

Numerical Modelling of District Heating and Cooling network

Leanne Dong

May 7, 2020

Overview

- 1 Fundamentals
 - General concepts
 - Graph theory
 - Solution of linear systems
 - Solution of nonlinear systems
- 2 Hydraulic Model
 - Numerical Solution
- 3 Thermal model
 - Thermal Calculation
- 4 Hydraulic-Thermal Model
 - Calculation

District Heating and Cooling Networks (DHN)

- The most common method for heating building in cities
- Usually consist of supply and return pipes that deliver heat in form of hot water or steam

Nonlinear system and root finding

- A system of nonlinear equation is a set of equations

$$f_1(x_1, x_2, \cdot, x_n) = 0,$$

$$f_2(x_1, x_2, \cdot, x_n) = 0,$$

$$\vdots$$

$$f_n(x_1, x_2, \cdot, x_n) = 0.$$

- There are three type of nonlinear system in hydraulic model. The solutions are usually found via **Newton-Raphson** or Hardy Cross Method.
- An example of nonlinear system from the DHN in Scharnhauser Park (Hassine and Eicker 2011)

$$x^2 - 2x - y + 0.5 = 0$$

$$x^2 + 4 * y^2 - 4 = 0$$

- To find the solution (root), we would use the C++ Linear algebra library **Eigen**.

```
#include <iostream>
#include <usr/include/eigen3/Eigen/Dense>
#include <cmath>
#include <vector>

void newton2d()
{
    // F
    auto F = [] (const Eigen::Vector2d &x){
        Eigen::Vector2d res;
        res << pow(x(0),2) -2*x(0)-x(1)+0.5, pow(x(0),2) +4*pow(x(1),2)-x(1)-4;
        return res;
    };

    // jacobian of F
    auto DF= [] (const Eigen::Vector2d &x){
        Eigen::Matrix2d J;
        J << 2*x(0)-2, -1,
            2*x(0), 8*x(1);
        return J;
    };

    Eigen::Vector2d x, x_ast, s;
    x << 2, 0.25; // initial value
    x_ast << 1.9007, 0.3112; // solution
    double tol=1E-10;

    std::vector<double> errors;
    errors.push_back((x-x_ast).norm());

    do
    {
        s = DF(x).lu().solve(F(x));
        x = x-s; // newton iteration
        errors.push_back((x-x_ast).norm());
    }
    while (s.norm() > tol*x.norm());

    // create eigen vector from std::vector
    unsigned int n = errors.size();
    Eigen::Map<Eigen::VectorId> err(errors.data(), n);

    std::cout << "solution" << std::endl;
    std::cout << x << std::endl;

    std::cout << "Errors:" << std::endl;
    std::cout << err << std::endl;

    // compute the convergence rate of each iteration
    Eigen::VectorId logDiff = err.bottomRows(n-1).array().log() - err.topRows(n-1).array().log();
    Eigen::VectorId rates = logDiff.bottomRows(n-2).cwiseQuotient(logDiff.topRows(n-2));

    std::cout << "Rates:" << std::endl;
    std::cout << rates << std::endl;

#include "newton2d.hpp"

int main()
{
    newton2d();
}

//The solution is found as 1.94 and 0.39.
```

Assumption

- The basic assumption for the calculation is incompressible media.
- Computation of flow distributions in DHN are mainly based on the Kirchhoff law for **current** and **voltage** in circuits: The two equations describe **flow rate** and **pressure losses** in the network
- Consider the PDE describing an one dimensional flow through a horizontal pipe which can be systematically derived from the Navier-Stokes equations.

$$\frac{l}{A} \frac{d\dot{m}}{dt} + \Delta p + R|\dot{m}|\dot{m} = 0 \quad (1)$$

Note: Δp denoting the difference in pressure head between the two pipes ends and \dot{m} is the mass flow rate. The variable R stands for the hydraulic resistance of the pipe element, which is postulated to be a function of the physical properties such as length, roughness and diameter.

Network Topology

The description of the heating network is based on graph theory. First we need to form the topological matrix which show the structure of the mesh in matrix form. (To be added)

Solution of network equation system

Basic principle

Numerical solution via finite elements

Model Validation

Network structure

to be added

Calculations

to be added