Optical tomography using the GIIR method

Alex P. Martinez

Biomedical Simulation Lab

Dept. of Mechanical and Industrial Engineering

Presentation Outline

- The problem
- Optical diffusion
  - Radiative transfer equation
  - Diffusion approximation
- Perturbation methods
- The GIIR method
  - The method
  - Finite differences
  - Adjoint differentiation
  - Conjugate gradient descent
  - Regularization
- Results
- References

The problem

- To find approximate optical properties in a medium using a finite set of measurements in a computationally efficient manner
- A finite-difference method can be used to simulate radiative transfer
- The method is used to find optical properties (an inverse problem)
- Advantages compared to other imaging modalities:
  - Infrared light is non-ionizing (it can be used for continuous monitoring)
  - Faster image acquisition (compared to MRI for example)
  - Optical imaging systems can be made portable
    (> 1T magnets not needed like in MRI)

When is radiation non-ionizing?

Radiation is non-ionizing if the energy of its photons is lower than the energy needed to detach electrons from atoms.

Energy of a photon is given by:

$$E = h\nu$$

Therefore, radiation above a certain frequency will be ionizing. This frequency is about 10^{16} Hz (ultraviolet light), higher than the frequency of infrared light (10^{12} to 10^{14} Hz).

The radiative transfer equation

- Infrared light is used because its absorption is minimal in tissue compared to other wavelengths of light
- Therefore the radiative transport equation is applicable.
- Under more assumptions applicable for tissue, the radiative transport equation can be simplified to a diffusion equation
- The equation can be derived from energy conservation on a ray of light traveling though tissue
- This equation is the most exact approximation to radiative transfer but a Monte Carlo method modeling actual photons yields better solutions for most real applications.

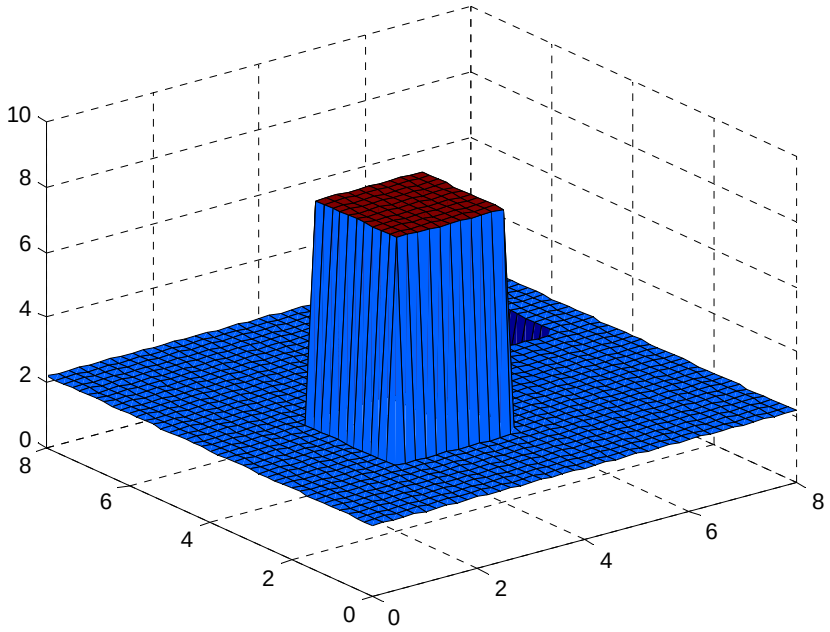
The diffusion approximation

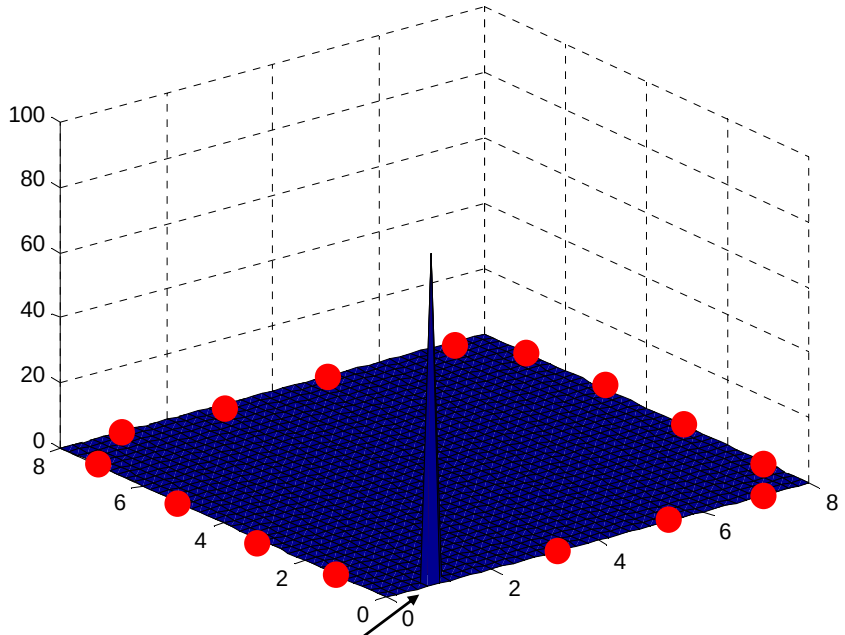
Also referred to as the delta-Eddington approximation:

$$\frac{\partial U}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial U}{\partial x}\right) + \frac{\partial}{\partial y}\left(D\frac{\partial U}{\partial y}\right) - c\mu_a U + S$$

Example of the forward problem:

Diffusion coefficient and initial condition:

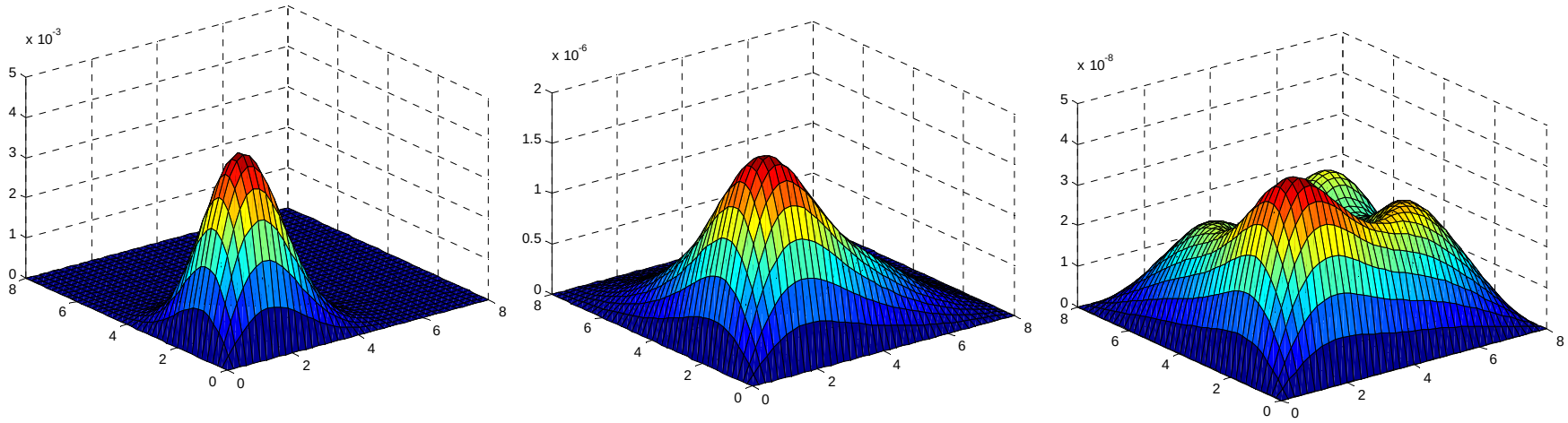




1 of 16 source/detectors

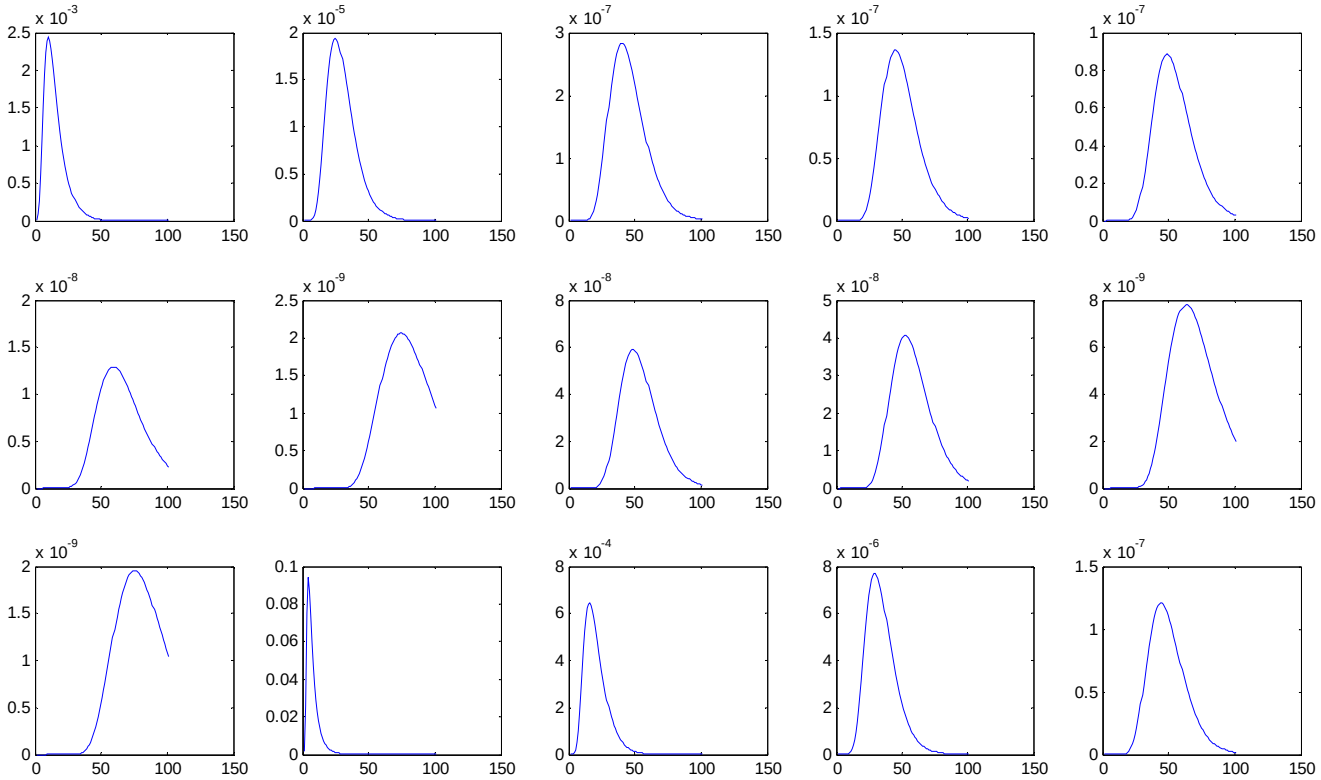
Example of the forward problem:

Diffusion of light:



Example of the forward problem:

Readout of 16-1 receivers:



Perturbation methods

A simple approach to solving the inverse optical diffusion problem.

The relation between measurements and estimated optical properties can be written as follows:

$$\mathbf{M}_p = \mathbf{F}[\theta_e]$$

A Taylor expansion about θ_e is:

$$\mathbf{M} = \mathbf{F}[\theta_e] + \mathbf{F}'\theta_e(\theta - \theta_e) + \mathbf{F}''(\theta - \theta_e)^T(\theta - \theta_e) + \ldots$$

The difference between actual and estimated optical properties is:

$$\Delta\mathbf{M} = \mathbf{M} - \mathbf{F}[\theta_e] = \mathbf{F}'\theta_e(\theta - \theta_e) + \ldots$$

Perturbation methods

A linear estimate of this difference is:

$$\Delta \mathbf{M} \approx \mathbf{J}[\theta_e]\Delta\theta$$

Where:

$$\mathbf{J}[\theta_e] = \mathbf{F}'[\theta_e]$$

$$\Delta\theta = \theta - \theta_e$$

A solution to the inverse problem is then:

$$\Delta\theta \approx \mathbf{J}^{-1}[\theta_e]\Delta\mathbf{M}$$

Where the result is used to find the optical properties:

$$\theta = \Delta\theta + \theta_e$$

Comments

- Linear approximation is “good” only if $\theta_e \approx \theta$
- Finding the inverse of the Jacobian is hard (full and ill-conditioned)
- The previous issue can be partially resolved by minimizing the following function:

$$\|\mathbf{J}[\theta_e]\Delta\theta - \Delta\mathbf{M}\|$$

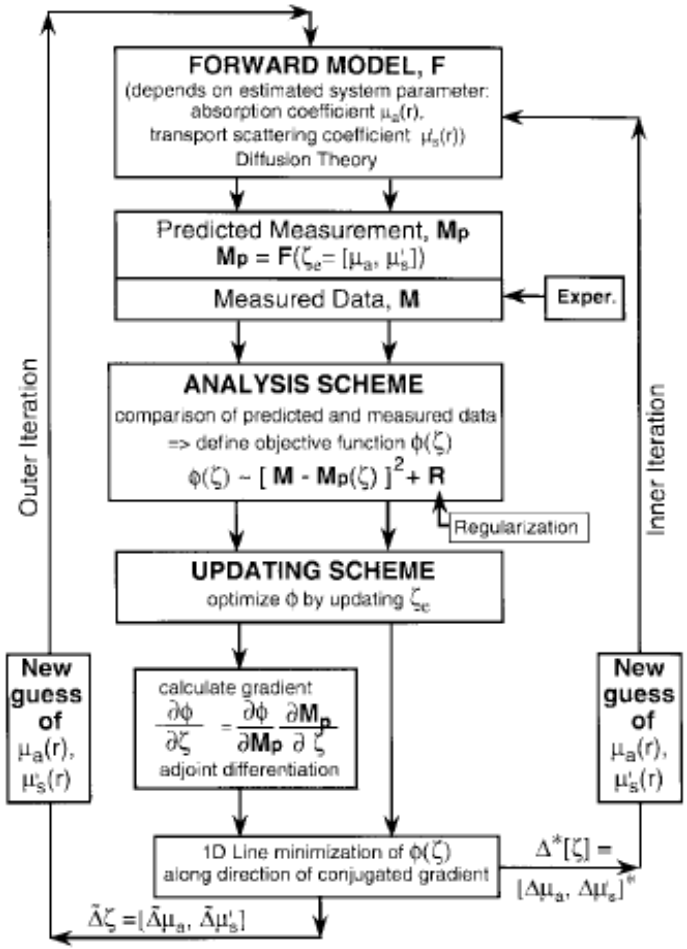
- Jacobian can also be made more diagonally dominant using a preconditioning matrix and then solving an equivalent problem that takes less computational effort
- This method can be used recursively

The GIIR method

GIIR stands for gradient-based iterative image reconstruction. The GIIR Method has the following steps:

- Step 1: find an initial guess for the optical properties
- Step 2: using a numerical approximation to the diffusion equation, find the gradient of a cost function with respect to a finite amount of optical parameters
- Step 3: use a gradient based method to find a new guess for the optical properties
- “Rinse and repeat”

Flow diagram of the GIIR method



Flow diagram from [2].

Comments

- Common scheme to solve diffusion equation is implicit finite differences
- Solving implicit scheme takes, per time step, $O(N^2)$ computations
- Option normally used to find gradients of a numerical scheme is numerical differentiation.
- Besides having truncation and cancellation errors, it takes $O(N^2)$ computations per outer iteration
- Both can actually be done in $O(N)$ computations
- This drops the total computation cost from $O(N^2)$ to $O(N)$ per outer iteration

Finite difference schemes

- Finite difference schemes can be used to discretize the diffusion equation
- This is done using finite differences
- Finite differences are useful in rectangular domains with structured grids
- The boundary condition was set to 0 for all schemes
- There are 3 main ways of doing this in 2 dimensions
- These 3 methods have different stability criteria and grid dependent errors

Explicit method

- In this method, the future light intensity is determined from the same node at a previous time step and its closest 4 adjacent nodes (done in $O(N)$ per iteration):

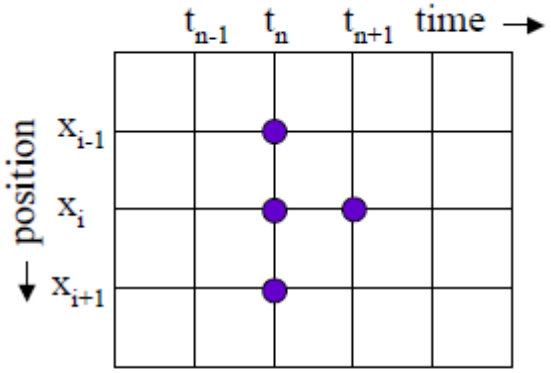
$$U^{n+1} = BU^n$$

- This scheme is stable under the following condition:

$$\Delta t < \frac{\Delta^2}{4(maxD_s)}$$

- The error for this scheme is the following:

$$O(\Delta t), O(\Delta^2)$$



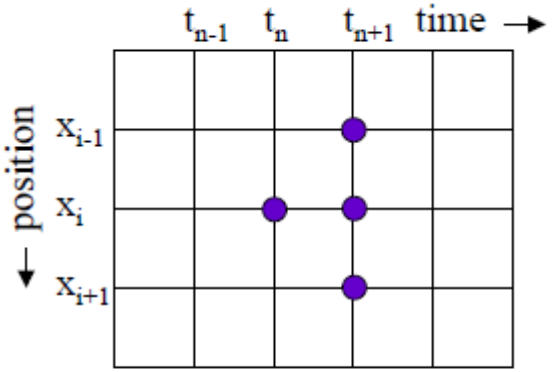
Implicit method

- In this method, the future light intensity is determined from the same node at a previous time step and its closest 4 adjacent nodes at the same time step (done in $O(N^2)$ per iteration):

$$AU^{n+1} = U^n$$

- This scheme is unconditionally stable
- The error for this scheme is the following:

$$O(\Delta t), O(\Delta^2)$$



Alternating directions implicit (ADI) method

- In this method, the future light intensity is determined implicitly in one direction, and explicitly in the other. This is done in one ‘half’ a time step, then the directions are reversed by rearranging the node numbering (done in $O(N)$ per iteration):

$$AU^{n+1/2} = BU^{n}$$

$$AU^{n+1}_{reordered} = BU^{n+1/2}_{reordered}$$

- This scheme is unconditionally stable

- Surprisingly, the error for this scheme is the following:

$$O(\Delta t^2), O(\Delta^2)$$

Numerical differentiation

Finite difference approximations of any accuracy can be obtained from the Taylor series expansion of the equation:

$$f(x+h) = f(x) + f^{'}(x)h + \frac{f^{''}(x)}{2!}h^2 + ...$$

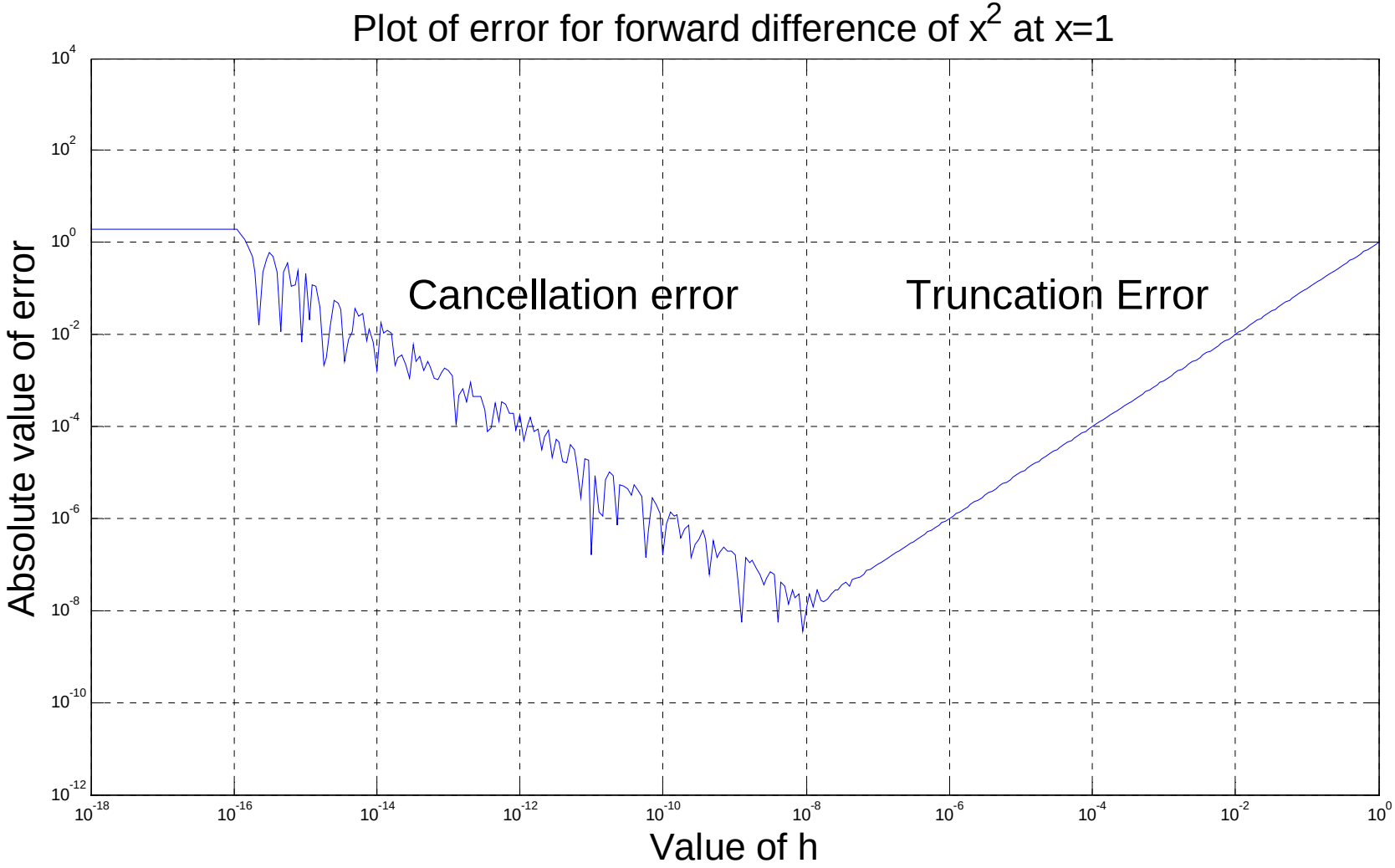
Rearranging using the first term yields a forward difference:

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

One can also do the Taylor series approximation about $-h$, which yields a backward difference:

$$f^{'}(x) = \frac{f(x) - f(x-h)}{h}$$

Effect of h on error of forward difference



Automatic differentiation (AD)

- Also referred to as algorithmic differentiation
- Can be used to find derivatives of program outputs for fixed inputs
- For some functions this can be done with the same computational cost as evaluating the function itself
- There are 2 types: forward mode or backward mode (adjoint)
- AD is not automatic symbolic differentiation or numerical differentiation

AD example

We'll evaluate the gradient of the following function:

$$y = x_1 x_2 + x_1 x_2 sin(x_2) + x_2 x_3^2$$

$$(\nabla y \ @ \ \mathbf{x} = [2 \ 0 \ 3]^T)$$

Its value and gradient at the given point are:

$$y(\mathbf{x}) = 0$$

$$\nabla y(\mathbf{x}) = [0 \ 11 \ 0]^T$$

AD forward mode example

Program trace	Forward mode trace
$v_{-1} = x_1 = 2$	$\nabla v_{-1} = [1\ 0\ 0]^T$
$v_{-2} = x_2 = 0$	$\nabla v_{-2} = [0\ 1\ 0]^T$
$v_{-3} = x_3 = 3$	$\nabla v_{-3} = [0\ 0\ 1]^T$
$v_1 = v_{-1}v_{-2} = 0$	$\nabla v_1 = v_{-1}\nabla v_{-2} + \nabla v_{-1}v_{-2} = [0\ 2\ 0]^T$
$v_2 = v_1 sin(v_{-2}) = 0$	$\nabla v_2 = v_1 cos(v_{-2})\nabla v_{-2} + \nabla v_1 sin(v_{-2}) = [0\ 0\ 0]^T$
$v_3 = v_{-2}v_{-3}^2 = 0$	$\nabla v_3 = 2v_{-2}v_{-3}\nabla v_{-3} + \nabla v_{-2}v_{-3}^2 = [0\ 9\ 0]^T$
$v_4 = y = v_1 + v_2 + v_3 = 0$	$\nabla v_4 = \nabla y = \nabla v_1 + \nabla v_2 + \nabla v_3 = [0\ 11\ 0]^T$

AD backward mode example

- Instead of taking a gradient for all input variables, we could take the derivative of each trace variable with the output variable
- This is done working backwards from y, to each of the input variables
- If we want to find $\overline{v}_1 \equiv \partial y / \partial v_1$ (adjoint) from a trace with $y = v_4$, we apply the chain rule:

$$
\begin{aligned}
y &= v_1 + v_2 + v_3 \\
v_2 &= v_1 sin(v_2) \\
\frac{\partial y}{\partial v_1} &= \frac{dy}{dv_4}\frac{\partial v_4}{\partial v_1} + \frac{dy}{dv_2}\frac{\partial v_2}{\partial v_1} \\
\overline{v}_1 &= \overline{v}_4 + \overline{v}_2 sin(v_{-2})
\end{aligned}
$$

- We can also do this in 2 steps (called accumulation):

$$
\begin{aligned}
\overline{v}_1 &= \overline{v}_4 \\
\overline{v}_1 &= \overline{v}_1 + \overline{v}_2 sin(v_{-2})
\end{aligned}
$$

AD backward mode example

Backward mode trace (using values from last trace and accumulation)

$$\overline{v}_4 = y = 1$$

$$y = v_1 + v_2 + v_3 = 0$$
$$\overline{v_3} = \overline{v_4} = 1$$
$$\overline{v_2} = \overline{v_4} = 1$$
$$\overline{v_1} = \overline{v_4} = 1$$

$$v_2 = v_1 sin(v_{-2}) = 0$$
$$\overline{v}_1 = \overline{v}_1 + \overline{v}_2 sin(v_{-2}) = 1$$
$$\overline{v}_{-2} = \overline{v}_{-2} + \overline{v}_2 v_1 cos(v_{-2}) = 9$$

$$v_3 = v_{-2} v_{-3}^2 = 0$$
$$\overline{v}_{-2} = \overline{v}_3 v_{-3}^2 = 9$$
$$\overline{v}_{-3} = \overline{v}_3 2 v_{-2} v_{-3} = 0$$

$$v_1 = v_{-1} v_{-2} = 0$$
$$\overline{v}_{-2} = \overline{v}_{-2} + \overline{v}_1 v_{-1} = 11$$
$$\overline{v}_{-1} = \overline{v}_1 v_{-2} = 0$$

Comments

- Amount of computations is 3 times higher if derivative is calculated using forward mode
- For functions of the form $f : \mathbf{R}^n \Rightarrow \mathbf{R}$, forward mode derivation takes $O(np)$ computations
- In reverse mode, it takes only $O(p)$ computations
- The opposite is true if the function is of the form $f : \mathbf{R} \Rightarrow \mathbf{R}^m$
- For general functions $F : \mathbf{R}^n \Rightarrow \mathbf{R}^m$ where a Jacobian is calculated, the two methods can be combined
- Doing it in the least amount of time referred to as the optimal Jacobian accumulation problem

AD of numerical scheme

For one view, the sensitivity with respect to light intensity at a node and given time step is as follows:

$$\frac{d\phi}{dU_s^n} = \sum_{r \in S} \frac{d\phi}{dU_r^{n+1}} \frac{\partial U_r^{n+1}}{\partial U_s^n} + \frac{\partial \phi}{\partial U_s^n}$$

$$\frac{\partial \phi}{\partial U_s^n} = (Y_s^n - U_s^n) \; if \; s \in M \; and \; n \in T$$

For all nodes:

$$\frac{d\phi}{dU^n} = \left[\frac{\partial U^{n+1}}{\partial U^n}\right]^T \frac{d\phi}{dU^{n+1}} + \frac{\partial \phi}{\partial U^n}$$

Differentiating the update rule:

$$\frac{\partial U^{n+1}}{\partial U^n} = A^{-1}B$$

$$\frac{d\phi}{dU^n} = B^T (A^{-1})^T \frac{d\phi}{dU^{n+1}} + \frac{\partial \phi}{\partial U^n}$$

AD of numerical scheme

For the final sensitivity:

$$\frac{d\phi}{d\theta_p} = \sum_n \sum_{r \in S} \frac{d\phi}{dU_r^n} \frac{\partial U_r^n}{\partial \theta_p}$$

For all optical properties being optimized:

$$\frac{d\phi}{d\theta} = \sum_n \left[\frac{\partial U^n}{\partial \theta} \right]^T \frac{d\phi}{dU^n}$$

Differentiating the update rule:

$$\frac{dA}{d\theta_p} U^{n+1} + A \frac{\partial U^{n+1}}{\partial \theta_p} = \frac{dB}{d\theta_p} U^n$$

$$\frac{\partial U^{n+1}}{\partial \theta_p} = A^{-1} \left[\frac{dB}{d\theta_p} U^n - \frac{dA}{d\theta_p} U^{n+1} \right] = A^{-1} X_p$$

$$\frac{\partial U^{n+1}}{\partial \theta} = A^{-1} X$$

AD of numerical scheme

The final expression for the sensitivity is:

$$\frac{d\phi}{d\theta} = \sum_n X^T (A^{-1}) \frac{d\phi}{dU^n}$$

This equation accumulates the contributions of:

$$\frac{d\phi}{dU^n} = B^T (A^{-1})^T \frac{d\phi}{dU^{n+1}} + \frac{\partial \phi}{\partial U^n}$$

With:

$$\frac{d\phi}{dU^N} = \frac{\partial \phi}{\partial U^N}$$

AD software

- A program can be differentiated using AD by transforming the code
- The most obvious one is source transformation
- The function is sent to a program that outputs the derivative of the function
- This method incurs the least overhead
- Another method is operator overloading
- In this method, language features are used that allow the programmer to use the same operation like + for different purposes depending on the types or classes being manipulated

AD software

	Programming language	
Type	C++	MATLAB
Operator overloading	■ FADBAD++ ■ RAD	■ ADMAT
Source transformation	■TAC++ (20000 € license!)	■ ADiMAT (only forward mode)

Conjugate gradient descent

- Effective method for solving large sparse matrix problems
- This is equivalent to computing the minimum of a quadratic from
- It can be shown that conjugate gradient descent is faster than steepest gradient descent in most cases
- The above results can be used to find the minimum for other nonlinear forms if their gradients can be computed

Quadratic form

A quadratic form is the following:

$$f(x) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} + c$$

$$\mathbf{x}^T\mathbf{A}\mathbf{x} > 0 \; for \; all \; \mathbf{x} \neq 0$$

If $\mathbf{A}$ is square, symmetric and positive definite, $f(x)$ is minimized by the solution to the related matrix problem:

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

Steepest gradient descent

The residual is the following:

$$\mathbf{r}_{(i)} = \mathbf{b} - \mathbf{A}\mathbf{x}_{(i)} = \mathbf{A}(\mathbf{x} - \mathbf{x}_{(i)}) = -\mathbf{A}\mathbf{e}_{(i)} = -f'(\mathbf{x}_{(i)})$$

To minimize $f(x)$, we can move in the direction of the residual to a new point:

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} + \alpha \mathbf{r}_{(i)}$$

Then ‘rinse and repeat’ until a norm of the residual is below a certain tolerance, usually chosen as a fraction of the same norm applied to the original residual:

$$||\mathbf{r}_{(i)}|| < \varepsilon ||\mathbf{r}_{(0)}||$$

Steepest gradient descent

The value of α should be a value that minimizes f along a line with direction $\mathbf{r}_{(i)} = -f'(\mathbf{x}_{(i)})$. A value where its derivative should be zero:

$$\frac{d}{d\alpha} f(\mathbf{x}_{(i)} + \alpha \mathbf{r}_{(i)}) = 0$$

Using the chain rule:

$$\frac{d}{d\alpha} f(\mathbf{x}_{(i+1)}) = f'(\mathbf{x}_{(i+1)})^T \frac{d}{d\alpha} \mathbf{x}_{(i)} = \mathbf{r}_{(i+1)}^T \mathbf{r}_{(i)}$$

And:

$$\alpha = \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{r}_{(i)}^T \mathbf{A} \mathbf{r}_{(i)}}$$

Steepest gradient descent

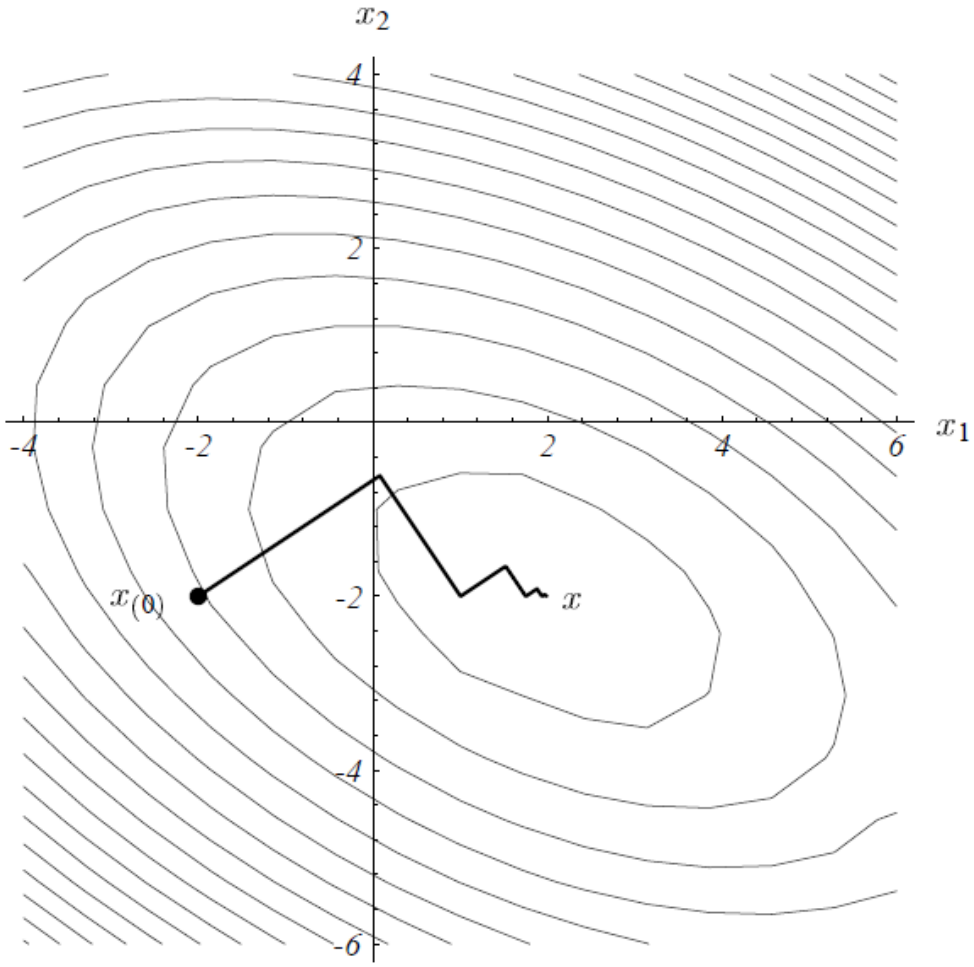


Diagram from [8].

Conjugate gradient descent

Conjugate gradient uses directions that are conjugate with respect to A:

$$\mathbf{u}^{\mathrm{T}}\mathbf{A}\mathbf{v} = 0$$

The solution can be expressed in mutually orthogonal basis:

$$\mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{p}_i$$

$$\mathbf{b} = \mathbf{A}\mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{A}\mathbf{p}_i$$

$$\mathbf{p}_k^{\mathrm{T}}\mathbf{b} = \mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_i = \alpha_k \mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_k$$

$$\alpha = \frac{\mathbf{p}_{(i)}^{T}\mathbf{b}_{(i)}}{\mathbf{p}_{(i)}^{T}\mathbf{A}\mathbf{p}_{(i)}}$$

Nonlinear conjugate gradient descent

The conjugate gradient descent method consists of 2 steps per iteration: line minimization and finding a new search direction.

For nonlinear minimization problems the same techniques can be used but convergence is not guaranteed.

For line search, the golden section search can be used:

$$\mu_{(i+1)} = \mu_i + 1.618\Delta\mu$$

This is repeated until a bound for a local minimum is found. Then quadratic interpolation is used to find the minimum.

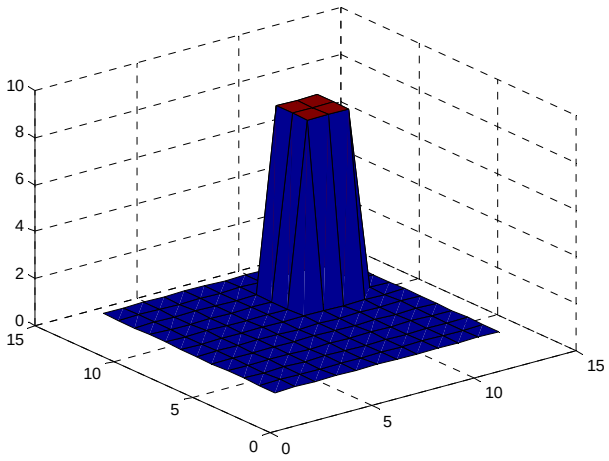
For a new search direction, the Polak-Riberie scheme can be used. Other methods that are used in other implementations of GIIR are quasi-Newton methods.

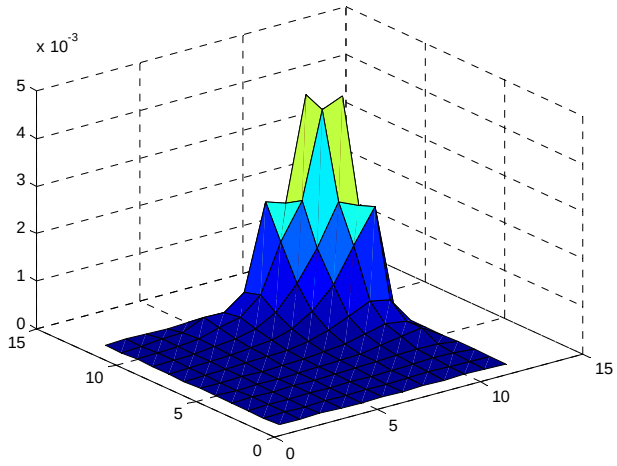
Results

- Implemented explicit, implicit and ADI finite differences for the problem setup in [2]
- The implicit method used a programmed version of bi-CGStab that takes advantage of A being pentadiagonal, making it faster than MATLAB's implementation for matrix sizes higher than ~400
- The explicit method has smallest discretization error per computation time
- However, since the noise is assumed higher than the discretization error, ADI should be chosen
- The AD method in [1] was implemented in MATLAB for an implicit scheme and tested using numerical differentiation

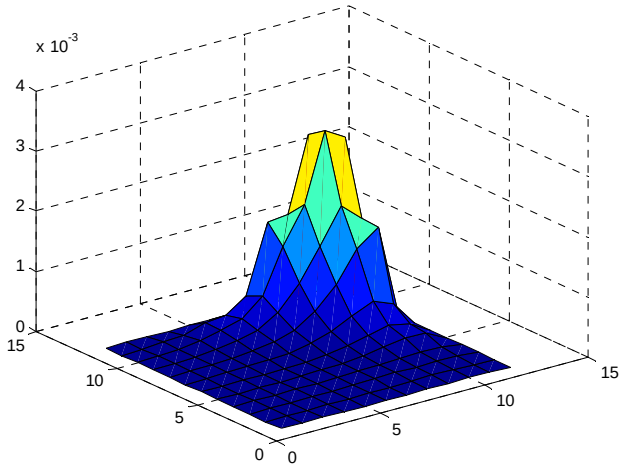
Results

Test optical property distribution and negative of the gradient using numerical differentiation and the AD method outlined in [1]:



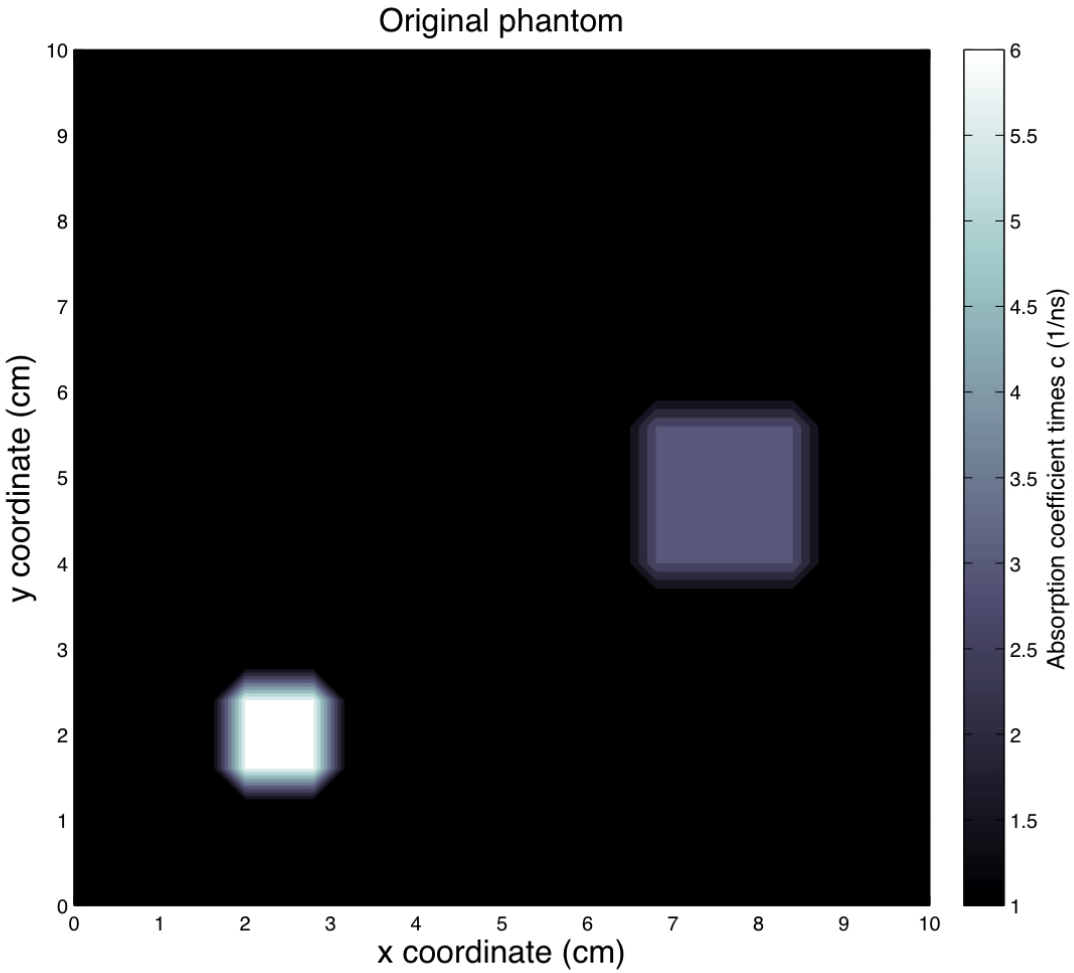


Computation time: 19.62s

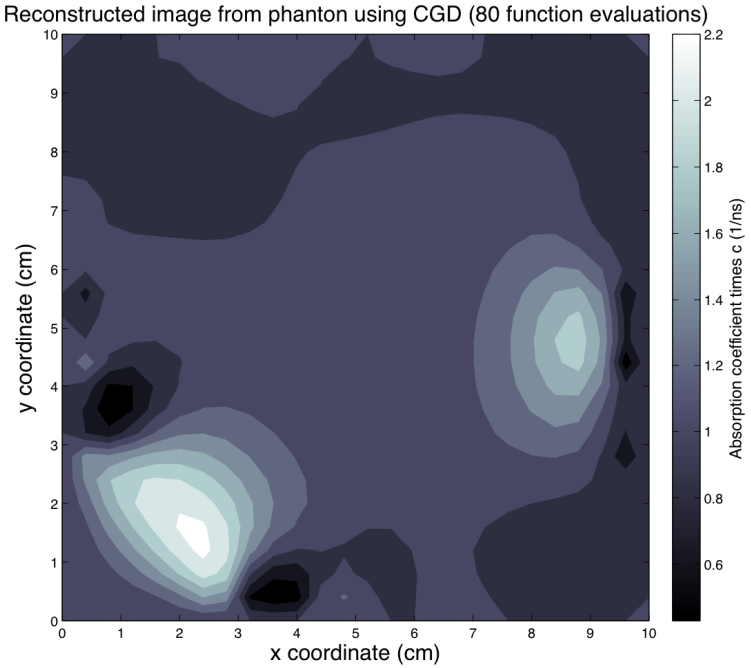


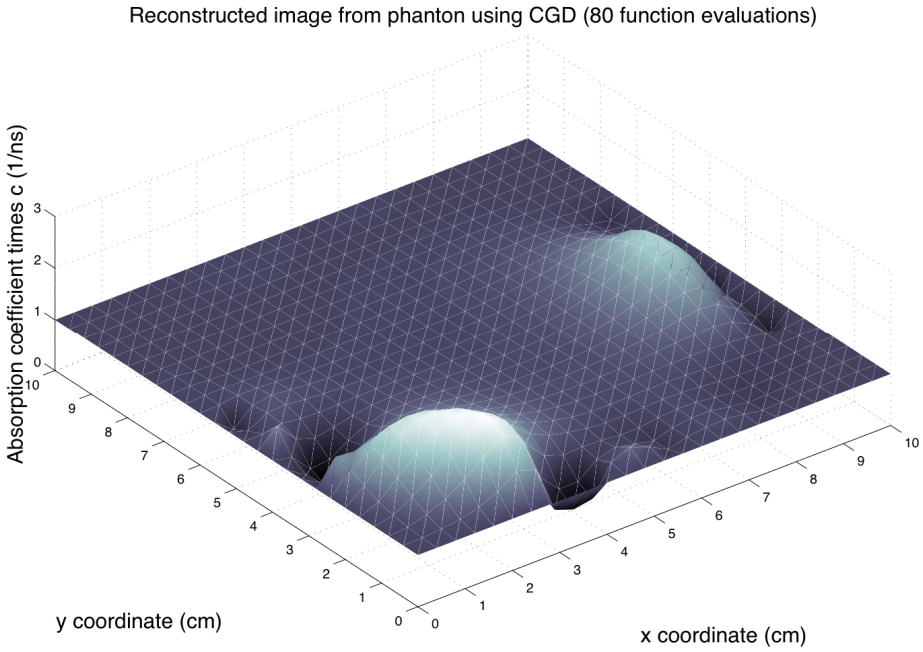
Computation time: 4.55s

Results


Original phantom
y coordinate (cm)
x coordinate (cm)
Absorption coefficient times c (1/ns)
0
1
2
3
4
5
6
7
8
9
10
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6

Results


Reconstructed image from phanton using CGD (80 function evaluations)
y coordinate (cm)
x coordinate (cm)
Absorption coefficient times c (1/ns)


Reconstructed image from phanton using CGD (80 function evaluations)
Absorption coefficient times c (1/ns)
y coordinate (cm)
x coordinate (cm)

References

1. Suhail S. Saquib et al. Model-Based image reconstruction from time resolved data. *SPIE proceedings series*, 3034(2):369-380, 1997.
2. Lihong V. Wang. *Biomedical Optics: Principles and Imaging*. Wiley-Interscience, 1st edition, 2007.
3. Andreas H. Hielscher et al. Gradient-Based Iterative Image Reconstruction Scheme for Time-Resolved Optical Tomography. *IEEE Transaction on Medical Imaging*, 18(3):262-271, 1999.
4. Robert D. Richtmyer. *Difference Methods for Initial-Value Problems*. Interscience Publishers, 2nd edition, 1967.
5. Ronald M. Errico. What is an Adjoint Model? Online copy at: http://www.cse.buffalo.edu/~peter/refs/DataAssimilation/Adjoint/Errico_1997.pdf

References

6. Andreas Griewank. *Evaluating derivatives: principles and techniques of algorithmic differentiation.* SIAM, 2nd edition, 2008.
7. FADBAD++. Source and manuals found online at: http://www.fadbad.com
8. Jonathan Richard Shewchuk. An introduction to the Conjugate Gradient Method Without the Agonizing Pain. Edition 1.25, 1994. Online copy at: http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf

Thank you for your time.
Questions?