Squintability and Other Metrics for Assessing Projection Pursuit Indexes, and Guiding Optimisation Choices

H. Sherry Zhang[1], Dianne Cook[2], Nicolas Langrené[3], Jessica Wai Yin Leung[2]

ARTICLE HISTORY
Compiled March 3, 2025

[1] Department of Statistics and Data Sciences, University of Texas at Austin, Austin, United States
[2] Department of Econometrics and Business Statistics, Monash University, Melbourne, Australia
[3] Department of Mathematical Sciences, Guangdong Provincial/Zhuhai Key Laboratory of Interdisciplinary Research and Application for Data Science, BNU-HKBU United International College, Zhuhai, China

ABSTRACT
The projection pursuit (PP) guided tour interactively optimises a criterion function known as the PP index, to explore high-dimensional data by revealing interesting projections. Optimisation of some PP indexes can be non-trivial, if they are non-smooth functions, or the optimum has a small "squint angle", detectable only from close proximity. Here, measures for calculating the smoothness and squintability properties of the PP index are defined. These are used to investigate the performance of a recently introduced swarm-based algorithm, Jellyfish Search Optimiser (JSO), for optimising PP indexes. The performance of JSO for visualising data is evaluated across various hyper-parameter settings and compared with existing optimisers for the guided tour. The JSO algorithm has been implemented in the R package, `tourr`, and functions to calculate smoothness and squintability measures are implemented in the `ferrn` package.

KEYWORDS
projection pursuit; jellyfish search optimiser (JSO); optimisation; grand tour; high-dimensional data; exploratory data analysis

1. Introduction

Projection pursuit (PP) (Kruskal 1969; Friedman and Tukey 1974; Huber 1985) is a dimension reduction technique aimed at identifying informative linear projections of

CONTACT: H. Sherry Zhang. Email: huize.zhang@austin.utexas.edu. Dianne Cook. Email: dicook@monash.edu. Nicolas Langrené. Email: nicolaslangrene@uic.edu.cn. Jessica Wai Yin Leung. Email: Jessica.Leung@monash.edu.

data. This is useful for exploring high-dimensional data, and creating plots of the data that reveal the main features to use for publication. The method involves optimising an objective function known as the PP index (e.g., Hall 1989; Cook, Buja, and Cabrera 1993; Lee and Cook 2010; Loperfido 2018, 2020), which defines the criterion for what constitutes interesting or informative projections. Let $X \in \mathbb{R}^{n\times p}$ be the data matrix, $A \in \mathbb{R}^{p\times d}$ be an orthonormal matrix, where A belongs to the Stiefel manifold $\mathcal{A} = V_d(\mathbb{R}^p)$. The projection $Y = XA$ is a linear transformation that maps data from a p-dimensional space into a d-dimensional space. The index function $f(XA) : \mathbb{R}^{n\times d} \to \mathbb{R}$ is a scalar function that measures an interesting aspect of the projected data, such as deviation from normality, presence of clusters, or non-linear structure. For a fixed sample of data, PP finds the orthonormal basis A that maximises the index value of the projection, $Y = XA$:

$$\max_{A\in\mathcal{A}} \quad f(XA) \quad \text{subject to} \quad A'A = I_d \tag{1}$$

It is interesting to note that when using PP visually, one cares less about A than the plane described by A, because the orientation in the plane is irrelevant. The space of planes belongs to a Grassmann manifold. This is usually how the projection pursuit guided tour (PPGT) (Cook et al. 1995) operates, when using geodesic interpolation between starting and target planes. It interpolates plane to plane, removing irrelevant within plane spin, and is agnostic to the basis (A) used to define the plane. Thus, indexes which are used for the PPGT should be rotationally invariant.

Index functions are quite varied in form, partially depending on the data that is being

projected. Figure 1 shows two examples. Huber plots (Huber 1990) of 2D data sets are in (a) and (c), showing the PP index values for all 1D projections of the 2D data in polar coordinates, which reveals the form of these functions. The dashed circle is a baseline set at the average value, and the straight line marks the optimal projection. Plots (b) and (d) show the respective best projections of the data as histograms. Indexes like the holes, central mass and skewness (Cook, Buja, and Cabrera 1993) are generally smooth for most data sets, but capture only large patterns. Many indexes are noisy and non-convex, requiring an effective and efficient optimisation procedure to explore the data landscape and achieve a globally optimal viewpoint of the data. The skewness index computed for trimodal data, in (a), is smooth with a large squint angle but has three modes, and thus is not convex. The binned normality index (a simple version of a non-normality index as described in Huber 1985) computed on the famous RANDU data, in (c), is noisier and has a very small squint angle. The discreteness cannot be seen unless the optimiser is very close to the optimal projection.

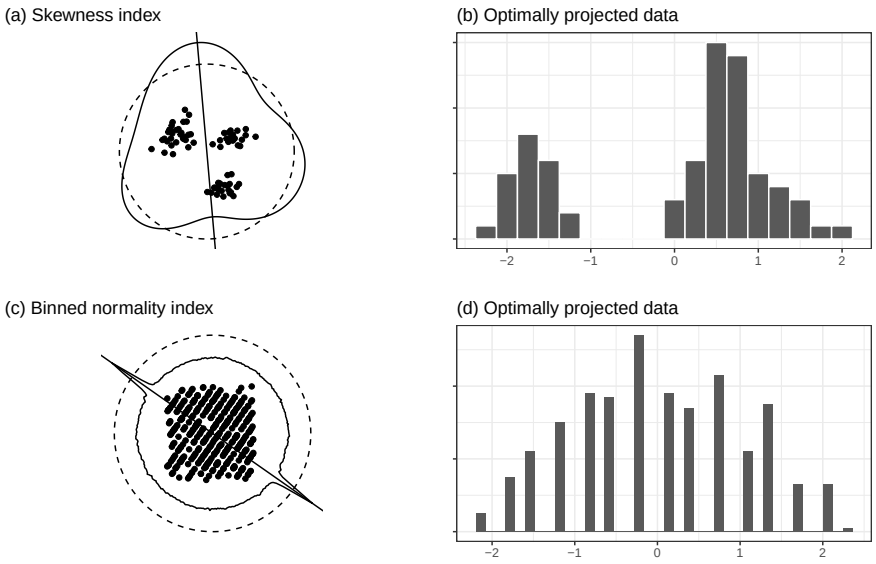

Figure 1. Examples of PP indexes with large (top row) and small (bottom row) squint angles, shown with a Huber plot, and histogram of the projected data corresponding to the optimal projection. A Huber plot shows the PP index values for all 1D data projections in polar coordinates. The skewness index on the trimodal data is also smoother than the binned normality index on RANDU.

Optimisation of PP is often discussed when new indexes are proposed (Posse 1995; Marie-Sainte, Berro, and Ruiz-Gazen 2010; Grochowski and Duch 2011). Cook et al. (1995) tied the optimisation more closely to the index, when they introduced the PPGT, which monitors the optimisation visually so that the user can see the projected data leading in and out of the optimum. An implementation is available in the `tourr` package (H. Wickham et al. 2011) in R (R Core Team 2023). Zhang et al. (2021) illustrated how to diagnose optimisation processes, particularly focusing on the guided tour, and revealed a need for improved optimisation. While improving the quality of the optimisation solutions in the tour is essential, it is also important to be able to view the data projections as the optimisation progresses. Integrating the guided tour with a global optimisation algorithm that is efficient in finding the global optimal and enables viewing of the projected data during the exploration process is a goal.

Here, the potential for a Jellyfish Search Optimiser (JSO) Rajwar, Deep, and Das (2023) for the PPGT is explored. JSO, inspired by the search behaviour of jellyfish in the ocean, is a swarm-based metaheuristic designed to solve global optimisation problems. Compared to traditional methods, JSO has demonstrated stronger search ability and faster convergence, and requires fewer tuning parameters. These practicalities make JSO a promising candidate for enhancing PP optimisation.

The primary goal of the study reported here is to investigate the performance of JSO in PP optimisation for the guided tour. It is of interest to assess how quickly and closely the optimiser reaches a global optimum, for various PP indexes that may have differing complexities. To observe the performance of JSO with different types of PP indexes, metrics are introduced to capture specific properties of the index including squintability (based on Tukey and Tukey 1981's squint angle) and smoothness. Here, we mathematically define metrics for squintability and smoothness, which is a new

contribution for PP research. A series of simulation experiments using various datasets and PP indexes are conducted to assess JSO's behaviour and its sensitivity to hyper-parameter choices (number of jellyfish and maximum number of tries). The relationship between the JSO performance, hyper-parameter choices and properties of PP indexes (smoothness and squintability) is analysed to provide guidance on selecting optimisers for practitioners using projection pursuit. Additionally, this work should guide the design of new PP indexes and facilitate better optimization for PP.

The paper is structured as follows. Section 2 introduces the background of the PPGT, reviews existing optimisers and index functions in the literature. Section 3 introduces the metrics that measure two properties of PP indexes, smoothness and squintability, and Section 4 describes the new JSO to be used for the PPGT. Section 5 outlines two simulation experiments to assess JSO's performance: one comparing JSO's performance improvements relative to an existing optimiser, Creeping Random Search (CRS), and the other studying the impact of PP index properties on optimisation performance, and Section 6 presents the results. Section 7 discusses the implementation of JSO in the `tourr` package and the PP property calculation in the `ferrn` package. Section 8 summarises the work and provides suggestions for future directions.

2. Projection pursuit, tours, index functions and optimisation

A tour on high-dimensional data is constructed by geodesically interpolating between pairs of planes. Any plane is described by an orthonormal basis, A_t, where t represents time in the sequence. The term "geodesic" refers to maintaining the orthonormality constraint so that each view shown is correctly a projection of the data. The PP guided tour operates by geodesically interpolating to target planes (projections) which have

high PP index values, as provided by the optimiser. The geodesic interpolation means that the viewer sees a continuous sequence of projections of the data, so they can watch patterns of interest forming as the function is optimised. There are five optimisation methods implemented in the `tourr` package:

- a pseudo-derivative, that searches locally for the best direction, based on differencing the index values for very close projections.
- a brute-force optimisation (CRS).
- a modified brute force algorithm described in Posse (1995).
- an essentially simulated annealing (Bertsimas and Tsitsiklis 1993) where the search space is reduced during the optimisation.
- a very localised search, to take tiny steps to get closer to the local maximum.

There are numerous PP index functions available: introduced in Huber (1985), Cook, Buja, and Cabrera (1993), Lee et al. (2005), Lee and Cook (2010), Grimm (2016), Ursula Laa and Valencia (2022). Most are relatively simply defined, for any projection dimension, and implemented because they are relatively easy to optimise. A goal is to develop PP indexes based on scagnostics Leland Wilkinson and Wills (2008), but the blockage is their optimisation as these tend to be noisy, with potentially small squint angles.

An initial investigation of PP indexes, and the potential for scagnostics is described in Laa and Cook (2020). To be useful here an optimiser needs to be able to handle index functions that are possibly not very smooth. In addition, because data structures might be relatively fine, the optimiser needs to be able to find maxima that occur with a small squint angle, that can only be seen from very close by. One last aspect that is useful is for an optimiser to return local maxima in addition to the global one because data can

contain many different and interesting features.

3. Properties of PP indexes

Laa and Cook (2020) has proposed five criteria for assessing projection pursuit indexes (smoothness, squintability, flexibility, rotation invariance, and speed). Since not all index properties affect the optimisation process, the focus here is on the first two properties, *smoothness* (Section 3.1) and *squintability* (Section 3.2), for which metrics are proposed to quantify them.

3.1. *Smoothness*

This subsection proposes a metric for the smoothness of a projection pursuit index.

A classical way to describe the smoothness of a function is to identify how many continuous derivatives of the function exist. This can be characterized by Sobolev spaces (Adams and Fournier 2003).

Definition 1 (Sobolev space)**.** *The Sobolev space* $W^{k,p}(\mathbb{R})$ *for* $1 \leq p \leq \infty$ *is the set of all functions* f *in* $L^p(\mathbb{R})$ *for which all weak derivatives* $f^{(\ell)}$ *of order* $\ell \leq k$ *exist and have a finite* L^p *norm.*

The Sobolev index k in Definition 1 can be used to characterize the smoothness of a function: if $f \in W^{k,p}$, then the higher k, the smoother f. While this Sobolev index k is a useful measure of smoothness, it can be difficult to compute or even estimate in practice.

To obtain a computable estimator of the smoothness of the index function f, we propose

an approach based on random fields. If a PP index function f is evaluated at some random bases, as is done at the initialization stage of JSO, then these random index values can be interpreted as a random field, indexed by a space parameter, namely the random projection basis. This analogy suggests to use this random training sample to fit a spatial model. We propose to use a Gaussian process equipped with a Matérn covariance function, due to the connections between this model and Sobolev spaces, see for example Porcu et al. (2024).

The distribution of a Gaussian process is fully determined by its mean and covariance function. The smoothness property comes into play in the definition of the covariance function: if a PP index is very smooth, then two close projection bases should produce close index values (strong correlation); by contrast, if a PP index is not very smooth, then two close projection bases might give very different index values (fast decay of correlations with respect to distance between bases). Popular covariance functions are parametric positive semi-definite functions. In particular, the Matérn class of covariance functions has a dedicated parameter to capture the smoothness of the Gaussian field.

Definition 2 (Matérn covariance function)**.** *The Matérn covariance function K is defined by*

$$K(u) = K_{\nu,\eta,\ell}(u) := \eta^2 \frac{\left(\sqrt{2\nu}\frac{\|u\|}{\ell}\right)^{\nu}}{\Gamma(\nu)2^{\nu-1}} \mathcal{K}_{\nu}\left(\sqrt{2\nu}\frac{\|u\|}{\ell}\right) , \tag{2}$$

where $\|u\|$ is the Euclidean norm of $u \in \mathbb{R}^{p\times d}$, $\nu > 0$ is the smoothness parameter, η is the outputscale, ℓ is the lengthscale, and $\mathcal{K}_{\nu}$ is the modified Bessel function [DLMF 10.25].

The Matérn covariance function can be expressed analytically when ν is a half-integer,

the most popular values in the literature being $\frac{1}{2}$, $\frac{3}{2}$ and $\frac{5}{2}$ (Rasmussen and Williams 2006). The parameter ν, called *smoothness parameter*, controls the decay of the covariance function. As such, it is an appropriate measure of smoothness of a random field, as shown by the simulations on Figure 2 and Figure 3. For example, Karvonen (2023) showed that if a function f has a Sobolev index of k, then the smoothness parameter estimate ν in (2) cannot be asymptotically less than k. See the survey Porcu et al. (2024) for additional results on the connection between the Matérn model and Sobolev spaces. An interesting result is that the asymptotic case $\nu \to \infty$ coincides with the Gaussian kernel: $K_\infty(u) = \exp(-\|u\|^2/2)$.

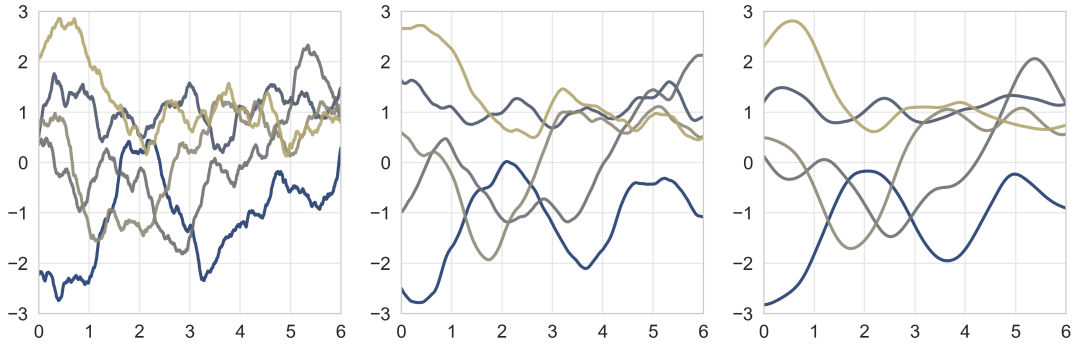

Figure 2. Five random simulations from a Gaussian Process defined on $\mathbb{R}$ with zero mean and Matérn-ν covariance function, with $\nu = 1$ (left), $\nu = 2$ (middle), and $\nu = 4$ (right), showing that higher values of ν produce smoother curves.

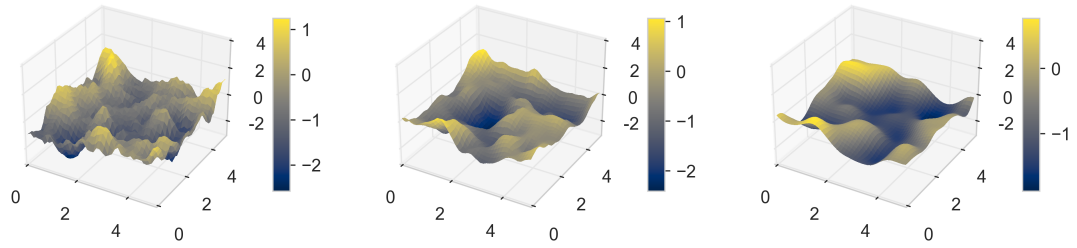

Figure 3. One random simulation from a Gaussian Process defined on $\mathbb{R}^2$ with zero mean and Matérn-ν covariance function, with $\nu = 1$ (left), $\nu = 2$ (middle), and $\nu = 4$ (right), showing that higher values of ν produce smoother surfaces.

In view of these results, the parameter ν is suggested as a measure of the smoothness of the PP index function by fitting a Gaussian process prior with Matérn covariance

on a dataset generated by random evaluations of the index function, as done at the initialization stage of the jellyfish search optimization. There exist several R packages, such as `GpGp` (Guinness, Katzfuss, and Fahmy 2021) or `ExaGeoStatR` (Abdulah et al. 2023), to fit the hyper-parameters of a GP covariance function on data, which is usually done by maximum likelihood estimation. In this project, the `GpGp` package is used.

Definition 3. *Let* $\mathbf{A} = [A_1, \dots, A_N] \in (\mathbb{R}^{p\times d})^N$ *be d-dimensional projection bases, let* $\mathbf{y} = [f(XA_1), \dots, f(XA_N)]$ *be the corresponding PP index values, and let* $\mathbf{K} = [K_\theta(A_i - A_j)]_{1\le i,j\le N} \in \mathbb{R}^{N\times N}$ *be the Matérn covariance matrix evaluated at the input bases, where the vector* θ *contains all the parameters of the multivariate Matérn covariance function* K *(smoothness, outputscale, lengthscales). The log-likelihood of the parameters* θ *is defined by*

$$\mathcal{L}(\theta) = \log p(\mathbf{y}\,|\mathbf{A}, \theta) = -\frac{1}{2}\mathbf{y}^\top(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log(\det(\mathbf{K} + \sigma^2\mathbf{I})) - \frac{N}{2}\log(2\pi) \quad (3)$$

where the nugget parameter σ *is the standard deviation of the intrinsic noise of the Gaussian process. The optimal parameters (including smoothness) are obtained by maximum log-likelihood*

$$\theta^* = \max_\theta \mathcal{L}(\theta) \quad (4)$$

The resulting optimal smoothness parameter ν *is chosen as our smoothness metric.*

The value of the optimal smoothness parameter $\nu > 0$ can be naturally interpreted as follows: the higher ν, the smoother the index function.

3.2. *Squintability*

This section defines the projection distance and introduces the squintability metric, followed by a description of two numerical approaches for computing squintability.

Definition 4 (projection distance)**.** *Let $A \in \mathbb{R}^{p\times d}$ be a d-dimensional orthonormal matrix, and let A^* be the optimal matrix that achieves the maximum index value for a given data. The projection distance between A and A^*, $r(A, A^*)$, is defined by $r(A, A^*) = \|AA' - A^*A^{*\prime}\|_F$ where $\|.\|_F$ denotes the Frobenius norm, given by $\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}$.*

Definition 5 (squint angle)**.** *Let A and B be two d-dimensional orthonormal matrices in $\mathbb{R}^p$. The squint angle θ between the subspace spanned by A and B is defined as the smallest principal angle between these subspaces: $\theta = \min_{i\in\{1,\cdots,d\}} \arccos(\tau_i)$, where τ_i are the singular values of the matrix $M = A^TB$ obtained from its singular value decomposition.*

Squintability can be defined based on how the index value $f(XA)$ changes with respect to the projection distance $r(A, A^*)$, over the course of the JSO. Suppose that the optimization starts with an arbitrary candidate matrix A_0 at a distance $r_0 := r(A_0, A^*)$ from the optimal one. A possible way to define squintability is as follows:

Definition 6 (squintability)**.** *Let $g : \mathbb{R} \mapsto \mathbb{R}$ be a decreasing function that maps the projection distance $r(A, A^*)$ to the index value $f(XA)$, such that $g(r) = g(r(A, A^*)) = f(XA)$. The squintability of an index function f is defined by*

$$\varsigma(f) = \frac{g(r_0/2) - g(r_0)}{g(0) - g(r_0)} \in [0, 1] \tag{5}$$

Remark that the amount by which the index value can improve from the starting matrix A_0 to the optimal one A^* is given by $g(0) - g(r_0)$. As a result, equation (5) represent the proportion of this maximum improvement which has been achieved by the time the distance r_0 to the optimal matrix has been reduced by half ($r_0/2$).

It is expected that this value should be high in the case of high squintability (fast increase in g early on), and low in the case of low squintability (any substantial increase in g happens very late, close to the optimal angle). This suggests that this half-point improvement metric should provide a sensible measure of squintability.

From Tukey and Tukey (1981) and Laa and Cook (2020), a large squint angle implies that the objective function value is close to optimal even when the perfect view to see the structure is far away. A small squint angle means that the PP index value improves substantially only when the perfect view is close by. As such, low squintability implies rapid improvement in the index value when near the perfect view. For PP, a small squint angle is considered to be undesirable because it means that the optimiser needs to be very close to be able to "see" the optimum. Thus, it could be difficult for the optimiser to find the optimum.

It is expected that for a PP index with high squint angle, the optimization (1) should make substantial progress early on. Conversely, for a PP index with low squint angle, it might take a long while for the optimization to make substantial progress, as the candidate projections would need to be very close to the optimal one for the structure of the index function to be visible enough to be amenable to efficient optimization. This observation suggests that the half-point index value improvement percentage provides an appropriate mathematical definition of squintability, which matches the intuition behind this concept, while being amenable to numerical computation.

To compute the squintability metric (5) in practice, several approaches are possible. The first one is to propose a parametric model for g, and use it to obtain an explicit formula for ς. Numerical experiments suggest a scaled sigmoid shape as described below. Define

$$\ell(x) := \frac{1}{1 + \exp(\theta_3(x - \theta_2))} \ , \tag{6}$$

which is a decreasing logistic function depending on two parameters θ_2 and θ_3, such that $\ell(\theta_2) = \frac{1}{2}$. Then, define

$$g(x) = (\theta_1 - \theta_4)\frac{\ell(x) - \ell(r_0)}{\ell(0) - \ell(r_0)} + \theta_4 \ , \tag{7}$$

which depends on three additional parameters, θ_1, θ_2, and r_0, such that $g(0) = \theta_1$ and $g(r_0) = \theta_4$. Under the parametric model (7), the squintability metric (5) can be shown to be equal to

$$\varsigma = \frac{g\,(r_0/2) - \theta_4}{\theta_1 - \theta_4} = \frac{\ell(r_0/2) - \ell(r_0)}{\ell(0) - \ell(r_0)} \ . \tag{8}$$

In practice, the parameters of this model (7) can be estimated numerically, for example by non-linear least squares, and then used to evaluate ς as in equation (8).

Alternatively, one can estimate (5) in a nonparametric way, for example by fitting g using kernel regression, then numerically estimate ς from its definition (5).

4. The jellyfish optimiser

The Jellyfish Search Optimiser (JSO) mimics the natural movements of jellyfish, which include passive and active motions driven by ocean currents and their swimming patterns, respectively. In the context of optimization, these movements are abstracted to explore the search space, aiming to balance exploration (searching new areas) and exploitation (focusing on promising areas). The algorithm aims to find the optimal solution by adapting the behaviour of jellyfish to navigate towards the best solution over iterations (Chou and Truong 2021).

To solve the optimisation problem embedded in the PP guided tour, a starting projection, an index function, the number of jellyfish, and the maximum number of trials (tries) are provided as input. Then, the current projection is evaluated by the index function. The projection is then moved in a direction determined by a random factor, influenced by how far along we are in the optimisation process. Occasionally, completely new directions may be taken like a jellyfish might with ocean currents. A new projection is accepted if it is an improvement compared to the current one, rejected otherwise. This process continues and iteratively improves the projection, until the pre-specified maximum number of trials is reached.

Algorithm: Jellyfish Optimizer Pseudo Code

Input: `current_projections`, `index_function`, `trial_id`, `max_trial`

Output: `optimised_projection`

Initialize `current_best` as the projection with the best index value from `current_projections`, and `current_idx` as the array of index values for each projection in `current_projections`

for each `trial_id` in 1 to `max_tries` **do**

Calculate the time control value, c_t, based on `current_idx` and `max_trial`

if c_t is greater than or equal to 0.5 **then**

Define trend based on the `current_best` and `current_projections`

Update each projection towards the trend using a random factor and orthonormalisation

else

if a random number is greater than $1 - c_t$ **then**

Slightly adjust each projection with a small random factor (passive)

else

For each projection, compare with a random jellyfish and adjust towards or away from it (active)

Update the orientation of each projection to maintain consistency

Evaluate the new projections using the index function

if any new projection is worse than the current, revert to the `current_projections` for that case

Determine the projection with the best index value as the new `current_best`

if `trial_id` $\geq$ `max_trial`, print the last best projection **exit**

return the set of projections with the updated `current_best` as the `optimised_projection`

The JSO implementation involves several key parameters that control its search process in optimization problems. These parameters are designed to guide the exploration and

exploitation phases of the algorithm. While the specific implementation details can vary depending on the version of the algorithm or its application, the focus is on two main parameters that are most relevant to our application: the number of jellyfish and the maximum number of tries.

5. Assessing the optimisers

This section explains the details of two simulation studies: (1) comparison between the existing Creeping Random Search (CRS) (Zhang et al. 2021; Laa and Cook 2020) and JSO, and (2) examining the factors affecting the performance of JSO.

The second simulation examines various factors affecting the performance of JSO. The JSO performance is evaluated under different hyper-parameters combinations (number of jellyfish and the maximum number of tries). The effect of index properties (smoothness and squintability) along with JSO hyper-parameters and data dimension variations is captured and studied using logistic regression.

5.1. *Performance of JSO relative to CRS*

The CRS is the main optimisation routine currently used for the guided tour. Here the two optimisers are compared on the task of finding the pipe using the `holes` index in data with dimensions, $d = 6, 8, 10, 12$. Fifty simulated data sets are used for each dimension. JSO uses 100 jellyfish with a maximum of 100 tries, while the CRS allows a maximum of 1000 tries at each iteration before the algorithm terminates. These choices enable fair comparison between CRS and JSO, that conforms to how they are used in practice.

The performance of the optmisers is measured by the success rate, which is defined as the proportion of simulations that achieves a final index value within 0.05 of the best index value found among all 50 simulations. Figure 4 illustrates why the choice of 0.05 is reasonable: the pipe is not recognisable in the projected data. This is motivated by Laa and Cook (2020)'s approach to investigating PP indexes.

The results of the simulation are collected using the data structure used in Zhang et al. (2021) for assessing PP optimisers. The design parameters are stored along with index value, projection basis, random seed, and computation time.

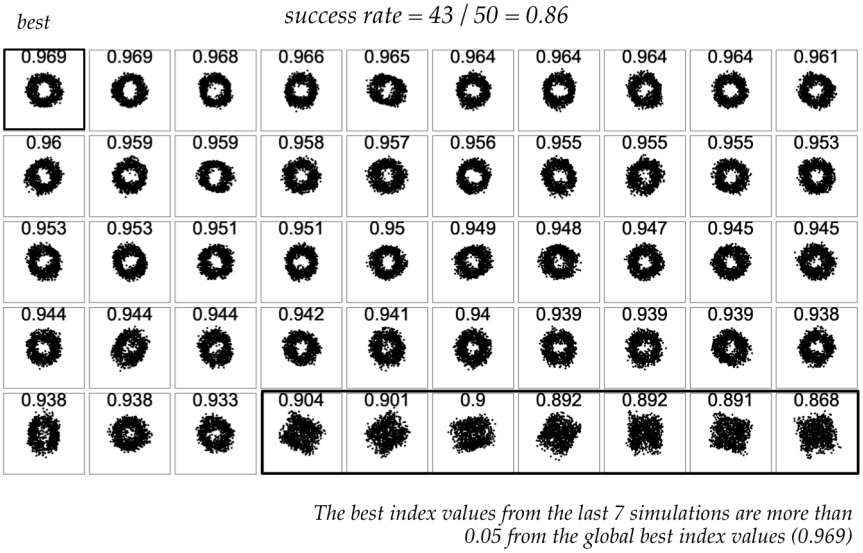

Figure 4. How success rate is calculated, illustrated using the optimal projections from 50 optimisations of 8D pipe data sorted by index value. The pipe shape is recognisable in the projection index values between 0.933-0.969. Of the 50 simulations, 43 achieved an index value within 0.05 of the best, resulting in a success rate of 0.86.

5.2. *Factors affecting JSO success rate: JSO hyper-parameters and index properties*

To examine the performance of JSO across various hyper-parameter combinations, the pipe-finding problem using the holes index is repeated for fifty times in each hyper-

parameter combination: 20, 50, and 100 jellyfish and a maximum of 50 and 100 attempts.

In addition, to assess the performance of JSO across various PP problems, two different data shapes, pipe and a sine wave, are considered in various dimensional spaces using six different PP indexes: `dcor2d_2`, `loess2d`, `MIC`, `TIC`, `spline`, and `stringy`, under varied JSO hyper-parameters. An example of holes and pipes data with the syntax to generate them are provided in the supplementary material. This results in a total of 76 scenarios, comprising of 30 computed on the pipe data and 46 on the sine-wave data. This study does not consider dimensions lower than four, as exploring the search space in such cases is relatively straightforward. Similarly, extremely high-dimensional cases with complex data structures, where no method can reliably and consistenly uncover the underlying structure due to the curse of dimensionality, are also excluded, as they fall outside the scope of this study. A list of combinations of the data shape, dimensions and PP indexes that are considered in this study is provided in Table 1. Again, JSO is run 50 times for each scenario to calculate the success rate for each hyper-parameter combination.

Smoothness and squintability are computed following the procedures outlined in Section 3.1 and Section 3.2 and as illustrated in Figure 5 and Figure 6. To compute smoothness, 500 random bases are simulated. Index values are calculated for each random basis, followed by fitting a Gaussian process model to obtain the smoothness measure for the index.

To compute squintability, 50 random bases are sampled and interpolated to the optimal basis with a step size of 0.005. Index values and projection distances are calculated for these interpolated bases. A binning procedure is applied to average the index values within each 0.005 projection distance bin. The four-parameter scaled logistic function

(7) is fitted to the index values against projection distances, estimated by non-linear least squares to obtain the squintability measure as Equation (8).

A generalised linear model is fitted using a binomial family and a logit link function to assess the factors affecting success rate. Predictors are smoothness, squintability, and JSO hyper-parameters. A success rate of 0 indicates the optimiser did not come sufficiently close to the optimal projection.

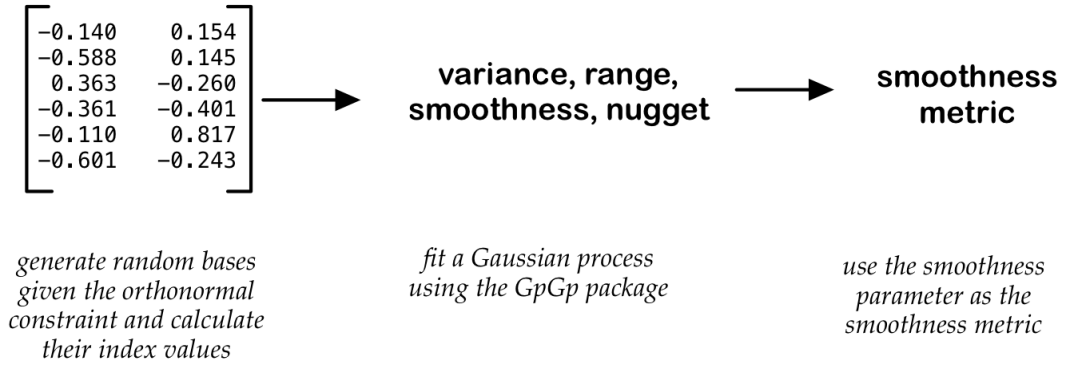

Figure 5. Steps for calculating smoothness in a projection pursuit problem. Given the target shape, data dimension and the index function: 1) sample random bases given the orthonormality contraint and calculate their corresponding index values, and 2) fit a Gaussian process model of index values against the sampled bases to obtain the smoothness measure.

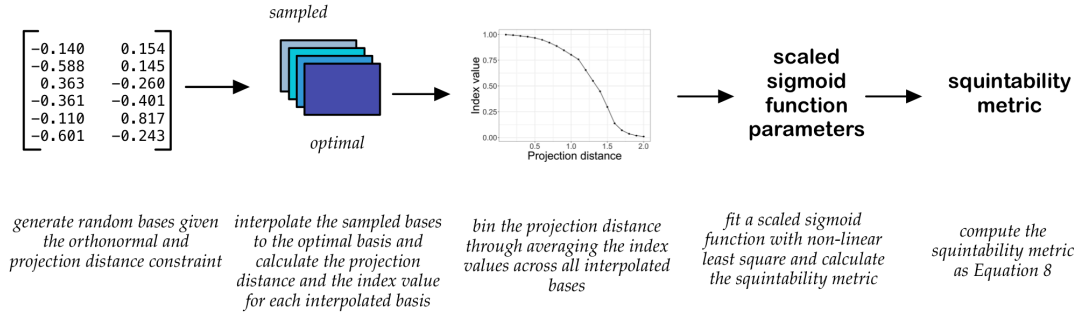

Figure 6. Steps for calculating squintability in a projection pursuit problem. Given the target shape, data dimension and the index function: 1) sample random bases given the orthonormality and projection distance contraint and calculates their corresponding index values, 2) interpolate the sampled bases to the optimal basis and calculate the projection distance and the index value. 3) bin the index values by projection distances through averaging index value, 4) fit the scaled sigmoid function in equation (5) to the binned index values against projection distances using non-linear least square to obtian the squintability measure using equation (8).

6. Results

This section summarises the findings from the simulations that compare the JSO performance with the existing CRS optimiser, and the relationship between optimisation success and hyper-parameter choices and PP index properties.

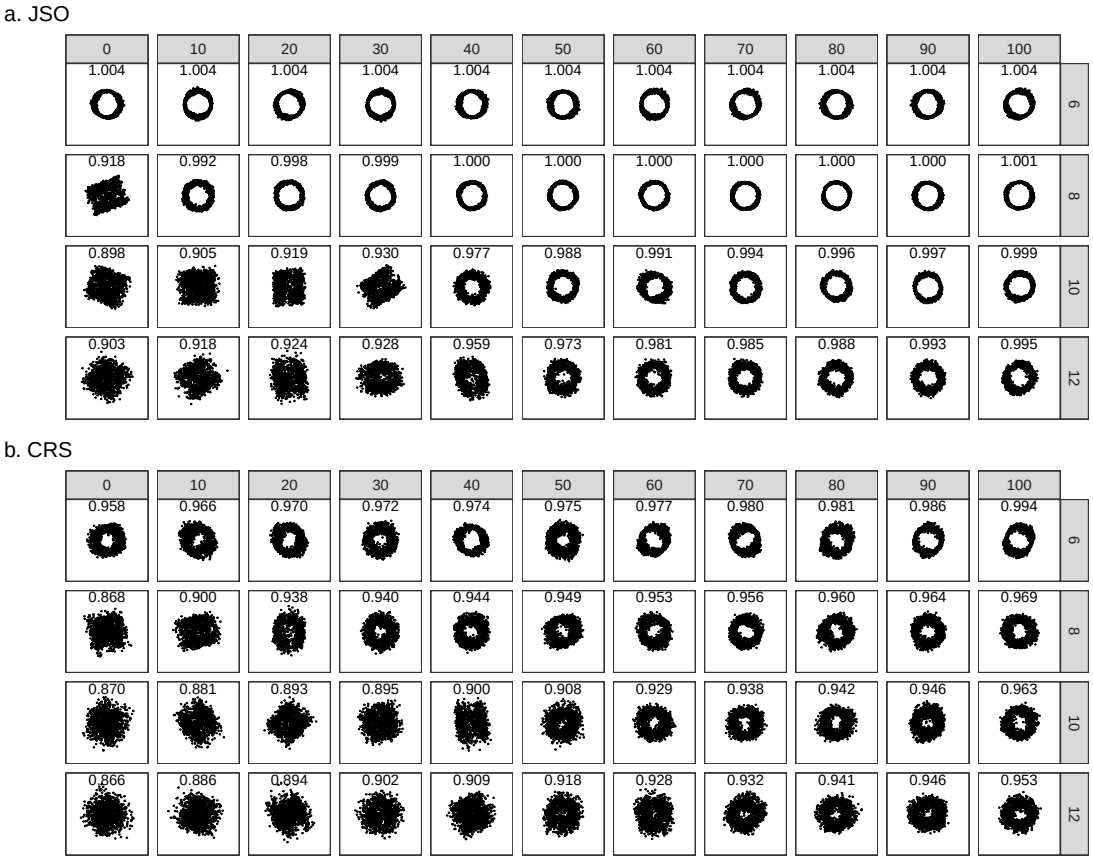

Figure 7. Visual comparison of JSO and CRS results, using optimal data projections obtained over 50 simulations of the pipe data. Rows correspond to data dimension. Columns correspond to quantiles of the index values, with 0 being minimum, 100 being maximum, and 50 the median. JSO achieves the better views of the pipe generally than CRS. As dimension increases both have more difficulty finding the pipe.

6.1. *Performance of JSO relative to CRS*

The performance of JSO is assessed relative to the existing CRS optimiser based on the optimal projection obtained. The final projections for the pipe data sets found by the

two optimisers are shown in Figure 7, with rows corresponding to data dimension. The 50 simulations in each data dimension are sorted by index value, and the projections corresponding to each 10th quantile value are shown, with minimum at left and maximum at right. Index value is printed on the plot. The purpose is to summarise the views of the data resulting from different optimisations, and hence compare results between the two optimisers. Generally, the JSO does a more consistent job of finding the pipe structure clearly. As the data dimension increases, both optimisers struggle and less clearly capture the circle shape.

6.2. *Effect of hyper-parameters effect on JSO success rate*

The effect of JSO hyper-parameters (number of jellyfish and the maximum number of tries) on the success rate is presented in Figure 8. The uncertainty is quantified through 500 bootstrap replicates for each case. As the number of jellyfish and maximum tries increase, the success rate also increases. For problems with relatively lower dimensional search spaces (4 and 6 dimensions), small parameter values (20 jellyfish and a maximum of 50 tries) are sufficient for space exploration and thus can already achieve a high success rate. Higher parameter values (i.e. 100 jellyfish and a maximum of 100 tries) enhances exploration of the search space, which is particularly beneficial in higher-dimensional problems (8, 10, and 12 dimensions) where a more complex search is necessary. However, in lower-dimensional problems, the benefit may be less pronounced, as the space is already relatively easy to navigate. While increasing both parameters enhances the performance of JSO, it also extends the computational time required for the optimisation, which can be computationally intensive when evaluating the index function (such as scagnostic indexes) multiple times across numerous iterations.

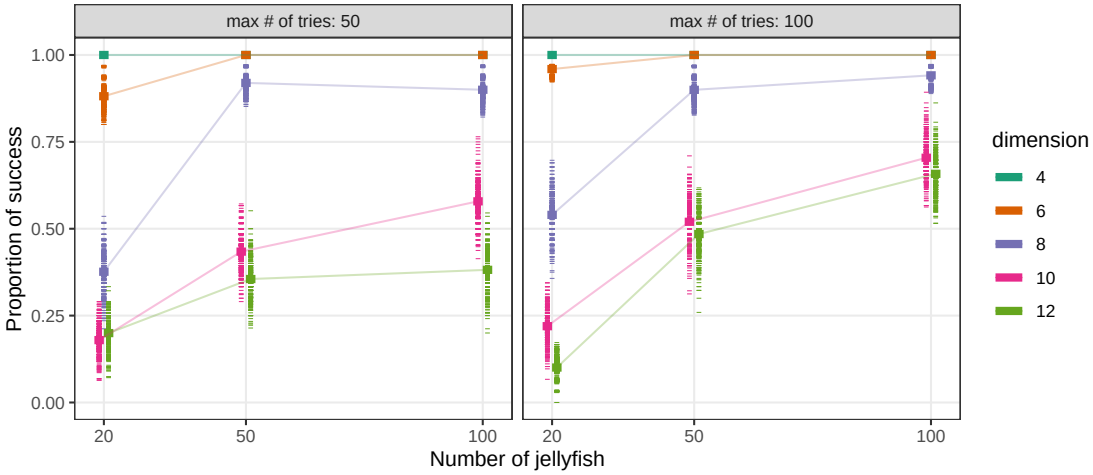

Figure 8. Summary of the relationship between JSO hyper-parameters on optimisation success, using the holes index for pipe data of dimensions 4-12. Bootstrap samples show the variability in success rate. Success rate mostly plateaus by 50 jellies. The JSO has some difficulty finding the pipe when dimension is higher than 8! Maximum number of tries has little effect.

6.3. *Effect of index properties on JSO success rate*

Smoothness and squintability are calculated across a collection of pipe-finding and sine-wave finding problems to construct the relationship between success rate, JSO hyper-parameters, and index properties. Table 1 presents the parameters estimated from the Gaussian process (outputscale η, lengthscale ℓ, smoothness ν, and nugget σ) and from the scaled logistic function (θ_1 to θ_4) for calculating smoothness and squintability in each PP problem considered. The column ν is used as the smoothness metric and the column ς is calculated as equation (8) as the squintability metric.

Table 2 presents the results from fitting a logistic regression model using the proportion of success as the response and smoothness, squintability, dimension, number of jellyfish and maximum number of tries, as predictors. The fit suggests that the JSO success rate is only affected by squintability, dimension and number of jellies. As expected, the JSO success rate is higher with higher squintability and/or more jellyfish, and lower when dimension is higher (higher-dimensional optimization is more difficult). Interestingly,

Table 1. Parameters estimated from the Gaussian process (outputscale η, lengthscale ℓ, smoothness ν, and nugget σ) and scaled logistic function (θ_1 to θ_4 and ς) for the pipe-finding and sine-wave finding problems. The columns ν and ς represent the smoothness and squintability measures respectively.

shape	index	d	ν	ς
sine	splines	4	3.5489	0.5723
sine	splines	6	3.1149	0.5415
sine	splines	8	2.7400	0.5394
sine	TIC	4	3.4854	0.3839
sine	TIC	8	2.7223	0.3785
sine	TIC	6	3.0807	0.3747
sine	skinny	6	2.2352	0.0973

the JSO is unaffected by the smoothness of the index function. This is consistent with the way random search algorithms jump from value to value without taking local regularity into account, as opposed to gradient-based optimizers for example. Allowing for more tries also does not affect success rate significantly. A unit increase in squintability increases the success rate by 24%. As dimension increases by one the success rate almost halves. Increasing the number of jellies by 10 increases the success rate by 32%.

Table 2. Jellyfish success rate relative to index properties and jellyfish hyper-parameters. This is the summary from a logistic regression fit to smoothness, squintability, dimension, number of jellyfish and maximum number of tries. Interestingly, squintability and dimension strongly affect jellyfish optimisation success. The number of jellies marginally affects success, but index smoothness, and increasing the number of tries do not.

Characteristic	**OR (95% CI)**[1]	**p-value**
Smoothness	1.00 (0.90 to 1.11)	0.928
Squintability	1.24 (1.13 to 1.39)	**<0.001**
Dimension	0.57 (0.44 to 0.73)	**<0.001**
Number of jellyfish	1.32 (1.17 to 1.50)	**<0.001**
Maximum number of tries	1.06 (0.94 to 1.21)	0.333

[1]OR = Odds Ratio, CI = Confidence Interval

Table 3. Fit statistics for the model.

Deviance	DF Residual	Null Deviance	DF Null
20.69	70	49.29	75

The simulation result suggests that squintability and smoothness are likely dependent. Intuitively, a highly smooth search space exhibits gradual changes, making it easier to perceive the overall structure from a distance, thereby increasing the squint angle. Table 3 compare the model fits when squintability and smoothness are included separately. Given that the null deviance remains the same across models, the model with squintability yields a smaller residual deviance and lower AIC and BIC values, indicating a better fit. This suggests that squintability captures the relevant variation more effectively than smoothness. While it may seem counterintuitive at first that smoothness is not a significant factor, this outcome is reasonable in this context, as the swarm-based method used in this study, JSO, is designed to perform well even in non-smooth search spaces by leveraging population-based exploration.

This study intends to serve as an illustration that the proposed measure effectively captures the key factors influencing the success rate of PP. The interactions between the two variables were not examined in this study due to the limited number of data points, nor was the linearity assumption of the relationship with success rate explicitly tested.

7. Practicalities

The simulation studies have compared the JSO with CRS and shown how smoothness and squintability affect the JSO success rate. Here we explain how they can be used

for new index and optimiser development. Using the JSO optimiser for PPGT requires a change in user behaviour. For all the existing optimisers a single optimisation path is followed. The JSO has multiple optimisation paths, and needs additional tools to incorporate into the PPGT, as explained here.

7.1. *Using the JSO in a PPGT*

To use JSO for PPGT in the `tourr` package, specify `search_f = search_jellyfish` in the guided tour. Unlike existing optimisers, the animation function `animate_*()` won't directly render the tour path for JSO due to the generation of multiple tour paths. To visualise the path visited by each individual jellyfish, assign the animation to an object (`res <- animate_xy(...)`). This will save the bases visited by JSO, along with relevant metadata, as a tibble data object (see Zhang et al. 2021 for more details on the data object). The bases visited for individual jellyfish can then be extracted and viewed using the `planned_tour()` function.

7.2. *Computing the index properties for your new index*

The `ferrn` package (Zhang et al. 2021) provides functionality for computing the smoothness and squintability metrics. Both metrics require sampling random bases using the `sample_bases()` function, followed by the metric calculation with `calc_smoothness()` or `calc_squintability()`.

7.2.1. Smoothness

To sample bases for calculating smoothness, the following arguments are required: the index function, the dataset, and the number of random bases to sample. The output of

`sample_bases()` is a data frame with a list-column of sampled basis matrix and index value. Parallellasation is available to speed up the index value computation through the `parallel = TRUE` argument.

The `calc_smoothness()` function takes the output from `sample_bases()` and fits a Gaussian process model to the index values against the sampled basis, as the location, to obtain the smoothness metric. Starting parameters and additional Gaussian process arguments can be specified and details of the fit can be accessed through the `fit_res` attribute of the output.

7.2.2. Squintability

Bases sampling for squintability includes an additional step of interpolating between the sampled bases and the best basis. This step is performed when the arguments `step_size` and `min_proj_dist` in the `sample_bases()` function are set to non-NA numerical values. Given the projection distance typically ranging from 0 to 2, it is recommended to set `step_size` to 0.1 or lower, and `min_proj_dist` to be at least 0.5 to ensure a meaningful interpolation length.

The `calc_squintability()` function computes squintability using two methods: 1) parametrically, by fitting a scaled sigmoid function through non-linear least square (`method = "nls"`), and 2) non-parametrically, using kernel smoothing (`method = "ks"`). A `bin_width` argument is required to average the index values over the projection distance before the fit. For the parametric case, the output provides the estimated parameters for the scaled sigmoid function (θ_1 to θ_4) and the calculated squintability metric as Equation (8). For the non-parametric case, it shows the maximum gradient attained (`max_d`), the corresponding projection distance (`max_dist`), and the squintability

metric as their products.

8. Conclusion

This paper has presented new metrics to mathematically define desirable features of PP indexes, squintability and smoothness, and used these to assess the performance of the new jellyfish search optimiser. The metrics will be generally useful for characterising PP indexes, and help with developing new indexes.

In the comparison of the JSO against the currently used CRS, as expected the JSO vastly outperforms CRS, and provides a high probability of finding the global optimum. The JSO obtains the maximum more cleanly, with a slightly higher index value, and plot of the projected data showing the structure more clearly.

The JSO performance is affected by the hyper-parameters, with a higher chance of reaching the global optimum when more jellyfish are used and the maximum number of tries is increased. However, it comes at a computational cost, as expected. The performance declines if the projection dimension increases and if the PP index has low squintability. The higher the squintability the better chance the JSO can find the optimum. However, interestingly smoothness does not affect the JSO performance.

One future direction of this work is to test the JSO, along with its variations and other swarm-based optimisers, on a broader range of indexes, for example, scagnostic indexes.

9. Acknowledgement

The article has been created using Quarto (Allaire et al. 2022) in R (R Core Team 2023). The source code for reproducing the work reported in this paper can be found at: https:

//github.com/huizezhang-sherry/paper-jso. The simulation data produced in Section 5 can be found at https://figshare.com/articles/dataset/Simulated_raw_data/26039506. Nicolas Langrené acknowledges the partial support of the Guangdong Provincial/Zhuhai Key Laboratory IRADS (2022B1212010006) and the UIC Start-up Research Fund UICR0700041-22.

The R packages used in this work include: `tidyverse` (Hadley Wickham et al. 2019); `patchwork` (Pedersen 2024); `ggh4x` (van den Brand 2024); `broom` (Robinson, Hayes, and Couch 2024); `kableExtra` (Zhu 2024); `ferrn` (Zhang et al. 2021); and `cassowaryr` (Mason et al. 2022).

Supplementary materials

The supplementary materials available at https://github.com/huizezhang-sherry/paper-jso include: 1) details of the indexes used in the simulation study, 2) the script to get started with using JSO in a PPGT and calculating smoothness and squintability, as explained in Section 7, and (3) the full code to reproduce the plots and summaries in this paper.

References

Abdulah, Sameh, Yuxiao Li, Jian Cao, Hatem Ltaief, David Keyes, Marc Genton, and Ying Sun. 2023. "Large-Scale Environmental Data Science with ExaGeoStatR." *Environmetrics* 34 (1): e2770. https://doi.org/10.1002/env.2770.

Adams, Robert, and John Fournier. 2003. *Sobolev Spaces*. Vol. 140. Pure and Applied Mathematics. Elsevier.

Allaire, J. J., Charles Teague, Carlos Scheidegger, Yihui Xie, and Christophe Dervieux. 2022. *Quarto* (version 1.2). https://doi.org/10.5281/zenodo.5960048.

Bertsimas, Dimitris, and John Tsitsiklis. 1993. "Simulated Annealing." *Statistical Science* 8 (1): 10–15. https://doi.org/10.1214/ss/1177011077.

Chou, Jui-Sheng, and Dinh-Nhat Truong. 2021. "A Novel Metaheuristic Optimizer Inspired by Behavior of Jellyfish in Ocean." *Applied Mathematics and Computation* 389 (January): 125535. https://doi.org/10.1016/j.amc.2020.125535.

Cook, D., A. Buja, and J. Cabrera. 1993. "Projection Pursuit Indexes Based on Orthonormal Function Expansions." *Journal of Computational and Graphical Statistics* 2 (3): 225–50. https://doi.org/10.2307/1390644.

Cook, D., A. Buja, J. Cabrera, and C. Hurley. 1995. "Grand Tour and Projection Pursuit." *Journal of Computational and Graphical Statistics* 4 (3): 155–72. https://doi.org/10.1080/10618600.1995.10474674.

DLMF. 2024. "NIST Digital Library of Mathematical Functions." https://dlmf.nist.gov/10.25.

Friedman, J. H., and J. W. Tukey. 1974. "A Projection Pursuit Algorithm for Exploratory Data Analysis." *IEEE Transactions on Computers* C-23 (9): 881–90. https://doi.org/10.1109/T-C.1974.224051.

Grimm, Katrin. 2016. "Kennzahlenbasierte Grafikauswahl." Doctoral thesis, Universität Augsburg.

Grochowski, Marek, and Włodzisław Duch. 2011. "Fast Projection Pursuit Based on Quality of Projected Clusters." In *Adaptive and Natural Computing Algorithms*, edited by Andrej Dobnikar, Uroš Lotrič, and Branko Šter, 89–97. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-20267-4_10.

Guinness, Joseph, Matthias Katzfuss, and Youssef Fahmy. 2021. "GpGp: Fast Gaussian

Process Computation Using Vecchia's Approximation." R package.

Hall, Peter. 1989. "On Polynomial-Based Projection Indices for Exploratory Projection Pursuit." *The Annals of Statistics* 17 (2): 589–605. https://doi.org/10.1214/aos/1176347127.

Huber, Peter J. 1985. "Projection Pursuit." *The Annals of Statistics* 13 (2): 435–75. https://doi.org/10.1214/aos/1176349519.

———. 1990. "Data Analysis and Projection Pursuit." Technical Report PJH-90-1. Dept. of Mathematics, Massachusetts Institute of Technology.

Karvonen, Toni. 2023. "Asymptotic Bounds for Smoothness Parameter Estimates in Gaussian Process Interpolation." *SIAM/ASA Journal on Uncertainty Quantification* 11 (4): 1225–57. https://doi.org/10.1137/22M149288X.

Kruskal, J. B. 1969. "Toward a Practical Method Which Helps Uncover the Structure of a Set of Observations by Finding the Line Transformation Which Optimizes a New 'Index of Condensation'." In *Statistical Computation*, edited by R. C. Milton and J. A. Nelder, 427–40. New York: Academic Press. https://doi.org/10.1016/B978-0-12-498150-8.50024-0.

Laa, Ursula, and Dianne Cook. 2020. "Using Tours to Visually Investigate Properties of New Projection Pursuit Indexes with Application to Problems in Physics." *Computational Statistics* 35 (3): 1171–1205. https://doi.org/10.1007/s00180-020-00954-8.

Lee, Eun-Kyung, and Dianne Cook. 2010. "A Projection Pursuit Index for Large p Small n Data." *Statistics and Computing* 20 (3): 381–92. https://doi.org/10.1007/s11222-009-9131-1.

Lee, Eun-Kyung, Dianne Cook, Sigbert Klinke, and Thomas Lumley. 2005. "Projection Pursuit for Exploratory Supervised Classification." *Journal of Computational and Graphical Statistics* 14 (4): 831–46. https://doi.org/10.1198/106186005X77702.

Loperfido, Nicola. 2018. "Skewness-Based Projection Pursuit: A Computational Approach." *Computational Statistics and Data Analysis* 120 (C): 42–57. https://doi.org/10.1016/j.csda.2017.11.001.

———. 2020. "Kurtosis-Based Projection Pursuit for Outlier Detection in Financial Time Series." *The European Journal of Finance* 26 (2-3): 142–64. https://doi.org/10.1080/1351847X.2019.1647864.

Marie-Sainte, Souad Larabi, Alain Berro, and Anne Ruiz-Gazen. 2010. "An Efficient Optimization Method for Revealing Local Optima of Projection Pursuit Indices." In *Swarm Intelligence*, edited by Marco Dorigo, Mauro Birattari, Gianni A. Di Caro, René Doursat, Andries P Engelbrecht, Dario Floreano, Luca Maria Gambardella, et al., 60–71. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-15461-4_6.

Mason, Harriet, Stuart Lee, Ursula Laa, and Dianne Cook. 2022. *Cassowaryr: Compute Scagnostics on Pairs of Numeric Variables in a Data Set*. https://CRAN.R-project.org/package=cassowayr.

Pedersen, Thomas Lin. 2024. *Patchwork: The Composer of Plots*. https://CRAN.R-project.org/package=patchwork.

Porcu, Emilio, Moreno Bevilacqua, Robert Schaback, and Chris Oates. 2024. "The Matérn Model: A Journey Through Statistics, Numerical Analysis and Machine Learning." *Statistical Science* 39 (3): 469–92. https://doi.org/10.1214/24-STS923.

Posse, Christian. 1995. "Projection Pursuit Exploratory Data Analysis." *Computational Statistics and Data Analysis* 20 (6): 669–87. https://doi.org/10.1016/0167-9473(95)00002-8.

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Rajwar, Kanchan, Kusum Deep, and Swagatam Das. 2023. "An Exhaustive Review of the Metaheuristic Algorithms for Search and Optimization: Taxonomy, Applications, and Open Challenges." *Artificial Intelligence Review*, 1–71. https://doi.org/10.1007/s10462-023-10470-y.

Rasmussen, Carl Edward, and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. The MIT Press.

Robinson, David, Alex Hayes, and Simon Couch. 2024. *Broom: Convert Statistical Objects into Tidy Tibbles*. https://CRAN.R-project.org/package=broom.

Tukey, P. A., and J. W. Tukey. 1981. *Graphical Display of Data in Three and Higher Dimensions*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. Wiley. https://books.google.com.au/books?id=WBzvAAAAMAAJ.

Ursula Laa, Andreas Buja, Dianne Cook, and German Valencia. 2022. "Hole or Grain? A Section Pursuit Index for Finding Hidden Structure in Multiple Dimensions." *Journal of Computational and Graphical Statistics* 31 (3): 739–52. https://doi.org/10.1080/10618600.2022.2035230.

van den Brand, Teun. 2024. *Ggh4x: Hacks for 'Ggplot2'*. https://CRAN.R-project.org/package=ggh4x.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, H., D. Cook, H. Hofmann, and A. Buja. 2011. "tourr: An R Package for Exploring Multivariate Data with Projections." *Journal of Statistical Software* 40 (2): 1–18. https://doi.org/10.18637/jss.v040.i02.

Wilkinson, L., A. Anand, and R. Grossman. 2005. "Graph-Theoretic Scagnostics." In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, 157–64. https://doi.org/10.1109/INFVIS.2005.1532142.

Wilkinson, Leland, and Graham Wills. 2008. "Scagnostics Distributions." *Journal of Computational and Graphical Statistics* 17 (2): 473–91. https://doi.org/10.1198/106186008X320465.

Zhang, H. Sherry, Dianne Cook, Ursula Laa, Nicolas Langrené, and Patricia Menéndez. 2021. "Visual Diagnostics for Constrained Optimisation with Application to Guided Tours." *The R Journal* 13: 624–41. https://doi.org/10.32614/RJ-2021-105.

Zhu, Hao. 2024. *kableExtra: Construct Complex Table with 'Kable' and Pipe Syntax*. https://CRAN.R-project.org/package=kableExtra.