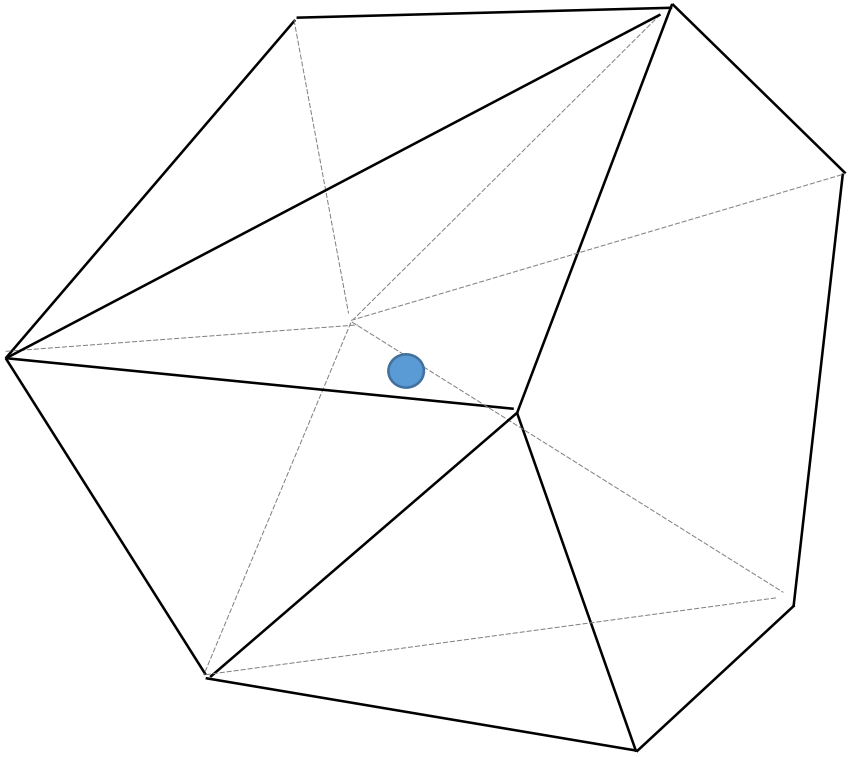
Optimizing Unstructured Grids for TPFA

Abhinav Mishra and Sheleem Kashem

Cell Centre Location



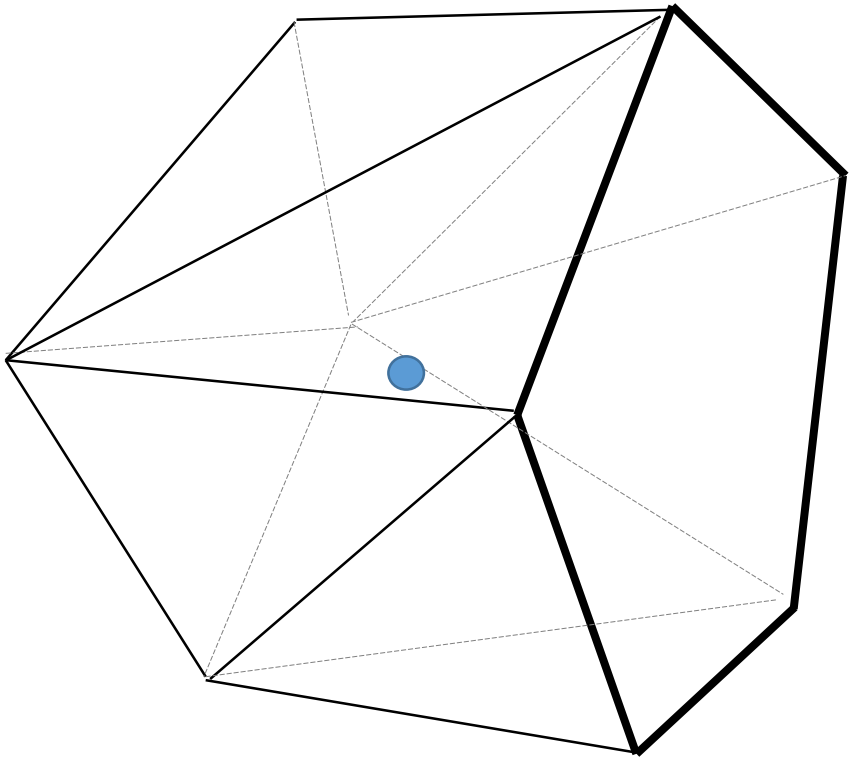
Cell centre is used for:
1. Cell Volume computation
2. Flux discretization

Cell Centre Location

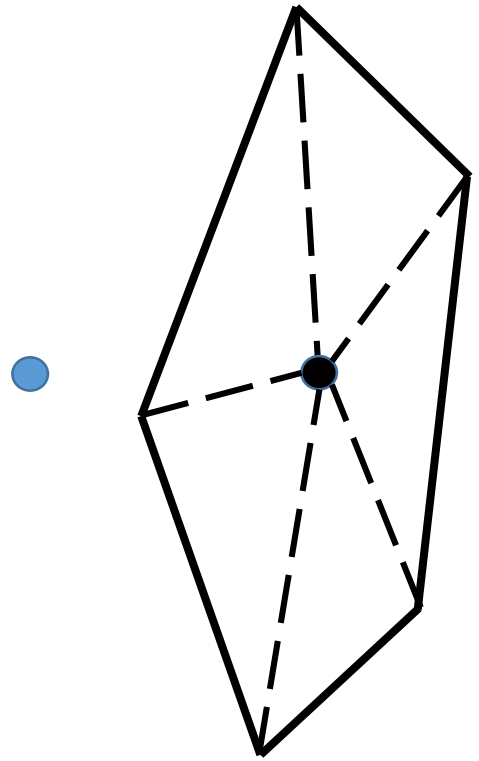
We want to make the cell centre location:

1. Irrelevant to the cell volume computation.
2. Optimal for the flux discretization.

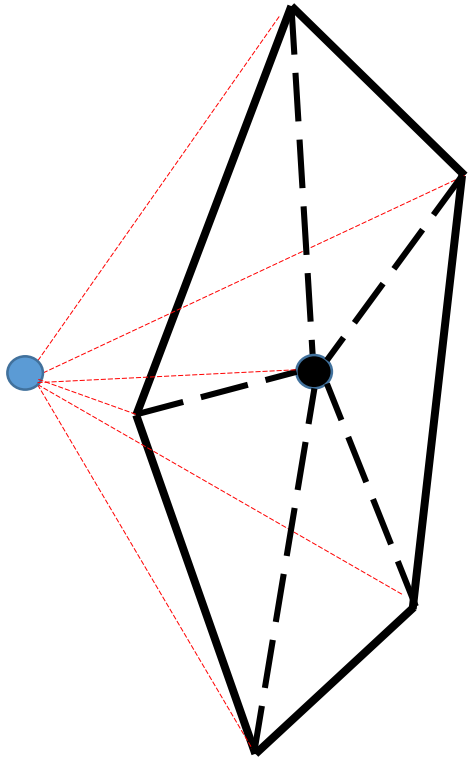
Cell Volume



Cell Volume



Cell Volume



Cell Volume

Divergence of position vector $\mathbf{r} = [x\ y\ z]^T$ is just

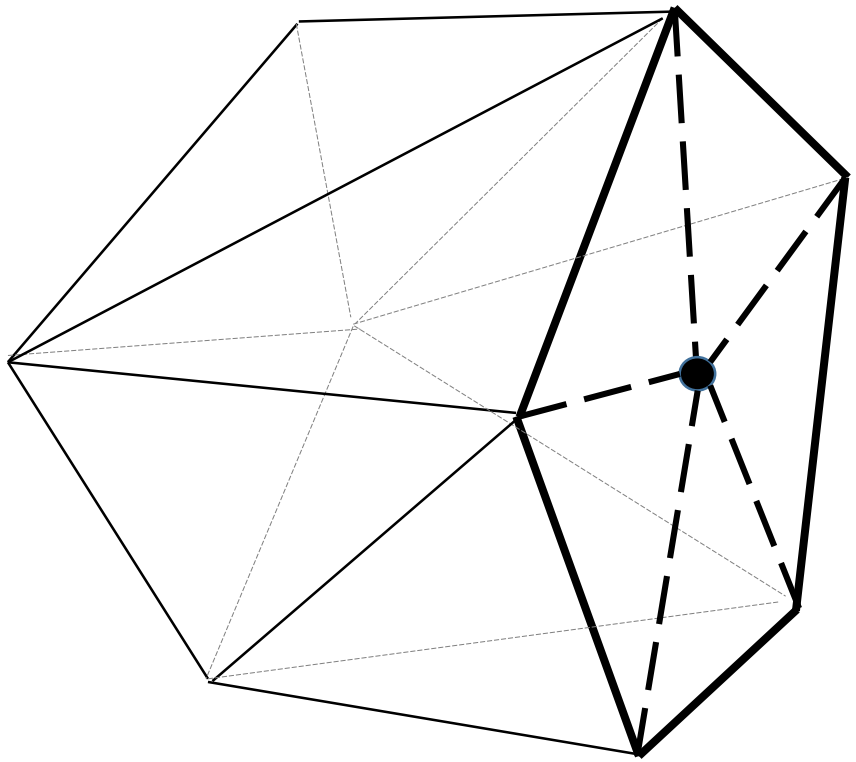
$$\nabla \cdot \mathbf{r} = \frac{\partial x}{\partial x} + \frac{\partial y}{\partial y} + \frac{\partial z}{\partial z} = 3$$

Using this identity and Divergence Theorem

$$Cell\ Volume = \iiint_V 1\ dV = \frac{1}{3} \iiint_V \nabla \cdot \mathbf{r}\ dV = \frac{1}{3} \oiint_S \mathbf{r} \cdot \mathbf{n}\ dS$$

Cell Volume

Triangulate the surface S as before:



Cell Volume

Then if S is composed of triangles T_1,…, T_n

$$Cell\ Volume = \frac{1}{3}\oiint_S \mathbf{r}\cdot\mathbf{n}\,dS = \frac{1}{3}\sum_i\left(\oiint_{T_i}\mathbf{r}\cdot\mathbf{n}_i\,dT_i\right)$$

Equation of a plane (for each triangle) is

$$\mathbf{r}\cdot\mathbf{n}_i = k$$

where k is some constant. So we can just pick one of the vertices of the triangle and substitute this in for $\mathbf{r}$.

Cell Volume

$$Cell\ Volume = \frac{1}{3}\sum_i \left(\oiint_{T_i} \mathbf{r} \cdot \mathbf{n}_i \, dT_i \right)$$

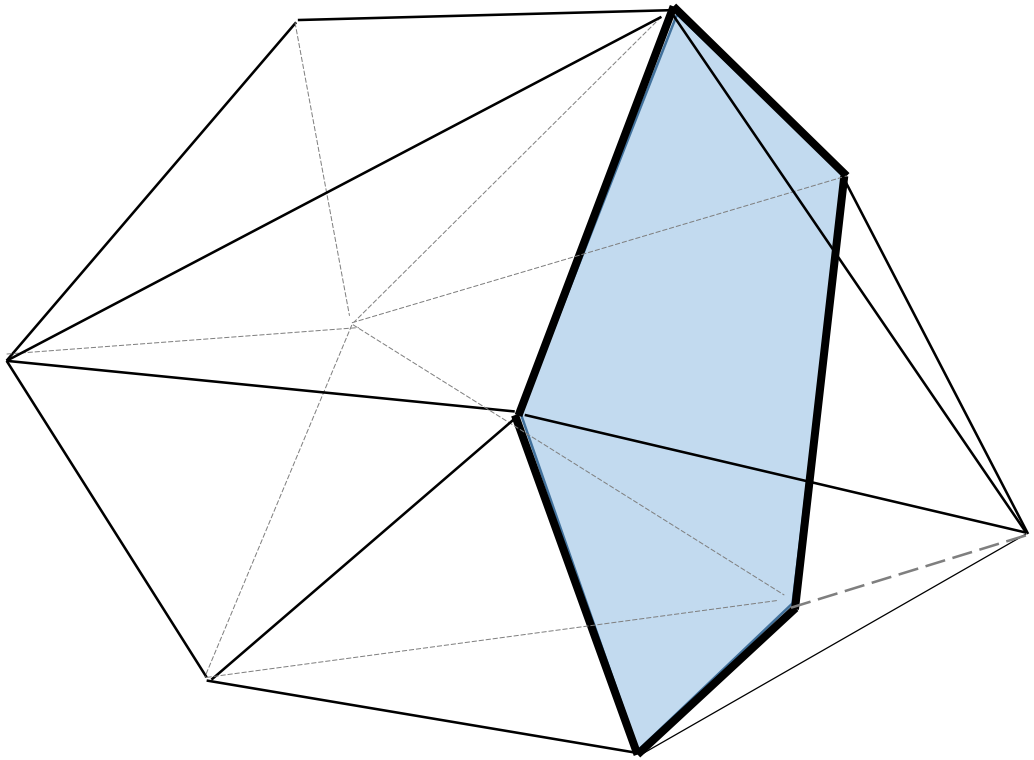
$$= \frac{1}{3}\sum_i \left(\mathbf{a}_i \cdot \mathbf{n}_i \oiint_{T_i} dT_i \right)$$

$$= \frac{1}{6}\sum_i \mathbf{a}_i \cdot [(\boldsymbol{b}_i - \mathbf{a}_i) \times (\boldsymbol{c}_i - \mathbf{a}_i)]$$

where $\mathbf{a}_i$, $\mathbf{b}_i$, $\mathbf{c}_i$ are the vertices of triangle i.

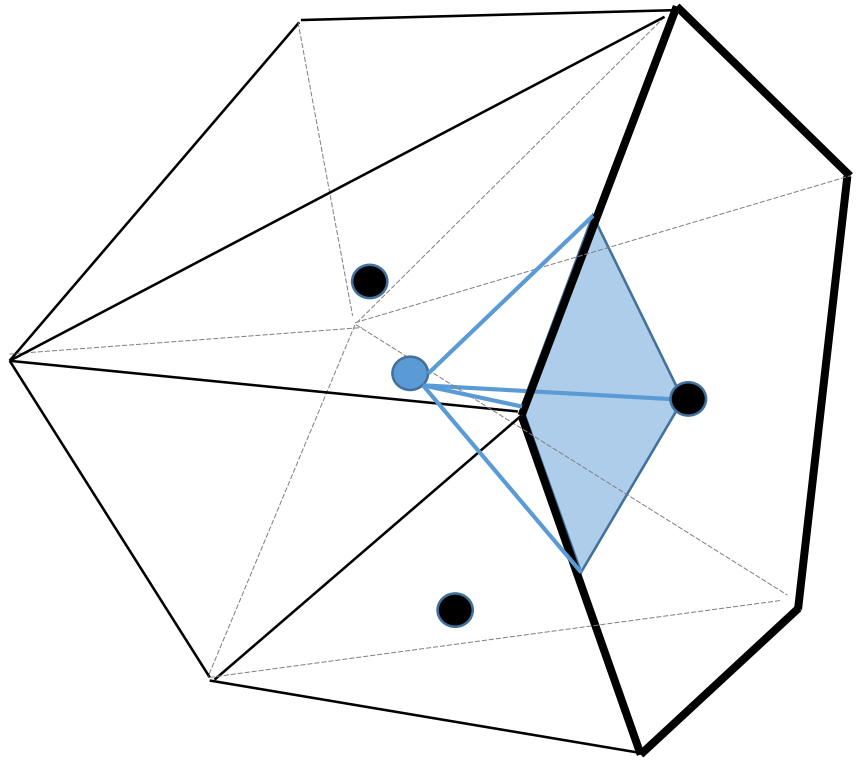
Cell Volume

Only compute contribution from shared faces once for both cells:



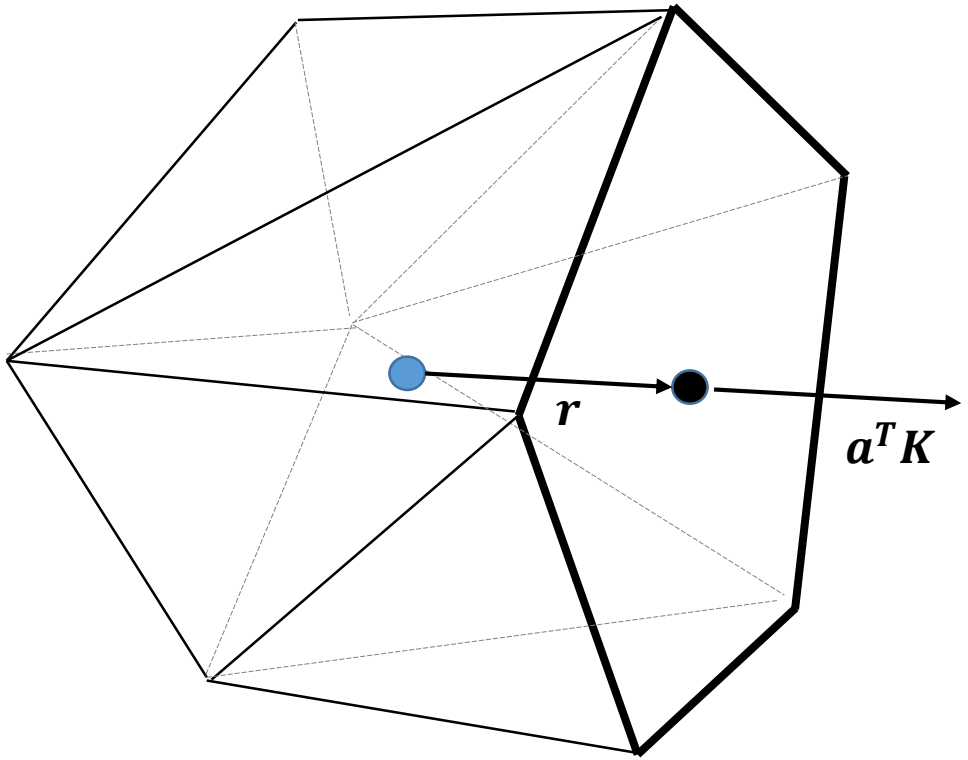
Flux Discretization

Cell centre location determines piecewise linear potential variation



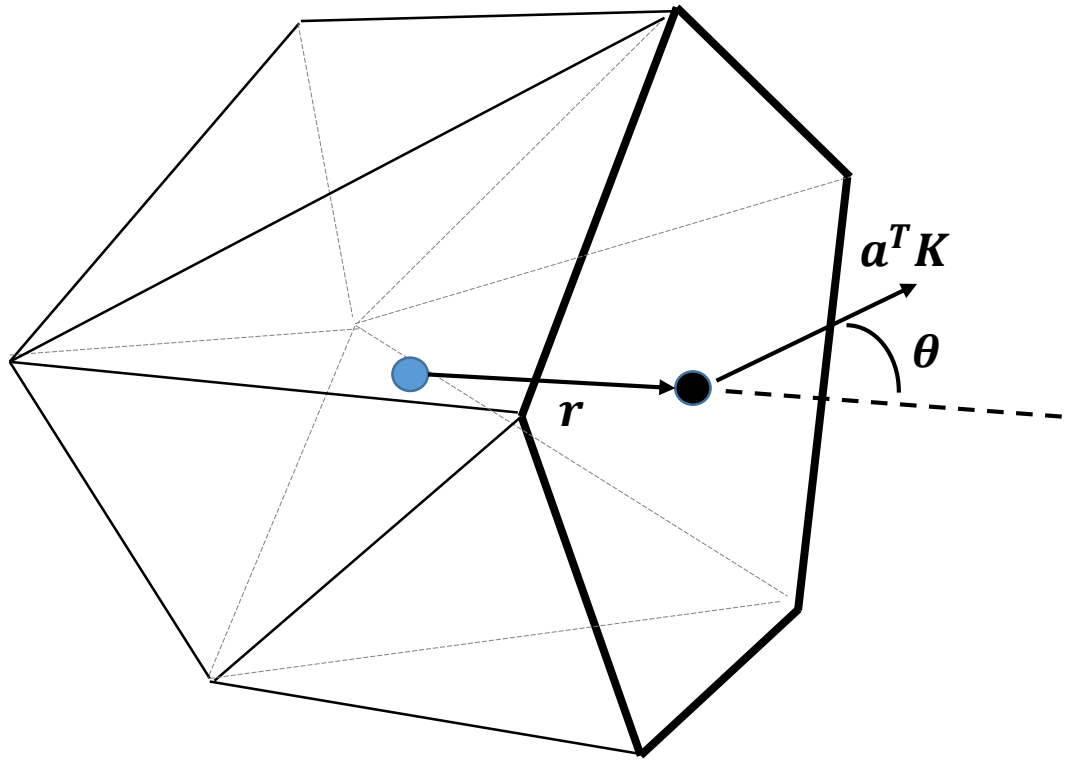
Flux Discretization

Two-Point Flux Approximation (TPFA) assumes K-orthogonal grid



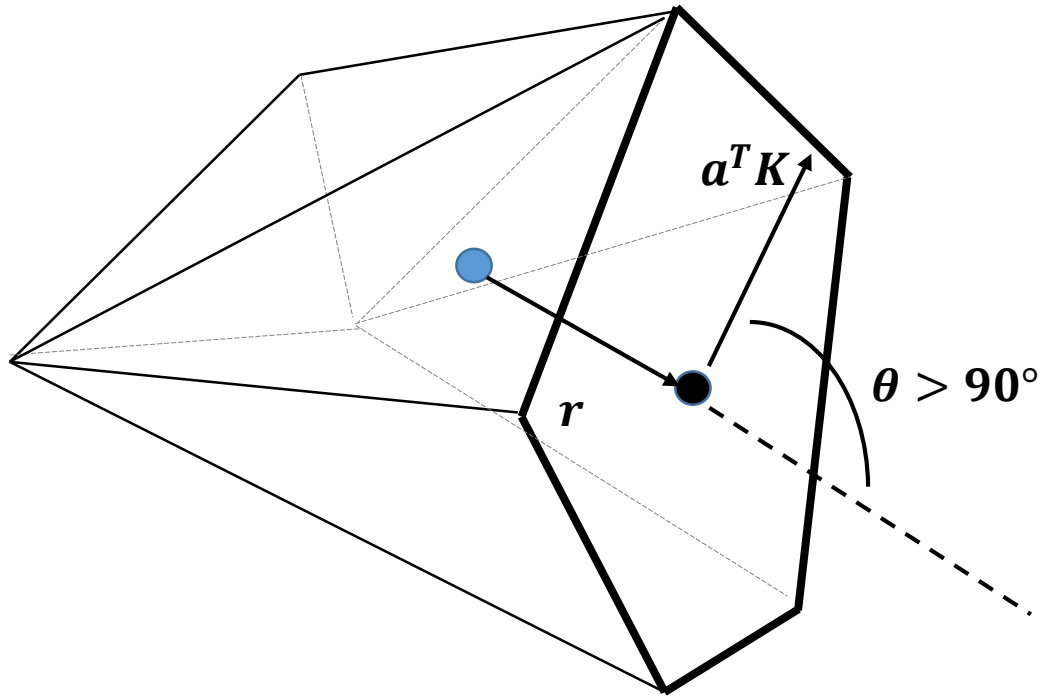
Flux Discretization

Multi-Point Flux Approximation (MPFA) appropriate if not K-orthogonal



Flux Discretization

Negative TPFA transmissibility possible even for convex cells



Flux Discretization

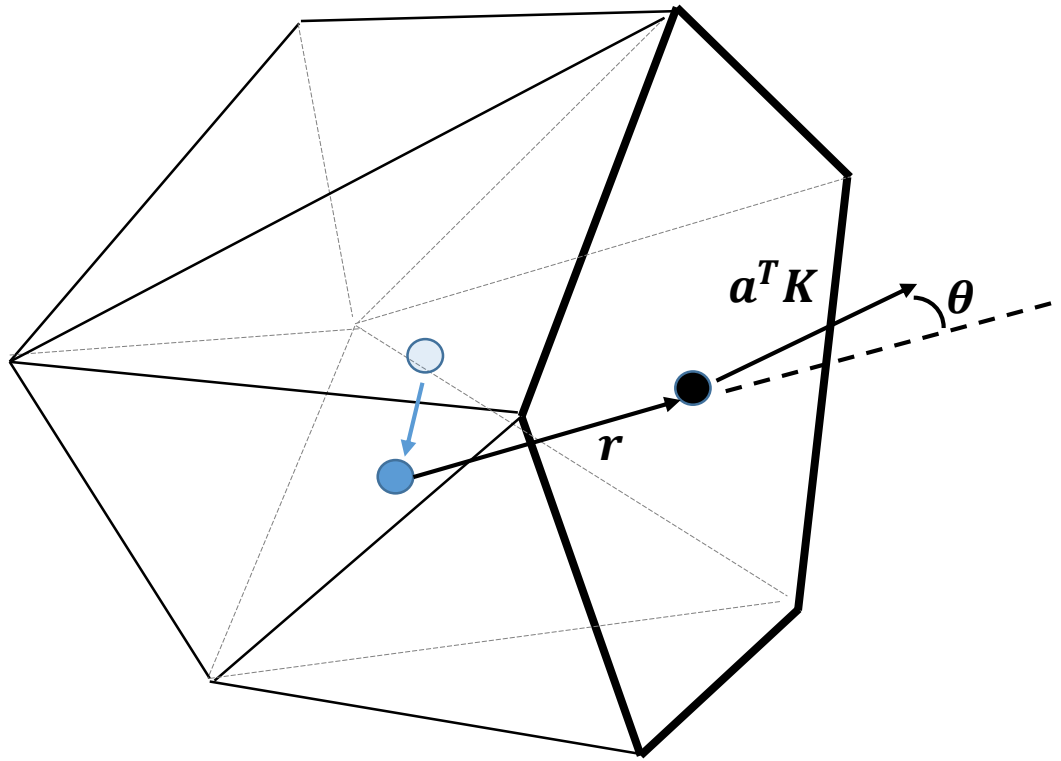
- MPFA is more expensive (denser Jacobian) and less robust (conditionally convergent, conditionally monotone) than TPFA.

- MPFA L-Method, Vertex Centred Schemes and Meshfree Methods address some of these issues, but compound the complexity and introduce new problems.

- Desirable to post-process grid (after generation step) to improve suitability for TPFA and keep life simple.

Flux Discretization

Optimize cell centre locations to improve K-orthogonality



Flux Discretization

Perform global optimization to minimize objective function

$$J(\mathbf{c}_1, \dots, \mathbf{c}_N) = \sum_{i=1}^{N} \left(\frac{1}{n_i} \sum_{j=1}^{n_i} \theta_j^i(\mathbf{c}_i) \right)$$

subject to the following constraints:

1. Cell centre $\mathbf{c}_i$ must be inside cell i
2. $\theta_j^i < 90° \; \forall \; i, j$

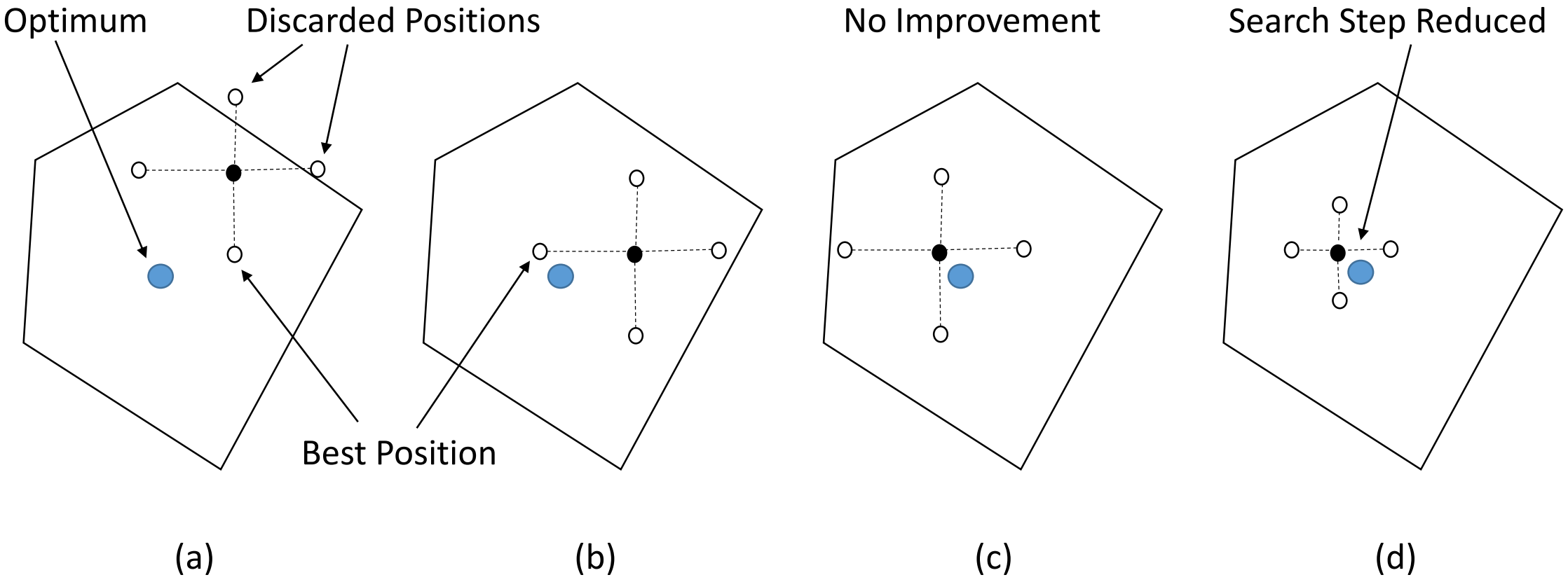
Flux Discretization

Several possible ways of performing the optimization, eg:

1. Direct (“compass”) search
2. Gradient descent
3. Convex optimization

We choose method 1 for simplicity.

Direct (“Compass”) Search



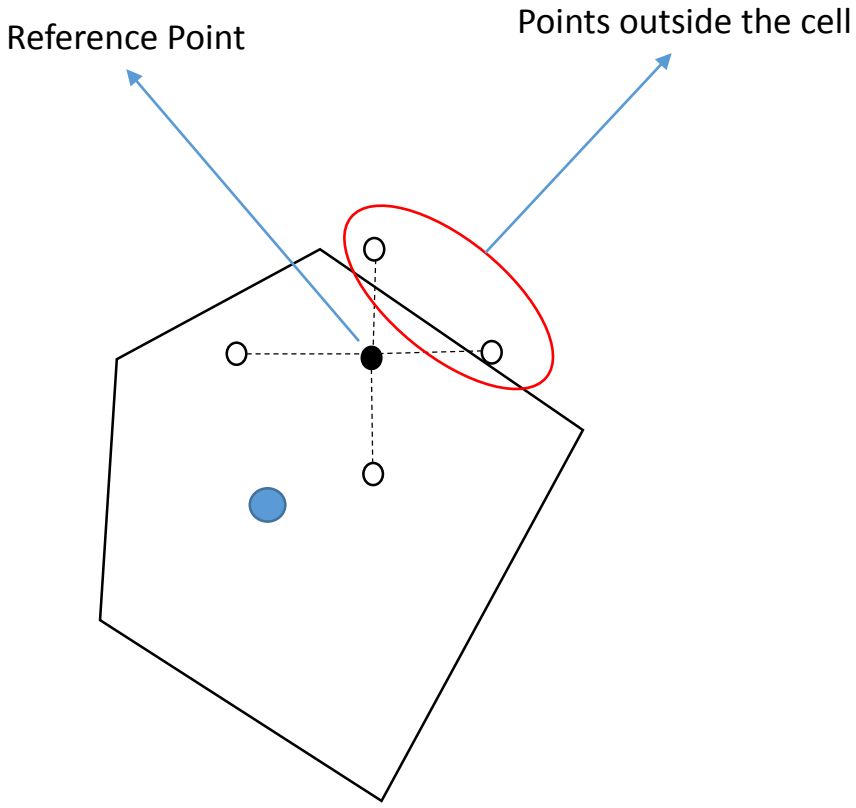
Implementation

- The algorithm is a basic brute force mechanism.
- It compares all the possible positions of the cell centres by computing their cost functions and selecting the one having the smallest cost.
- You have the option to specify the step size, the number of times to iterate over the cells and get a detailed report of every iteration and the change in the cost function.

```
while (step > Globals.INITIAL_STEP/stepSize) {
    changed = false;
    currentBest = calculateAverageAngles(cell);
    Vector[] positions = probablePositions(cell, step);

    for(Vector v : positions) {
        cell.setCentre(v);
        probableBest = calculateAverageAngles(cell);

        if (probableBest < currentBest) {
            currentPosition = v;
            currentBest = probableBest;
            changed = true;
        }
    }

    if (changed) {
        bestPosition = currentPosition;
        cell.setCentre(bestPosition);
        originalCentre = currentPosition;
    } else {
        cell.setCentre(originalCentre);
        step /= 2;
    }
}
```

Optimisation

- To determine whether the prospective point is inside the cell or not, we use a reference point which is already inside the cell.
- Start with the cell having the least amount of neighbours.
- Option to stop the iterations if the last *k* iterations had the same overall cost.



Verification and Result

- For testing purposes a sugar cube grid was supplied with wrong cell centres. The algorithm replaced all the centres with the most optimal cell centre, the barycentre in this case.
- All the methods were unit tested.
- For other type of grids, the change in cost varied from 1.5% to 5%.

References

1. Karimi-Fard, M. Grid Optimization to Improve Orthogonality of Two-point Flux Approximation for Unstructured 3D Fractured Reservoirs, 11th European Conference on the Mathematics of Oil Recovery, September 2008

Source code

https://github.com/mishrabhinav/orthogonal-grid-gen