Robustness and Accuracy of Donor Search Algorithms on Partitioned Unstructured Grids

Beatrice Roget and Jay Sitaraman
University of Wyoming
Outline

• Motivation
  – Introduction to PUNDIT
  – Characterization of search robustness issues
  – Description of current search algorithm (Approximate Inverse Map)
• Alternating Digital Tree Search
• Exact Inverse Map
• Performance results for few test cases
• Conclusions
What is PUNDIT?

• PUNDIT stands for Parallel Unsteady Domain Information Transfer
  – Domain connectivity module in HELIOS (Developed through DoD/HIARMS/CREATE-AV program)

• PUNDIT provides fully automated domain connectivity support in parallel (distributed memory) computing systems
  – Requests only local grid and solution data as known to solvers
  – Uses solver based grid partitioning

• Salient Features of PUNDIT
  – Implicit fringe determination search strategy, i.e fringes are not explicitly specified
  – In the case of multiple overlapping grids, grids with best resolution is used for flow solution and all others are interpolated
  – More search operations than traditional explicit hole-cutting techniques since candidate receptor points can include the entire grid
  – Minimum hole-cutting using ray-tracing

Implicit Fringe Determination

Interpolate flow variables between multiple meshes and solvers every solution iteration

Determine fringes, donors, hole points, interpolation weights (Domain Connectivity)

- Automation
- Optimal connectivity
- Interpolation accuracy
- More expensive

PEGASUS (Rogers 2003)
NAVAIR (Lee 2004)
Implicit-hole cutting

Implicit-hole cutting procedure attempts to find donor cells for all grid nodes. Donors are selected if they have better resolution capacity. Resolution capacity is a heuristic parameter that quantifies solution quality (cell volume is used now for donor cells and averaged cell volume for grid nodes).

Fringes: Grid nodes that could find donor cells of better resolution capacity, flow solution will be interpolated to these nodes.

Hole points: Grid nodes that could not find donors but were found inside a solid wall.

Solver points: Grid nodes where flow variables are being solved.

Fringes and solver points are mutually exclusive to maintain donor quality.
AGARD A2 slotted airfoil solution
Original Search Algorithm

- **Meta-data structure**
  - Approximate Inverse Map (aIM) for efficient search

(a) Intersecting spheres (partitioned grids shown)
(b) Oriented bounding boxes created using inertial bisection
(c) Inverse map is created by dividing the bounding boxes into smaller sub-blocks and re-ordering the cells based on "cell-center" containment
(d) Only sub-blocks that contain mesh cells are shown.
Donor search using stencil walk

Vision space bins

Spiral search path

receiver point

Stencil walk path
Search Issues: Orphans and Incomplete Fringes

- **Orphan point**
  - **Field point** that immediately borders a hole-point (both near-body and off-body)
  - **Compulsory receptor** that did not get a donor (only in near-body meshes)

- **Incomplete Fringe point**
  - **Field point** that contain a hole-point in its discretization stencil (both near-body and off-body)
• We found most test cases have a few off-body orphans and incomplete fringes

• Further investigation showed that most of these orphans are generated because the donor-search fails at partition boundaries

• Problem may be in the extension of the stencil-walk algorithm (especially to walk back into the domain)
Orphan point problem: example

No cell center in subblock → spiral search

Grid boundary

Query point

Subblock containing query point
Orphan point problem: example

- Spiral search region
- Closest cell center = start point for stencil walk
- Stencil walk
Orphan point problem: example

Spiral search region

Exit domain

→ Spiral search for possible walk-back
Orphan point problem: example

boundary faces within spiral search region (centroid in subblock)
Orphan point problem: example

- Walk-back point is missed because boundary face (center) is outside of spiral search region.
- Realized that we need a more robust search algorithm

Standard ADT search problem in 2D:

Given a list of elements (in 2D, elements=2D points): \( \{x_1, x_2, \ldots, x_N\} \)

determine all elements inside a search region: limits \( \{x_{\min}, x_{\max}\} \)

\( y_{\min} < y_i < y_{\max} \)

Example in 2D:

Algorithm adapted from: “An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems”
Step 1: organize elements in a binary tree structure:

- At each tree node, divide elements into two groups according to position along a dimension (using median will result in a balanced tree)
- Alternate dimensions
- Each tree node is associated with an element and a region of space

Example in 2D:
Step 2: search the ADT for elements inside search region:

- At each tree node, check for region overlap with search region
- If overlap, check for element containment
- If no overlap, ignore all node descendants

Example in 2D:
Adapting the ADT algorithm for PUNDIT

Elements in ADT are **cell bounding boxes** i.e., 6D elements:

\[
\{ x_l, y_l, z_l, x_u, y_u, z_u \}
\]

**PUNDIT** problem: find the element containing Point \( P = \{ x_P, y_P, z_P \} \) i.e. bound **search point coord.**

i.e.

\[
0 < x_l < x_P < x_u < 1 \\
0 < y_l < y_P < y_u < 1 \\
0 < z_l < z_P < z_u < 1
\]

**ADT** problem: find the element(s) inside search region i.e. bound **element coordinates**

**PUNDIT** problem can be re-phrased to fit **ADT** problem by bounding the coordinates of the element (bounding box):

\[
0 < x_l < x_P < x_u < 1 \\
0 < y_l < y_P < y_u < 1 \\
0 < z_l < z_P < z_u < 1
\]
1/ Initial guess for stencil walk should always be inside target subblock:

query point

start point

2/ In case of stencil walk exiting grid boundary, check all other boundary faces Inside target subblock (by bounding box):

→ even number of new intersections: query point is outside grid domain

→ odd number of new intersections: query point is inside grid domain (a donor cell can be identified)
Improving robustness

Advantages of improved method (Exact Inverse Map)

1/ removes need for time-consuming spiral search

2/ query points in subblocks with no true intersection with grid domain can be identified immediately as field points

3/ Robust: no orphan point should be generated

Cost:
Need to identify a cell point inside each subblock to serve as stencil walk initial guess: increased preprocessing time
Cell point search

Problem = Identifying a cell point inside each **subblock without cell center**
(to serve as stencil walk initial guess)

This is done by geometric considerations:

if such a point exists, one of the following must be true:

- a cell vertex is inside the subblock (case 1)
- a cell edge intersects a face of the subblock (case 2)
- a cell face intersects an edge of the subblock (case 3)
- a cell contains the subblock entirely (case 4)

Otherwise, no such point exists and the subblock has no true intersection with the grid domain (case 5)

For each subblock without cell center, cells with bounding box intersection are identified. The unique list of vertices, edges, and faces they are composed of is then extracted. Since cases 1 to 5 are increasingly time-consuming to identify, they are checked in this order.
case 1: cell vertex inside subblock

Cell point search: case 1
case 2: cell edge intersects subblock face
case 3: cell face intersects subblock edge

Cell face/ SB edge intersection points

Cell center (outside subblock)

Cell point inside subblock
case 4: cell entirely contains subblock

Need only check if one vertex of subblock is inside cell

Cell point inside subblock = subblock centroid
PUNDIT problem: example in 2D

25 subblocks
PUNDIT problem: example in 2D

25 subblocks   ------  16 subblocks with cell center
PUNDIT problem: example in 2D

25 subblocks ------ 16 subblocks with cell center
7 subblocks with cell point
25 subblocks ------ 16 subblocks with cell center
7 subblocks with cell point
2 subblocks with no grid intersection
Search example in 2D

1 boundary intersection $\rightarrow$ query point outside domain
Search example in 2D

query point outside domain
Search example in 2D

Initial guess

query point

donor cell

2 boundary intersections → query point inside domain
Search example in 2D

- donor cell
- query point
- Initial guess
Optimal Subblock Size?

- **Pre-processing time decreases** as sub-block size increases (less sub-blocks without cell center)
- **Search time increases** as sub-block size increases (more cells to search among)
- **Optimal subblock size** observed to be such that about 20% subblocks contain cell centers (**20% cell containment**)

---

**Graphs:**

- **TRAM:** Pre-processing time decreases with increasing sub-block size, search time increases, optimal sub-block size observed to contain 20% cell centers.
- **UH60:** Similar to TRAM, but with different scale.
- **MDART:** Similar to TRAM, but with different scale.

---

% of subblocks containing cell centers

Sub-block size

---

20% cell containment
Results: TRAM case

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Near-Body cells</td>
<td>0.85 Millions</td>
</tr>
<tr>
<td>Off-Body cells</td>
<td>17.33 Millions</td>
</tr>
<tr>
<td>Nb of Processors</td>
<td>16</td>
</tr>
</tbody>
</table>
Results: TRAM case

**ACCURACY**

<table>
<thead>
<tr>
<th></th>
<th>RECEPTORS</th>
<th>ORPHANS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADT method</td>
<td>223326</td>
<td>0</td>
</tr>
<tr>
<td>approx. Inverse Map method</td>
<td>222827 (-499)</td>
<td>0</td>
</tr>
<tr>
<td>exact Inverse Map method</td>
<td>223326 (-0)</td>
<td>0</td>
</tr>
</tbody>
</table>

**SPEED**

Task share (% of total time per time step)

- **ADT**
  - Solve: 43%
  - Search: 57%
  - Preproc: < 1%

- **approx. Inverse Map**
  - Solve: 76%
  - Search: 23%
  - Preproc: 1%

- **exact Inverse Map**
  - Solve: 94%
  - Search: 4%
  - Preproc: 2%

Total time per time step (sec)

- ADT: 200 sec
- aIM: 100 sec
- eIM: 50 sec
<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Near-Body cells</td>
<td>4.57 Millions</td>
</tr>
<tr>
<td>Off-Body cells</td>
<td>7.3 Millions</td>
</tr>
<tr>
<td>Nb of Processors</td>
<td>128</td>
</tr>
</tbody>
</table>
Results: UH60 case

### ACCURACY

<table>
<thead>
<tr>
<th></th>
<th>RECEPTORS</th>
<th>ORPHANS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADT method</td>
<td>81223</td>
<td>0</td>
</tr>
<tr>
<td>approx. Inverse Map</td>
<td>80998 (-225)</td>
<td>50</td>
</tr>
<tr>
<td>exact Inverse Map</td>
<td>81203 (-20)</td>
<td>0</td>
</tr>
</tbody>
</table>

### SPEED

- **Task share (% of total time per time step)**
  - ADT: 66% solve, 34% search, 3% exact Inverse Map, <1% approx. Inverse Map
  - approx. Inverse Map: 93% search, <1% solve, <1% exact, <1% approx.
  - exact Inverse Map: 97% solve, <1% search, <1% exact, <1% approx.

- **Total time per time step (sec)**
  - ADT: 40
  - approx. Inverse Map: 30
  - exact Inverse Map: 30
Results: MDART case

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Near-Body cells</td>
<td>15.57 Millions</td>
</tr>
<tr>
<td>Off-Body cells</td>
<td>65.29 Millions</td>
</tr>
<tr>
<td>Nb of Processors</td>
<td>240</td>
</tr>
</tbody>
</table>
Results: MDART case

ACCURACY

<table>
<thead>
<tr>
<th>Method</th>
<th>Receptors</th>
<th>Orphans</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADT method</td>
<td>1370834</td>
<td>0</td>
</tr>
<tr>
<td>approx. Inverse Map method</td>
<td>1369041</td>
<td>-1793</td>
</tr>
<tr>
<td>exact Inverse Map method</td>
<td>1370784</td>
<td>-50</td>
</tr>
</tbody>
</table>

SPEED

Task share (% of total time per time step)

- ADT: 30% (solve), 67% (search), 3% (preproc)
- approx. Inverse Map: 76%, <1%
- exact Inverse Map: 12%, 2%
- 86%

Total time per time step (sec)

- ADT
- aIM
- eIM
Conclusions

Explored two methods for donor searches (ADT and eIM)

- **ADT search**
  - most robust and accurate for donor search (no orphans or incomplete fringes in any case)
  - 10-20 times slower than both aIM and eIM methods
  - 2-3 orders of magnitude faster than brute force
  - Gold standard to verify accuracy of donor search

- **Exact Inverse Maps (eIM)**
  - Accuracy comparable to ADT
  - 2-5 times faster than approximate inverse maps
  - Optimal sub-block size found to correspond to 20% cell center containment

*This work was supported by grant funding through the Army Research Office with Dr. Roger Strawn as the Technical Monitor*