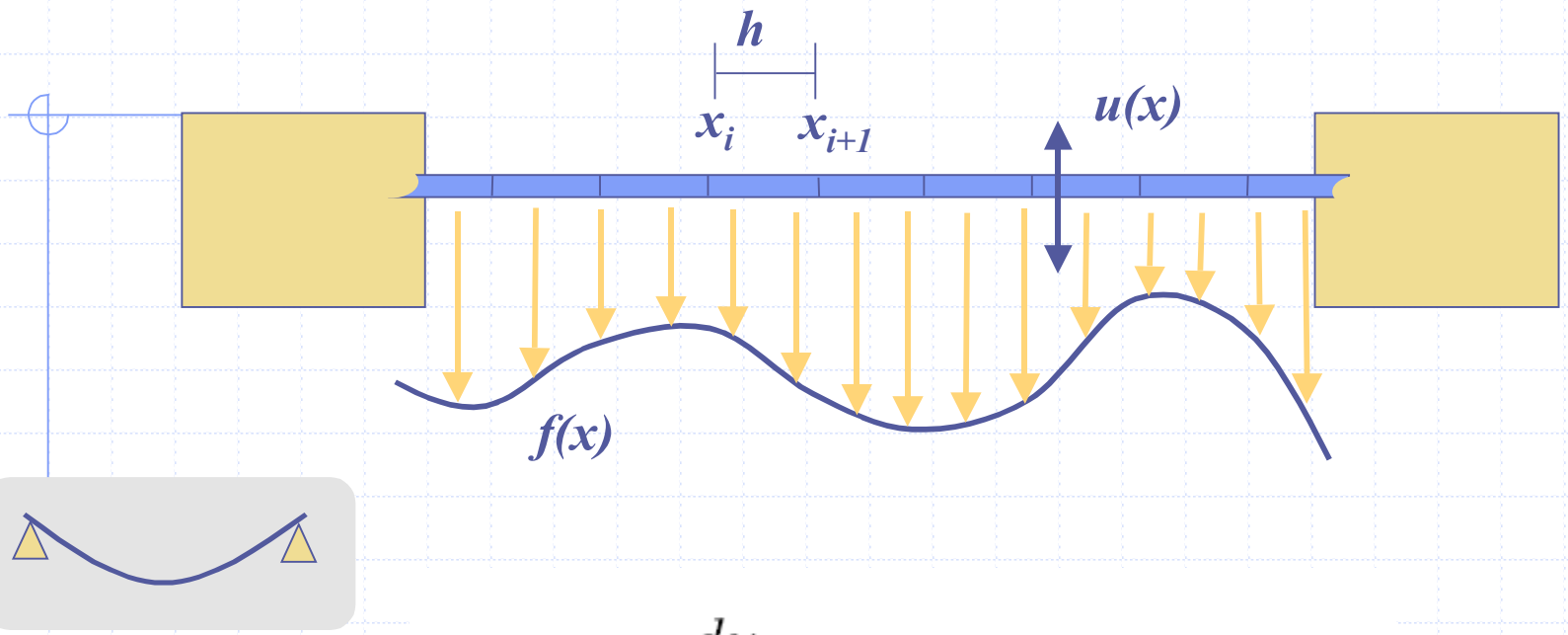
Fields

One of the fundamental concepts of mathematical physics is that of a *field* – i.e., a spatial distribution of some mathematical object representing a physical quantity.

The power of this idea is that it allows the modeling of a number of very important phenomena… "electromagnetism", "thermal conduction", "fluid dynamics", "solid mechanics" – to name a few – and of the combinations thereof.

Claudio Mattiussi *The Finite Volume, Finite Element, and Finite Difference Methods as Numerical Methods for Physical Field Problems*

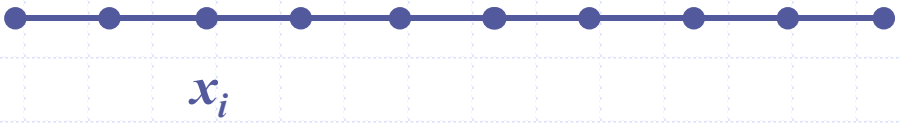


$$u = 0, \quad \frac{du}{dx} = 0 \qquad \text{for } x \in \{0, 1\}$$

$$-E\frac{d^2u}{dx^2} + S\frac{d^4u}{dx^4} = f \qquad \text{for } 0 \leq x \leq 1$$

Discrete Formulation

Discretization

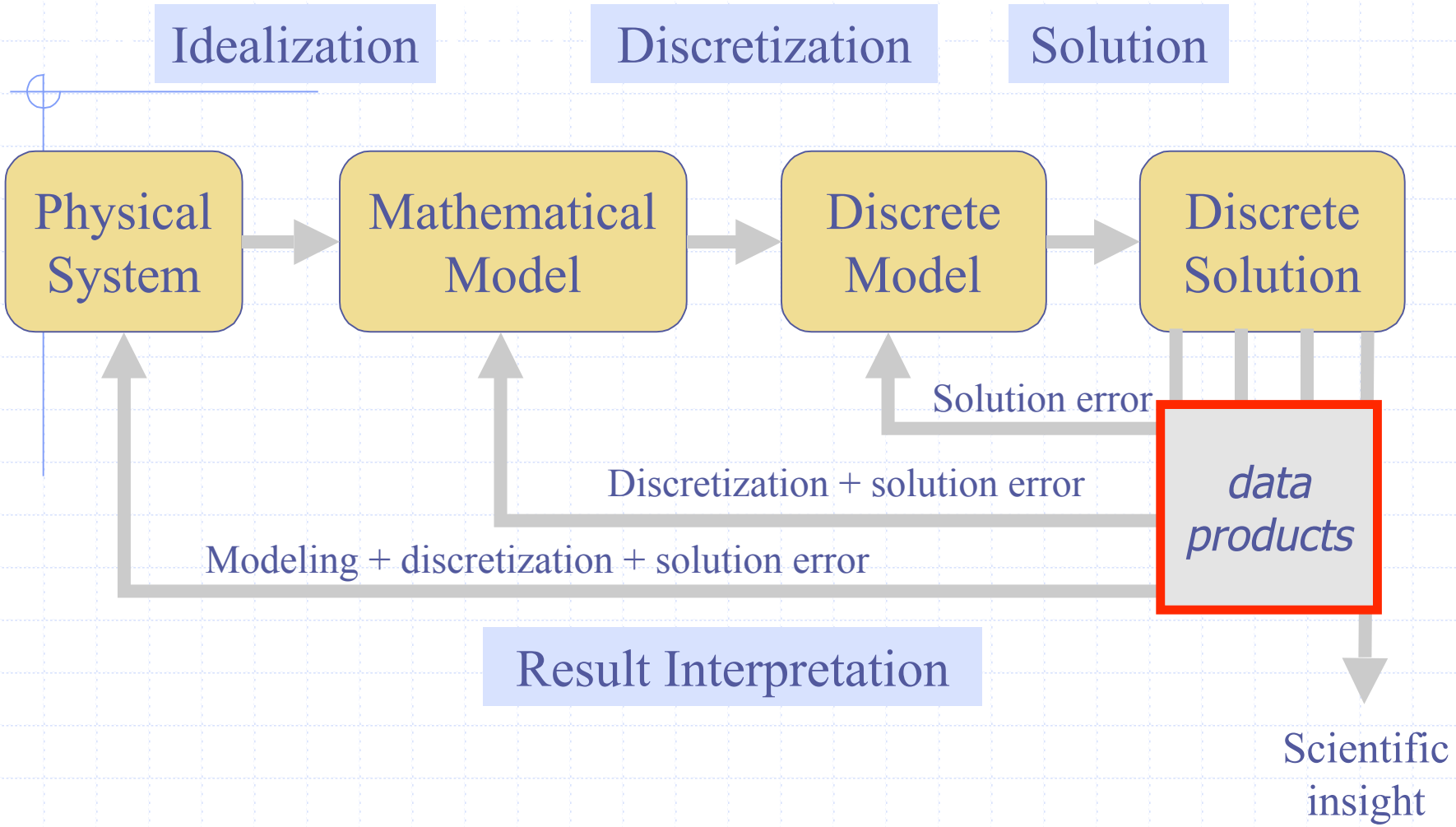


$$E\frac{1}{h^2}(-u_{i+1} + 2u_i - u_{i-1}) + S\frac{1}{h^4}(u_{i+2} - 4u_{i+1} + 6u_i - 4u_{i-1} + u_{i-2}) = f(x_i)$$

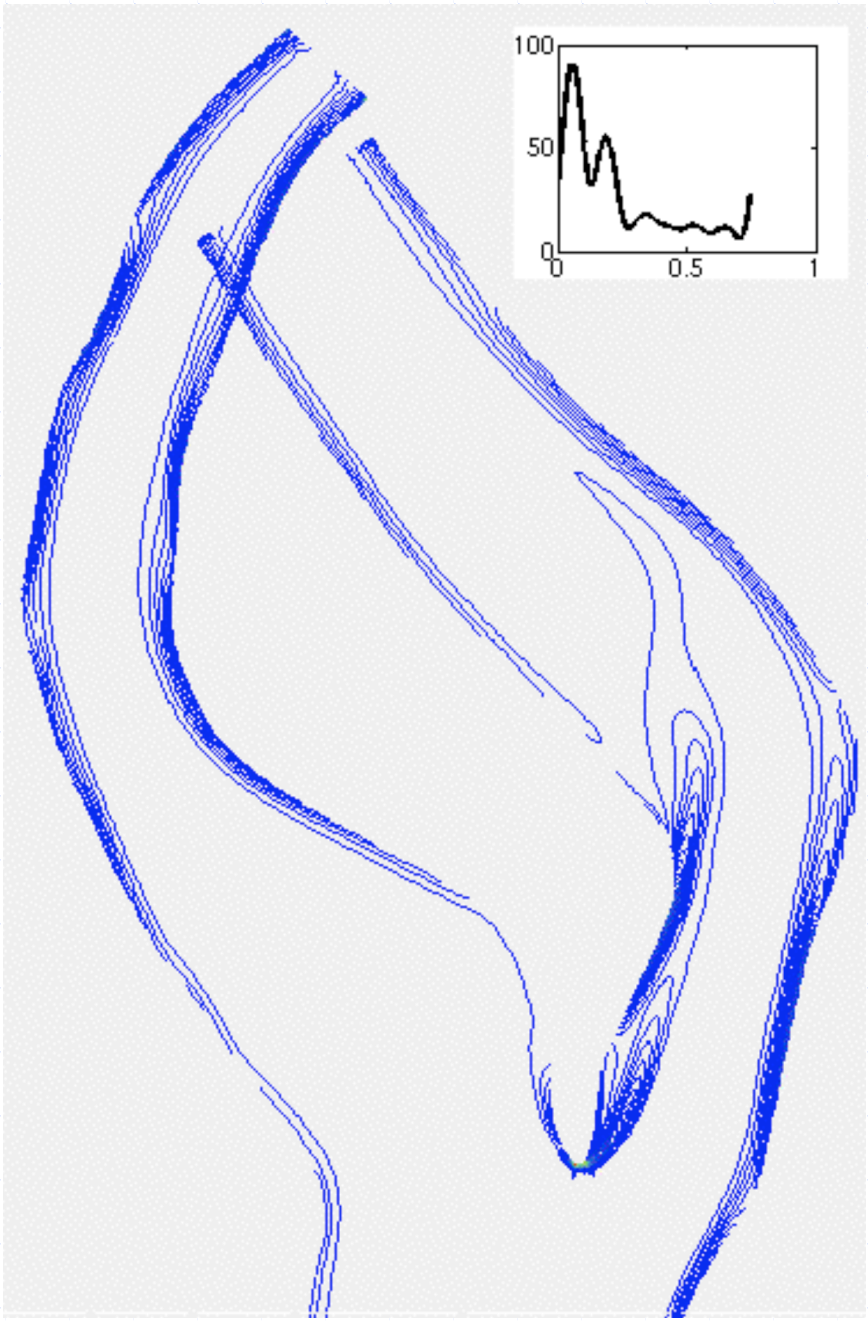
Solve

$$A = E\frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix} + S\frac{1}{h^4}\begin{bmatrix} 6 & -4 & 1 & \cdots & 0 \\ -4 & 6 & -4 & \cdots & 0 \\ 1 & -4 & 6 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 6 \end{bmatrix}$$

Results Interpretation



adapted from lecture notes by Carlos Felippa, University of Colorado



Transition in a Stenosed Carotid Artery

"…magnitude of vorticity…."

"…volume flow rate…."

"…close to the center plane…"

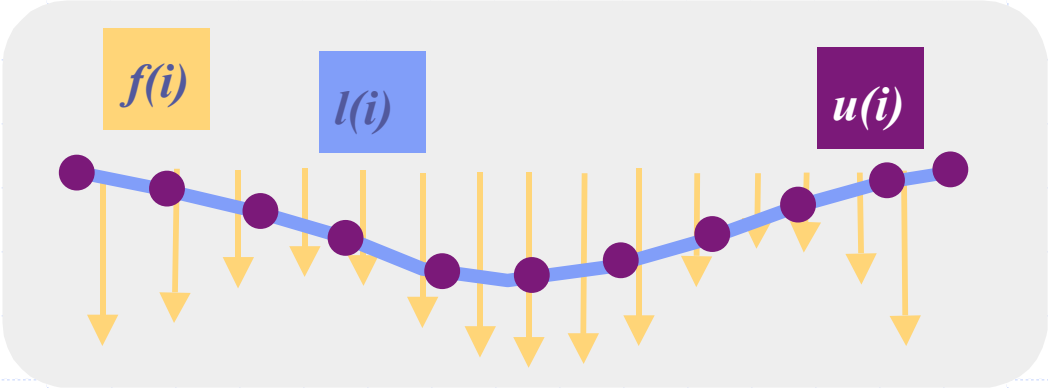
"…contours represent the vorticity distribution renormalized by the min and max at each instant in time."

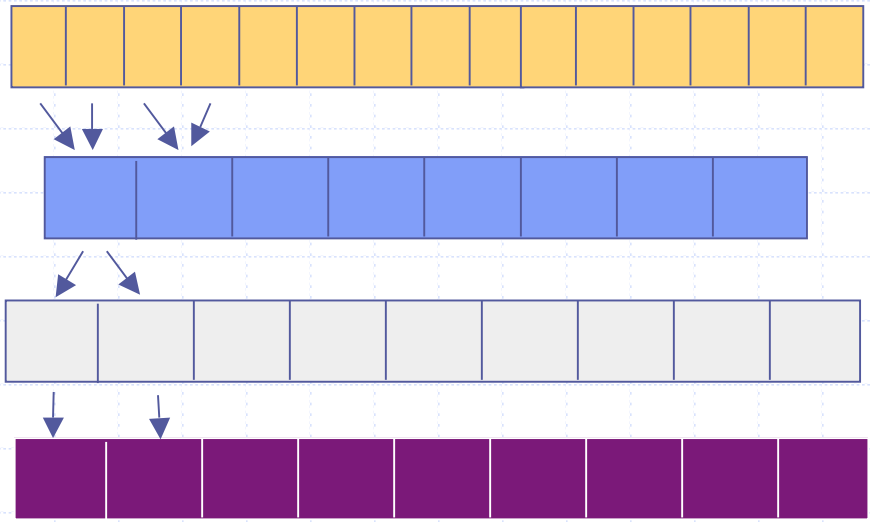
Bassiouny, Fischer, et al.
University of Chicago
Argonne National Laboratory
www-unix.mcs.anl.gov/~fischer/car_trans.html

Roadmap

- *Structured vs. Unstructured Meshes*
- *Querying Unstructured Meshes*
- *Manipulating Unstructured Meshes*

Matlab, C, netCDF, HDF





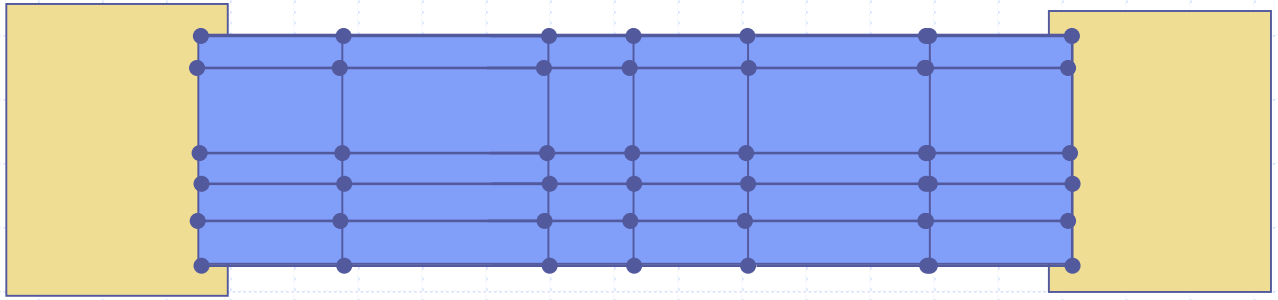
$f(k)$ *inputs defined on a different grid*

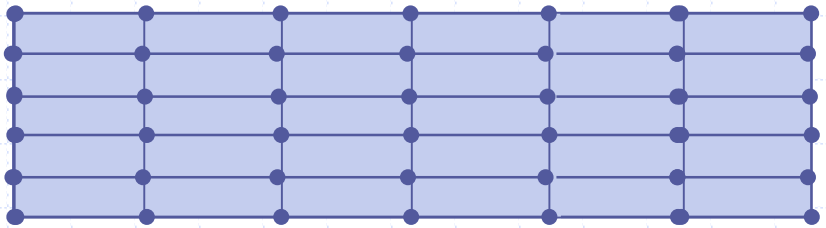
$l(j)$ *edge lengths*

$x(i)$ *non-uniform node spacing*

$u(i)$

Higher Dimensions



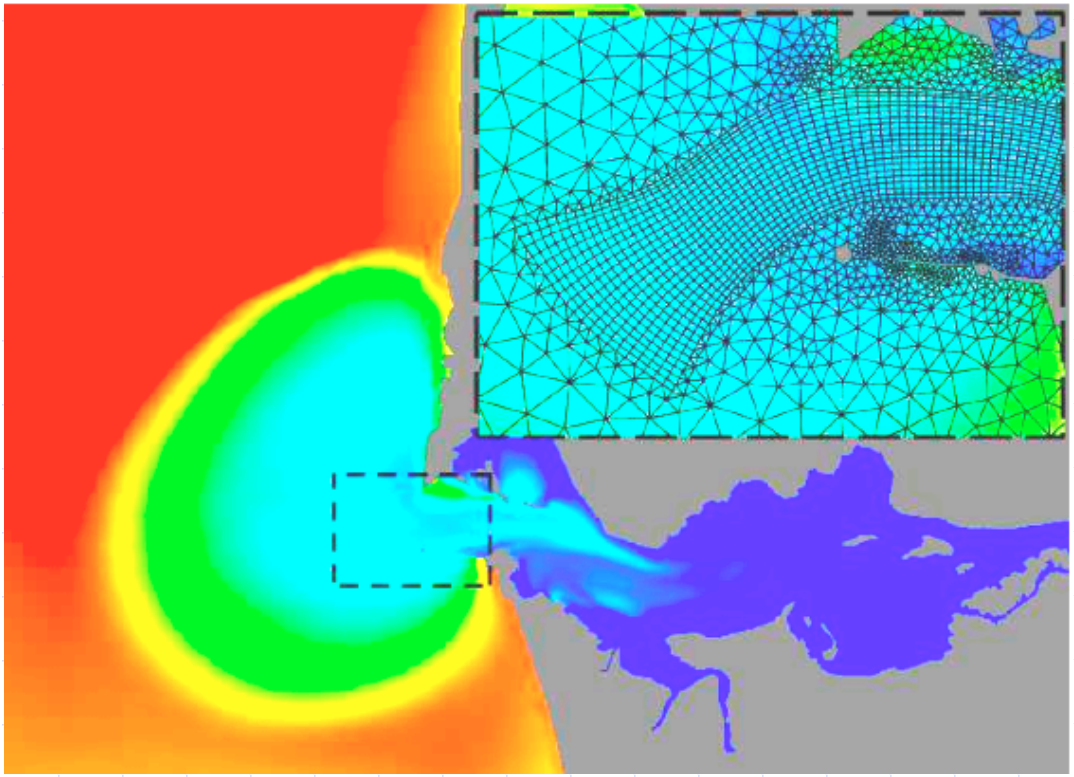


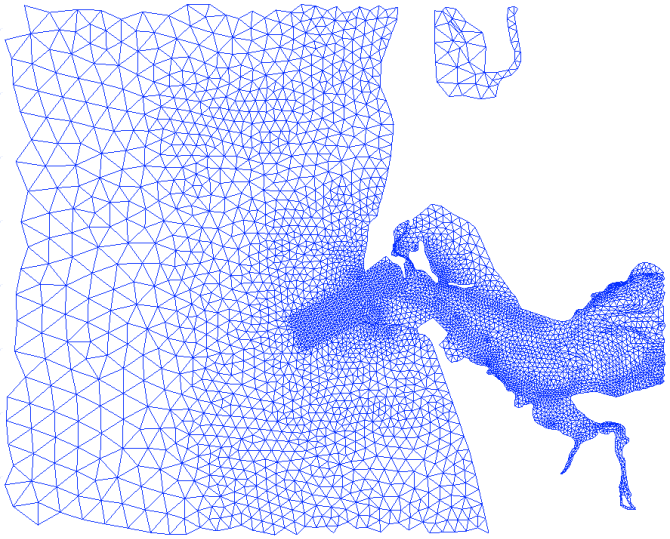
"structured grids"

Structured Grids are Easy

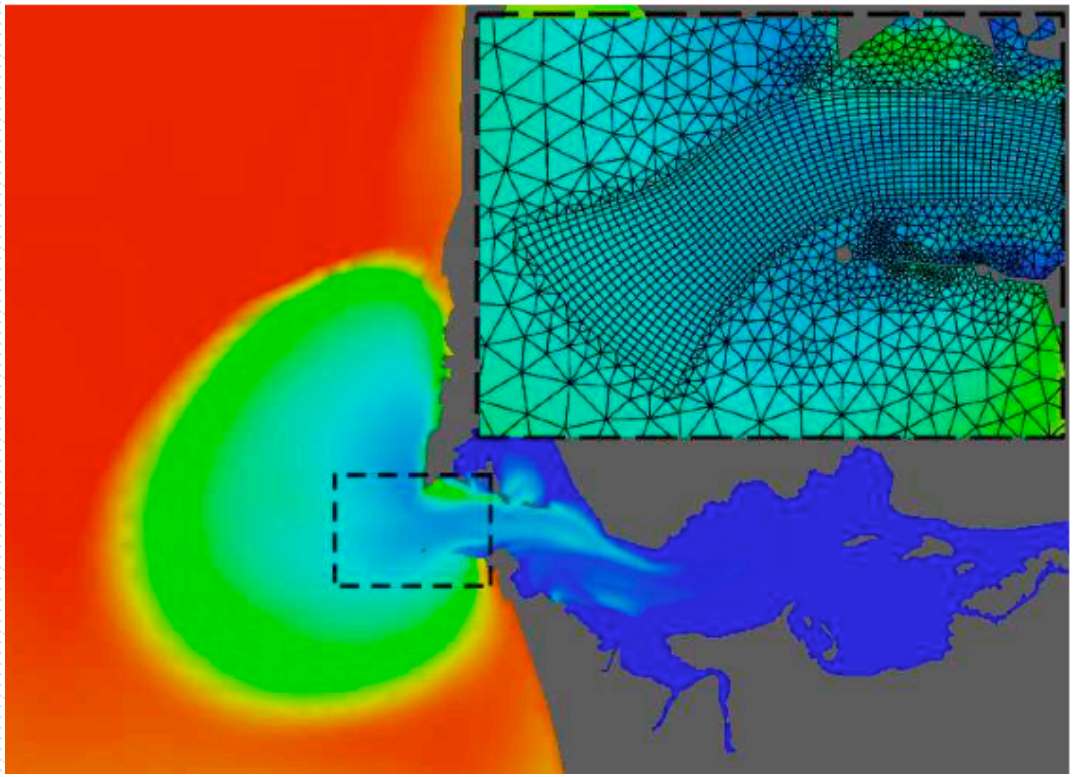
- The data model
  (Cartesian products of coordinate variables)
- implies a representation,
  (Multidimensional arrays)
- an API,
  (Reading and writing subslabs)
- and an efficient implementation
  (Address calculation using array "shape")

Unstructured Grids





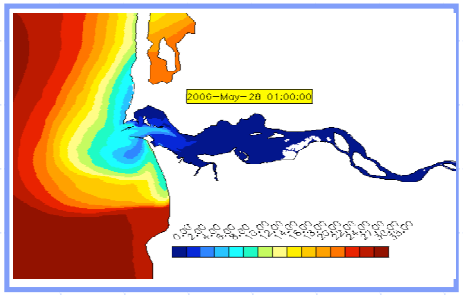
Unstructured Grids



Coastlines are not rectilinear, unfortunately

Structured grids do a poor job of modeling complex features and complicate multi-scale analysis.

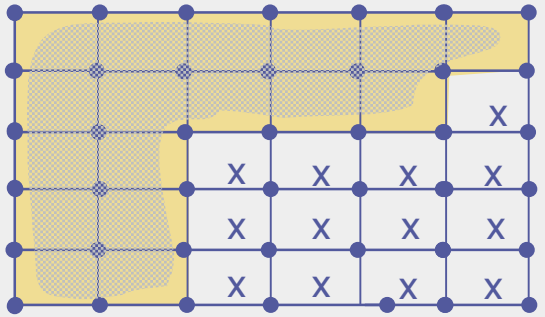
"unstructured grids", however, complicate processing.

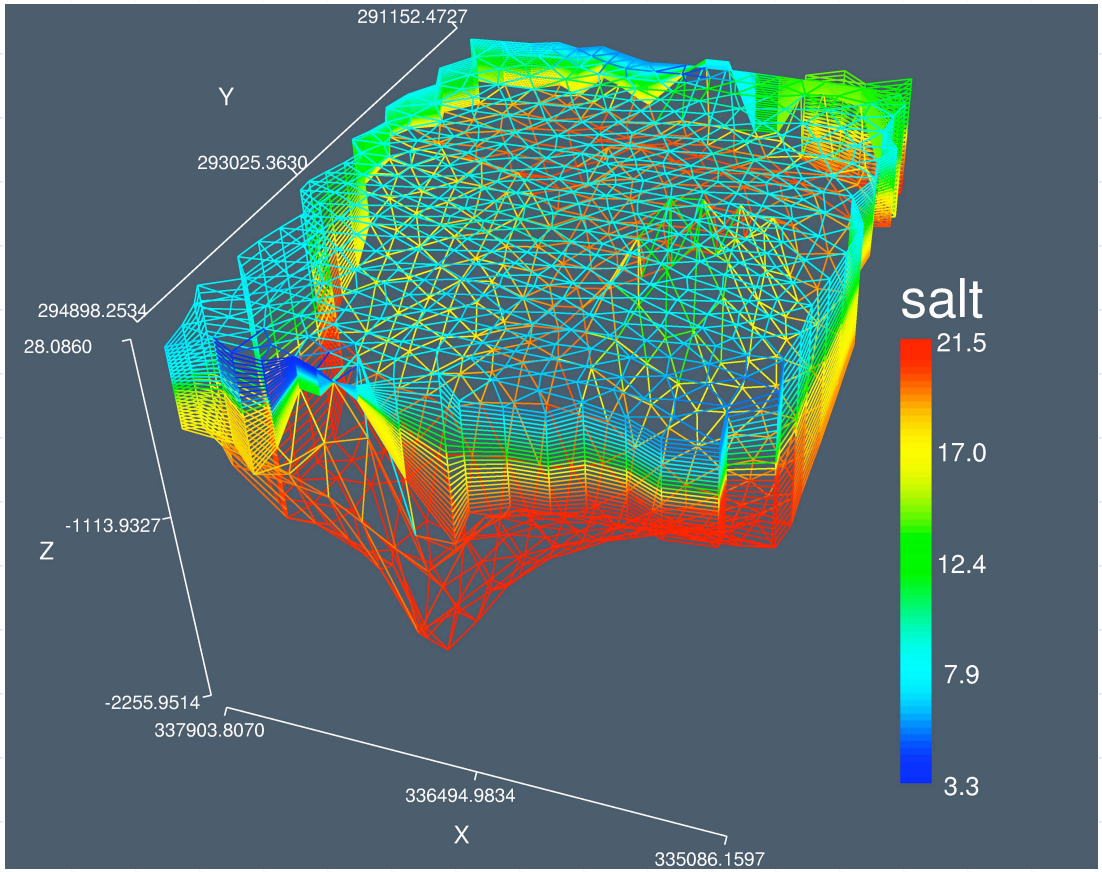


Columbia River Estuary

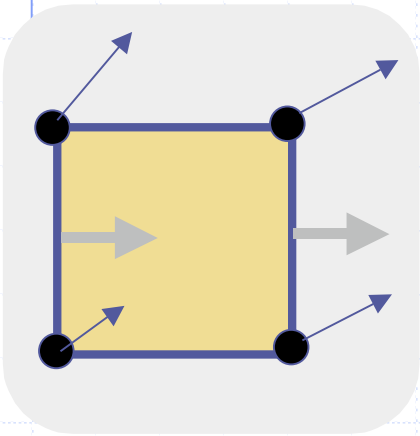
red = high salinity (~34psu)

blue = fresh water (~0 psu)



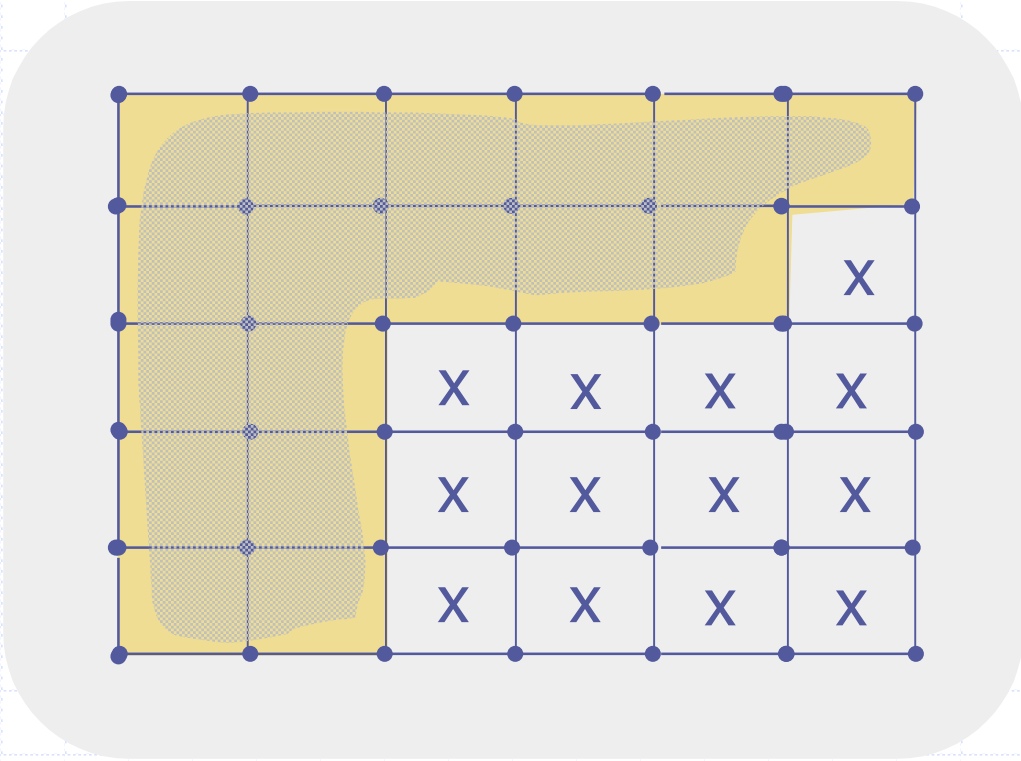

291152.4727
Y
293025.3630
294898.2534
28.0860
Z
-1113.9327
-2255.9514
337903.8070
336494.9834
X
335086.1597
salt
21.5
17.0
12.4
7.9
3.3

Structured Grids are Insufficient

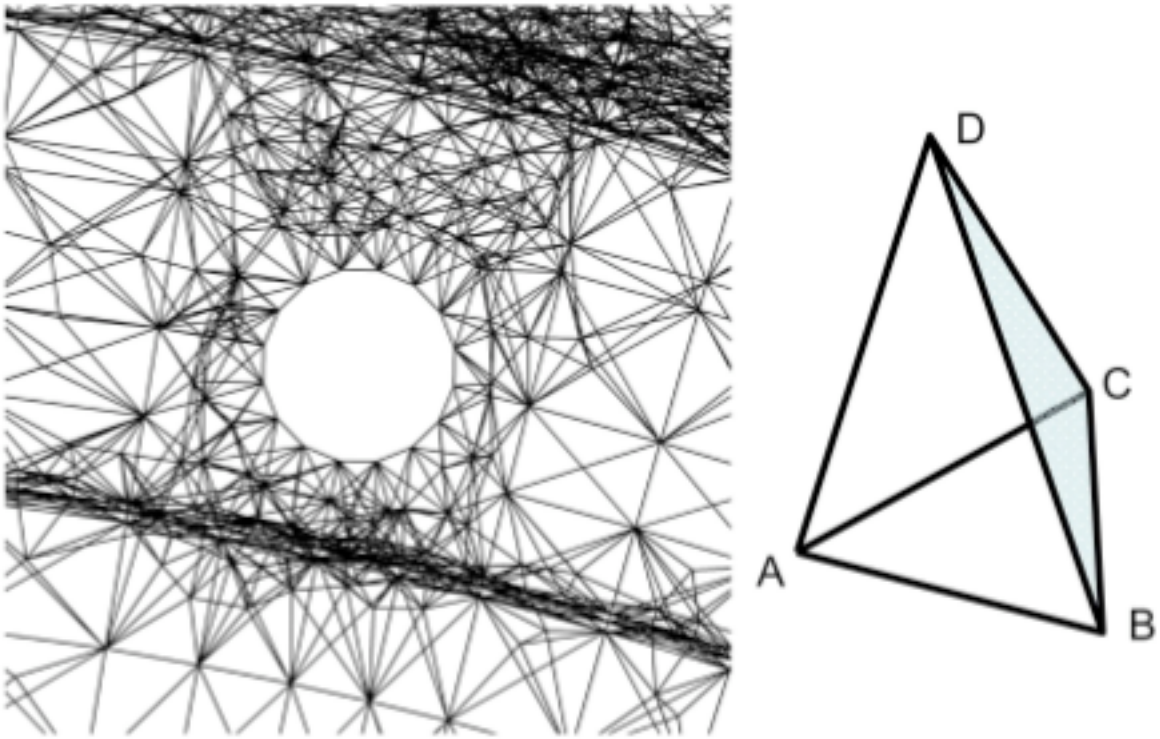


1) Data associated with cells at multiple dimensions

2) complex domains



Tetrahedral Meshes

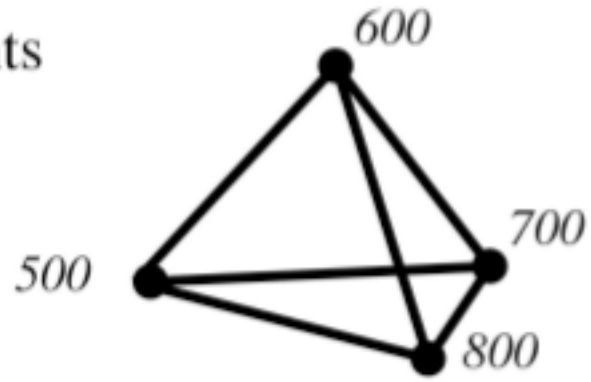


(a) (b)

RDBMS representation

EID	Node 1	Node 2	Node 3	Node 4	Data
100	500	600	700	800	…

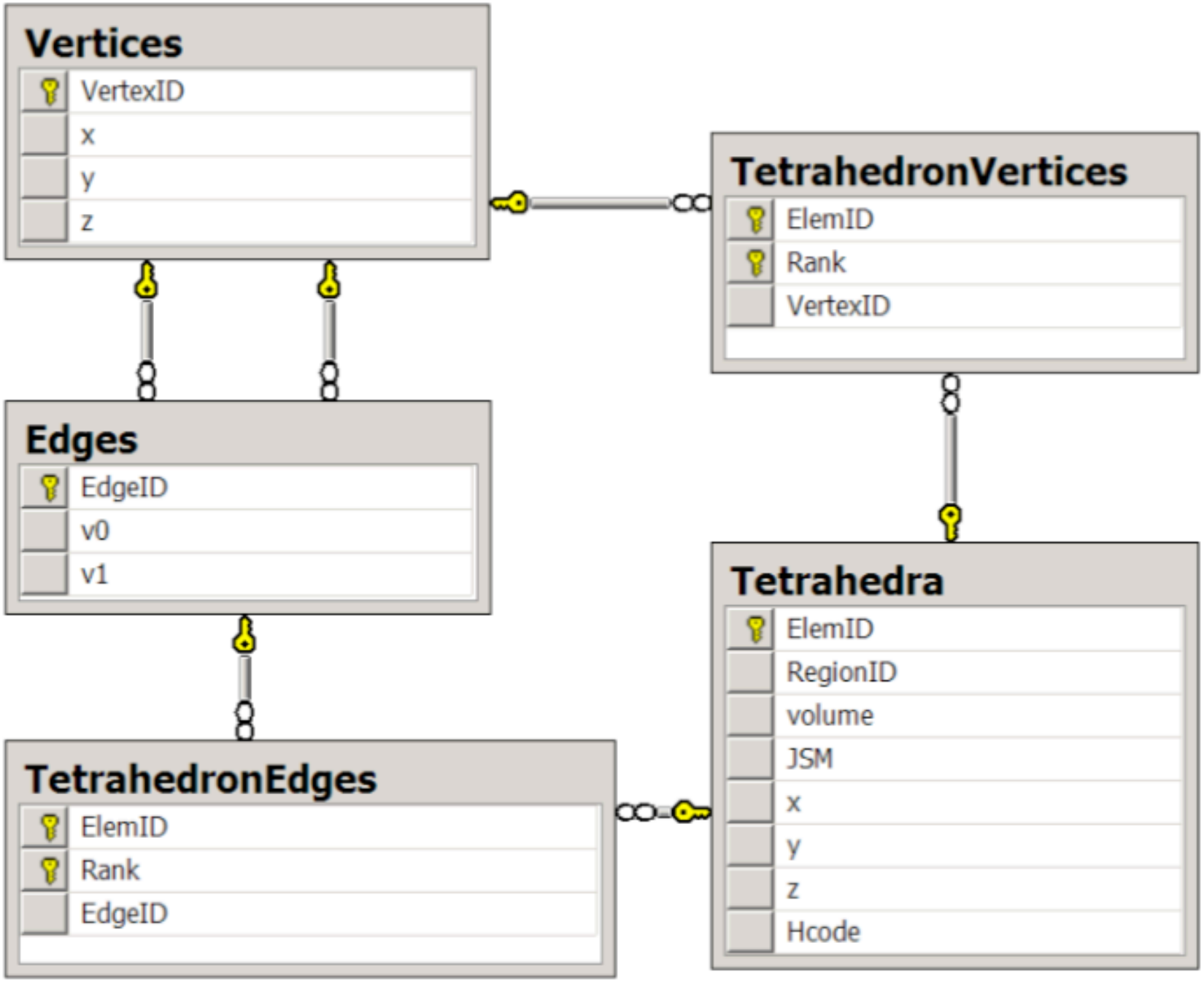
Elements



Element 100

NID	x	y	z	Data
500				
600				
700				
800				

Nodes


Vertices
VertexID
x
y
z
TetrahedronVertices
ElemID
Rank
VertexID
Edges
EdgeID
v0
v1
Tetrahedra
ElemID
RegionID
volume
JSM
x
y
z
Hcode
TetrahedronEdges
ElemID
Rank
EdgeID

Spatial

Cell(cellid :: int, cell :: polygon)

Node(nodeid :: int, point :: point)

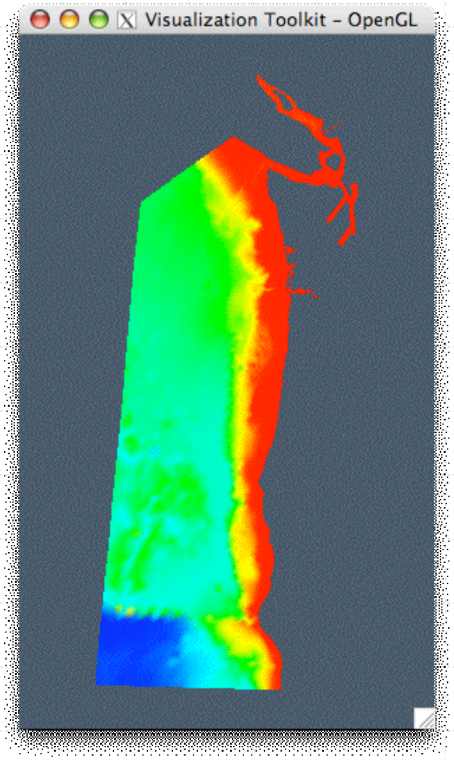
Problem

- Point Query
- Range Query (Bbox)
- Feature queries

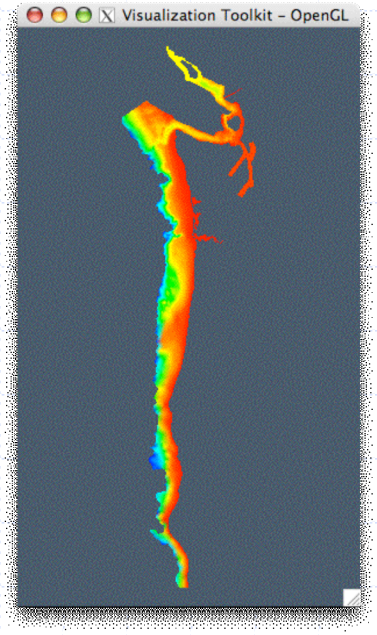
Feature Query example

Find bathymetry > 15

H =



rH =



color: bathymetry

Directed Local Search

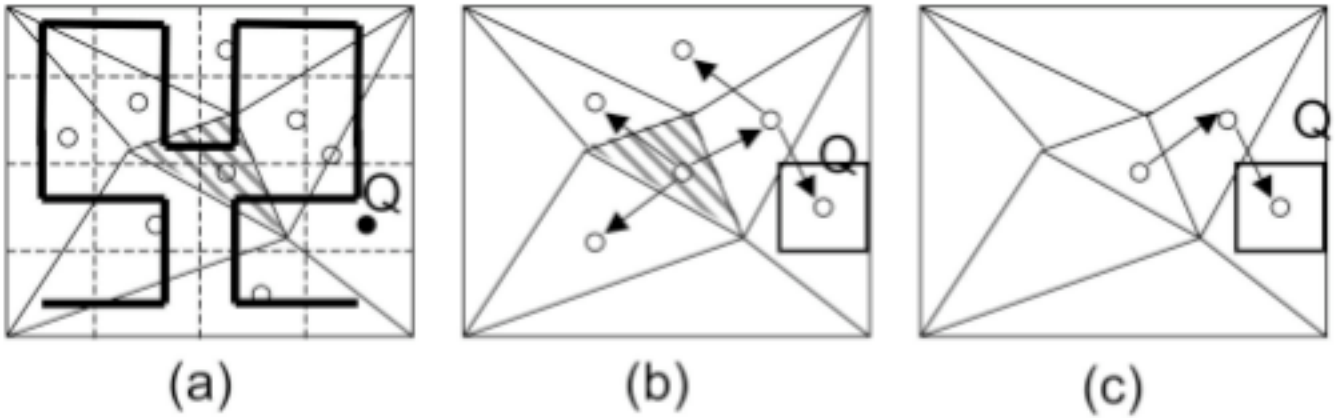


Figure 9: (a) Using the Hilbert order to select a starting element for point Q. (b) Hilbert_BFS example. (c) Hilbert_Direct example.

Space Efficient Encoding

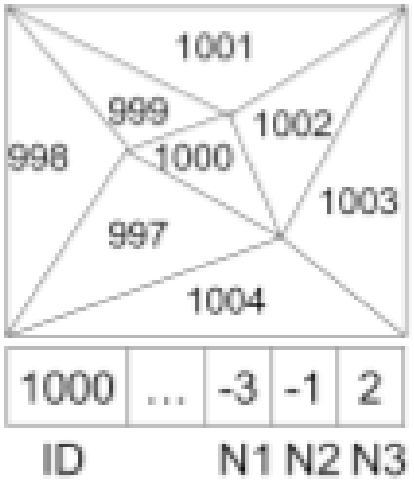


Figure 11: 2D example of adjacency compression. Logical ID differences require fewer bits.

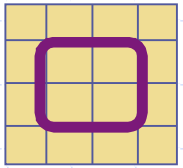
A different problem

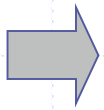
Hypothesis

Programs that create or manipulate gridded datasets exhibit an algebraic structure that can be used to facilitate reasoning about them and improve their performance.

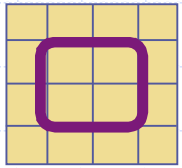
Algorithm-Oriented vs. Model-Oriented

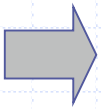
We want:



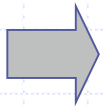


in VTK:



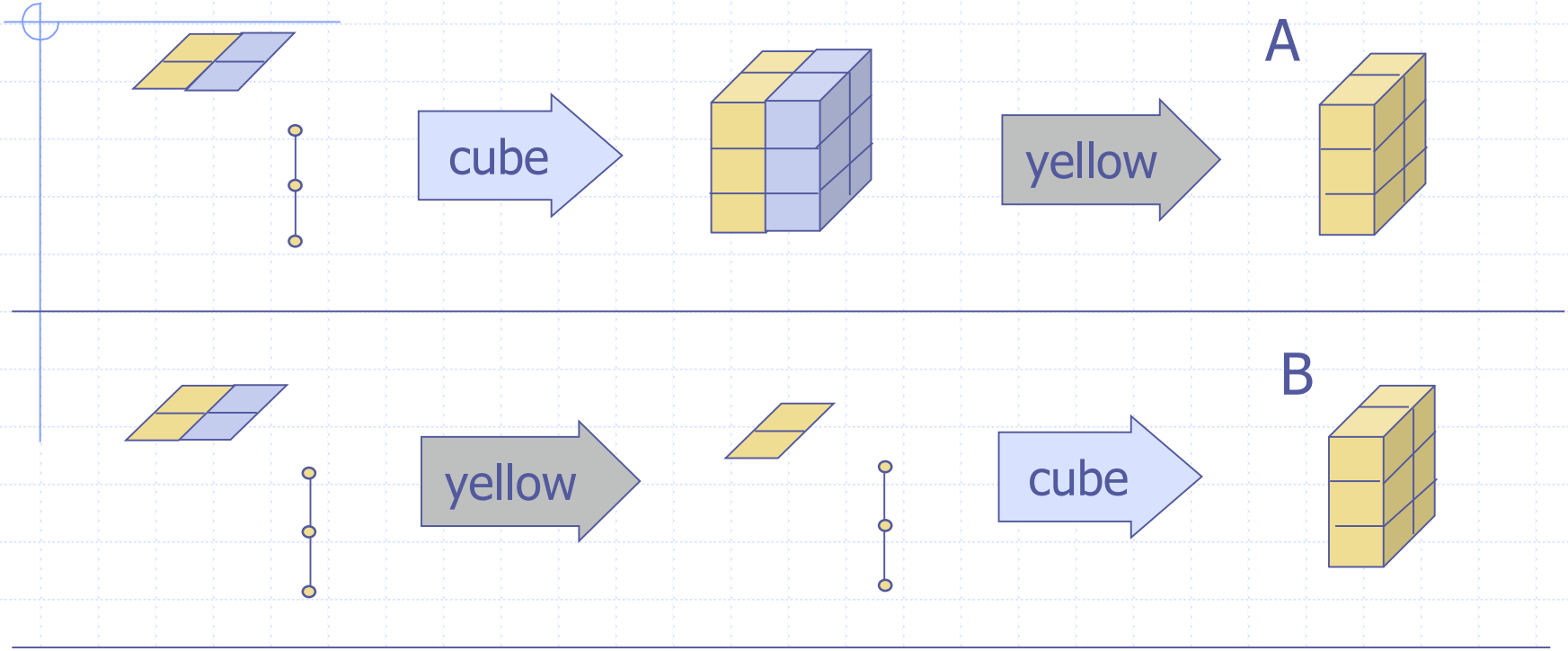


vtkExtractGeometry
vtkThreshold
vtkExtractGrid
vtkExtractVOI
vtkThresholdPoints



- Different C++ classes, each dependent on data characteristics.
- Changes to data characteristics require changes to the program
- Logical equivalences are obscured

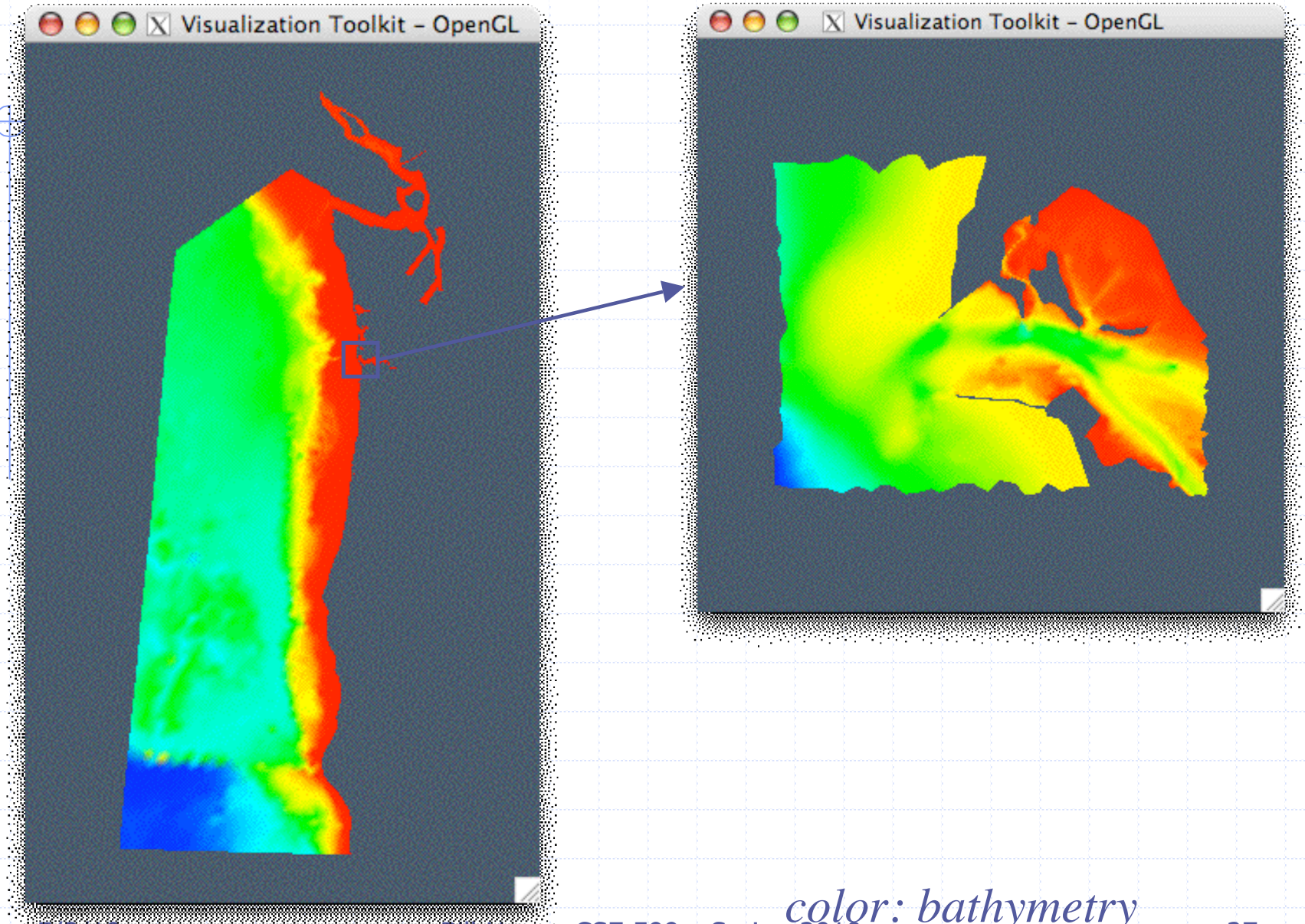
Exposing Equivalences for Algebraic Optimization



- B is equivalent to A, but less costly to compute
- Implies 'Cube' and 'Yellow' should be commutative
- Difficult to determine which vtk operations, if any, have this property *("Documentational Semantics")*

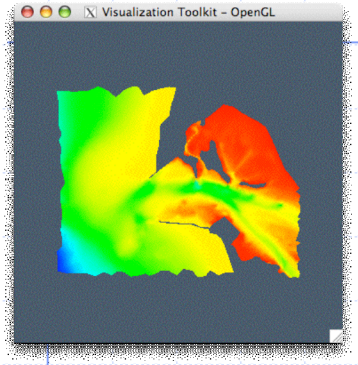
full grid: Eastern Pacific

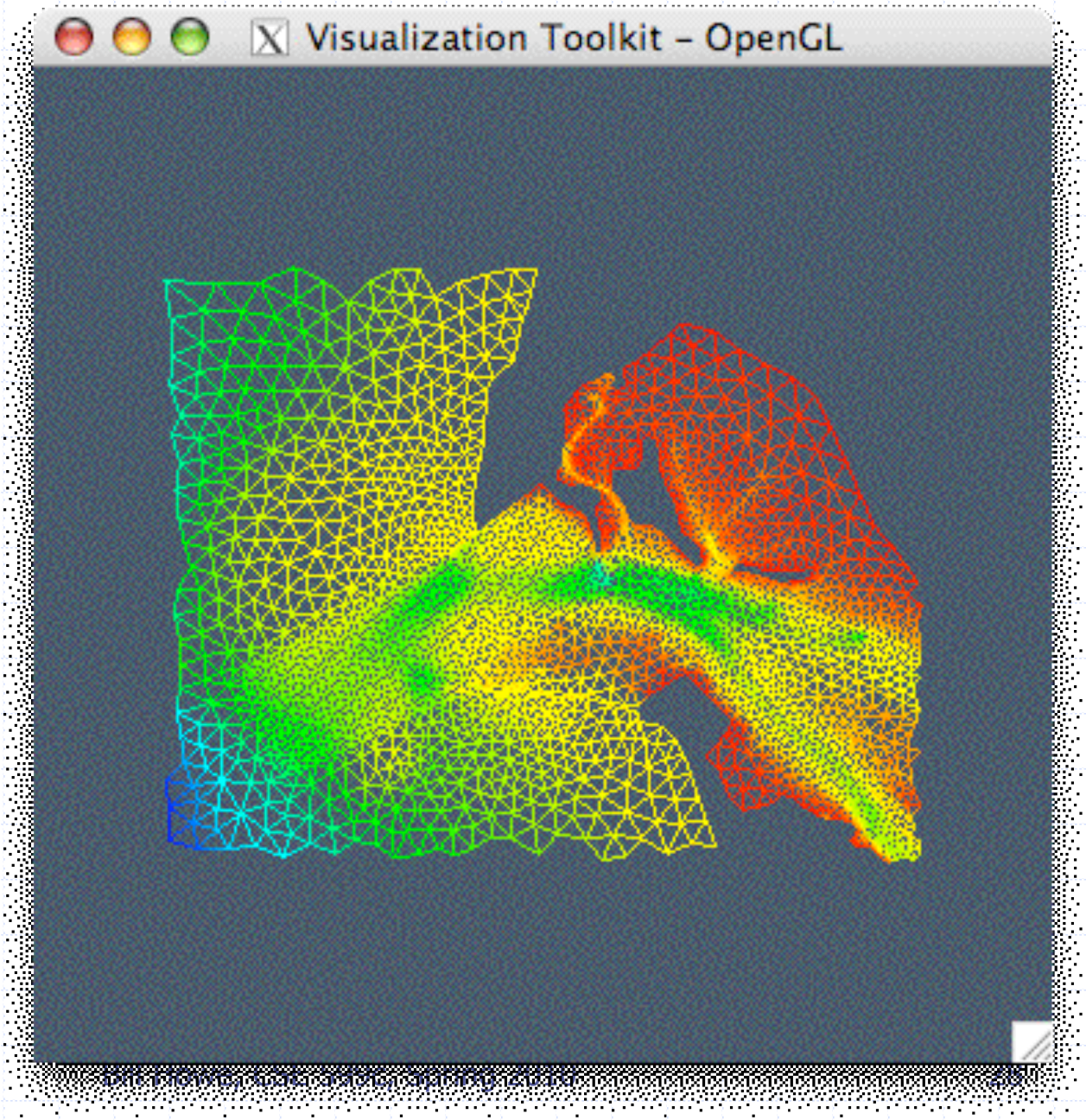
subset: mouth of CR



color: bathymetry

Grid is *well-supported* (no ragged edges)





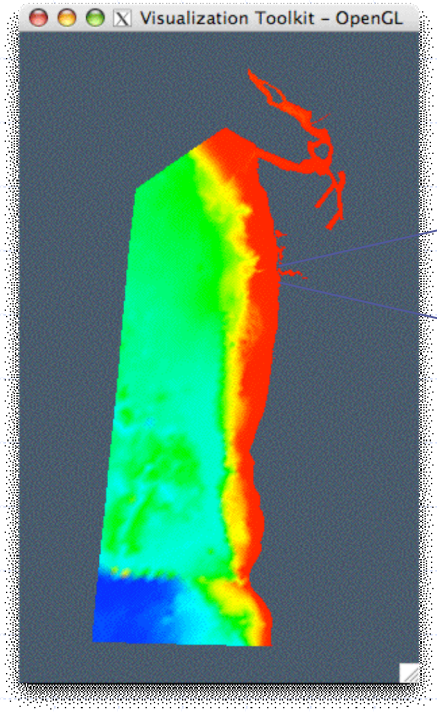
Example (1)

```
H = Scan(context, "H")
rH = Restrict("(326<x) & (x<345) & (287<y) & (y<302)", 0, H)
```

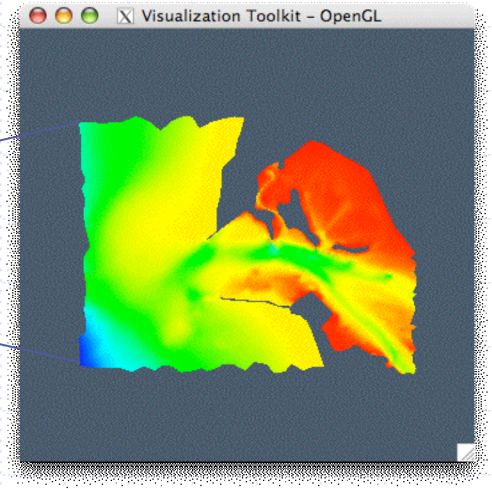
predicate

dimension

H =



rH =



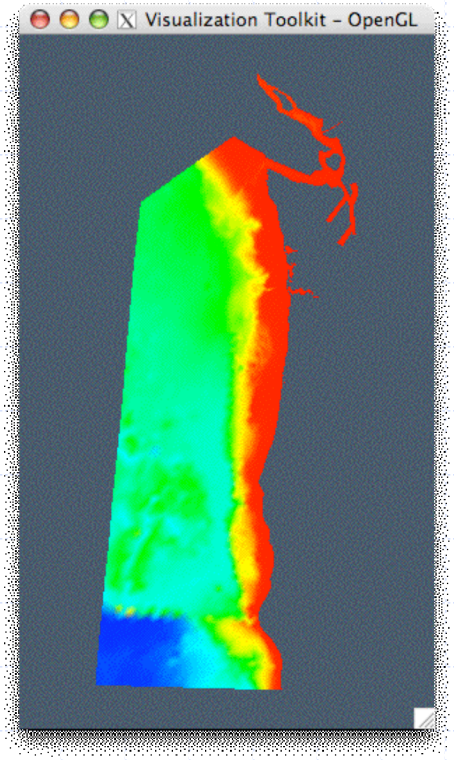
color: bathymetry

Example (2)

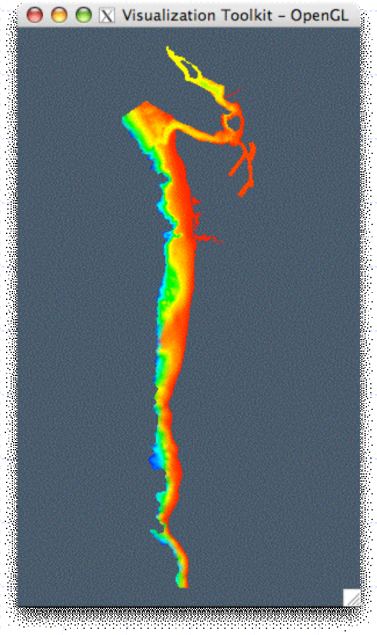
predicate *dimension*

```
rH = Restrict("h<500", 0, H)
```

H =

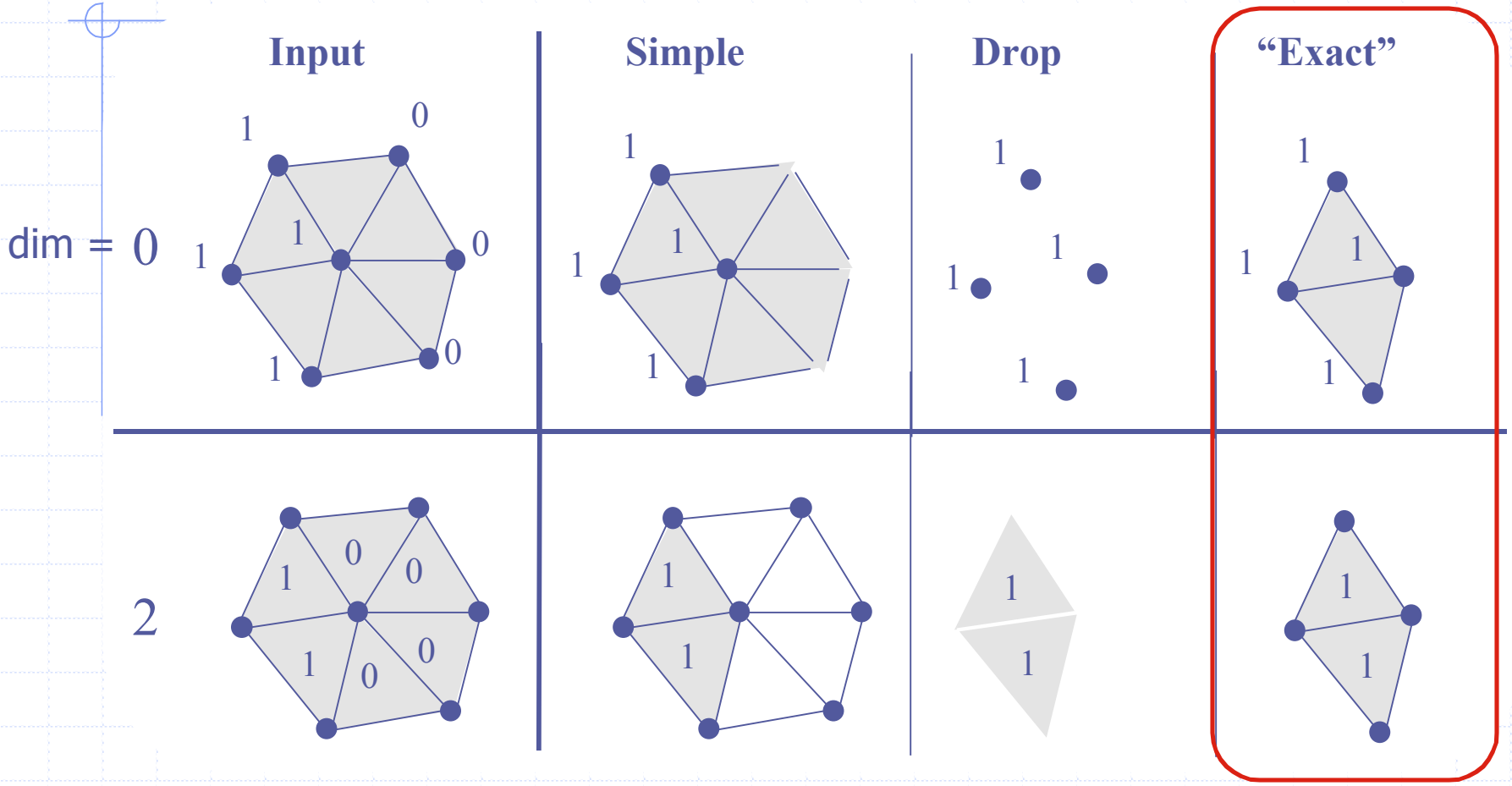


rH =



color: bathymetry

Restrict Semantics

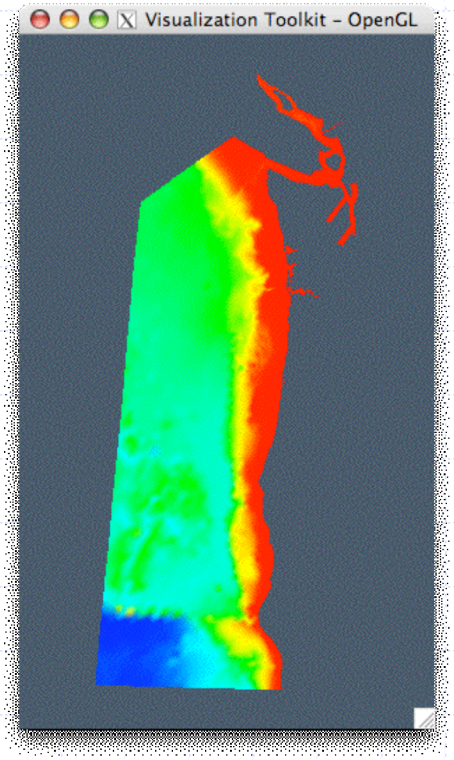


Simple, Drop, and other semantics expressible indirectly

Example (3)

```
H = Scan(context, "H")
B = Bind(H, 0, context, "surf")
```

H =



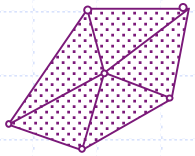
B =

color: bathymetry *color: surface elevation*

Example (4)

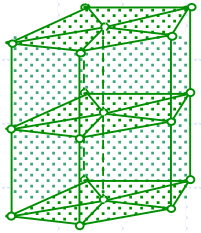
```
H = Scan(context, "H")
V = Scan(context, "V")

HxV = Cross(H,V)
```

H =

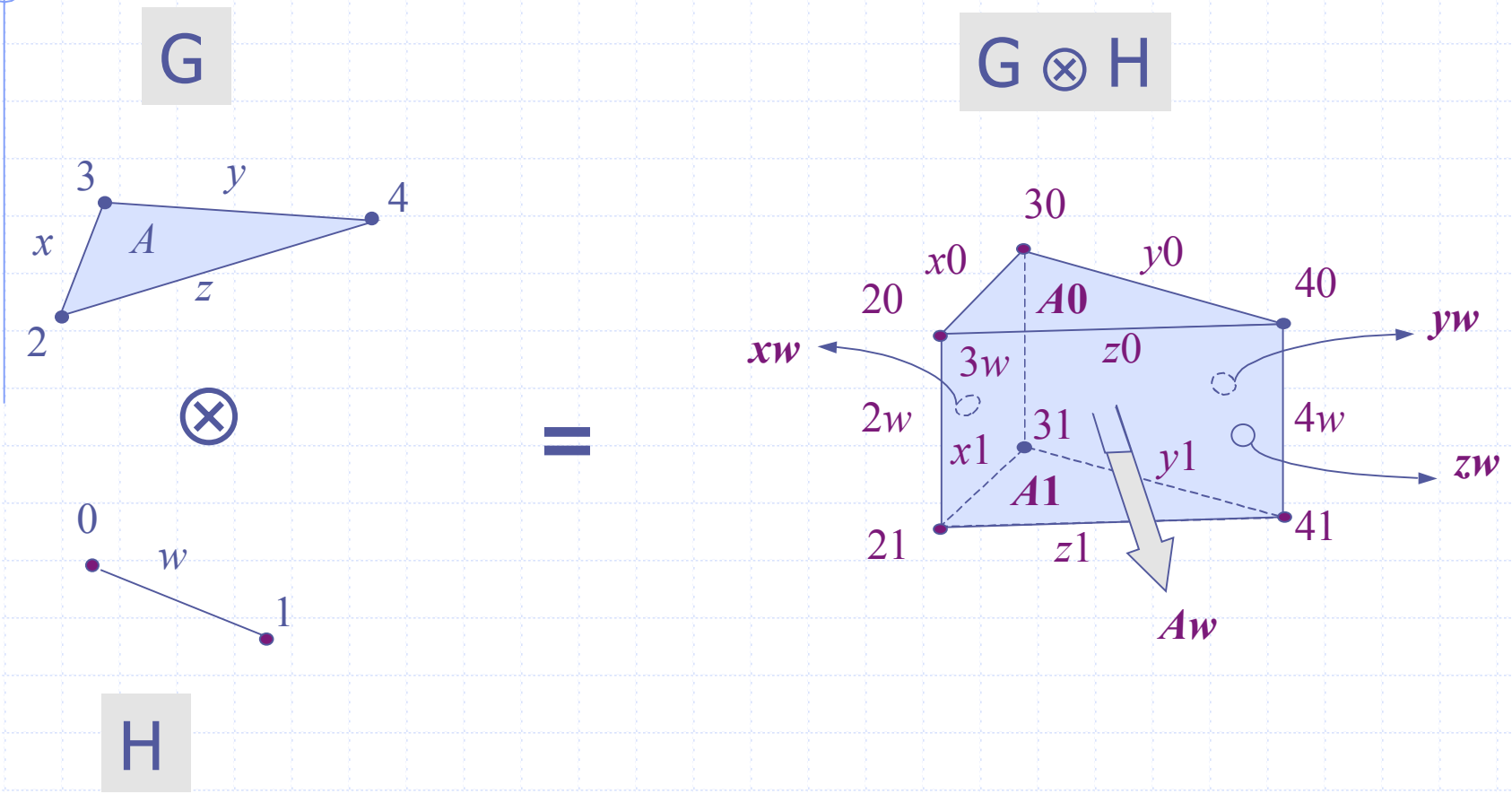


V =

HxV =

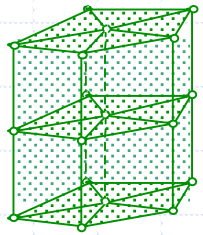


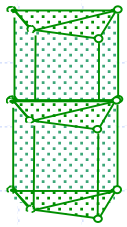
Cross Product



Example (5)

```
H = Scan(context, "H")
V = Scan(context, "V")

HxV = Cross(H,V)

rHxV = Restrict("h<500", 0, HxV)
```

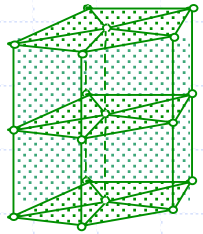
HxV = 

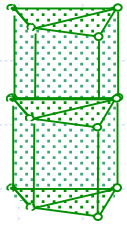
rHxV = 

Example (6)

```
H = Scan(context, "H")
V = Scan(context, "V")

rH = Restrict("h<500", 0, HxV)

HxV = Cross(rH,V)
```

reorder operators

HxV = 

rHxV = 

same result, guaranteed

Cross and Restrict *commute* (under certain conditions)

Related Work

- Non-Discrete Data Models
  - Fiber Bundles/Sheafs Butler and Pendley 89, 91; Treinish 98
  - Cellular Complexes Mattiussi 97
  - Fields Moran 01, Bryson et al. 96
- Discrete Data Models
  - "Mesh" Lee et al 02; Baldwin, Abdulla, Critchlow 03
  - "Grid" Berti 02
- Extended Data Structure Models
  - Arrays Marathe et al. 97, Libkin et al 96
  - Images e.g., Nes, Kersten 98
  - Volumes/Regions Winter et al 02, Wang et al. 03

Related Work

- No known solutions that
  - Support generalized grids
  - Report on an implementation
  - Support algebraic reasoning
- We want these things, plus:
  - Dimension-agnosticism
  - Efficiency

Roadmap

- Introduction
  - Computational Science and Data Products
  - Thesis
  - Strawman Data Structures
  - Related Work
- *Grids and Gridfields*
- Uses of Gridfields
- Experimental Evaluation
- Conclusions and Outlook

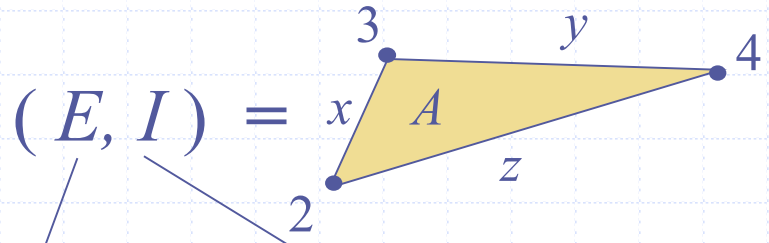
Key Themes

- Separate Grid from Data
- Separate Geometry from Topology

- Geometry is modeled as ordinary data

Grid

- A cell is a featureless object assigned an integer dimension
- A grid is a pair:
  - E : a set of cells, and
  - I : a partial order that respects cell dimension (the *incidence relation*)



$E_0 = \{2,3,4\}$

$E_1 = \{x,y,z\}$

$E_2 = \{A\}$

$I =$

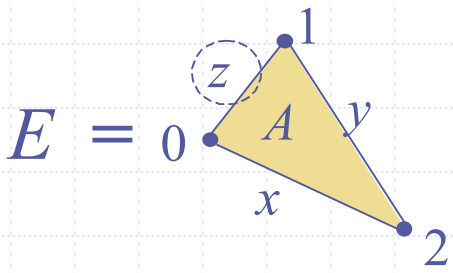
2	z
4	z
z	A
3	y
4	y
y	A
2	x
3	x
x	A

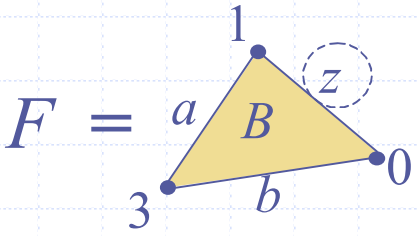
...plus the transitive closure

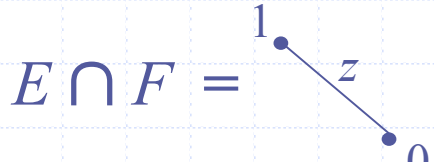
Grid Operations

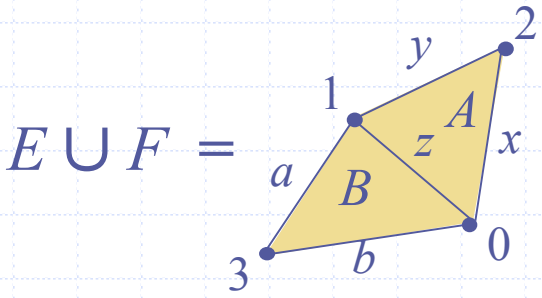
- Lifted Set Operations
  - Union
  - Intersection
  - Difference
  - Cross Product

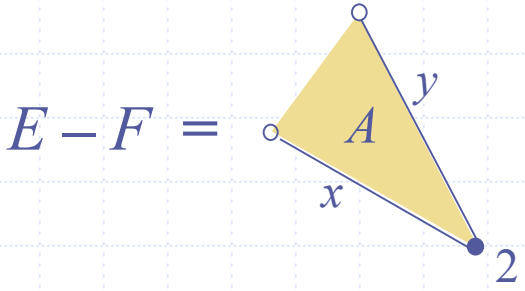
Grid Operations





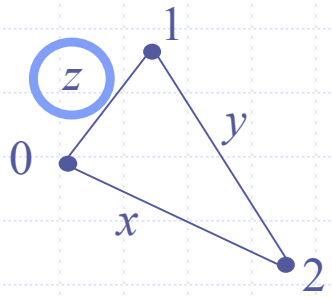




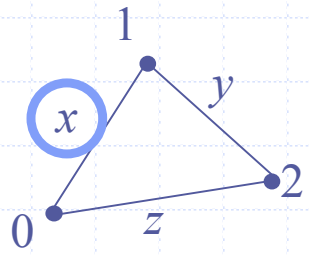


Grid Operations: A Complication

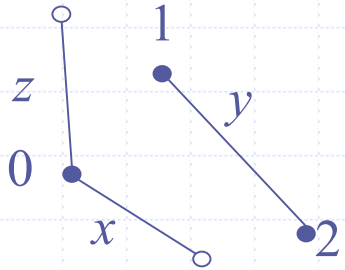
$G =$



$H =$



$G \cap H =$



Subgrids and Compatibility

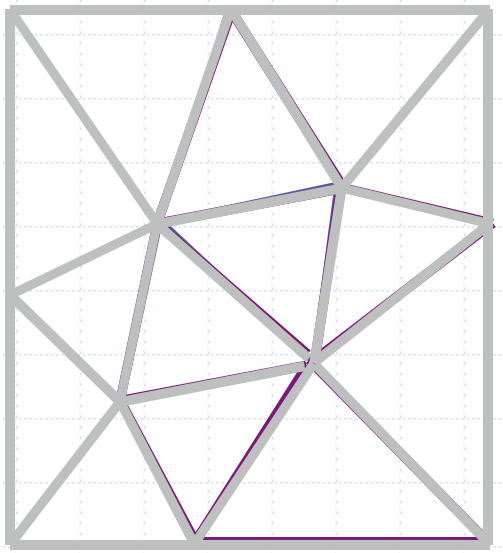
$$E \subseteq F \cup E$$

$$E \cap F \subseteq E$$

$E=$

$G=$

$F=$



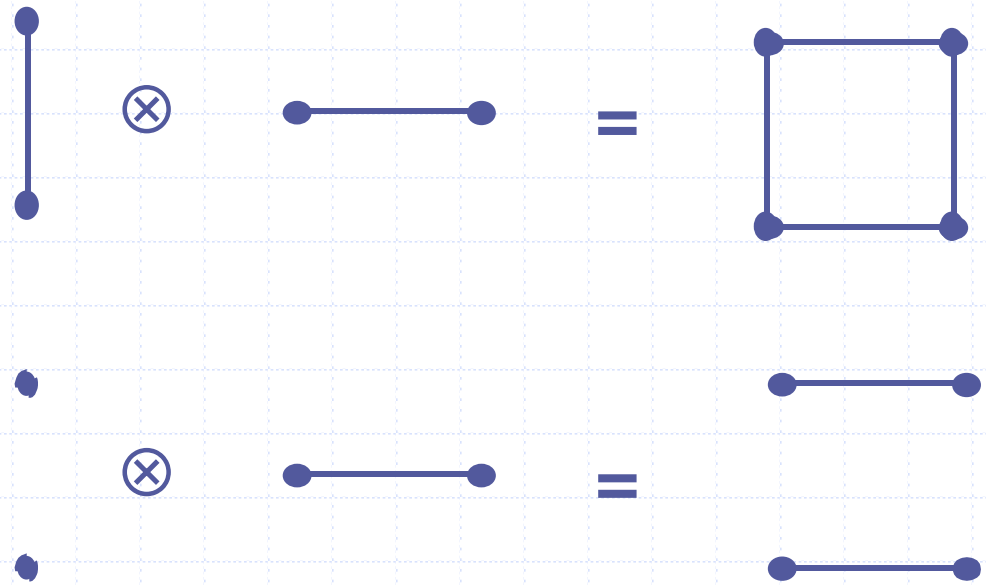
necessary and sufficient that E and F be *compatible*; i.e.,

$$\exists G \quad \textit{s.t.} \quad E \subseteq G \quad F \subseteq G$$

Grid Operations

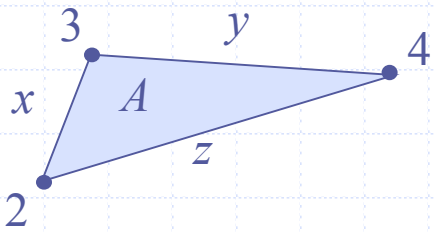
- Lifted Set Operations
  - Union
  - Intersection
  - Difference
  - *Cross Product*

Cross Product

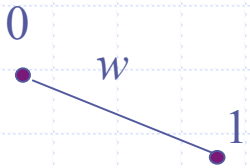


Cross Product (2)

G



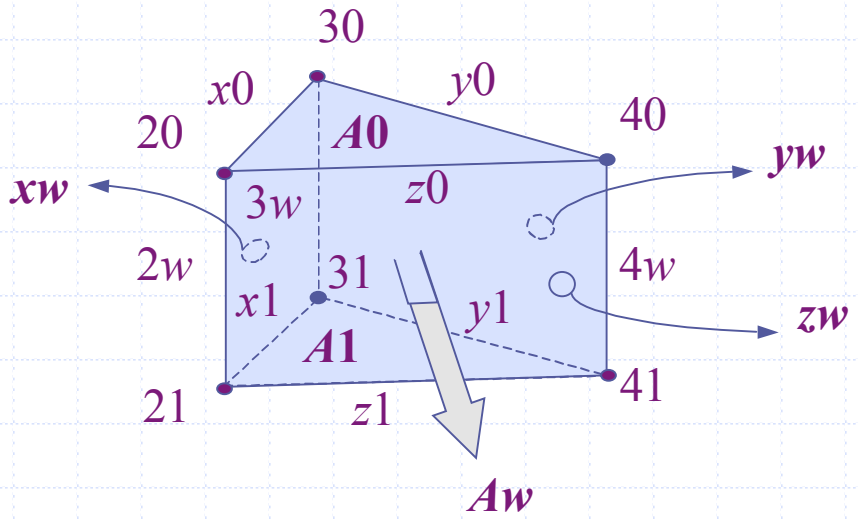
⊗



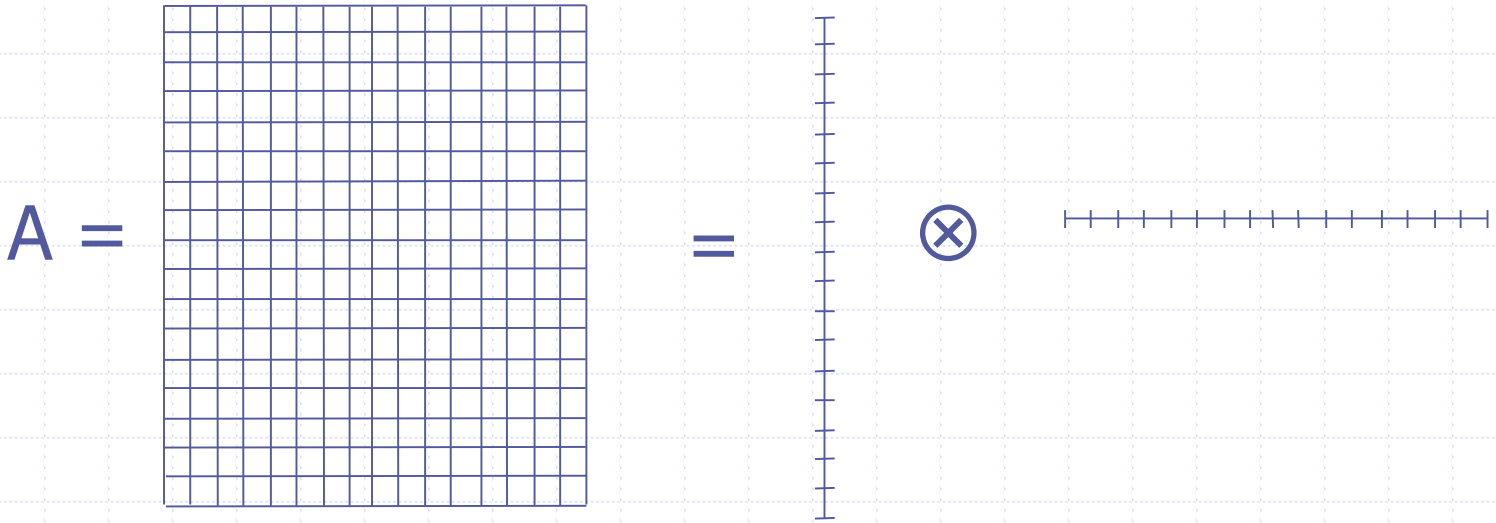
H

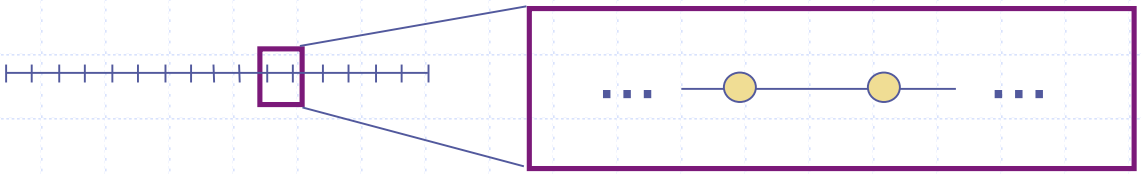
=

G ⊗ H



Constructing GridFields (1)

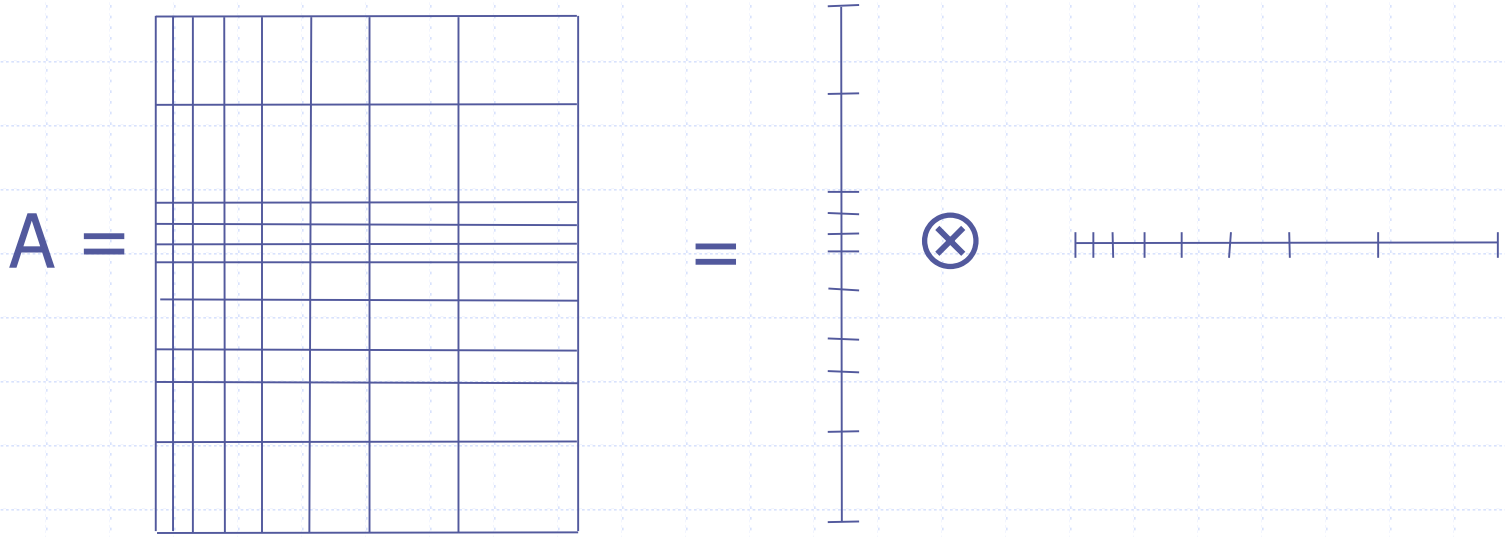


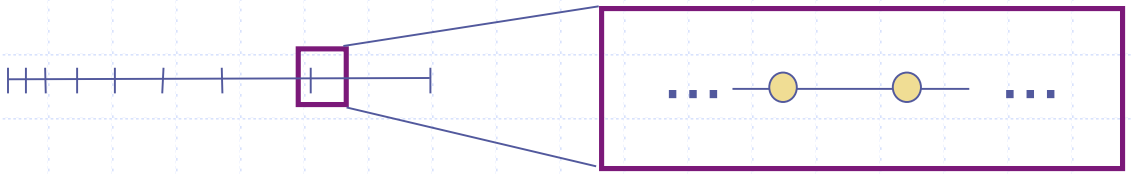


$x_i = \Delta x * i$

x_i is proportional to position i

Constructing GridFields (2)

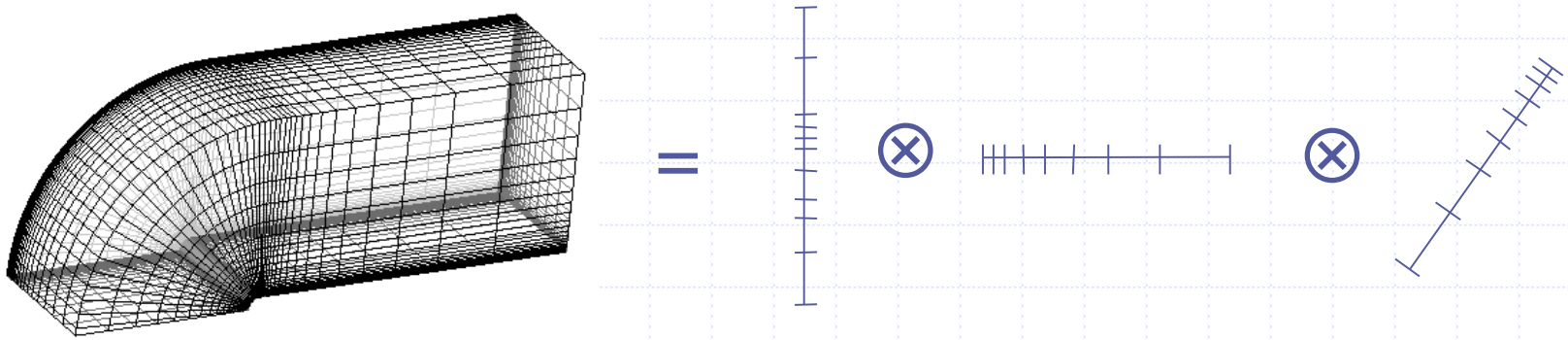




x_i is explicit

x_i is an arbitrary function of position i

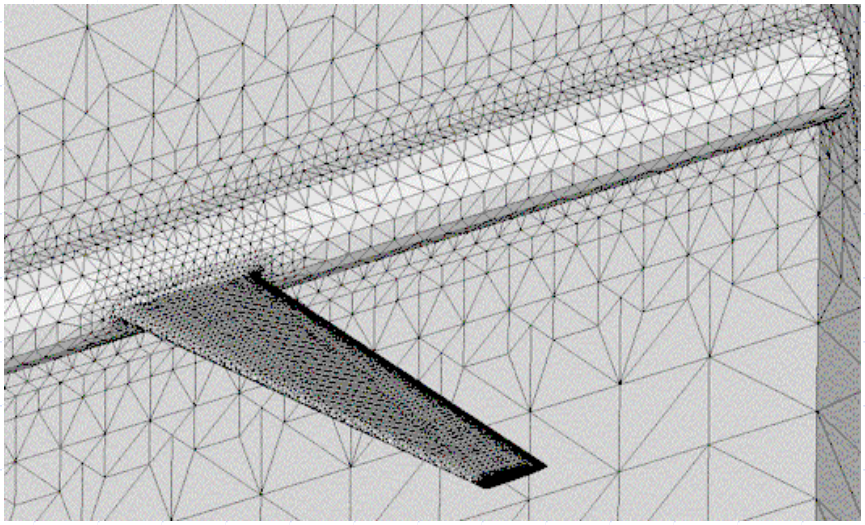
Constructing GridFields (3)



x_i is a function of positions i, j, k

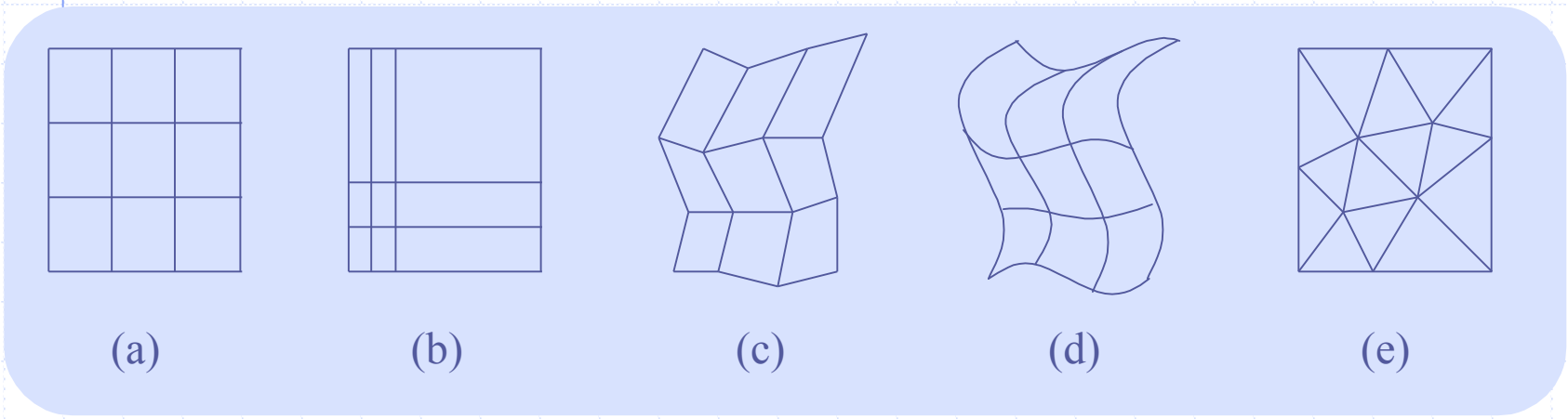
...as are y_i, z_i

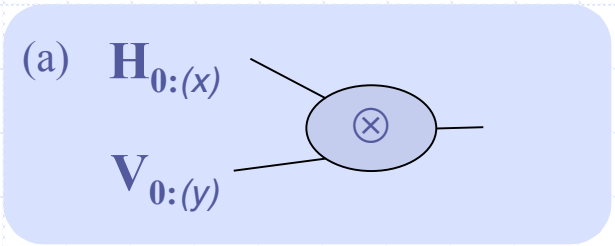
Constructing GridFields (4)

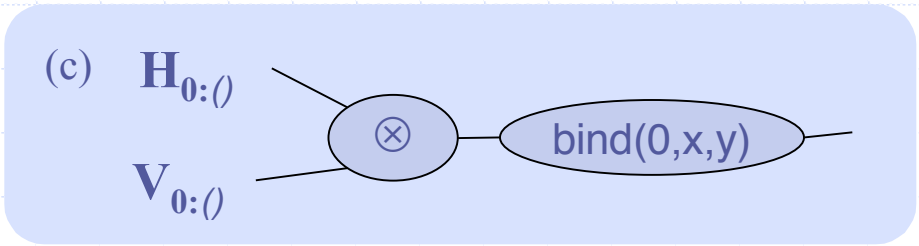


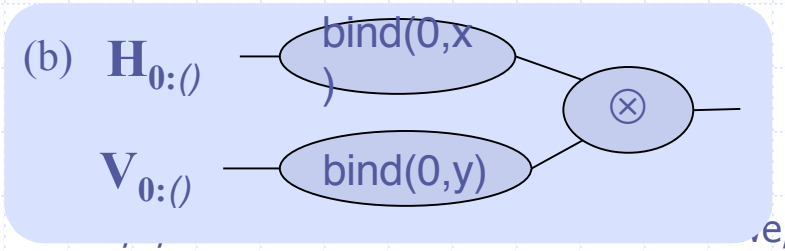
No product structure

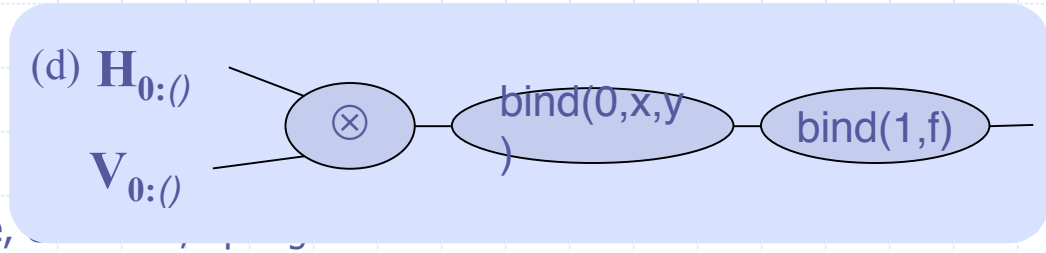
Constructing GridFields











Grid Model Highlights

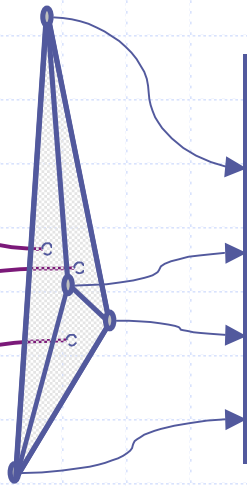
- Arbitrary dimension
- Characterization of "sensible" Grids
- No dependence on geometry
- Conceptual economy

Roadmap

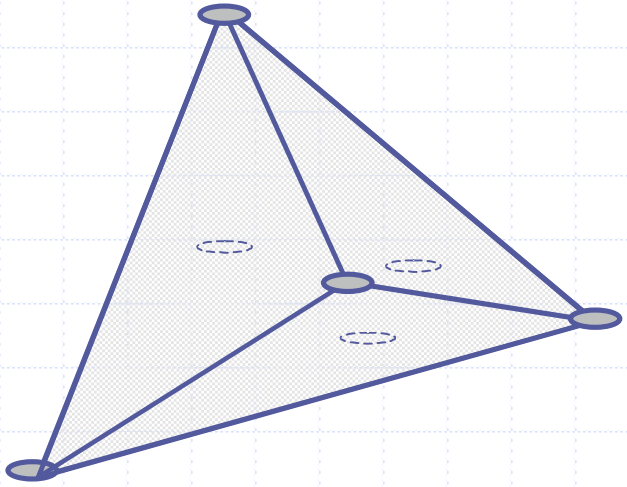
- Introduction
- *Grids and GridFields*
  - Grids
  - Grid Operations and Relations
  - Gridfields and Gridfield Operations
- Operators and Optimization
- Experimental Evaluation
- Conclusions and Outlook

GridFields

flux	area
11.5	3.3
13.9	5.5
13.1	4.5

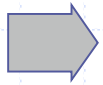


x	y	salt	temp
13.8	10.6	29.4	12.1
13.9	9.4	29.8	12.5
14.3	9.0	28.0	12.0
13.4	9.0	30.1	13.2



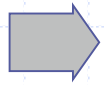
- GridField: A grid with attributes bound to cells of one or more dimensions.

GridField Operations

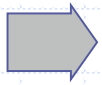


- Lifted grid operations
  - **Union, Intersection, Cross Product**
- **Scan/Bind**

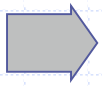
  - Read a grid/attribute from disk
- **Restrict**



  - Remove cells that do not satisfy a predicate
- **Accrete**

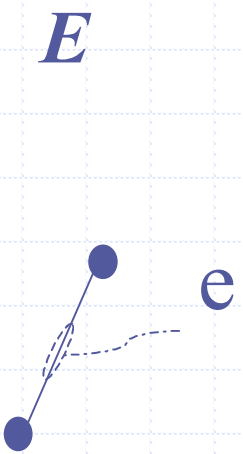


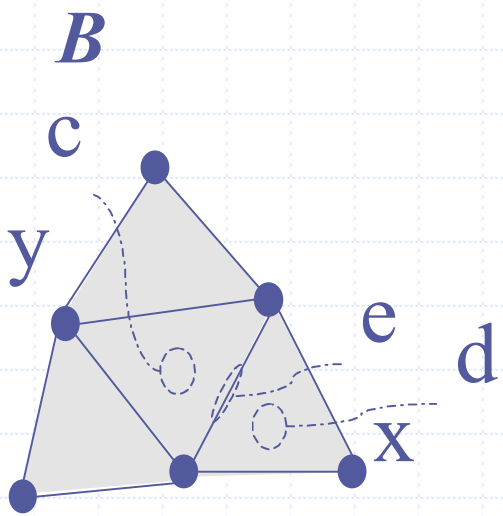
  - Grow a grid by adding neighbors of cells
- **Regrid**

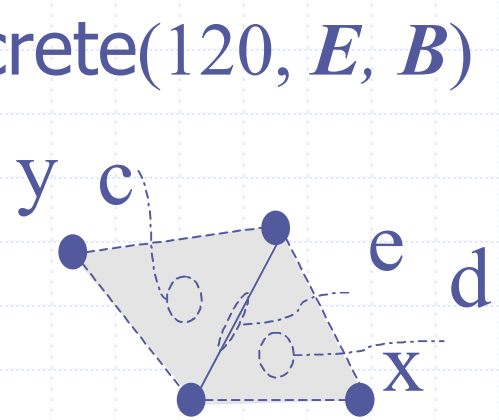


  - Map the data of one grid onto another

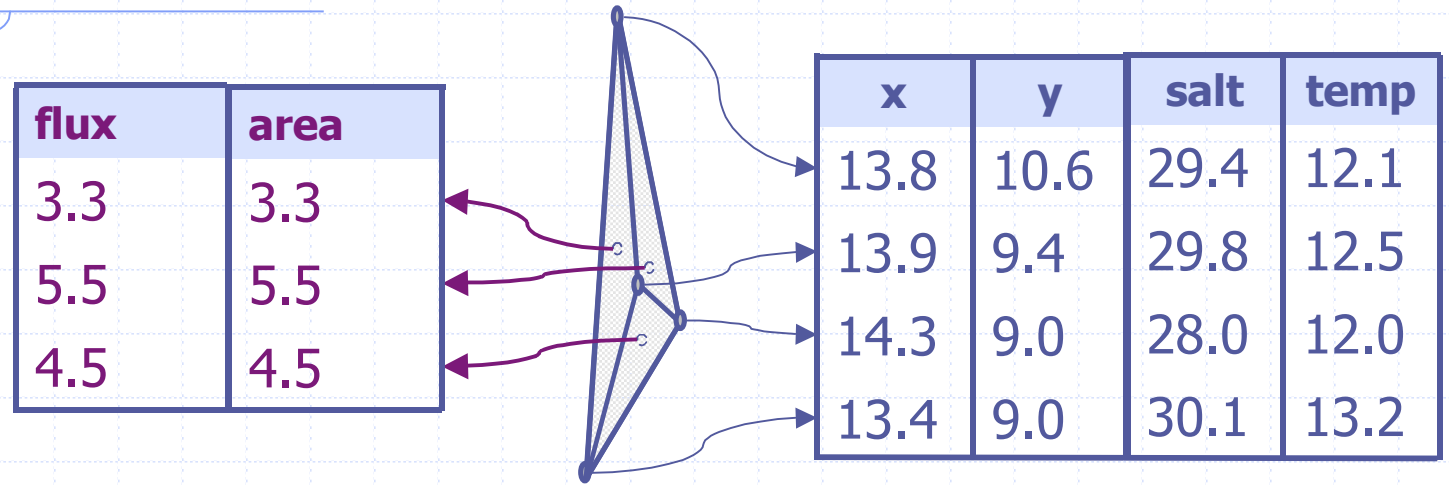
Accrete







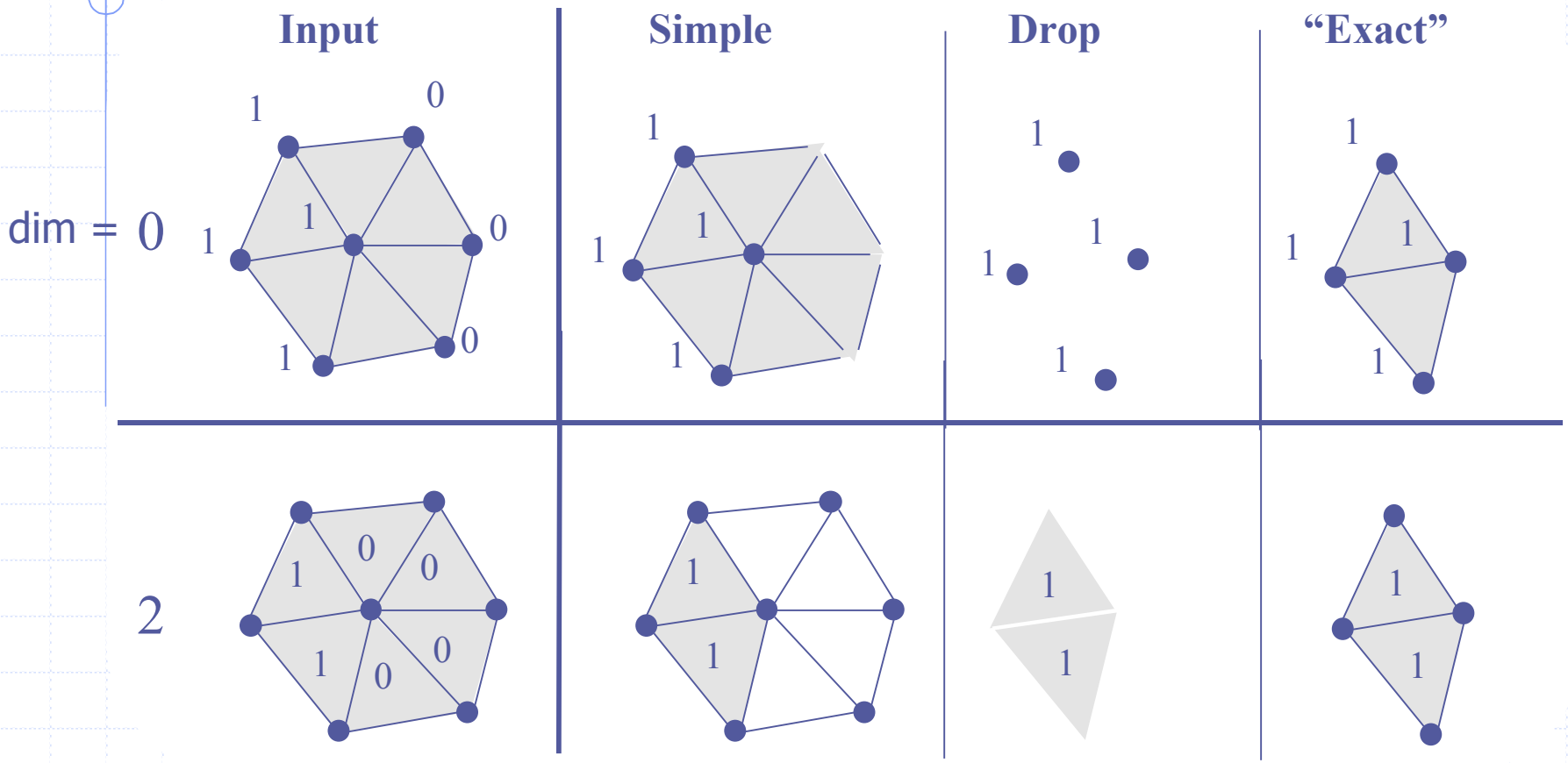
Scan/Bind



flux	area
3.3	3.3
5.5	5.5
4.5	4.5

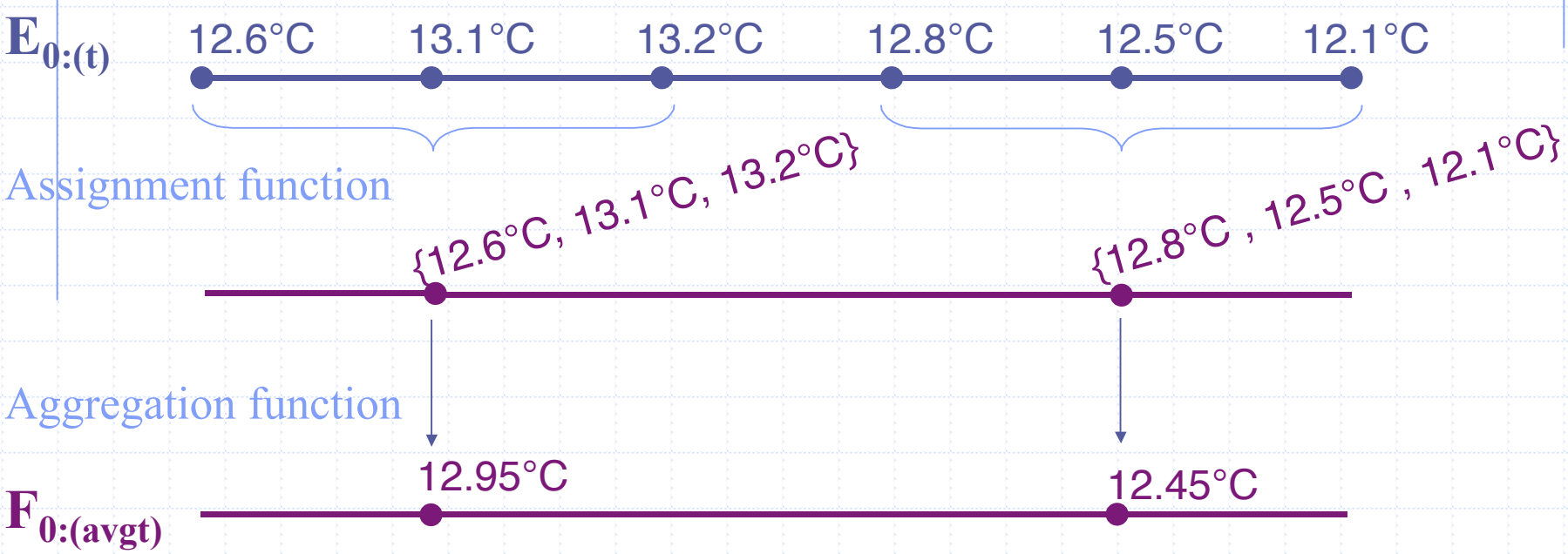
x	y	salt	temp
13.8	10.6	29.4	12.1
13.9	9.4	29.8	12.5
14.3	9.0	28.0	12.0
13.4	9.0	30.1	13.2

Restrict: Which cells are kept?



Simple, Drop, and other semantics expressible with the Accrete operator

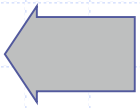
Regrid by Example



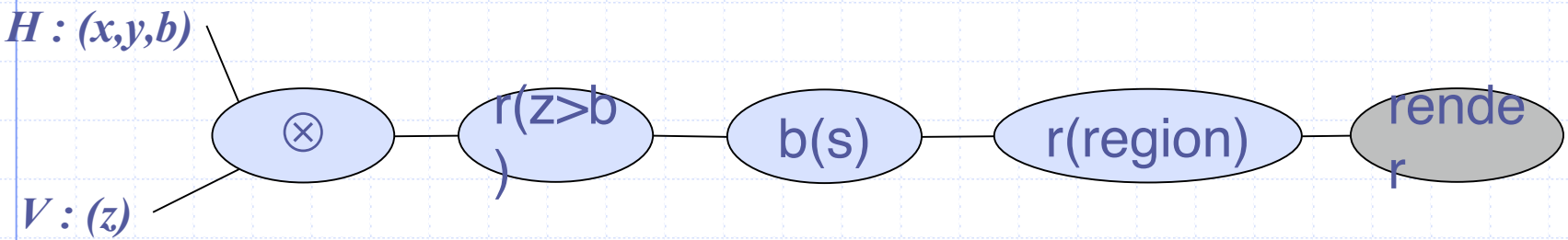
Regrid(S, T, assign, aggregate)

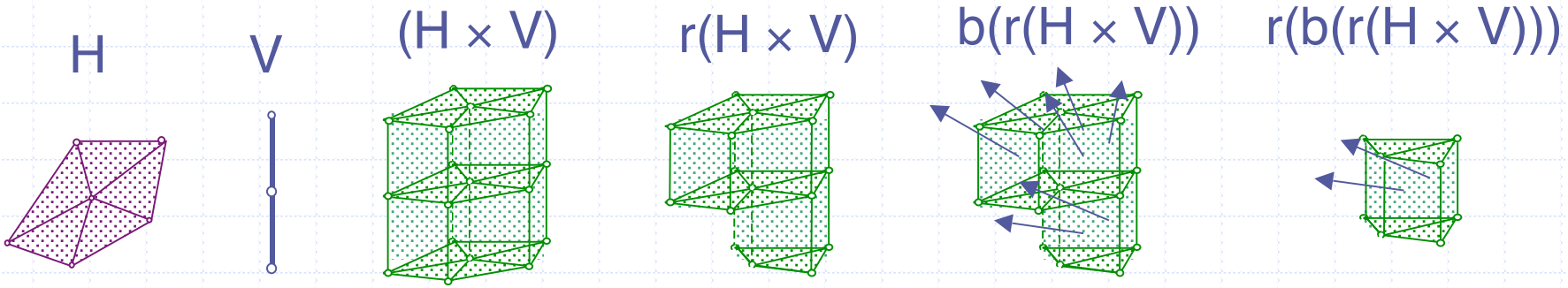
Regrid Specializations

- Apply
  - assign each cell to itself
  - used for basic arithmetic; very common
- Project
  - assign each cell to itself; used to evict attributes
- Neighborhood
  - assign cells from a local neighborhood
  - used for vector calculus
- Unify
  - Assign all k-cells to the cell of a singleton grid



Example Recipe



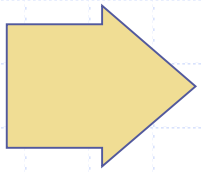


Roadmap

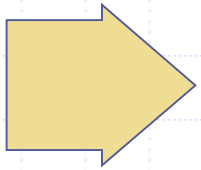
- Introduction
- Grids and GridFields
  - Grids
  - Grid Operations and Relations
  - GridFields
- *Uses of GridFields*
- Experimental Evaluation
- Conclusions and Outlook

Uses of GridFields

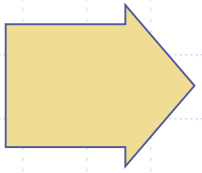
- Succinct Description of Existing Data and Products

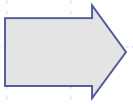


- Optimize existing products

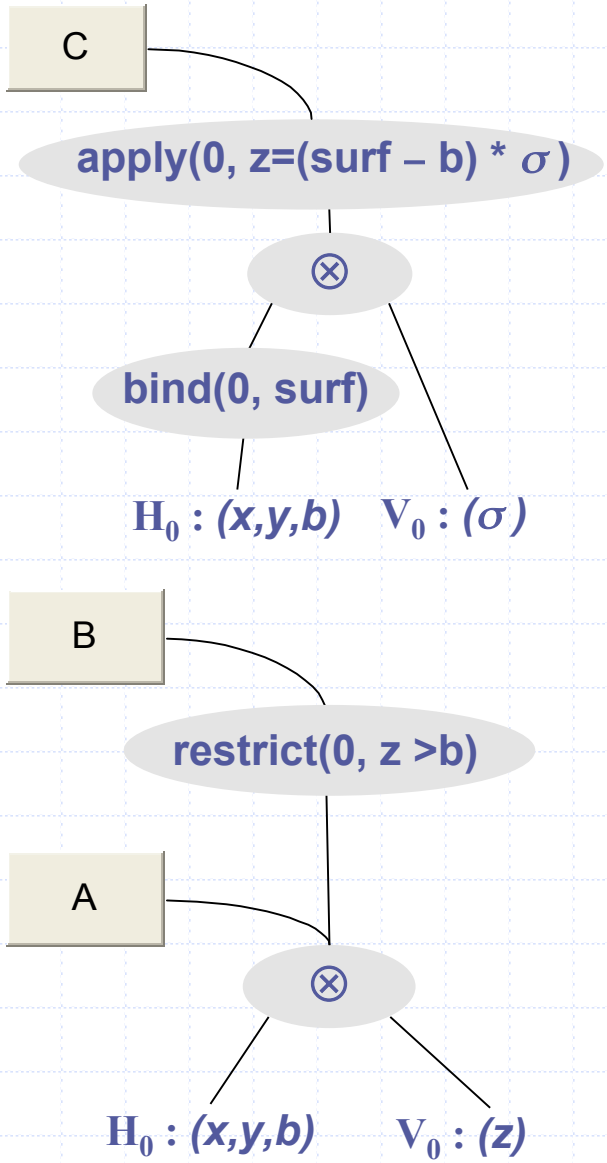


- Define new products

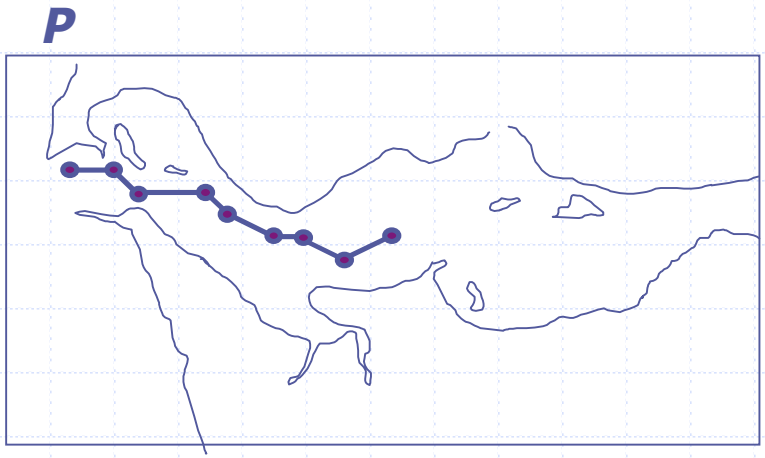


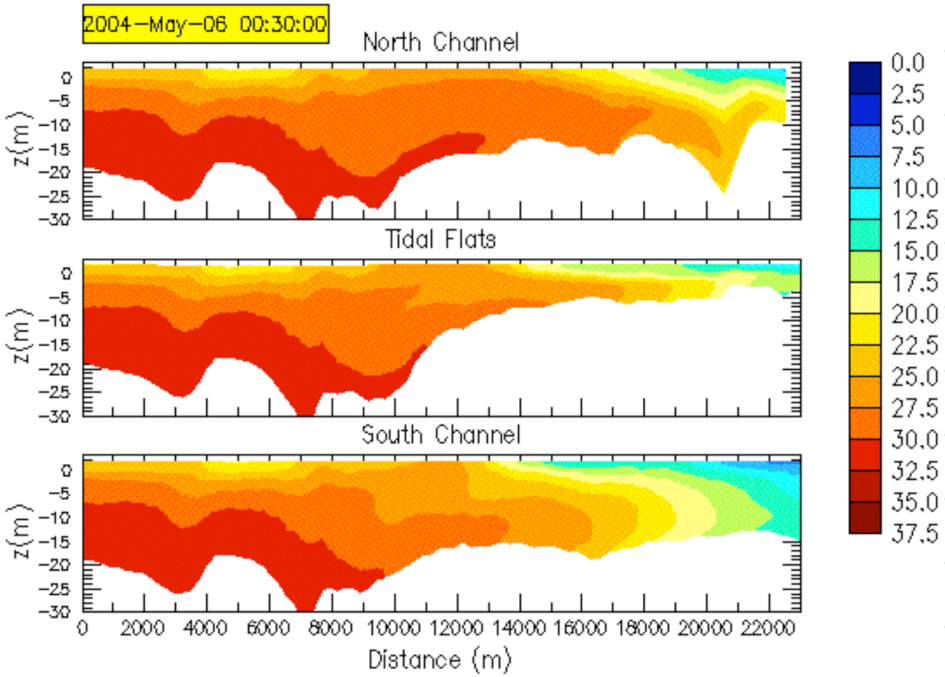


Inactive VTK control

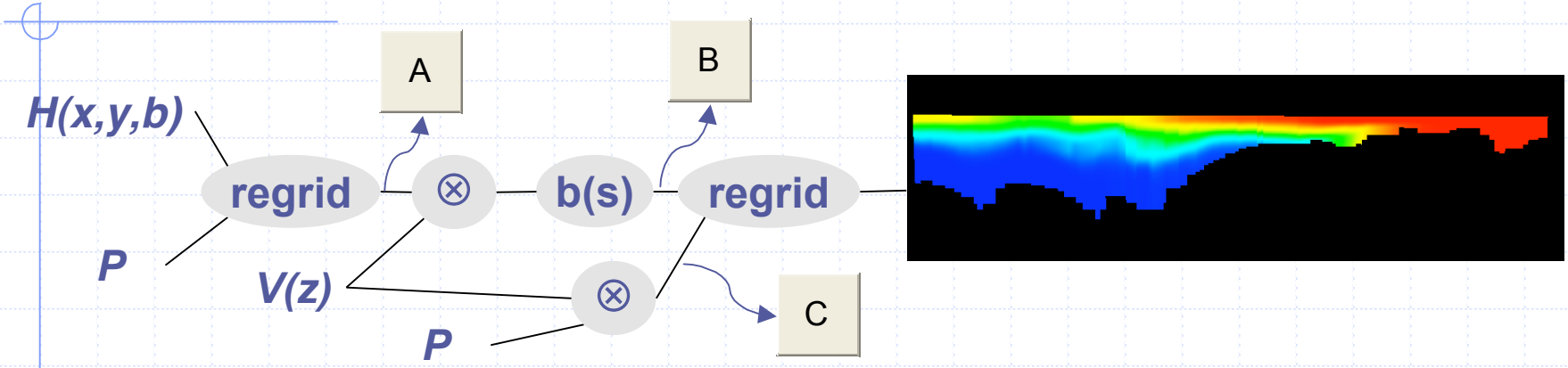


Transect (Vertical Slice)



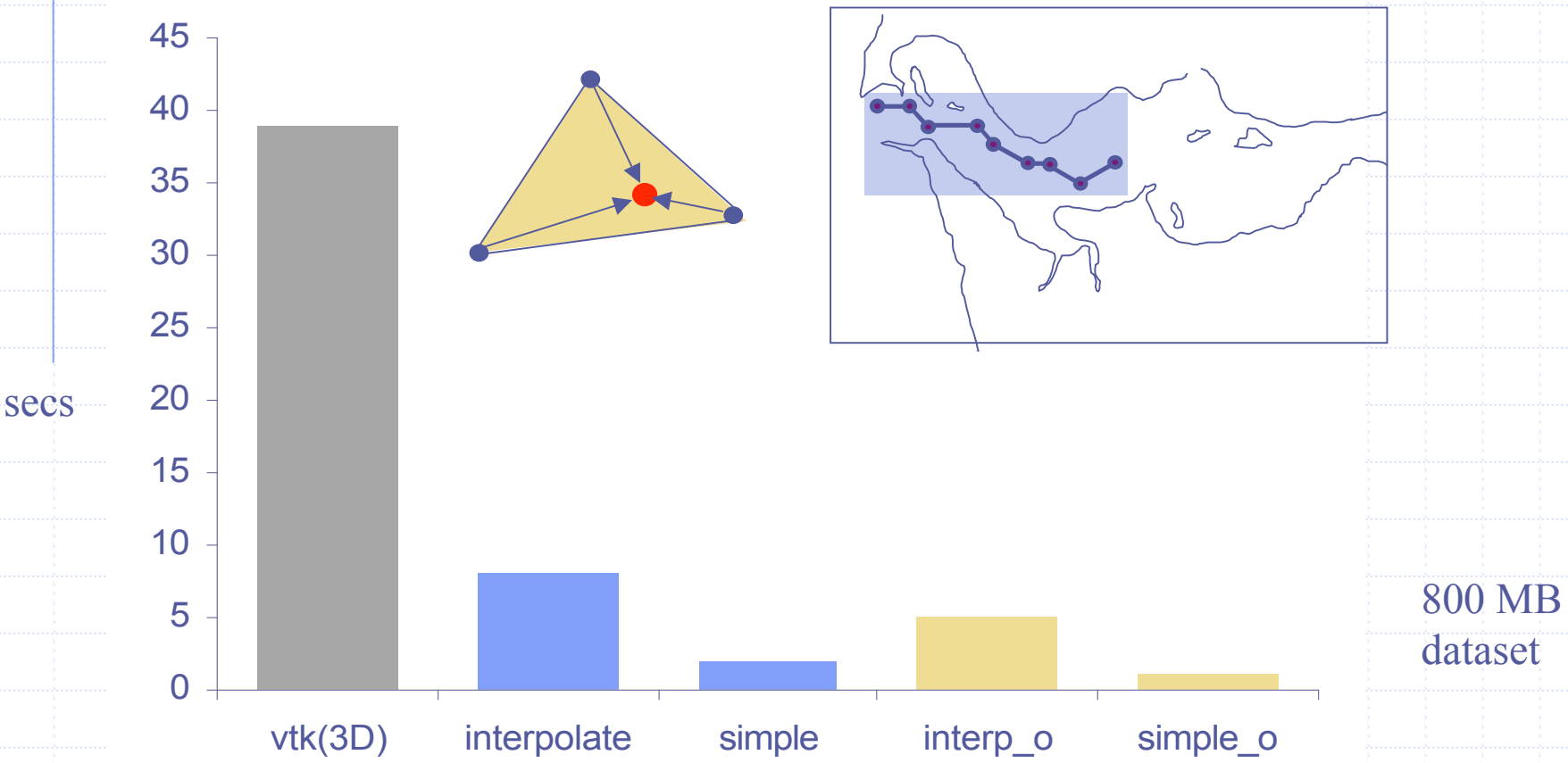


Transect (Vertical Slice)

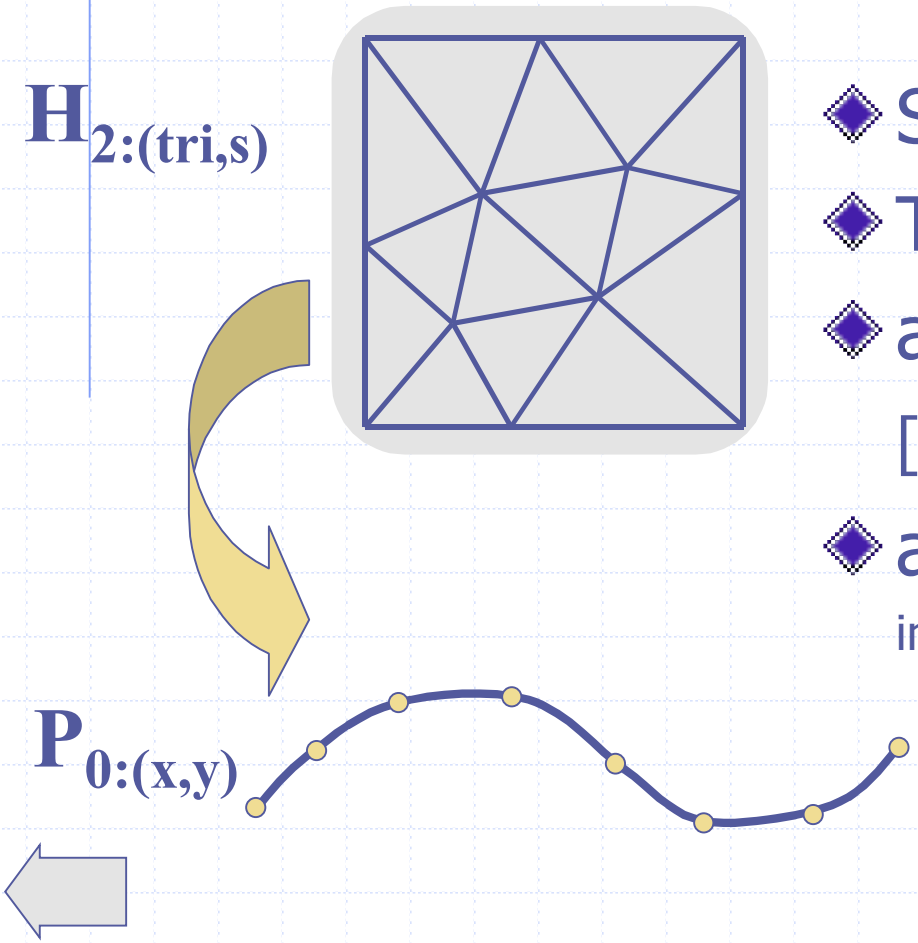


Inactive VTK control

Transect: Results



Regrid as a Spatial Join

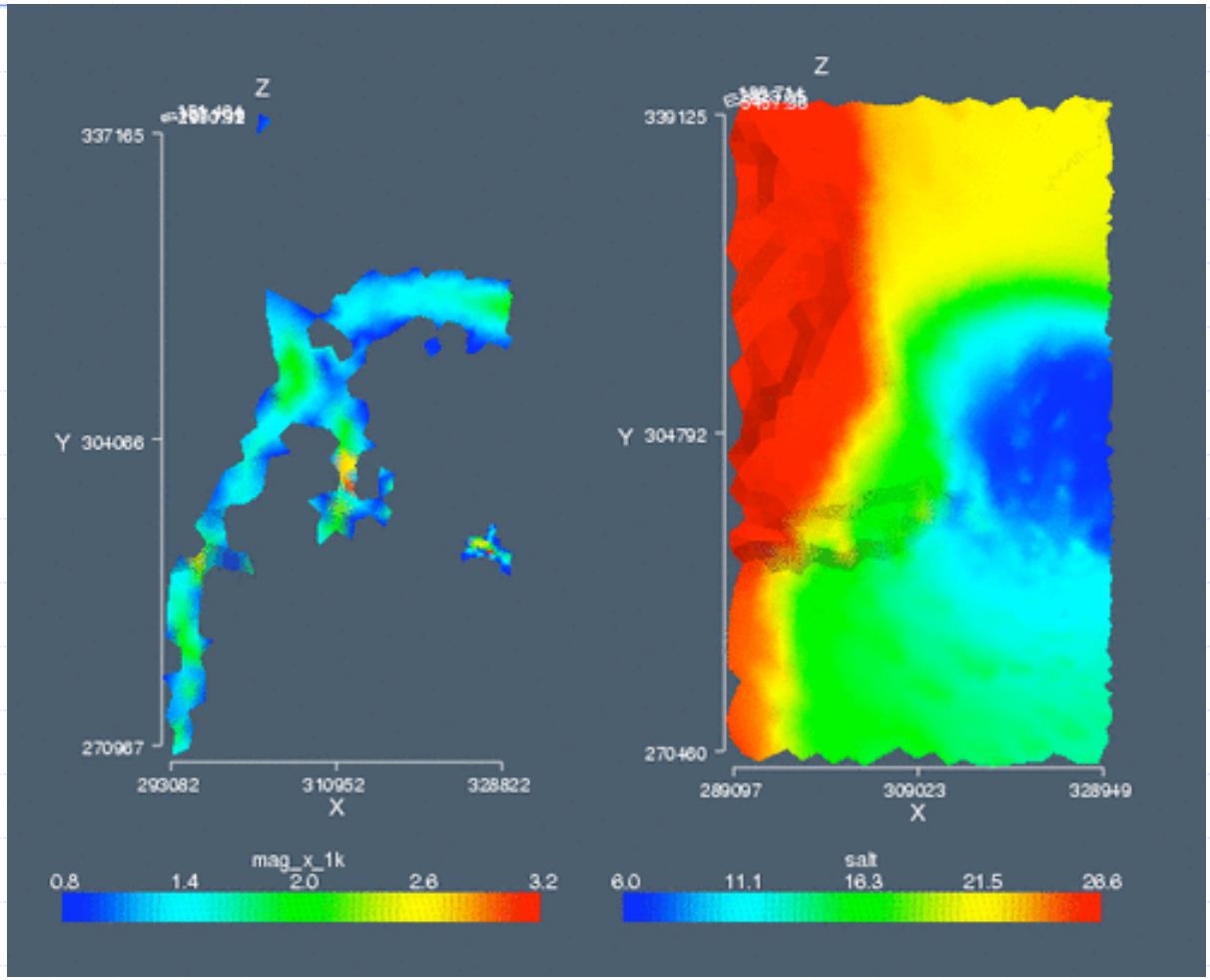


- Source GridField: **H**
- Target GridField: **P**
- assignment function:

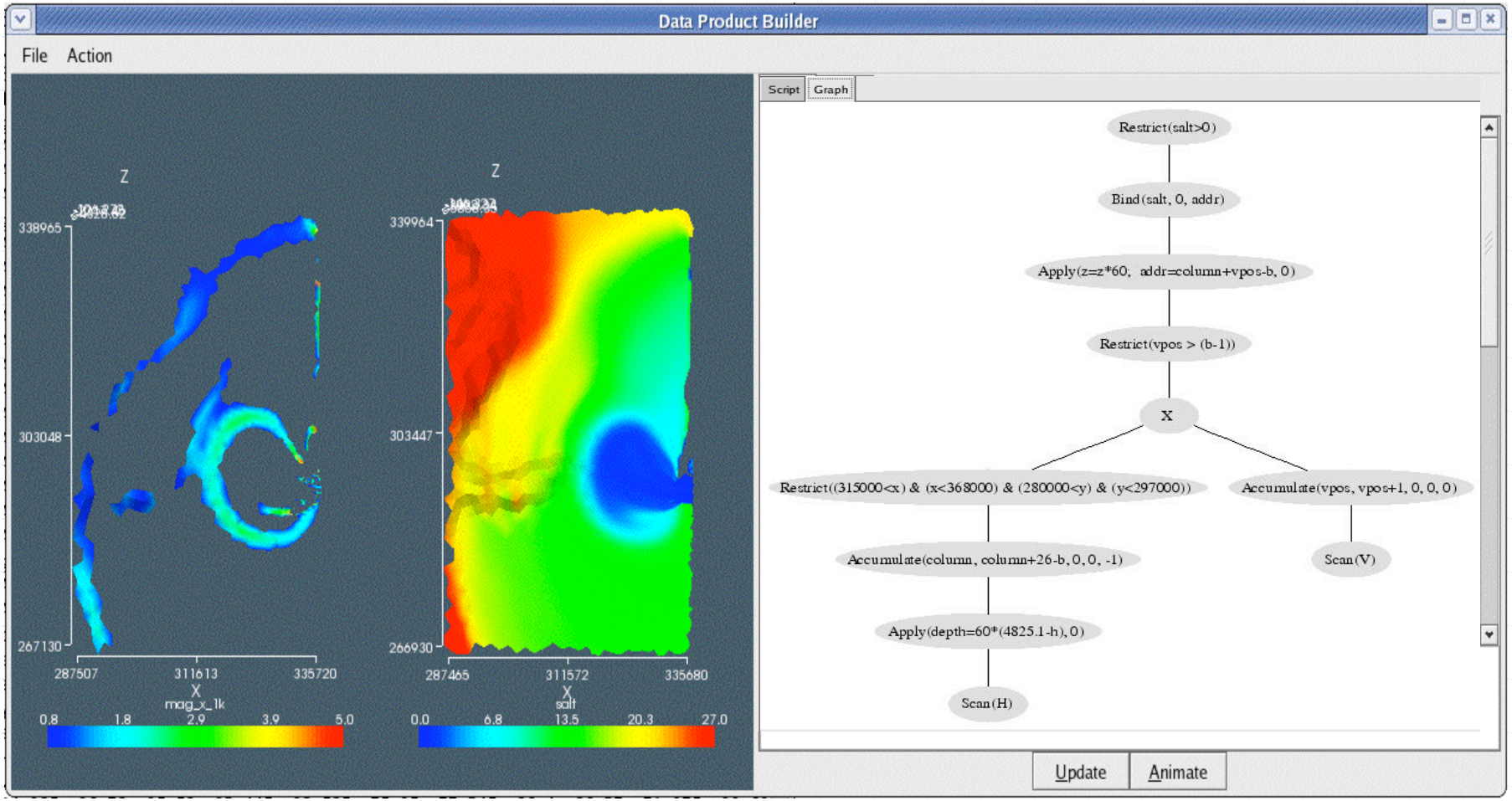
  $[d \mid contains(d,c), d \in H_2, c \in P_0]$
- aggregation function:

  interpolate

Define a Product: Plume Front



DPBuilder

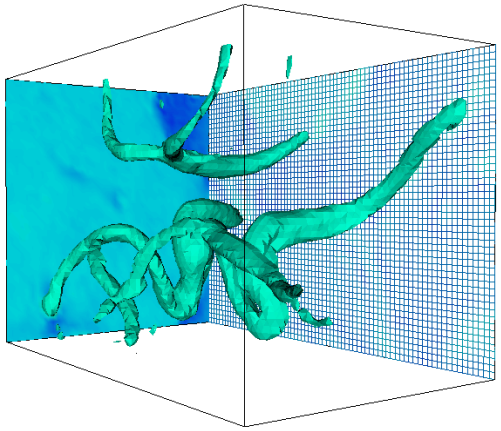

Data Product Builder
File
Action
Z
338965
303048
267130
287507
311613
335720
X
mag_x_1k
0.8
1.8
2.9
3.9
5.0
Z
339964
303447
266930
287465
311572
335680
X
salt
0.0
6.8
13.5
20.3
27.0
Script
Graph
Restrict(salt>0)
Bind(salt, 0, addr)
Apply(z=z*60; addr=column+vpos-b, 0)
Restrict(vpos > (b-1))
X
Restrict((315000<x) & (x<368000) & (280000<y) & (y<297000))
Accumulate(vpos, vpos+1, 0, 0, 0)
Accumulate(column, column+26-b, 0, 0, -1)
Scan(V)
Apply(depth=60*(4825.1-h), 0)
Scan(H)
Update
Animate

Roadmap

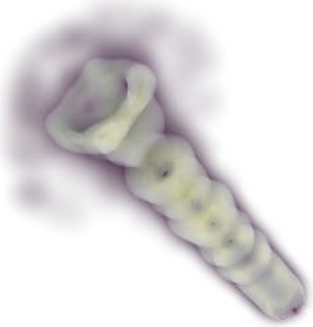
- Introduction
- Grids and GridFields
- Uses of GridFields
- *Experimental Evaluation*
- Conclusions and Outlook

One Test: Determine the Cost of Abstraction

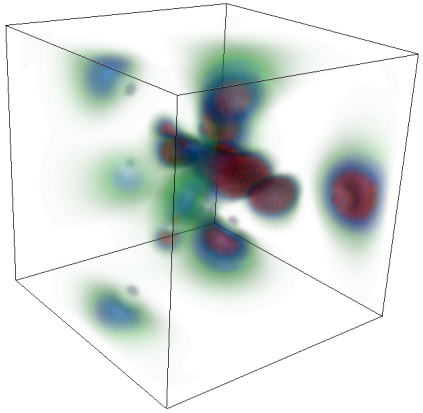
- Structured grids admit implicit representations
  - incidence tests and
  - subslab extraction are efficient
- The gridfield implementation uses explicit cells, but can omit unneeded cells on a per-recipe basis
- Some tasks require that the implicit cells be materialized anyway



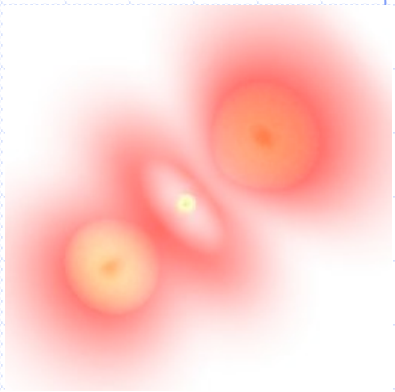
Carotid
(76 x 49 x 45)



Fuel
(64 x 64 x 64)

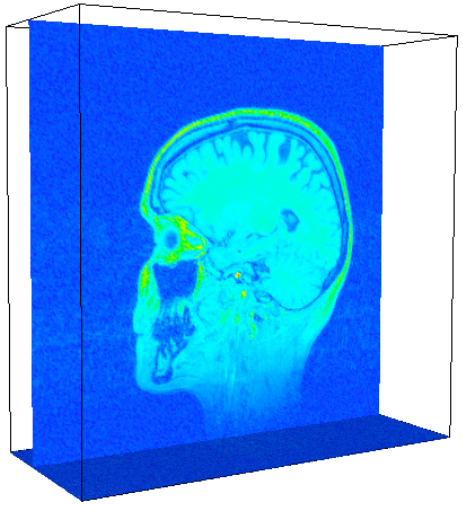


IronProt
(68 x 68 x 68)

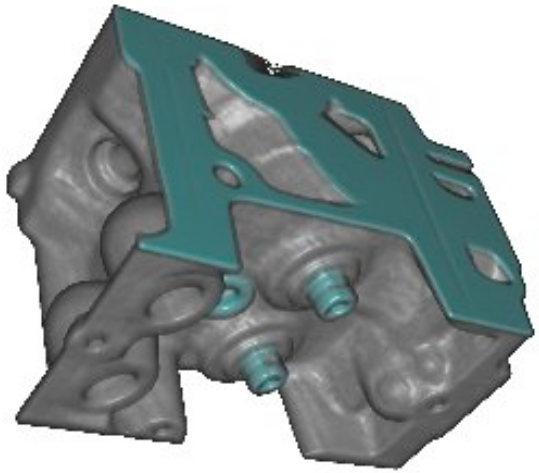


Hydrogen
(128 x 128 x 128)

Tomato
(256 x 256 x 64)

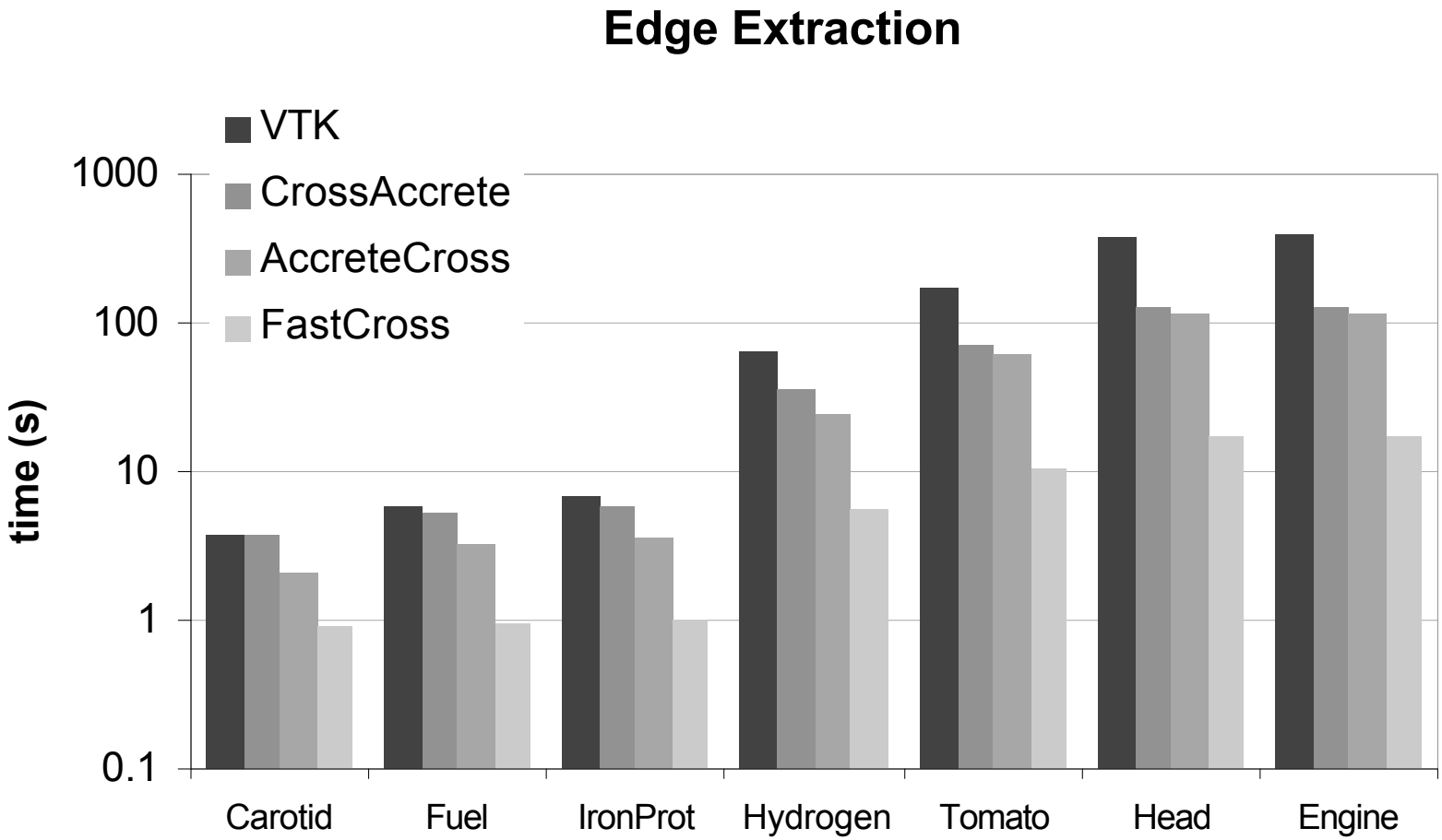


Head
(256 x 256 x 109)



Engine
(256 x 256 x 110)

Extracting Edges



Other Experiments

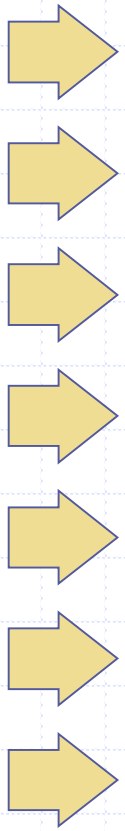
- Relational Implementation
- Cross Product Implementations
- Benefits of Bind/Restrict Commutativity
- Predicate Order on Irregular Grids
- Analysis of Alternative Data Structures

Roadmap

- Introduction
- Grids and GridFields
- Operators and Optimization
- Experimental Evaluation
- Conclusions and Outlook
  - Final Example
  - Other Results
  - Contributions

Additional Examples

- Plume Definition
- Regularize a Grid
- Effects of Predicate Order
- Plume Volume
- Minimum Cell Diameter
- Persistent Eddies
- Feature Extraction



Other Results

- Analysis and evaluation of alternative grid representations (Chapter 3)
- Working with ad hoc file formats (Chapter 4)
- Programming interfaces (Chapter 5)
- Prototype application (Section 5.3.4)
- Relational implementation (Section 3.2.6, 6.3)

Contributions

- As a reasoning tool
  - Howe, Maier, Baptista, *A Language for Spatial Data Manipulation*, Journal of Environmental Informatics, November 2003
- Implementation and algebraic optimization
  - Howe, Maier, *Algebraic Manipulation of Scientific Datasets*, VLDB 2004.
  - One of four papers selected for publication in the VLDB Journal "Best of Conference" issue.
- Prototype demonstration
  - Howe, Maier, *Querying and Visualizing Gridded Datasets for e-Science*, ICDE 2005 (demo).
- Interfacing with In Situ Data
  - Howe, Maier, *Retrofitting A Model to Existing Environmental Data*, SSDBM 2005

Thesis Redux

Programs that create or manipulate gridded datasets exhibit an algebraic structure that can be used to facilitate reasoning about them and improve their performance.

We express data products in oceanography, medicine, and seismology

We can express efficient algorithms, and we demonstrate the efficacy of algebraic optimization

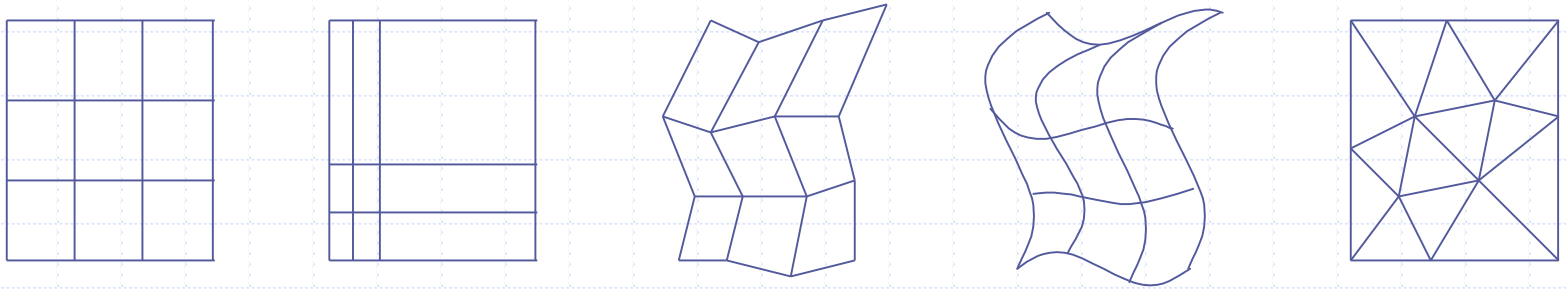
E.g., we capture the differences between the evolving CORIE grids

Acknowledgements

- This work supported by the NSF ITR program
- Many thanks to Antonio Baptista and the entire CORIE/CMOP team

CORIE

CMOP
Center for Coastal
Margin Observation
& Prediction



Enhanced Arrays: netCDF

```
netcdf deform.nc {
dimensions:
        x = 7 ;
        y = 5 ;
variables:
        float x(x) ;
                x:units = “centimeters” ;
        float y(y) ;
                y:units = “centimeters” ;
        float u(x,y) ;
                u:name = “displacement” ;
                u:units = “centimeters” ;
// global attributes
        :name = “thin plate deformation” ;
data:
        x    = 0.1, 0.5, 0.8, … ;
        y    = 2.2, 2.25, 2.8, … ;
        u    = 0.004, 0.006, 0.007, … ;
}
```

dimension (name, size)

variables

by convention, one variable per dimension has the same name as the dimension.

type

u(x,y) means u is a function of x, y

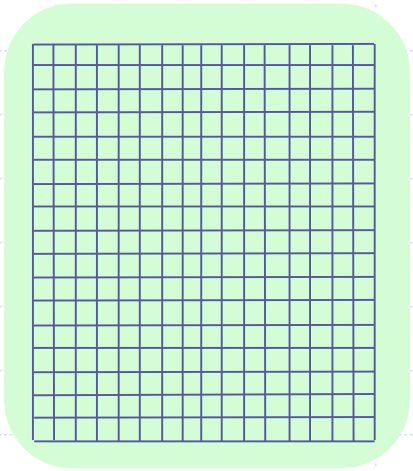
metadata

data

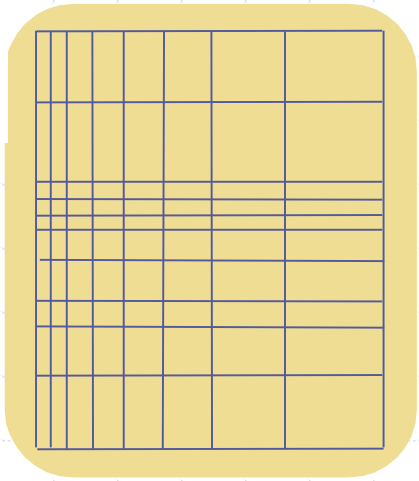
ASCII format for illustration only

Expressiveness of NetCDF

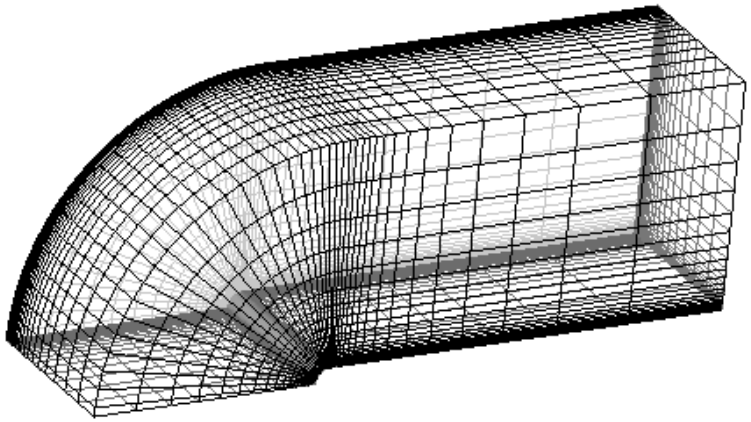
a)



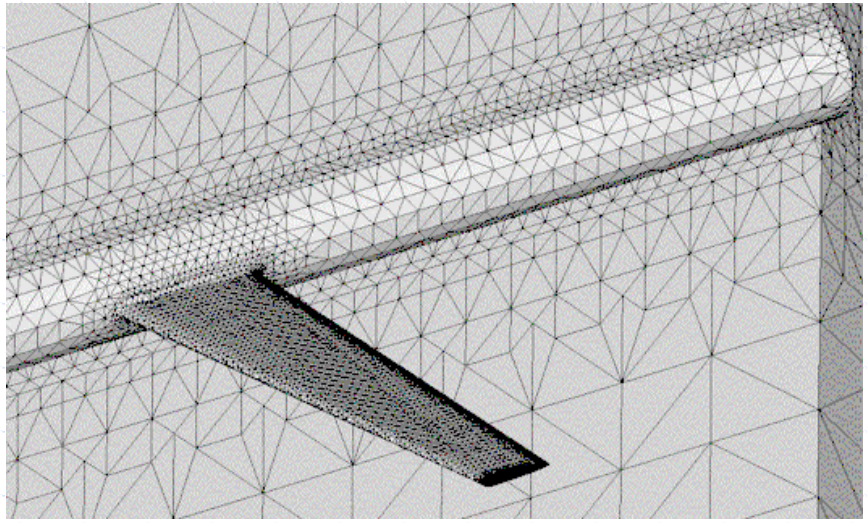
b)



c)



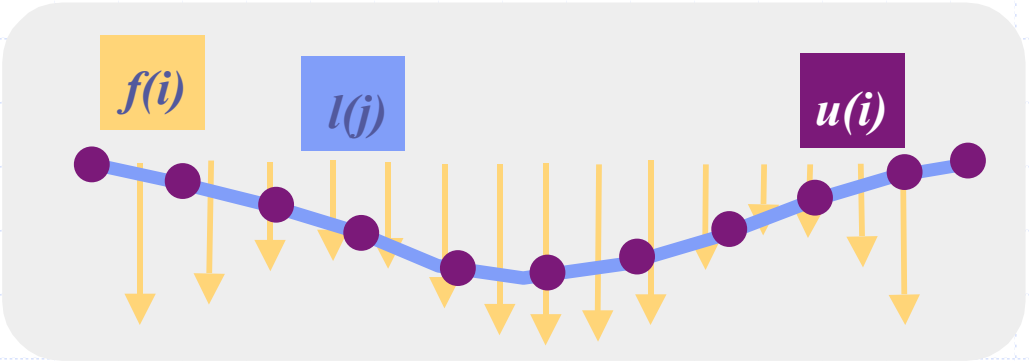
d)

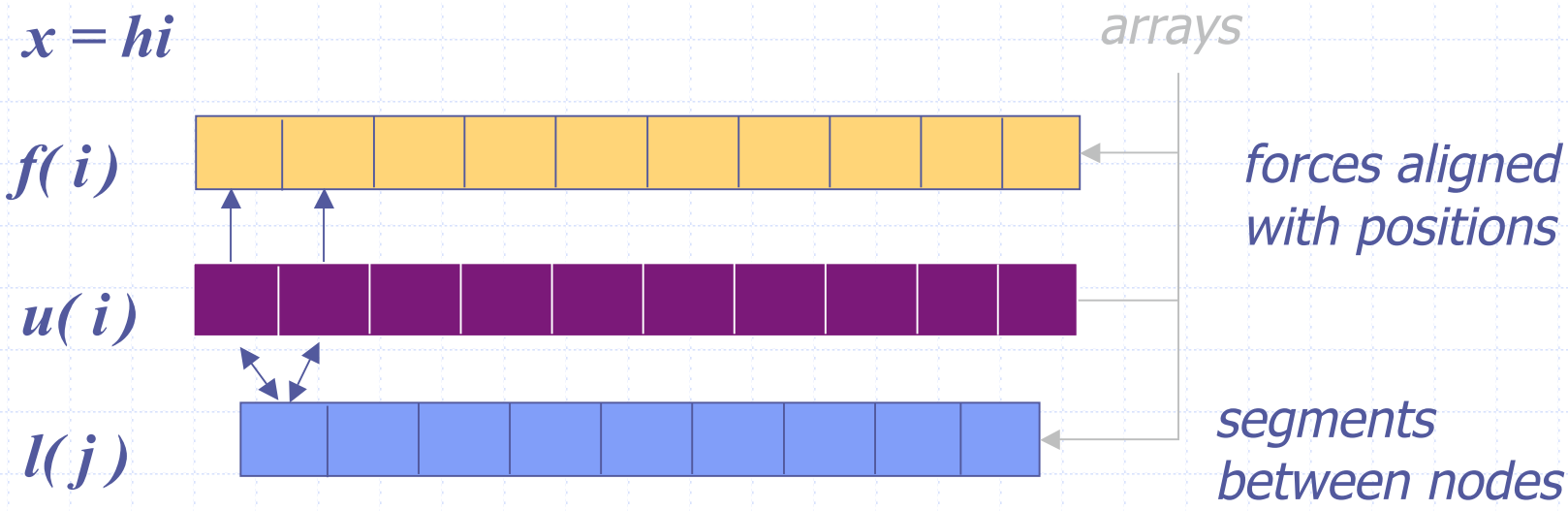


Future Work

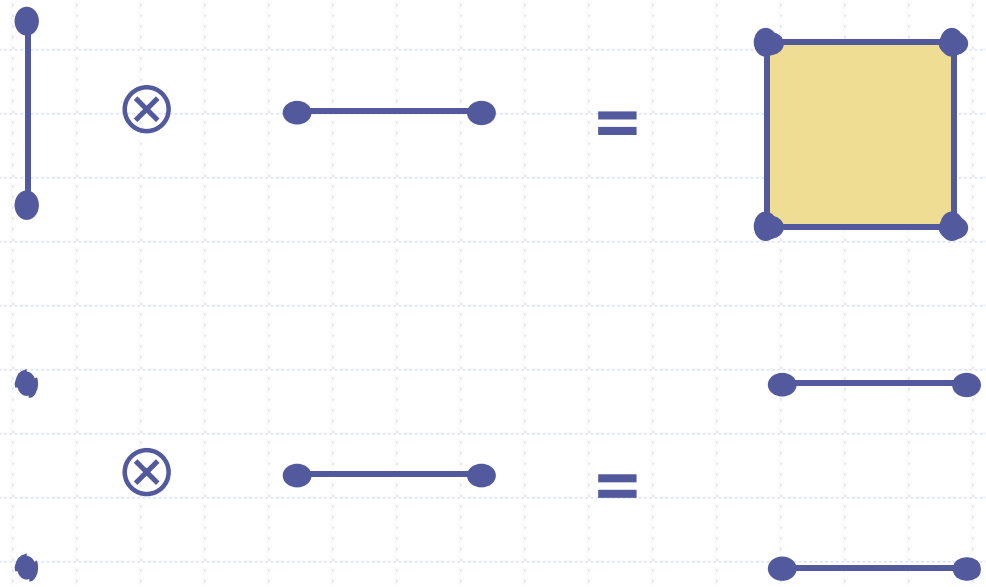
- Automate Optimization
- Characterize the performance gain using "reduced" grids
- Accommodate specialized representations and algorithms

Bare Arrays

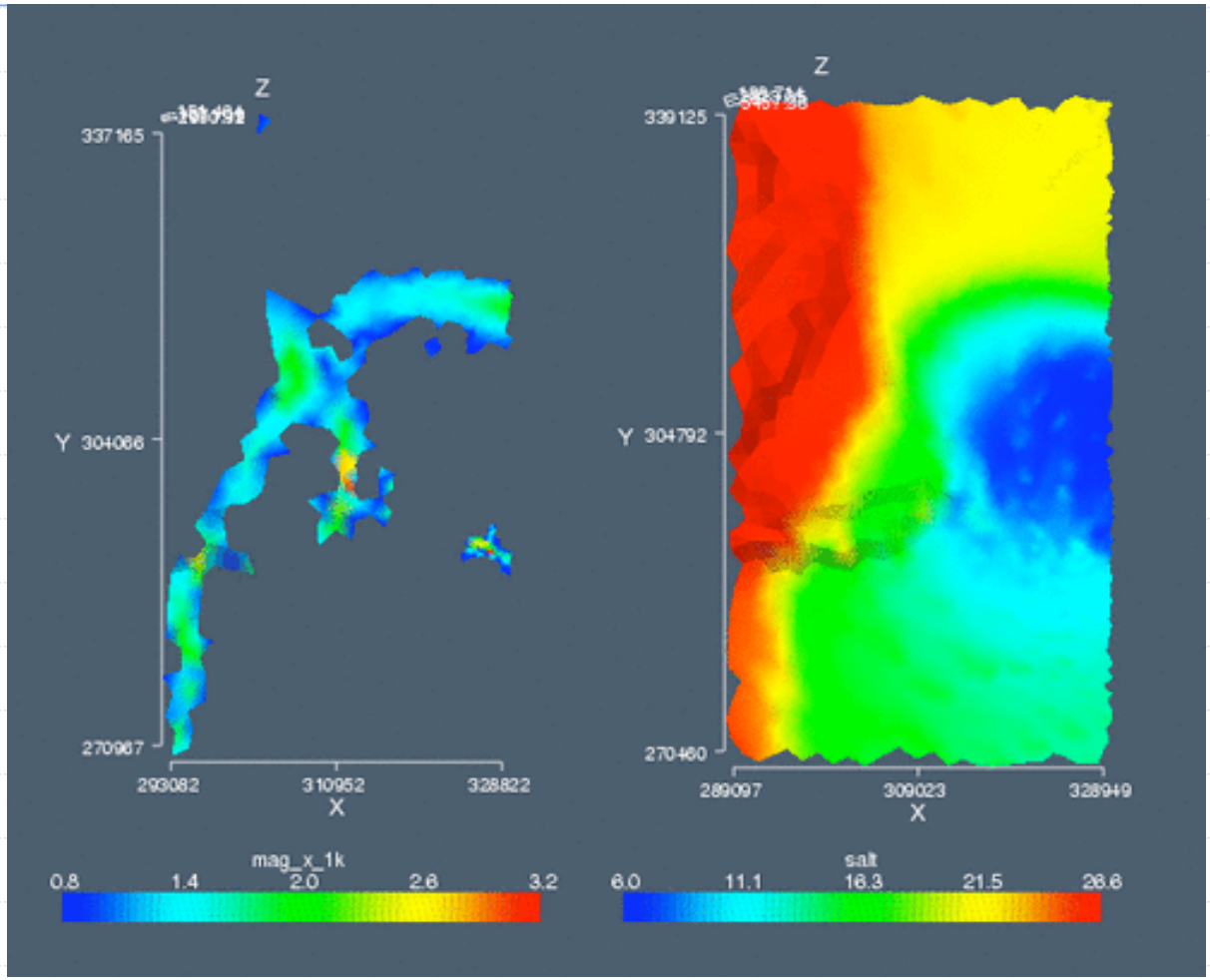

f(i)
l(j)
u(i)


x = hi
arrays
f(i)
u(i)
l(j)
forces aligned
with positions
segments
between nodes

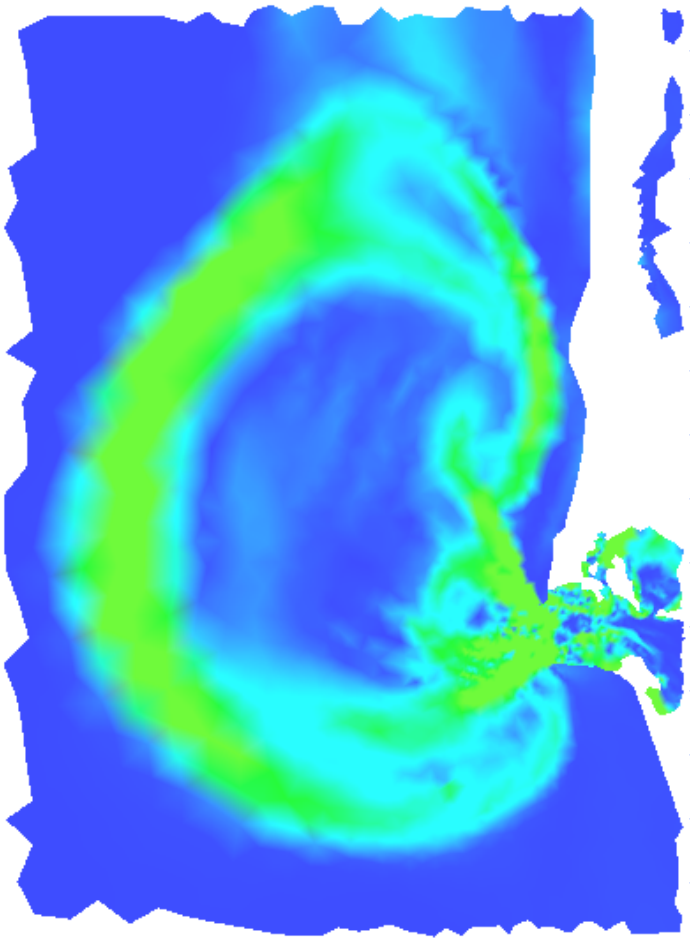
Cross Product (1)



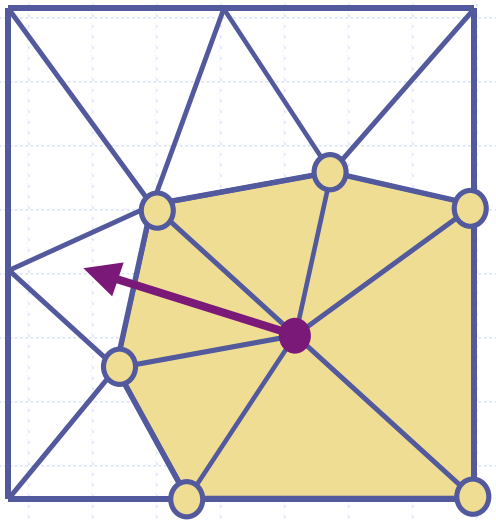
Example: Plume Definition (1/3)



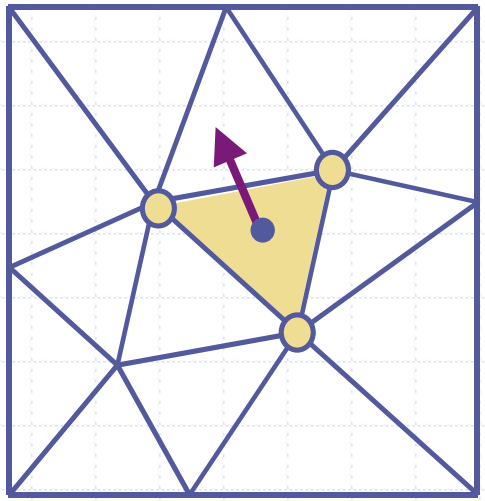
Example: Plume Definition (2/3)



(a)



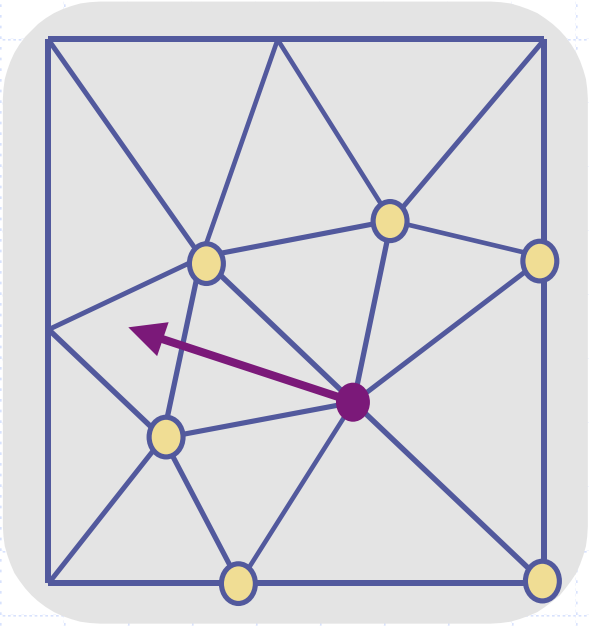
(b)



Example: Plume Definition (3/3)

Estimating Gradient

$G_{0:(x,y,s)}$

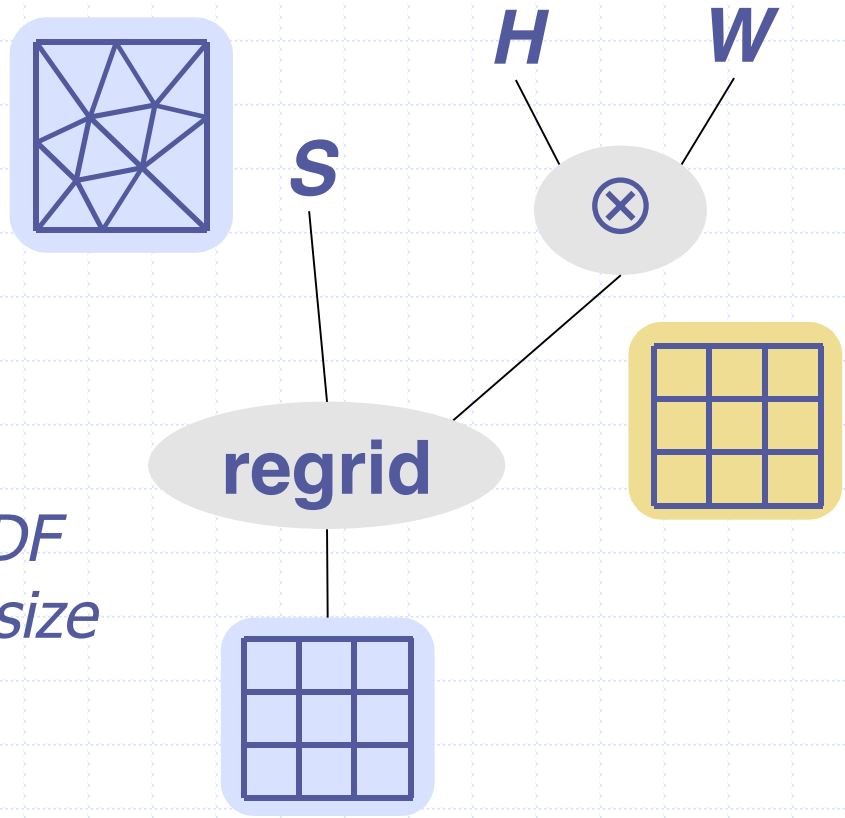


- Source GridField: **G**
- Target GridField: **G**
- assignment function:

  $[c \mid c < d > e,\ d \in G_2,\ c,e \in G_0]$
- aggregation function:

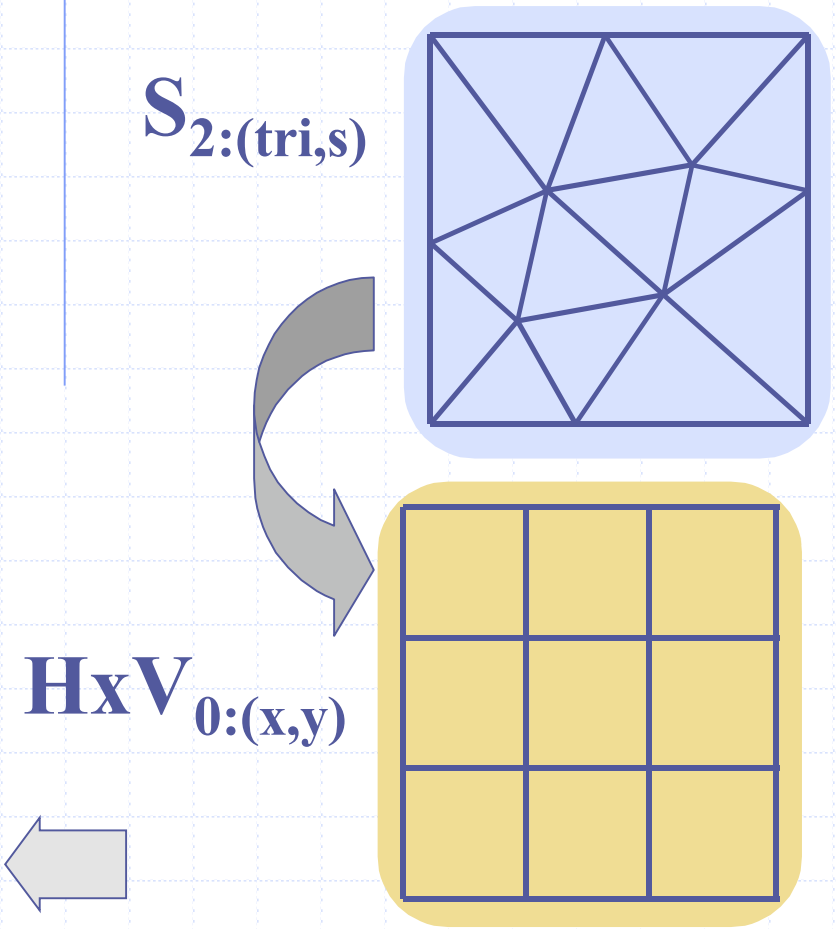
  add up the vectors

Example: Regularize a Grid (1/2)



Used to generate netCDF files with user-defined size and resolution

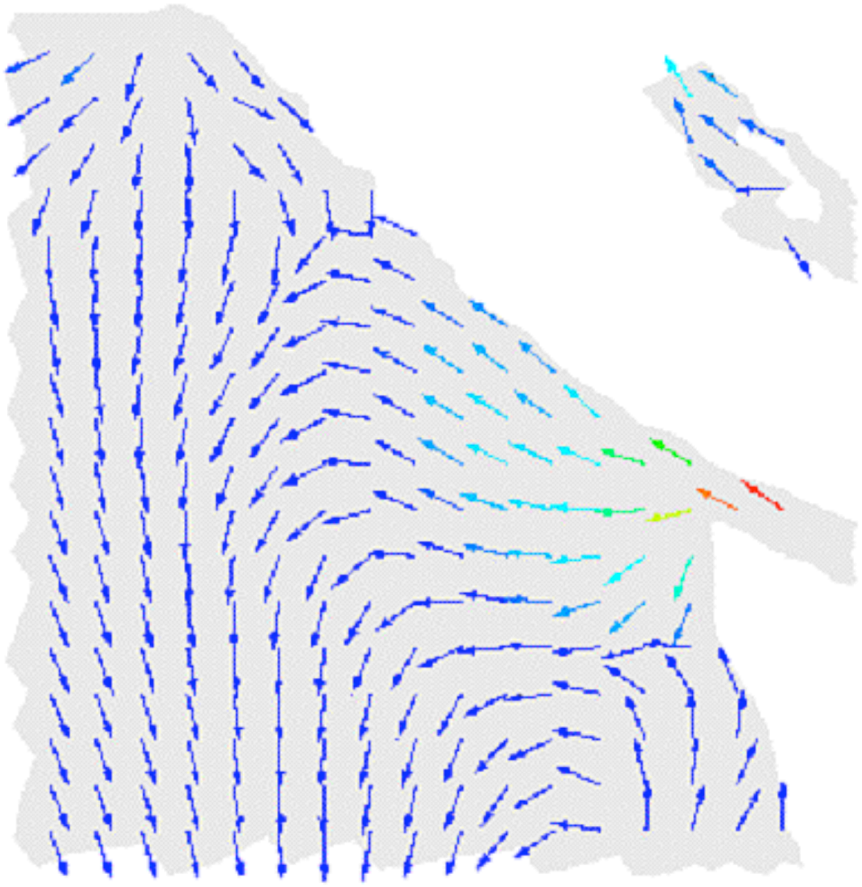
Example: Regularize a Grid (2/2)



- Source GridField: **S**
- Target GridField: **HxV**
- assignment function:
  [d | contains(d,c), d $\in G_2$, c $\in F_0$]
- aggregation function:
  interpolate

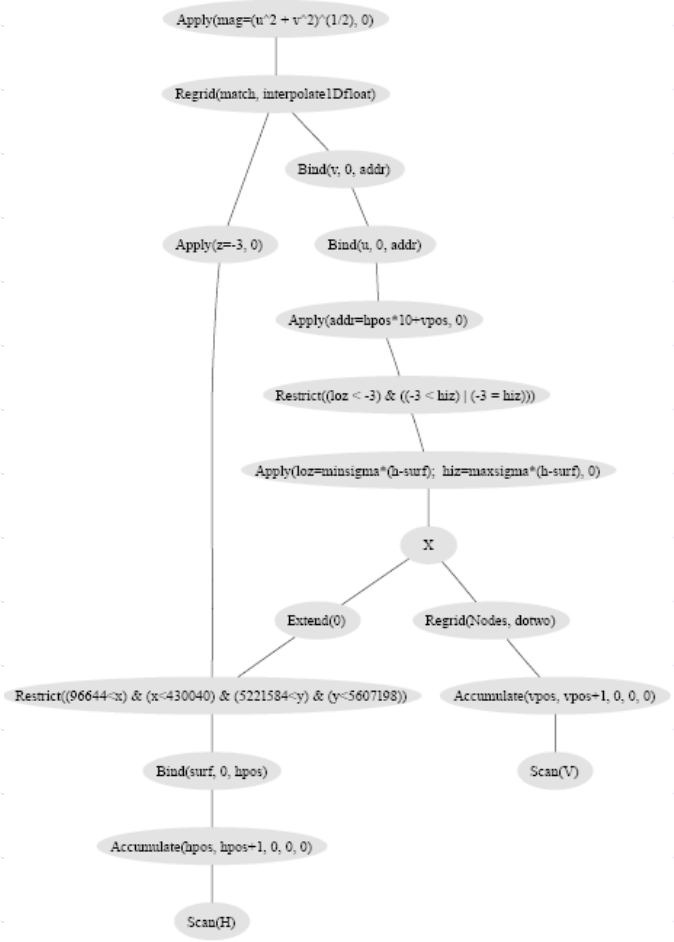
implements a spatial join

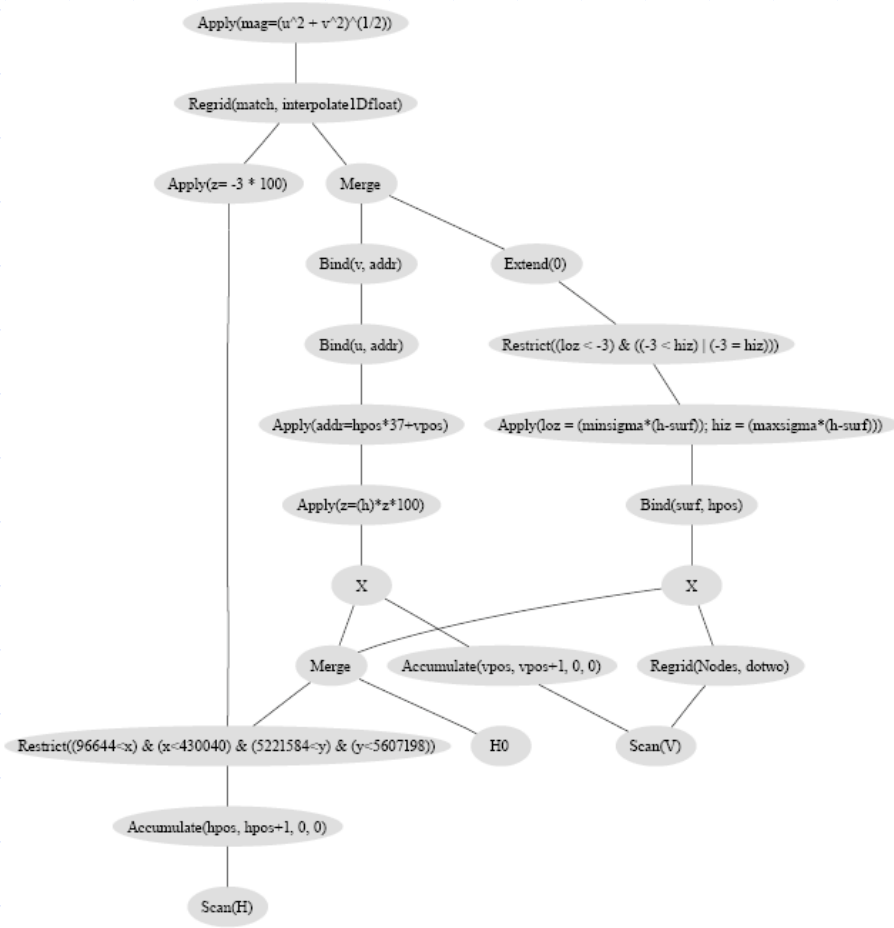
Example: Persistent Eddies (1/3)



- Average over time
- Interpolate at depth

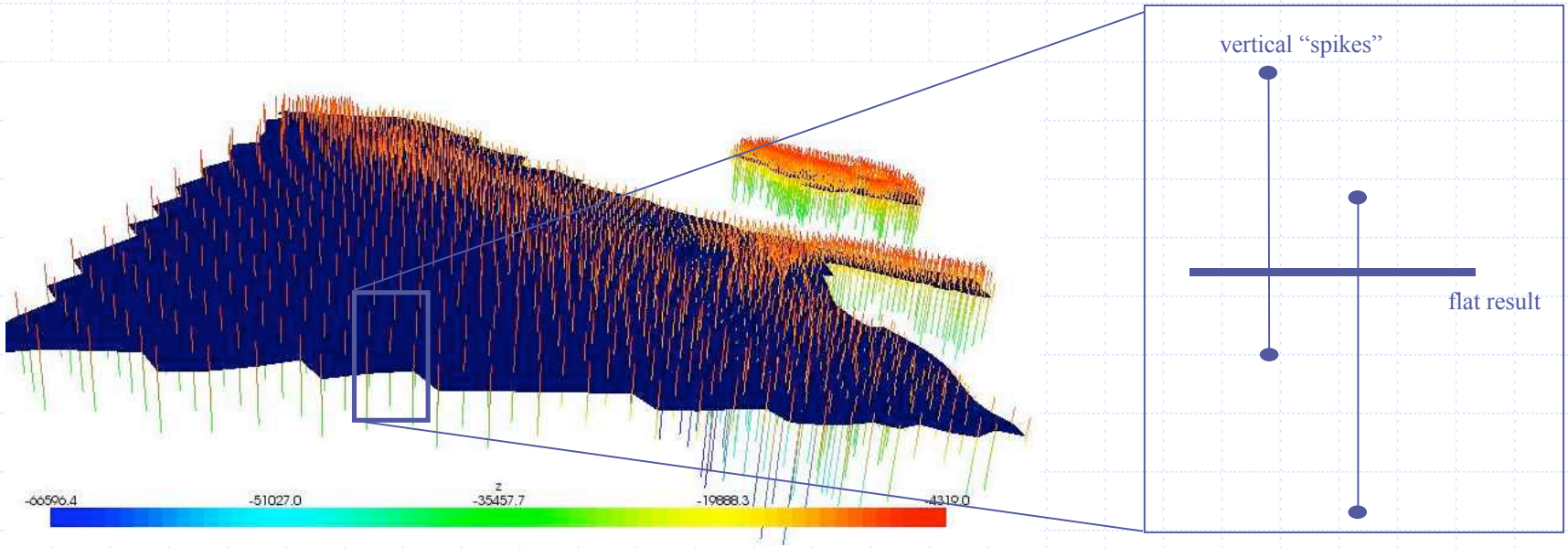
Example: Persistent Eddies (2/3)



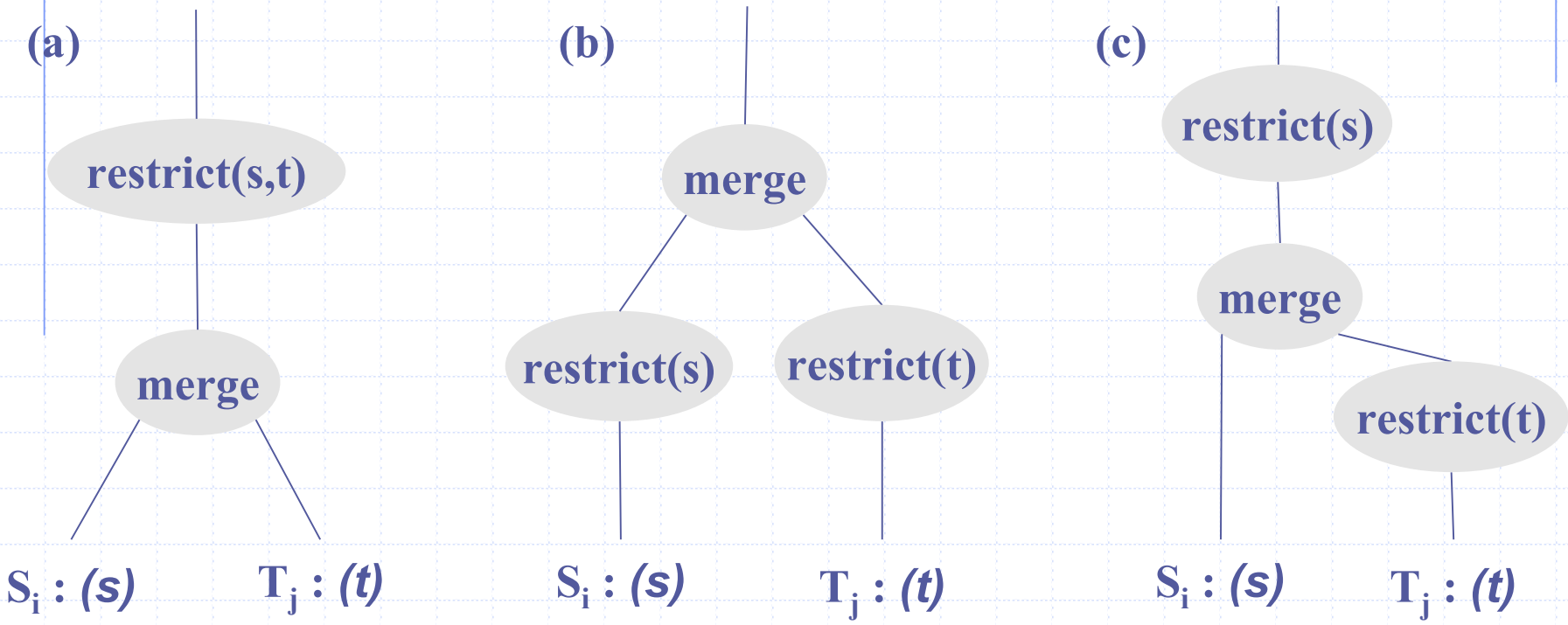


Example: Persistent Eddies (3/3)

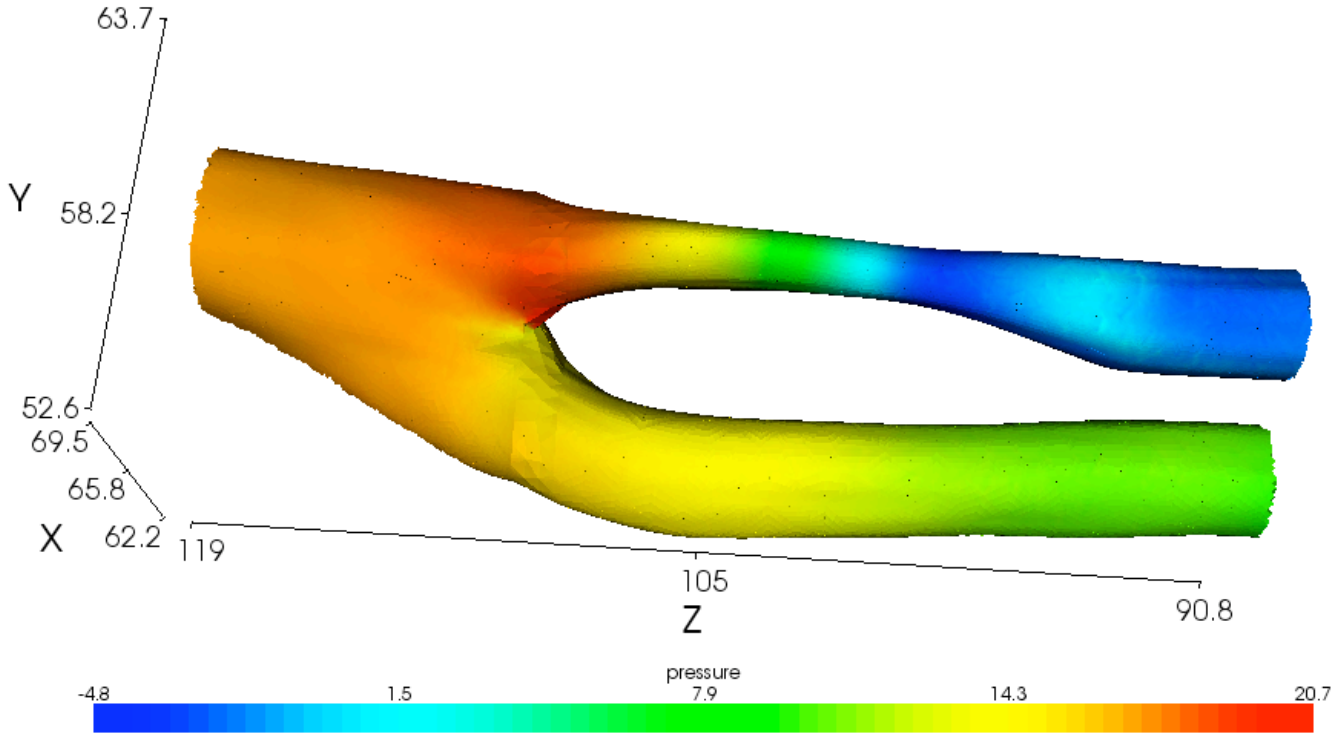
Interpolate over Depth



Example: Predicate Order (1/4)

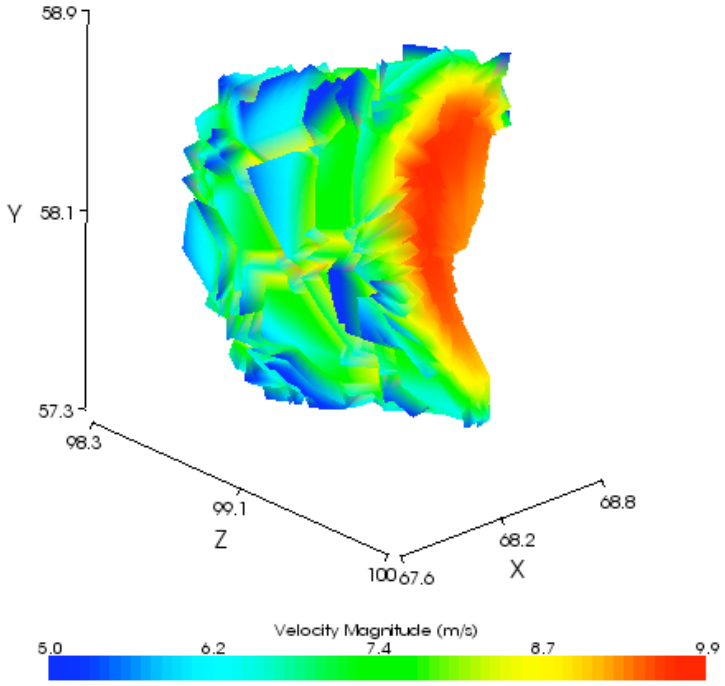


Example: Predicate Order (2/4)

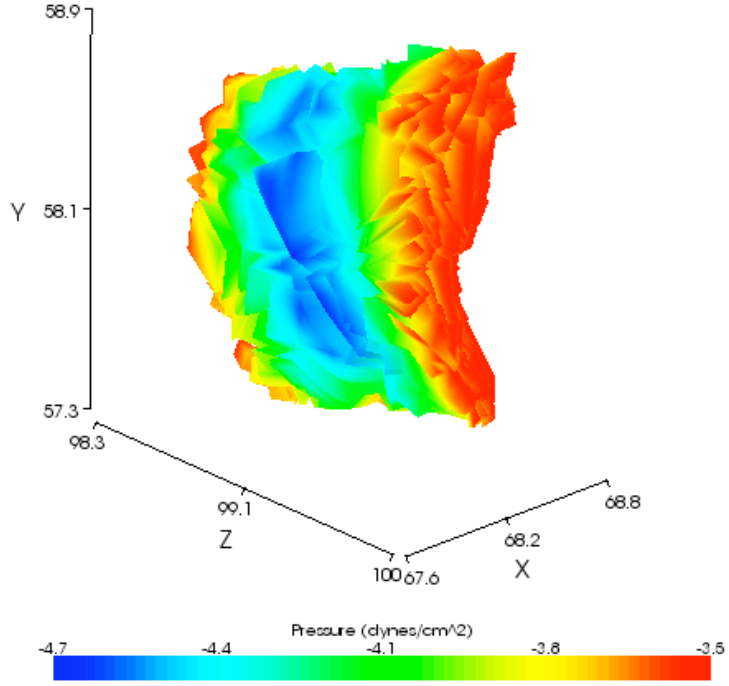


pressure

Example: Predicate Order (3/4)



Velocity



Pressure

Example: Predicate Order (4/4)

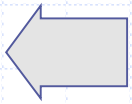
$$\text{merge}(\text{restrict}_{pressure}(\mathbf{S}), \text{restrict}_{velocity}(\mathbf{T})) \quad : \quad 20.15\text{s}$$

$$\text{restrict}_{pressure}(\text{merge}(\mathbf{S}, \text{restrict}_{velocity}(\mathbf{T})) \quad : \quad 13.06\text{s}$$

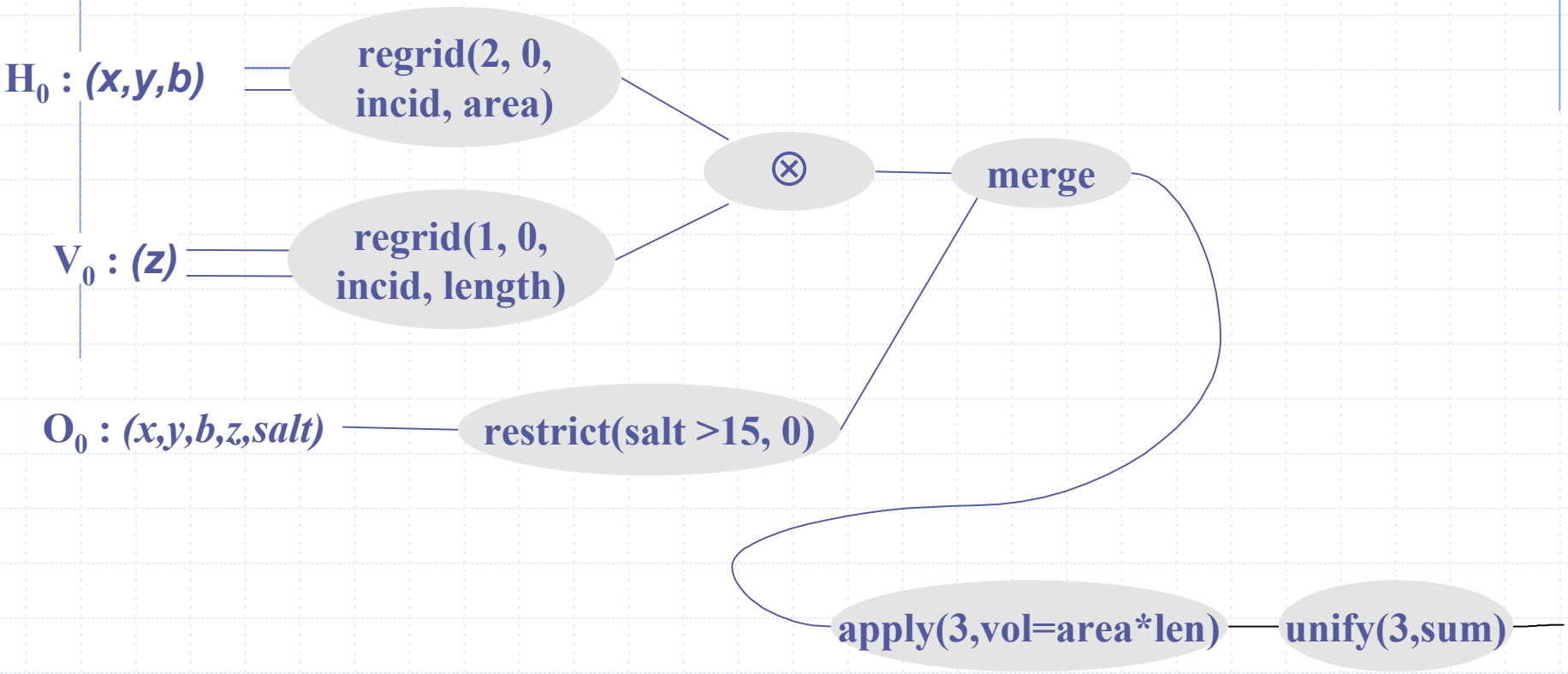
$$\text{restrict}_{velocity}(\text{merge}(\text{restrict}_{pressure}(\mathbf{S}), \mathbf{T})) \quad : \quad 10.53\text{s}$$

$$\text{restrict}_{velocity}(\text{restrict}_{pressure}(\text{merge}(\mathbf{S}, \mathbf{T}))) \quad : \quad 12.77\text{s}$$

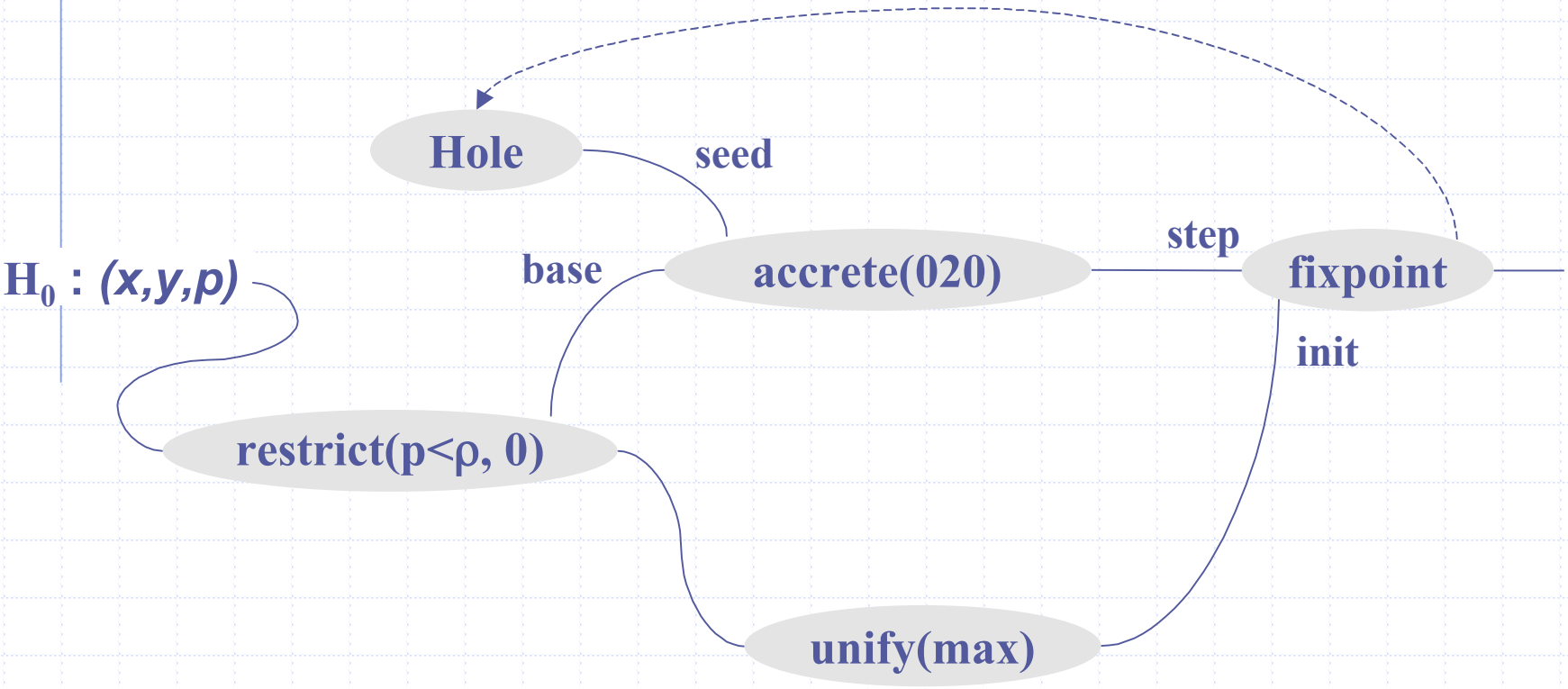
$$\text{restrict}_{velocity \wedge pressure}(\text{merge}(\mathbf{S}, \mathbf{T}))) \quad : \quad 10.75\text{s}$$



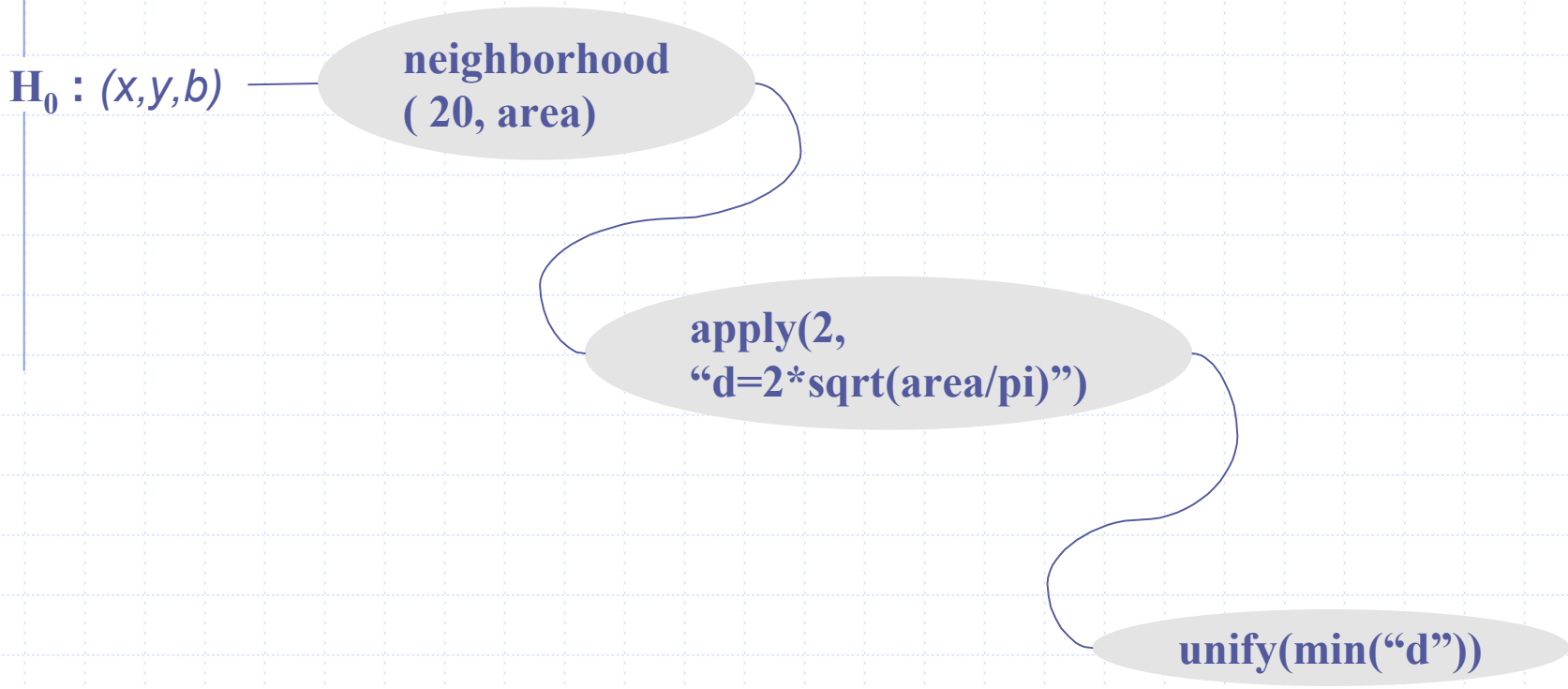
Example: Plume Volume



Example: Feature Extraction



Example: Minimum Cell Diameter



Subgrids and Compatibility

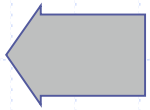
$$E \subset F$$

$$\equiv (X, \preceq_E) \subset (Y, \preceq_F)$$

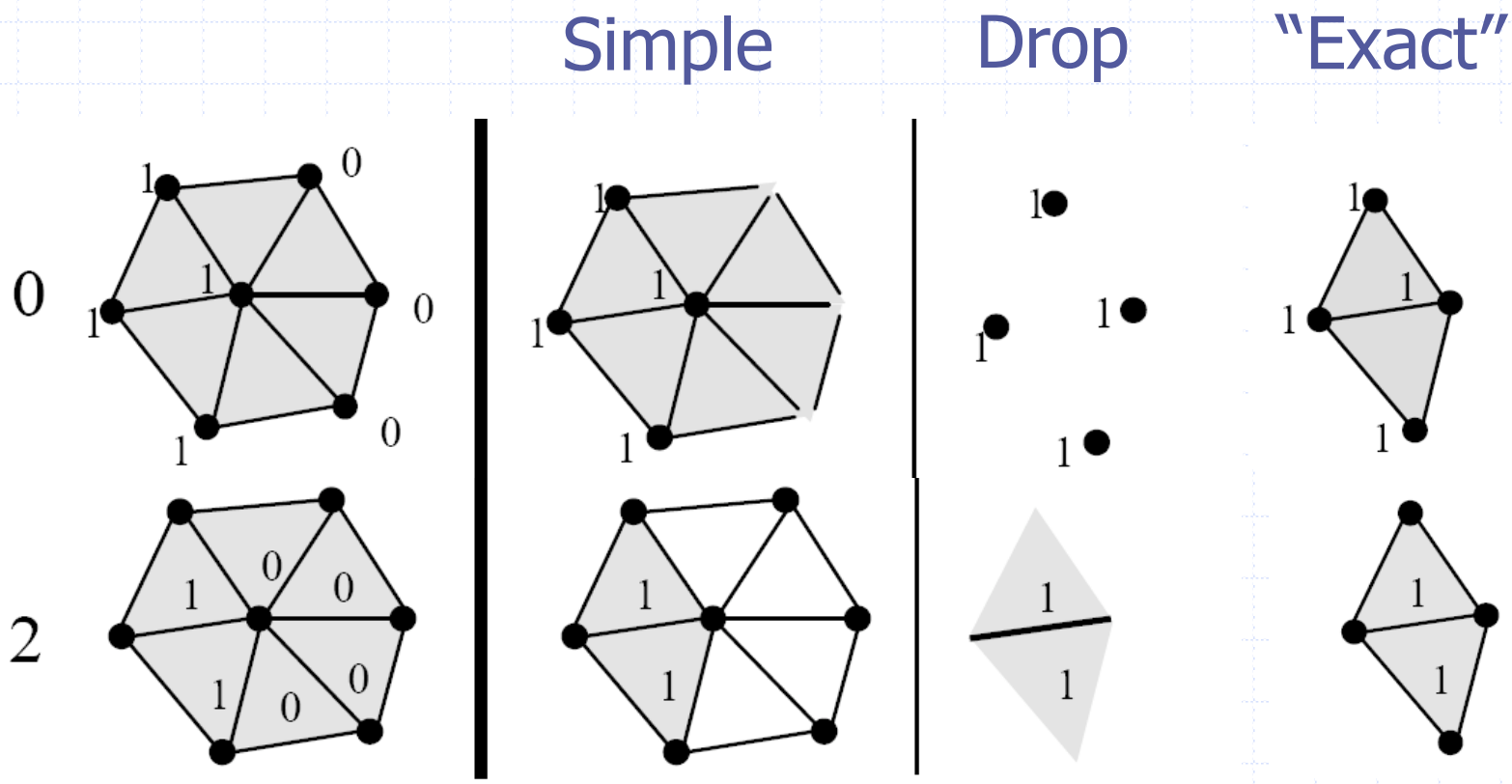
set of cells

incidence relation

$$X \subset Y \qquad \forall x, y \in X.\ x \preceq_E y \Leftrightarrow x \preceq_F y$$



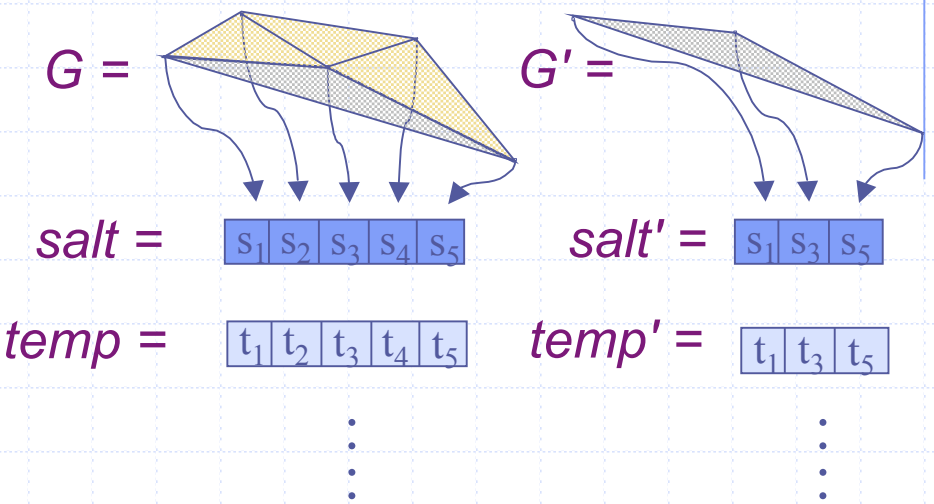
Restrict: Which cells are kept?

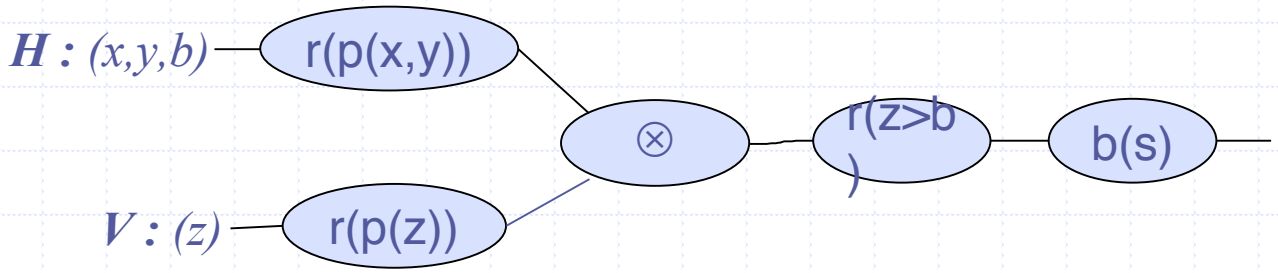


Simple, Drop, and other semantics expressible with the Accrete operator

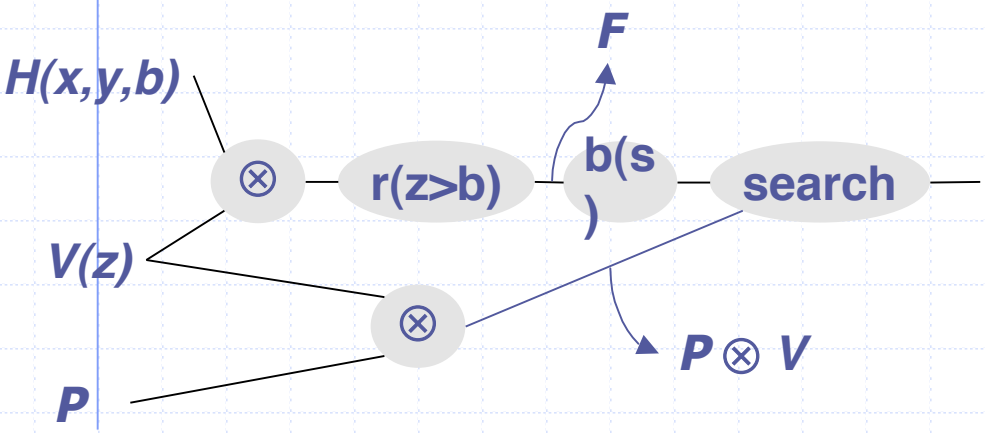
Bind as a Merge Join

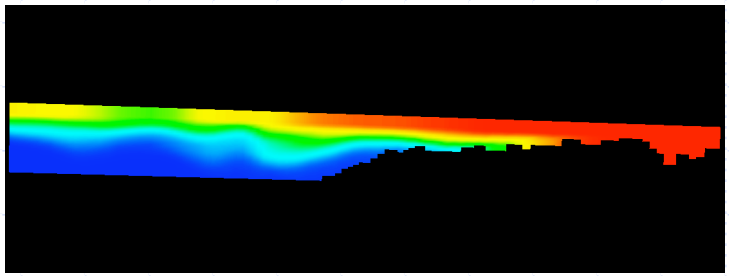
- salt,temp defined on *G*
- Materialize pointers to elements of salt, temp
- Bind salt, temp to a subgrid of *G, G'*



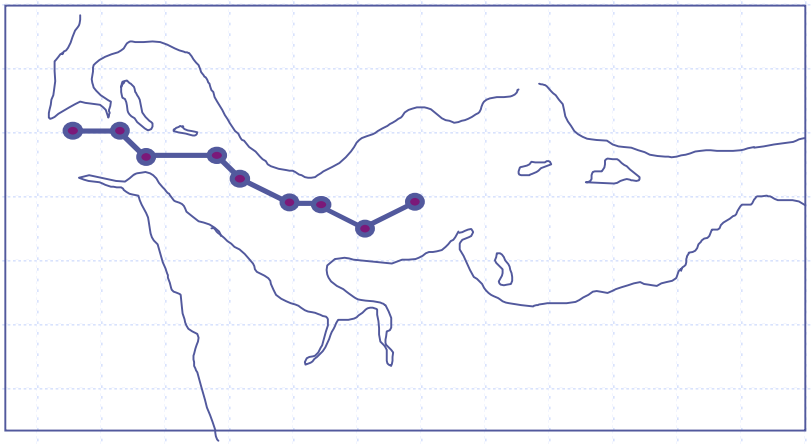


Transect (Vertical Slice)

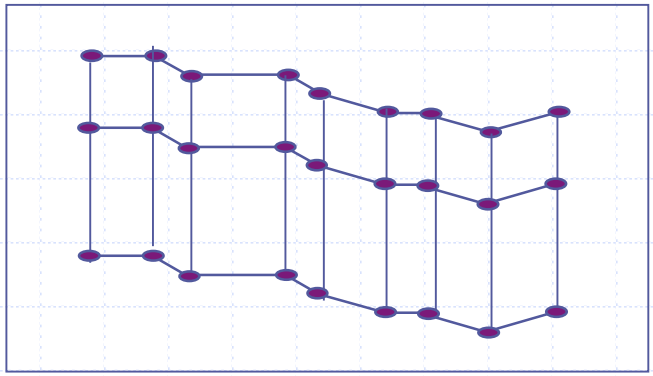




P



P ⊗ V

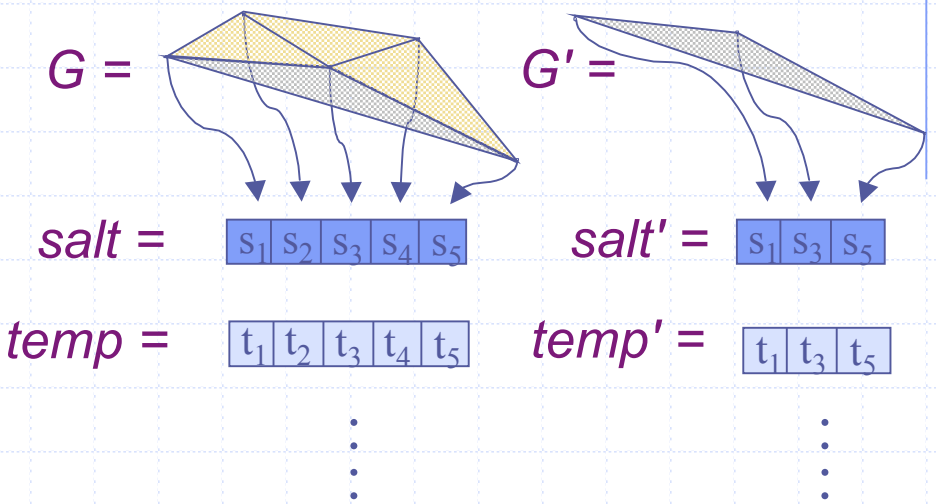


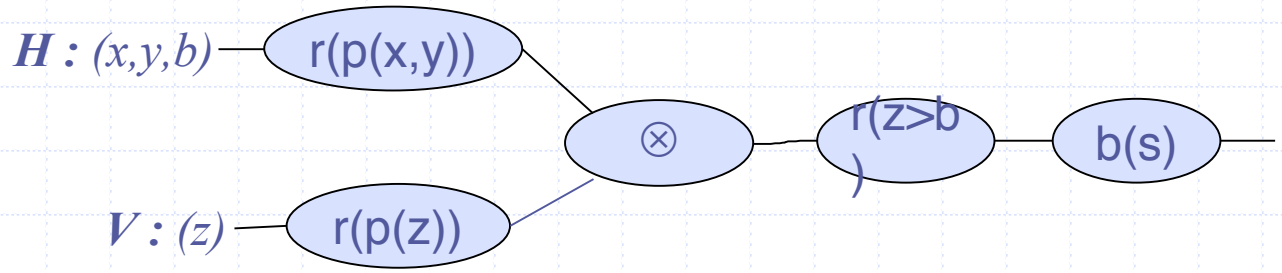
Lessons Learned

- Strive for *Middleware*
  - Output as important as input
- Model useful prior to implementation
  - Documentation
  - "Conversation Lens"
- Pay as you go
  - Deploy mini-apps on demand

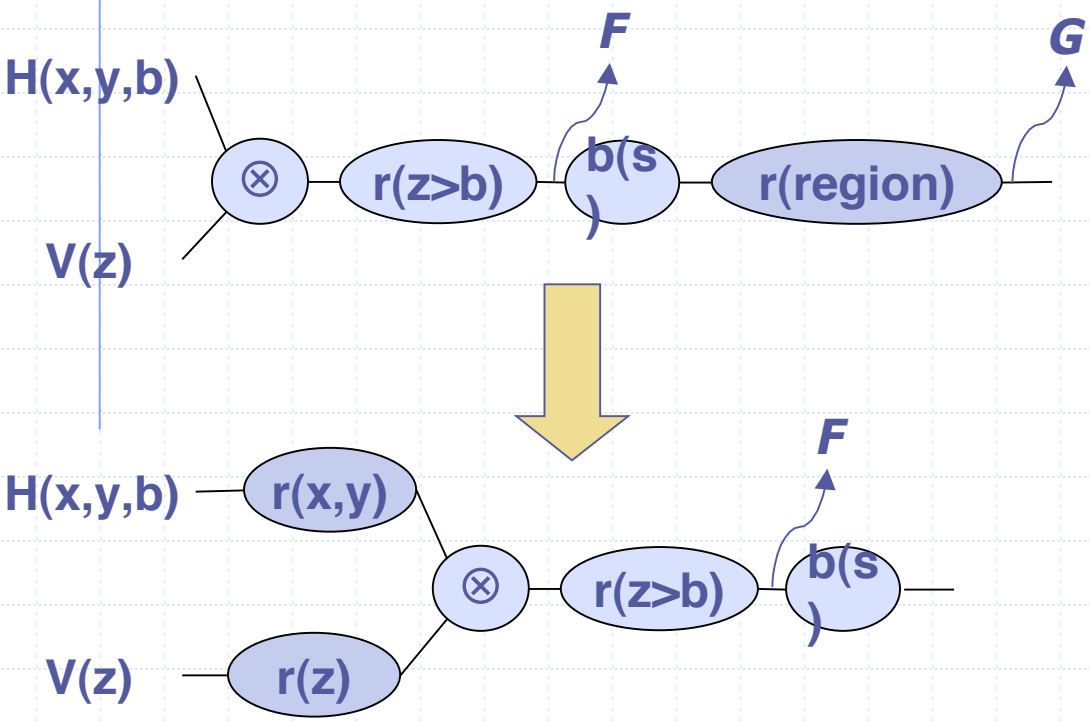
Optimization: Push Restricts

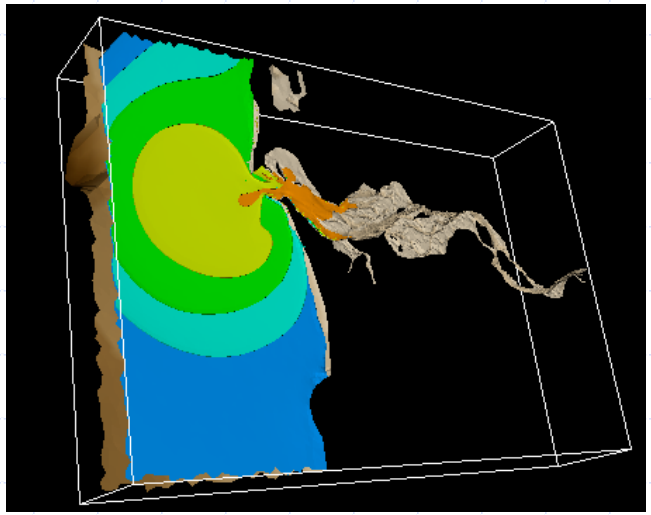
- salt,temp defined on *G*
- Materialize pointers to elements of salt, temp
- Bind salt, temp to a subgrid of *G, G'*





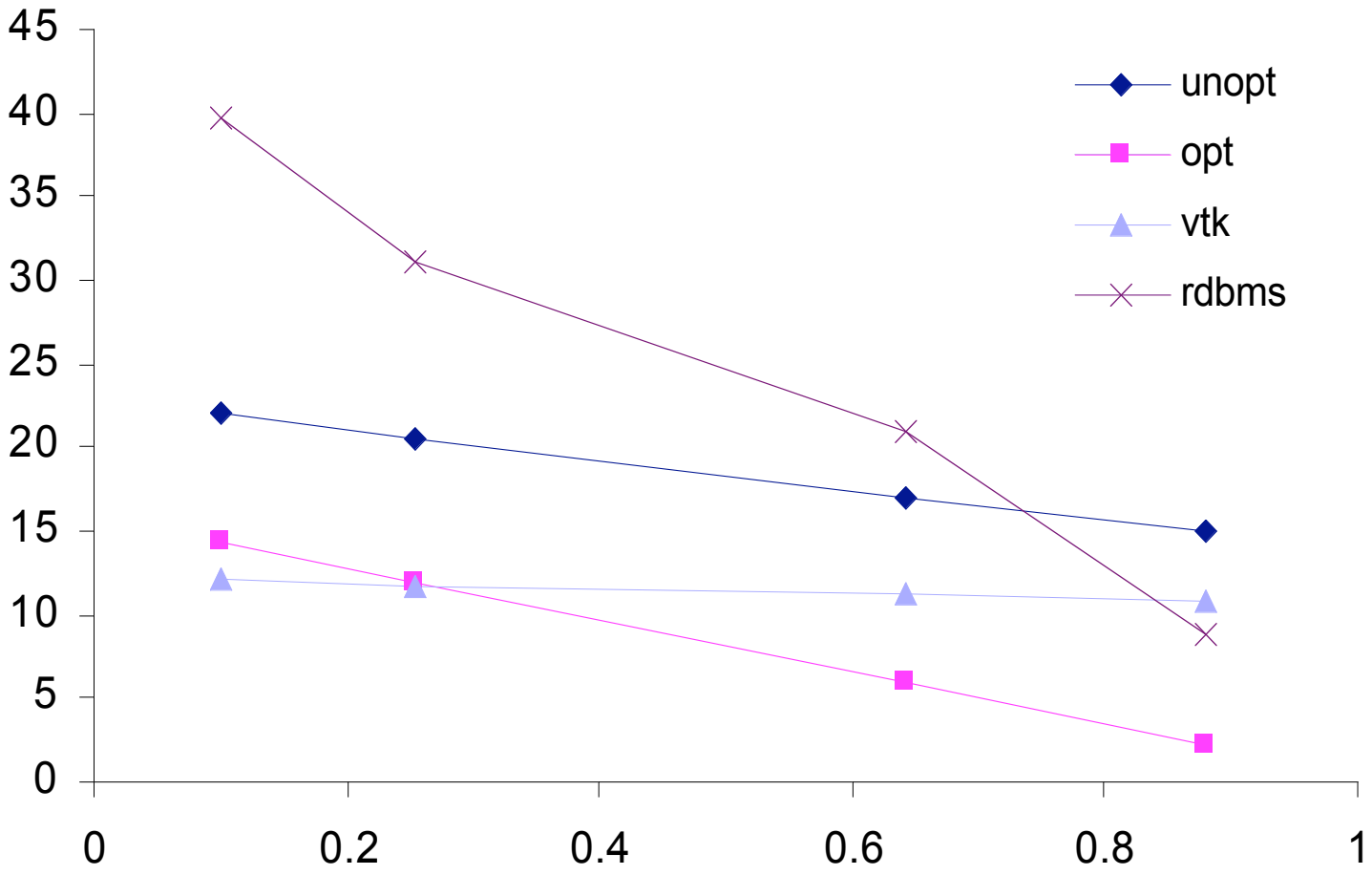
Optimization: Push Restricts

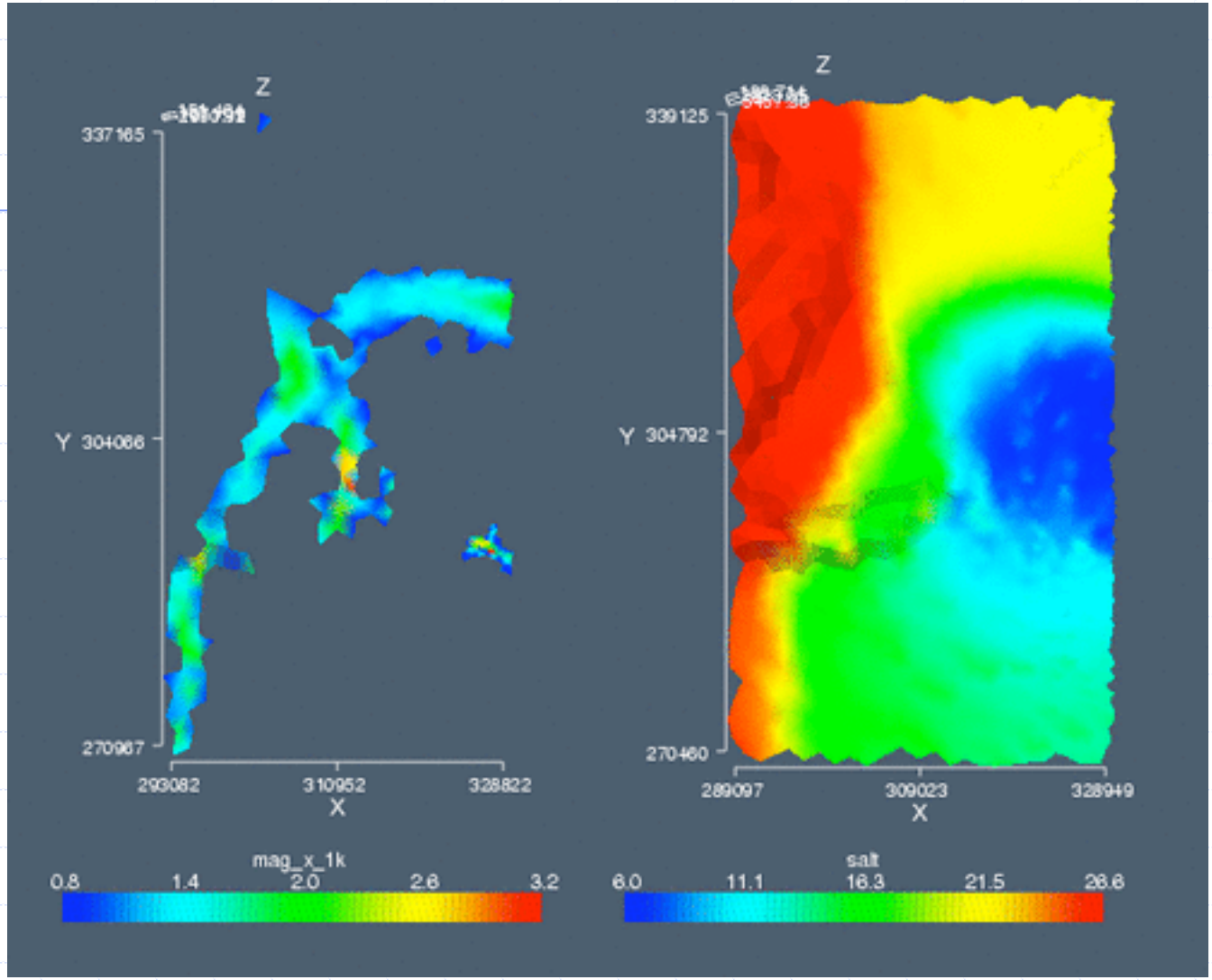




*Howe, Maier, Algebraic Manipulation of Scientific Datasets. VLDB Journal, 14:4, 2005

Optimization: Push Restricts

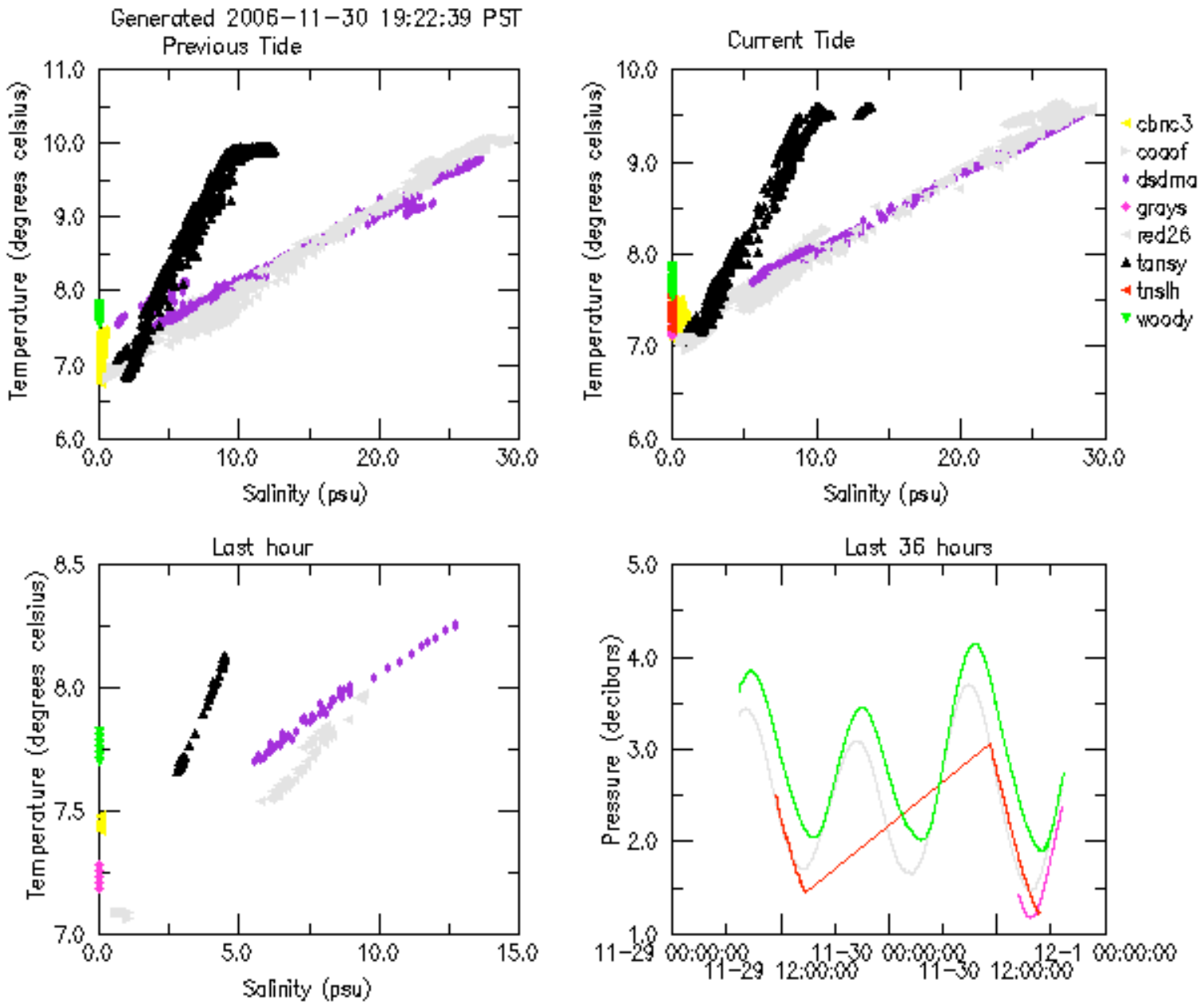




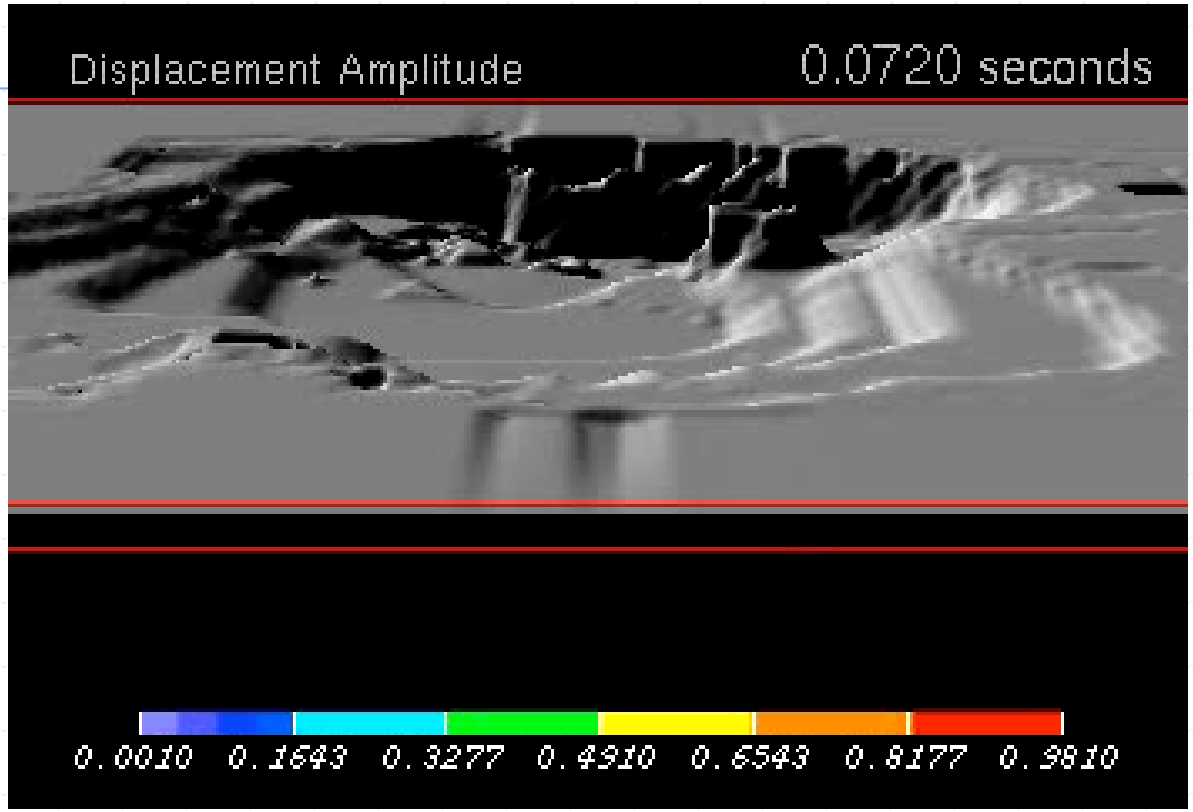
river mouth

Salinity gradient

Salinity


Generated 2006-11-30 19:22:39 PST
Previous Tide
Temperature (degrees celsius)
Salinity (psu)
Current Tide
cbnc3
cooof
dsdma
grays
red26
tansy
tnslh
woody
Last hour
Last 36 hours
Pressure (decibars)
11-29 00:00:00
11-30 00:00:00
12-1 00:00:00
11-29 12:00:00
11-30 12:00:00

Produced by Lloyd Treinish, IBM Watson Research Center



1994 Northridge aftershock, San Fernando valley

http://www.cs.cmu.edu/~quake/quakeviz.html

Animations created by Greg Foss, Pittsburgh Supercomputing Center and CMU Quake project, from synthetic data generated by the CMU Quake project

Inactive VTK control

Load

- Now: Consider candidate data structures for capturing the features of gridded datasets
- Next: Related Work

GridField Interpretations

- "Enhanced Raster Representation"
  - Cheap, intuitive raster arithmetic
  - Ragged edges ok,
  - non-uniform cells
  - data bound to multiple dimensions

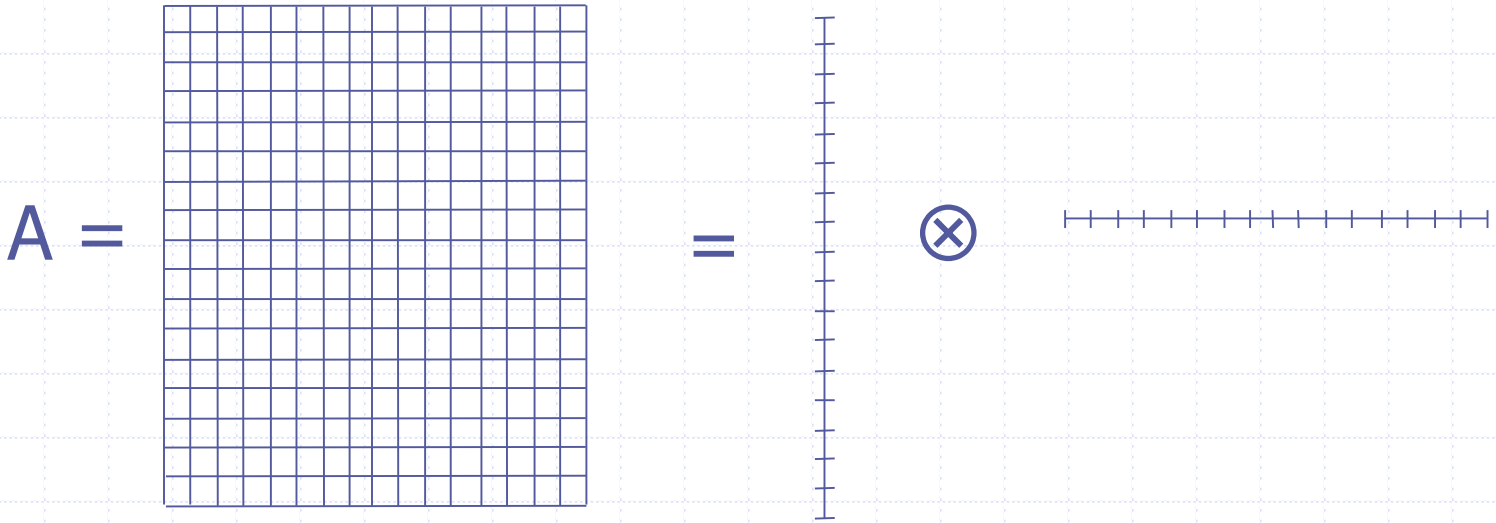
GridField Interpretations

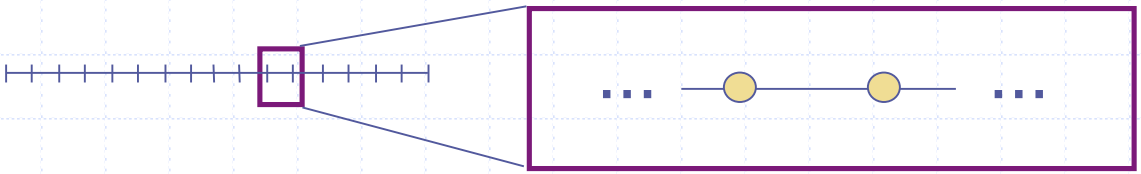
- "Enhanced Vector Representation"
  - Geometric Shape + Bound Tuple vs. Geometric Shape + Bound Dataset
  - Transform these shapes in sophisticated, non-geometric ways
- "Efficient/Indexed Collections of Shapes"
  - Non-redundant geometric coordinates
  - column-oriented representations

Consider the VTK Library

- Apply a function to a dataset?
  - vtkProgrammableAttributeFilter, vtkFunctionParser, others
- Aggregate over a region?
  - no generalized aggregation filter
- Construct multidimensional datasets?
  - grid construction is “manual”

Constructing GridFields (1)

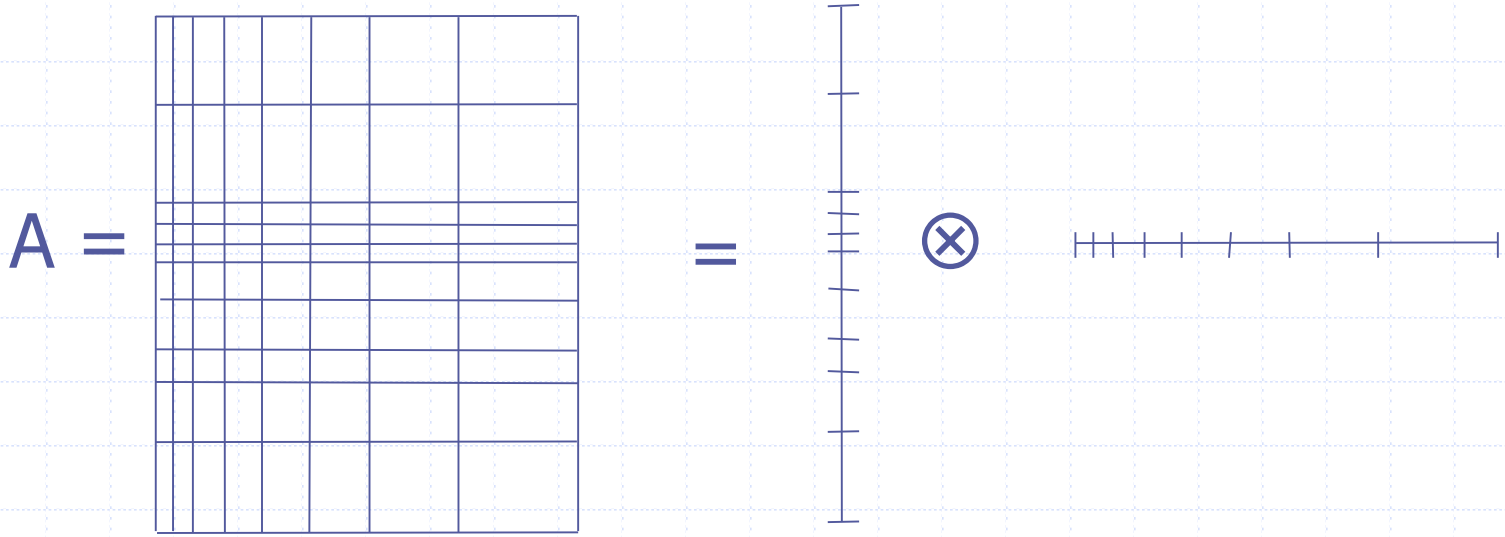


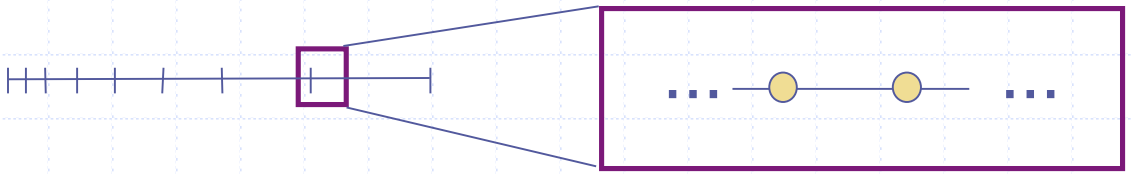


$x_i = \Delta x * i$

x_i is proportional to position i

Constructing GridFields (2)

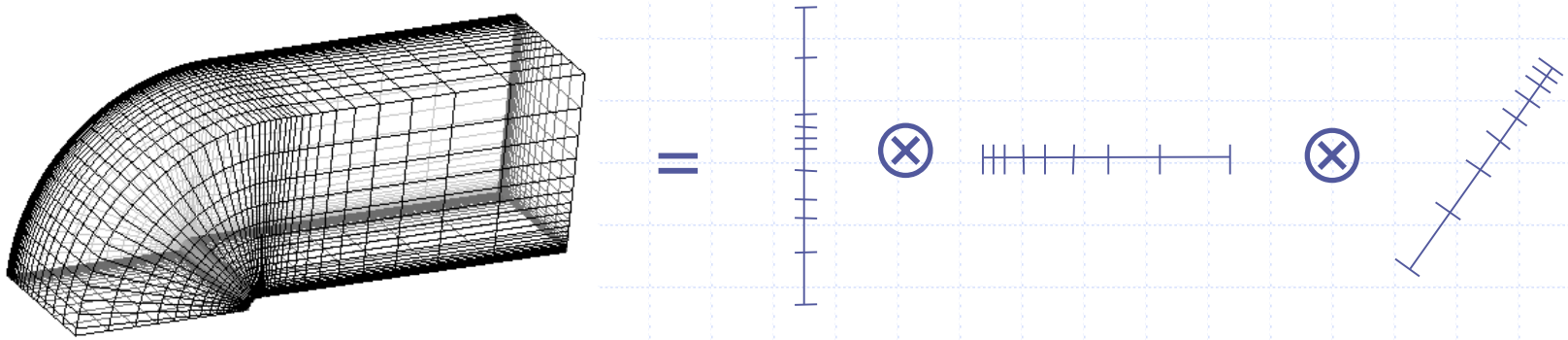




x_i is explicit

x_i is an arbitrary function of position i

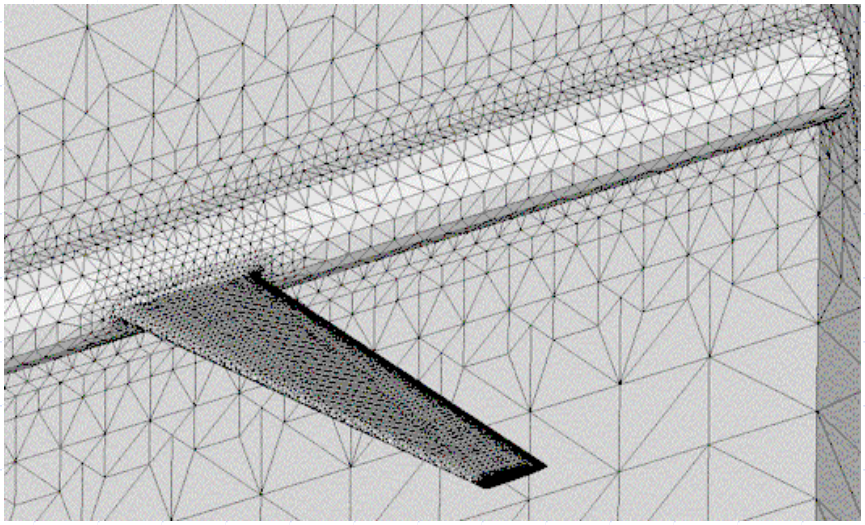
Constructing GridFields (3)



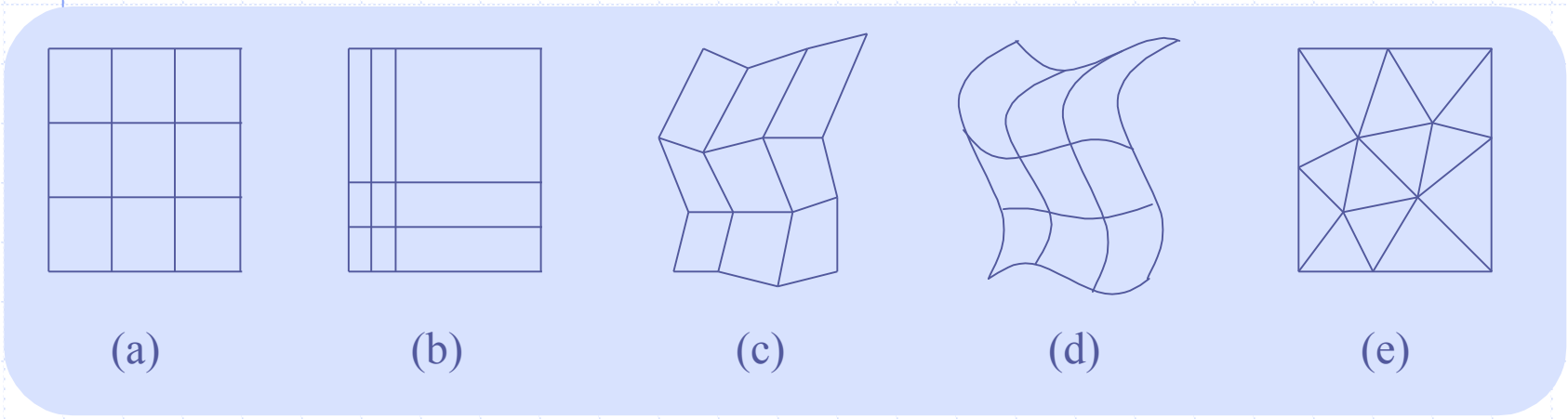
x_i is a function of positions i, j, k

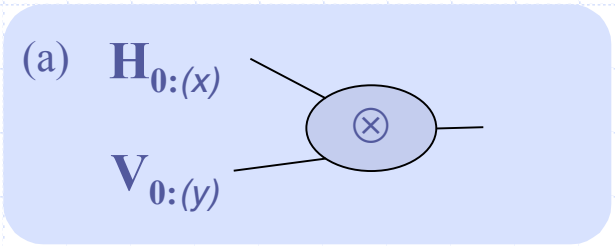
...as are y_i, z_i

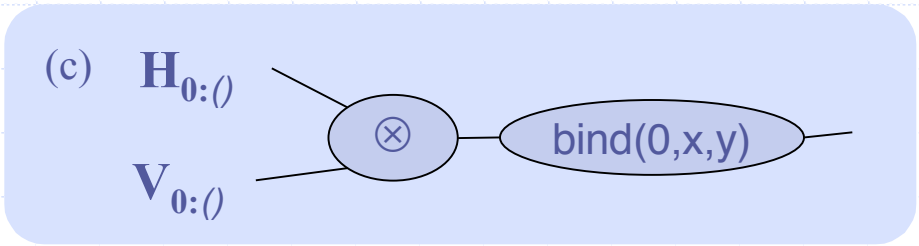
Non-Product GridFields (4)

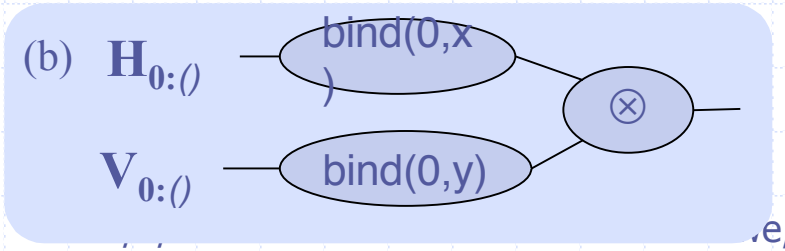


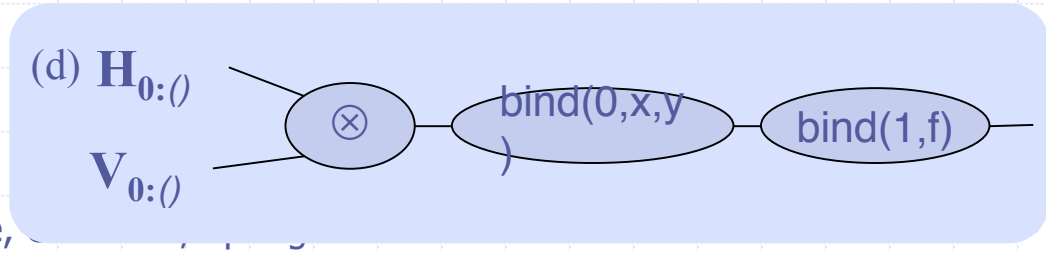
Constructing GridFields





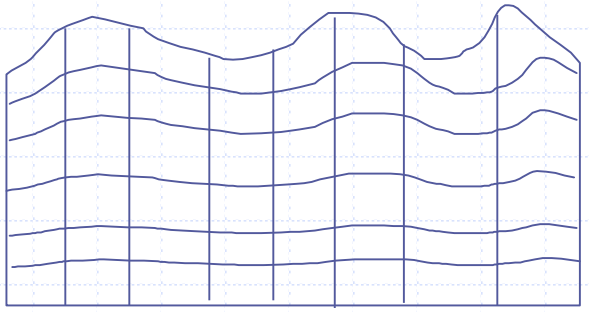




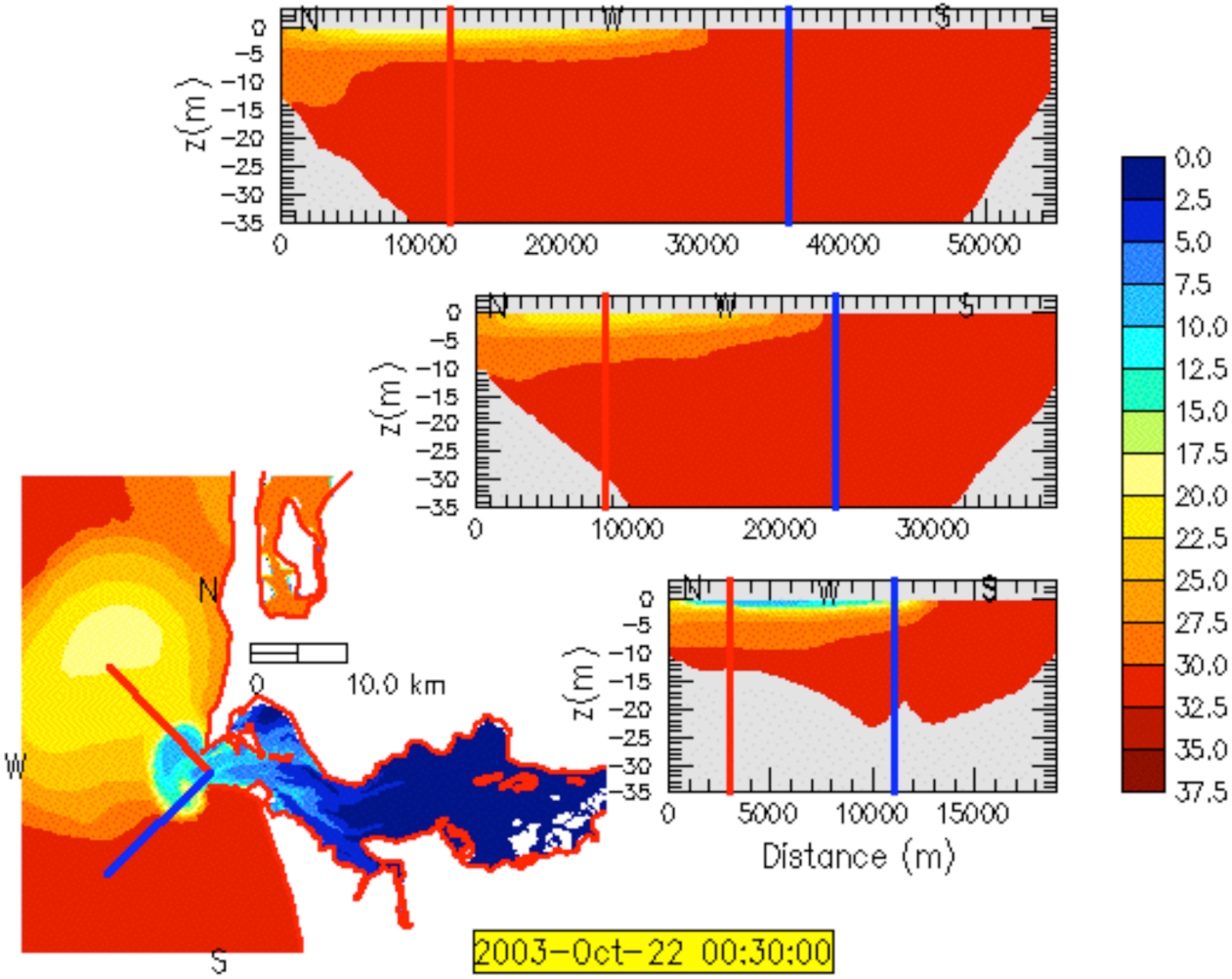


Topology Matters

Sigma grid: geometry changes with time



Topology doesn't change from timestep to timestep



Salinity, Columbia River, CORIE

Consider the VTK Library

- Apply a function to a dataset?
  - vtkProgrammableAttributeFilter, vtkFunctionParser, others
- Aggregate over a region?
  - no generalized aggregation filter
- Construct multidimensional datasets?
  - grid construction is “manual”

Overview: Finale

- Model
  - Todo: Regrid refinement, Iterate Op
- Implementation and Representation
  - Todo: Assemble the AIFL tower, derive a cost function
- Evaluation
  - Todo: Experiments in other domains, more extensive experiments in the primary domain

Native Representations

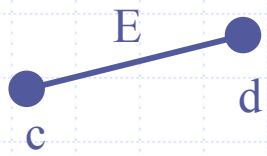
- Goal:
  - Derive a cost function for our query plans
  - But we must accommodate different representations
- Claim: Operators and assn/aggr functions can be defined using only 4 logical access methods
  - So: Implement these for a variety of representations

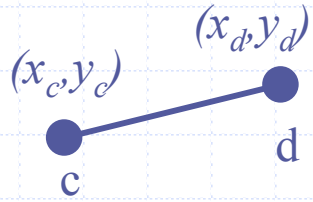
Timeline

- Regrid refinement: 2 weeks
- Iterate operator: 3 weeks
- Assemble the AIFL tower: 5 weeks
- Derive and test a cost function: 2 weeks
- Simple optimizer: 3 weeks
- Experiments, other domains: 5 weeks
- Experiments, primary domain: 3 weeks
- Writing: ongoing, plus 5 weeks

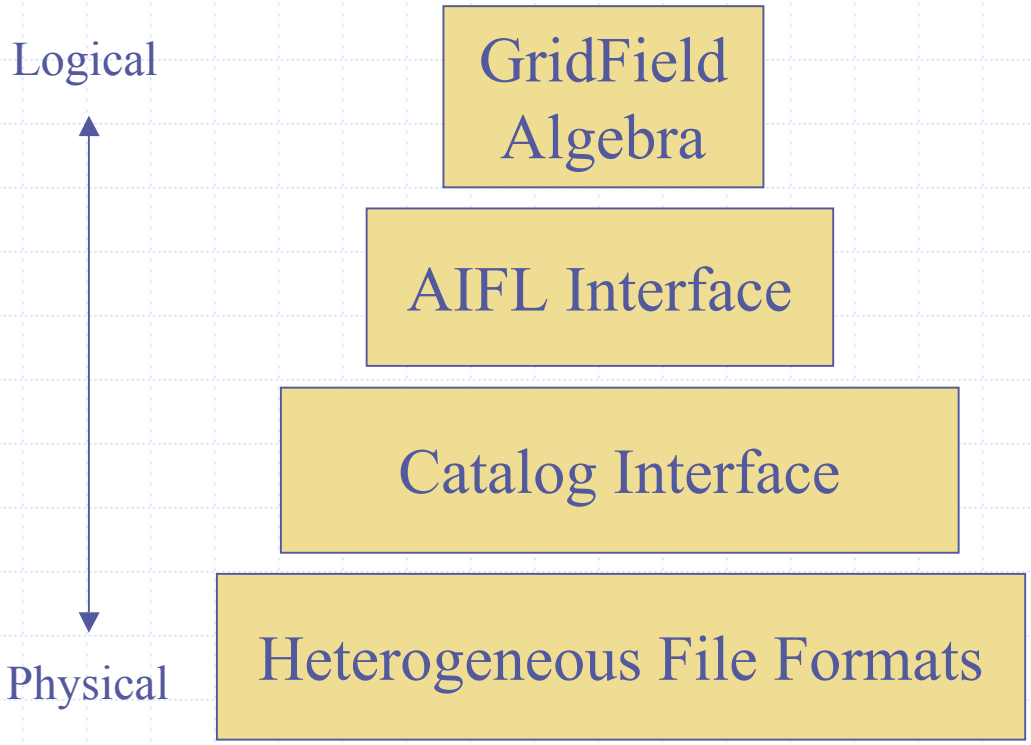
AIFL Interface

- Adjacency: A_i *(optional)*
  - for each cell *c* of dimension *i*, return cells of dimension *i* adjacent to *c*
- Incidence: I_{ik}
  - for each cell *c* of dimension *i*, return cells of dimension *k* incident to *c*
- Field data: F_i
  - for each cell *c* of dimension *i*, return the data bound to *c*
- Lookup: L_i
  - return all the cells of dimension *i* that have a particular data value bound





Accessing Native Data



AIFL Tower?

Maybe we don't want to implement access methods for every possible representation

Describing Native File Formats

```
ragged : {
  header : 10c
  n : i
  r_sizes : n * {
    r_size : i
  }
  outer : n * {
    inner : r_size * {
      v:f u:f
    }
  }
}
```

```
"RAGGED_ARR"
4
1, 3, 1, 2
(0.6,0.4)
(0.2,0.9),(1,3),(1.7,2.6)
(2.1,9.4)
(3.8,8.9),(4.2,1.6)
```

```
5241 4747 4544 5f41 5252
0000 0400 0000 0100 0000
0300 0000 0100 0000 0200
0000 9a99 193f cdcc cc3e
cdcc 4c3e 6666 663f 9a99
993f cdcc 4c40 9a99 d93f
6666 2640 6666 0640 6666
1641 3333 7340 6666 0e41
6666 8640 cdcc cc3f
```

- Portions of the file implement AIFL methods
  - r_size is an attribute of a gridfield (F_i)
  - v and u are attributes of another gridfield
- Catalog tells us which portions should be accessed to evaluate AIFL methods for a particular gridfield

** *Howe, Maier, Retrofitting a Data Model to an Existing Environmental Repository, SSDBM 2005*

Overview

- Model
  - Todo: Regrid refinement, Iterate Op
- Implementation and Representation
  - Todo: Assemble the AIFL tower, derive a cost function
- Evaluation

Evaluation

- Show: GridFields are competitive with relational and visualization-based solutions
- Show: Cost function is accurate
- Show: Expressiveness
  - Apply techniques to datasets in multiple domains:
    - Oceanography
    - Seismic Modeling
    - Mechanical Engineering
    - Water Treatment

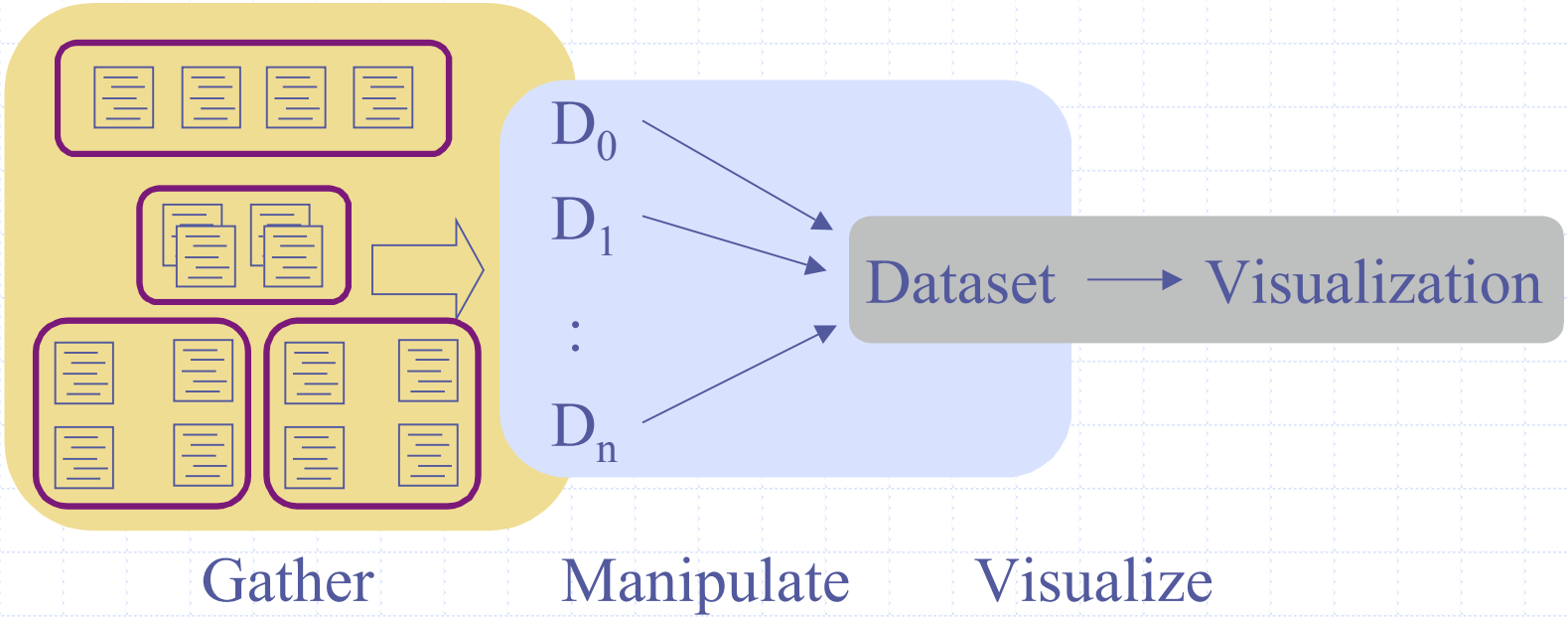
Overview

- Model
  - Todo: Regrid refinement, Iterate Op
- Implementation and Representation
- Evaluation

Regrid: ToDo

- Some difficulties:
  - Regrid is difficult to optimize
  - assn/aggr functions are difficult to write
  - Some transformations are difficult to express
- Todo: Refine the language for writing assn/aggr
  - *more on this in a bit*

Data*base* of Data*sets*



A *catalog* permits access to *logical datasets*, regardless of size, location, or manner of decomposition into files

Relational Databases?

- *Claim: An RDBMS is useful for implementing the catalog, but is inappropriate for storing the datasets themselves*

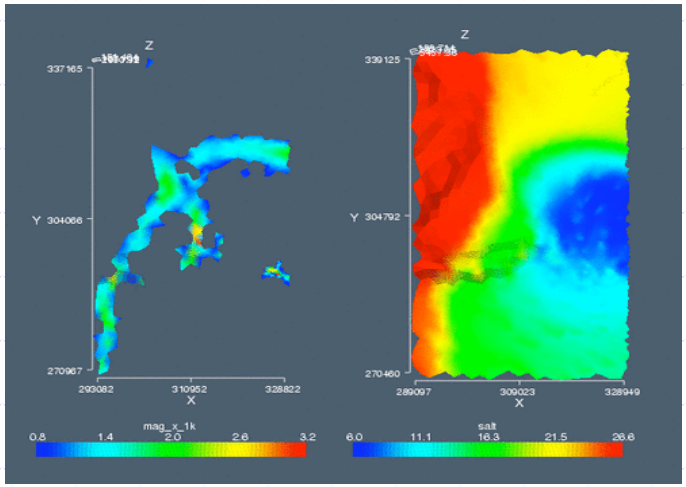
- Simulation data is primarily read-only, RDBMS optimized for transaction processing
- RDBMS requires more space for the same data
- Impedance mismatch between RDBMS and visualization tools
- Requires scientists to relinquish control of their data's representation

Visualization Tools?

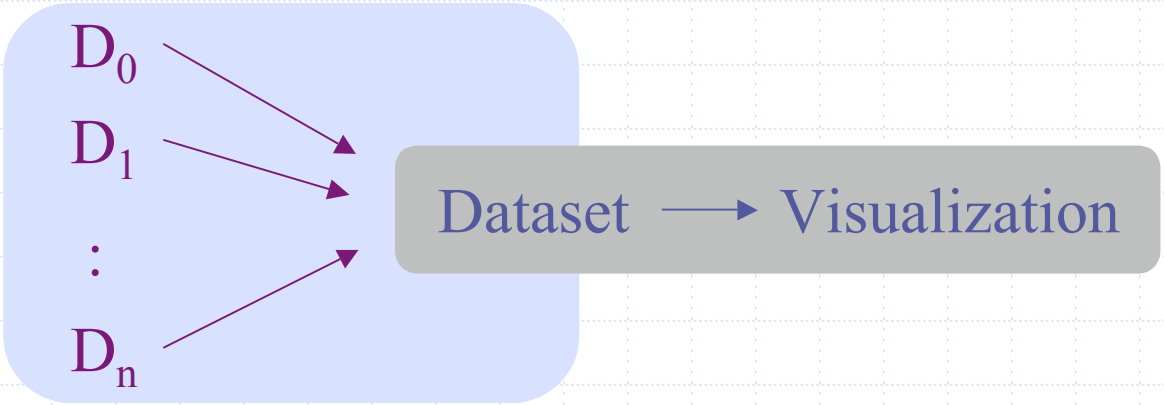
A

B

Inactive VTK control

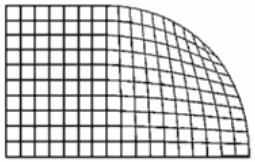


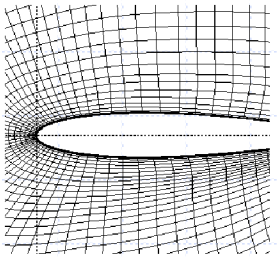
Local File ⟶ Visualization

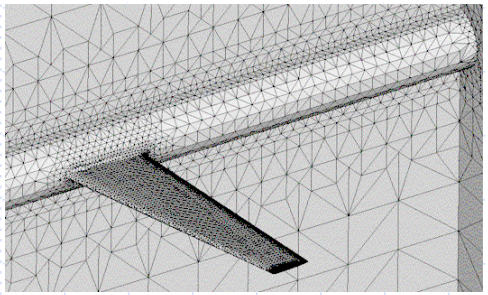


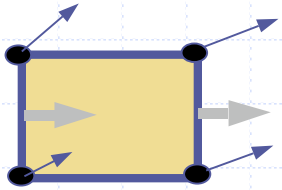
We seek a solution that is

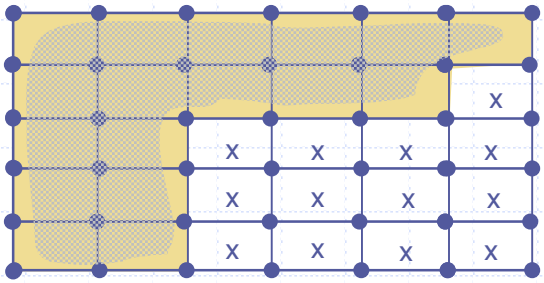
- Comprehensive
- Dimension-agnostic
- Efficient











Dissertation TOC (1)

- Simulation in the Physical Sciences
  - Continuous Physical Systems
  - Domains
  - Scaling up
  - Integration

Dissertation TOC (2)

- Model
  - Grid
  - GridField
  - Operators
  - Properties

Dissertation TOC (3)

- Implementation
  - Operator Algorithms
  - Regrid
  - Iterate
  - A Canonical GridField Application

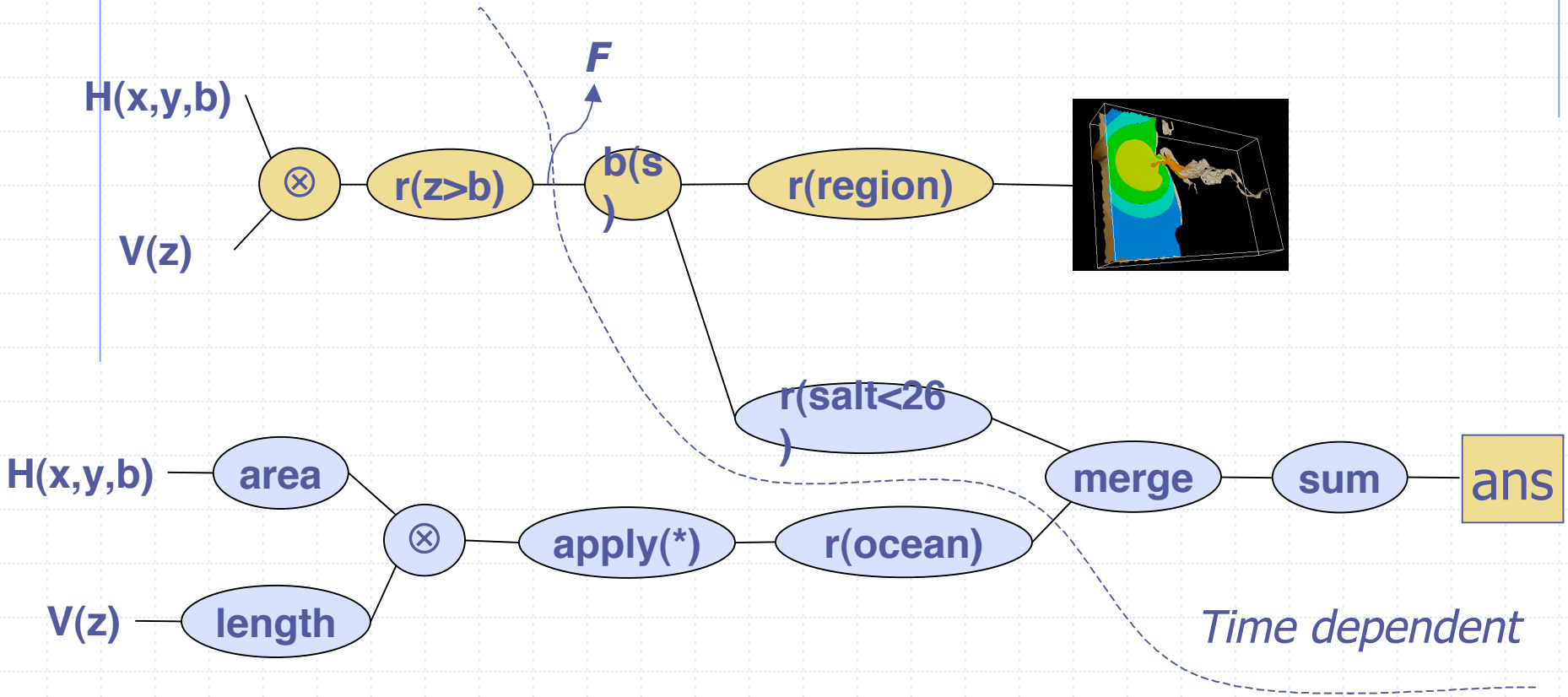
Dissertation TOC (4)

- Working with Native Data Sources
  - Describing Native Data Sources
  - Accessing Native Data Sources

Dissertation TOC (5)

- Evaluation
  - Performance
  - Evaluating Algebraic Optimization
  - Evaluating Native Access
  - Expressiveness
    - CORIE
    - Water Quality
    - QUAKE

Compare Plan...

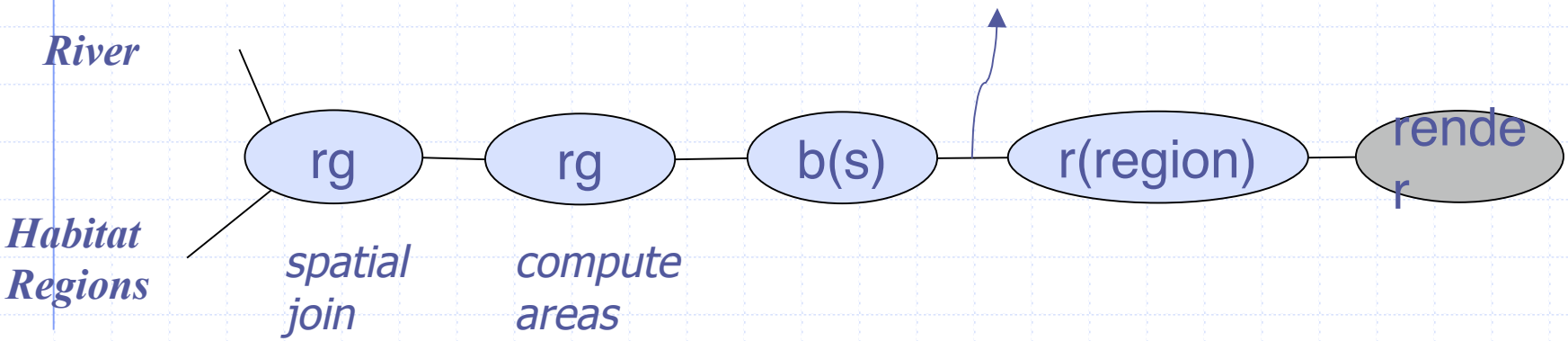


with Code

```
for(l=0;l<file_head.nsteps;l++){
  byte_length2ds = (long) (header_skip +
          (long)l*(sizeof(float)*(1+file_head.grid.nodes)+sizeof(int)*(1+file_head.grid.nodes)));
  byte_length3ds = (long) (header_skip + (long) l*(sizeof(float)*(1+Nvalues)+sizeof(int)*(1+file_head.grid.nodes)) +
          (long)(sizeof(float)*1+ sizeof(int)*(1+file_head.grid.nodes)));

  fseek(f63,byte_length3ds,0);
  fseek(felv,byte_length2ds,0);

  if (fread(&ttime, sizeof(float), 1, felv) != 1) {
    break;
  }
  if (fread(&time_step, sizeof(int), 1, felv) != 1) {
    break;
  }
  fprintf(stderr,"time = %f time step = %d\n",ttime,time_step);
  ttime = ttime/86400.0+corie_start_time;
  if (fread(file_head.grid.top_layer,sizeof(int), file_head.grid.nodes,felv) != file_head.grid.nodes){
    break;
  }
  if (fread(elev,sizeof(float),file_head.grid.nodes,felv) != file_head.grid.nodes){
    break;
  }
  if (fread(var63, sizeof(float), Nvalues, f63) != Nvalues) {
    break;
  }

  plume_volume = 0.0;
  plume_area = 0.0;
```

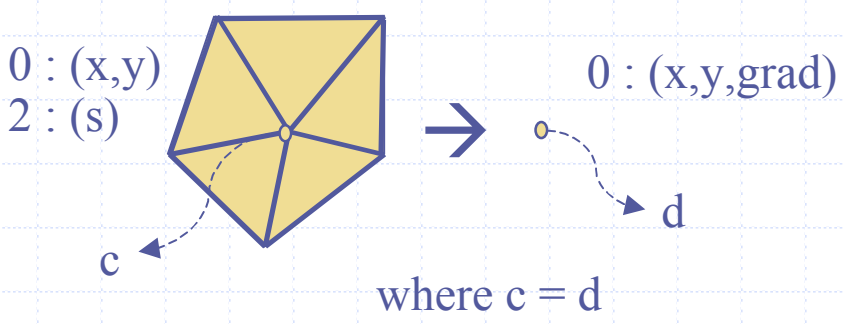
```
  for (j=0;j<file_head.grid.elements;j++){
    n1 = file_head.grid.nm[j*3]-1;
    n2 = file_head.grid.nm[j*3+1]-1;
    n3 = file_head.grid.nm[j*3+2]-1;
    if (flag[n1] == 1 && flag[n2] == 1 && flag[n3] == 1){
      av_depth = (file_head.grid.depth[n1]+file_head.grid.depth[n2]+file_head.grid.depth[n3])/3.0;
      av_elev = (elev[n1]+elev[n2]+elev[n3])/3.0;

      check_flag = 0;
      for (k=0;k<file_head.layers.nlevels;k++) {
       s1 = s2 = s3 = -99.0;
       wt1 = wt2 = wt3 = 0.0;
       count = 0;
       if ((k >= (file_head.grid.bottom_layer[n1]-1)) && (k <= (file_head.grid.top_layer[n1]-1))){
        s1 = var63[start_loc_scalar[n1]+k-file_head.grid.bottom_layer[n1]+1];
        count++;
        wt1 = 1.0;
       }
       if ((k >= (file_head.grid.bottom_layer[n2]-1)) && (k <= (file_head.grid.top_layer[n2]-1))){
        s2 = var63[start_loc_scalar[n2]+k-file_head.grid.bottom_layer[n2]+1];
        count++;
        wt2 = 1.0;
       }
       if ((k >= (file_head.grid.bottom_layer[n3]-1)) && (k <= (file_head.grid.top_layer[n3]-1))){
        s3 = var63[start_loc_scalar[n3]+k-file_head.grid.bottom_layer[n3]+1];
        count++;
        wt3 = 1.0;
       }
       if (count > 0.0){
        av_sal = (s1*wt1+s2*wt2+s3*wt3)/count;
        if (av_sal < cutoff){
          check_flag = 1;
          if (k == 0){ z1 = -file_head.layers.zmsl; }
          else if(-av_depth > zpos[k-1]){ z1 = -av_depth;  }
          else{ z1 = zpos[k-1]; }
          if (av_elev < zpos[k]){z2 = av_elev;  }
          else{ z2 = zpos[k];  }
          plume_volume += (z2-z1)*el_areas[j];
        }
       }
      }
      if (check_flag == 1){  plume_area += el_areas[j];}
    }
  }
```

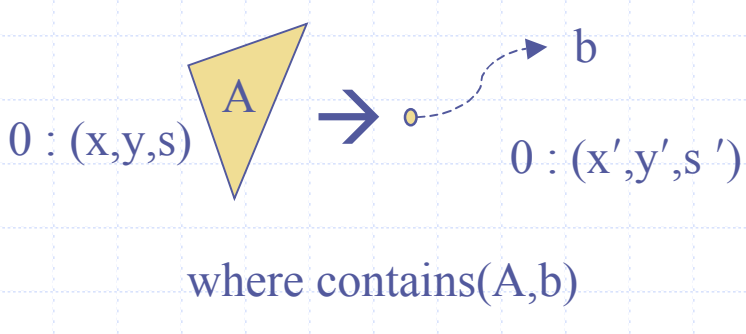
Abstract, Executable Specifications



Regrid: ToDo

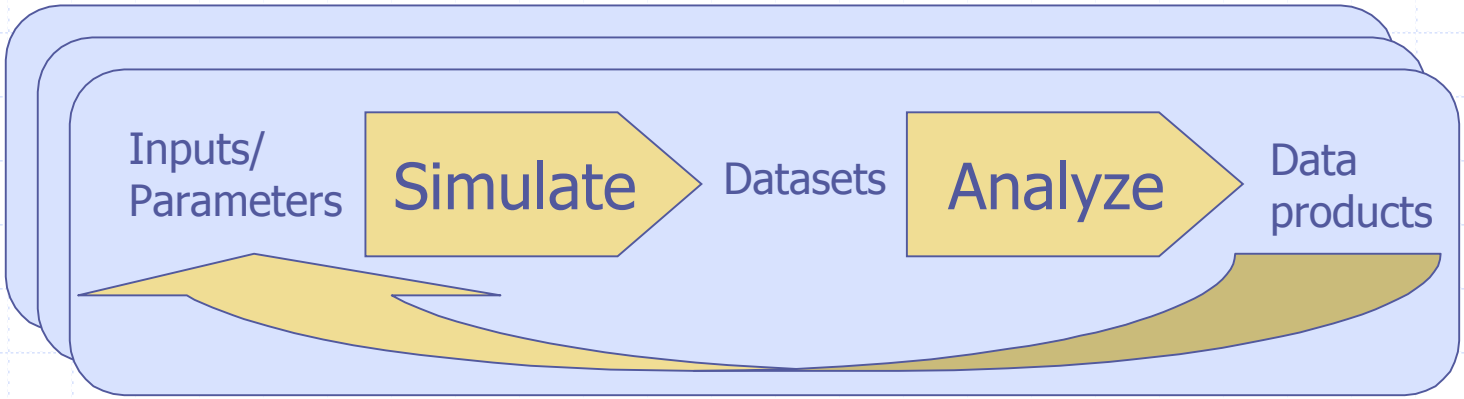
- Some difficulties:
  - Regrid is difficult to optimize
  - assn/aggr functions are difficult to write
  - Some transformations are difficult to express
- Todo: Refine the language for writing assn/aggr
  - write assn/aggr functions in terms of *stencils*



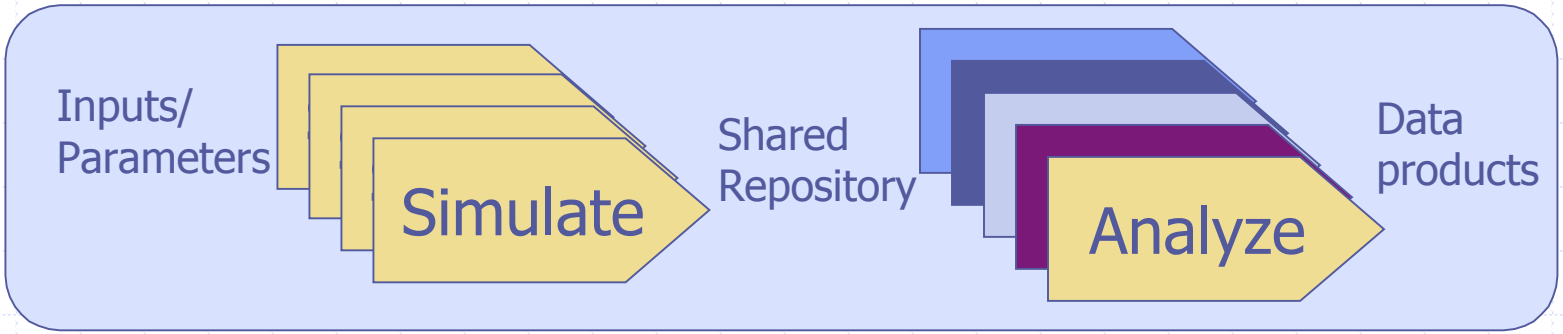


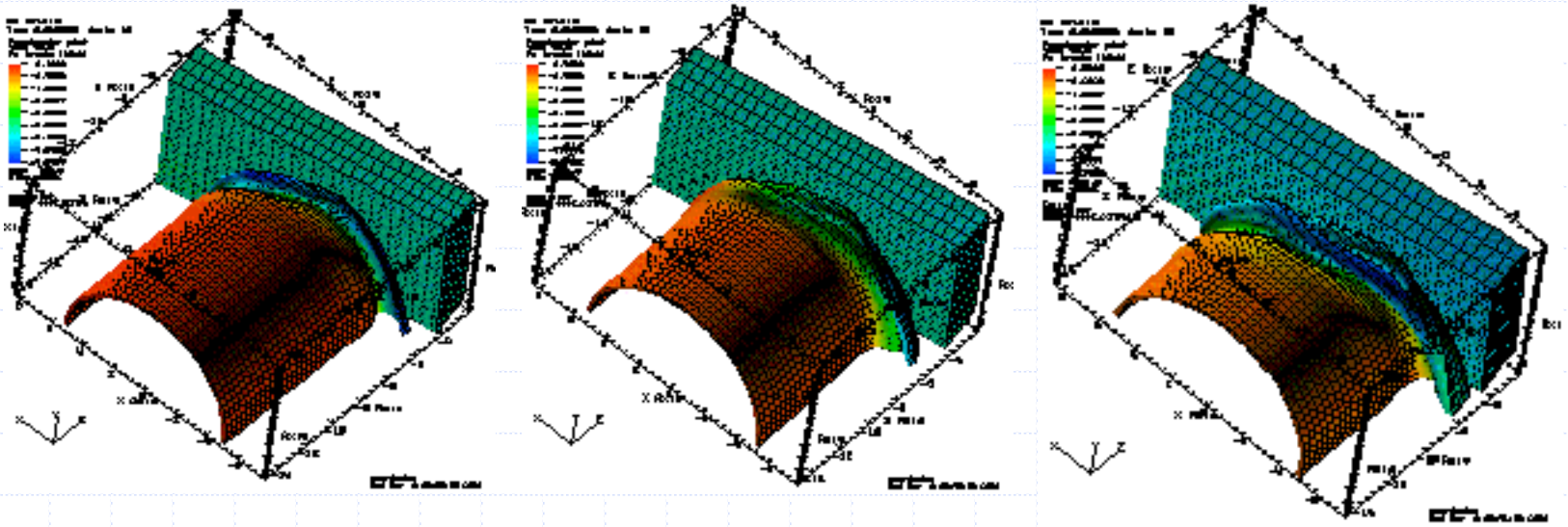
Simulation for Science

demand simulations



supply simulations

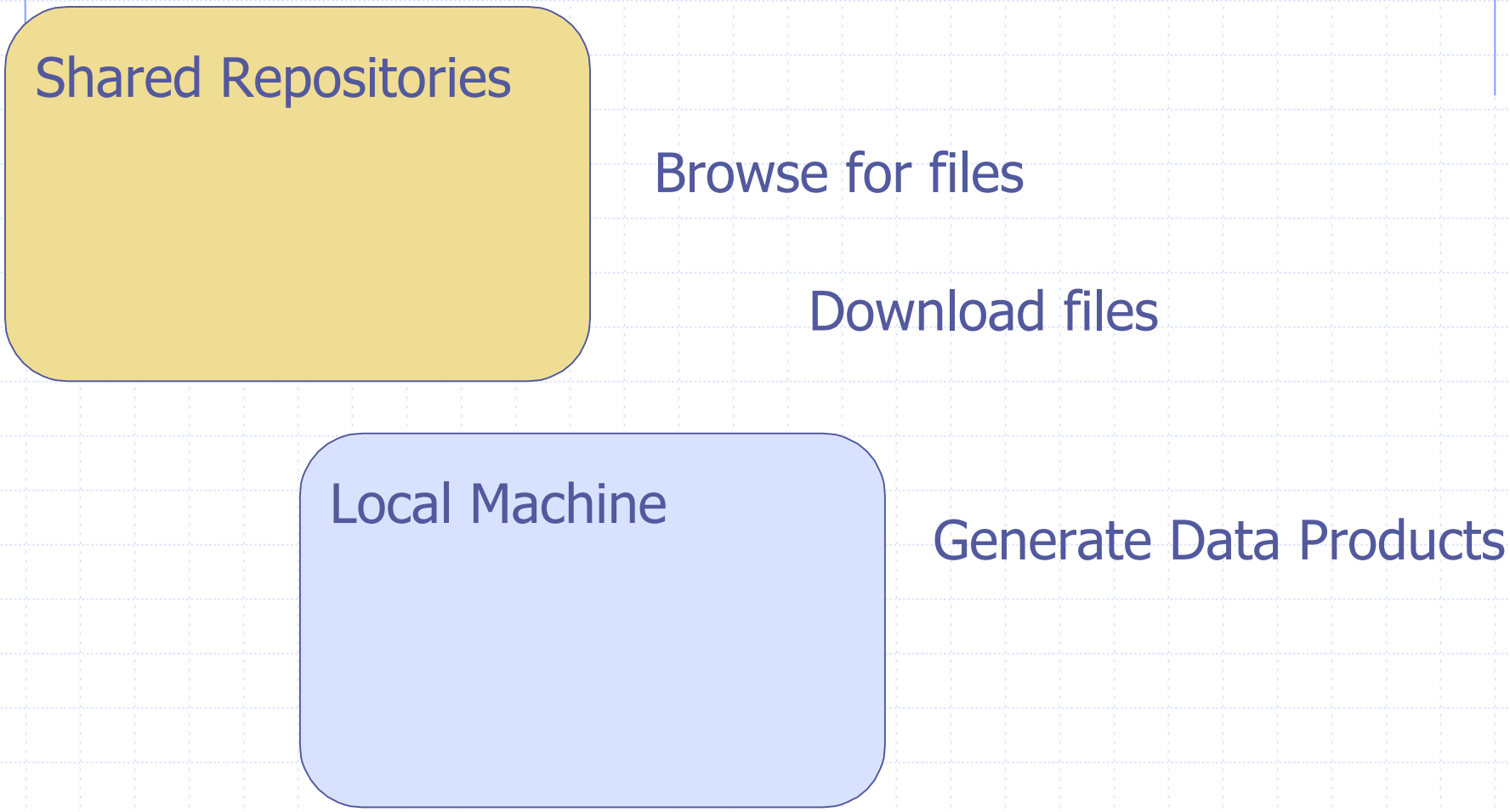




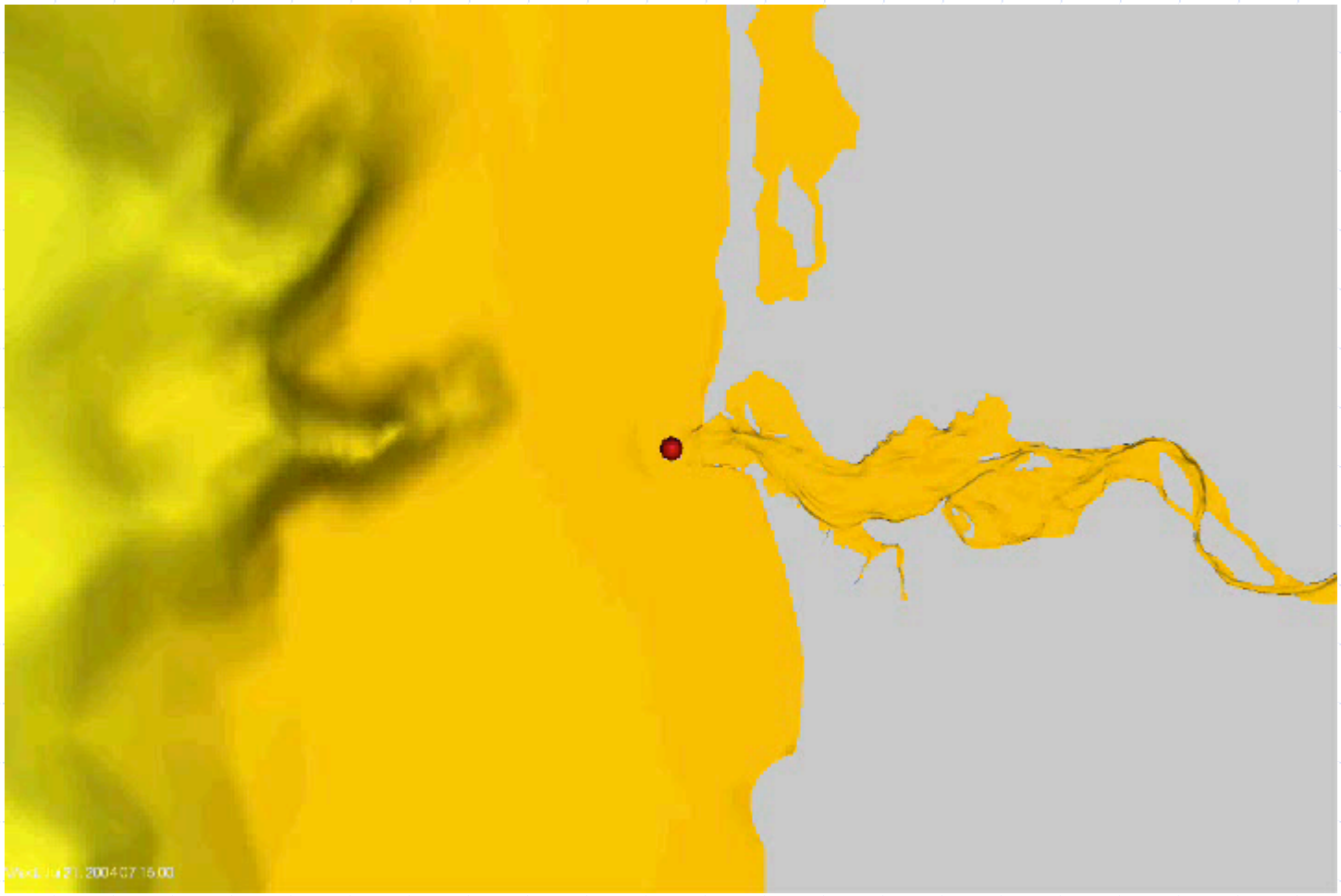
Modeling and Querying Scientific Simulation Mesh Data

B. Lee, R. Snapp, L. Chen (U. Vermont), and I. Song (Drexel U.)

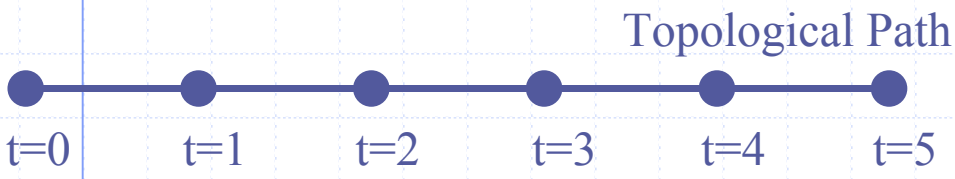
http://www.bnl.gov/epscor/files/doc/abstracts/lee.doc


Shared Repositories
Browse for files
Download files
Local Machine
Generate Data Products

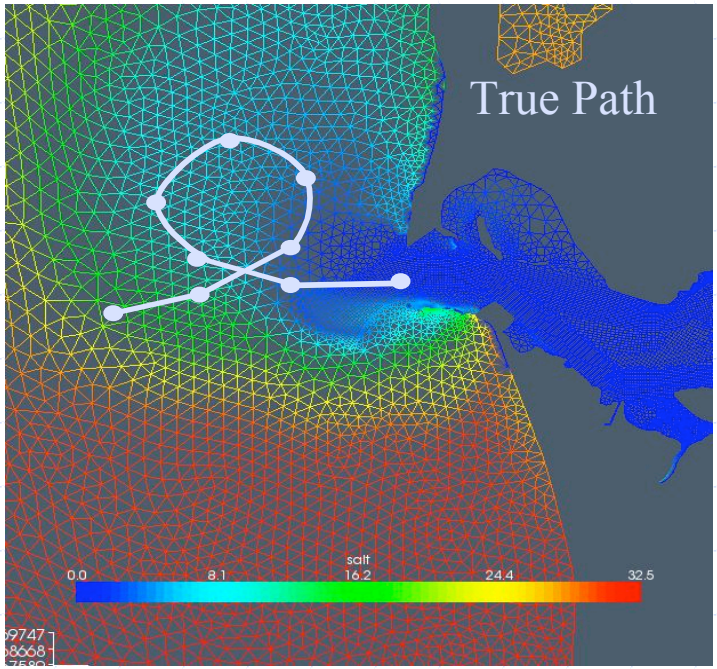
Operator: Iterate



Iterate



Target grid, T

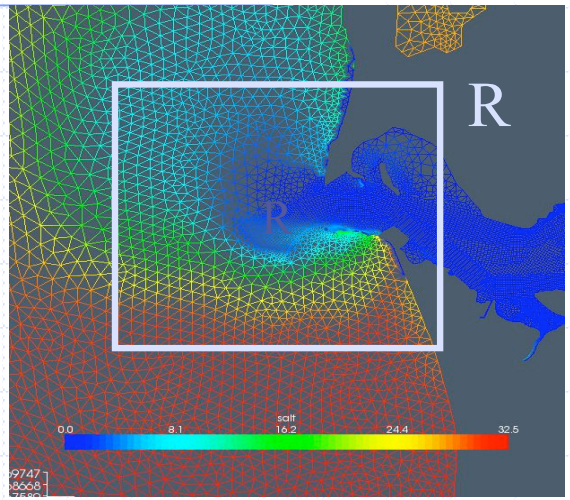


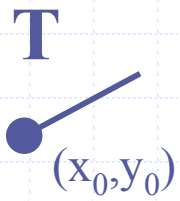
Source grid, S

We can't express this operation as a regrid

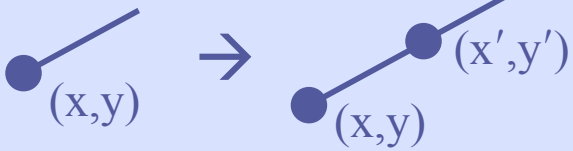
- The position at t = n depends on positions at t < n, and velocity flow data from the source grid.

Iterate





Rule 1:



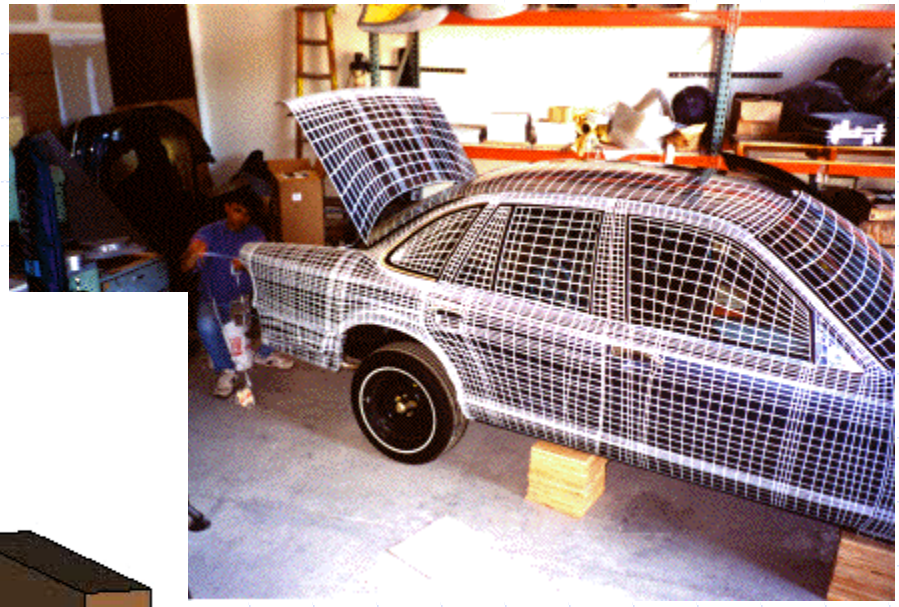
if (x,y) in R

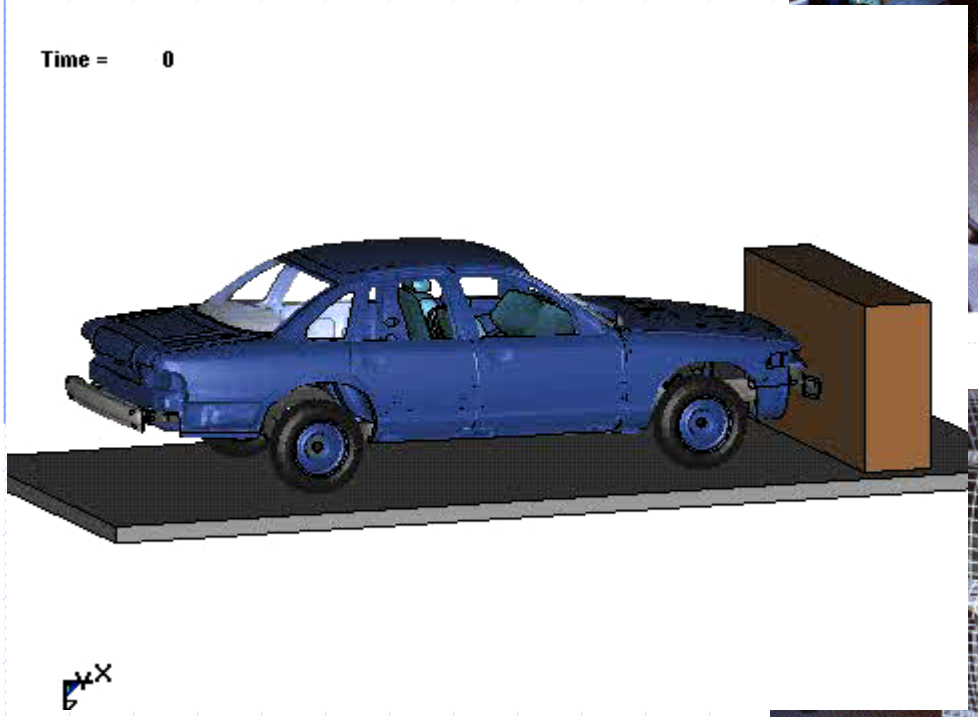
where (x′,y′) is a function of (x,y), and the nearby velocity data in S

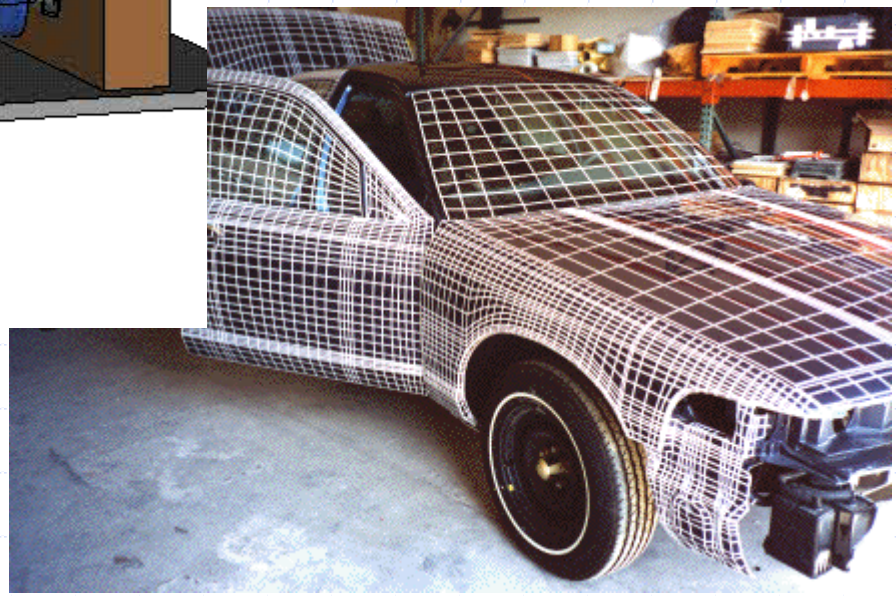
Rule 2:

if (x,y) not in R

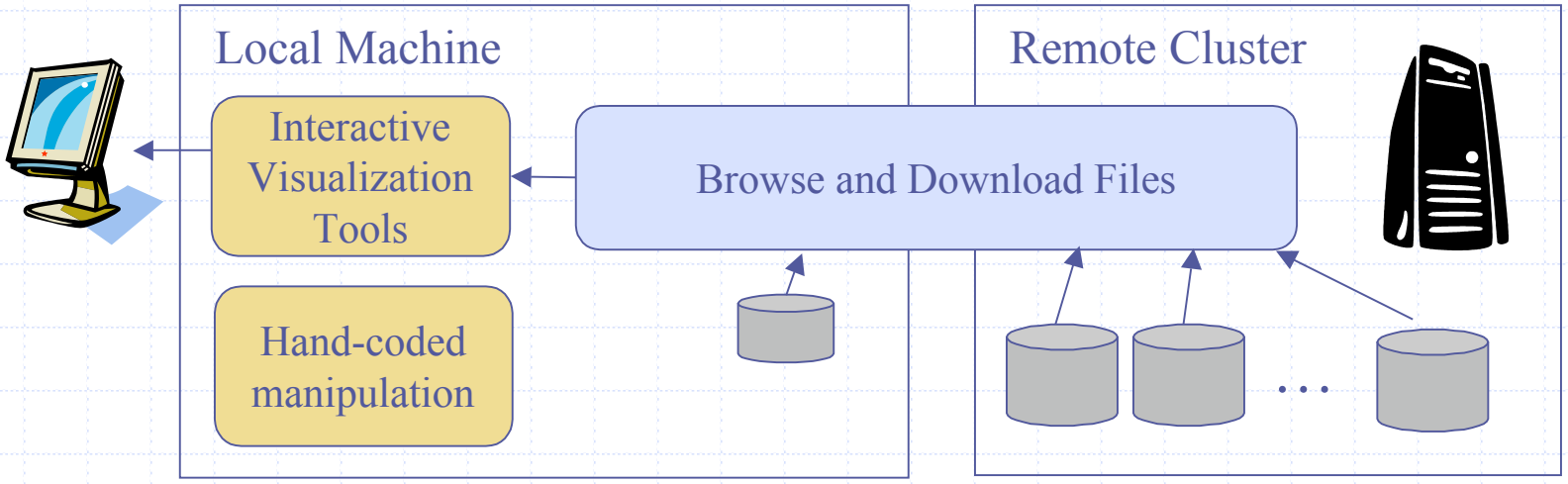
Iterate finds a fix point of these rules




Time = 0



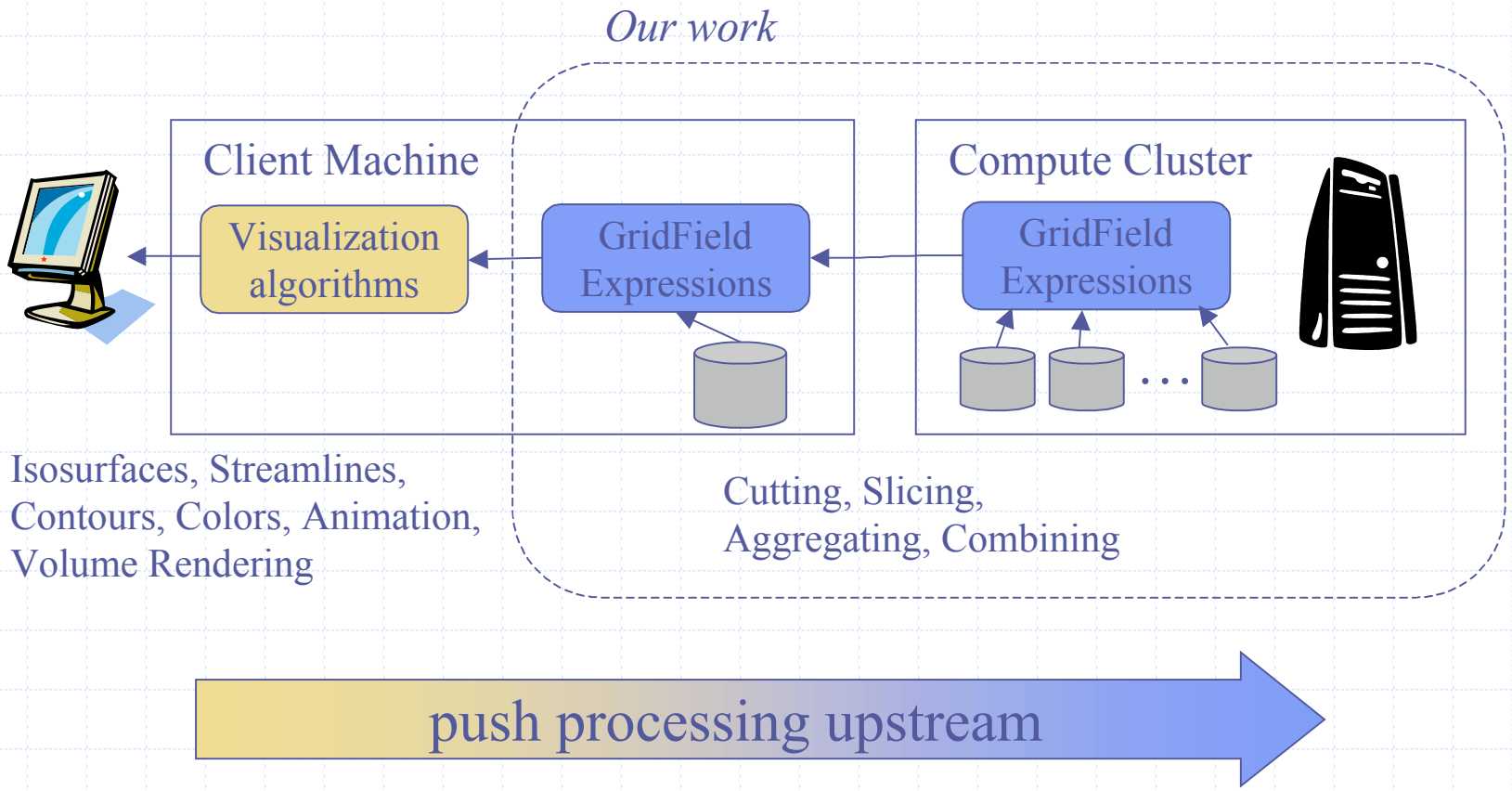
Separating Data Manipulation from Data Visualization



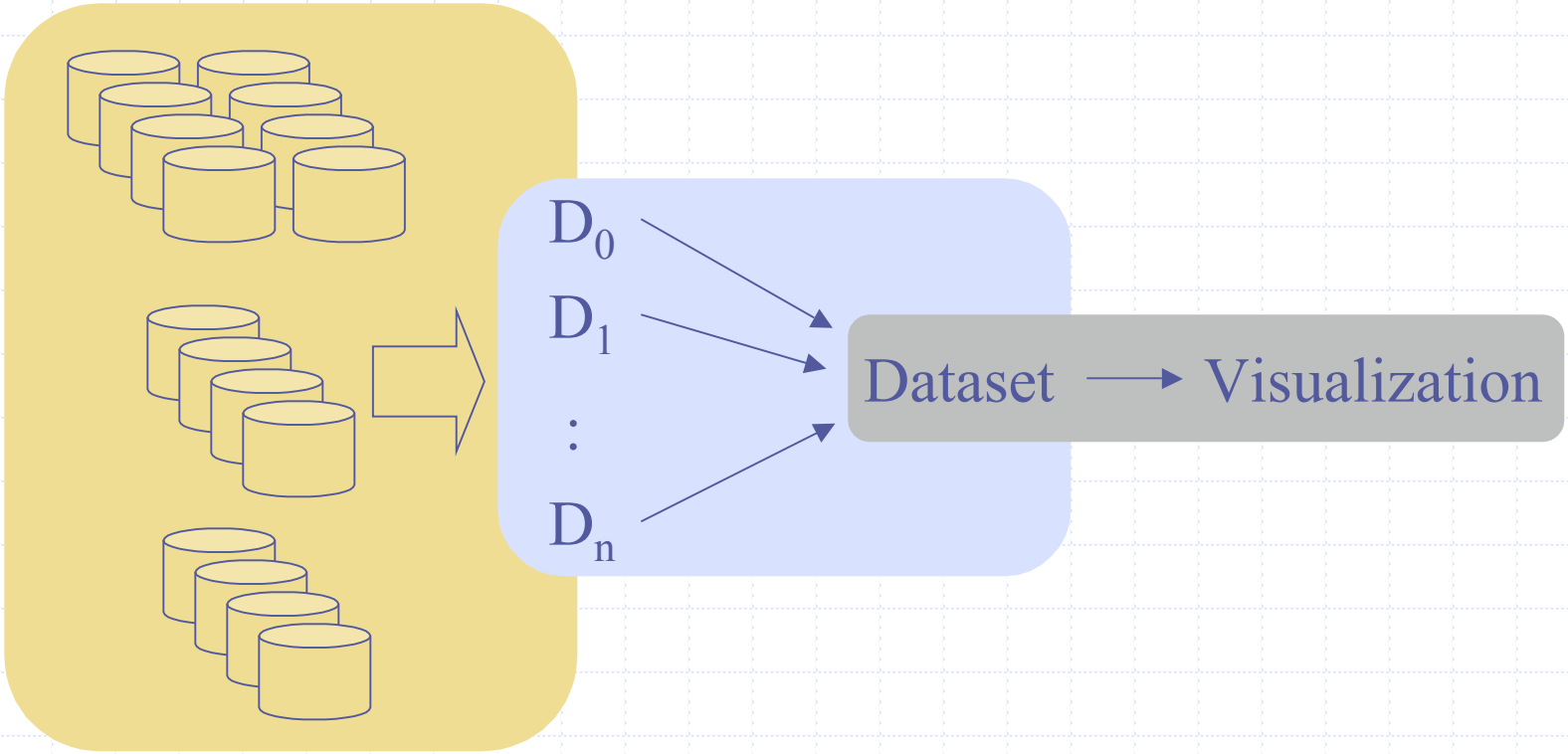
Cutting, Slicing,
Aggregation, Combination

Isosurfaces, Streamlines,
Contours, Colors, Animation,
Volume Rendering

Separating Data Manipulation from Data Visualization



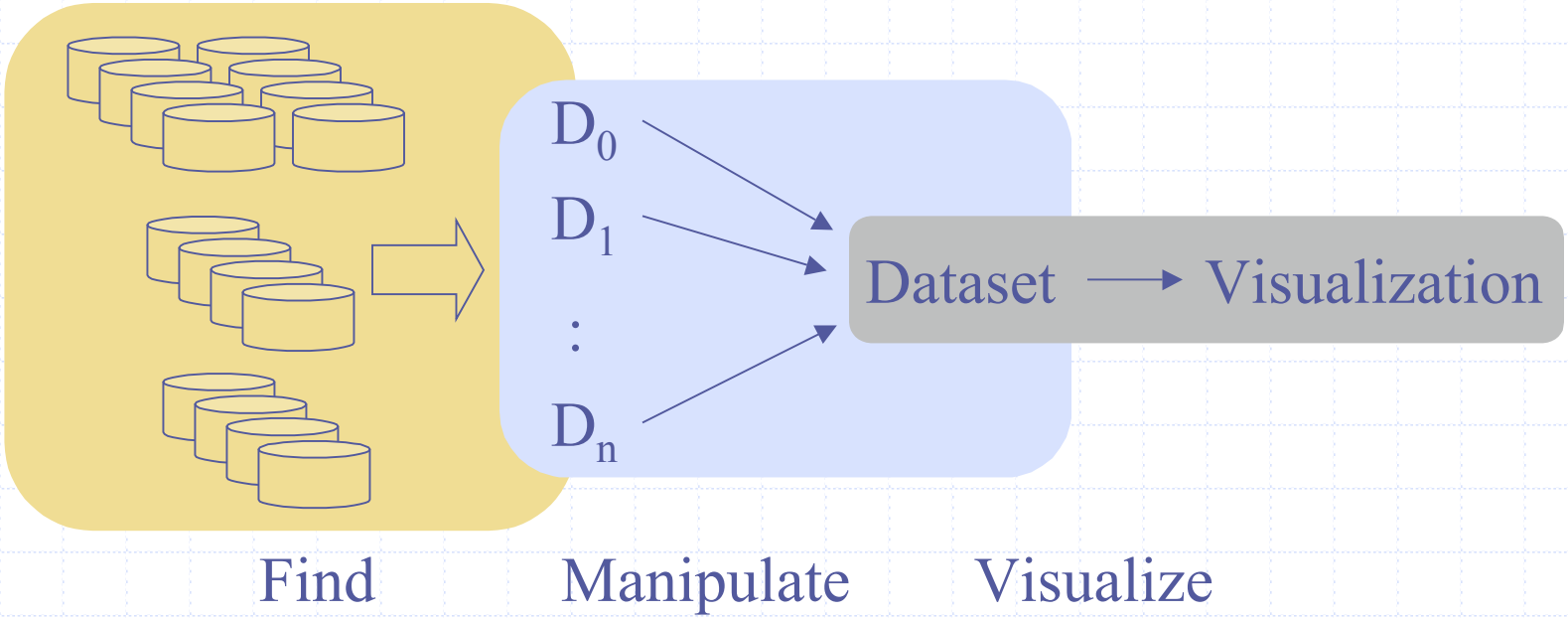
Data*base* of Data*sets*



Repository *Working set* *Result* *Visualization*

Find Manipulate Visualize

Data*base* of Data*sets*



Allow access to *logical datasets*, regardless of how they are decomposed into files

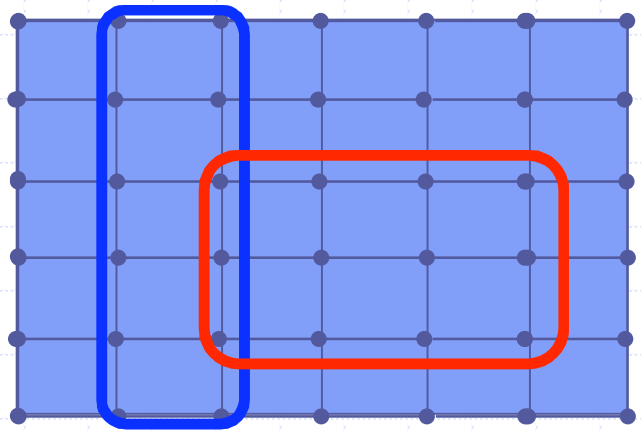
Logical datasets found using a *catalog*

Dissertation TOC

- Motivation
  - Simulating Continuous Physical Systems
  - Domains
  - Scaling up
  - Integration
- Model
  - Grid
  - GridField
  - Operators
  - Properties
- Implementation
  - Operator Algorithms

netCDF Operations

- Read/Write metadata
- Read/Write whole datasets
- Read/Write elements
- Read/Write hyperslabs



Related Work