An algorithm for quadrature on spherical polygonal grids

Christopher Subich
Christopher.Subich@canada.ca
Environment & Climate Change Canada
Dorval, Québec

PDEs on the Sphere
30 April 2019
Montreal, Québec
Outline

1. Objectives
2. Algorithm
3. Results
4. Conclusions & Future Work
Objectives

- Core objective: develop higher-order methods on the spherical icosahedral grid
- Cell-based methods require integrals
  - Finite volume: compute cell averages
  - Finite element: compute inner products
  - Dynamical cores: integrate physics forcings
- Icosahedral grid is necessarily irregular: no simple transformations of a canonical hexagon
- For “online” use, we seek an equivalent to Gaussian quadrature
  - Minimize number of quadrature points per cell
  - Maximize number of (relevant) moment functions integrated exactly
Previous work

- Not very much previous work in this area
- Reeger and Fornberg [2015]: Quadrature rule for the entire sphere given node locations
  - Not directly applicable; we need quadrature for individual volumes
- Mousavi et al [2010]: Quadrature rules for arbitrary polygons
  - Arbitrary planar polygons – spherical polygons add significant complications
  - Root algorithm here is the basis for this work
- Existing numerical methods use geometric arguments for quadrature (e.g.):
  - TRiSK [Thuburn et al, 2009] – cell centroids give a single quadrature point and second order method
  - Cell-centered FV [Subich, 2018] – large tensor-product quadrature rules for precomputation of stencils
Quadrature on a plane

Problem: Given moment functions $M_i(\vec{x})$, find $\vec{x}_j$ and weights $w_j$ s.t.

$$\sum_j M_i(\vec{x}_j) \cdot w_j = \int_{\Omega} M_i(\vec{x}) dA$$

Start with a large, exact quadrature rule, and iterate:
- Eliminate a point
- Adjust remaining degrees of freedom via Newton’s method to “fix” quadrature rule
- Iterate until adjustment no longer converges

Implementation details:
- Helps to have orthogonal moment functions, giving well-conditioned Jacobian (sphere: “mostly-orthogonal”)
- Flexibility in removing points (here augment underdetermined system to push weights towards 0)
- Iteration can incorporate constraints (Mousavi also builds symmetric quadrature rules)
Quadrature on a sphere

- Nice properties of the sphere give some obvious correspondences:
  - Spherical polygons $\rightarrow$ planar polygons via orthographic projection
  - Planar monomials $\rightarrow$ spherical harmonics $\rightarrow$ monomials in Cartesian coordinates

- Each comes with a drawback:
  - Orthographic projection has $(1 + x^2 + y^2)^{-3/2}$ determinant of metric tensor – quadrature rules depend on element size
  - Spherical harmonics are more numerous than monomials in $(x, y)$
  - Spherical harmonics are not linearly independent over small elements
Moment functions

- Consider element rotated to north pole \((z = 1)\)
- Locally, spherical harmonics look like 1, \(x\), \(y\), \(z\), . . .
  - Not all functions are equal
  - \(z = \sqrt{1 - x^2 - y^2} \approx 1 - \frac{1}{2}(x^2 + y^2)\)
  - Dropping \(z\) gives error equivalent to dropping higher-order \(x\), \(y\) terms!

- Two options for resulting quadrature rule:
  - Keep all spherical harmonics up to order \(k\)
    - \(O(\Delta x^{k+1})\) error
    - Exact element-wise integration of spherical harmonics
    - . . . but significant problems with local linear dependence
  - Drop all \(z\) terms from moment functions
    - Also \(O(\Delta x^{k+1})\) error
    - Need fewer quadrature points
    - . . . but no exact integration of spherical harmonics
Moment functions

- For well-conditioned iteration, we want moment functions that are mostly orthogonal.
- Expensive on a per-element basis, but doable once for a reference area.
- Cheat: use square region on orthographic projection:
  - Functions in \( x \) and \( y \) retain their form.
  - Grid elements are small enough that projection does not make a leading-order difference.
  - Orthogonalization is only approximate.
- Moments with \((x,y)\) only are easy: scaled Legendre polynomials.
  - Number of moments to order \( i \) is \((i + 1)(i + 2)/2\), matching triangular truncation.
- Moments with \( z \) are more complicated, must compute via Gram-Schmidt process.
  - Total number of moments to order \( i \) is \((i + 1)^2\).
Moment functions

An example

- Region $(x, y)$, $|x|, |y| \leq \frac{1}{10}$, second-order
- Normalized to root mean square 1
- “Planar” moments (no $z$):
  - $1$
  - $10\sqrt{3}x$, $10\sqrt{3}y$
  - $\sqrt{5} (150x^2 - \frac{1}{2})$, $300xy$, $\sqrt{5} (150y^2 - \frac{1}{2})$
- “Spherical” moments (with $z$):
  - $\frac{3150000}{\sqrt{67}} z + \frac{1581000}{\sqrt{67}} x^2 + \frac{1581000}{\sqrt{67}} y^2 - \frac{6300031}{2\sqrt{67}}$
  - $\frac{10500000\sqrt{1590705237}}{48203189} xz - \frac{10450865\sqrt{1590705237}}{48203189} x$
  - $\frac{10500000\sqrt{1590705237}}{48203189} yz - \frac{10450865\sqrt{1590705237}}{48203189} y$

- Coefficients become very large, very quickly, with near-cancellation of large numbers
- Requires very high working precision, but evaluation via Newton iteration possible in ordinary floating point

Page 9 – 30 April 2019
Moment functions

In pictures

Moment functions to order 1
Moment functions

In pictures

Moment functions to order 2

Page 11 – 30 April 2019
Moment functions

In pictures

Moment functions to order 3

Page 12 – 30 April 2019
Moment functions

In pictures

Moment functions to order 4
Moment functions

In pictures

Moment functions to order 5
Results

- Generated quadrature rules for SVCT-optimized grids
  - Pedro Peixoto’s iModel repository (https://github.com/pedrospeixoto/iModel)
  - Grid levels 1 (42 elements) through 6 (40,962)
- Quadrature rules for 1st – 6th-order moments
  - 2nd–7th-order quadrature rules
  - Both “triangular” and “complete” truncations
- Generation procedure:
  - Define scale factor $r_0$ based on average cell size
  - Compute near-orthogonal moment functions
  - Generate reference quadrature rule for a regular hexagon
  - Adjust regular rule to each element via Newton’s method
Minimal rules

<table>
<thead>
<tr>
<th>Order</th>
<th>Triangular</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>17</td>
</tr>
</tbody>
</table>

- Number of points is close to theoretical lower bound
- Three degrees of freedom per point
- Noticeable “penalty” for full truncation
- Worse with increasing order
Sample quadrature rules

Sample cell (1, 0)

Sole point at element centroid
Sample quadrature rules

Sample cell (1, 1)

Two points necessary to integrate \( z \) exactly
Sample quadrature rules

Sample cell (4, 0)
Sample quadrature rules

Quadrature points are not constrained to the element interior, but remain close.
Convergence rates

Quadrature errors (triangular)

\[ \| \cdot \|_\infty \text{ error for test function } 1 + \tanh(9(z - x - y)) \]

Triangular truncation
Convergence rates

Quadrature errors (complete)

- $\| \cdot \|_\infty$ error for test function $1 + \tanh(9(z-x-y))$
- Full set of moments
Conclusions

- Near-optimal element-wise quadrature on a spherical domain is possible
- Code available now: github.com/csubich/squidpack
  - Rough but working state, generates the figures here
- Numerical next steps:
  - Table-maker’s problem
    - Use higher-precision calculations for full-precision output
    - Current code loses a digit or two
    - High-precision refinement is computationally slow
  - Extend to higher orders yet
    - Core method should extend to arbitrary order
    - Convergence becomes trickier — more iterations per element at 6th versus 1st order
    - Potential for diminishing returns
  - More generic spherical polygons
Conclusions

More future work

- Use the method in a proper numerical model
  - Rotation term requires computation of $\langle \vec{\Omega} \times \vec{u} \rangle$
  - Not everything is a flux
  - Finite element: what do high-order shape functions look like on this grid?

- Add constraints
  - Each generate rule has a few more degrees of freedom than required
  - Add optimization pass for e.g. “keep quadrature points inside the element”
  - Potential for more strongly-constrained rules
    - Multimoment FV-style: quadrature points are vertices plus one free node inside the element
References


