Derivative Free Optimization

Tom Savage

Introduction

- In many problems in science, engineering and AI, objective and constraint functions are available only as the output of a black-box, not providing derivative information,
- Thus necessitating the use of zeroth-order or derivative-free methods.

What this talk isn't...

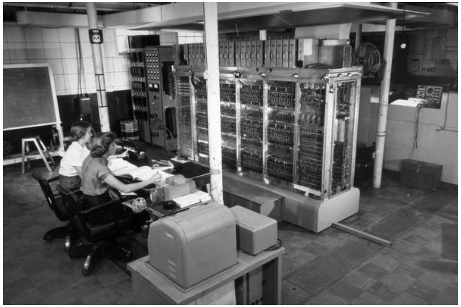
There is no mention of genetic algorithms in this paper, and generally the field of derivative free optimisation ignores these algorithms...**why?**

- Lots of DFO originate from the field of applied mathematics. Stochastic search methods are very difficult to analyse, prove, etc... Therefore in a mathematician's eyes they're worth studying less.

- There doe seem to be a bit of a divide here and it's an interesting talking point.

Introduction

One of the earliest calculations on MANIC, an early computer was the approximate solution of a six-dimensional non-linear least squares problem using a derivative free coordinate search. Today they are used routinely in automation and tuning for machine learning



Motivation

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \underbrace{\in \Omega \subseteq \mathbb{R}^n}_{\text{Constraint Set}} \end{aligned}$$

Motivation

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \underbrace{\in \Omega \subseteq \mathbb{R}^n}_{\text{Constraint Set}} \end{aligned}$$

- The setting is such that we assume we can't take the derivative of $f(\mathbf{x})$ (or we don't know it).

- This doesn't mean we can't use derivatives in our algorithm. For example if we decide to approximate $f(\mathbf{x})$.

Direct Search Methods

Before you ask...

$$\nabla f(\mathbf{x}) \approx \frac{f(\mathbf{x}+h) - f(\mathbf{x})}{h}$$

A direct-search method is a method that uses only function values and ‘does not “in its heart” develop an approximate gradient’.

Direct Search Methods

Before you ask...

$$\nabla f(\mathbf{x}) \approx \frac{f(\mathbf{x}+h) - f(\mathbf{x})}{h}$$

> *A direct-search method is a method that uses only function values and 'does not "in its heart" develop an approximate gradient'.*

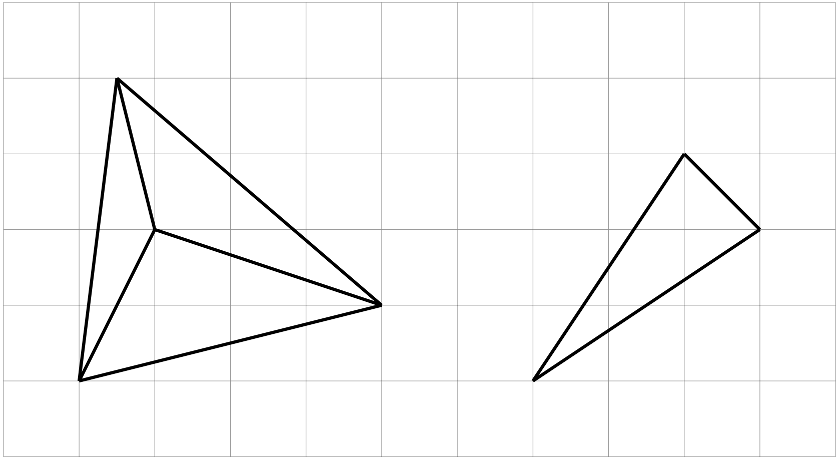
- At their core, direct-search methods use comparisons of function values to choose where to move in search space. Though later on this changes

Simplex Methods

- Distinct from the Linear Programming simplex method, the most common derivative free simplex method is also known as the *Nelder-Mead* method.

- First define a *simplex* as a collection of $n + 1$ linearly independent points in $\mathbb{R}^n$.

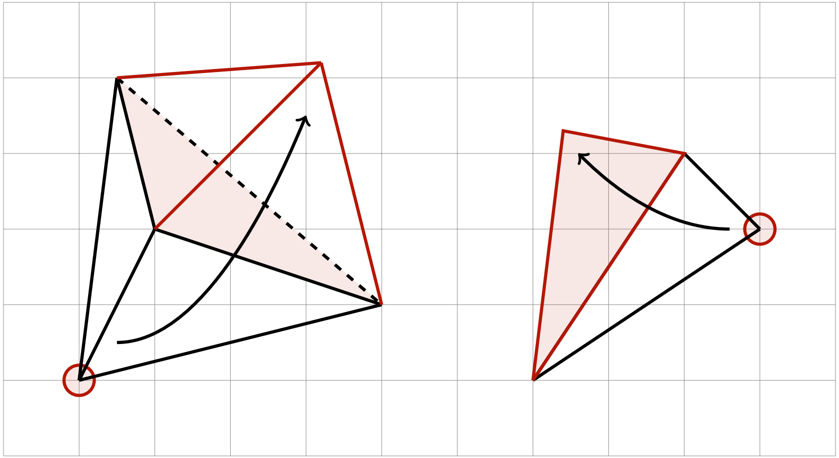
Some Simplexes



Simplex Algorithm

1. Evaluate the function at each simplex vertex.
2. Choose the worst point and reflect it in the plane of other vertices.

Simplex Iteration - Reflection

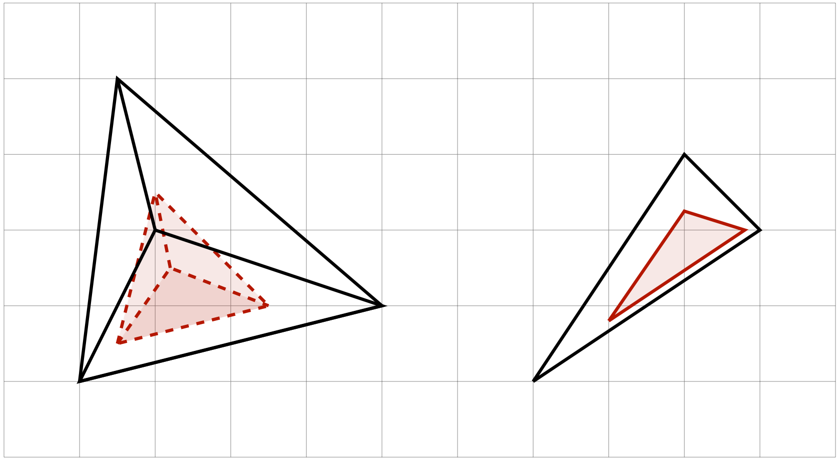


Simplex Algorithm

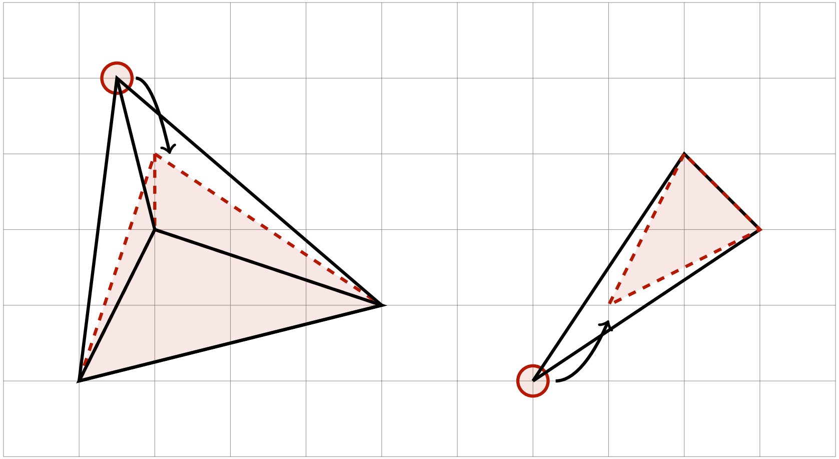
- In addition, operations such as *expand* and *contract*, and *shrink* allow the simplex to distort to account for possible curvature.

- Method is used in MATLAB’s *fminsearch* search function and featured in *Numerical Recipes*, cited over 125000 times.

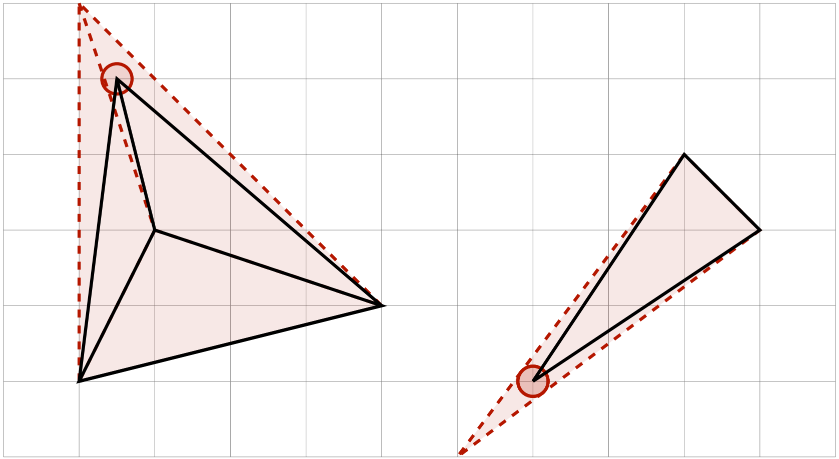
Simplex Iterations - Shrink



Simplex Iteration - Contraction

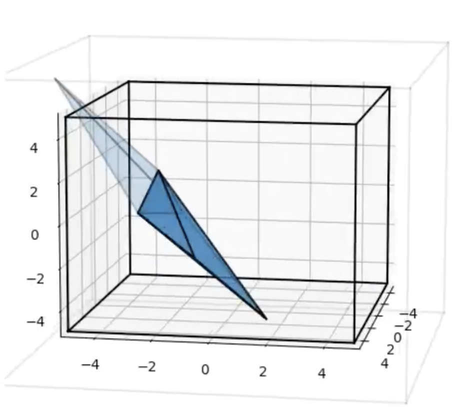


Simplex Iteration - Expansion



Example

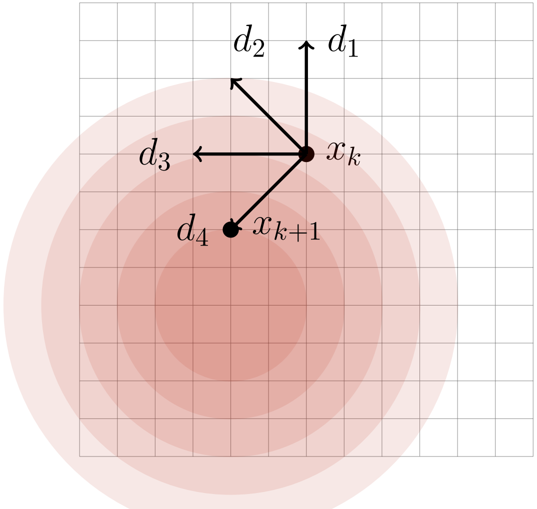
What does this look like... (click me)



GIF

Directional direct-search methods

- Hypothesise some directions $d_i \in D_k$ and set a step-size α
- Check steps, and if a significant decrease occurs, take the step
- If no decrease occurs then decrease α

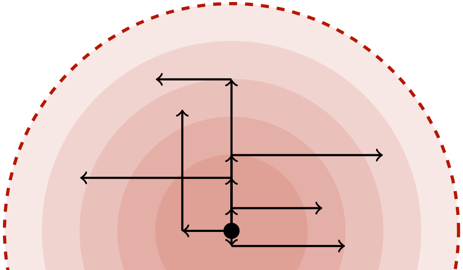


Directional direct-search methods

Typical results to prove the convergence of directional search methods require:

$$\mathbf{x} \in \Omega \left(\mathbf{x} = \sum_{i=1}^{|D_k|} \lambda_i d_i \right)$$

i.e.



Extensions

- Other extensions include *Mesh Adaptive Direct Search (MADS)*, which effectively constructs a ‘mesh’ of possible directions d_j.

- By parameterising this mesh, its resolution can be increased, getting around the issue in pattern-search methods of a finite number of search directions. By taking the set of directions $\mathbf{D}$ from ‘nodes’ in this mesh.

Worst Case Complexity Analysis

- Generally, refers to an upper bound on the number of function evaluations N_ϵ required to attain an ϵ-optimal solution.

- First-order ϵ accuracy: $||\nabla f(\mathbf{x}_k)|| \leq \epsilon$

- Second-order ϵ accuracy: $\max\{||\nabla f(\mathbf{x}_k)||, -\lambda_k\} \leq \epsilon$, where λ_k denotes the minimum eigenvalue of $\nabla^2 f(\mathbf{x}_k)$

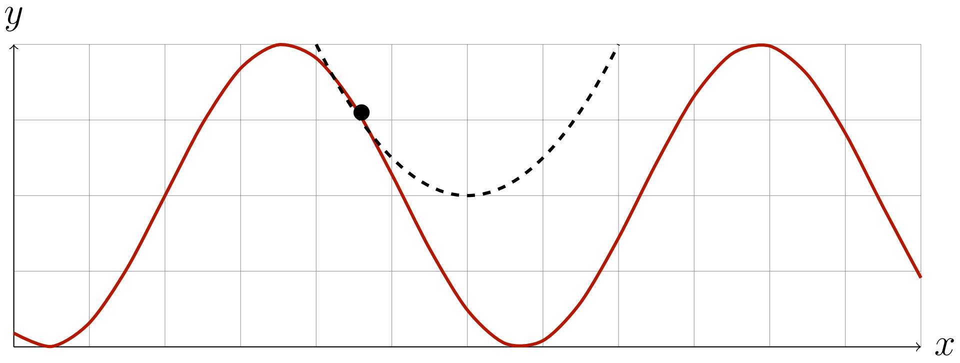
Model-based methods

- “Updates based primarily on the predictions of a model that serves as a surrogate of the objective function or of a related merit function”

- For these methods we assume smoothness of f and thus operate with smooth models.

- Later on we will exploit knowledge of non-smoothness.

Situation



Polynomial Models

- The most-commonly used model for DFO.
- $\mathcal{P}^{d,n}$ is the space of polynomials over n variables and degree d. Also have a basis ϕ.
- For example: quadratic models have the basis: $\phi(\mathbf{x}) = [1, x_1, x_2, \ldots, x_n, x_1^2, ..., x_n^2, x_1 x_2, ..., x_{n-1} x_n]^T$

$$m(\mathbf{x}) = \sum_{i=1}^{\text{dim}(\mathcal{P}^{d,n})} a_i \phi_i(\mathbf{x})$$

.

Polynomial Models

Given a set of p points $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_p\}$, we construct a model by solving:

$$\Phi(\mathbf{Y})\mathbf{a} = \begin{bmatrix} f(\mathbf{y}_1) \\ \vdots \\ f(\mathbf{y}_p) \end{bmatrix}$$

for $\mathbf{a}$. Where $\Phi(\mathbf{Y}) \in \mathbb{R}^{p \times \text{dim}\mathcal{P}^{d,n}}$.

- For this we require that points $\mathbf{Y}$ are affinely independent.

Quadratic interpolation models

- Quality is determined by the position of the underlying points being interpolated. For example is a model m approximates f using points far away from the region of interest.

- We of course require function values $\{f(\mathbf{y}_i) : \mathbf{y}_i \in \mathbf{Y}\}$, and when f is computationally expensive to evaluate, the $(n+1)(n+2)/2$ points required can be a burden in order to solve for $\mathbf{a}$.

What if we want to use less points?

Resolve the remaining $(n+1)(n+2)/2$ degrees of freedom by choosing the model parameters such that:

$$\min_{m \in \mathcal{P}^{2,n}} ||\nabla^2 m(\widehat{\mathbf{x}}) - \mathbf{H}||_F^2$$
$$\text{s.t.} \quad m(\mathbf{y}_i) = f(\mathbf{y}_i), \quad \mathbf{y}_i \in \mathbf{Y}$$

i.e. such that the hessian of our quadratic model is closest to a specified hessian $\mathbf{H}$ in the region of interest $\hat{x}$.
Existence: at least $n+1$ points in $\mathbf{Y}$ are affinely independent.

What if $p > \dim \mathcal{P}^{d,n}$?

System becomes over-determined and we now have to formulate a least-squares problem instead of solving for $\mathbf{a}$ directly.

- Primary reason for low-degree polynomial models are they are easier to optimise than higher degree.
- Methods to solve the sub-problem (minimise the quadratic model) are a key concern and often seek to perform as view operations as possible.

Radial basis function interpolation models

An additional way to include non-linearity in an approximate model with the form:

$$m(\mathbf{x}) = \sum_{i=1}^{|\mathbf{Y}|} b_i \psi(||\mathbf{x} - \mathbf{y}_i||) + \underbrace{\mathbf{a}^T \phi(\mathbf{x})}_{\text{low-order polynomial}}$$

In other words, a model evaluation consists of a weighted sum of a function that is dependant on the distance from each data point, as well as an optional polynomial model.
An example basis function is $\psi(r) = \exp -\gamma^{-2} r^2$.

Constructing an RBF model

$$\begin{bmatrix} \psi(||\mathbf{y}_1 - \mathbf{y}_1||) & \cdots & \psi(||\mathbf{y}_1 - \mathbf{y}_{|\mathbf{Y}|}||) & \phi(\mathbf{y}_1)^T \\ \vdots & & \vdots & \vdots \\ \psi(||\mathbf{y}_{|\mathbf{Y}|} - \mathbf{y}_1||) & \cdots & \psi(||\mathbf{y}_1 - \mathbf{y}_{|\mathbf{Y}|}||) & \phi(\mathbf{y}_{|\mathbf{Y}|})^T \\ \phi(\mathbf{y}_1) & \cdots & \phi(\mathbf{y}_{|\mathbf{Y}|}) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} f(\mathbf{y}_1) \\ \vdots \\ f(\mathbf{y}_{|\mathbf{Y}|}) \\ \mathbf{0} \end{bmatrix}$$

Using least squares, solve for $\mathbf{a}$ and $\mathbf{b}$.

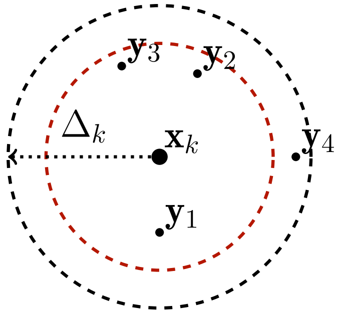
Trust-region methods

Distinguishing characteristic of derivative-free model-based trust-region methods is how they manage $\mathbf{Y_k}$, the set of points used to construct m_k the model. For example:

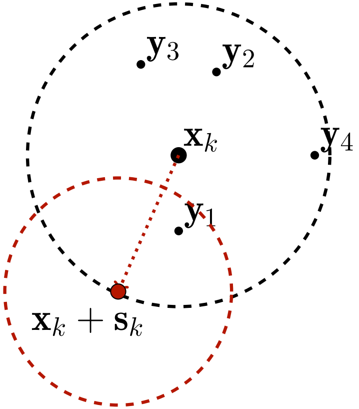
- Do we have a **stencil** of points around $\mathbf{x}_k$?
- Do we spend computational time figuring out where to best improve our model?
- If we do, are we likely to reach the optimum of the overall problem faster?

Derivative-free trust-region algorithm

1. Select a subset of $\mathbf{Y}_k$ or augment $\mathbf{Y}_k$ and build a model m_k using their function values.
2. If the model is accurate within the trust region characterised by radius Δ_k, $\Delta_k \leftarrow \gamma_{dec}\Delta_k$.
3. Else, update $\mathbf{Y}_k$ with new points to make m_k accurate within the trust region.



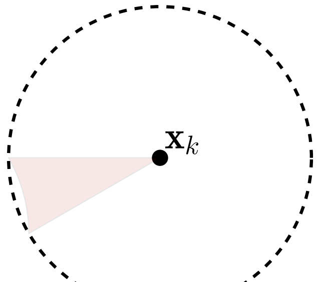
4. Generate a direction $\mathbf{s}_k$ that minimises m_k with a step size Δ_k, i.e. on the edge of the trust region.
5. $\rho_k \leftarrow \frac{f(\mathbf{x}_k)-f(\mathbf{x}_k+\mathbf{s}_k)}{m_k(\mathbf{x}_k)-m_k(\mathbf{x}_k+\mathbf{s}_k)}$
6. If ρ_k is less than a value, add a model improving point $\mathbf{Y}_k$ and increase the trust region size, otherwise reduce the size of the trust region $\Delta_k \leftarrow \gamma_{dec}\Delta_k$.
7. Then choose to either take the step or not based on $f(\mathbf{x}_{k+1})$.



Other classes of trust-region method

- **Wedge methods** add constraints to the direction of search within a trust region.

- If derivatives can be obtained but are expensive, then an approximate model can be constructed of the first-order information.



Hybrid methods and miscellanea

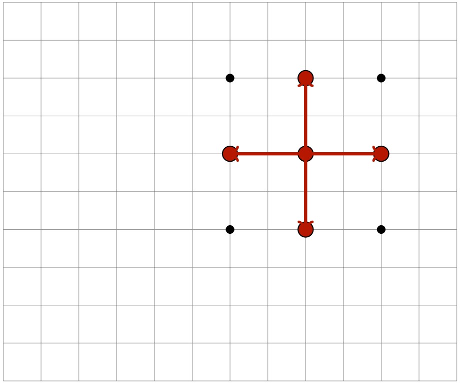
Most of derivative free method can be classified within direct-search or model-based methods, but some fall outside these catagories:

- Method proposes minimising a quadratic model in order to locate a new simplex vertex over a trust region.

- Effectively any combination of the previous algorithms.

Implicit Filtering

1. Evaluate the function at each point surrounding $\mathbf{x}_k$ on a grid.
2. If there is no decrease, shrink the stencil, if there is a decrease...
3. Calculate gradient, Hessian estimate (BFGS).
4. Take Newton step, repeat.



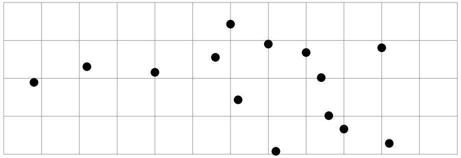
Randomized methods for deterministic objectives

These methods have promising theoretical properties but practitioners may be put off by non-deterministic behaviour. You may know a few… Particle Swarm, Genetic Algorithms.
Can we think of possibly the simplest algorithm of all?

Random Search

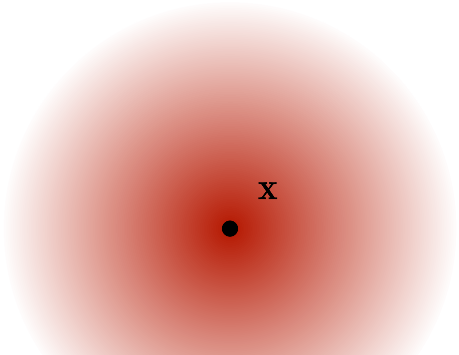
Easy to implement and exhibits perfect scaling (more points evaluated $\rightarrow$ lower function value.

1. Initial location $\widehat{\mathbf{x}}$.
2. Generate $\mathbf{x} \in \Omega$.
3. if $f(\mathbf{x}) < f(\widehat{\mathbf{x}})$ then $\widehat{\mathbf{x}} \leftarrow \mathbf{x}$
4. Repeat.



Nesterov Random Search

$$f_\mu(\mathbf{x}) = \underbrace{\sqrt{\frac{\det(\mathbf{B})}{(2\pi)^n}}}_{\text{Scaling term}} \underbrace{\int_{\mathbb{R}^n} f(\mathbf{x} + \mu\mathbf{u}) \exp\left(-\frac{1}{2}\mathbf{u}^T\mathbf{B}\mathbf{u}\right) \mathbf{du}}_{\text{Gaussian smoothed evaluation}}$$



Nesterov Random Search

$$\nabla f_\mu(\mathbf{x}) = \frac{1}{\mu}\sqrt{\frac{\det(\mathbf{B})}{(2\pi)^n}} \int_{\mathbb{R}^n} \overbrace{(f(x+\mu\mathbf{u}) - f(\mathbf{x}))}^{\text{finite-difference}} \exp\left(-\frac{1}{2}\mathbf{u}^T\mathbf{B}\mathbf{u}\right) \mathbf{B}\mathbf{u}d\mathbf{u}$$

In other words: $\nabla f_\mu(\mathbf{x})$ is an approximation of $\nabla f(\mathbf{x})$ (taking the expectation of the finite difference) in the smooth case.

Nesterov Random Search

Instead of computing the integral, or even trying many values of $\mathbf{u}$ we could approximate the gradient with the *directional derivative*, i.e. just choose one value of $\mathbf{u}$.

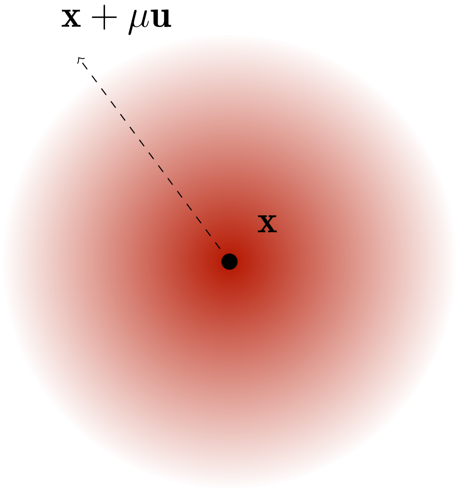
$$g_\mu(\mathbf{x}) = \frac{f(\mathbf{x} + \mu\mathbf{u}) - f(\mathbf{x})}{\mu}\mathbf{B}\mathbf{u}$$

$$\hat{g}_\mu(\mathbf{x}) = \frac{f(\mathbf{x} + \mu\mathbf{u}) - f(\mathbf{x} - \mu\mathbf{u})}{2\mu}\mathbf{B}\mathbf{u}$$

Nesterov Random Search

$$g_\mu(\mathbf{x}) = \frac{f(\mathbf{x} + \mu\mathbf{u}) - f(\mathbf{x})}{\mu}\mathbf{B}\mathbf{u}$$

Directional derivative returns a scalar, so we multiply by the matrix $\mathbf{B}$ and random vector $\mathbf{u}$ to approximate the smooth gradient.



Nesterov Random Search - Algorithm

For $k = 0, 1, 2, \ldots$

1. At $\mathbf{x}_k$.
2. Generate $\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{B}^{-1})$; compute $\mathbf{g}(\mathbf{x}_k; \mathbf{u}_k)$.
3. $\mathbf{x}_{k+1} \leftarrow$ proj$(\mathbf{x}_k - \alpha_k \mathbf{B}^{-1} \mathbf{g}(\mathbf{x}_k; \mathbf{u}_k), \Omega)$

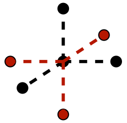
An important result for this algorithm is that is was perhaps the first WCC result where f may be both non-smooth and non-convex.
With the essence taken from computing the expectation of Gaussian distributions $\mathbb{E}_{\mathbf{U}_{k-1}}[||\nabla f_{\hat{\mu}}(\mathbf{x}_k)||] \leq \epsilon$

Methods for convex objectives

As with derivative based optimisation, convexity can be exploited. Typically for a convex method one demonstrates that

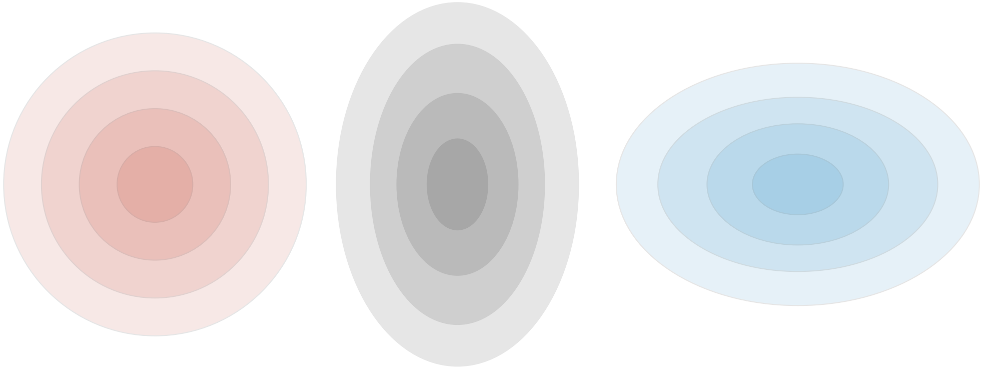
$$\lim_{k\to\infty} f(\mathbf{x}_k) - f(\mathbf{x}^*) = 0$$

where $\mathbf{x}^*$ is a global minimiser. Generally convex derivative free methods should and do match WCC for derivative based counterparts, scaled by n. As with n evaluations one can approximate the gradient.



Methods for convex-stochastic optimisation

Here we assume $\bar{f}(\mathbf{x};\xi)$ is convex in $\mathbf{x}$ for every ξ (a stochastic variable).



Methods for convex-stochastic optimisation

- We can think of this problem as a gambler pulling the arms of slot machines, each with a different win percentage. What is the gamblers best strategy?

- **Except...** there are an infinite number of slot machines!

- One for every place in our search space

Provided that $\mathbb{E}_\xi[\bar{f}(x,\xi)] = f(x)$, the gambler's best long-run strategy is to constantly play $x_* \in$ argmin$f(x)$

WCC Proof

We've mentioned WCC as a metric earlier, the majority of these methods have and must be presented with a WCC to be taken seriously...lets do one. We'll prove

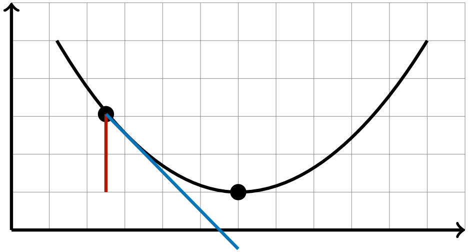
$$\mathbb{E}\left[\sum_{t=1}^{n} f(\mathbf{x}_t, \xi_t)\right] - \min_{\mathbf{x}} \sum_{t=1}^{n} f(\mathbf{x}, \xi_t) \leq RG\sqrt{n}$$

for an expected gradient descent algorithm, where we have a sequence of n function evaluations (all convex), G is some error ($||g_t|| \leq G$)

Expected Gradient Descent WCC

Let $\mathbf{x}_*$ be a point that minimises $\sum_{t=1}^{n} f(\mathbf{x}, \xi_t)$. f_t is a convex function appearing in the t^{th} evaluation. Since f is convex, the difference between $f_t(\mathbf{x}_t)$ and $f_t(\mathbf{x}_*)$ can be bounded by the gradient

$$f_t(\mathbf{x_t}) - f_t(\mathbf{x}_*) \leq \nabla f_t(\mathbf{x}_t) \cdot (\mathbf{x}_t - \mathbf{x}_*)$$



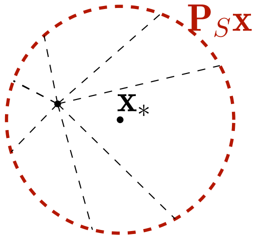
$$\begin{aligned} f_t(\mathbf{x_t}) - f_t(\mathbf{x}_*) \leq & \nabla f_t(\mathbf{x}_t) \cdot (\mathbf{x}_t - \mathbf{x}_*) \\ & \mathbb{E}[\mathbf{g}_t | \mathbf{x}_t] \cdot (\mathbf{x}_t - \mathbf{x}_*) \\ & \mathbb{E}[\mathbf{g}_t \cdot (\mathbf{x}_t - \mathbf{x}_*) | \mathbf{x}_t] \end{aligned}$$

Taking the expectation on both sides…

$$\mathbb{E}[f_t(\mathbf{x_t}) - f_t(\mathbf{x}_*)] \leq \mathbb{E}[\mathbf{g}_t \cdot (\mathbf{x}_t - \mathbf{x}_*)]$$

Considering the following inequality which is true for convex functions

$$||\mathbf{P}_S(\mathbf{x}) - \mathbf{x}_*|| \leq ||\mathbf{x} - \mathbf{x}_*||$$



Considering $||\mathbf{x}_t - \mathbf{x}_*||^2$ as a metric we wish to go to zero.

Considering $\mathbf{x}$ can be expressed as $\mathbf{x}_t - \eta\mathbf{g}_t$

$$\begin{aligned}
||\mathbf{x}_{t+1} - \mathbf{x}_*||^2 &= ||\mathbf{P}_S(\mathbf{x}_t - \eta\mathbf{g}_t) - \mathbf{x}_*||^2 \\
&\leq ||\mathbf{x}_t - \eta\mathbf{g}_t - \mathbf{x}_*||^2 \\
&= ||\mathbf{x}_t - \mathbf{x}_*||^2 + \eta^2||\mathbf{g}_t||^2 - 2\eta(\mathbf{x}_t - \mathbf{x}_*) \cdot \mathbf{g}_t \\
&\leq ||\mathbf{x}_t - \mathbf{x}_*||^2 + \eta^2 G^2 - 2\eta(\mathbf{x}_t - \mathbf{x}_*) \cdot \mathbf{g}_t \\
\mathbf{g}_t \cdot (\mathbf{x}_t - \mathbf{x}_*) &\leq \frac{||\mathbf{x}_t - \mathbf{x}_*||^2 - ||\mathbf{x}_{t+1} - \mathbf{x}_*||^2 + \eta^2 G^2}{2\eta}
\end{aligned}$$

$$\mathbf{g}_t \cdot (\mathbf{x}_t - \mathbf{x}_*) \leq \frac{||\mathbf{x}_t - \mathbf{x}_*||^2 - ||\mathbf{x}_{t+1} - \mathbf{x}_*||^2 + \eta^2 G^2}{2\eta}$$

$$\mathbb{E}[f_t(\mathbf{x_t}) - f_t(\mathbf{x}_*)] \leq \mathbb{E}[\mathbf{g}_t \cdot (\mathbf{x}_t - \mathbf{x}_*)]$$

$$\mathbb{E}[f_t(\mathbf{x_t}) - f_t(\mathbf{x}_*)] \leq \mathbb{E}\left[\frac{||\mathbf{x}_t - \mathbf{x}_*||^2 - ||\mathbf{x}_{t+1} - \mathbf{x}_*||^2 + \eta^2 G^2}{2\eta}\right]$$

$$\mathbb{E}[f_t(\mathbf{x_t}) - f_t(\mathbf{x}_*)] \leq \mathbb{E}\left[\frac{||\mathbf{x}_1 - \mathbf{x}_*||^2}{2\eta} + n\frac{\eta^2 G^2}{2\eta}\right]$$

$$\mathbb{E}[f_t(\mathbf{x_t}) - f_t(\mathbf{x}_*)] \leq \frac{R^2}{2\eta} + n\frac{\eta G^2}{2}$$

when $\mathbf{x}_1 = \mathbf{0}$ and $S \subseteq R\mathbb{B}$

$$\mathbb{E}[f_t(\mathbf{x_t}) - f_t(\mathbf{x}_*)] \leq \frac{R^2}{2\eta} + n\frac{\eta G^2}{2}$$

when step-size $\eta = R/G\sqrt{n}$

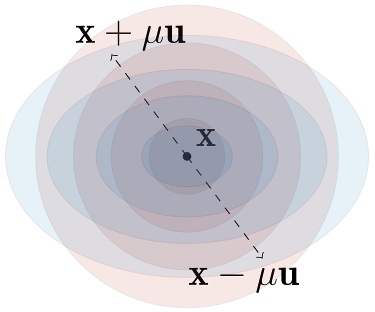
$$\mathbb{E}\left[\sum_{t=1}^{n} f(\mathbf{x}_t, \xi_t)\right] - \min_{\mathbf{x}} \sum_{t=1}^{n} f(\mathbf{x}, \xi_t) \leq RG\sqrt{n}$$

That's enough maths for now...

Two-point feedback

$$g_\mu(\mathbf{x}) = \frac{f(\mathbf{x}+\mu\mathbf{u},\xi_1) - f(\mathbf{x}-\mu\mathbf{u},\xi_2)}{2\mu}\mathbf{B}\mathbf{u}$$

By using a central difference of a smooth gradient approximation, you effectively get two realisations of ξ, a property that can be exploited.



Methods for structured objectives

A frequently encountered problem is

$$f(\mathbf{x}) = \frac{1}{2}||\mathbf{F}(\mathbf{x})||_2^2 = \frac{1}{2}\sum_{i=1}^{p} F_i(\mathbf{x})^2.$$

For example… **parameter estimation.**

Methods for structured objectives

DFO methods include:

- constructing individual models of F_i to produce $\nabla\mathbf{F}(\mathbf{x})$.
- Doing ’gradient’ steps one F_i at a time.
- Estimation of $\mathbf{F}$ by building linear models of F_i using central differences
- ... and many many more.

Sparse objective derivatives

Sometimes we know

$$\nabla^2 f(\mathbf{x})_{ij} = \nabla^2 f(\mathbf{x})_{ji} = 0 \quad \text{for all } (i,j) \in S$$

where S is some sparsity pattern. Similarly, as Damien previously discussed we can consider *partially separable* objectives of the form

$$f(\mathbf{x}) = \sum_{i=1}^{p} F_i(\mathbf{x}) = \sum_{i=1}^{p} F_i(\{\mathbf{x}_i\}_{j\in S})$$

Sparse objective derivatives

Knowing variable interactions allows us to reduce the degrees of freedom in producing models for model based DFO. For example:

- Dropping quadratic terms corresponding to non-interacting pairs (i, j).

- Quadratic terms selected using L_1 norms (promote sparsity).

- Exploit knowledge of f by choosing a particular *stencil*.

Methods for constrained optimization

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x},\mathbf{y}) \\ \text{subject to:} \quad & \mathbf{x} \in \Omega \subset \mathbb{R}^n \\ & \mathbf{y} \in \mathbf{N}, \end{aligned}$$

Where $\mathbf{N}$ is some finite set of discrete values, e.g. $\{A, B, C, D\}$.

Constraints can either be known, for example the minimum temperature difference of a heat exchanger must be above 5 degrees and this relation is known algebraically. Or may be unknown, for example, a simulation must converge. It may not be possible to characterise what set of inputs causes a simulation not to converge.

Algebraic Constraints

Our feasible region can be characterised by

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : c_i(\mathbf{x}) \leq 0, i \in I\}.$$

We have a number of different ways of dealing with these constraints…

- Penalty Approaches
- Filter Approaches
- Modelled Constraints

Penalty Approaches

The exact penalty function is denoted as

$$f(\mathbf{x}) + \frac{\rho}{2} \sum_i \max\{0, c_i(\mathbf{x})\}.$$

For ρ sufficiently large the minimum of this function is the same as that of the optimisation problem with constraints. Although the $\max\{0, c_i(\mathbf{x})\}$ is non-smooth, the problem remains convex if originally. Additionally this can be seen as a ’composite’ function and previous approaches can be taken advantage of.

Penalty Approaches

A more popular penalty function is the *quadratic penalty function*,

$$f(\mathbf{x}) + \rho \sum_i \max\{0, c_i(\mathbf{x})\}^2.$$

Here we maintain smoothness, exactness and convexity.

Penalty Approaches

We could also consider Lagrangian-based merit functions by assigning multipliers λ_i to each constraint. The Lagrangian and augmented Lagrangian are as follows:

$$L(\mathbf{x}; \lambda) = f(\mathbf{x}) + \sum_{i \in I} \lambda_i c_i(\mathbf{x}),$$

$$L_A(\mathbf{x}; \lambda; \rho) = f(\mathbf{x}) + \sum_{i \in I} \lambda_i c_i(\mathbf{x}) + \rho \sum_i \max\{0, c_i(\mathbf{x})\}^2.$$

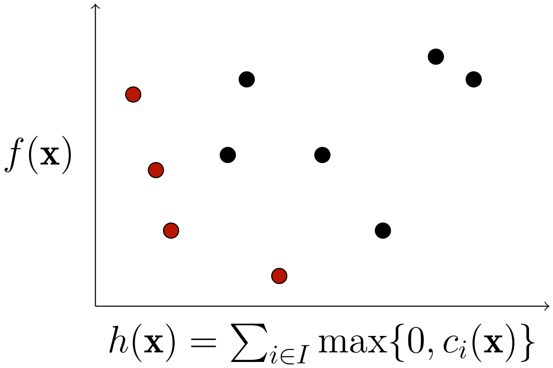
Penalty Approaches

Typically all these approaches rely on ρ dynamically changing throughout iterations. Normally ρ is increased as we assume $\mathbf{x}$ starts from outside of Ω and we then enforce constraints more aggressively throughout the scheme.

When does this not work?.. *non-computational experiments or functions*.

Filter Approaches

Filter approaches look at the constrained problem as a bi-objective optimisation problem.



Practical Considerations

Concurrent function evaluations should decrease wall-clock time as functions can be evaluated in parallel.

- Use multiple starting points.
- Evaluate different models.
- Generate multiple solutions for a model (e.g. solving with different radii)

Finite differences allow for n concurrent evaluations with forward difference and $2n$ concurrent evaluations with central difference.

Practical Considerations - Multistart methods

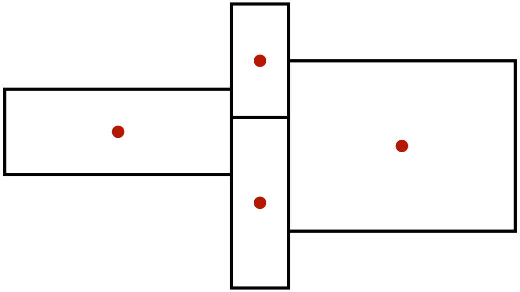
Used if we know or are unsure about the convexity of a problem.

- Split the search space Ω into subspaces.

- Share information between local searches.

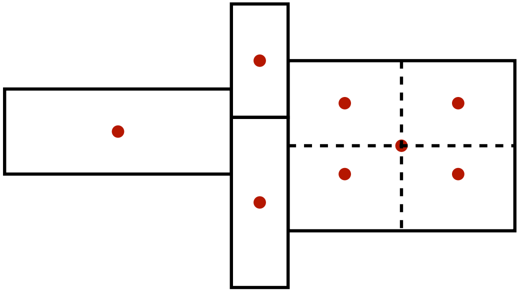
Other global optimisation methods

DIRECT - (DIviding RECTangles)

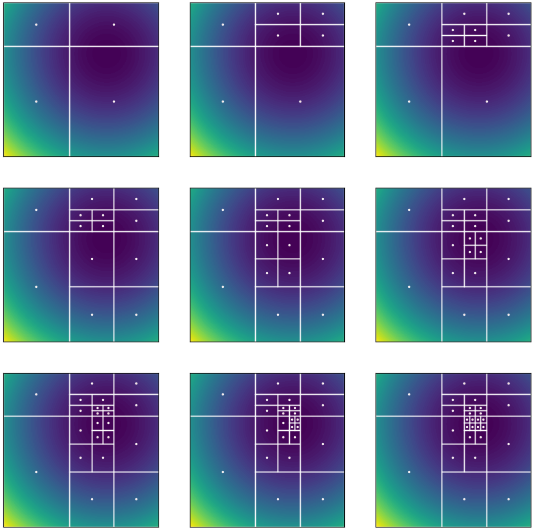


Solutions are scored based on the size of the largest length side as well as function value. Penalised big rectangles and large function values.

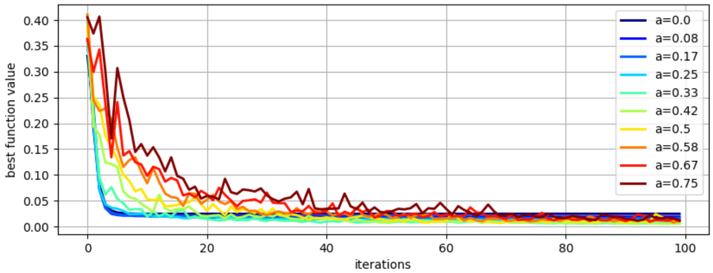
Other global optimisation methods



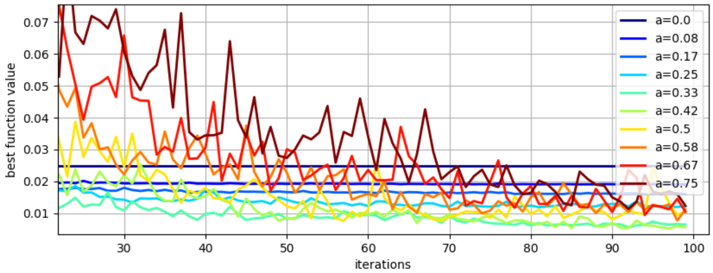
Split rectangle into partitions and repeat.
Extensions include Multilevel coordinate search (MCS) whereby function values are taken at vertices (and can be reused between rectangles).



$f(x) = f_f(x) - aL(x)$



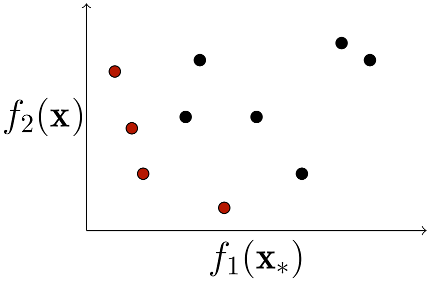
$f(x) = f_f(x) - aL(x)$



Methods for multi-objectives

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{F}(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \Omega \subset \mathbb{R}^n \end{aligned}$$

where objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$. Typically objectives are conflicting.



DF multi-objectives

Combine all objectives into a single function.

$$f_{MO}(\mathbf{x}) = \alpha f_1(\mathbf{x}) + (1 - \alpha)_2(\mathbf{x})$$

Change α in order to find solutions along the pareto-front. Only relevant for convex objectives really. Other methods include the α constraint method.

DF multi-objectives

Aside from just making a single objective in some sense, there is another approach.

- Construct a trust region of both objectives and a third that weights the two objectives.

Summary

First we introduced the setting of derivative free optimisation and looked into direct search methods including:

- Simplex method (Nelder Mead)
- Directional direct search methods, taking advantage of a stencil.

Then we looked into model-based methods including:

- Polynomials (under and over determined)
- Radial basis functions

Summary

Using these approximate models we motivated trust-region model based methods and described the generic trust-region algorithm. Then looked at one or two methods that fall into neither of these categories.

- Quadratic simplex models
- Implicit filtering

Summary

Looked at methods for deterministic objectives:

- Random search

- Nesterov Random Search

As well as the general scenario for stochastic convex optimisation, proving WCC for expected gradient descent.
Then went into structures that can be exploited in DFO such as sparsity, least-squares problems.

Summary

We also looked at the inclusion of constraints in DFO problems where can can:

- Enforce various types of penalties.
- Model our constraints.
- Use Filtering methods.

Finally we looked at multistart approaches, dividing rectangles, multiple objectives.

Finally

Larson, J., Menickelly, M., Wild, S. M. (2020). Derivative-free optimization methods. Acta Numerica, 287–404. https://doi.org/10.1017/S0962492919000060

Follows the same structure as this presentation, so if you want to look more into a topic, go here, find the approximate section and follow the references! It contains many more details than I've presented here (through fear of completely boring everyone).

Thanks everyone!