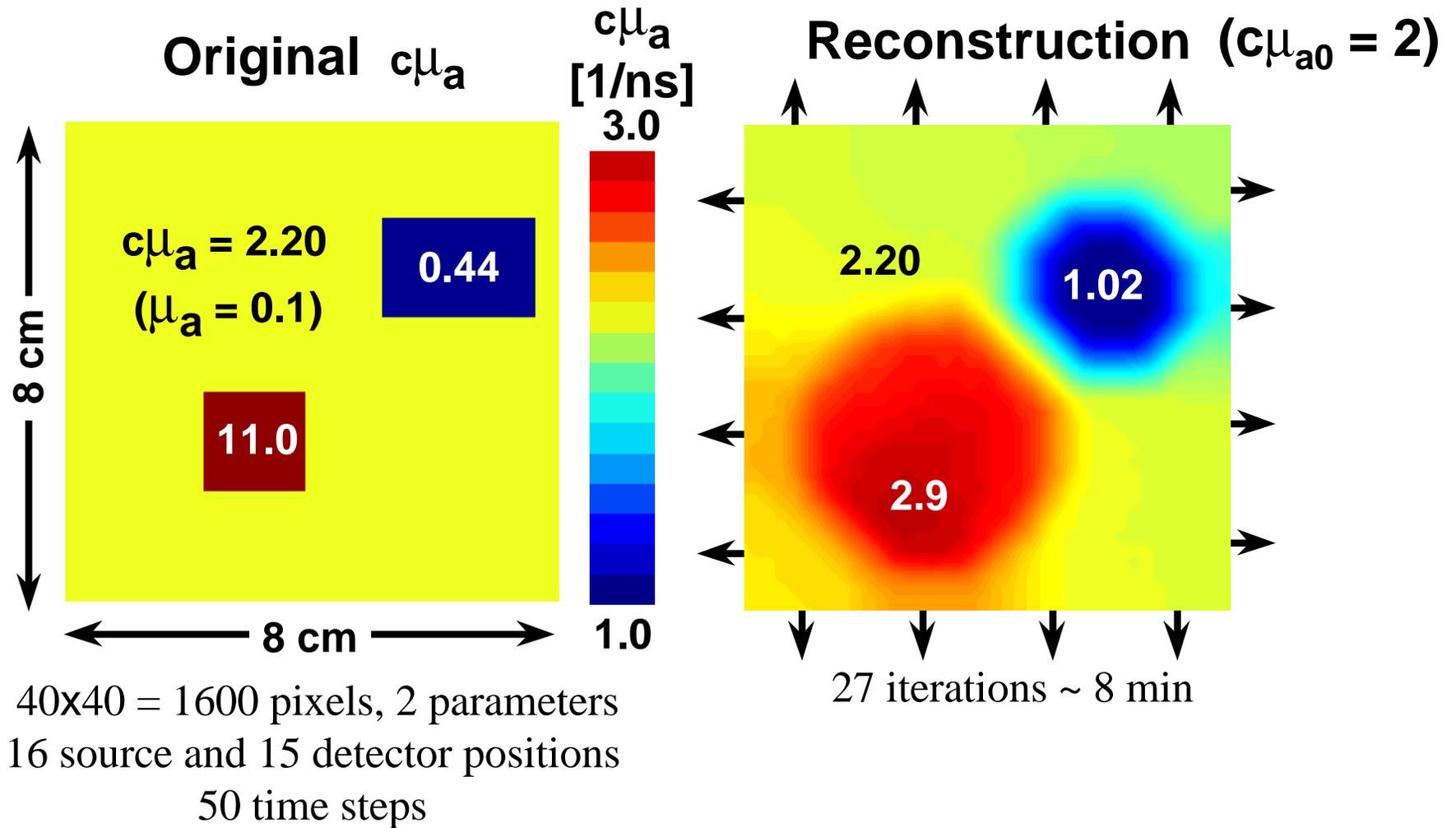
Simultaneously determine D and μ_a

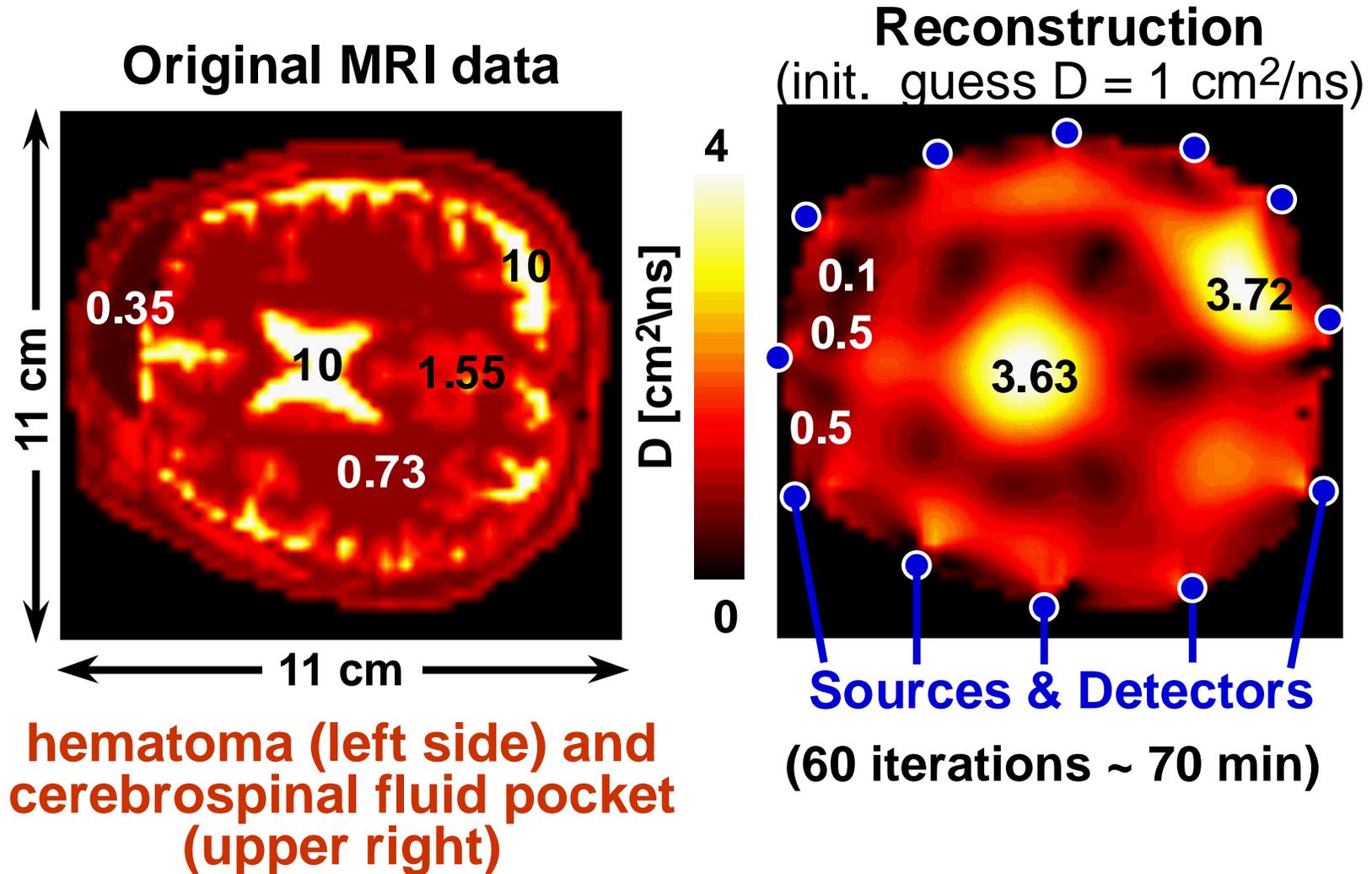


40x40 = 1600 pixels, 2 parameters;
16 sources and 15 detectors;
50 time steps

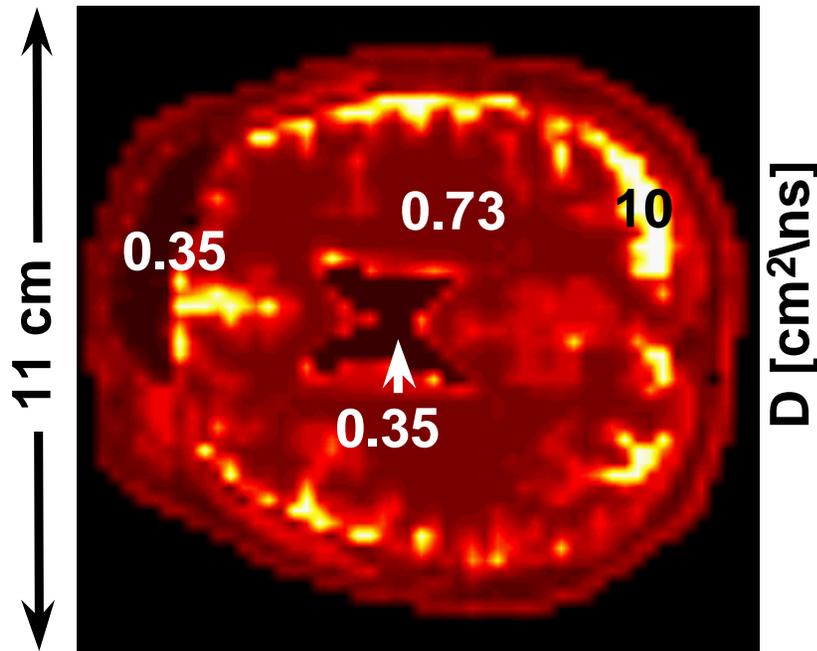
**Time for gradient calcs. for 27 iterations,
relative to time for a single forward calc., =
16x27 = 432 by adj. diff.
1600x2x16x27 = 1,382,400 by finite diff.**

Simultaneously determine D and μ_a



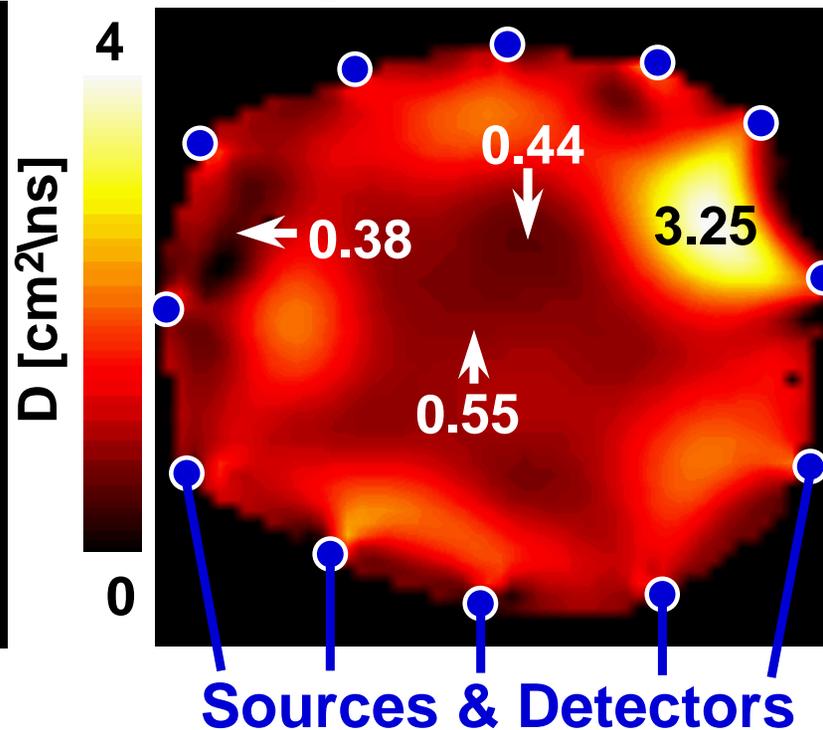


Original MRI data



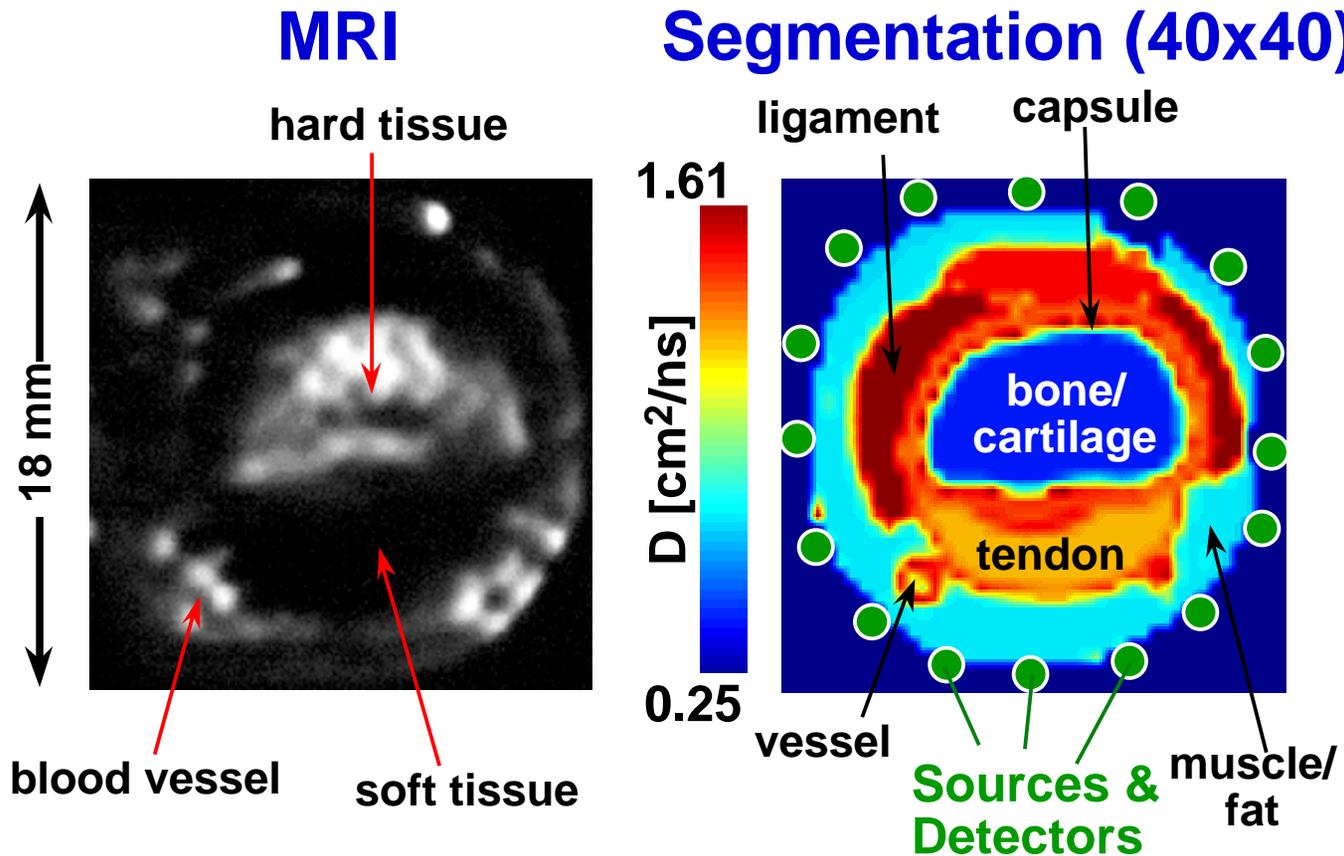
**blood-filled ventricle
(occurs in 15-30% of all
preterm infants)**

Reconstruction
(init. guess $D = 1 \text{ cm}^2/\text{ns}$)



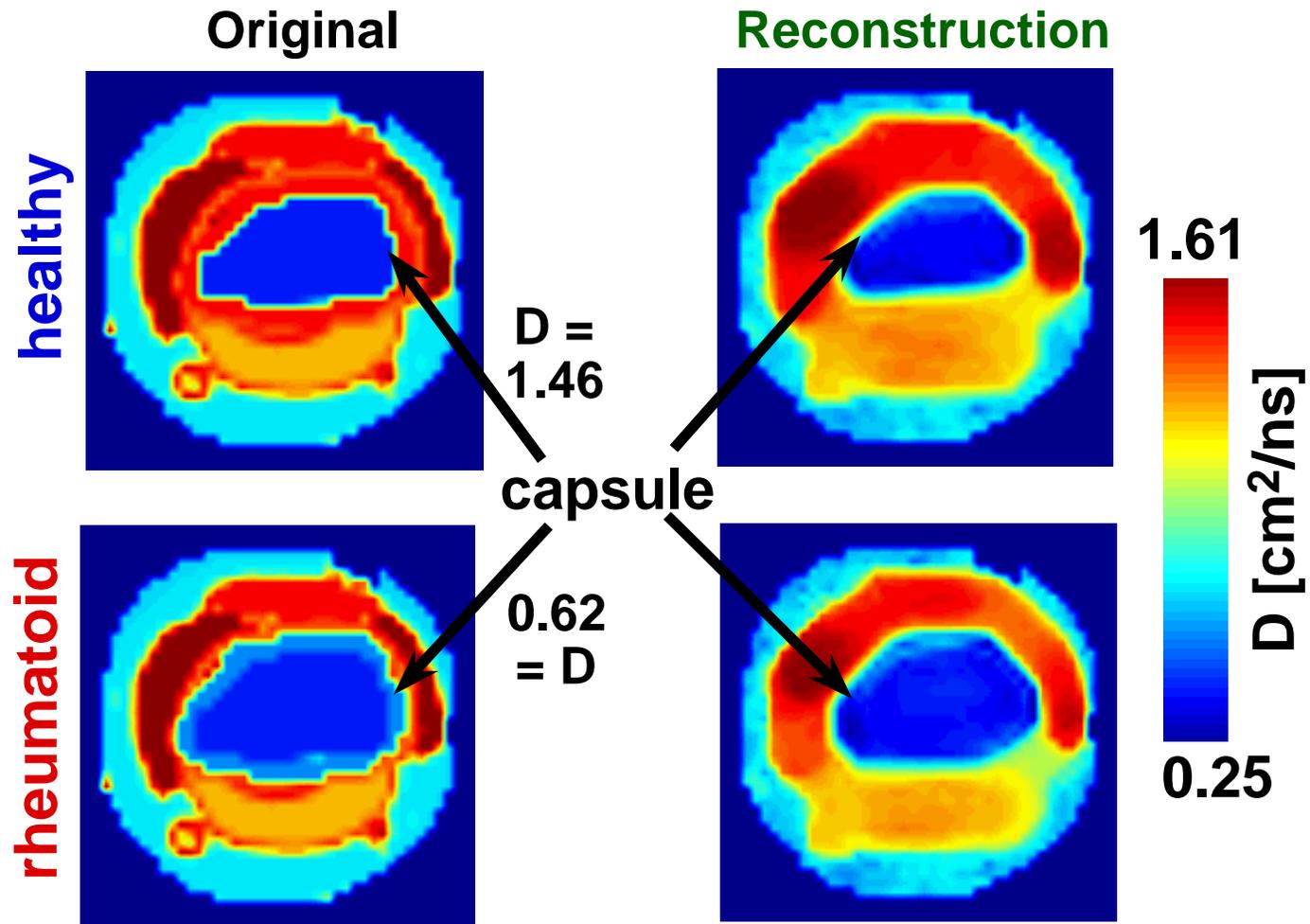
(60 iterations ~ 70 min)

Create Optical Image of Finger Joint

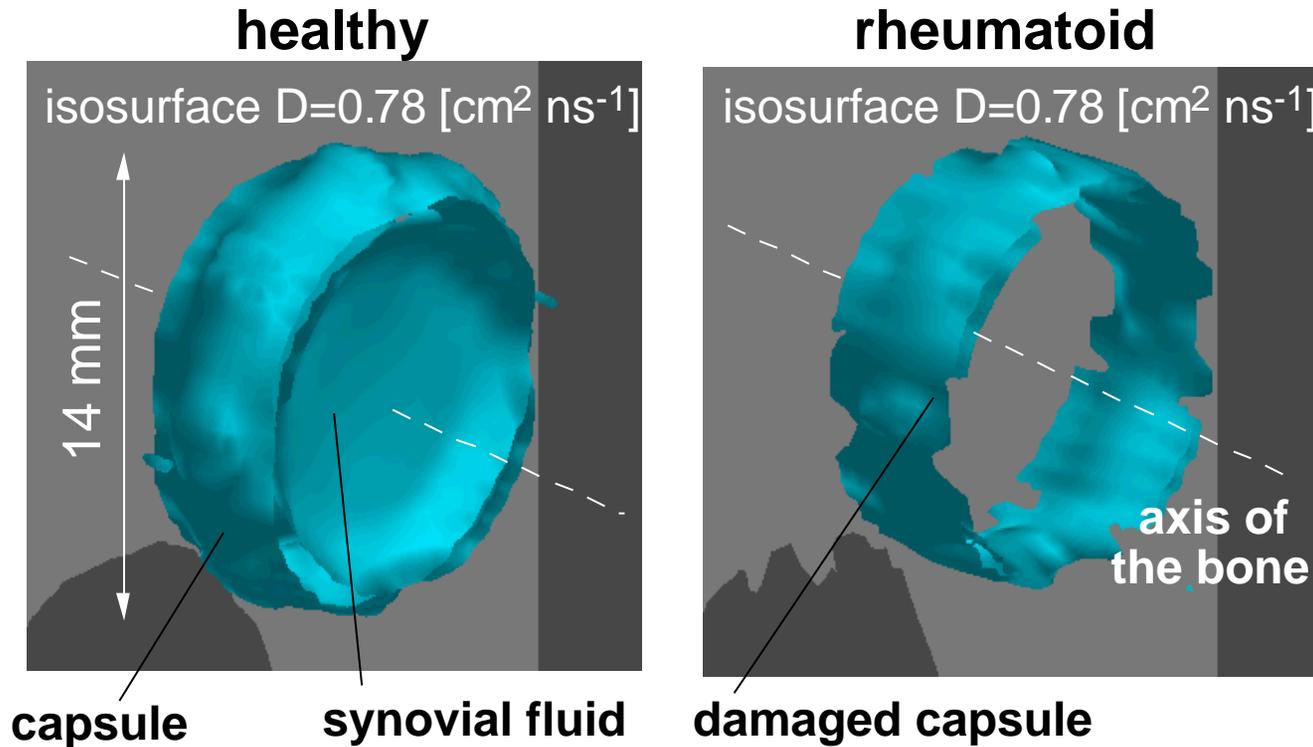


Simulated data obtained from optical image for source and detector positions shown

Reconstruction of Capsule



3D Reconstruction of Capsule



3D volume: $9 \times 30 \times 30 = 8100$ voxels
8 sources x 8 detectors x 4 layers

Summary

- Propagation of light in tissue modeled as diffusion process
- Solve forward problem using finite-difference, implicit method
- Diffusion coefficients determined by matching simulated to actual measurements
- Adjoint differentiation essential to solution by providing gradients with respect to diffusion coefficients at cost of only one extra forward calculation
- Adjoint differentiation is generally useful when objective function is differentiable

Bibliography

- “Inversion based on computational simulations,” K. M. Hanson, et al., in *Maximum Entropy and Bayesian Methods*, pp. 121-135 (Kluwer, 1998)
- “Model-based image reconstruction from time-resolved diffusion data,” S. S. Saquib, et al., *Proc. SPIE* **3034**, pp. 369-380 (1997)
- “Gradient-based iterative image reconstruction scheme for time-resolved optical tomography,” A. H. Hielscher, et al., *IEEE Trans. Med. Imag.* **18**, pp. 262-271 (1999)
- “Two- and three-dimensional optical tomography of finger joints for diagnostics of rheumatoid arthritis,” A. D. Klose, et al., *Proc. SPIE* **3566** (1998)

Find these under author’s home page <http://www.lanl.gov/home/kmh/>

References for adjoint differentiation

- "Adjoint differentiation of hydrodynamic codes," M. L. Rightley et al., in CNLS Research Highlights, April, 1998 (<http://public.lanl.gov/kmh/publications/CNLS97.pdf>)
- "Recipes for adjoint code construction," R. Giering et al., *ACM Trans. Math. Soft.* **24**, pp. 437-474 (1998). See also TAMC web pages below.
- Web sites for automatic differentiation packages:
ADIFOR: <http://www-unix.mcs.anl.gov/autodiff/ADIFOR/>
TAMC: <http://klima47.dkrz.de/tamc/>
ODYSSEE: <http://www.inria.fr/RRRT/RT-0224.html>

Part 2 - More about adjoint differentiation

- Other applications and uses of adjoint differentiation
- More about level of abstraction of implementation
- Automatic differentiation packages
- Use of check pointing to reduce resources needed to store forward solutions
- Interesting directions for future research

Further applications of adjoint differentiation

- Some significant applications under development
 - inversion of transport equation (A. Klose, FU Berlin)
 - oceanographics (Ralf Giering, JPL)
 - atmospherics (R. Errico, UCAR; R. Fovell, UCLA)
 - useful for simulations of up to several days
 - hydrodynamics (Rudy Henninger, LANL)

Potential uses of adjoint differentiation

- Reconstruction: imaging through refractive media
 - seismology, medical and NDE ultrasound, ...
- Matching large-scale simulations to data:
 - atmosphere and ocean models, fluid dynamics, hydrodynamic
- Optimization in large engineering design problems:
 - optical lens systems, geometry of integrated circuits, aerodynamic shape, engines
- Uncertainty analysis
 - sensitivity of uncertainty variance to each contributing cause
- Markov Chain Monte Carlo
 - generation of random samples from a prob. dens. function

Markov Chain Monte Carlo

- Generate random samples from a prob. dens. function
 - random walk around probability density function
- Simplest approach is based on evaluation of pdf (Metropolis)
 - efficiency drops for many variables ($\epsilon \sim 0.3/n$)
- Gradient of pdf can improve efficiency
 - Langevin-Hastings algorithm
 - Hamiltonian hybrid method ($\epsilon \sim$ independent of n)
- Gradient may be efficiently evaluated using adjoint differentiation

Further details about adjoint calculation

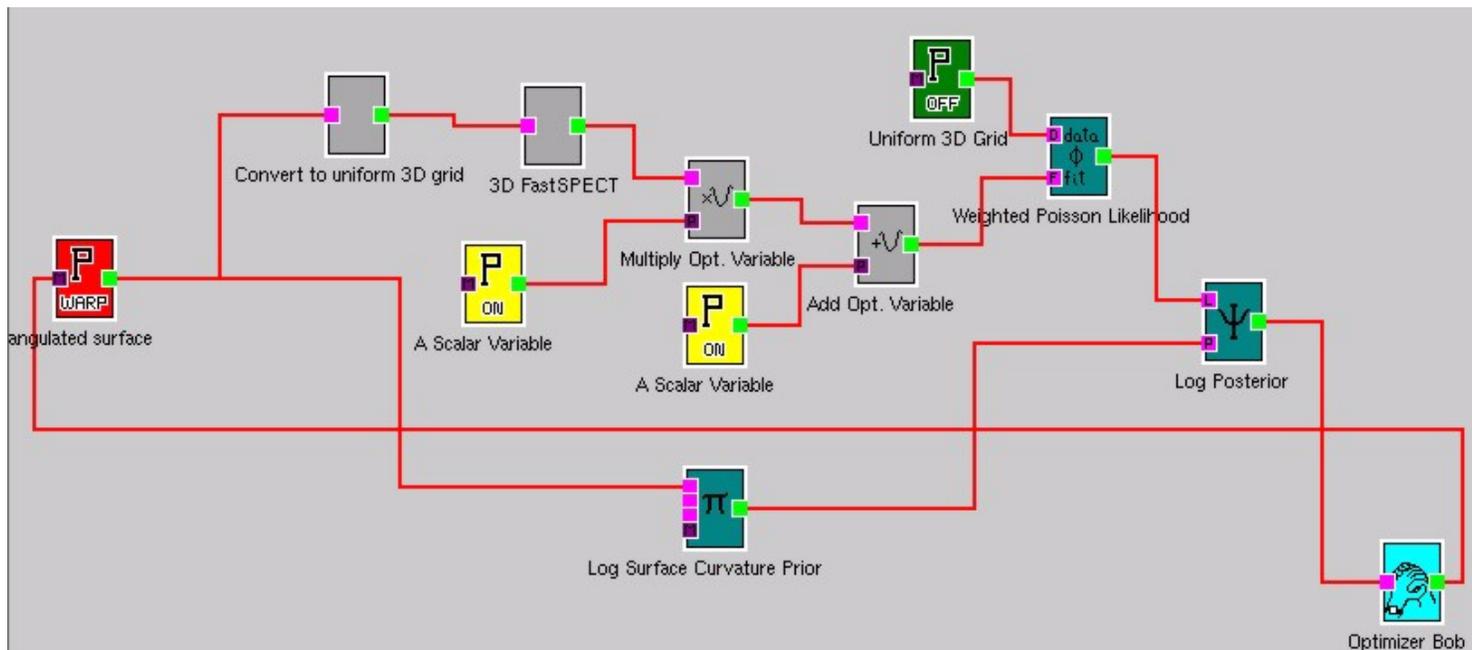
- Can be implemented a various levels of abstraction
- Automatic differentiation packages can be helpful
 - given forward code, provides an auxiliary code to calculate derivatives
 - potentially useful, but may require revising forward code
- Need to save variables from forward calculation implies huge storage requirements
 - check point the forward calculation; save intermediate results only occasionally

Level of abstraction of implementation

- One can choose to differentiate forward code at various levels of abstraction - from coarse to fine
 - analytic-model based
 - e.g., differentiate partial differential equations and solve
 - not advised because forward code only approximates model
 - algorithm based
 - differentiate each algorithm (e.g., Bayes Inference Engine)
 - code based
 - interpret programming code (FORTRAN, C, etc.)
 - automatic differentiation utilities
 - instruction based
 - reverse sequence of CPU instructions

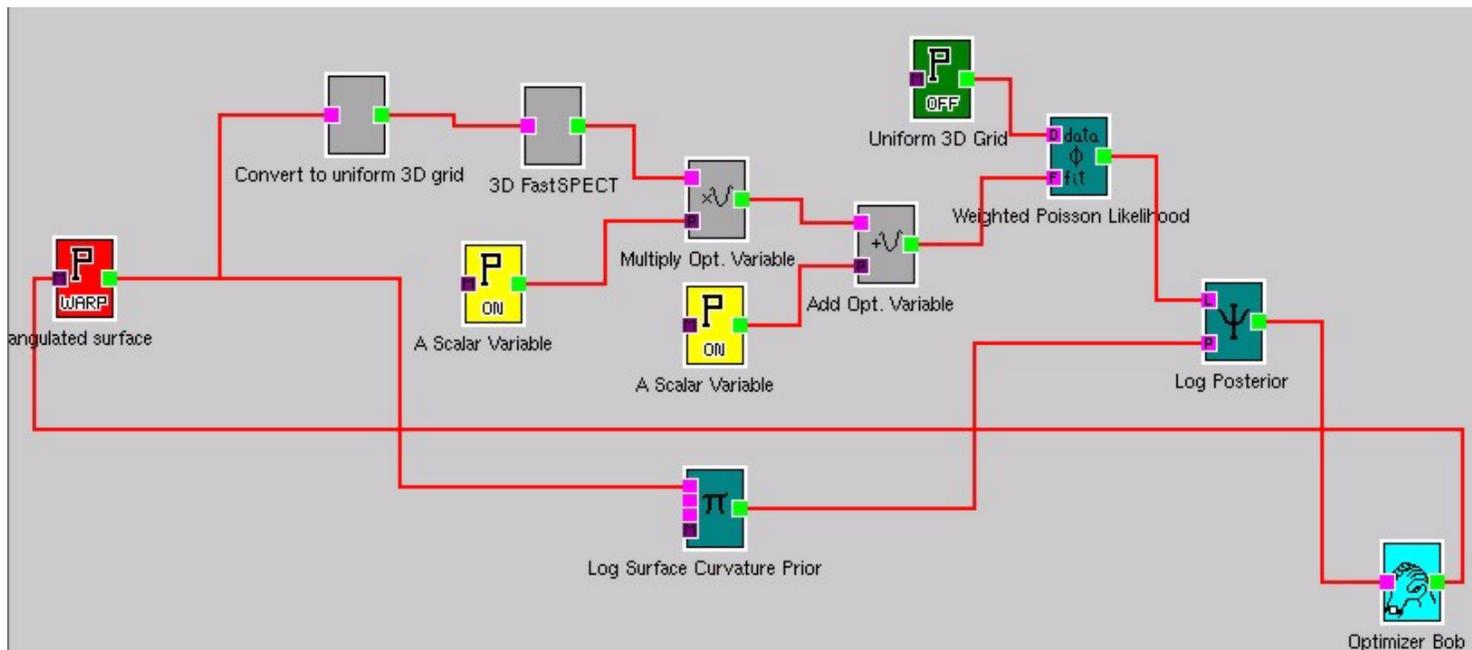
Example of algorithm-based approach

- Bayes Inference Engine (BIE) created at LANL
 - modeling tool for interpreting radiographs
 - BIE programmed by creating data-flow diagram, as shown here for 3D reconstruction problem
- **Adjoint differentiation crucial to BIE success**



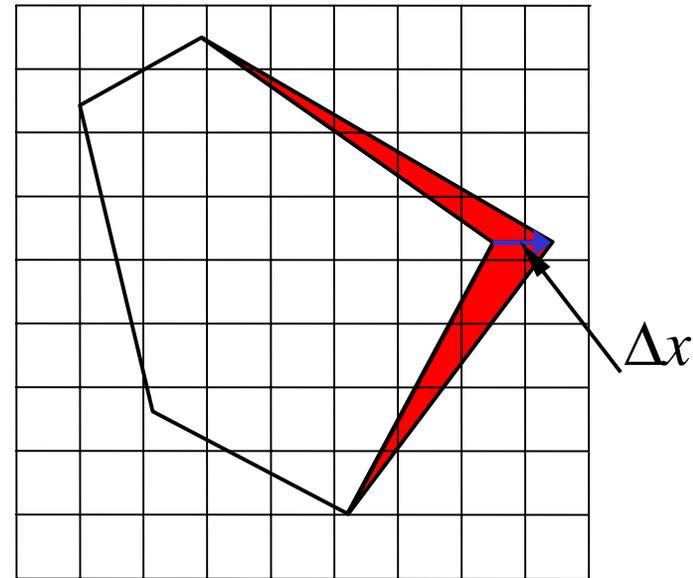
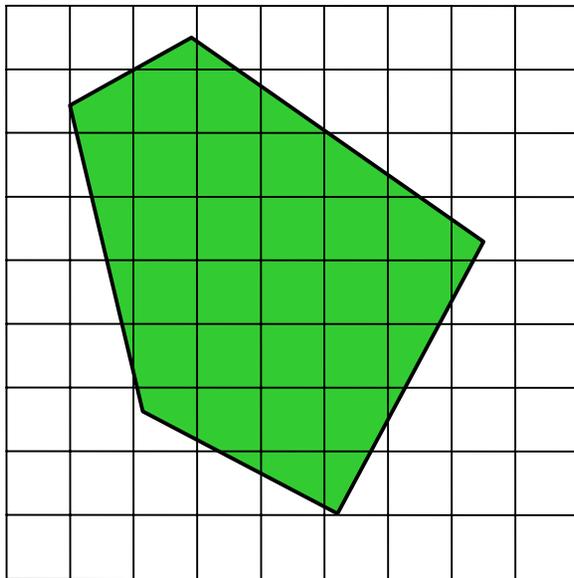
Bayes Inference Engine uses modules

- Each module uses an algorithm to calculate:
 - its transform in forward direction
 - adjoint derivative in reverse direction (implemented as separate code)
- Optimizer (lower right)
 - gets objective function from log-posterior; provokes forward data flow
 - gets gradient from Parameter module; provokes reverse data flow



Example of algorithm-based approach

- Calculation of overlap of polyhedron with pixelated grid
 - green is area of overlap;
 - input is vertex positions, output is overlap with each pixel
- Derivative is limit of change in overlap caused by change in position of each vertex
 - example shown in red;
 - adjoint output is derivative of ϕ wrt vertex positions



Adjoint differentiation for linear transforms

- Linear transforms represented by matrix multiplications:

$$\mathbf{z} = \mathbf{B}\mathbf{A}\mathbf{x}$$

$$\varphi = |\mathbf{z} - \mathbf{c}|^2$$

where \mathbf{z} and \mathbf{x} are vectors containing the variables,
 \mathbf{A} and \mathbf{B} are fixed matrices, and
 \mathbf{c} is a vector of constants (e.g., measurements)

- Derivative:

$$\frac{d\varphi}{d\mathbf{z}} = 2(\mathbf{z} - \mathbf{c})$$

$$\frac{d\varphi}{d\mathbf{x}} = \mathbf{A}^T \mathbf{B}^T \frac{d\varphi}{d\mathbf{z}} = \mathbf{A}^T \mathbf{B}^T 2(\mathbf{z} - \mathbf{c})$$

- For linear transforms, adjoint differentiation is the same as multiplication by the transpose of the matrix

Code-based forward derivative calculation

- Differentiate a few lines of FORTRAN code:

```
Y = EXP (A*X)
```

```
Z = B*Y*SIN (Y)
```

```
PHI = ALOG (Z*Z)
```

where X, Y, and Z are variables and A and B are constants

- The forward derivative code adds lines to forward code:

```
Y = A*X
```

```
DYDX = A*EXP (X)
```

```
Z = B*Y*SIN (Y)
```

```
DZDX = (B*SIN (Y) + B*Y*COS (Y) ) *DYDX
```

```
PHI = ALOG (Z*Z)
```

```
DPHIDX = (2.*Z) * (1./ (Z*Z) ) *DZDX
```

- Similar to differentiation of analytic functions

Code-based adjoint calculation of derivatives

- Differentiate a few lines of FORTRAN code:

```
Y = EXP (A*X)
```

```
Z = B*Y*SIN (Y)
```

```
PHI = ALOG (Z*Z)
```

where X, Y, and Z are variables and A and B are constants

- The reverse (adjoint) code looks like (note, DPHIDX is $\frac{\partial \phi}{\partial x}$):

```
DPHIDZ = (2.*Z) * (1./ (Z*Z) )
```

```
DPHIDY = DPHIDZ* (B*SIN (Y) + B*Y*COS (Y) )
```

```
DPHIDX = DPHIDY*A*EXP (X)
```

- forward code must be executed before adjoint code
- variables from forward calculation needed for adjoint calculation
- result for DPHIDX same as for forward calculation
- NB: this example is only meant to indicate how adjoint calculation is done; it is not particularly helpful in this situation

Code-based adjoint calculation of derivatives

- Differentiate a few lines of FORTRAN code:

Y = EXP (A*X)

Z = B*Y*SIN (Y)

PHI = ALOG (Z*Z)

where X, Y, and Z are variables and A and B are constants

- The reverse (adjoint) code looks like (note, DPHIDX is $\frac{\partial \phi}{\partial x}$):

DPHIDZ = (2.*Z) * (1./ (Z*Z))

DPHIDY = DPHIDZ* (B*SIN (Y) + B*Y*COS (Y))

DPHIDX = DPHIDY*A*EXP (X)

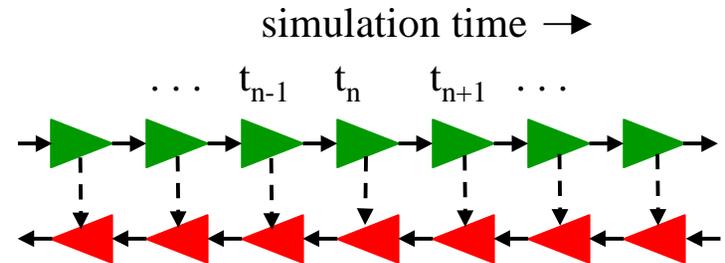
- forward code must be executed before adjoint code
- variables from forward calculation needed for adjoint calculation
- result for DPHIDX same as for forward calculation
- NB: this example is only meant to indicate how adjoint calculation is done; it is not particularly helpful in this situation

Automatic differentiation tools

- Several tools exist for automatically differentiating codes written in FORTRAN77
 - ADIFOR3 (Carle, Rice U; Bischof, Griewank, et al., ANL)
 - operates in both forward and reverse directions
 - works for large codes; follows ADICT principle
 - TAMC (R. Giering, JPL, prev. MIT & MPI-Meteorology)
 - operates in both forward and reverse directions
 - works for large codes; follows ADICT principle
 - GRESS (Hordewel, et al., ORNL)
 - operates in both forward and adjoint directions
 - can not compute gradients wrt many parameters for large calcs.
 - for adjoint, stores derivatives for each line of the forward code
 - ODYSSEE (INRIA, France)

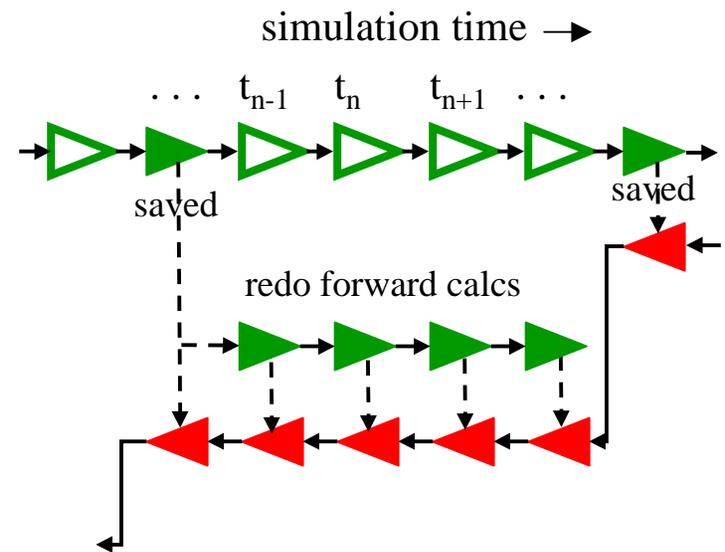
Sequence of forward and adjoint calculations

- Normal sequence for ADICT
 - first do full forward calculation (i.e., for all time steps)
 - then do full adjoint calculation in reverse direction, from final time back to beginning
- Adjoint calculation may need state of system from forward calculation
 - required link shown as vertical dashed lines
 - all necessary state variables must be saved from forward calculation to provide information to adjoint calculation



Check pointing forward results

- Adjoint calculation may need state of system from forward calculation
 - this requirement may exceed available memory
- Check pointing
 - first do full forward calc., **saving** state of system at selected times
 - then do adjoint calc. piecewise, repeating forward calc. to obtain states intermediate to those saved
 - trades off memory for compute time (for N time steps save $\sim \sqrt{N}$ storage for one extra forward calc.)



Potential extensions of adjoint differentiation

- Higher order derivatives
 - $\frac{\partial^2 \phi}{\partial x_i \partial x_j}$ requires 2 forward and 2 adjoint calculations
 - large intermediate matrices \Rightarrow restrict to $\sum_j \frac{\partial^2 \phi}{\partial x_i \partial x_j} x_j$
- Incorporate derivatives into data structures
 - with each variable vector \mathbf{x} , associate $\delta \mathbf{x}$ and $\partial / \partial \mathbf{x}$
 - for each transformation $f(\mathbf{x})$, associate $\frac{\partial f}{\partial \mathbf{x}}$ with capabilities for forward and adjoint propagation
 - useful in symbolic languages, such as Maple (S. Gull)
 - facilitated in object-oriented setting (Bayes Inference Engine)
- Construct new programming paradigm based on these composite data structures in OO environment
 - view computer code as establishing links between transforms