Optimization and Intersection of Functions

Alicja Wiączkowska

April 25, 2024

Abstract

The project concentrates around methods of finding optimum of complicated function. There appears also problem of searching for roots of a function as well as the intersection points of many curves. Many sollutions to theese problems can be found by using Chebfun package.

1 Rootfinding

1.1 Roots of polynomial

To find the zeros of a polynomial $a_nx^n + a_{n-1}x^{n-1} + ... + a_1x + a_0$ we can use built in function 'roots(p)', where p is a vector of polynomial coefficients $p = [a_n, a_{n-1}, ..., a_1, a_0]$.

1.2 Rootfinding using Chebfun

If f is defined as chebfun object we can determine its zeroes using function 'roots(f)'. It interpolates the given function f in Chebyshev points with Chebyshev polynomial p_k of automatically chosen degree k. Then if the degree is high (more than 50) the interpolant is broken into smaller pieces recursively. On each small piece zeros are then found as eigenvalues of a colleaque matrix (analogue for Chebyshev polynomials of a Vandermonde matrix used for polynomials, just the base differs). The eigenvalues of the colleague matrix associated with a Chebyshev polynomial provide the roots of that polynomial in the same way that the eigenvalues of the companion matrix associated with a monic polynomial provide its roots. This method is quite fast and accurate.

2 Intersection of functions

To find intersection points of two functions we need to find the roots of equation

$$f(x) - g(x) = 0$$

so if we define $h(x) := f(x) - g(x)$ we only need to find the roots of h using previously discussed methods.

To find the intersection points of N functions $f_1, f_2, ..., f_N$ we need to find all elements of set

$$\{x \in \Omega : (\exists_{\alpha \in \mathbf{R}})(\forall_{i \in \{1,2,...,N\}}) f_i(x) = \alpha\}.$$

This can be achieved by using the alogrythm above:

1. Find roots of $h_2(x) := f_1(x) - f_2(x)$ and save them into set $A_2 := \{x \in \Omega : f_1(x) - f_2(x) = 0\}$.
2. Let $A_i := \{x \in A_{i-1} : f_1(x) - f_i(x) = 0\}$ for $i \in \{3, 4..., N\}$.

   We can find the elements of set A_i by checking if element $x \in A_{i-1}$ satisfies the equation $f_1(x) - f_i(x) = 0$. Alternatively we can take the intersection of sets:

   $$A_i = A_{i-1} \cap \{x \in \Omega : f_1(x) - f_i(x) = 0\}$$

3. Recursively found A_N is the answear.

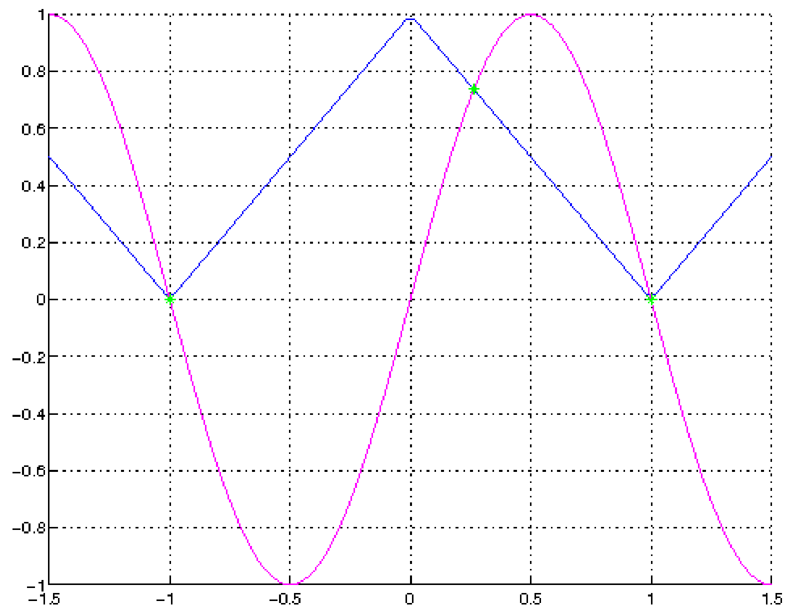


Figure 1: Intersecion points of 2 functions

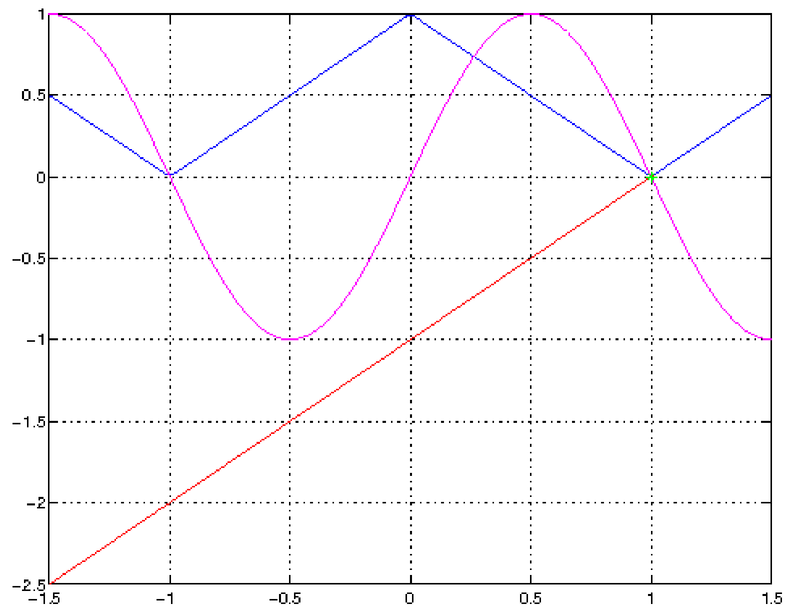


Figure 2: Intersecion point of 3 function

3 Chebfun minimisation function

3.1 Finding all local extrema

Since f is defined as chebfun object it by default is defined on range $[-1,1]^d$ if not specified diffrently. The area is bounded.

Local extrema of smooth functions can be located by finding zeros of the derivative. Chebfun has its own built-in function for diffrentiation 'diff(f)'. So one of methods of computing local extrema is to find roots by typing 'roots(diff(f))'.

Another way of finding all extrema is using function "[y,x]=minandmax(f,'local')", which returns 2 vectors: first is vector of values, and second is vector of arguments of the extrema. This function returns also endpoints of f and non-smooth extrema, because they are analogues to zeros of the derivative where the sign derivative changes from one to the another.

There are also Chebfun functions "[y,x]=min(f,'local')" and "[y,x]=max(f,'local')".

3.2 Finding global minimum

To find the global minimum we can use the function "[minval,minpos]=min(f,'global')" or just '[minval,minpos]=min(f)', where 'minpos' is the argument for the minimum and 'minval' is its value.

Chebfun computes the result by checking the values of f on extremas and endpoints

4 Other ways of finding global minimum

4.1 Grid search

In grid search method of finding the minimum we need a manually specified set of n points from the domain, usually equidistant when the domain is in 1 dimention or lattice points in 2d.

Then we evaluate the function f on those points and choose the smallest value. We treat the point with smallest value as global miniumum.

This method can be used on functions that are not continuous or differentiable.

If nearest points around certain one $\tilde{x}$ have larger values of f than $\tilde{x}$, we can treat point $\tilde{x}$ as local minimum. Similarly for local maximum, where $\tilde{x}$ is larger than points around it.

4.2 Random search

In random search method of finding the minimum we need a set of n randomly choosen points from the domain, usually using the uniform distribution.

Then we evaluate the function f on those points and choose the smallest value. We treat the point with smallest value as global miniumu.

If there are some points around certain one $\tilde{x}$ and they have larger values of f than $\tilde{x}$, we can treat point $\tilde{x}$ as local minimum. Similarly for local maximum, where $\tilde{x}$ is larger than points around it.

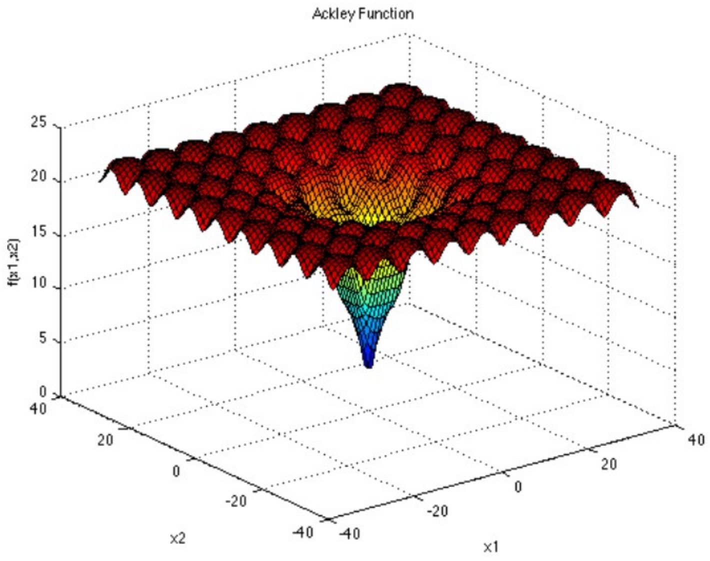
This method can be used on functions that are not continuous or differentiable.

5 Comparison of minimisation methods on 2d complicated functions

All functions were minimised using n^2 points. The results of the distance between computed argument of minimum and actual argument were plotted in respect to number of points. In each case interpolation using Chebyshev Polynomials (in 2d) returned the best approximation of argument of global minimum.

5.1 Ackley function

ACKLEY FUNCTION



$$f(\mathbf{x}) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

.

Recommended variable values are: a = 20, b = 0.2 and c = 2π.

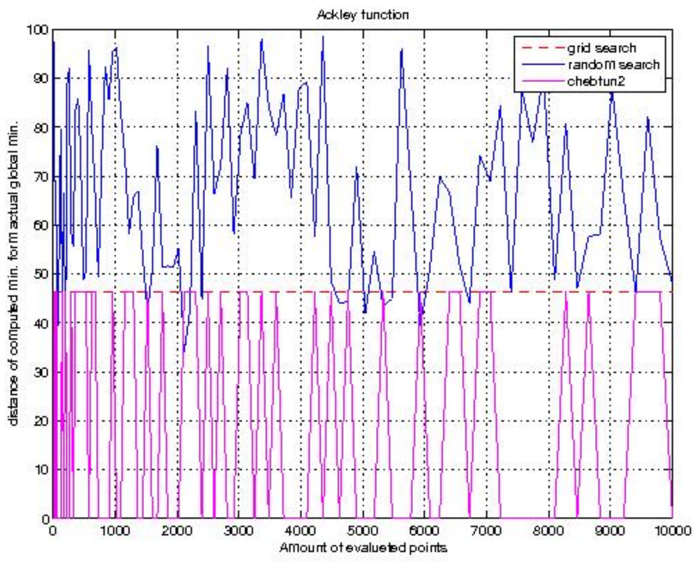
Input Domain:

The function is usually evaluated on the hypercube $x_i \in [-32.768, 32.768]$, for all i = 1, …, d, although it may also be restricted to a smaller domain.

Global Minimum:

$f(\mathbf{x}^*) = 0$, at $\mathbf{x}^* = (0, \ldots, 0)$

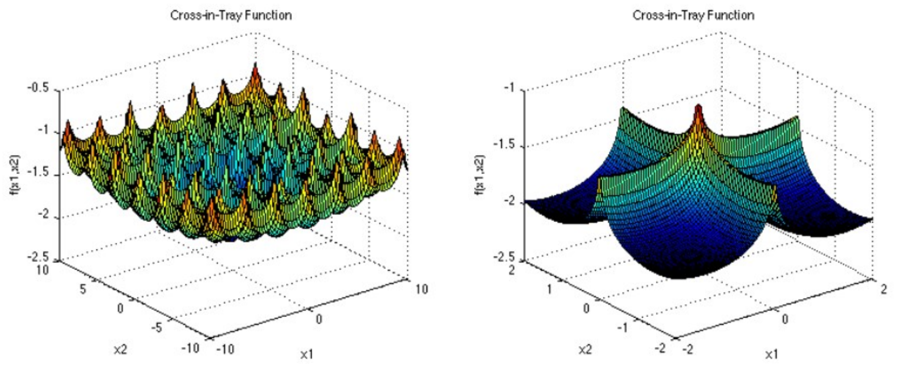
.



.

5.2 Cross-in-Tray function

CROSS-IN-TRAY FUNCTION



$$f(\mathbf{x}) = -0.0001\left(\left|\sin(x_1)\sin(x_2)\exp\left(\left|100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right|\right)\right|+1\right)^{0.1}$$

.

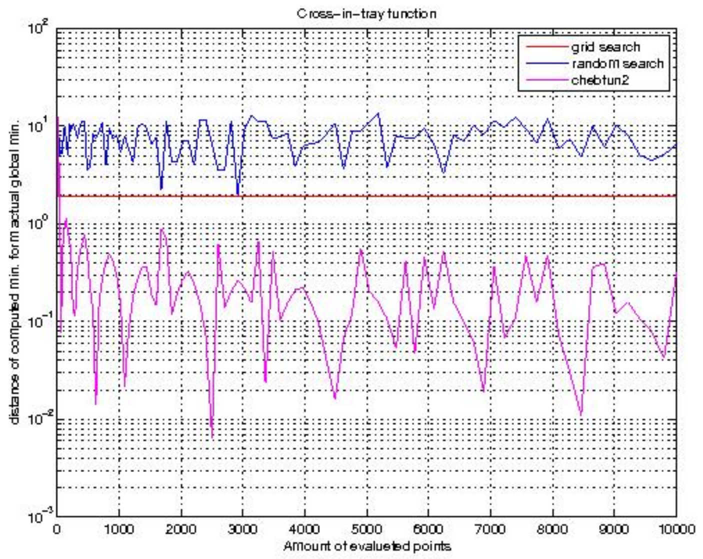
Input Domain:

The function is usually evaluated on the square $x_i \in [-10, 10]$, for all i = 1, 2.
However we will take square[0,10]x[0,10]

Global Minima:

$f(\mathbf{x}^*) = -2.06261$, at $\mathbf{x}^* = (1.3491, -1.3491)$, $(1.3491, 1.3491)$, $(-1.3491, 1.3491)$ and $(-1.3491, -1.3491)$

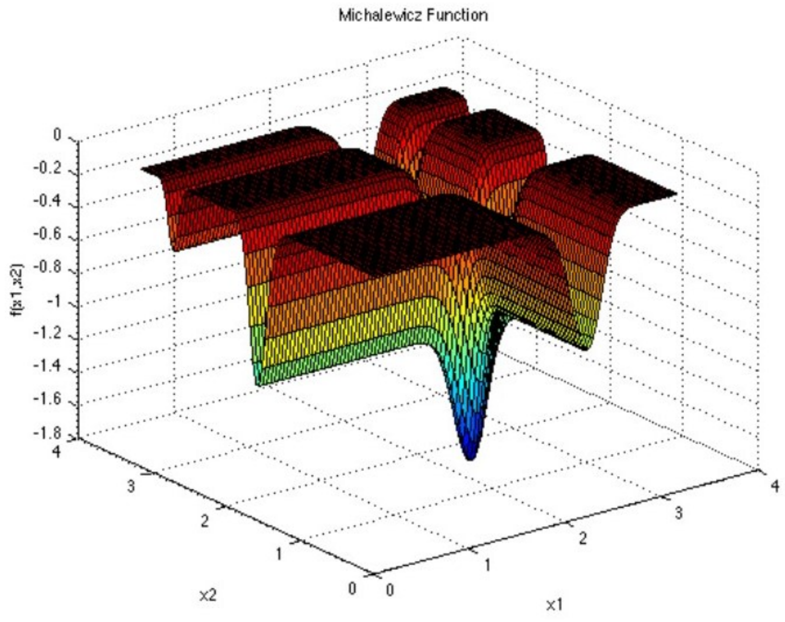
.



.

5.3 Michalewicz function

MICHALEWICZ FUNCTION



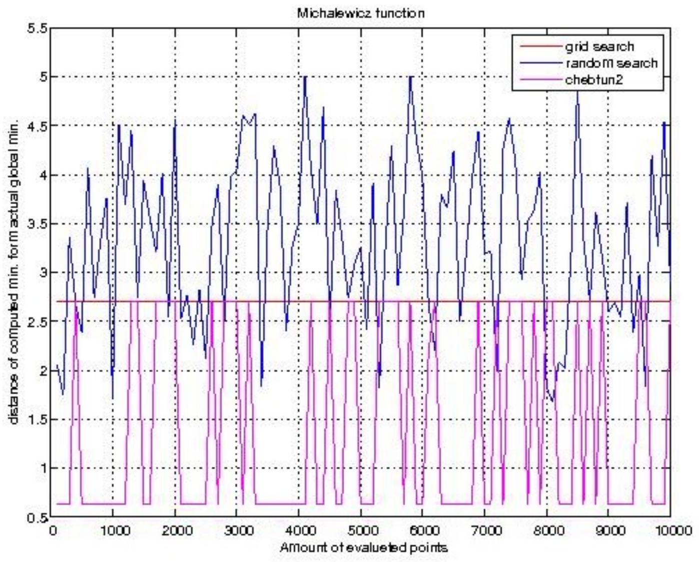
$$f(\mathbf{x}) = -\sum_{i=1}^{d} \sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$$

Input Domain:

The function is usually evaluated on the hypercube $x_i \in [0, \pi]$, for all i = 1, …, d.

Global Minima:

at $d = 2$: $f(\mathbf{x}^*) = -1.8013$, at $\mathbf{x}^* = (2.20, 1.57)$


Michalewicz function
grid search
random search
chebfun2
distance of computed min. form actual global min.
Amount of evalueted points
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5

.

References

[1] https://www.mathworks.com/help/matlab/ref/roots.html

[2] https://blogs.mathworks.com/cleve/2012/10/08/chebfun-roots/

[3] https://www.chebfun.org/docs/guide/guide03.html

[4] https://www.chebfun.org/docs/guide/chebfun_guide.pdf

[5] https://www.mathworks.com/matlabcentral/fileexchange/47023-chebfun-current-version

[6] https://en.wikipedia.org/wiki/Hyperparameter_optimization

[7] https://en.wikipedia.org/wiki/Random_search

[8] https://www.sfu.ca/ ssurjano/michal.html

[9] https://www.sfu.ca/ ssurjano/ackley.html

[10] https://www.sfu.ca/ ssurjano/crossit.html