Shape Optimization of a Heat Exchanger

ID:4873

October 21, 2017

Abstract

In this project we design the shape of a heat exchanger using numerical optimization techniques. Although small, this problem is representative of numerical optimization since it involves three basic ingredients, the shape parameterization, the PDE (partial differential equation) solver, and the optimization tool. We first describe the analysis method, the geometry parameterization, and the mathematical statement of the design problem, then results and conclusion are given.

1 Physics Model

The physics model in this project is a two-dimentional (2D) heat exchanger, as shown in Figure **??**. Since the temperature of water is higher then air, there is a heat flux through the heat exchanger which heat up the air. The goal is to maximum the heat flux per length through changing the air-side shape of the heat exchanger.

The geometric constraints and material property are given below:

$$\begin{aligned} L &= 0.05m \\ \kappa &= 20W/(mK) \\ T_{air} &= 20^oC \\ T_{water} &= 90^oC \end{aligned} \tag{1}$$

The height of the heat exchanger is limited between $h_{min} = 0.01m$ and $h_{max} = 0.05m$.

2 Mathematical Model

To use an numerical optimization technique, the physics model in Section **??** must be rewritten into a mathematical model. The one used in this work is as follows:

$$\begin{aligned} &max\ \ f(h(x)) \\ &\ \ s.t.\ \ h_{min} \leq h(x) \leq h_{max} \end{aligned} \tag{2}$$

where h is the height of the shape, f, the objective function, is the heat flux through the air-side face. The relation between the heat flux f and the shape of the air-side face is governed by the heat transfer equation.

Although looking concise, this problem cannot be solved analytically, and numerical methods must be employed. To this end, the height $h(x)$ has to be replaced by some discrete variables, the process of which is called shape parameterization. The shape parameterization used here is a weighted summation of sinusoids with different frequency:

$$h(x) = a_1 + \sum_{i=2}^{N_{var}} \sin(\frac{2\pi ix}{L}) \tag{3}$$

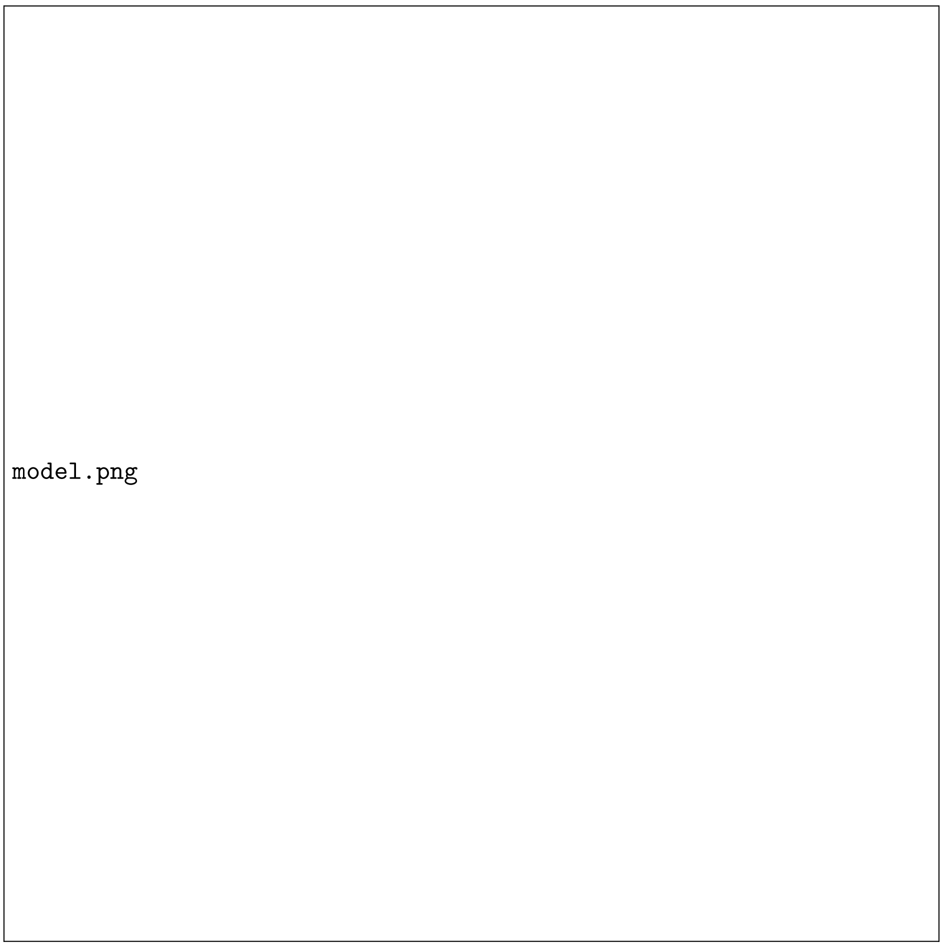

Figure 1: Illustration of the heat-exchanger design problem

where a_i is the design variable, and N_{var} is the number of design variables.

Although not necessary, in numerical optimization we always solve a minimization problem instead of a maximization problem. If the problem itself is a maximization, it can be be easily transformed into a minimization one by negating the objective function. Taking this into account, (**??**) can be rewritten as

$$\begin{aligned} min &\ \ -f(a) \\ s.t. &\ \ h_{min} \leq h(a,x) \leq h_{max} \\ &\ \ f(a) = f(h(a)) \end{aligned} \tag{4}$$

where $a \in \mathbf{R}^{N_{var}}$

3 Analysis function

As mentioned, the heat flux can be calculated based the height. In this work, it is realized in a given function `CalcFlux`. The signature of `CalcFlux` is

```
function [flux, T, dTdx, xy] = CalcFlux(L, h, Nx, Ny, kappa, Ttop, Tbot)
```

where

- L - length of domain in x direction
- h - height as a function of x and a;
- Nx - number of elements along the x direction
- Ny - number of elements along the y direction
- kappa - thermal conductivity
- Ttop - the ambient air temperature along the top of the domain, ie, T_{water}
- Tbot - the fluid temperature along the bottom of the domain, ie, T_{air}
- flux - heat flux per unit length out of top/bottom boundary
- T - temperature over the elements; note size(T) = (Nx*Ny,1)
- dTdx - derivative of T in x direction over the elements
- xy - mesh nodes

4 Implementation

The implementation in Matlab is attached in Appendix **??**∼**??**. Since the optimization tool and the analysis function are both available, the main task is to give the shape parameterization, and build the workflow of the optimization process.

It is worth mentioning that the optimization tool, `fmincon`, takes a function which itself takes only design variables as parameters. This is different from `CalcFlux` shown in Section **??**. Therefore, we need to define a wrapper functions. The first wrapper, defined in Appendix **??**, takes all parameters necessary to compute the objective. The second one, defined in Appendix **??**, is

```
objFunc = @(a) calcObjective(a, x, Ny, kappa, Ttop, Tbot, @calcHeight, @CalcFlux);
```

which, based on the first wrapper `calcObjective`, only takes the design variable a as parameters.

Additionally, since `fmincon` only accepts linear constrains in the form of $Ax \leq b$, we also need to transform the original constrains on h, $h_{min} \leq h \leq h_{max}$ into this form. This is realized in Appendix **??**.

5 Results and discussion

Given $N_{var} = 5, Nx = 20, Ny = 10$, and $a = [0.05, 0.02, 0.01, 0, 0]$ as initial guess, the optimum heat flux is 10097.554, 44.25% larger than the nominal value, 7000. The convergence history is listed in Table ??; see `https://www.mathworks.com/help/optim/ug/fmincon.html` for the descriptions of each column.

As can be seen, it only takes 6 steps for `fmincon` to converge. The final shape of air-side face is shown in ??.

In addition, there are several worth-mention observations in the whole algorithm. First, the results depend on parameters, such as N_{var}, N_x, N_y. This is not difficult to understood: More design variables permit more degree of freedom an d hence, result in larger heat flux. Second, even with this parameters fixed, the convergence heavily depends on the initial guess of design variable a. For example, with $a = [0.05, 0, 0, 0, 0]$, the iteration converges to $h = 0.01$, with corresponding heat flux $f = 6363.636$. This could be explained by the multiple local minimums of the objective function.

Table 1: Convergence history of `fmincon`

Iter	Func-count	Fval	Feasibility	Step-length	Norm of Step	1sr-order-optimality
0	6	-2.837216e+03	2.569e-02	1.000e+00	0.000e+00	1.011e+05
1	12	-7.349836e+03	3.469e-18	1.000e+00	2.467e-02	2.952e+05
2	18	-7.666839e+03	3.469e-18	1.000e+00	2.986e-03	1.408e+05
3	24	-8.090623e+03	1.388e-17	1.000e+00	3.893e-03	8.257e+04
4	30	-8.648809e+03	5.551e-17	1.000e+00	7.012e-03	6.774e+04
5	36	-1.009755e+04	2.082e-17	1.000e+00	1.495e-02	1.134e+05
6	37	-1.009755e+04	2.082e-17	7.000e-01	1.046e-17	1.746e-10

Appendices

A calcHeight.m

```
%
% This function computes the height. It is the parameterization
% of shape of the heat exchanger. The basis is sine functions.
%
function [ h ] = calcHeight(a, x)
Nvar = size(a, 1);
Np = size(x, 1);
L = x(Np);

h = zeros(Np, 1);
for i = 1 : Np
  h(i) = a(1);
  for j = 2 : Nvar
    tmp = 2.0 * pi * (j-1) * x(i) / L;
    h(i) = h(i) + a(j) * sin(tmp);
```

`shape.png`

Figure 2: Optimized shape of the heat exchanger, $N_{var} = 5, Nx = 20, Ny = 10$

```
  end
end
end
```

B calcConstraints.m

```
%
% This function establish the constrains, ie, we have to
% tansform the original constraints
%   h1 <= h <= h2
% into a form of
%   A*a <= b
% which is acceptable by fmincon. Note h1 <= h is equivalent to
%   h <= -h1
%
function [ A, b ] = getConstraints(Nvar, x)
Np = size(x, 1);
L = x(Np);
A = zeros(2 * Np, Nvar);
b = zeros(2 * Np, 1);

for i = 1 : Np
  A(i,     1) =  1.0;        % dh/da1
  A(Np+i, 1) = -1.0;
  for k = 2 : Nvar
    tmp = sin(2 * pi * (k-1) * x(i) / L);
    A(i,       k) =  tmp;  % dh/dak
    A(Np + i, k) = -tmp;
  end
  b(i)       = 0.05;     % upper bound
  b(Np + i) = -0.01;    % lower bound
end
end
```

C calcObjective.m

```
%
% This function computes the objective. Since the PDE solver
% is already given in function calcFlux, this funtion is
% a wrapper of parameterization and PDE solver.
%
function [obj_val] = calcObjective(a, x, Ny, kappa, Ttop, Tbot, calcH, calcFlux)
Np = size(x, 1);
Nx = Np - 1;
L = x(Np);
% get height
h = calcH(a, x);
% get flux
flux = calcFlux(L, h, Nx, Ny, kappa, Ttop, Tbot);
```

```
% negate the flux
obj_val = -flux;
end
```

D optimize.m

```
% speficy some parameters of physics
L = 0.05;
kappa = 20.0;
Ttop = 293.15;
Tbot = 363.15;

% dimension of problem
Nx = 20;
Ny = 10;
Nvar = 5;

% grid spacing in x direction
x = zeros(Nx + 1, 1);
for i = 1 : Nx + 1
x(i) = (i-1) * L / Nx;
end

% get constraint coefficients
[Aineq, bineq] = getConstraints(Nvar, x);

% initial value for design variables
% a0 = ones(Nvar, 1);
a0 = zeros(Nvar, 1);
a0(1) = 0.05;
a0(2) = 0.02;
a0(3) = 0.01;

% the final wrapper of objective function which only takes design variable
% \p a
objFunc = @(a) calcObjective(a, x, Ny, kappa, Ttop, Tbot, @calcHeight, @CalcFlux);

% parameters of optimizer
optns = optimoptions(@fmincon, 'Display', 'iter', 'Algorithm', 'sqp');

% call the optimizer
[a, fval] = fmincon(objFunc, a0, Aineq, bineq, [], [], [], [], [], optns);

% manually check constraints
h = calcHeight(a, x);
tol = 1.e-12;
if hmin < 0.01-tol || hmax > 0.05+tol
fprintf('Error:␣height␣constraints␣violated,␣hmin␣=␣%16.12f,␣hmax␣=␣%16.12f.\n', hmin, hm
end
```

```
% plot the shape
plot(x,h);
```