A modular approach to simulation with automatic sensitivity calculation

Ken Hanson

Methods of Advanced Scientific Simulation
Los Alamos National Laboratory

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

Overview

- Bayes Inference Engine (BIE)
  - interactive tool for radiographic analysis; data-flow diagram
  - based on “transformation” modules
  - adjoint differentiation incorporated in modules
- Tomographic reconstruction - inverse problem
  - approach data fitting as optimization problem
- Optical tomography
  - solve time-dependent diffusion of IR light using adj. diff.
  - amenable to modular approach
- Hamiltonian MCMC requires gradient
  - is very efficient when combined with adjoint differentiation

Design Goals for Bayes Inference Engine

- Develop capability to accurately model radiographs
  - forward simulation model to include previously ignored radiographic effects (cone beam, tilt, spectrum, blur)
  - interactive, flexible, and adaptable model-building tool
  - must handle complex, nonlinear models
- Allow optimization of models to match radiographs
  - fast nonlinear optimization wrt many parameters
- Bayesian framework
  - solve ill-posed inverse (reconstruction) problems
  - uncertainty analysis

Advanced features of the BIE

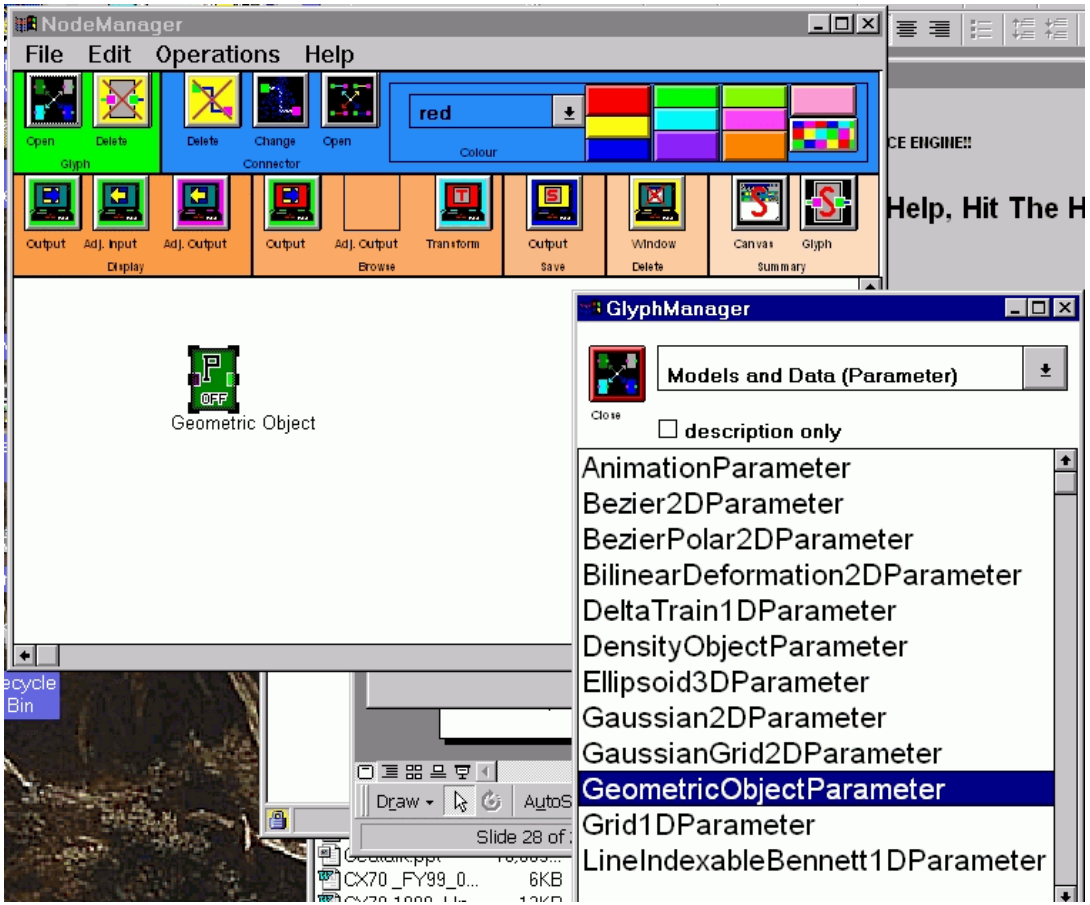
- Advanced image/object models
  - models constructed using interactive graphical interface
  - geometric description of regions via deformable models
  - variable density models for interior of each region
- Quasi-Newton gradient-based optimization
  - maximize posterior to obtain MAP estimate
  - Adjoint Differentiation In Code Technique (ADICT)
- Bayesian uncertainty analysis
  - Markov Chain Monte Carlo (MCMC)
  - posterior stiffness

Object-oriented design

- OO design is crucial to BIE success
  - flexibility in developing simulation models
  - easy to construct data-flow diagram; can **reverse** flow
  - encourages simplified approach to algorithm construction

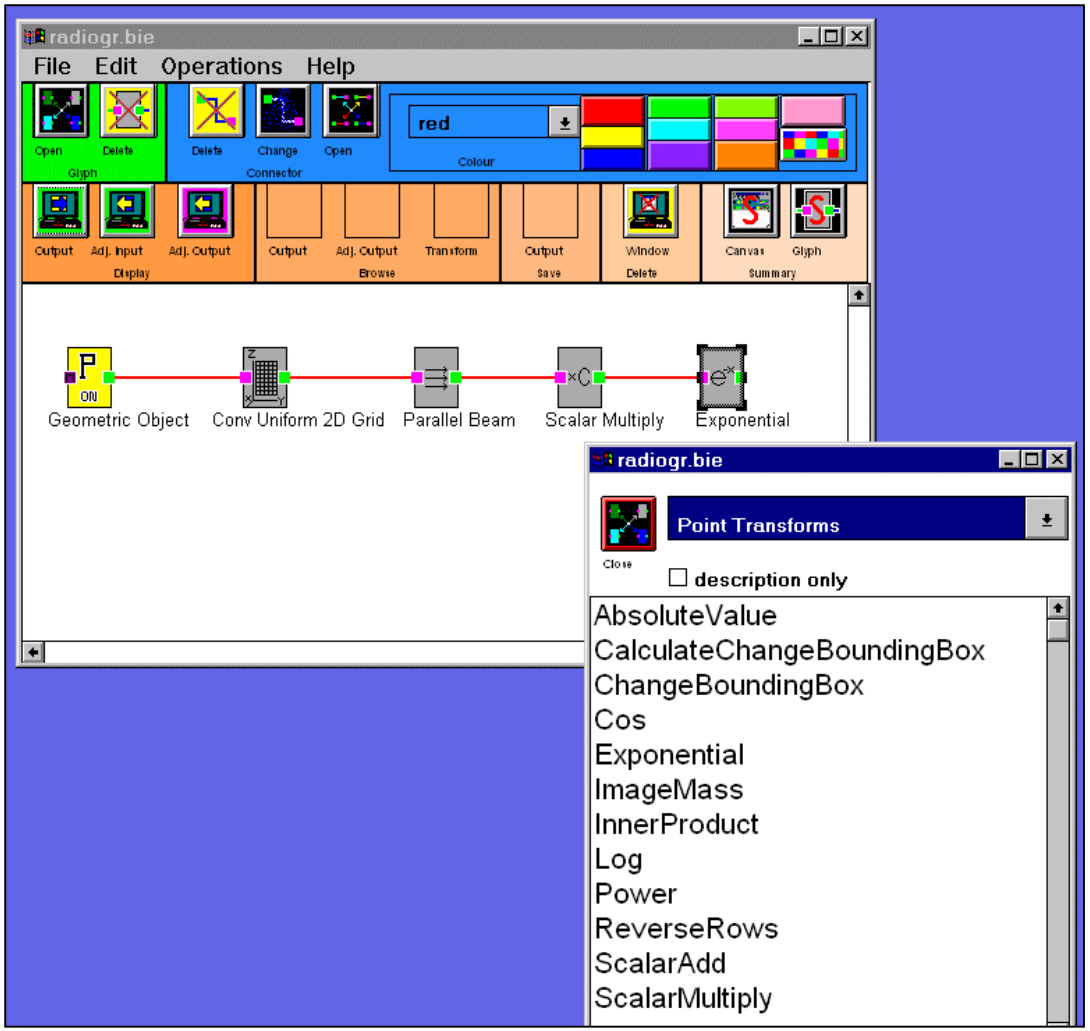
BIE Canvas

- Analyst interacts with BIE through the canvas
- Desired module is selected from list in GlyphManager
- Module placed on canvas; needed parameters set via pop-up panel



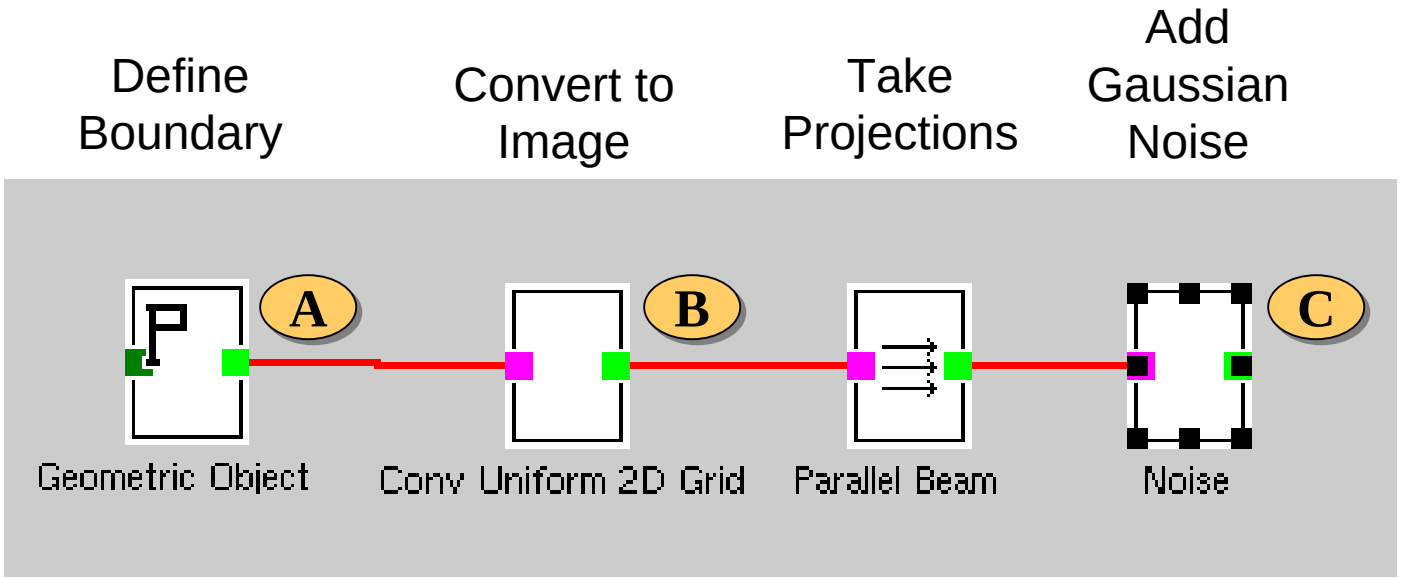
Graphical programming: data-flow diagram

- Analyst connects modules together to create data-flow diagram
- Objective is to create a model that accurately simulates the radiographic process and matches the data
- Data flowing through connections typically consist of large data structures, e.g., images



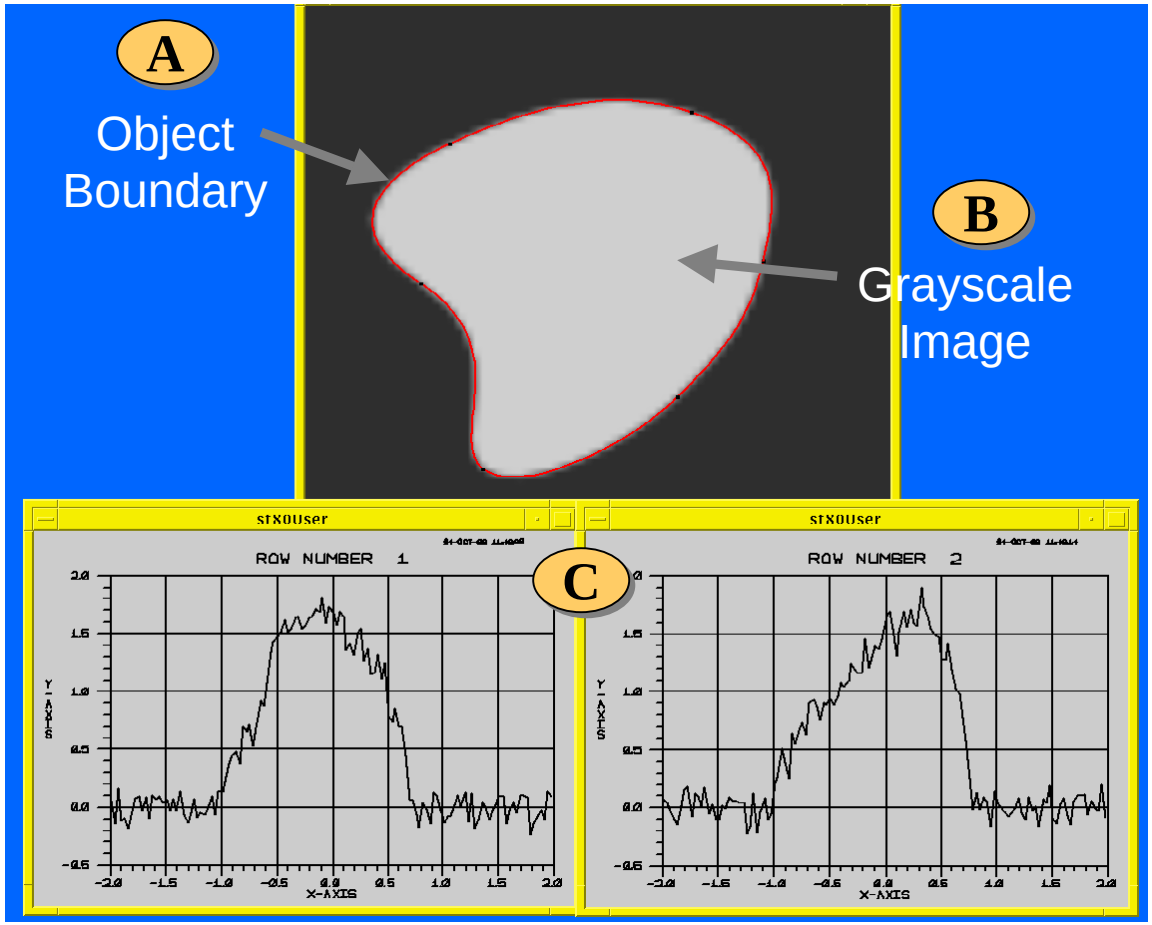
Create noisy projection data

- Create noisy projections of object whose boundary is defined in terms of geometric model
- Output of each module can be displayed



Create noisy projection data

– Outputs from data-flow diagram



Vertical Projection

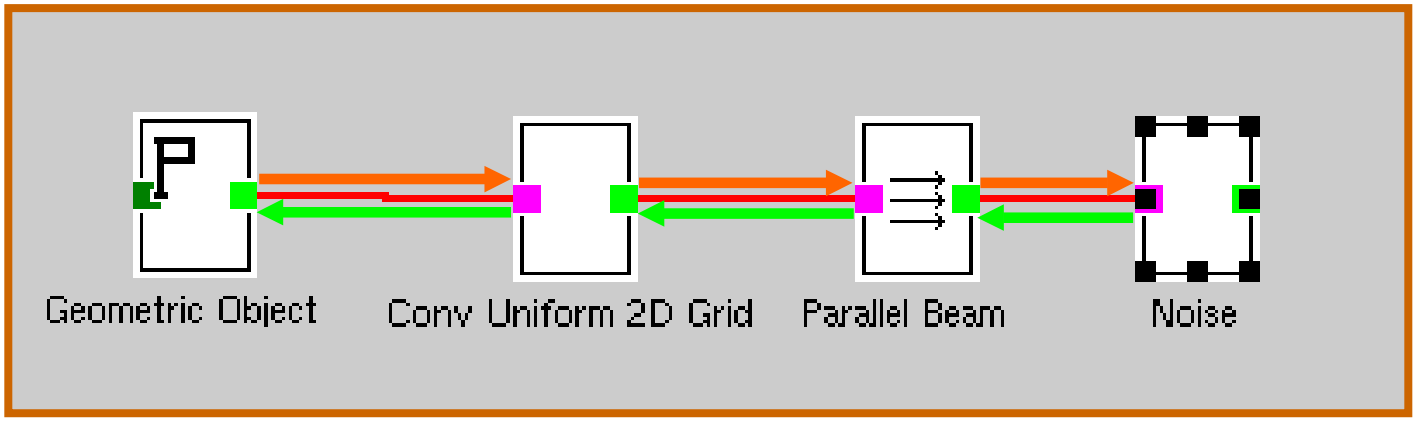
Horizontal Projection

Some design aspects of the BIE

- Modules provide functionality
  - each module transforms input data to obtain output data
  - created at run time (real-time instantiation)
  - interactive or “live”; mouse click to
    - view input and output data or set parameters
  - Model-View-Controller paradigm
    - function - displayed icon - response to mouse/keyboard
  - self contained, autonomous units
    - only responds to queries (messages)
- Connections provide the logic of sequence of transforms
  - link each module to its input and output modules

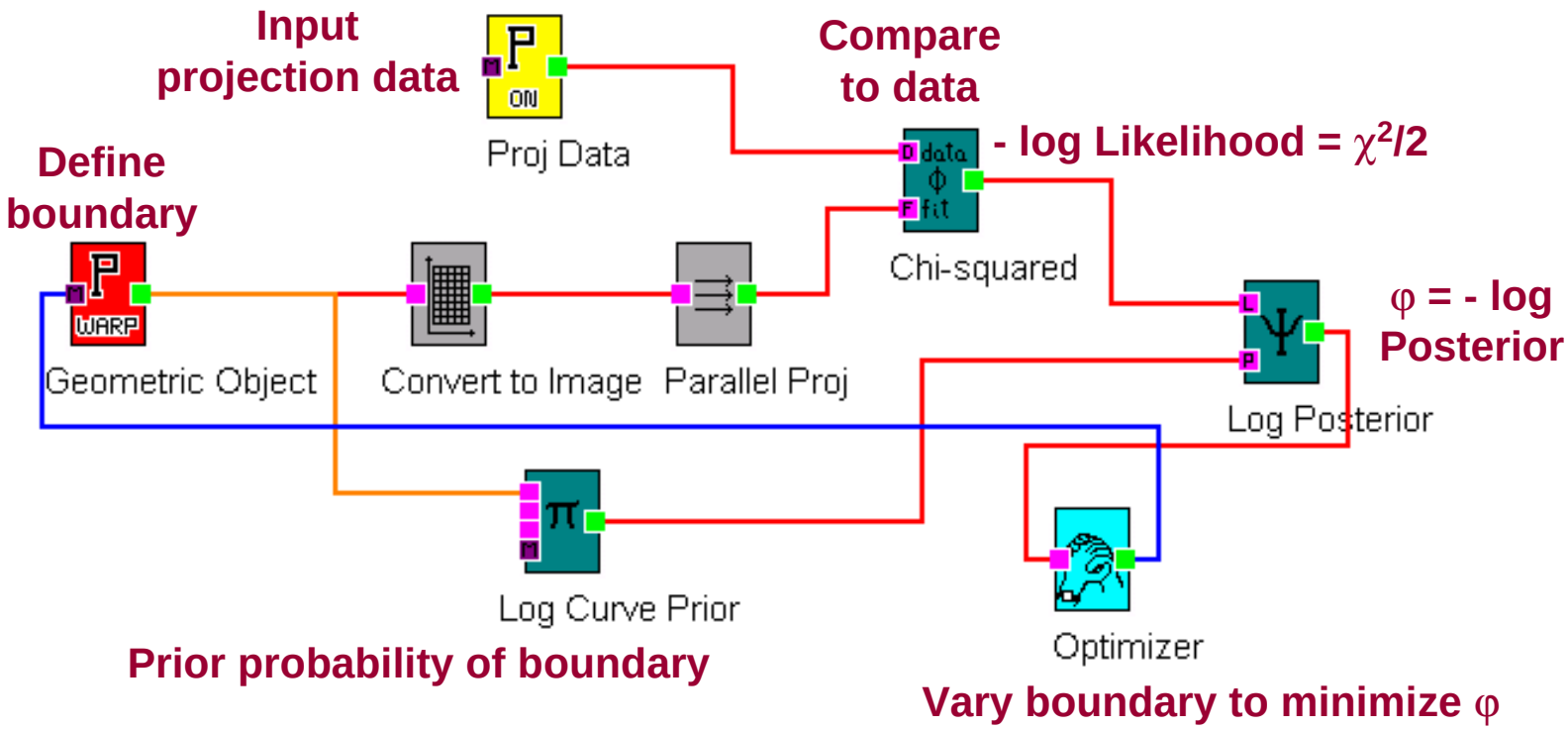
BIE modules are autonomous agents

- Each BIE module
  - responsible for its own calculation (and its adjoint)
  - responds to requests from modules connected to its output
  - in turn, asks its input modules for their outputs
- No need for supervision to choreograph calculations
- Facilitated by OO design
- Reverse flow of data is easy (needed for adjoint differentiation)



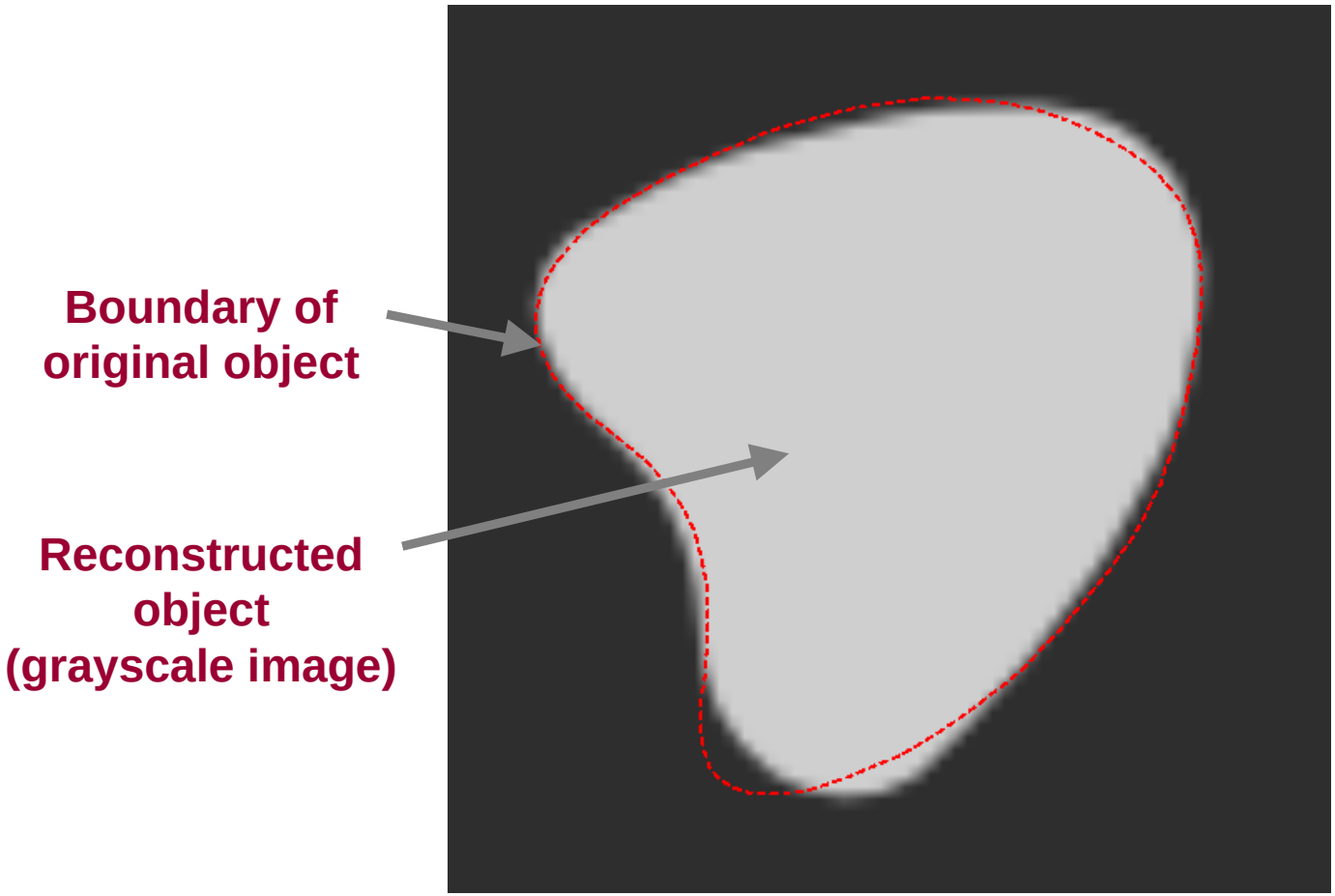
MAP reconstruction of object boundary

- Maximum posterior estimate obtained by optimization
- Gradients calculated by adjoint differentiation

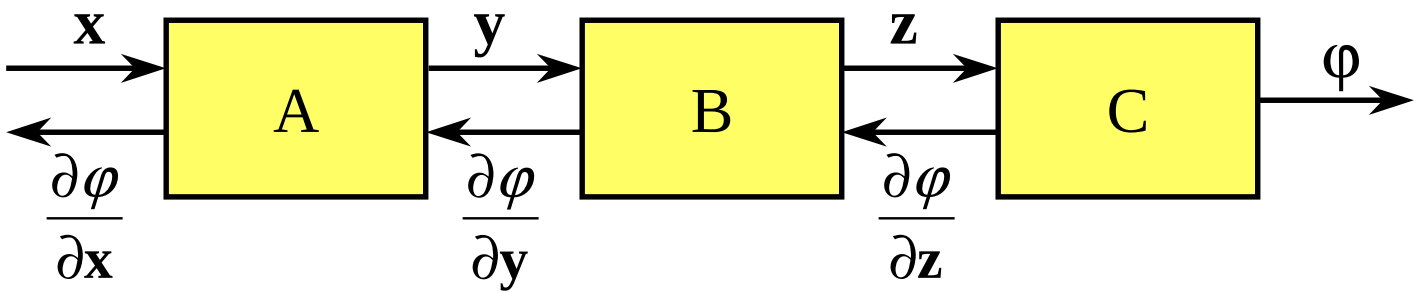


MAP reconstruction of object boundary

– Maximum posterior estimate obtained by optimization



ADICT
Adjoint Differentiation In Code Technique

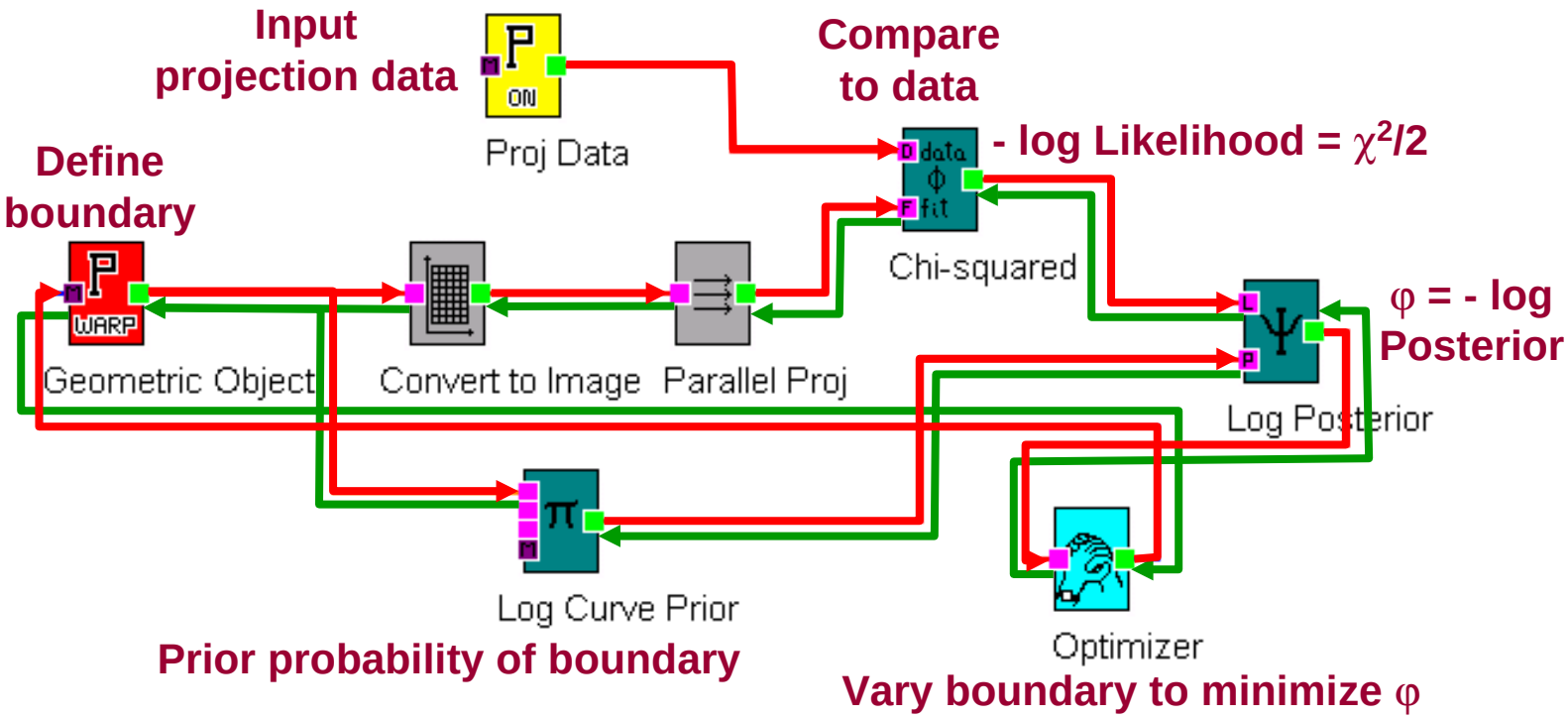


- For sequence of transformations that converts data structure **x** to scalar φ
- Derivatives $\frac{\partial \varphi}{\partial \mathbf{x}}$ are efficiently calculated in the reverse (adjoint) direction
- **Code-based approach**: logic of adjoint code is based explicitly on the forward code or on derivatives 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 **x**

All derivatives computed in time comparable to forward calculation

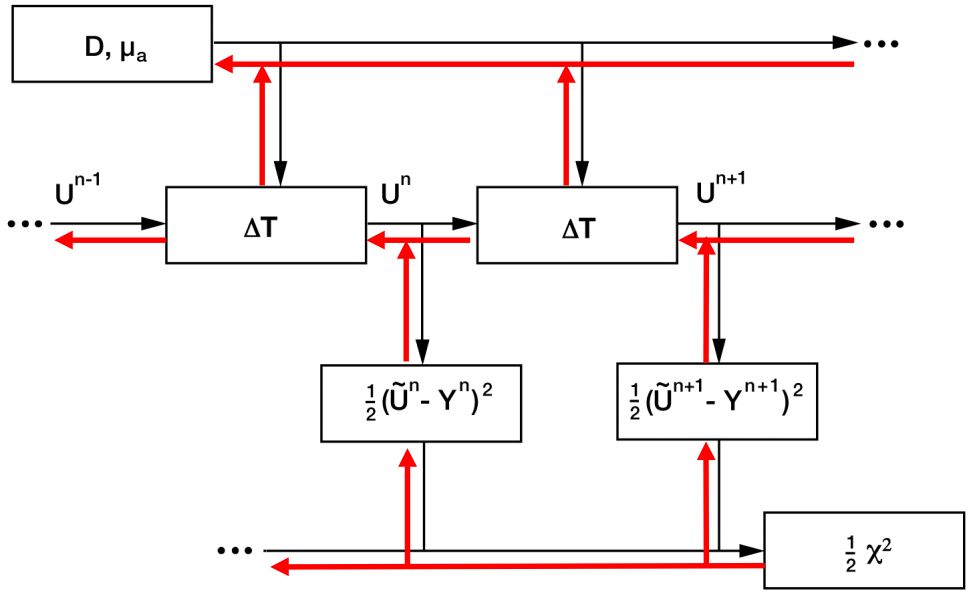
Adjoint differentiation - reverse data flow

- Red arrows show forward flow of data for simulation
- Green arrows show reverse flow required for adjoint calculation
- Optimizer (lower right) requests gradient from GeometricObject



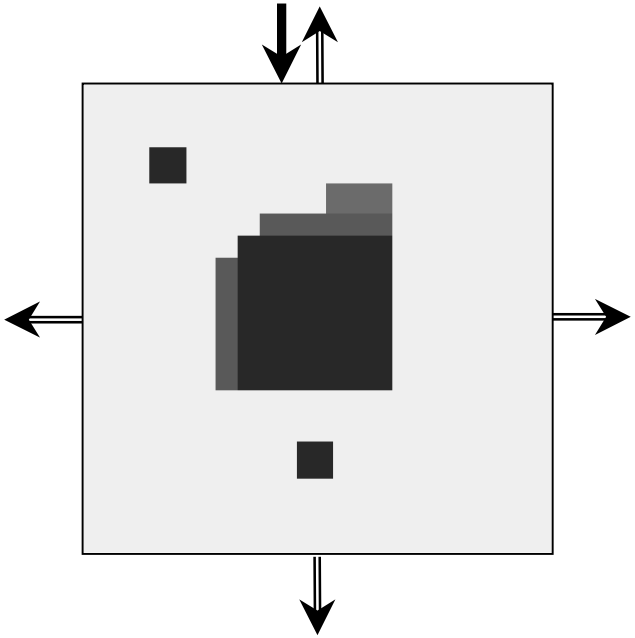
Adjoint differentiation in time-dependent diffusion calculation

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

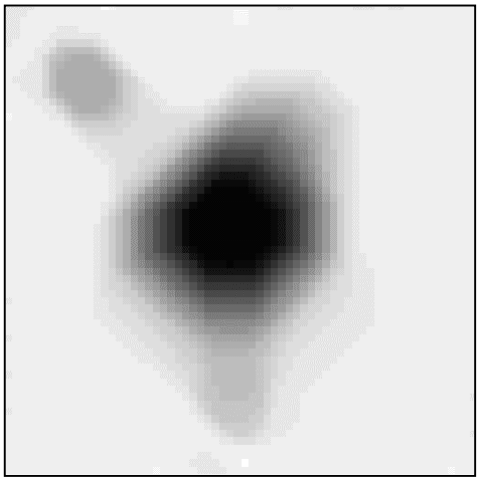


Optical tomographic reconstruction

Original diffusion-coef. image



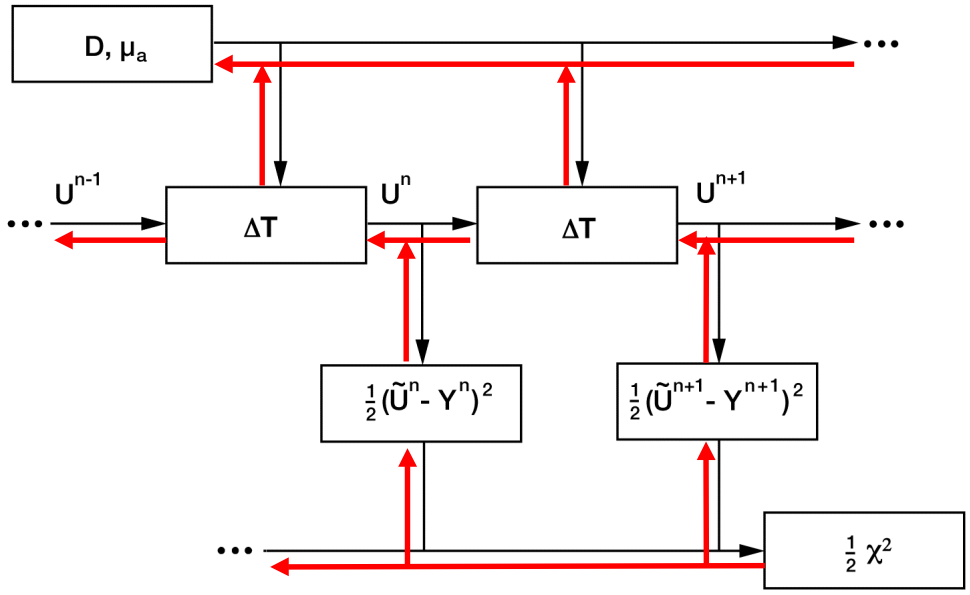
Reconstruction
(64 X 64 pixels)



- Measurements consist of 4 input pulse locations, 4 detector locations (intensity measured vs. time)
- Reconstruction based on
  - conjugate-gradient optimization of 4000 pixel values to obtain best match to light-diffusion data and **adjoint differentiation**

Modular approach to diffusion calculation

- Each box represents a computational module
  - there are just a few different types of modules (reuse!)
  - each module can be decomposed into set of submodules
- Sequence of time steps obtained by connecting modules
- This scenario suggests programming paradigm



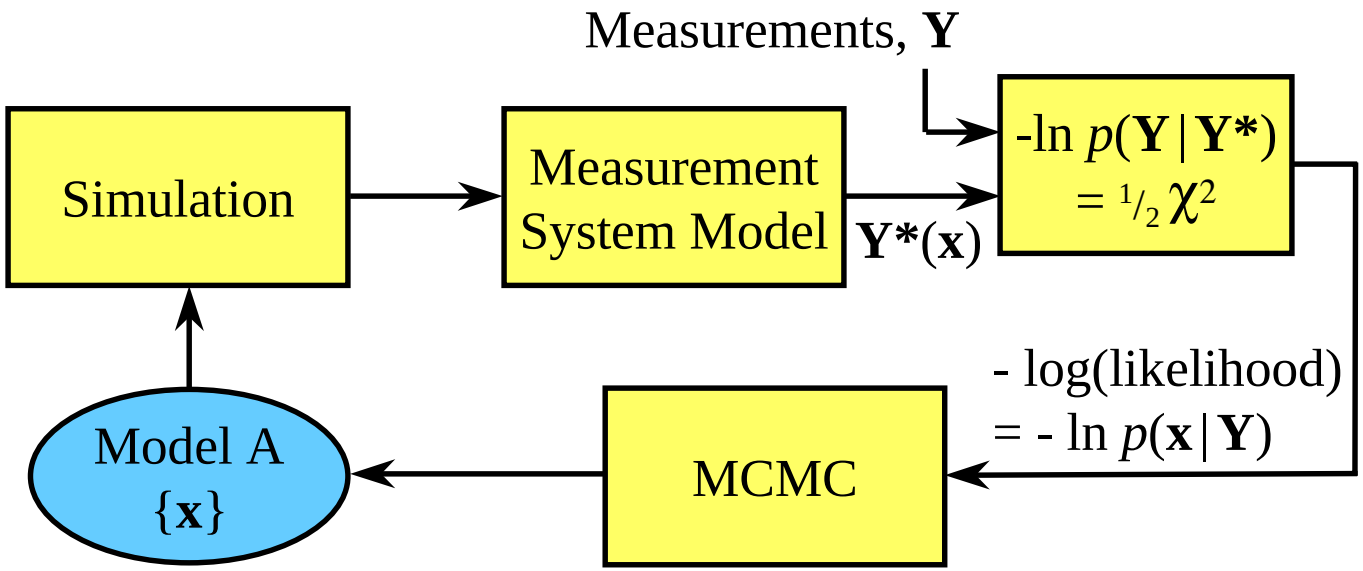
Modular approach to diffusion calculation

- Suggests building a toolkit to
  - simulate code
  - collection of modules to create simulation code
  - build in adjoint differentiation into each module
  - programming consists of connecting modules together
  - adjoint differentiation is automatic
  - gradient-based optimization

MCMC - problem statement

- Parameter space of n dimensions represented by vector $\mathbf{x}$
- Given an "arbitrary" **target** probability density function (pdf), $q(\mathbf{x})$, draw a set of random samples $\{\mathbf{x}_k\}$ from it
- Only requirement is that, given $\mathbf{x}$, one be able to evaluate $Cq(\mathbf{x})$, where C is an unknown constant;
  thus, $q(\mathbf{x})$ need not be normalized
- Although focus here is on continuous variables, MCMC applies to discrete variables as well

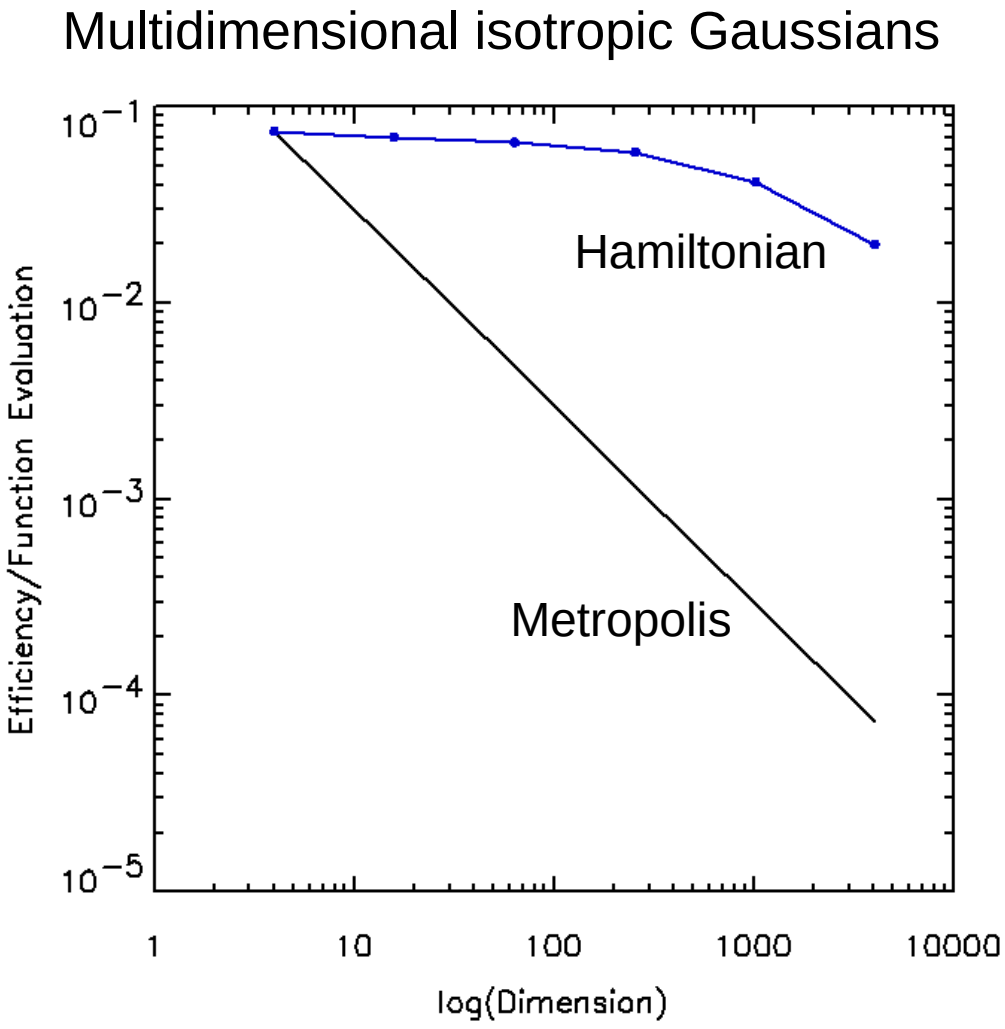
Use of MCMC in Bayesian data analysis



- - log(likelihood) value is result of calculation; function of model parameters **x**
- Markov Chain Monte Carlo (MCMC) algorithm draws random samples of **x** from posterior probability $p(\mathbf{x}|\mathbf{Y})$
- Produces plausible set of parameters {**x**}, which can be used to characterize posterior distribution, e.g., in terms of mean and covariance matrix

Hamiltonian MCMC

- MCMC efficiency versus number dimensions
  - Hamiltonian method: drops little with *n*
  - Metropolis method: drops as 0.3/*n*
- Hamiltonian method much more efficient at high dimensions
- But, **only if** gradients calculated using adjoint differentiation



Bibliography

- "An object-oriented optimization system," G. S. Cunningham et al., *Proc. IEEE Int. Conf. Image Processing*, *Vol. III*, pp. 826-830 (1994)
- "An interactive tool for Bayesian inference," G. S. Cunningham et al., *Rev. Prog. in Quant. Nondestr. Eval.* **14**A, pp. 747-754, (1995)
- "The Bayes Inference Engine," K. M. Hanson et al., *Maximum Entropy and Bayesian Methods*, pp. 125-134 (Kluwer, 1996)
- "A computational approach to Bayesian inference," K. M. Hanson et al., *Computing Science and Statistics* **27**, pp. 202-211 (Interface,1996)
- "Operation of the Bayes Inference Engine," K. M. Hanson et al., *Maximum Entropy and Bayesian Methods*, pp. 309-318 (Kluwer, 1999)

Find these under author's home page http://www.lanl.gov/home/kmh/

Acknowledgements

- Developers
  - Ken Hanson, Greg Cunningham, Bob McKee, John Pang, Xavier Battle, Marion Yapuncich, Duane Flamig, Jim Harsh, Connie Franklin, Salem Machaka, Igor Koyfman, George Jennings, David Wolf, Roger Bilisoly, Hanji Zhao
- Testers and critics
  - Barry Warthen, Pete Robins, Steve Calverley, Todd Kauppila, Judith Binstock
- General discussions
  - James Gee, Mark Zander, Kyle Myers, Bob Wagner, Julian Besag, Jim Gubernatis, Richard Silver, Steve Gull, John Skilling