Structured Grids

Michael Hanke

School of Engineering Sciences

Program construction in C++ for Scientific Computing

Outline

Introduction

Given

- a geometry
- a partial differential equation
- initial and boundary conditions

we need to

1. Discretize the domain (generate a grid)
2. Approximate the PDE on the grid (e.g., by finite elements or finite differences)
3. Solve the discretized pde

Aim: Develop C++ features when implementing a class for so-called structured grids.

Different Types of Grids

Grid

Subdivision of domain into small cells or a finite set of points intended for approximating PDEs by algebraic equations

Structure

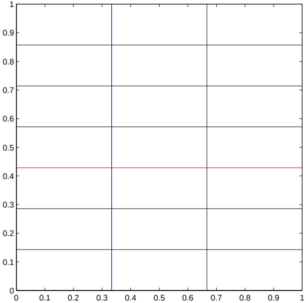
- Unstructured grids
- Structured grids
  - Cartesian
  - Boundary-fitted

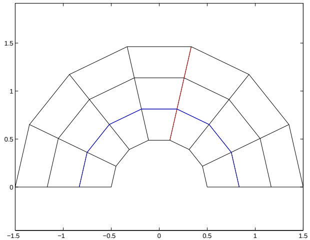
Boundary representation

- Cartesian
- Boundary-fitted
  - Unstructured
  - Structured

Here, we will use *structured boundary-fitted grids*.

A Structured Grid





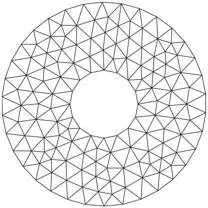
Structured grids are indexed along coordinate axes:

$$\xi = \text{"radius"}, \quad \eta = \text{"angle"}$$

$$x = (1/2 + \xi)\cos(\pi\eta), \quad y = (1/2 + \xi)\sin(\pi\eta)$$

Note: The stright lines in the right plot are an artifact from matlab.

Unstructured Grids



+ Generality
  - Handles complex geometries
  - “Straightforward” generation and refinement
- Inefficiency
  - Indirect addressing (inefficient cache usage, many dereferences)
  - Parallelization difficult

Unstructured Grids (cont)

Example implementation:

```
double x[n], y[n];      // Node coordinates
int triang[m][3];       // Nodes in triangles
```

Coordinates must be accessed via

```
x[triang[i][0]], y[triang[i][0]]
```

Alternatively:

```
Point P[n];         // Node coordinates
int triang[m][3];
```

Access:

```
P[triang[i][j]]
```

General Considerations

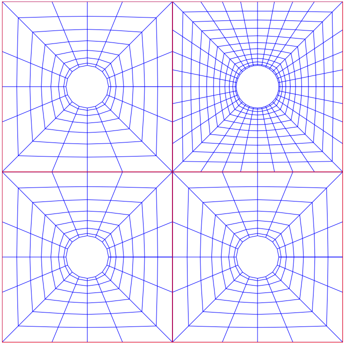
Grid generation

- Should the grid be used once or several times?
- Many grid points gives better accuracy at expense of increased computation time.
- How to distribute grid points: resolve geometry or solution? Both?

Grid properties

- Orthogonality – a "skewed" grid has larger coefficient in truncation error
- Grid size variation – numerical diffusion and stability restrictions in numerical schemes

Multi Block Grids



Divide the domain into blocks when a mapping from the unit square (cube) cannot be found.

- Blocks can be overlapping instead of adjacent (eg, NURBS)
- Nodes on common edges may be different
- Division usually done by hand or "semi-automatically"

Generating a Single Grid

We consider boundary-fitted structured grids, only.
Methods:

- Explicit methods
  - Analytical transformations
  - Algebraic grid generation (transfinite interpolation)
- Implicit methods: The transformation is implicitely determined, often by PDEs.
  - Elliptic grid generation
  - Variational grid generation
  - Hyperbolic and parabolic grid generation

Approach:

1. Divide domain into blocks
2. Generate grid on edges
3. Generate grid on domain (2D) or sides (3D)
4. Generate grid on volume (3D)

Algebraic Grid Generation

- We consider domains $\Omega \subset \mathbb{R}^2$ which can be naturally mapped onto the unit square.
- More precisely, we assume $(\xi, \eta) \in [0,1]^2 \subset \mathbb{R}^2$.
- Moreover, we assume that a one-to-one mapping Φ from the boundary of the unit square onto the boundary of Ω is known.

Aim: Extend Φ to a (smooth) one-to-one mapping $\Phi : [0,1]^2 \to \Omega$.

Basic Idea

- Assume that the mapping Φ as described above is available.
- For given m, n, a uniform grid on $[0,1]^2$ can be defined by:

$$\xi_i = ih_1, \quad h_1 = 1/m, \quad i = 0, \dots, m,$$
$$\eta_j = jh_2, \quad h_2 = 1/n, \quad j = 0, \dots, n.$$

- A strucured grid on Ω can then simply be obtained via

$$x_{ij} = \Phi_x(\xi_i, \eta_j), \quad y_{ij} = \Phi_y(\xi_i, \eta_j), \quad i = 0, \dots, m, j = 0, \dots, n.$$

- *How to get Φ?*

Interpolation Construction

- Assume that we have two strictly monotone functions φ_0, φ_1 with the properties

$$\begin{aligned} \varphi_0(0) &= 1, \quad \varphi_0(1) = 0, \\ \varphi_1(0) &= 0, \quad \varphi_1(1) = 1. \end{aligned}$$

- Then, an interpolation between two points $x, y \in \mathbb{R}^2$ can be be defined by

$$f(s) = \varphi_0(s)x + \varphi_1(s)y.$$

- Example: $\varphi_0(s) = 1 - s$, $\varphi_1(s) = s$. (linear interpolation)

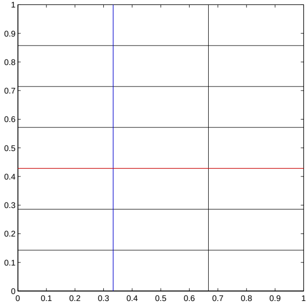
Transfinite Grid Generation

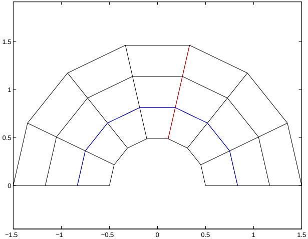
- Application of this interpolation in both ξ-directions provides us with

$$
\begin{aligned}
x(\xi,\eta) = {} & \varphi_0(\xi)x(0,\eta) + \varphi_1(\xi)x(1,\eta) + \varphi_0(\eta)x(\xi,0) + \\
& \varphi_1(\eta)x(\xi,1) - \varphi_0(\xi)\varphi_0(\eta)x(0,0) - \\
& \varphi_1(\xi)\varphi_0(\eta)x(1,0) - \varphi_0(\xi)\varphi_1(\eta)x(0,1) - \\
& \varphi_1(\xi)\varphi_1(\eta)x(1,1)
\end{aligned}
$$

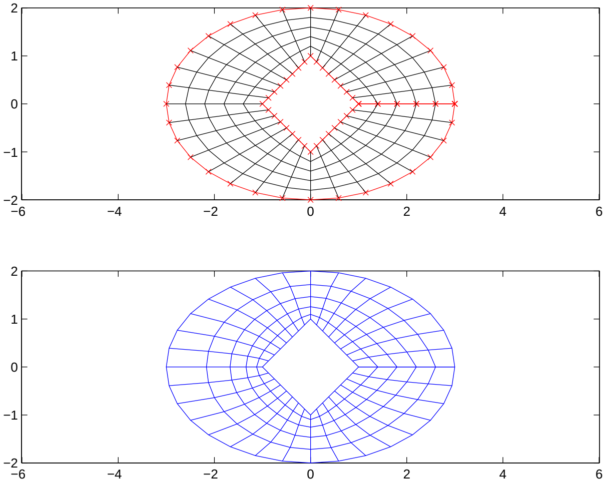
- The subtractions take care of the domain corners.
- This procedure can be generalized to have different kind of interpolations in the ξ and η directions.

A Simpe Example



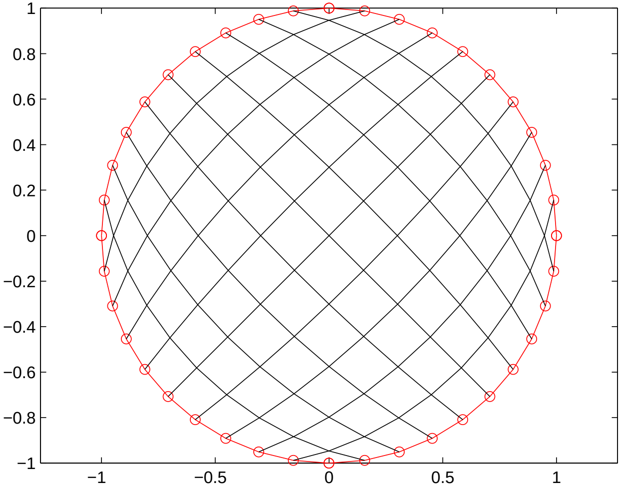


Example: Domain With a Hole

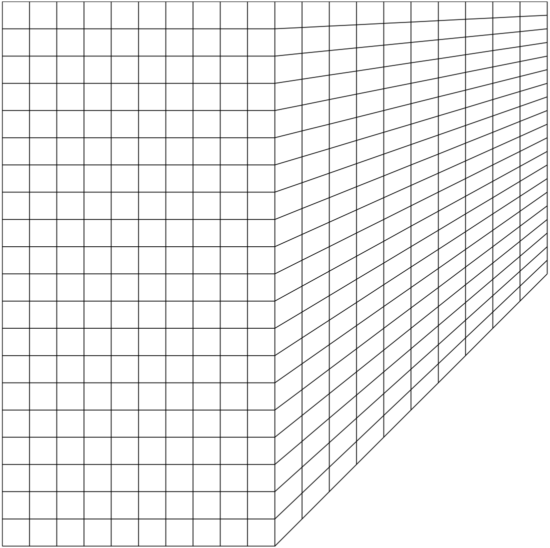


The upper figure shows a grid generated by algebraic grid generation while the second one contains an enhancement by an elliptic process.

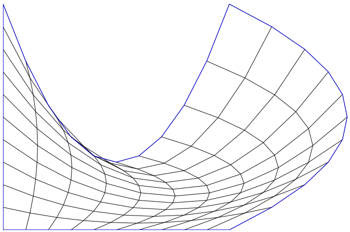
A Final Example

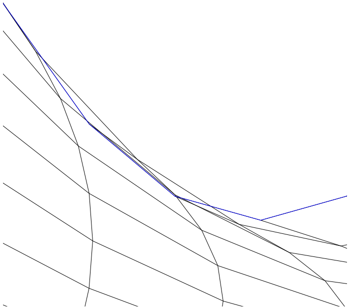


Problem: Propagating Boundary Discontinuity



Problem: Non-Convex Domains





Node Distribution

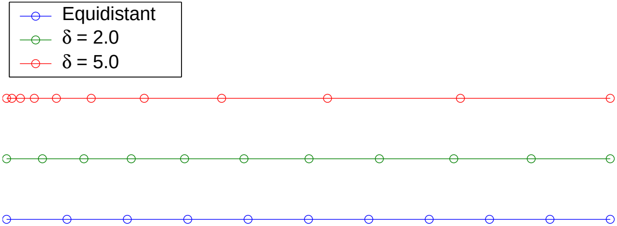
- In the examples above, we have started with a uniform discretization (with respect to the arc length parameter $s \in [0, 1]$) at the boundaries.
- In case of highly non-uniform solutions (e.g., with boundary layers), it might be wise to use a nonuniform distribution in order to keep the grid small.
- Idea:
  - Let us be given a uniform distribution with respect to an artificial parameter $\sigma \in [0, 1]$.
  - The artificial parameter is then mapped analytically to $s \in [0, 1]$: $s = T(\sigma)$.
  - This provides the nodes $s_i = T(\sigma_i)$ with respect to the arc length.

    For this to work, $T : [0, 1] \to [0, 1]$ must be strictly monotone, continuous and $T(0) = 0$, $T(1) = 1$.

Node Distribution (cont)

- T is often chosen according to the principle of truncation error equidistribution.
- Example: Hyperbolic tangent stretching

$$T(\sigma) = 1 + \frac{\tanh \delta(\sigma - 1)}{\tanh \delta}$$



Summary

- Principles of structured grid generation
- Implications on computational efficiency
- Algebraic grid generation
- Nonuniform grids

- What comes next:
    - Inheritance: How to implement classes for structured grids