Studying the Performance of the Jellyfish Search Optimiser for the Application of Projection Pursuit

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

ARTICLE HISTORY
Compiled September 9, 2024

[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 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 criteria 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 optima has a small "squint angle", detectable only from close proximity. To address these challenges, this study investigates 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. Additionally, methods for calculating the smoothness and squintability properties of the PP index are proposed. They are used to assess the optimiser performance in the presence of PP index complexities. A simulation study illustrates the use of these performance metrics to compare the JSO with existing optimisation methods available for the guided tour. The JSO algorithm has been implemented in the R package, `tourr`, and functions to calculate smoothness and squintability are available in the `ferrn` package.

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

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.

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 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), Loperfido (2020)), which defines the criteria 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, non-linear structure, or other features of interest. 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 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.

Optimisation of PP is often discussed when new indexes are proposed (Posse 1995a; 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 optima. An implementation is available in the `tourr` package (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 optimi-

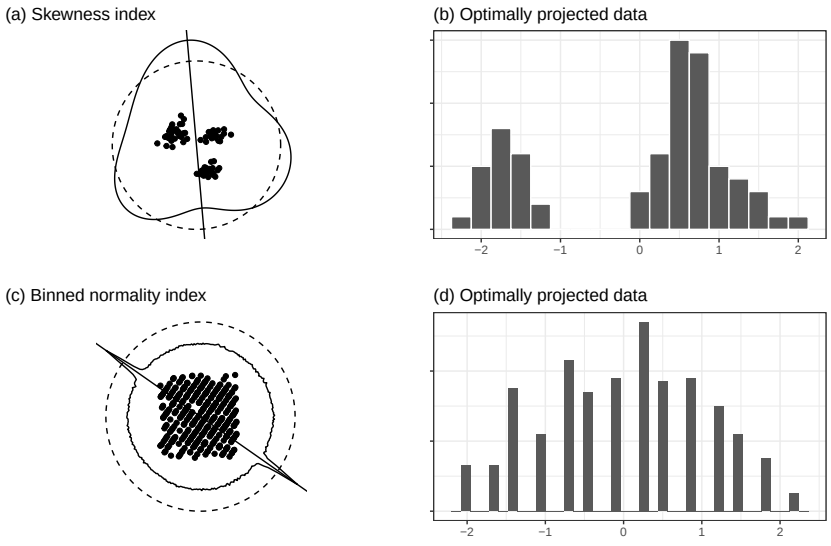

Figure 1. Examples of PP indexes with large (top row) and small (bottom row) squint angles, shown with a Huber plot, and histogram of the projected data corresponding to the optimal projection. A Huber plot shows the PP index values for all 1D data projections in polar coordinates.

sation 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) (see Chou and Truong (2021), and 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 gobal optima, 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 describes JSO and introduces the metrics that measure different properties of PP indexes, smoothness and squintability. Section 4 summarises the results of two simulation experiments done 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 different PP index properties on optimisation performance. Section 5 presents the results. Section 6 discusses implementation details and insights for practitioners. Section 7 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 (unsophisticated) optimisation methods implemented in the `tourr` package:

- `search_geodesic()`: provides a pseudo-derivative optimisation. It searches locally for the best direction, based on differencing the index values for very close projections. Then it follows the direction along the geodesic path between planes, stopping when the next index value fails to increase.
- `search_better()`: also known as Creeping Random Search (CRS), is a brute-force optimisation searching randomly for projections with higher index values.
- `search_better_random()`: is essentially simulated annealing (Bertsimas and Tsitsiklis 1993) where the search space is reduced as the optimisation progresses.
- `search_posse()`: implements the algorithm described in Posse (1995b).
- `search_polish()`: is a very localised search, to take tiny steps to get closer to the local maximum.

There are several PP index functions available: `holes()` and `cmass()` (Cook, Buja, and Cabrera 1993); `lda_pp()` (Lee et al. 2005); `pda_pp()` (Lee and Cook 2010); `dcor2d()` and `splines2d()` (Grimm 2016); `norm_bin()` and `norm_kol()` (Huber 1985);

`slice_index()` (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 be able to incorporate more complex PP indexes, for example, based on scagnostics (L. Wilkinson, Anand, and Grossman (2005), Leland Wilkinson and Wills (2008)).

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 which 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 global because data can contain many different and interesting features.

3. The jellyfish optimiser and properties of PP indexes

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.

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 con-

tinuous derivatives of the function exist. This can be characterized by Sobolev spaces (Adams and Fournier 2003).

Definition 1. *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. *The Matérn covariance function K is defined by*

$$K(u) = K_{\nu,\eta,\ell}(u) := \eta^2 \frac{\left(\sqrt{2\nu}\frac{u}{\ell}\right)^{\nu}}{\Gamma(\nu)2^{\nu-1}} \mathcal{K}_{\nu}\left(\sqrt{2\nu}\frac{u}{\ell}\right) \ , u \geq 0 \tag{2}$$

where $\nu > 0$ is the smoothness parameter, η is the outputscale, ℓ is the lengthscale, and $\mathcal{K}_{\nu}$ is the modified Bessel function [DLMF 10.25]. A multivariate extension $K(u)$, $u \in \mathbb{R}^d$ can be obtained by products of univariate covariance functions (2).

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: $K_{\infty}(u) = \exp(-u^2/2)$.

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.

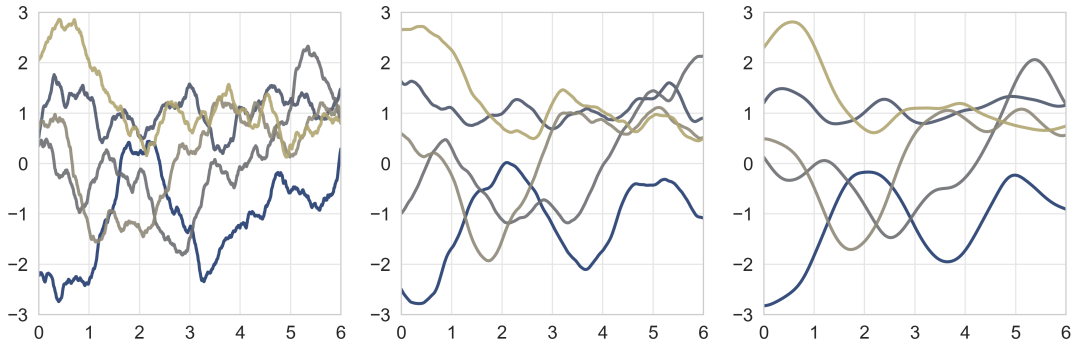

Figure 2. Five random simulations from a Gaussian Process defined on $\mathbb{R}$ with zero mean and Matérn-ν covariance function, with $\nu = 1$ (left), $\nu = 2$ (middle), and $\nu = 4$ (right), showing that higher values of ν produce smoother curves.

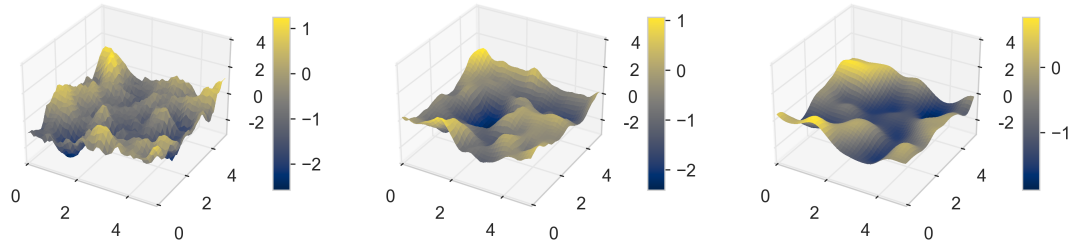

Figure 3. One random simulation from a Gaussian Process defined on $\mathbb{R}^2$ with zero mean and Matérn-ν covariance function, with $\nu = 1$ (left), $\nu = 2$ (middle), and $\nu = 4$ (right), showing that higher values of ν produce smoother surfaces.

2023), to fit the hyperparameters 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* $\mathbf{A} = [A_1, \dots, A_N] \in (\mathbf{R}^{p\times d})^N$ *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*

Here the formal definition of projection distance and squint angle are given, before the definition of squintability. Two approaches to compute this metric numerically, are

described.

Definition 4 (projection distance)**.** *Recall that $A \in \mathbf{R}^{p\times d}$ is a d-dimensional orthonormal matrix. Let A^* be the optimal matrix that achieves the maximum index value for a given data. The projection distance $d(A, A^*)$ between A and A^* is defined as $d(A, A^*) = \|AA' - A^*A^{*\prime}\|_F$ 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 as how the index value $f(XA)$ changes with respect to the projection distance $d(A, A^*)$, over the course of the JSO:

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

$$\varsigma(f) = -c \times \max_d g'(d) \times \arg\max_d g'(d) \tag{5}$$

where c is a constant scaling factor, $-\max_d g'(d)$ represents the largest gradient of $-g$

and $\arg\max_d g'(d)$ *represents the projection distance at which this largest gradient is attained.*

It is expected that these two values should be both high in the case of high squintability (fast increase in f early on), and both low in the case of low squintability (any substantial increase in f happens very late, close to the optimal angle). This suggests that their product (5) should provide a sensible measure of squintability. The multiplicative constant 4, which can be deemed arbitrary, does not change the interpretation of the squintability metric ς; it is here to adjust the range of values of ς and simplify the explicit formula for ς obtained later on.

From the definition, the following proposition can be derived:

Proposition 1. *Let* $g_1(d)$ *be a convex decreasing function and* $g_2(d)$ *be a concave decreasing function both defined on [0, 1] where* $d \in [0, D]$. *Let* $d_1 := \arg\max_d |g_1'(d)|$ *and* $d_2 := \arg\max_d |g_2'(d)|$.

$$\varsigma(g_1) < \varsigma(g_2)$$

For a convex function, $g_1'(d) < 0 \ \forall d$ *and is non-decreasing. For a concave function,* $g_2'(d) < 0 \ \forall d$ *and is non-increasing. As such, it follows that* $d_1 < d_2$.

Suppose both functions achieve the same maximum gradient $-g_{\max}$, *i.e.,* $\max g_1'(d_1) = \max g_2'(d_2) = -g_{\max}$,

$$\varsigma(g_1(d_1)) = -c \max g_1'(d_1) d_1 = -c(-g_{\max}) d_1 \leq -c(-g_{\max}) d_2 = -c \max g_2'(d_2) d_2 = \varsigma(g_2(d_2))$$

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 optima. Thus, it could be difficult for the optimiser to find the optima.

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 extreme values of f' (the ones for which $f'' = 0$, assuming that f is twice differentiable), and the projection distances for which these values are attained, are crucial in the mathematical definition of squintability.

To compute the squintability metric (5) in practice, several approaches are possible. The first one is to propose a parametric model for f, 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

$$f(x) = (\theta_1 - \theta_4)\frac{\ell(x) - \ell(x_{\max})}{\ell(0) - \ell(x_{\max})} + \theta_4 \ , \tag{7}$$

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

$$\varsigma = \frac{(\theta_1 - \theta_4)\theta_2\theta_3}{\ell(0) - \ell(x_{\max})} \ . \tag{8}$$

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 f using kernel regression, then numerically estimate the angle at which $-f'$ attains its highest value.

4. Simulation study

The JSO performance is compared with an existing optimiser, Creeping Random Search (CRS) (Zhang et al. 2021; Laa and Cook 2020) used in the PP guided tour to explore JSO's behaviour under different hyper-parameter and data dimension combinations. The second simulation studies the effect of index properties (smoothness and squintability), along with JSO hyper-parameters, and data dimension, on the success rate of the JSO performance. This section describes the simulation details, with the results deferred to Section 5.

4.1. *Performance of JSO relative to CRS*

The performance of JSO is investigated both in comparison to the existing optimizer, CRS, and across various hyper-parameter values. The performance is measured by the success rate, 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 (see Figure 4 for an illustration). This comparison is based on projection pursuit to find the pipe shape investigated by Laa and Cook (2020) using the `holes` index.

Fifty simulations are conducted with both JSO and CRS, in four different data dimensions ($d = 6, 8, 10, 12$). JSO uses 100 jellyfishes with a maximum of 100 tries, while the CRS allows a maximum of 1000 samples at each iteration before the algorithm terminates. The different numbers account for the multiple paths of JSO to enable fairer comparison with CRS. The results of the simulation are collected using the data structure proposed in Zhang et al. (2021) for assessing JSO, where the design parameters are stored along with index value, projection basis, random seed, and computation time.

Fifty additional simulations are conducted for each hyper-parameter combination to

analyze how they affect the JSO success rate. This includes variations in the number of jellyfish (20, 50, and 100) and the maximum number of tries (50 and 100).

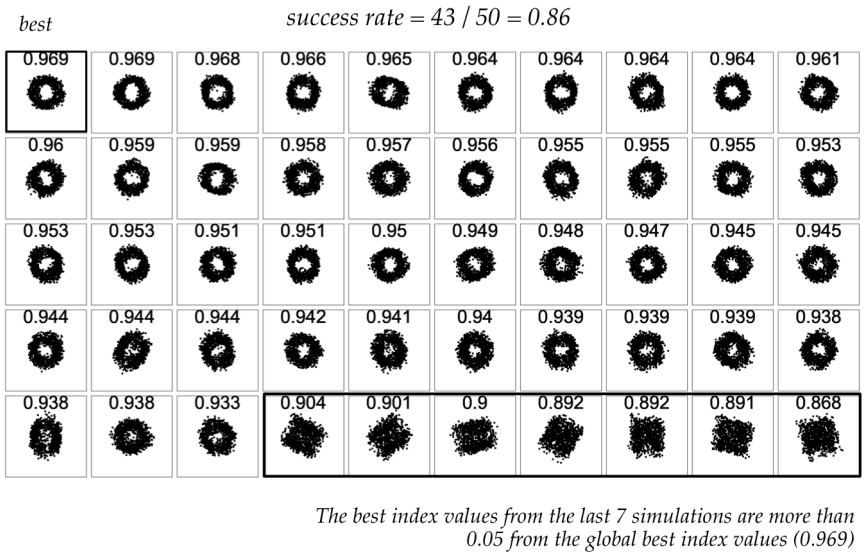

Figure 4. Illustration of success rate calculation: Final projections based on projection pursuit to find the pipe shape in 8D data using the holes index, optimised by CRS, in 50 simulations. The 50 final projections are sorted by their index values. The highest index value found across all simulations is 0.969. Out of the 50 simulations, 43 achieved an index value within 0.05 of the best, resulting in a success rate of 0.86 (43/50).

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

To assess JSO's performance across various scenarios, two different data shapes, pipe and a sine wave, are investigated in 6D and 8D spaces using six different PP indexes: `dcor2d_2`, `loess2d`, `MIC`, `TIC`, `spline`, and `stringy`, with varied JSO hyper-parameters. A total of 52 combinations result, comprising of 24 computed on the pipe data and 28 on the sine-wave data. Again, JSO is run 50 times to calculate the success rate for each projection pursuit.

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, 300 random bases are simulated. Index values and projection distance (to the optimal basis) are calculated for each random basis before fitting the 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 and the index values are averaged with a bin width of 0.005. A four-parameter scaled logistic function is fitted to the index values against projection distances, estimated by non-linear least squares. The squintability measure is then calculated as (8).

To construct a relationship among success rate, index properties (smoothness and squintability), and jellyfish hyper-parameters, a generalised linear model is fitted using a binomial family and a logit link function. The data is pre-processed by 1) scaling the JSO hyper-parameters by a factor of 10 for interpretation, 2) creating a new binary variable `long_time` to indicate cases with an average run time over 30 seconds, and 3) re-coding the success rate for the `stringy` index as 0, because none of the 50 simulations correctly identified the sine-wave shape.

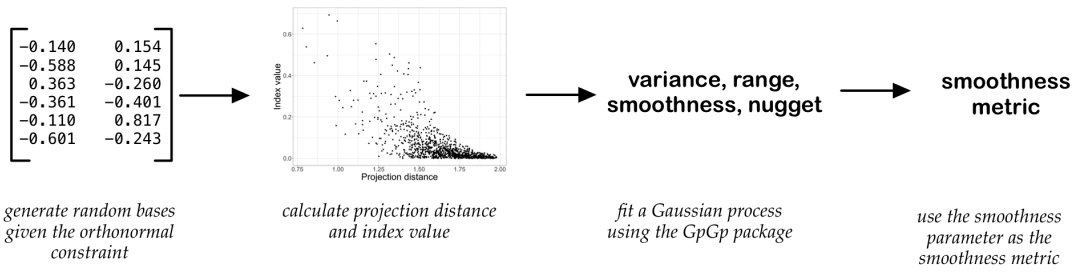

Figure 5. Illustration of steps to calculate smoothness. For a given projection pursuit problem defined by the shape to find, data dimension and the index function, 1) sample random bases given the orthonormality contraint, 2) calculate the projection distance and the index value for each random basis, and 3) fit a Gaussian process model of index values against projection distances to obtain the smoothness measure.

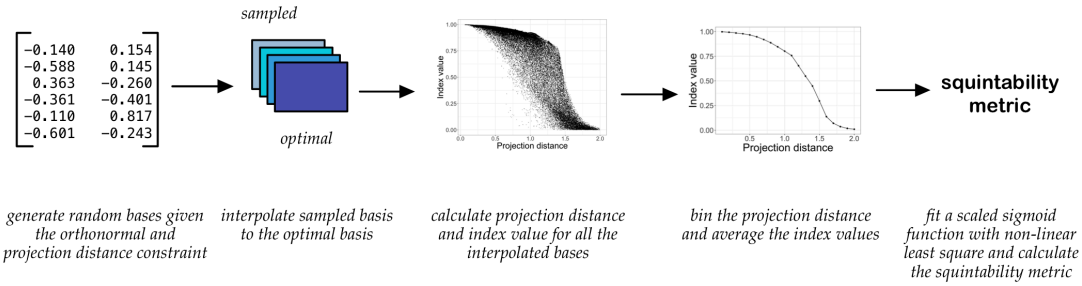

Figure 6. Illustration of steps to calculate squintability. For a given projection pursuit problem defined by the shape to find, data dimension and the index function, 1) sample random bases given the orthonormality and projection distance contraint, 2) interpolate the sampled bases to the optimal basis and calculate the projection distance and the index value for each interpolated basis. 3) bin the index values by projection distances to obtain the average index value for each bin, 4) fit the scaled sigmoid function in equation (5) to the binned index values against projection distances using non-linear least square, 5) calculate the squintability measure using equation (8) with parameters estimated from the model.

5. Results

The results from the first simulation described in Section 4 are analysed based on the final projections across two optimisers (JSO and CRS) and the success rate across JSO hyper-parameters. In the second simulation, 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.

The final projections found by the two optimisers (JSO and CRS) are presented in Figure 7, broken down by 10th quantile, faceted by the data dimensions. In the 6-dimensional data scenario, JSO consistently identifies a clear pipe shape. The CRS also finds the pipe shape but with a wide rim, suggesting a further polish search may be required. With increasing dimensions, JSO may not always identify the pipe shape due to random sampling, but it still finds the pipe shape in over 50% of cases. Compared to CRS, JSO achieves higher index values and clearer pipe shapes across all quantiles in data of 8, 10, 12 dimensions, suggesting its advantage in exploring high-dimensional spaces.

The success rate calculated at each hyper-parameter combination (number of jellyfish and the maximum number of tries) is presented in Figure 8. As the number of jellyfish and maximum tries increase, the success rate also increases. For simpler problems (6 dimensions), small parameter values (20 jellyfishes and a maximum of 50 tries) can already achieve a high success rate. However, larger parameter values (i.e. 100 jellyfishes and a maximum of 100 tries) are necessary for higher-dimensional problems (8, 10, and 12 dimensions). Increasing both parameters enhances the performance of JSO, but 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.

The index properties, including smoothness and squintability, offer numerical metrics to characterise the complexity of projection pursuit optimisation problems. Table 1 presents the parameters for calculating both metrics estimated from the Gaussian process (variance, range, smooth, and nugget) and the scaled logistic function (θ_1 to θ_4) for each case considered in the simulation. The column "smooth" is used as the smoothness metric and the column "squint" is calculated as equation (8) as the squintability metric. Table 3 presents the results of fitting a generalised linear model with a binomial family and a logit link function to a sample of data in Table 2, where all the three components (success rate, JS hyper-parameters, and index properties) are combined. The model suggests that JSO success rate is positively associated with the two hyper-parameters, as well as with the index properties: smoothness and squintability. Specifically, using 10 more jellyfish and 10 more tries increases the odd ratio of success by 24.11% and 11.93%, respectively. However, being flagged with long runtime and an increase of data dimension reduce the success rate by 41.72% and 53.36%, respectively. The variable `squintability` and `dimension` are significant, suggesting their importance relative to

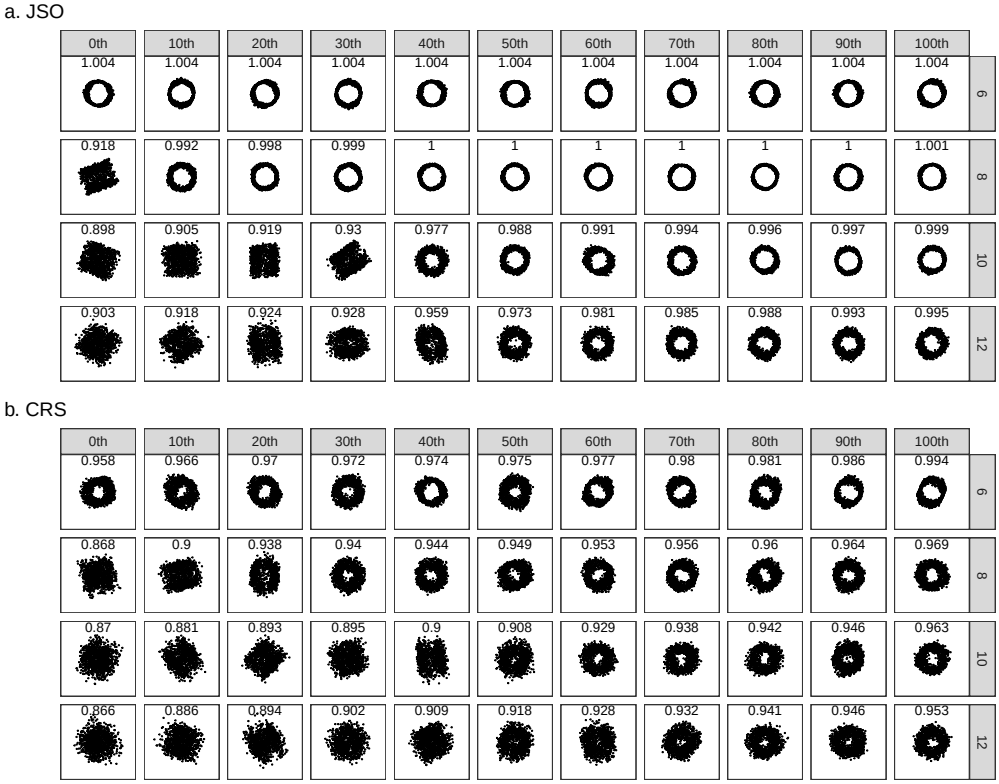

Figure 7. Projections found by the JSO and CRS at each 10th quantile across 50 simulations. The projection pursuit problem is to find the pipe shape using the holes index in the 6, 8, 10, and 12-dimensional spaces. The JSO uses 100 jellyfishes and a maximum number of tries of 100. The CRS uses a maximum of 1000 tries in each step of random sampling step before the algorithm terminates. In the 6-D data space, JSO always finds a clear pipe shape while the CRS also finds the pipe shape but with a wide rim. At higher data dimensions, JSO finds a higher index value and a clearer pipe shape across all the quantiles than the CRS

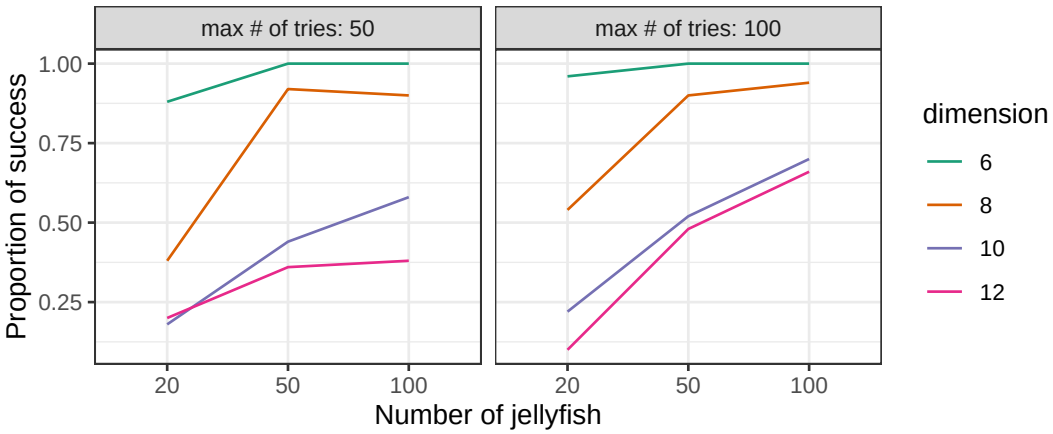

Figure 8. Proportion of simulations reaches near-optimal index values in the pipe-finding problem using the holes index. The proportion is calculated based on the number of simulations, out of 50, that achieve an index value within 0.05 of the best-performing simulation. As the dimensionality increases, the proportion of simulations reaching the optimal index value increases.

JSO hyper-parameters in the optimisation success.

6. Practicalities

The simulation study illustrates how these metrics can be used to assess a PP index, and the optimisation process. In general, we would expect that the JSO is going to outperform most other optimisers. From an exploratory perspective, it take s shift in the GTPP practice because one no longer has a single path to follow. Here we describe how to adjust to using the JSO optimiser for exploring high-dimensional data, and how to use the new metrics for evaluating a new index.

6.1. *Using the JSO in a PPGT*

To use JSO for PPGT in the `tourr` package, specify `search_f = search_jellyfish` in the guided tour. Since multiple tour paths are explored by the jellyfish, the animation function `animate_*()` will not render the tour path for JSO. To visualise path visited by

Table 1. Parameters estimated from the Gaussian process (including variance, range, smoothness, and nugget) and scaled logistic function (θ_1 to θ_4) for the pipe-finding and sine-wave finding problems. The column "smooth" and "squint" represent the smoothness and squintability measures.

	shape	index	d	variance	range	smooth	nugget	θ_1	θ_2	θ_3	θ_4	squint
1	pipe	holes	6	0.00	0.37	2.36	0.22	1.02	0.86	3.37	0.02	3.05
2	pipe	holes	8	0.00	0.18	2.19	0.82	1.01	0.87	3.26	0.03	2.96
3	pipe	holes	10	0.00	0.11	2.19	3.03	1.02	0.88	3.15	0.02	2.95
4	pipe	holes	12	0.00	0.15	2.29	1.58	1.01	0.88	3.34	0.00	3.12
5	sine	MIC	6	0.02	0.09	2.46	0.08	0.89	0.57	1.62	-0.02	1.26
6	sine	MIC	8	0.02	0.08	2.64	0.08	0.93	0.33	1.31	-0.03	0.77
7	sine	TIC	6	0.12	0.11	2.44	0.09	0.95	0.54	1.72	-0.03	1.32
8	sine	TIC	8	0.12	0.10	2.47	0.09	0.95	0.56	1.72	-0.03	1.37
9	sine	dcor2d	6	0.03	0.17	2.66	0.11	0.95	1.04	2.74	-0.02	2.95
10	sine	loess2d	6	0.08	0.34	1.99	0.31	1.02	1.04	2.65	0.08	2.76
11	sine	splines2d	6	0.04	0.24	2.54	0.10	1.01	1.05	2.73	-0.01	3.12
12	sine	stringy	6	0.00	0.01	1.54	38.17	1.05	0.01	254.73	0.05	2.96

Table 2. The first 7 rows of the datasets processed for modelling.

index	d	smoothness	squintability	n. jellyfish	max. tries	success rate	time (sec)
MIC	6	2.46	1.26	20	50	0.12	2.48
MIC	6	2.46	1.26	20	100	0.24	8.95
MIC	6	2.46	1.26	50	50	0.52	5.65
MIC	6	2.46	1.26	50	100	0.64	13.22
MIC	6	2.46	1.26	100	50	0.76	19.45
MIC	8	2.64	0.77	20	50	0.08	2.57
MIC	8	2.64	0.77	20	100	0.08	4.96

Table 3. Model estimates of proportion of jellyfish success on index properties and jellyfish hyper-parameters from the generalised linear model with a binomial family and a logit link function. The variable smoothness, squintability, number of jellyfish and maximum number of tries are positively associated with JSO success rate while data dimension and being flagged as long runtime are negatively associated with the success rate. The variable squintability and dimension are significant, suggesting their importance relative to jellyfish hyper-parameters in the optimisation success.

term	estimate	std.error	statistic	p.value
Intercept	-4.52	5.33	-0.85	0.40
Smoothness	1.19	1.92	0.62	0.53
Squintability	2.06	0.68	3.01	0.00
Dimension (d)	-0.63	0.26	-2.46	0.01
Long time	-0.87	1.29	-0.68	0.50
N. jellyfish	0.22	0.13	1.70	0.09
Max. tries	0.11	0.15	0.75	0.45

each individual jellyfish, assign the animation to an object (`res <- animate_xy(...)`). This will save the bases visited by the optimiser, 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 as a planned tour. The following code uses JSO, with 20 jellyfish and a maximum of 50 tries, on the penguins data using the `lda_pp` index.

```
# clean up the penguins data
penguins_cl <- palmerpenguins::penguins |>
  filter(!is.na(bill_length_mm)) |>
  rename(bl = bill_length_mm, bd = bill_depth_mm,
         fl = flipper_length_mm, bm = body_mass_g)
penguins_df <- penguins_cl |> select(bl:bm) |> rescale()
# run the projection pursuit with JSO
res <- animate_xy(penguins_df,
                  guided_tour(lda_pp(cl = penguins_cl$species),
                              search_f = search_jellyfish,
                              n_jellies = 20, max.tries = 50))
```

The bases visited by jellyfish No. 1 is extracted with `filter(loop = 1)` (the variable `loop` in the data object is the jellyfish ID) and the `check_dup()` function removes the bases in the iteration without improvements. The bases are then supplied as a `planned_tour()` in `animate_xy()` to visualise the tour path of jellyfish No.1.

```
# extract the bases from the first jellyfish
bases <- res |> filter(loop == 1) |> pull(basis) |> check_dup(0.1)
```

```
# visualise its path with a planned tour

animate_xy(data = penguins_df, tour_path = planned_tour(bases),

           col = penguins_cl$species)
```

Figure 9 shows six frames from the tour jellyfish No.1 animation. The Gentoo species (red) was mixed with Chinstrap (yellow) initially and gradually separated out as flipper length (`fl`) gaining more weights.

Figure 9. Visualising the tour path of a single jellyfish (No.1) in the penguins data set. The projection pursuit is optimised by the JSO with 20 jellyfishes and a maximum of 50 tries. The bases visited by jellyfish No.1 is extracted and constructed as a planned tour. The jellyfish successfully finds the structure of the data set, separating the three species of penguins.

6.2. *Computing the metrics for your new index*

The `ferrn` package provides functionality for computing the smoothness and squintability metrics. Both cases require sampling random bases using the `sample_bases()` function, followed by calculating the specific metric with either `calc_smoothness()` or

`calc_squintability()`.

6.2.1. Smoothness

To sample bases for calculating smoothness, the required arguments include `idx` (the index function used), `data` (the dataset), `n_basis` (the number of random bases to sample), and `best` (the best basis to compute projection distance). The following code samples 300 bases for using the `holes` index to find the pipe shape in the 6D `sine1000` dataset:

```
library(GpGp)
library(fields)
library(tourr)
library(ferrn)
basis_smoothness <- sample_bases(
  idx = "holes", data = sine1000, n_basis = 300,
  best = matrix(c(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1), nrow = 6))
basis_smoothness |> head(3)
```

```
# PP index:                holes
# No. of bases:            3
# interpolation step size: NA
# Min. proj. dist.:        NA
     id basis          dist index
  <int> <list>        <dbl> <dbl>
1     1 <dbl [6 x 2]>  1.15 0.676
2     2 <dbl [6 x 2]>  1.77 0.790
```

```
3     3 <dbl [6 x 2]>  1.79 0.778
```

The output of `sample_bases()` is a data frame with four columns: `id` (the basis ID), `basis` (the basis matrix sampled), `dist` (the projection distance to the best basis), and `index` (the index value). To speed up the computation of index values, parallellasation is available 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 projection distances to compute the smoothness metric. The starting parameter for the fit can be specified using the `start_params` argument, and additional Gaussian process arguments can be provided through `other_gp_params`. The following code calculates the smoothness metric from the previously sampled bases:

```
(sm_res <- calc_smoothness(basis_smoothness))
```

```
# PP index:     holes
# No. of bases: 300 [6 x 2]
    variance range smoothness nugget convergence
       <dbl> <dbl>      <dbl>  <dbl> <lgl>
1 0.00000672  18.1       1.03  1138. TRUE
```

The smoothness metric is the smoothness parameter from the Gussian process and all details of the fit can be accessed using `attr(sm_res, "fit_res")`.

6.2.2. Squintability

Sampling bases for calculating squintability includes an additional step of interpolating between the sampled bases and the best basis. This is specified through setting the `step_size` and `min_proj_dist` arguments in the `sample_bases()` function to non-

NA numerical values. Since the projection distance typically ranges 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. The following code samples 100 bases for calculating squintability in the same scenario:

```
basis_squint <- sample_bases(
   idx = "holes", data = sine1000, n_basis = 100,
  best = matrix(c(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1), nrow = 6),
  step_size = 0.1, min_proj_dist = 1.5)
calc_squintability(basis_squint, method = "nls", bin_width = 0.01)
```

```
# PP index:     holes
# No. of bases: 100 -> 1853
# method:       nls
  theta1 theta2 theta3 theta4 squint
   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1  0.274  0.960   2.21  0.612  0.203
```

The output provides the estimated parameters for the scaled sigmoid function (θ_1 to θ_4) and the squintability metric for the parametric case. For the non-parametric case, it shows the (`max_d`), the corresponding projection distance (`max_dist`), and teh squintability metric as their products.

7. 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 out-performs CRS, and provides a high probability of finding the global optima. 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 optima 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 optima. However, interestingly smoothness does not affect the JSO performance.

8. Acknowledgement

The article is 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. Nicolas Langrené acknowledges the partial support of the Guangdong Provincial Key Laboratory IRADS (2022B1212010006, R0400001-22) and the UIC Start-up Research Fund UICR0700041-22.

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/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 Computing C* 23: 881–89.

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.

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." *Ann. Statist.* 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.

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.

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/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/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.

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.

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

———. 1995a. "Projection Pursuit Exploratory Data Analysis." *Computational Statistics & Data Analysis* 20 (6): 669–87. https://doi.org/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.

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.

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. http://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.