SF2656/FSF3565 Assignment 3

Björn Wehlin

Structured Grid

In this project you will implement a general class for modeling "4-sided" domains and structured grids on them. In particular, we will work with the domain shown in Figure 1, whose corners are located at $(-10, 0)$, $(5, 0)$, $(5, 3)$, and $(-10, 3)$, and whose bottom boundary is given by the function $f(x) := 1/(2g(x))$, where

$$g(x) := \begin{cases} 1 + \exp(-3[x+6]), & x \in [-10, 3), \\ 1 + \exp(3x), & x \in [-3, 5]. \end{cases}$$

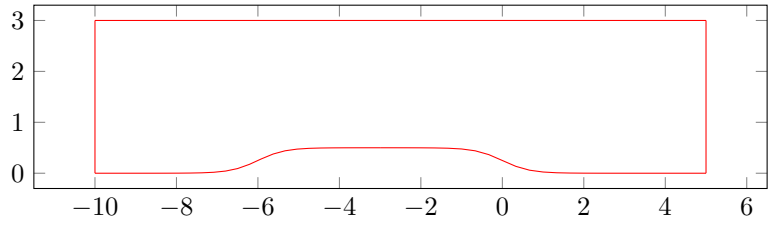

Figure 1: Domain to be gridded.

Consider the following abstract base class (`Point` is as in Homework 2):

```
class Curve {
public:
    virtual ~Curve() = default;
    virtual Point at(double t) const = 0;
};
```

Here, `Curve::at` returns a 2D `Point` in the physical domain, parameterized by $t \in [0, 1]$.

Task 1: Straight Lines and Domain Class (1pt)

- Derive a class `StraightLine` from `Curve` to represent the left, right and top sides of the domain. The class should contain two endpoints, and `at` should interpolate linearly between the points.

- Create a class `Domain` that contains the four boundary curves. For now, replace the bottom curve with a straight line from $(-10, 0)$ to $(5, 0)$.

Task 2: Algebraic Grid Generation (4pts)

Next, implement algebraic grid generation using the method shown in the lectures (transfinite interpolation, TFI). Let $n > 2$ be an integer, and let $h = 1/n$. Your program should generate a grid $(x(ih, jh), y(ih, jh))$, $i, j = 0, \ldots, n$.

The resulting grid point matrices, $[x(ih, jh)]_{ij}$ and $[y(ih, jh)]_{ij}$ should be written to *separate files*, with each row in the file corresponding to a value of i, and consecutive j values on each row separated by a space. An example file for a 5×5 grid may look something like the following.

```
0 0.75 1.5 2.25 3
0.165286 0.873964 1.58264 2.29132 3
0.499722 1.12479 1.74986 2.37493 3
0.012558 0.759419 1.50628 2.25314 3
0 0.75 1.5 2.25 3
```

You can use the supplied Python script to plot your grid using the two grid matrix files as input.

Hint: Be very careful about the orientation of the boundary curves!

Task 3: Bottom Curve (4pts)

We will reparameterize the bottom curve using the arclength reparameterization presented in the lectures. First, derive a class `EquationCurve` from `Curve`, starting from the following skeleton.

```
1  class EquationCurve : public Curve
2  {
3  public:
4      virtual ~EquationCurve() = default;
5      Point at(double t) const override;
6  private:
7      virtual Point gamma(double t) const = 0;
8      virtual Point gammaprime(double t) const
9  };
```

The class corresponds to a smooth regular curve $\gamma\colon [0, 1] \to \mathbb{R}^2$. The function `gamma` returns $\gamma(t)$. The function `gammaprime` returns $\dot{\gamma}(t)$.[1]

You are free to choose whether `gammaprime` is pure virtual, or if it should have a default implementation (what should this be?).

`EquationCurve::at` should use `gamma` and `gammaprime` to compute the points of the reparameterized curve. This computation involves taking an integral, for which you can either use your adaptive Simpson integrator from Homework 1,

[1] Recall that if $\gamma(t) = (\gamma_1(t), \gamma_2(t))$, then $\dot{\gamma}(t) = (\dot{\gamma_1}(t), \dot{\gamma_2}(t))$.

or, e.g., `boost::math::quadrature` (see `https://www.boost.org/doc/libs/1_86_0/libs/math/doc/html/quadrature.html`).

Finally, we derive a class `BottomCurve` from `EquationCurve`. This class should implement $\gamma(t) = f((1-t)(-10) + 5t)$, linearly interpolating over the domain of f so that γ has domain $[0, 1]$.

Replace the bottom curve with an instance of `BottomCurve` and rerun the program to generate a grid for the correct domain.

Plot the resulting grid using the supplied Python script!

Task 4: Point Cache (1pt)

Likely the slowest part of your program will be computing lots of integrals. However, because of the structure of the grid, we will be recomputing a few values over and over. In this final part of the assignment, you will add a cache to `EquationCurve`.

First add an associative container, such as `std::unordered_map<double, Point>`, that stores already computed pairs $(t, \tilde{\gamma}(t))$, where $\tilde{\gamma}$ is the arc-length reparameterization of γ with $t \in [0, 1]$.

Then, in `EquationCurve::at`, first check whether the cache contains the supplied t, and if so, return the corresponding point. Otherwise, compute the point and, before returning from the function, store the computed pair in the cache.

Because `EquationCurve::at` is `const`, you will have to mark your cache member variable `mutable`. This makes it so that the variable can be modified even from `const` member functions.

Usually, we want to avoid using `mutable`, but for things like caches it is reasonable since, to the user, it does not appear as if the owning object has been modified in any way. We just have to be careful that if something is changed on the object that makes the cache invalid, the cache will have to be purged (although this should not be necessary in our case).

Measure the impact on performance from having a cache vs. not having a cache as the number of grid points varies.

Submission

The programming exercises should be done individually, or in groups of two. Hand in a report containing:

- Comments and explanations that you think are necessary for understanding your program. (But do not comment excessively.)
- The output of your program according to the tasks. Don't forget to draw conclusions!
- Printout of the source code in PDF format. (We need this to be able to comment on your code, points will be deducted for missing to do this.)

You can either put the source code as an appendix to your written report, or you can hand in separate PDF(s).

In addition, all source code for your program(s) in .zip/.tar format and instructions for how to compile/run.

Important! Your submission must contain at least the following two files:

- PDF report (not zipped!) with your findings and source code
- .zip/.tar archive containing your source code

Submissions not following this format will not be accepted and will have to be resubmitted (these will be considered late submissions).

Acknowledgment This assignment is largely the same as one by Michael Hanke.

Good luck!