Finding rare trajectories in transiently chaotic systems

Leonardo L. D. L. Hügens

J. M. Viana Parente Lopes

Introduction

In simplified terms, a dynamical system is a set of there entities: a function $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^n, \boldsymbol{r} \mapsto \boldsymbol{F}(\boldsymbol{r})$ (also refered to as a "map"), a domain $\Gamma \in \mathbb{R}^n$, and a algorithm. This algorithm consists in choosing an initial position $\boldsymbol{r}_0 \in \Gamma$, mapping it to another position through the function, $\boldsymbol{r_1} = \boldsymbol{F}(\boldsymbol{r_0})$, and repeating the last step recursively ($\boldsymbol{r_2} = \boldsymbol{F}(\boldsymbol{r_1})$, ...). We define the escape time of the initial position $t(\boldsymbol{r_0})$ as the number of these iterations needed for the jumping point to fall outside the domain Γ. A visualization of this process is represented below, in Figure 1. This is a very basic definition, but it is completely sufficient in the context of this project.

Our main goal is to find algorithms that can find points $\boldsymbol{r_0}$ with maximum escape time, implement them in computer programs, and finally tweak and compare them.

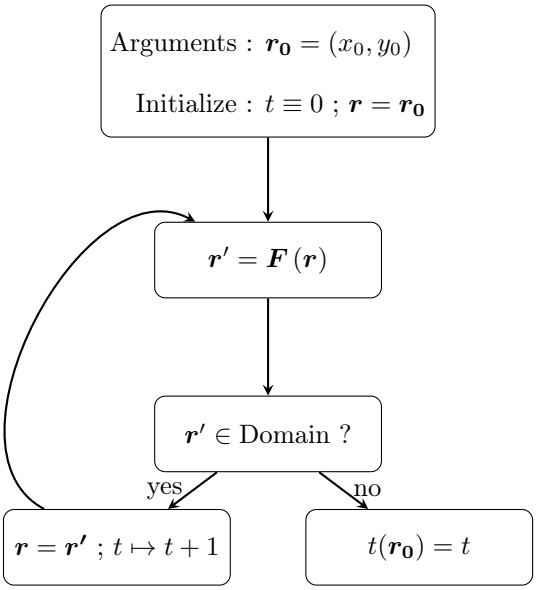


Figure 1: Representation of the iteration process.

Test Map

The map that will be used throughout this study is the iconic Hénon map, with $k = 6$:

$$x_{n+1} = k - x^2 - y_n$$

$$y_{n+1} = x_n$$

A very aesthetically pleasing way to represent bi-dimensional dynamical systems in the point of view of the escape time is to plot it in the z axis, along with the corresponding points in the x-y plane. Below, in Fig. 2, we represent the Hénon map in this manner. Note that we limited the plot to this box to see the base structure more clearly, as the maximum escape times we have found are about 600 iterations.

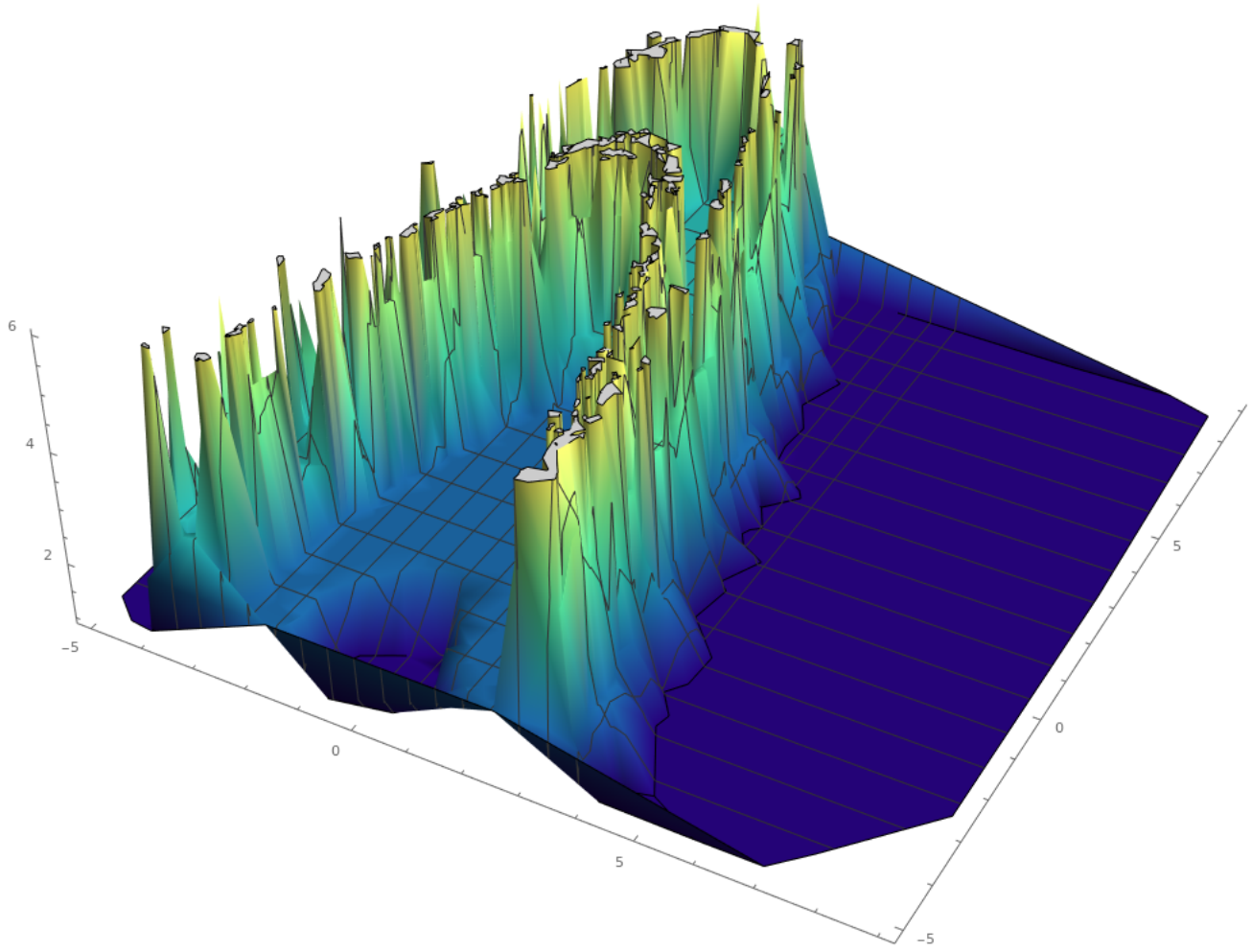


Figure 2: Landscape of the henon map with k = 6.

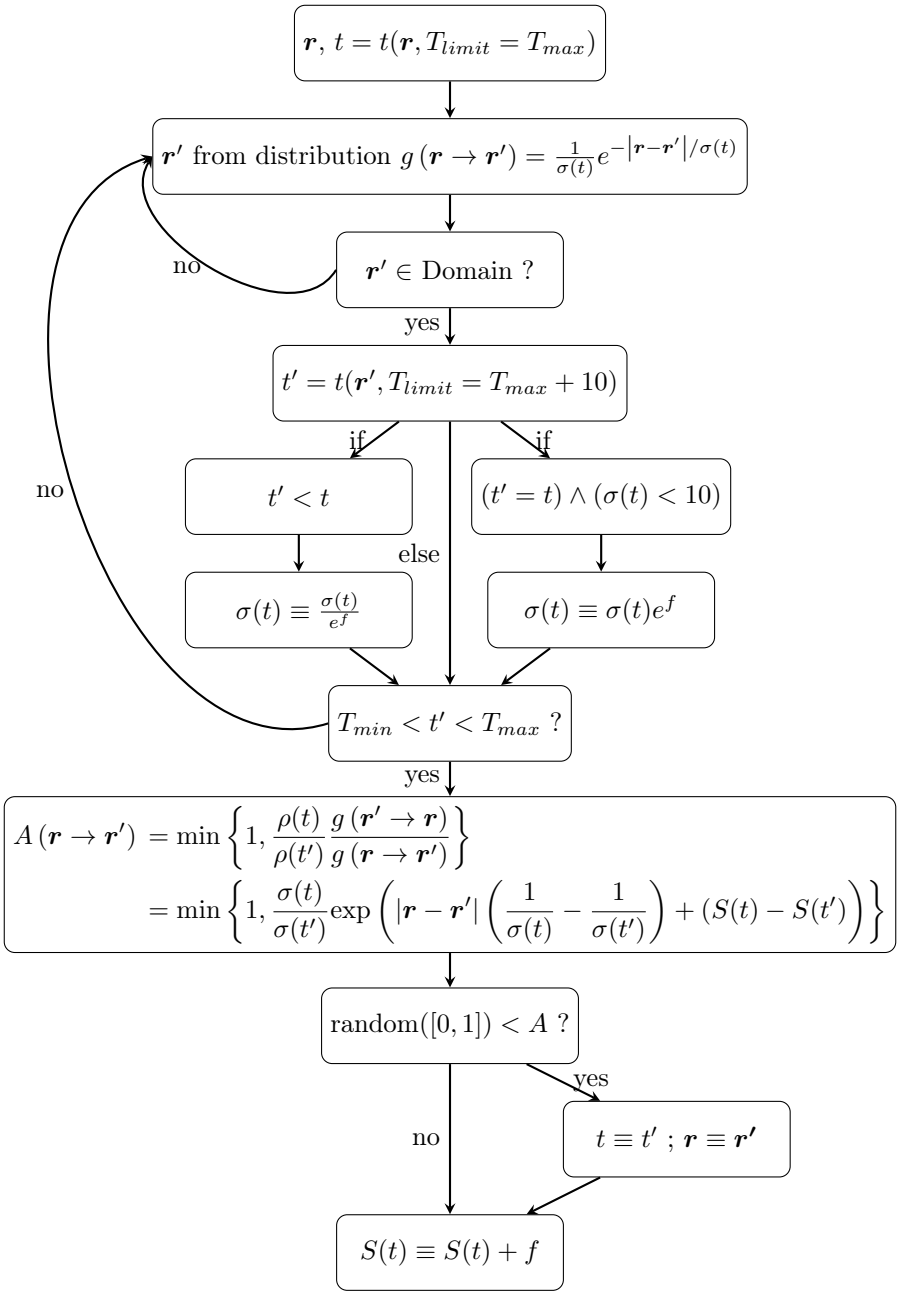
Generic Maximization Algorithms

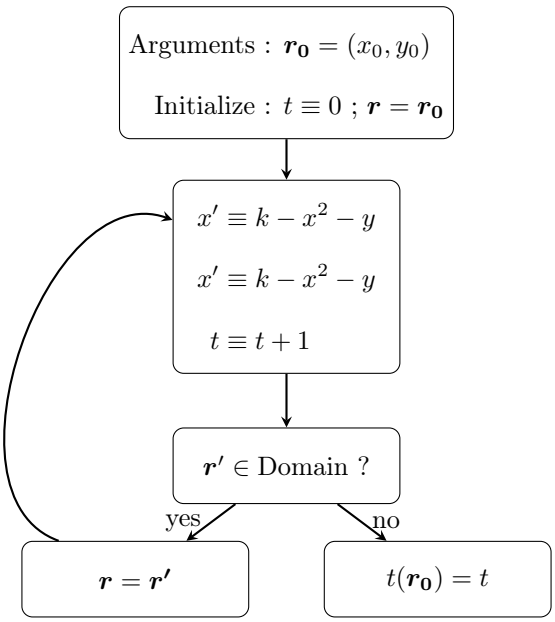
The source https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html provides several generic minimization algorithms implemented in the scipy.optimize package, ready to use.

The following algorithms are automatically unusable due to the need of a gradient of the function to be minimized, which is generaly impossible to obtain due to the roughness of the landscape: Newton-Conjugate-Gradient (NCG), Trust-Region NCG, Trust-Region Truncated Generalized Lanczos / NCG, Trust-Region Nearly Exact.

At first sight, the usable algorithms are those who only rely on the evaluation of the function, those which belong to the **Pattern Search** family (also known as direct search, derivative search, or black-box search).

0.1 Multicanonical Sampling



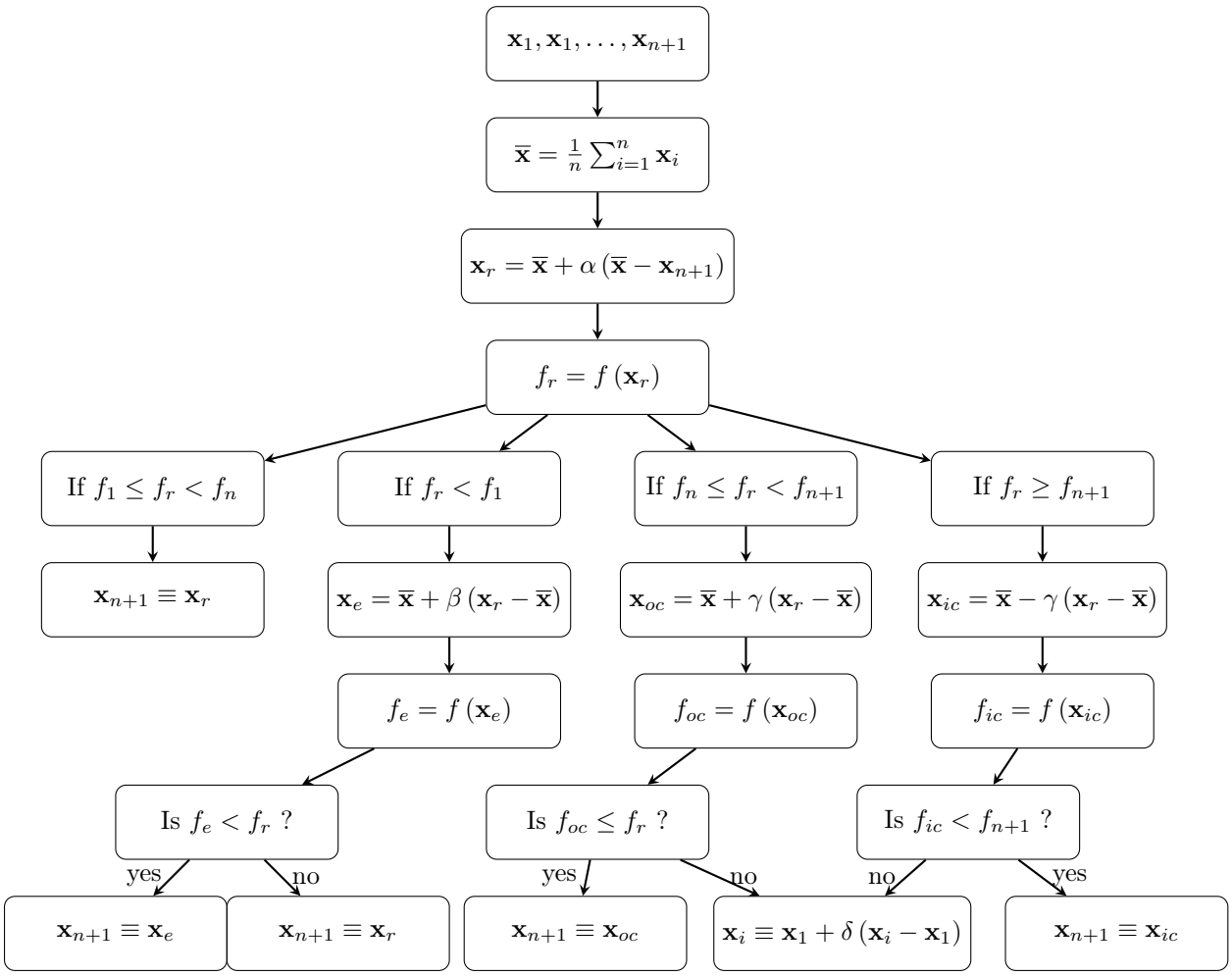


0.2 Nelder-Mead Algorithm

Reference: Gao, F. and Han, L. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. 2012. Computational Optimization and Applications. 51:1, pp. 259-277

Let $f : \mathbb{R}^n \to \mathbb{R}$ be the function to be minimized. A n-dimensional simplex is the convex hull (smallest convex set) delimited by n+1 points (vertices). Let Δ denote the simplex and $\mathbf{x}_1$,...,$\mathbf{x}_{n+1}$ the vertices which define it. The method makes use of the following definitions, with $\alpha > 0$, $\beta > 1$, $0 < \gamma < 1$ and $0 < \delta < 1$.

One iteration of the algorithm proceeds as follows:



An optimal choice of initial vertices of the simplex and $\{\alpha, \beta, \gamma, \delta\}$ values is highly dependent on the nature of the problem. Althought there is no special termination condition associated with the algorithm, Nelder and Mead used the standard deviation of the values of the function on the vertices of the simplex.

0.3 Powell's Method

References: Brent, Richard P. (1973). "Section 7.3: Powell's algorithm". Algorithms for minimization without derivatives; Flannery, BP (2007); Numerical Recipes: The Art of Scientific Computing.

To understand this method, we must first have the definition of *conjugate* directions. Consider the following Taylor series approximation to a function $f(\mathbf{X})$ and a point $\mathbf{P}$:

$$f(\mathbf{x}) = f(\mathbf{P}) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j + \cdots \approx c - \mathbf{b} \cdot \mathbf{x} + \frac{1}{2} \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}$$

where

$$c \equiv f(\mathbf{P}) \quad \mathbf{b} \equiv -\nabla f \Big| \mathbf{P} \quad [\mathbf{A}]_{ij} \equiv \frac{\partial^2 f}{\partial x_i \partial x_j} \Big|_{\mathbf{P}}$$

and the approximation to the functions gradient:

$$\nabla f = \mathbf{A} \cdot \mathbf{x} - \mathbf{b}$$

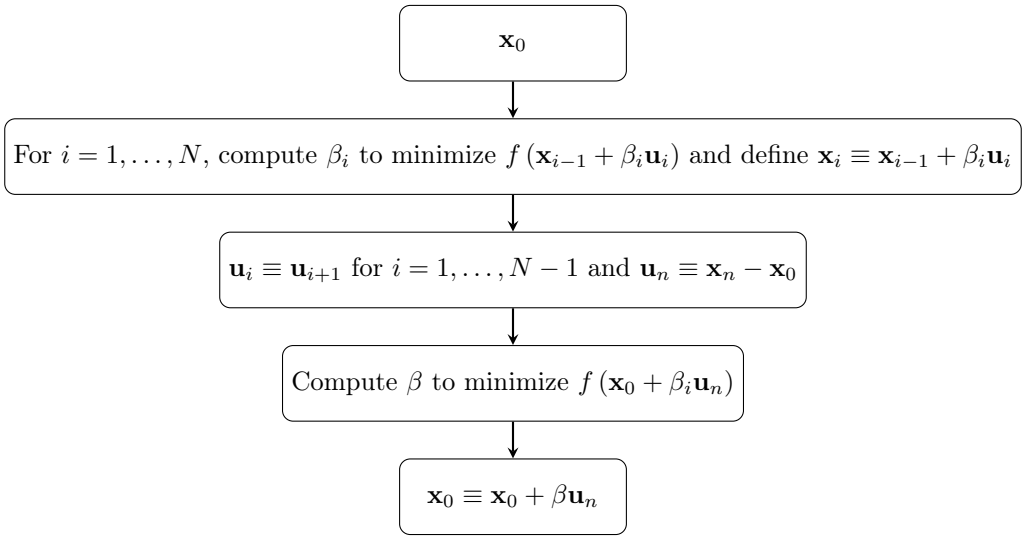
Therefore, the gradient will vanish - the function will be at a neighbouring extremum - at $\mathbf{P} + \mathbf{x}$ where $\mathbf{x}$ is the solution to $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$. If we move along that direction to a minimum and now consider moving in a direction $\mathbf{v}$, in order not to ruin the previous minimization, we pretend that the gradient continues to be perpendicular to $\mathbf{x}$. Considering a change in the gradient:

$$\delta(\nabla f) = \mathbf{A} \cdot (\delta \mathbf{v})$$

we therefore are interested in directions $\mathbf{v}$ that satisfy:

$$0 = \mathbf{u} \cdot \delta(\nabla f) = \mathbf{u} \cdot \mathbf{A} \cdot \mathbf{v}$$

When this equation holds for two directions $\mathbf{u}$ and $\mathbf{v}$, they are said to be *conjugate*. Powell's method produces n mutually conjugate directions for an objective n-dimensional function, and needs an 1-dimensional minimization algorithm such as Golden Search to minimize $f(\mathbf{x})$ in a particular direction. The initial directions $\mathbf{u}_0, ..., \mathbf{u}_{i+1}$ are usually chosen to be the cartesian basis vectors. An iteration of the algorithm is represented below.



1 The Rosenbrock function

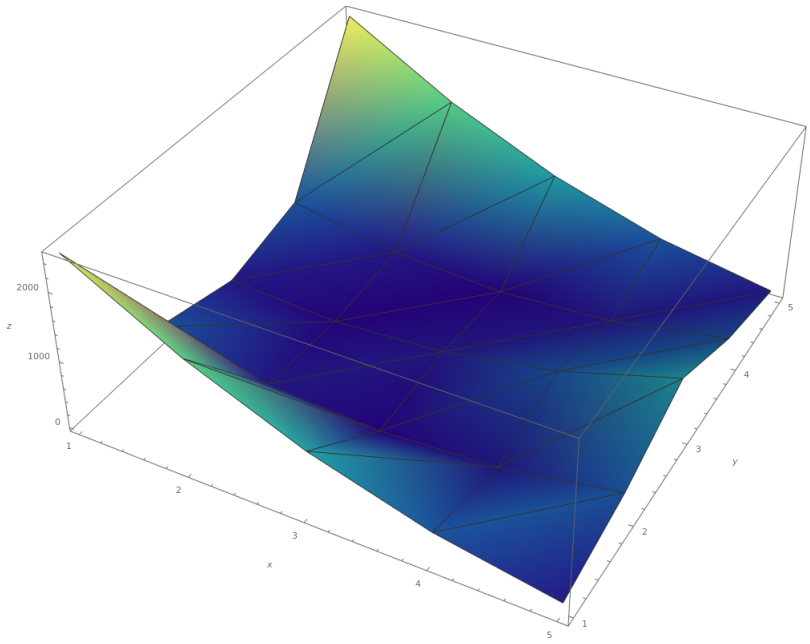


Figure 3: Rosenbrock function ($z = f(x, y)$) for $a = 1$ and $b = 100$.

The Rosenbrock function (represented in the figure above) is a non-convex function which is used as a performance test problem for optimization algorithms. The global minimum is inside a long, narrow, parabolic shaped flat valley, and minimization algorithms usually converge rapidly to this valley, but converging to the actual global minimum is harder. The function is defined as:

$$f(x, y) = (a - x)^2 + b\left(y - x^2\right)^2$$

Its partial derivatives are:

$$\frac{\partial f}{\partial x}(x, y) = 2(a - x) - 4bx(y - x^2)$$

$$\frac{\partial f}{\partial y}(x, y) = 2b(y - x^2)$$

Calculating extremum points:

$$\frac{\partial f}{\partial y}(x, y) = 0 \implies y = x^2$$

$$(y = x^2 \wedge \frac{\partial f}{\partial x}(x, y) = 0) \implies (x = a \wedge y = a^2)$$

$$f(a, a^2) = 0$$

The global minimum is therefore located at (a, a^2) and its value is 0.