Fields, Meshes, and Interpolation (Part 3)
Outline

• Review
• Interpolation
• Cell Location
• Project 2
Rectilinear meshes

- Rectilinear meshes are easy and compact to specify:
  - Locations of X positions
  - Locations of Y positions
  - 3D: locations of Z positions
- Then: mesh vertices are at the cross product.
- Example:
  - $X=\{0,1,2,3\}$
  - $Y=\{2,3,5,6\}$
Rectilinear meshs aren’t just the easiest to deal with ... they are also very common.
Quiz Time

- A 3D rectilinear mesh has:
  - \( X = \{1, 3, 5, 7, 9\} \)
  - \( Y = \{2, 3, 5, 7, 11, 13, 17\} \)
  - \( Z = \{1, 2, 3, 5, 8, 13, 21, 34, 55\} \)

- How many points? \(= 5 \times 7 \times 9 = 315\)
- How many cells? \(= 4 \times 6 \times 8 = 192\)
Definition: dimensions

- A 3D rectilinear mesh has:
  - X = \{1, 3, 5, 7, 9\}
  - Y = \{2, 3, 5, 7, 11, 13, 17\}
  - Z = \{1, 2, 3, 5, 8, 13, 21, 34, 55\}

- Then its dimensions are 5x7x9
How to Index Points

• Motivation: many algorithms need to iterate over points.

```c
for (int i = 0 ; i < numPoints ; i++)
{
    double *pt = GetPoint(i);
    AnalyzePoint(pt);
}
```
Schemes for indexing points

<table>
<thead>
<tr>
<th>Logical point indices</th>
<th>Point indices</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,5 1,5 2,5 3,5 4,5 5,5</td>
<td>30 31 32 33 34 35</td>
</tr>
<tr>
<td>0,4 1,4 2,4 3,4 4,4 5,4</td>
<td>24 25 26 27 28 29</td>
</tr>
<tr>
<td>0,3 1,3 2,3 3,3 4,3 5,3</td>
<td>18 19 20 21 22 23</td>
</tr>
<tr>
<td>0,2 1,2 2,2 3,2 4,2 5,2</td>
<td>12 13 14 15 16 17</td>
</tr>
<tr>
<td>0,1 1,1 2,1 3,1 4,1 5,1</td>
<td>6 7 8 9 10 11</td>
</tr>
<tr>
<td>0,0 1,0 2,0 3,0 4,0 5,0</td>
<td>0 1 2 3 4 5</td>
</tr>
</tbody>
</table>

What would these indices be good for?
How to Index Points

• Problem description: define a bijective function, $F$, between two sets:
  – Set 1: $\{(i,j,k) : 0 \leq i < nX, 0 \leq j < nY, 0 \leq k < nZ\}$
  – Set 2: $\{0, 1, ..., n\text{Points}-1\}$

• Set 1 is called “logical indices”
• Set 2 is called “point indices”

Bijective: for every element in set 1, there is an element in set 2. And vice-versa.

Note: for the rest of this presentation, we will focus on 2D rectilinear meshes.
And then:

• We defined a bijective function for points
• And looked at its code
• And then did it again for cells
Outline

• Review
• Interpolation
• Cell Location
• Project 2
Linear Interpolation for Scalar Field $F$
Linear Interpolation for Scalar Field $F$

- General equation to interpolate:
  - $F(X) = F(A) + t*(F(B)-F(A))$
- $t$ is proportion of $X$ between $A$ and $B$
  - $t = (X-A)/(B-A)$
Quiz Time #4

• F(3) = 5, F(6) = 11
• What is F(4)? = 5 + (4-3)/(6-3)*(11-5) = 7

• General equation to interpolate:
  – F(X) = F(A) + t*(F(B)-F(A))
• t is proportion of X between A and B
  – t = (X-A)/(B-A)
Bilinear interpolation for Scalar Field F

\[ F(0,0) = 1 \quad F(0,1) = 1 \quad F(1,0) = 5 \quad F(1,1) = 6 \]

What is value of \( F(0.3, 0.4) \)?

\[ F(0.3, 0) = 8.5 \quad F(0.3, 1) = 2.5 \]

\[ = 6.1 \]

Idea: we know how to interpolate along lines. Let’s keep doing that and work our way to the middle.

• General equation to interpolate:
  \[ F(X) = F(A) + t*(F(B)-F(A)) \]
Interpolation for triangle meshes

• Two issues:
  – (1) how to locate triangle that contains point
    • (discuss later)
  – (2) how to interpolate to value within triangle
Idea #1

- More bilinear interpolation
Idea #1 (cont’d)

• Different triangle, similar idea...
Idea #2: Barycentric Coordinates

\[ V(P) = \frac{V(A_1)t_1 + V(A_2)t_2 + V(A_3)t_3}{t_1 + t_2 + t_3} \]
Outline

• Review
• Interpolation
• Cell Location
• Project 2
Cell location

- Problem definition: you have a physical location (P). You want to identify which cell contains P.

It is easy to identify the cell that contains P with our eyes. But more work to instruct a computer to do it. What are your thoughts?
Cell location idea #1 (bad)

- Iterate over every cell
- Check if each cell contains P

- Setup time: zero
- Search time: $O(N)$, where $N$ is the number of cells
- Search time for M queries: $O(M*N)$
Cell location idea #2 (good)

- Build “quadtree” data structure
  - (see next slide)
- Takes time to build, but then search is cheap

- Setup time: \(O(N \log N)\), where \(N\) is the number of cells
- Search time: \(O(\log N)\)
- Search time for \(M\) queries:
  - \(O(M \log N) + O(N \log N)\)
Cell location idea #2 (good)
Comparing ideas

• Bad idea: $O(M \times N)$

• Good idea:
  – $O(N \times \log N) + O(M \times \log N)$
  – = $O((N+M) \times \log N)$

• “Bad idea” is actually better if $M$ very small
For Project #2

• You will need to do cell location.
• You should exploit the properties of rectilinear grid.
  – Search along X coordinates for X-position
  – Search along Y coordinates for Y-position
• Setup time: none
• Search time: $O(\sqrt{N}) + O(\sqrt{N}) = O(\sqrt{N})$
Example:

- $X = \{ 0, 1, 3, 5, 8 \}$;
- $Y = \{ -1, 1, 3, 7, 9 \}$;
- Which cell contains $(2.5, 8.5)$?
- Answer: cell with logical indices $(1, 3)$
- Because
  - $X[1] < 2.5 < X[2] \iff$ so it logical index for $x$ is $1$
  - $Y[3] < 8.5 < Y[4] \iff$ so it logical index for $x$ is $3$
- Don’t believe me? Draw a picture…
Cell location for project 2

- Traverse X and Y arrays and find the logical cell index.
  - X={0, 0.05, 0.1, 0.15, 0.2, 0.25}
  - Y={0, 0.05, 0.1, 0.15, 0.2, 0.25}
- (Quiz) what cell contains (0.17,0.08)?
  = (3,1)
Facts about cell (3,1)

• It’s cell index is 8.
• It contains points (3,1), (4,1), (3,2), and (4,2).
• Facts about point (3,1):
  – Its location is (X[3], Y[1])
  – Its point index is 9.
  – Its scalar value is F(9).
• Similar facts for other points.
• we have enough info to do bilinear interpolation
Facts about cell (i, j)

• It’s cell index is \((nX-1)*j+i\).
• It contains points \((i,j), (i+1,j), (i,j+1), \) and \((i+1,j+1)\).
• Facts about point \((i,j)\):
  – Its location is \((X[i], Y[j])\)
  – Its point index is \((nX)*j+i\).
  – Its scalar value is \(F(nX*j+i)\).
• Similar facts for other points.
• \(\rightarrow\) we have enough info to do bilinear interpolation
Outline

• Projects & OH
• Review
• The Data We Will Study (pt 2)
  – Meshes
  – Interpolation
• Cell Location
• Project 2
Project 2: Field evaluation

• Goal: for point P, find F(P)
• Strategy in a nut shell:
  – Find cell C that contains P
  – Find C’s 4 vertices, V0, V1, V2, and V3
  – Find F(V0), F(V1), F(V2), and F(V3)
  – Find locations of V0, V1, V2, and V3
  – Perform bilinear interpolation to location P
Project 2

• Assigned today, prompt online
• Due January 20th, midnight (→ January 21st, 6am)
• Worth 7% of your grade
• I provide:
  – Code skeleton online
  – Correct answers provided
• You upload to Canvas:
  – source code
  – output from running your program
What’s in the code skeleton

• Implementations for:
  – GetNumberOfPoints
  – GetNumberOfCells
  – GetPointIndex
  – GetCellIndex
  – GetLogicalPointIndex
  – GetLogicalCellIndex

  – “main”: set up mesh, call functions, create output

} Our bijective function
What’s not in the code skeleton

```c
// pt: a two-dimensional location
// dims: an array of size two.
// The first number is the size of the array in argument X, the second the size of Y.
// X: an array (size is specified by dims).
// This contains the X locations of a rectilinear mesh.
// Y: an array (size is specified by dims).
// This contains the Y locations of a rectilinear mesh.
// F: a scalar field defined on the mesh. Its size is dims[0]*dims[1].
float EvaluationFunction(const float *pt, const int *dims, const float *X, const float *Y, const float *F)
{
    return 0; // IMPLEMENT ME!!
}
```

... and a few other functions you need to implement
Cell-centered data