System Overview

SCIRun / BioPSE
System Overview
Problem Solving Environments (PSEs)

Core
- Memory Mgmt
- Process Mgmt
- Framework
- Libraries

Chemistry
- Physics
- Biology
- Geoscience
- Medicine
- Astronomy
Integration and Interaction

Modeling

Simulation

Visualization

What If?

user guides

System Overview
SCIRun Goals

- System Overview
- Interactivity
- Usability
- Portability
- Utility
- Integration
- Extensibility

SCIRun

Core

Modeling
Simulation
Analysis
SCIRun and BioPSE

System Overview

Diagram:
- SCIRun
- BioPSE System
- Uintah System
- Uintah Package
- BioPSE Package
- CCA
- Fusion
- Virtual Soldier
Overview

• Computational Science
• Problem Solving Environments
• Dataflow
• Datatypes
• Software Organization
• Extensibility
• PowerApps
Elements of SCIRun

Visual programming Environment
Network Elements

- Dataflow Vocabulary
  - Module
  - Dataport
  - Datapipe
  - UI
Network Elements

- Dataflow Vocabulary
  - Module
  - Dataport
  - Datapipe
  - UI
Network Elements

• Dataflow Vocabulary
  • Module
  • Dataport
  • Datapipe
  • UI

System Overview

```java
void execute() {
  // get data from ports
  // get data from UI
  // … do work …
  // set data on UI
  // send data out ports
}
```
Module Status

- Run-time messages are sent to the module’s “log”

Startup messages

Log message indicator
- Gray: no messages
- Red: error
- Blue: warning/remark
Example Module: CastMatrix

System Overview

SCIRun/src/Dataflow/
Modules/Math/CastMatrix.cc
GUI/CastMatrix.xml
XML/CastMatrix.xml
Network Elements

- Dataflow Vocabulary
  - Module
  - Dataport
  - Datapipe
  - UI
Network Elements

- Dataflow Vocabulary
  - Module
  - Dataport
  - Datapipe
  - UI
Network Elements

- Dataflow Vocabulary
  - Module
  - Dataport
  - Datapipe
  - UI
Network Elements

- Dataflow Vocabulary
  - Module
  - Dataport
  - Datapipe
  - UI

System Overview
Network Elements

- Dataflow Vocabulary
  - Module
  - Dataport
  - Datapipe
  - UI

System Overview

Matrix
Field
ColorMap
Geometry
Network Elements

- Dataflow Vocabulary
  - Module
  - Dataport
  - Datapipe
  - UI
    - GuiVars
Network Design

- Dataflow Vocabulary
  - Module
  - Dataport
  - Datapipe
  - UI
- Send and Get
- GuiVars
- Scheduler
  - Dependencies
  - loops: send_intermediate
Network Design

- Dataflow Vocabulary
  - Module
  - Dataport
  - Datapipe
  - UI
- Send and Get
  - GuiVars
- Scheduler
  - Dependencies
  - loops: send_intermediate
Packages and Categories

System Overview
Managing Complexity: Subnets

System Overview
Managing Complexity: Subnets
Managing Complexity: Subnets

System Overview
Managing Complexity: Subnets

System Overview

[Diagram showing process flow with nodes labeled Gradient, Sample Field, Stream Lines, Direct Interpolate, and Show Field, connected with arrows and lines]
Managing Complexity: Subnets

System Overview
Managing Complexity: Subnets

System Overview
Managing Complexity: Annotations

System Overview

Using the zero to node as a ground reference, we build the finite element stiffness matrix for the incoming TetVolField. The TetVolField has a conductivity tensor at each element.

Solve the $A \times \phi = \text{flux}$ linear system. $A$ is the stiffness matrix, with node 0 pinned to ground, the flux vector is all zeroes except for the TetVol source and sink node positions, which have a 1 and -1 respectively (unit current passed between them). Solving for $\phi$ gives us the voltages everywhere at the nodes in the TetVolMesh.

Build the interpolant weights for the electrodes. For each node in the electrode field (second input field port), we select the single nearest node from the finite element mesh and assign it a weight of 1.

Build the element centroid leadfield matrix for the input finite element mesh (port 1) and electrode interpolant. We set the last electrode to ground, and iteratively pair it with each of the other electrodes. For each pair, right hand side column matrix that specifies the source-sink pair to have unit current. That right hand side to a solve matrix module, the system is solved, the gradient is computed, and the output is passed back to one column in our lead-field matrix.

Using the partial leadfield matrix we have built, we solve to get the voltages for each row of the system.
System Overview
show-text.net

System Overview
Dataflow Datatypes Revisited
Matrices: Class Hierarchy

- Matrix: base class
  - get, put, [ ], nrows, ncols, get_row, get_col, get_val, zero, mult, mult_transpose, print, {is_,as_},{sparse,dense,column}, cg_solve, bicg_solve, scalar_multiply

- SparseRowMatrix
  - int *rows, int *cols, double *a, int nnz;

- ColumnMatrix
  - double *data;

- DenseMatrix
  - double **data;
Matrices: External Libraries

• PETSc
  • Preconditioners: jacobi, bijacobi, sor, eisenstat, icc, ilu, asm, sles, lu, mg, spa, milu, nn, cholesky, ramg
  • Solvers: KSRICHARDSON, PSPCHEBYCHEV, KSPGG, KSPGMRES, KSPTCQMR, KSPBCG, KSPBGS, KSPTFQMR, KSPCR, KSPLSQR, KSPBICG, KSPPREONLY

• BLAS and Atlas
  • Faster linear-algebra via loop unrolling
Fields: Mesh + Data

Geometry
- Regular
- Irregular (basis)

Data
- int, float, double, ...
- Vector, Tensor, ...
- basis

Properties
- PointCloudField
- CurveField
- ScanlineField
- ImageField
- QuadSurfField
- TriSurfField
- LatVolField
- HexVolField
- TetVolField
Persistence

• Networks
• Serialize data for disk I/O
• Architecture independent
  • Smart pointers
  • Byte swapping
• Data files are (somewhat) human readable, but should not be generated / edited by anything other than SCIRun
  • Use “convert” programs
void Matrix::io(Piostream& stream)
{
    /* int version = */ stream.begin_class("Matrix", MATRIX_VERSION);
    PropertyManager::io(stream);
    stream.end_class();
}

void ColumnMatrix::io(Piostream& stream)
{
    /* int version = */ stream.begin_class("ColumnMatrix", COLUMNMATRIX_VERSION);
    Matrix::io(stream);

    stream.io(rows);
    if(stream.reading()) {
        data = scinew double[rows];
    }
    int i;
    for(i=0; i<rows; i++)
        stream.io(data[i]);
    stream.end_class();
}
Example Datasets

System Overview
Overview

- Computational Science
- Problem Solving Environments
- Dataflow
- Datatypes
- Software Organization
- Extensibility
- PowerApps
Source Tree Organization

SCIRun

Packages

BioPSE MatlabInterface

linux src docs

Core Dataflow StandAlone

Core: algorithms, datatypes, math, threads
Dataflow: network, modules, ports, scheduler
StandAlone: converters, utilities
SCIRun Categories

System Overview

Fields

• FieldsCreate
  – “sources” for new Fields
  – e.g. SampleLattice, FieldBoundary, ClipByFunction

• FieldsData
  – Just change data for an existing Field (Mesh untouched)
  – TransformFieldData, ManageFieldData, DirectMapping / ApplyMappingMatrix

• FieldsGeometry
  – Just change geometry for an existing Field (Data untouched)
  – Unstructure, HexToTet, QuadToTri
  – TransformField

• FieldsOther
  – Miscellaneous (FieldInfo, ChooseField, FieldMeasures, …)

Visualization

• ColorMaps
• Isosurfaces
Extensibility

• Leverage existing utilities
• Extensibility through *bridges*
Three Approaches

- **Data Level**
  - Command line tools to convert files
  - Communicating data across sockets
- **Library Level**
  - Teem, BLAS, ITK
- **Application Level**
  - Rewrite algorithms natively in SCIRun
**Converters**

- Convert between human-editable text (e.g. CVRTI .pts, .dat files) and SCIRun Persistent objects
- See examples in SCIRunData/convert-examples/
- Each converter gives you usage info if invoked without arguments:

  ```
  dmw stitch% TextToHexVolField
  
  Usage: TextToHexVolField pts hexes HexVolMesh [-noPtsCount] [-noElementsCount] [-oneBasedIndexing] [-binOutput] [debug]
  
  This program will read in a .pts (specifying the x/y/z coords of each point, one per line, entries separated by white space, file can have an optional one line header specifying number of points... and if it doesn't, you have to use the -noPtsCount command-line argument) and a .hex file (specifying i/j/k/l/m/n/o/p indices for each hex, also one per line, again with an optional one line header (use -noElementsCount if it's not there). The hex entries are assumed to be zero-based, unless you specify -oneBasedIndexing. And the SCIRun output file is written in ASCII, unless you specify -binOutput.
  ```

- Plug-ins for Readers / Writers
Data Import / Export
MatlabInterface

System Overview
Three Approaches

- **Data Level**
  - Command line tools to convert files
- **Library Level**
  - Teem, BLAS, ITK
- **Application Level**
  - Rewrite algorithms natively in SCIRun
ITK Integration
Teem
Overview

- Computational Science
- Problem Solving Environments
- Dataflow
- Datatypes
- Software Organization
- Extensibility
- PowerApps
All Modules are "Generic"
PowerApps: User-Friendly, Domain-Specific
Power Apps

System Overview
Problem specific applications

• Hide the complexities of dataflow
• Provide a simplified graphical user interface
• Focus on a specific task
Power Apps

System Overview

Processing Pane
Guide the user through specific processing steps
Power Apps

System Overview

Visualization Pane
Provide different visualization options
BioFEM

Encapsulation of the forward-fem network

- Change datasets
- Streamlines
- Isosurfaces
- Electrodes

BioTensor

Diffusion Tensor Imaging

Processing Steps

• Data Acquisition
• Registration
• Building Tensors

Visualization Options

• Planes
• Glyphs
• Isosurfaces
• Fibers

BioImage

Volume Rendering

Processing Steps

• Crop
• Resample
• Histogram Eq

Visualization Options

• Slices
• Window Width / Level
• MIPs
• Multi-dimensional Transfer Functions