

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

H. Sherry Zhang<sup>1</sup>, Dianne Cook<sup>2</sup>, Nicolas Langrené<sup>3</sup>, Jessica Wai Yin Leung<sup>2</sup>

## ARTICLE HISTORY

Compiled November 19, 2025

<sup>1</sup> Department of Statistics and Data Sciences, University of Texas at Austin, Austin, United States

<sup>2</sup> Department of Econometrics and Business Statistics, Monash University, Melbourne, Australia

<sup>3</sup> 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 optimizes a criterion function, known as the PP index, to gradually reveal projections of interest from high-dimensional data through animation. Optimization of some PP indexes can be non-trivial, if they are non-smooth functions, or when 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 Optimizer (JSO), for optimizing PP indexes. The performance of JSO in detecting the target pattern (pipe shape) is compared with existing optimizers in PP. Additionally, JSO’s performance on detecting the sine-wave shape is evaluated using different PP indexes (hence different smoothness and squintability) across various data dimensions ( $d = 4, 6, 8, 10, 12$ ) and JSO hyper-parameters. We observe empirically that higher squintability improves the success rate of the PP index optimization, while smoothness has no significant effect. 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 Guided Tour (PPGT); Jellyfish Search Optimizer (JSO)

---

CONTACT: H. Sherry Zhang. Email: [huize.zhang@austin.utexas.edu](mailto:huize.zhang@austin.utexas.edu). Dianne Cook. Email: [dicook@monash.edu](mailto:dicook@monash.edu). Nicolas Langrené. Email: [nicolaslangrene@uic.edu.cn](mailto:nicolaslangrene@uic.edu.cn). Jessica Wai Yin Leung. Email: [jessica.leung@monash.edu](mailto: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 optimizing 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 in the Stiefel manifold  $\mathcal{A} = V_d(\mathbb{R}^p)$ . The projection  $Y = XA$  is a linear transformation of the  $p$ -dimensional data to a  $d$ -dimensional space. The index function  $f(XA) : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}$  defines a statistics to measure a pattern of interest in 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 matrix  $A$  (also called *projection basis* in PP literature) that maximizes the index value of the projection:

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

where  $I_d$  is the identity matrix of dimension  $d$ .

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 optimization 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 optimizer is very close to the optimal projection.

Optimization of PP is sometimes 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 optimization closely to the index with the introduction of the PPGT, which monitors the optimization 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 (Wickham et al. 2011) in R (R Core Team 2023). Zhang et al. (2021) illustrated how to diagnose optimization processes, particularly focusing on the guided tour, and revealed

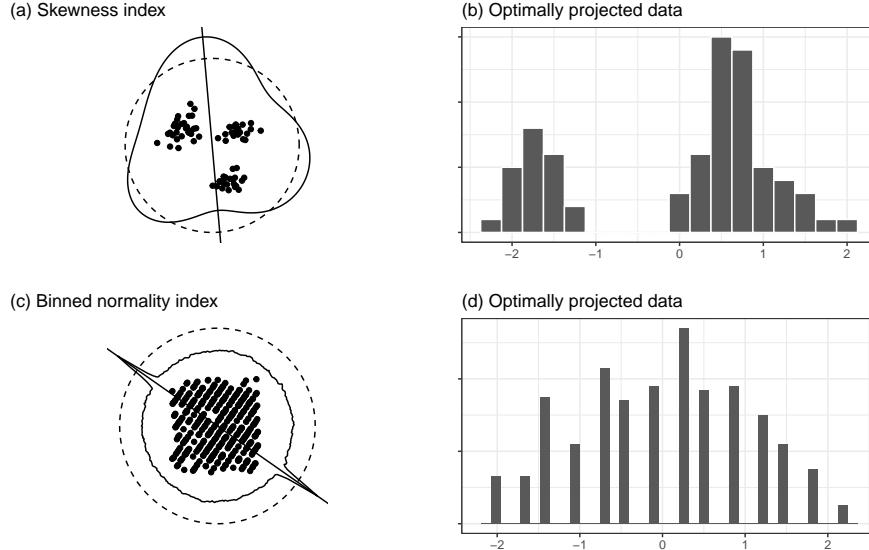


Figure 1. Examples of PP indexes with large (top row) and small (bottom row) squint angles, shown with a Huber plot, and histogram of the projected data corresponding to the optimal projection. A Huber plot shows the PP index values for all 1D data projections in polar coordinates. The skewness index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

a need for improved optimization. While improving the quality of the optimization solutions in the tour is essential, it is also important to be able to view the data projections as the optimization progresses. Integrating the guided tour with a global optimization algorithm that is efficient in finding the global optimal and enables viewing of the projected data during the exploration process is a goal.

In this work, the potential for a Jellyfish Search Optimizer (JSO) (Chou and Truong 2021; Rajwar, Deep, and Das 2023) for the PPGT is explored. The JSO, inspired by the search behavior of jellyfish in the ocean, is a swarm-based metaheuristic designed to solve global optimization problems. Compared to other metaheuristic 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 optimization.

The primary goal of the study reported here is to investigate the performance of JSO in

PP optimization for the guided tour. It is of interest to assess how quickly and closely the optimizer 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. We mathematically define metrics for squintability and smoothness, which is a new contribution for PP research. A series of simulation experiments are conducted using JSO to detect different target patterns (pipe and sine-wave) with different PP indexes (holes, MIC, TIC, dcor, loess, splines, skinny, and stringy), as well as JSO’s hyper-parameter choices (number of jellyfish and maximum number of iterations). This work should facilitate better optimization for PP and guide the choice of optimizer when designing new PP indexes.

The paper is structured as follows. Section 2 introduces the background of the PPGT, reviews existing optimizers and index functions in the literature. Section 3 introduces the metrics that measure two properties of PP indexes, smoothness and squintability. Section 4 describes the new JSO for PP. Section 5 outlines two simulation experiments to assess JSO’s performance: one comparing JSO’s performance improvements relative to an existing optimizer, Creeping Random Search (CRS), and the other studying the impact of PP index properties, controlling for data dimension and JSO hyper-parameters, on optimization 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 summarizes the work and provides suggestions for future directions.

## 2. Projection pursuit, tours, index functions and optimization

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 optimizer. 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 optimized. There are five optimization methods implemented in the `tourrr` package:

- a pseudo-derivative, that searches locally for the best direction, based on differing the index values for very close projections.
- a brute-force optimization (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 optimization.
- a very localized 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), Laa et al. (2022). Most are relatively simply defined, for any projection dimension, and implemented because they are relatively easy to optimize. A goal is to develop PP indexes based on scagnostics (Wilkinson, Anand, and Grossman 2005; Wilkinson and Wills 2008), but the blockage is their optimization 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 optimizer needs to be able to handle index functions that are possibly not very smooth. In addition, because the target structure in the data might be relatively fine, the optimizer 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 optimizer 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 optimization process, the focus of this work 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

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. For example, the loess and splines indexes are not differentiable due to the use of the max operator in their definitions. Scagnostic indexes, such as skinny and stringy, are defined based on graph elements (area, diameter, perimeter, and length) of alph hull or minimal spanning tree).

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, 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

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

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

The Matérn covariance function can be expressed analytically when  $\nu$  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  $\nu$ , 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  $\nu$  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  $\nu \rightarrow \infty$  coincides with the Gaussian kernel:  $K_\infty(u) = \exp(-\|u\|^2/2)$ .

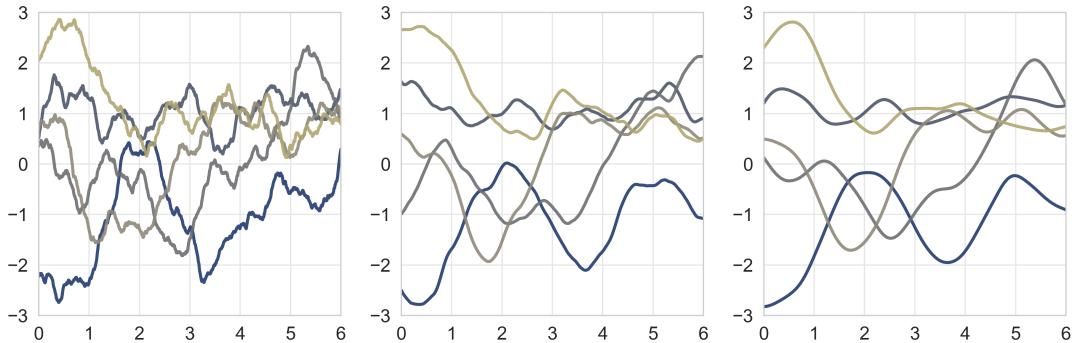


Figure 2. Five random simulations from a Gaussian Process defined on  $\mathbb{R}$  with zero mean and Matérn- $\nu$  covariance function, with  $\nu = 1$  (left),  $\nu = 2$  (middle), and  $\nu = 4$  (right), showing that higher values of  $\nu$  produce smoother curves.



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

In view of these results, the parameter  $\nu$  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. There exist several R packages, such as **GpGp** (Guinness, Katzfuss, and Fahmy 2021) or **ExaGeoStatR** (Abdullah 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,  $\mathbf{y} = [f(XA_1), \dots, f(XA_N)]$  be the corresponding PP index values, and  $\mathbf{K} = [K_\theta(A_i - A_j)]_{1 \leq i, j \leq N} \in \mathbb{R}^{N \times N}$  be the Matérn covariance matrix evaluated at the input bases, where the vector  $\theta$  contains all the parameters of the multivariate Matérn covariance function  $K$  (smoothness, outputscale, lengthscales). The log-likelihood of the parameters  $\theta$  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  $\sigma$  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  $\nu$  is chosen as our smoothness metric.

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

### 3.2. Squintability

The squintability metric is inspired by the concept of squint angle in Definition 4, originally introduced in Tukey and Tukey (1981) and later discussed in Laa and Cook (2020). Consider the space of all projection bases: those near the optimal basis correspond to projections with interesting structure we're researching. The size of this neighborhood depends on the choice of the index function. For some indexes, projections resemble the target even when relatively far from the optimum, corresponding to a large squint angle, while for others, projections need to be close to the optimum to reveal structure we're interested, resulting in a small squint angle. For PP, a small squint angle is considered to be undesirable because it means that the optimizer needs to sample into a very narrow region to be able to "see" the optimum, making it difficult for the optimizer to find the optimum.

**Definition 4** (squint angle). *Let  $A$  and  $B$  be two  $d$ -dimensional orthonormal matrices in  $\mathbb{R}^p$ . The squint angle  $\theta$  between the subspace spanned by  $A$  and  $B$  is defined as the smallest principal angle between these subspaces:  $\theta = \min_{i \in \{1, \dots, d\}} \arccos(\tau_i)$ , where  $\tau_i$  are the singular values of the matrix  $M = A^T B$  obtained from its singular value decomposition.*

To quantify how close projections need to be before structures of interest can be observed (the optimizer starts hill-climbing), we can track how the index value changes as we move from an initial projection basis towards the optimal one, that is, how index value changes as the projection distance becomes smaller.

**Definition 5** (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  $\|\cdot\|_F$  denotes the Frobenius norm, given by  $\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}$ .*

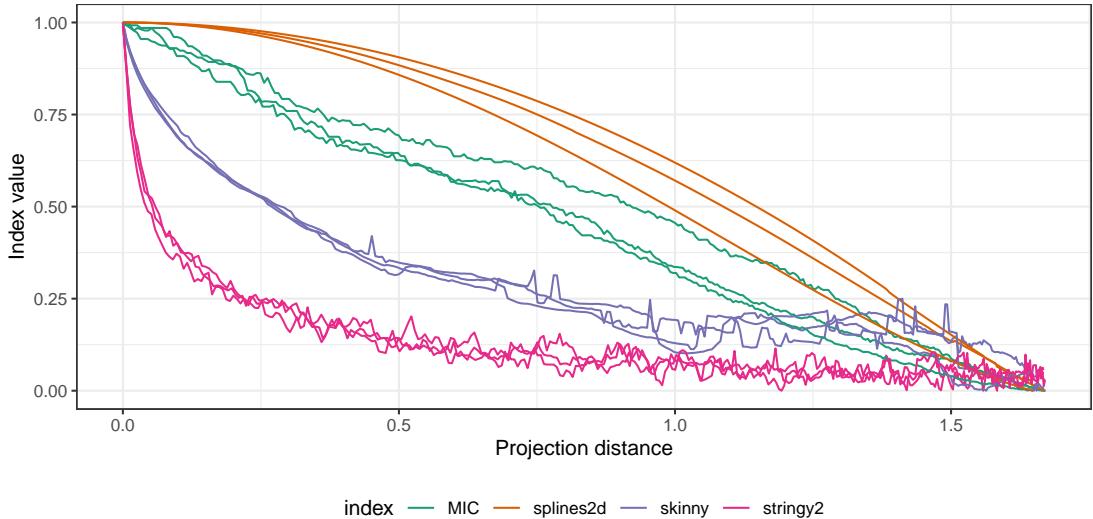


Figure 4. Simulated traces of index value as a function of projection distances for four PP index functions: MIC, splines2d, skinny, and stringy2. The index function MIC and splines2d make early progression towards the optimal index value, indicating a large squint angle, whereas the traces from skinny, and specially stringy2, show improvement only near the optimal projection, suggesting low squintability.

Figure 4 illustrates some simulated traces for different PP index functions: MIC, splines2d, skinny, and stringy2. 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.

**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*

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

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) represents 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 ( $r_0/2$ ).

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  $\varsigma$ . Numerical experiments suggest a scaled sigmoid shape as described below. Define

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

which is a decreasing logistic function depending on two parameters  $\theta_2$  and  $\theta_3$ , such that

$\ell(\theta_2) = \frac{1}{2}$ . Then, define

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

which depends on three additional parameters,  $\theta_1$ ,  $\theta_2$ , 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

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

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

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

#### 4. The jellyfish search optimizer

The Jellyfish Search Optimizer (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 behavior of jellyfish to navigate towards the best solution over iterations (Chou and Truong 2021).

To solve the optimization problem embedded in the PPGT with JSO, a starting projection,

an index function, the number of jellyfish, and the maximum number of iterations 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 time control function,  $c_t$ , whose value decreases as the number of trials increases (subject to randomness) to guide the exploration and exploitation phases of the algorithm. When  $c_t \geq 0.5$ , new directions will be taken like a jellyfish explores with the ocean trend, defined as the difference of the current best jellyfish and average of all the current jellyfish. When  $1 - c_t$  is smaller than a randomly generated number in  $[0, 1]$ , jellyfish will move passively with small random perturbations, otherwise, it will move actively towards or away from another jellyfish based on their index values. 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:** `optimized_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 `trial_id` and `max_trial`

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

        Define the ocean current trend as the difference of the `current_best` and the average of the `current_projections`

    Update each projection towards the ocean trend using a random factor and or-

```

thonormalisation

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 in current_projections and
    adjust towards or away from it based on their corresponding current_idx (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

exit

return the set of projections with the updated current_best as the
optimized_projection

```

The JSO implementation involves several key parameters that control its search process in optimization problems. 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 iterations.

## 5. Assessing the optimizers

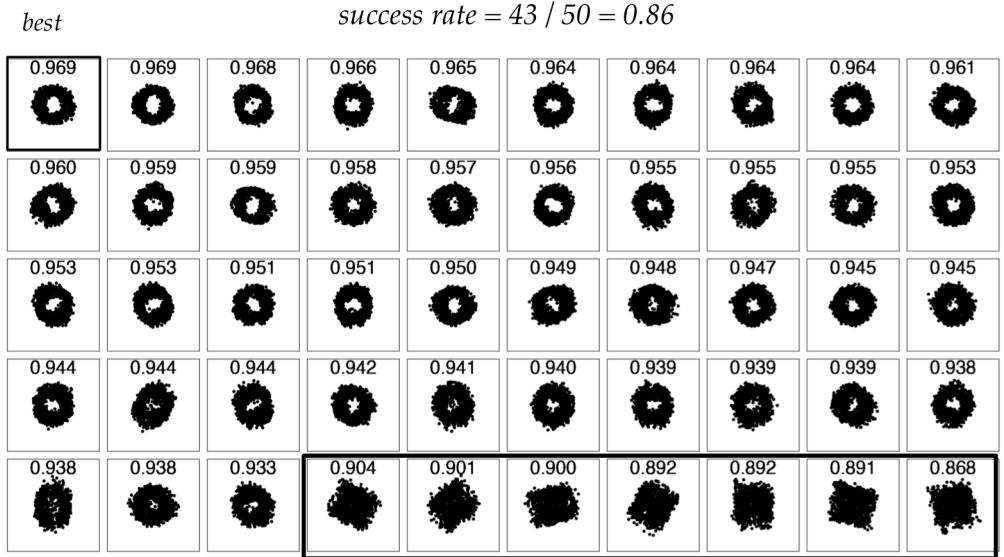
This section explains the details of two simulation studies: (1) a comparison between the JSO and an existing optimizer, creeping random search (Zhang et al. 2021; Laa and Cook 2020), and (2) a modelling study to examine how factors (smoothness, squintability, data dimension, and JSO hyper-parameters) affect its success rate in detecting holes and sine-wave pattern across a collection of index functions.

### 5.1. *Performance of JSO relative to CRS*

The CRS is the main optimization routine currently used for the guided tour. Here the two optimizers are compared on the task of finding the 2D pipe structure using the `holes` index in data at dimensions 6, 8, 10, and 12. Additional variables are Gaussian noise. JSO uses 100 jellyfish with 100 iterations, while CRS allows up to 1000 trials per iteration terminates if no better projection is found within these trials. These choices enable fair comparison between CRS and JSO, that conforms to how they are used in practice. Each task is repeated 50 times to evaluate the performance between the two optimizers.

The performance of the optimizers 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 5 illustrates why the choice of 0.05 is reasonable: the pipe is not recognizable 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 optimizers. The design parameters are stored along with index value, projection basis, and random seed used.



The best index values from the last 7 simulations are more than 0.05 from the global best index values (0.969)

Figure 5. How success rate is calculated, illustrated using the optimal projections from 50 optimizations of 8D pipe data, optimised by CRS, sorted by index value. The pipe shape is recognizable 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 relative to hyper-parameter choices, the `holes` index is used to find the 2D pipe in data of dimensions 4, 6, 8, 10, and 12. The levels of hyper-parameters are 20, 50, and 100 jellyfish and a maximum of 50 and 100 attempts. Each task is repeated fifty times to calculate the success rate as described in Section 5.1.

This set of simulations is expanded to assess the performance of JSO relative to the index properties, with a second data set type and additional PP indexes. The second data set has a sine wave in two dimensions and Gaussian noise in the remaining dimensions. Seven additional PP indexes that should be able to extract the sine wave are used (dcor, loess, MIC, TIC, splines, skinny and stringy2) following Laa and Cook (2020). (The supplementary material includes definitions of these PP indexes, and descriptions of the

pipe and sine wave data simulation.) The JSO hyper-parameters, number of jellyfish and maximum iterations, are also included. This results in a total of 77 scenarios, comprising of 30 computed on the pipe data and 47 on the sine wave data. The choice of 4-12 dimensions covers the range of easy to relatively difficult for the optimizers to detect the structured projections. The same approach is used, that each experiment is repeated 50 times and success rate is calculated at each level of the simulation design.

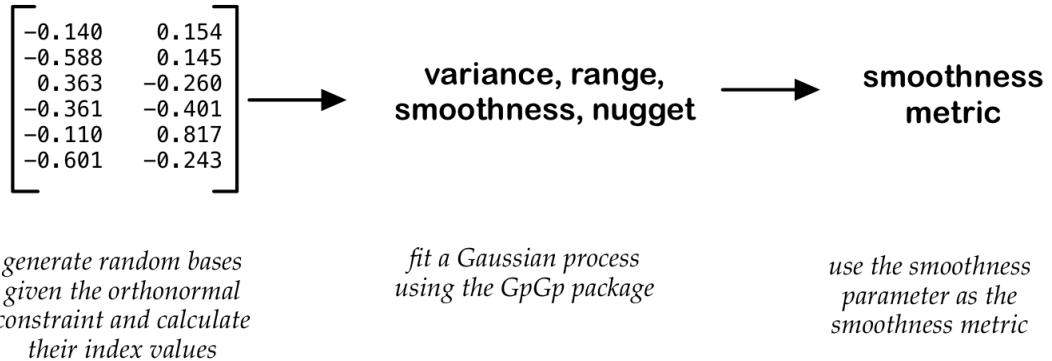


Figure 6. Steps for calculating smoothness in a projection pursuit problem: 1) sample random bases given the orthonormality constraint and calculate their corresponding index values, 2) fit a Gaussian process model of index values against the bases to obtain the smoothness measure, 3) take the smoothness parameter from the Gaussian process as the smoothness metric.

Recall that smoothness and squintability are metrics used to characterize the index function for PP tasks and are invariant to the optimizer and its hyper-parameters. Thus simulations with different JSO hyper-parameters for the same PP task share the same smoothness value. In total, smoothness and squintability metrics are calculated on the 23 PP tasks in our simulation.

Figure 6 describes the procedure to compute smoothness. For each PP task, 500 random bases are simulated and index values are calculated for each. A Gaussian process model is then fitted to the resulting data to obtain the smoothness measure for the index, as described in Section 3.1.

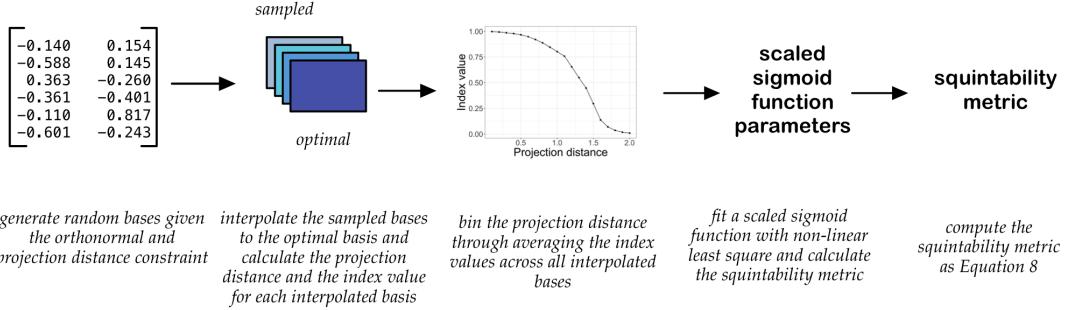


Figure 7. Steps for calculating squintability in a projection pursuit problem: 1) sample random bases given the orthonormality and projection distance constraint and calculates their corresponding index values, 2) interpolate the sampled bases to the optimal basis and calculate the projection distance and the index value for each interpolated basis, 3) average the index values by projection distances at each 0.005 distance bin, 4) fit the scaled sigmoid function in Equation (6) and (7), 5) calculate the squintability metric using Equation (8).

Figure 7 describes the procedure to compute squintability. For each PP task, 50 random bases are generated and interpolated to the optimal basis with a step size of 0.005. The index values are averaged over the distance window of 0.005. Index values are calculated for all the interpolated bases and averaged over the distance bin of 0.005. The logistic function in Equation (6) and (7) is fitted with non-linear least squares and calculate the squintability measure using Equation (8).

A generalized linear model is fitted using a quasibinomial family and a logit link function to assess the factors affecting the success rate. Predictors are smoothness, squintability, data dimension, and JSO hyper-parameters. For numerical stability related to small squintability values associated with the stringy index (squintability less than 0.002), we decide to use the rank of the two metrics, rather than the nominal values, in the model.

## 6. Results

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

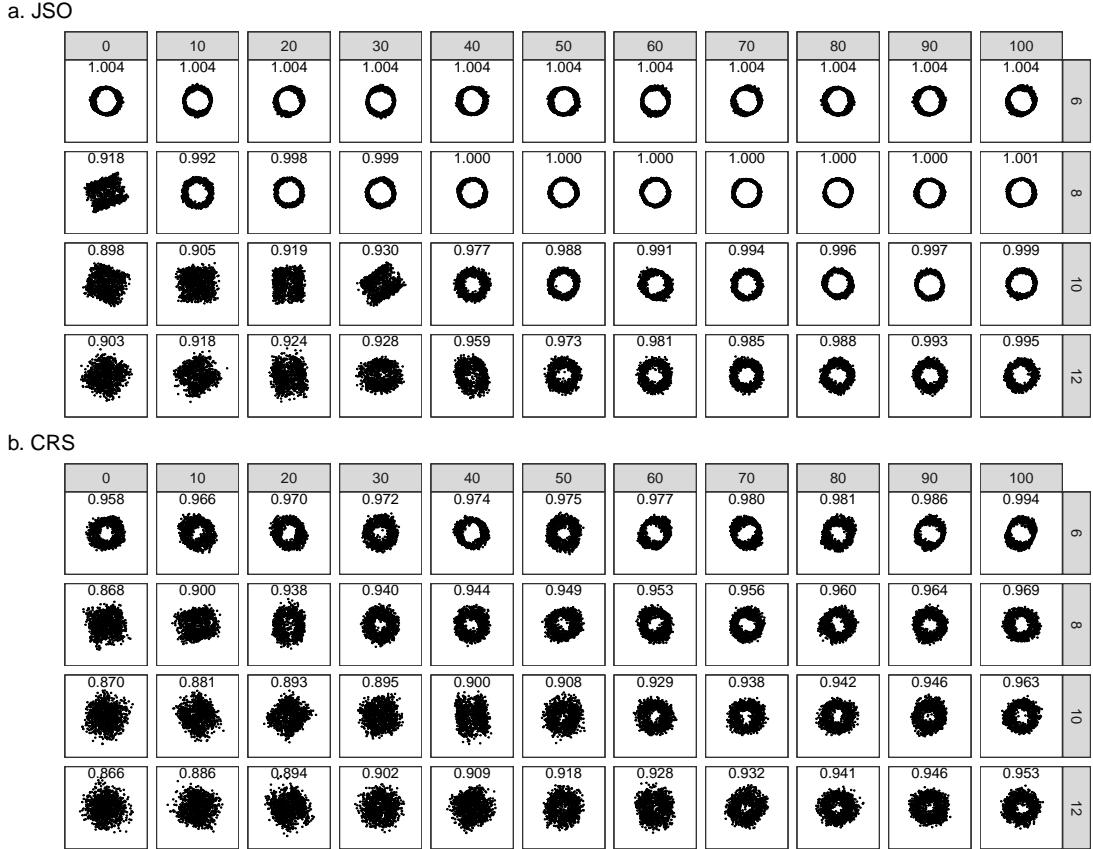


Figure 8. Visual comparison of JSO and CRS results, using optimal data projections obtained over 50 simulations of the pipe data. Rows correspond to data dimension. Columns correspond to quantiles of the index values, with 0 being the minimum, 100 being the maximum, and 50 the median. JSO achieves the 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 optimal projection is compared between JSO and CRS for the pipe data at dimensions 6, 8, 10, and 12 in Figure 8. The columns show the quantile of index value in the 50 optimal projections, with 0 being the minimum, 100 being the maximum, and 50 the median. The purpose is to summarize the views of the data resulting from different optimization, and hence compare results between the two optimizers. Generally, the JSO does a more consistent job of finding the pipe structure clearly. As the data dimension increases, both optimizers 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 iteration) on the success rate is summarized in Figure 9. Bootstrap resampling is used to quantify the uncertainty of the success rate through 500 bootstrap samples with replacement from the original 50 simulation results for each task: optimizing a high-dimensional data ( $d = 4, 6, 8, 10, \text{ or } 12$ ) with different number of jellyfish (20, 50, or 100) and maximum number of iteration (50 or 100). As the number of jellyfish and maximum iteration allowed increase, the success rate also increases. For problems in lower dimensional (4, 6) search spaces, small parameter values (20 jellyfish and 50 iterations) are sufficient to achieve a high success rate. Many more jellyfish (100 jellyfish and 100 iterations) are needed for finding the pipe structure in  $d = 10$ , and 12. This suggests that even JSO will struggle with high dimensions, and require substantially more computational time to have an acceptable success rate, especially for difficult PP indexes based on scagnostics.

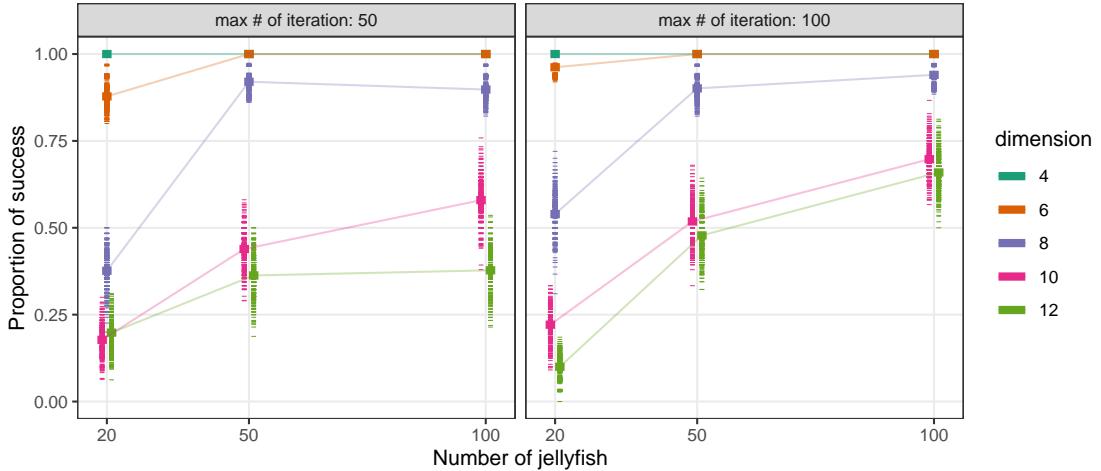


Figure 9. Summary of the relationship between JSO hyper-parameters on optimization success rate, using the holes index for pipe data of dimensions 4-12. Bootstrap samples show the variability in success rate. Success rate mostly plateaus by 50 jellyfish. The JSO has some difficulty finding the pipe when dimension is higher than 8 and maximum number of iteration has little effect.

Table 1. The columns  $\nu$  and  $\varsigma$  show the smoothness and squintability measures for three PP indexes for the sine wave data. The splines index has the largest squint angle and is the smoothest. The skinny index has low smoothness and smallest squint angle. As dimension increases the smoothness becomes smaller but squintability doesn't change much.

shape	index	d	$\nu$	$\varsigma$
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	6	3.0807	0.3747
sine	TIC	8	2.7223	0.3785
sine	skinny	4	2.5213	0.0810
sine	skinny	6	2.2352	0.0973

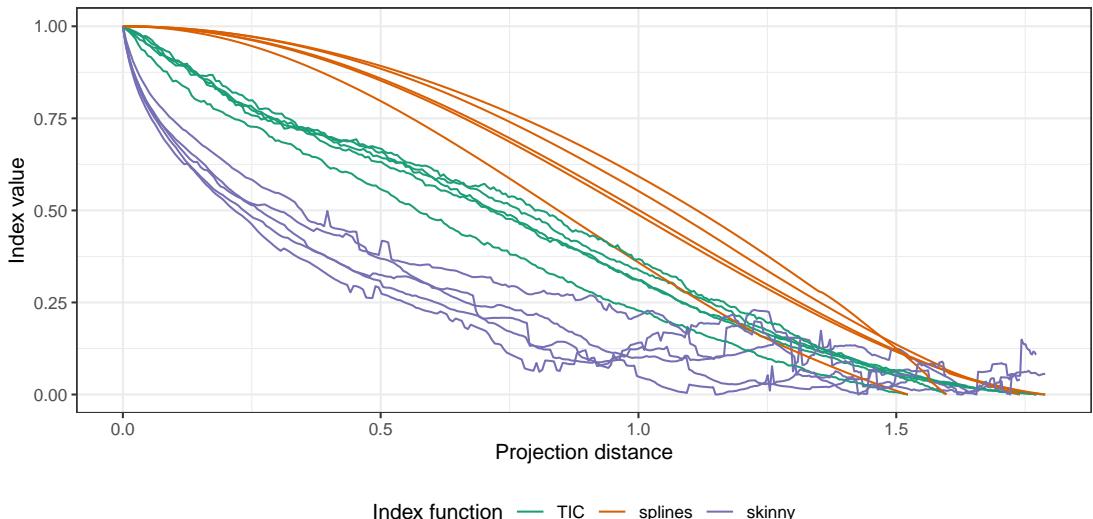


Figure 10. Five traces of index value against projection distance for each of the three indexes (TIC, spline, and skinny) on 6D sine wave data. The traces of the splines index are smooth, with a gradual change between noise and optimal projection. In contrast, the skinny index has a noisy trace that increases rapidly near the optimum. This illustrates the usefulness of the new measures for squintability and smoothness.

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 iterations. Interestingly, squintability and dimension strongly affect jellyfish optimization success. The number of jellies marginally affects success, but index smoothness, and increasing the number of iterations do not.

Characteristic	OR (95% CI)	p-value
Smoothness	1.01 (0.92 to 1.11)	0.810
Squintability	1.23 (1.12 to 1.38)	<0.001
Dimension	0.58 (0.45 to 0.73)	<0.001
Number of jellyfish	1.32 (1.17 to 1.50)	<0.001
Maximum number of iterations	1.06 (0.94 to 1.20)	0.328

Abbreviations: CI = Confidence Interval, OR = Odds Ratio

Table 3. Fit statistics for the model.

Deviance	DF Residual	Null Deviance	DF Null
20.89	71	49.77	76

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

Table 1 presents values computed for a subset of PP indexes on the sine wave data for dimensions 4-8. (A full table of the parameters estimated from the Gaussian process and from the scaled logistic function for calculating smoothness and squintability for all PP indexes and data combinations is included in the supplementary materials). The column  $\nu$  is used as the smoothness metric and the column  $\varsigma$  is calculated as equation (8) as the squintability metric.

Figure 10 shows a sample of traces from the indexes in Table 1 for the 6D sine wave data. This traces the PP index values computed for projections along an interpolated path between a randomly generated noise projection and the optimal projection (sine wave).

(Note index values were self-standardized to range between 0 and 1, for comparison.)

The splines index is very smooth with a gradual increase in value between the noise projection and the optimal projection. The TIC index is slightly less smooth and slightly less gradual change in value. Least desirable is the skinny index which is noisier and the optimization needs to be much closer to the optimal angle to have a higher value. This illustrates how the new measures for squintability and smoothness work to describe the PP indexes.

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 iteration, as predictors. The fit suggests that the JSO success rate is only affected by squintability, dimension and number of jellyfish. 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, 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 iterations also does not affect success rate significantly. A unit increase in squintability increases the success rate by 23%. As dimension increases by one the success rate almost halves. Increasing the number of jellies by 10 increases the success rate by 32%. Table 3 shows the fit statistics supporting that the model is reasonably strong.

In relation to the model fit, it should be noted that there is moderate correlation between squintability and smoothness, but this does not substantially affect the fit. Fitting smoothness alone results in this variable being weakly significant for explaining success rate, but the effect is dominated by squintability. The addition of interaction terms for squintability and data dimension, and smoothness and data dimension, improves the

model slightly, but not enough to be practically important or change the interpretation above. The CRS optimizer couldn't be used here because it is not efficient (Figure 8) but we would expect that smoothness does not overly affect it either. Both JSO and CRS are expected to be robust to small local fluctuations.

These results also support the use of the proposed measures to effectively capture key factors affecting the success rate of PP.

## 7. Practicalities

Using the JSO optimizer for PPGT in the `tourrr` package requires a change in user behavior. For all the existing optimizers a single optimization path is followed. The JSO has multiple optimization 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 `tourrr` package, specify `search_f = search_jellyfish` in the guided tour. The animation function `animate_*`() won't directly render the animated projection for JSO, as with other existing optimizers since multiple tour paths will need to be generated. To visualize the path visited by each individual jellyfish, assign the animation to an object (`res <- animate_xy(...)`). This will save all 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). Users can then view the path of individual jellyfish through extracting the relevant bases recorded and construct it as a planned tour (using the `planned_tour()` function). Users may wish to view the optimisation path of the most successful jellyfish, or a random selection of jellyfish, or some specific

runs that find local maximum.

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

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

### 7.2.1. Smoothness

To sample bases for calculating smoothness, the following inputs 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 bases and index values. Parallelization is available to speed up the index value calculation 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 bases, 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 calculating squintability includes an additional step of interpolating between the sampled bases and the optimal 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 squares (`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 fitting. For the parametric case, the output provides the estimated parameters for the logistic function ( $\theta_1$  to  $\theta_4$ ) 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 JSO. The metrics will be generally useful for characterizing 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 iteration 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.

Future work can focus on developing more efficient optimization algorithm for noise indexes with low squintability metric, which resemble a needle-in-a-haystack problem (Siemann et al. 2023). Additionally, if the index function is differentiable, automatic differentiation can be used to implement new gradient-based optimizers (Prince 2023).

## 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/huizehang-sherry/paper-jso>. The simulation data produced in Section 5 can be found at [https://figshare.com/articles/dataset/Simulated\\_raw\\_data/26039506](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** (Wickham, Vaughan, and Girlich 2024), **dplyr** (Wickham et al. 2023), **ggplot2** (Wickham 2016), **knitr** (Xie 2014), **gtsummary** (Sjoberg et al. 2021), **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, (3) the table of the Gaussian process parameters and logistic function parameters for all 23 PP problems investigated, along with (4) the full code to reproduce the plots and summaries in this paper.

## References

- Abdulah, S., Y. Li, J. Cao, H. Ltaief, D. Keyes, M. Genton, and Y. Sun. 2023. “Large-Scale Environmental Data Science with ExaGeoStatR.” *Environmetrics* 34 (1): e2770. <https://doi.org/10.1002/env.2770>.
- Adams, R., and J. Fournier. 2003. *Sobolev Spaces*. Vol. 140. Pure and Applied Mathematics. Elsevier.
- Allaire, J. J., C. Teague, C. Scheidegger, Y. Xie, and C. Dervieux. 2022. *Quarto* (version 1.2). <https://doi.org/10.5281/zenodo.5960048>.
- Bertsimas, D., and J. Tsitsiklis. 1993. “Simulated Annealing.” *Statistical Science* 8 (1): 10–15. <https://doi.org/10.1214/ss/1177011077>.
- Chou, J.-S., and D.-N. 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/>
- 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, K. 2016. “Kennzahlenbasierte Grafikauswahl.” Doctoral thesis, Universität Augsburg.
- Grochowski, M., and W. Duch. 2011. “Fast Projection Pursuit Based on Quality of Projected Clusters.” In *Adaptive and Natural Computing Algorithms*, edited by A. Dobnikar, U. Lotrič, and B. Šter, 89–97. Berlin, Heidelberg: Springer Berlin Heidelberg. [https://doi.org/10.1007/978-3-642-20267-4\\_10](https://doi.org/10.1007/978-3-642-20267-4_10).
- Guinness, J., M. Katzfuss, and Y. Fahmy. 2021. “GpGp: Fast Gaussian Process Computation Using Vecchia’s Approximation.” R package. <https://cran.r-project.org/package=GpGp>.
- Hall, P. 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, P. 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, T. 2023. “Asymptotic Bounds for Smoothness Parameter Estimates in Gaus-

- sian 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, U., and D. 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>.
- Laa, U., D. Cook, A. Buja, and G. 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>.
- Lee, E.-K., and D. 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, E.-K., D. Cook, S. Klinke, and T. 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, N. 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, S. L., A. Berro, and A. Ruiz-Gazen. 2010. “An Efficient Optimization

- Method for Revealing Local Optima of Projection Pursuit Indices.” In *Swarm Intelligence*, edited by M. Dorigo, M. Birattari, G. A. Di Caro, R. Doursat, A. P. Engelbrecht, D. Floreano, L. M. Gambardella, et al., 60–71. Berlin, Heidelberg: Springer Berlin Heidelberg. [https://doi.org/10.1007/978-3-642-15461-4\\_6](https://doi.org/10.1007/978-3-642-15461-4_6).
- Mason, H., S. Lee, U. Laa, and D. Cook. 2022. *cassowaryr: Compute Scagnostics on Pairs of Numeric Variables in a Data Set*. <https://CRAN.R-project.org/package=cassowayr>.
- Pedersen, T. L. 2024. *patchwork: The Composer of Plots*. <https://CRAN.R-project.org/package=patchwork>.
- Porcu, E., M. Bevilacqua, R. Schaback, and C. 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, C. 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](https://doi.org/10.1016/0167-9473(95)00002-8).
- Prince, Simon JD. 2023. *Understanding Deep Learning*. MIT press.
- 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, K., K. Deep, and S. 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, C. E., and C. K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. The MIT Press.
- Robinson, D., A. Hayes, and S. Couch. 2024. *broom: Convert Statistical Objects into Tidy Tibbles*. <https://CRAN.R-project.org/package=broom>.
- Siemann, Alexander E, Zekun Ren, Qianxiao Li, and Tonio Buonassisi. 2023. “Fast

- Bayesian Optimization of Needle-in-a-Haystack Problems Using Zooming Memory-Based Initialization (ZoMBI).” *Npj Computational Materials* 9 (1): 79.
- Sjoberg, D. D., K. Whiting, M. Curry, J. A. Lavery, and J. Larmarange. 2021. “Reproducible Summary Tables with the gtsummary Package.” *The R Journal* 13: 570–80. <https://doi.org/10.32614/RJ-2021-053>.
- 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>.
- Van Den Brand, T. 2024. *ggh4x: Hacks for ggplot2*. <https://CRAN.R-project.org/package=ggh4x>.
- Wickham, H. 2016. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. <https://ggplot2.tidyverse.org>.
- 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>.
- Wickham, H., R. François, L. Henry, K. Müller, and D. Vaughan. 2023. *dplyr: A Grammar of Data Manipulation*. <https://CRAN.R-project.org/package=dplyr>.
- Wickham, H., D. Vaughan, and M. Girlich. 2024. *tidyr: Tidy Messy Data*. <https://CRAN.R-project.org/package=tidyr>.
- 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, L., and G. Wills. 2008. “Scagnostics Distributions.” *Journal of Computational and Graphical Statistics* 17 (2): 473–91. <https://doi.org/10.1198/106186008X320465>.

- Xie, Y. 2014. “knitr: A Comprehensive Tool for Reproducible Research in R.” In *Implementing Reproducible Computational Research*, edited by V. Stodden, F. Leisch, and R. D. Peng. Chapman; Hall/CRC.
- Zhang, H. S., D. Cook, U. Laa, N. Langrené, and P. 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, H. 2024. *kableExtra: Construct Complex Table with kable and Pipe Syntax*. <https://CRAN.R-project.org/package=kableExtra>.