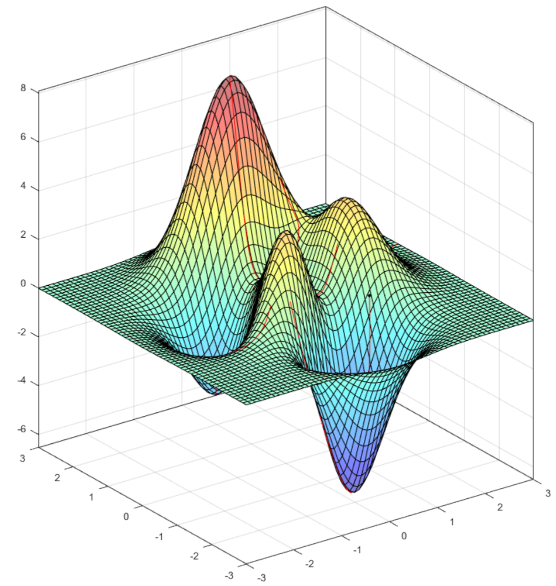
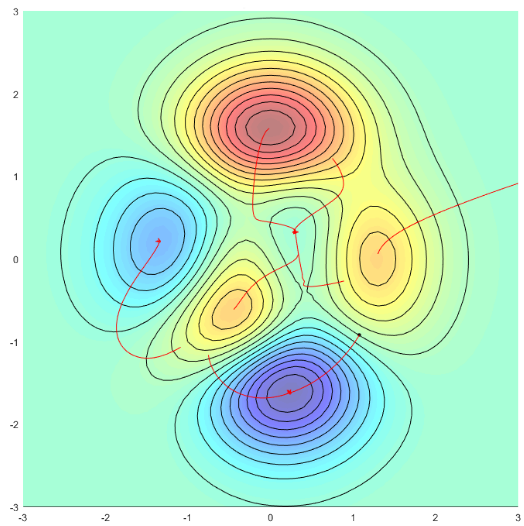
Shape Optimisation with PDE Constraints

Mike Fuller (`fuller@maths.ox.ac.uk`)
Supervised by Dr Alberto Paganini (`paganini@maths.ox.ac.uk`)

12th July 2018





Abstract: Shape optimisation is typically concerned with finding the shape which is optimal in the sense that it minimises a certain cost functional while satisfying given constraints. Often we find that the functional being solved depends on the solution of a PDE constraint.

In this report we briefly discuss unconstrained optimisation methods* before focussing on a shape optimisation test case that, despite its relative simplicity, exhibits an unexpected and interesting behaviour: solving this problem via Newton's method with a *truncated* second shape derivative performs better than using full second order information. We perform numerical (and some theoretical) investigations to try to shed light on this unexpected behaviour.

*One method we will explore is the *Steepest Descent* method; an example using `gradientDescentDemo` from the Machine Learning Toolbox is pictured below the title. MATLAB code by *Roger Jang* and *MIR Lab*, available at `mirlab.org/jang/matlab/toolbox/machineLearning`

Contents

1 Unconstrained Optimisation

We first recap on some iterative methods for solving unconstrained optimisation problems.

1.1 Setting the Scene

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. Suppose we want to find a value x^* such that f is minimal, i.e

$$x^* \in \arg\min_x f(x).$$

We can find such a value by means of an iterative algorithm. That is, given an initial guess $x^{(0)}$ for x^*, we construct a *minimising sequence* $(x^{(k)})$ of points such that

$$f(x^{(k)}) \longrightarrow f(x^*) \quad \text{as } k \to \infty.$$

Often we only require an approximate solution, so we can reduce computational effort by choosing a tolerance $\varepsilon > 0$ such that when $f(x^{(k)}) - f(x^*) \leqslant \varepsilon$, we terminate the algorithm.

1.2 Descent Methods

The minimising sequences we are going to discuss in this section are all of the form

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$$

where

- $\Delta x^{(k)}$ represents a *search direction* - which direction we travel in to approach x^*,
- $t^{(k)} \geqslant 0$ represents a *step length* - how far we travel along the vector $\Delta x^{(k)}$.

Recall that our aim is to minimise f, so we would like the sequence to have the property

$$f(x^{(k+1)}) < f(x^{(k)}) \tag{1}$$

for all k, unless $x^{(k)}$ is optimal. If f is differentiable, we can Taylor expand up to first-order and write

$$f(x^{(k+1)}) = f(x^{(k)} + t^{(k)} \Delta x^{(k)}) \approx f(x^{(k)}) + t^{(k)} \mathrm{d}f(x^{(k)}, \Delta x^{(k)})$$

where $\mathrm{d}f$ denotes the *directional derivative*

$$\mathrm{d}f(x, v) := \lim_{t \to 0} \frac{f(x + tv) - f(x)}{t}.$$

Hence, if $x^{(k)}$ is not optimal (so that $t^{(k)} \neq 0$), we can ensure (1) holds up to first-order if

$$\mathrm{d}f(x^{(k)}, \Delta x^{(k)}) < 0. \tag{2}$$

A search direction $\Delta x^{(k)}$ such that (2) holds is called a *descent direction.*

A generic descent method then goes as follows:

Given starting point x
repeat
  (1) Determine *descent direction*, Δx
  (2) Choose *step size*, $t > 0$
  (3) Set $x \leftarrow x + t\Delta x$
until *stopping criterion satisfied*

At the moment this is rather vague. We have a few questions to answer:

- How do we choose the step size?
- What is the descent direction?
- When do we stop the algorithm?

1.3 Line Search

To answer the first question, we discuss two methods of deciding the step size, called *exact line search (ELS)* and *backtracking line search (BLS)*. Here we assume that Δx is a descent direction.

In ELS, we choose t such that f is minimised along the ray $\{x + t\Delta x : t \geqslant 0\}$, i.e

$$t = \arg\min_{s \geqslant 0} f(x + s\Delta x).$$

It may be difficult to find t exactly, so it is generally cost effective to choose a t value such that f is only approximately minimised along the ray. For this reason we may choose to apply BLS, which generates the step size by the following algorithm:

Algorithm 1: Backtracking Line Search

```
Given descent direction Δx, α ∈ (0, 0.5), β ∈ (0, 1)
t := 1
while f(x + tΔx) > f(x) + αt df(x, Δx) do
    Set t ← βt
end
```

The resulting value of t is chosen as the step size. The line search is called "backtracking" as it starts with unit step size, then reduces it by the factor β until the stopping criterion

$$f(x + t\Delta x) > f(x) + \alpha t \, \mathrm{d}f(x, \Delta x)$$

holds. Since Δx is a descent direction, we have $\mathrm{d}f(x, \Delta x) < 0$, so

$$f(x + t\Delta x) \approx f(x) + t \, \mathrm{d}f(x, \Delta x) < f(x) + \alpha t \, \mathrm{d}f(x, \Delta x)$$

for small enough t, demonstrating that BLS eventually terminates. The choice of parameters α, β has a noticeable but not dramatic effect on convergence (see [1]); ELS can sometimes improve convergence rate but not dramatically. It is generally not worth the computational effort of using ELS.

1.4 Steepest Descent

We can find a descent direction by analysing the Taylor expansion of $f(x + \Delta x)$ about x up to first-order. The approximation is

$$f(x + \Delta x) \approx f(x) + \mathrm{d}f(x, \Delta x)$$

so we find the direction of *steepest descent* when we minimise the directional derivative. We define a *normalised steepest descent direction* Δx_{nsd} as a descent direction of unit length that minimises $\mathrm{d}f(x, \Delta x)$, i.e

$$\Delta x_{\mathrm{nsd}} \in \arg\min_{||\Delta x|| = 1} \mathrm{d}f(x, \Delta x)$$

for some norm $|| \cdot ||$ on $\mathbb{R}^n$.

To piece together the *Steepest Descent* method, it will be useful to introduce the concept of inner products and gradients, particularly when it comes to defining stopping criteria. Let V be a real vector space. An *inner product* $\langle \cdot, \cdot \rangle : V^2 \to \mathbb{R}$ is a function that satisfies the following properties:

- *Linearity:* $\langle \alpha u + \beta v, w \rangle = \alpha \langle u, w \rangle + \beta \langle v, w \rangle \quad \forall u, v, w \in V,\ \alpha, \beta \in \mathbb{R}$
- *Symmetry:* $\langle u, v \rangle = \langle v, u \rangle \quad \forall u, v \in V$
- *Positive definitenesss:* $\langle u, u \rangle \geqslant 0 \quad \forall u \in V, \quad \langle u, u \rangle = 0 \iff u = 0.$

It follows from the first two properties that $\langle \cdot, \cdot \rangle$ is *bilinear.*

We call $\nabla f(x)$ the *gradient* of f at x with respect to an inner product $\langle \cdot, \cdot \rangle$ if

$$\langle \nabla f(x), v \rangle = \mathrm{d} f(x, v) \quad \forall v.$$

We occasionally write $\nabla_{\langle \cdot, \cdot \rangle} f(x)$ to make it clear what the associated inner product is.

Lemma 1.1: If a norm $|| \cdot ||$ is induced by an inner product $\langle \cdot, \cdot \rangle$, i.e $||v|| = \sqrt{\langle v, v \rangle}$, then

$$\Delta x_{\text{nsd}} = -\frac{\nabla f(x)}{||\nabla f(x)||}. \tag{3}$$

Proof: We use the method of Lagrange multipliers. Let

$$L(\Delta x, \lambda) := \mathrm{d} f(x, \Delta x) + \frac{\lambda}{2}(||\Delta x||^2 - 1).$$

Then by definition of Δx_{nsd},

$$\begin{aligned} & \partial_{\Delta x} L(\Delta x_{\text{nsd}}, v) = \mathrm{d} f(x, v) + \lambda \langle \Delta x_{\text{nsd}}, v \rangle = 0 \quad \forall v \\ \iff \quad & \langle \lambda \Delta x_{\text{nsd}}, v \rangle = -\mathrm{d} f(x, v) \quad \forall v \\ \iff \quad & \lambda \Delta x_{\text{nsd}} = -\nabla f(x). \end{aligned}$$

Taking norms of both sides, and using the fact that $||\Delta x_{\text{nsd}}|| = 1$, we obtain $|\lambda| = ||\nabla f(x)||$. The result follows, using the fact that Δx_{nsd} is a steepest descent direction.

□

Geometrically speaking, we see that Δx_{nsd} is a step in the unit ball of $|| \cdot ||$ which extends farthest in the direction of $-\nabla f(x)$. It is more convenient in practice to use an unnormalised descent direction

$$\Delta x_{\text{sd}} := ||\nabla f(x)||_* \Delta x_{\text{nsd}}$$

where $|| \cdot ||_*$ denotes the *dual norm*

$$||v||_* := \sup_{||x|| \leqslant 1} v^T x.$$

We find that Δx_{sd} is simply the unnormalised version of (3), that is

$$\Delta x_{\text{sd}} = -\nabla_{\langle \cdot, \cdot \rangle} f(x)$$

and we arrive at the *Steepest Descent* method, which uses Δx_{sd} as a descent direction:

Algorithm 2: Steepest Descent

```
1 Given starting point x
2 repeat
3    (1) Set Δx ← Δx_sd
4    (2) Choose t via ELS/BLS
5    (3) Set x ← x + tΔx
6 until stopping criterion satisfied
```

To define a stopping criterion, recall that when x^* is optimal we have

$$\mathrm{d}f(x^*, v) = 0 \quad \forall v$$

so for a sufficiently smooth f, we expect the gradient to be small for $x \approx x^*$. For this reason, we can define a stopping criterion of the form

$$||\nabla f(x)|| \leqslant \varepsilon$$

for some chosen small $\varepsilon > 0$. Most practical uses of Steepest Descent check the stopping criterion after the first step.

1.5 Newton's Method

Which inner product do we use to define Δx_{sd}? An interesting choice, which forms the basis of *Newton's Method*, is to consider the inner product induced by the second derivative

$$\mathrm{d}^2 f(x, u, v) := \lim_{t \to 0} \frac{\mathrm{d}f(x + tv, u) - \mathrm{d}f(x, u)}{t}.$$

If it exists, we can extend our Taylor approximation up to second-order to get

$$f(x + \Delta x) \approx f(x) + \mathrm{d}f(x, \Delta x) + \frac{1}{2}\mathrm{d}^2 f(x, \Delta x, \Delta x) =: g(\Delta x).$$

If $\mathrm{d}^2 f(x, \cdot, \cdot)$ is positive definite then g is *strictly convex*[†], thus has a unique minimiser Δx_{nt}, called the *Newton step*. Since the second derivative exists, $\mathrm{d}^2 f(x, \cdot, \cdot)$ is also symmetric and linear, hence an inner product. By minimality, the Newton step must satisfy

$$\begin{aligned}
& \mathrm{d}g(\Delta x_{\mathrm{nt}}, v) = 0 \quad \forall v \\
\iff \quad & \mathrm{d}f(x, v) + \mathrm{d}^2(x, \Delta x_{\mathrm{nt}}, v) = 0 \quad \forall v \\
\iff \quad & \mathrm{d}^2(x, \Delta x_{\mathrm{nt}}, v) = -\mathrm{d}f(x, v) \quad \forall v
\end{aligned}$$

so Δx_{nt} is the negative gradient with respect to the inner product induced by $\mathrm{d}^2 f$, that is,

$$\Delta x_{\mathrm{nt}} = -\nabla_{\mathrm{d}^2 f(x, \cdot, \cdot)} f(x).$$

If f is quadratic, then $f(\Delta x_{\mathrm{nt}}) = g(\Delta x)$ exactly, so $x + \Delta x_{\mathrm{nt}}$ is the *exact* minimiser of f. Hence it should also be a very good estimate for x^* if f is approximately quadratic. Since f is assumed to be twice continuously differentiable, the quadratic model of f is very accurate for x near x^*, so $x + \Delta x_{\mathrm{nt}}$ is a very good estimate for x^*.

Before we introduce *Newton's method*, we construct a stopping criterion using the *Newton decrement*

$$\lambda(x) := \sqrt{\mathrm{d}f(x, \Delta x_{\mathrm{nt}})}.$$

Lemma 1.2: $\lambda(x)^2/2$ estimates the error $f(x) - f(x^*)$, based on the quadratic approximation of f.

Proof: The error based on the quadratic approximation is

$$\begin{aligned}
f(x) - \inf_y g(y) = \; f(x) - g(\Delta x_{\mathrm{nt}}) &= f(x) - \left(f(x) + \cancel{\mathrm{d}f(x, \Delta x_{\mathrm{nt}})} + \frac{1}{2}\mathrm{d}^2 f(x, \Delta x_{\mathrm{nt}}, \Delta x_{\mathrm{nt}}) \right) \\
&= f(x) - \left(f(x) - \frac{1}{2}\mathrm{d}f(x, \Delta x_{\mathrm{nt}}) \right) \\
&= \frac{1}{2}\lambda(x)^2.
\end{aligned}$$

□

[†]A function g is said to be *strictly convex* if the line segment connecting any two distinct points on the surface of g lies strictly above g, except at the endpoints.

We use this as a stopping criterion for Newton's Method, which goes as follows:

Algorithm 3: Newton's Method

Given starting point x, tolerance $\varepsilon > 0$
repeat
  (1) Set $\Delta x \leftarrow \Delta x_{\mathrm{nt}}$
  (2) Set $\lambda \leftarrow \lambda(x)$
  (3) **Stop** if $\lambda^2/2 \leqslant \varepsilon$
  (4) Choose t via BLS
  (5) Set $x \leftarrow x + t\Delta x$

Observe that this is more or less a general descent method, with the difference that the stopping criterion is checked after computing Δx_{nt}, rather than after updating the value of x.

Newton's Method has *quadratic convergence* near x^* (see [1]). Roughly speaking, this means that the number of correct digits doubles after each iteration. Newton's Method also scales with the size of the problem: its performance in $\mathbb{R}^{10000}$ is similar to problems in $\mathbb{R}^{10}$ say, with only a modest increase in the number of iterations.

A pitfall of Newton's Method is the cost of computing Δx_{nt}, which requires solving a system of linear equations. *Quasi-Newton* methods require less cost to form the search direction, sharing some advantages of Newton's Method such as rapid convergence near x^*, but we will not discuss such methods here.

2 Shape Differentiation

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. Provided f is smooth, we can differentiate to find its minimum values; in this section we extend the notion of differentiation of functions to *shape differentiation* of domains, with the aim of using these shape derivatives to find the domain Ω^* in a collection of admissible shapes $\mathcal{U}_{\text{ad}}$ which minimises the cost functional $\mathcal{J}[\Omega] := \int_\Omega f \,\mathrm{d}x$. That is,

$$\Omega^* \in \operatorname*{arg\,min}_{\Omega \in \mathcal{U}_{\text{ad}}} \mathcal{J}[\Omega].$$

2.1 Defining the Shape Derivative

Let $T : \mathbb{R}^n \to \mathbb{R}^n$ be a sufficiently smooth vector field. Then

$$\mathcal{J}[T(\Omega)] = \int_{T(\Omega)} f \,\mathrm{d}x = \int_\Omega (f \circ T) |\det DT| \,\mathrm{d}x$$

where DT denotes the *Jacobian* of T. Note that $T(\Omega) \neq \Omega$ in general, and similarly $\mathcal{J}[T(\Omega)] \neq \mathcal{J}[\Omega]$.

Furthermore let $V : \mathbb{R}^n \to \mathbb{R}^n$. We adopt the notation

$$\left\langle \frac{\mathrm{d}}{\mathrm{d}T} g[T], V \right\rangle := \lim_{t \to 0} \frac{g[T + tV] - g[T]}{t}$$

noting that this is different to the inner product notation used in Section 1.4. We then define the *shape derivative* of $\mathcal{J}$ at Ω in the direction V as

$$\mathrm{d}\mathcal{J}(\Omega, V) := \left\langle \frac{\mathrm{d}}{\mathrm{d}T} \mathcal{J}[T(\Omega)], V \right\rangle \bigg|_{T=I}$$

where I is the identity map. Therefore,

$$\begin{aligned}
\mathrm{d}\mathcal{J}(\Omega, V) &= \left\langle \frac{\mathrm{d}}{\mathrm{d}T} \left(\int_\Omega (f \circ T) |\det DT| \,\mathrm{d}x \right), V \right\rangle \bigg|_{T=I} \\
&= \int_\Omega \left\langle \frac{\mathrm{d}}{\mathrm{d}T} ((f \circ T) \det DT), V \right\rangle \bigg|_{T=I} \mathrm{d}x.
\end{aligned}$$

2.2 Shape Derivative Identities

We prove two identities of the shape derivative using the above.

Theorem 2.1: Let $\nabla \cdot V$ denote the *divergence* of V. Then

$$\mathrm{d}\mathcal{J}(\Omega, V) = \int_\Omega \nabla f \cdot V + f \nabla \cdot V \,\mathrm{d}x.$$

Proof: By the product rule,

$$\mathrm{d}\mathcal{J}(\Omega, V) = \int_\Omega \left\{ \left\langle \frac{\mathrm{d}}{\mathrm{d}T} (f \circ T), V \right\rangle \det DT + \left\langle \frac{\mathrm{d}}{\mathrm{d}T} \det DT, V \right\rangle (f \circ T) \right\} \bigg|_{T=I} \mathrm{d}x.$$

From the first term, we have

$$\left\langle \frac{\mathrm{d}}{\mathrm{d}T} (f \circ T), V \right\rangle = \frac{\mathrm{d}}{\mathrm{d}t} (f \circ (T + tV)) = \nabla (f \circ T) \cdot V$$

so that

$$\left\langle \frac{\mathrm{d}}{\mathrm{d}T} (f \circ T), V \right\rangle \det DT \bigg|_{T=I} = \nabla f \cdot V.$$

From the second term, using Jacobi's formula we have

$$\begin{aligned}\left\langle \frac{\mathrm{d}}{\mathrm{d}T} \det DT, V \right\rangle &= \lim_{t\to 0} \frac{\det D(T+tV) - \det DT}{t} \\ &= \lim_{t\to 0} \frac{\det(DT + tDV) - \det DT}{t} \\ &= \lim_{t\to 0} \frac{(\det DT + t\,\mathrm{Tr}(\mathrm{adj}(DT)DV) + O(t^2)) - \det DT}{t} \\ &= \mathrm{Tr}(\mathrm{adj}(DT)DV)\end{aligned}$$

where $\mathrm{Tr}(A), \mathrm{adj}(A)$ denote the *trace* and *adjugate* of A respectively. Hence

$$\left\langle \frac{\mathrm{d}}{\mathrm{d}T} \det DT, V \right\rangle (f \circ T)\Big|_{T=I} = \mathrm{Tr}(DV)f = f\nabla \cdot V$$

and the result follows. □

Theorem 2.2: We have the identity

$$\mathrm{d}\mathcal{J}(\Omega, V) = \int_{\partial\Omega} f(V \cdot n)\,\mathrm{d}S$$

where n is the outward pointing unit normal of the boundary $\partial\Omega$.

Proof: Using the Divergence Theorem,

$$\begin{aligned}\mathrm{d}\mathcal{J}(\Omega, V) &= \int_\Omega \nabla f \cdot V + f\nabla \cdot V\,\mathrm{d}x \\ &= \int_\Omega \nabla \cdot (Vf)\,\mathrm{d}x \\ &= \int_{\partial\Omega} (Vf) \cdot n\,\mathrm{d}S \\ &= \int_{\partial\Omega} f(V \cdot n)\,\mathrm{d}S\end{aligned}$$

where the second equality follows from vector calculus. □

3 Shape Optimisation Test Case

4 Conclusion

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimisation*. Cambridge University Press, 2004. web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf