User Manual

Penalised piecewise-linear (PPL) model for non-stationary extreme value analysis of peaks-of-threshold

Ed Mackay[1]

[1]College of Engineering, Mathematics and Physical Sciences, University of Exeter

January 2022

This work was funded by the UK EPSRC Supergen Offshore Renewable Energy Hub, project EP/S000747/1 on "Improved Models for Multivariate Metocean Extremes (IMEX)".

Contents

1 Introduction

This user guide describes MATLAB code for extreme value analysis of peaks-over-threshold (POT) with covariate dependence. The code estimates a model for extreme values of a response variable, Y, as a function of covariates, $\mathbf{X} = (X_1, X_2, ..., X_D)$. In the present implementation of the code, the covariate vector, $\mathbf{X}$, is assumed to have one or two dimensions, i.e. $D = 1$ or 2. Response values exceeding a high threshold are modelled using a generalised Pareto (GP) distribution, with scale and shape parameters assumed to be piecewise-linear functions of the covariate(s). The scale and shape parameters are estimated using maximum likelihood, penalised for the roughness of the estimates. The roughness penalty coefficients are estimated using a cross validation procedure to optimise the predictive performance of the model. The resulting model is referred to as the penalised piecewise-linear (PPL) model. In the present implementation, the covariates are assumed to be periodic.

The theoretical motivation for the model and inference method are described in detail in [1]. This document describes the MATLAB code and provides worked examples. Questions should be addressed to e.mackay@exeter.ac.uk.

2 Model description

The joint density of covariates $\mathbf{X}$ and response Y can be written in conditional form, as

$$f_{\mathbf{X},Y}(\mathbf{x}, y) = f_{\mathbf{X}}(\mathbf{x}) f_{Y|\mathbf{X}}(y|\mathbf{x}), \tag{2.1}$$

where $f_{\mathbf{X}}(\mathbf{x})$ is the density of covariates, and $f_{Y|\mathbf{X}}(y|\mathbf{x})$ is the density of the response, conditional on the covariates. We assume that the tail of $f_{Y|\mathbf{X}}(y|\mathbf{x})$ converges to a GP distribution. Our model for exceedances of high non-stationary threshold $u(\mathbf{x})$, can therefore be written $f_{Y|\mathbf{X}}(y|\mathbf{x}) = \zeta(\mathbf{x}) f_{\text{GP}}\left(y \,|\, u(\mathbf{x}), \sigma(\mathbf{x}), \xi(\mathbf{x})\right)$, $y > u(\mathbf{x})$, where $\zeta(\mathbf{x}) = \Pr\left(Y > u(\mathbf{x}) \,|\, \mathbf{X} = \mathbf{x}\right)$ is the threshold exceedance probability, and f_{GP} is the density of the GP distribution:

$$f_{\text{GP}}(y|u(\mathbf{x}), \sigma(\mathbf{x}), \xi(\mathbf{x})) = \begin{cases} \dfrac{1}{\sigma(\mathbf{x})} \left[1 + \xi(\mathbf{x}) \left(\dfrac{y - u(\mathbf{x})}{\sigma(\mathbf{x})}\right)\right]_+^{-1-1/\xi(\mathbf{x})}, & \xi(\mathbf{x}) \neq 0, \\ \dfrac{1}{\sigma(\mathbf{x})} \exp\left(-\left(\dfrac{y - u(\mathbf{x})}{\sigma(\mathbf{x})}\right)\right), & \xi(\mathbf{x}) = 0, \end{cases} \tag{2.2}$$

where $[\cdot]_+ = \max\{\cdot, 0\}$, and $\xi(\mathbf{x})$, $\sigma(\mathbf{x}) > 0$ and $u(\mathbf{x})$ are the shape, scale and threshold parameters. In the present implementation, we assume that threshold exceedance probability, ζ, is fixed at a user-defined level, constant with $\mathbf{x}$. Hence, for $y > u(\mathbf{x})$, our model for the joint density is

$$f_{\mathbf{X},Y}(\mathbf{x}, y) = \zeta \, f_{\mathbf{X}}(\mathbf{x}) \, f_{\text{GP}}\left(y \,|\, u(\mathbf{x}), \sigma(\mathbf{x}), \xi(\mathbf{x})\right), \quad y > u(\mathbf{x}). \tag{2.3}$$

Inference requires estimation of the covariate density, $f_{\mathbf{X}}(\mathbf{x})$, a non-stationary threshold $u(\mathbf{x})$, and non-stationary GP model with parameters $\sigma(\mathbf{x})$ and $\xi(\mathbf{x})$. In the present implementation of the PPL model, we also assume that covariates are periodic and therefore never extreme, so that kernel density estimation can be used to estimate $f_{\mathbf{X}}(\mathbf{x})$. Moreover, since ζ is assumed fixed, the threshold, $u(\mathbf{x})$, can be estimated as a local quantile of Y on the covariate domain. It then remains to estimate the GP model, with parameters $\sigma(\mathbf{x})$ and $\xi(\mathbf{x})$ which vary systematically as a function of covariate, taking piecewise-linear forms, with respect to a set of user-defined nodes.

Estimation of the model is split into four steps, described in Sections 4.1-4.4 below:

1. Estimation of covariate density and response threshold
2. Definition of node positions and triangulation
3. Cross-validation to select optimal roughness penalties
4. Model fitting and diagnostics

Two sets of functions are provided, for estimating the model with either one or two covariates. The use of the functions are illustrated using four examples scripts, corresponding to the four stages of estimation described above. The data used in the worked examples is described in Section 3.

3 Example data

The use of the PPL model is illustrated using data from the NORA10 hindcast [2], for a location in the northern North Sea. The hindcast provides time-series of significant wave height and dominant wave direction, for three hour sea-states for the 54-year period 1957-2010. Throughout, "direction" refers to the direction from which a storm travels expressed in degrees clockwise with respect to north. Since significant wave height is serially-correlated between sea states, storm peak significant wave height have been isolated from the hindcast time-series using the procedure described in [3]. Contiguous intervals of significant wave height above a low peak-picking threshold are identified, each interval now assumed to correspond to a storm event. The peak-picking threshold corresponds to a directional-seasonal quantile of significant wave height with specified non-exceedance probability, estimated using quantile regression. The maximum of significant wave height during the storm interval is taken as the storm peak significant wave height (henceforth H_s). The values of directional and seasonal covariates at the time of storm peak significant wave height are referred to as storm peak values of those variables. The resulting storm peak sample consists of 5388 values of H_s, with concurrent values of season and direction. The values are saved in the file `North_Sea_Data.mat`. The values of H_s are in metres. Storm-peak direction is in degrees North. Season is defined as the day of the year, for a standardised year consisting of 360 days.

4 Model fitting

4.1 Step 1: Estimation of covariate density and response threshold

In the first step of the estimation, the input data structure is defined and the covariate density, $f_{\mathbf{X}}(\mathbf{x})$, and extreme value threshold, $u(\mathbf{x})$, are estimated.

In the present version of the code, the following assumptions are made:

1. Covariates are assumed to be periodic in $[0, 360)$
2. Input values of the response are independent peak values

Based on these assumptions, covariate inputs must be mapped to $[0, 360)$, and peak values must be pre-identified using some method, such as that described in [3].

4.1.1 One-covariate code

The input data to the code is stored in a MATLAB structure with the following fields:

- `DATA.X` = $[N_{obs} \times 1]$ array of covariate data
- `DATA.Y` = $[N_{obs} \times 1]$ array of response data
- `DATA.Nyears` = length of dataset in years
- `DATA.name.dataset` = name of dataset (text)
- `DATA.name.X` = name of covariate (text)
- `DATA.name.Y` = name of response (text)
- `DATA.unit.X` = unit of covariate (text)
- `DATA.unit.Y` = unit of response (text)

The `name` and `unit` fields are used in the plotting functions to label axes. The covariate density and extreme value threshold are estimated on a user-defined grid of points. The covariate density is estimated using kernel density estimation, with a user-specified bandwidth. The threshold is estimated as a local quantile, with exceedance probability ζ, based on the C nearest observations (in terms of the covariate value) to each grid point. The initial threshold estimate is then smoothed using a Gaussian kernel, with user-specified bandwidth. Initial estimates of the GP scale and shape parameters are also produced, used to inform the choice of node positions in Step 2. The initial estimates of the GP parameters are the moment estimators,

based on the C nearest values to each grid point, under the assumption of a locally-stationary distribution. The settings are input as follows

- `DATA.grid.x` = $[1 \times N_{grid}]$ array of grid points in $[0, 360]$
- `DATA.grid.Nnearest` = number of nearest points, C, for threshold and GP parameter estimates
- `DATA.grid.KD_bw` = bandwidth for kernel density estimate of covariate PDF
- `DATA.grid.Thresh_bw` = bandwidth for smoothing threshold estimate
- `DATA.thresh_NEP` = threshold non-exceedance probability $= 1 - \zeta$

The `DATA` structure is then passed to the function `set_initial_gridded_estimates_1cov`:

```
DATA=set_initial_gridded_estimates_1cov(DATA);
```

This function adds the following fields to the `DATA` structure:

- `DATA.grid.pdf` = $[1 \times N_{grid}]$ array of kernel density estimates of covariate PDF, $f_X(x)$
- `DATA.grid.thresh_raw` = $[1 \times N_{grid}]$ array of initial estimates of non-stationary threshold $u(x)$
- `DATA.grid.thresh` = $[1 \times N_{grid}]$ array of smoothed estimates of non-stationary threshold $u(x)$
- `DATA.grid.sigma` = $[1 \times N_{grid}]$ array of initial estimates of GP scale parameter $\sigma(x)$
- `DATA.grid.xi` = $[1 \times N_{grid}]$ array of initial estimates of GP scale parameter $\xi(x)$
- `DATA.exceedance.Z` = $[N_{exc} \times 1]$ array of threshold exceedances $y_i - u(\mathbf{x}_i)$, $i = 1, ..., N_{exc}$, where $N_{exc} < N_{obs}$ is the number of observations exceeding the threshold
- `DATA.exceedance.X` = $[N_{exc} \times 2]$ array of covariate values, x_i, corresponding to threshold exceedances, $i = 1, ..., N_{exc}$
- `DATA.exceedance.thresh` = $[N_{exc} \times 1]$ array of non-stationary threshold values $u(\mathbf{x}_i)$, $i = 1, ..., N_{exc}$, corresponding to threshold exceedances

The gridded estimates of the covariate density, non-stationary threshold and initial estimates of the GP parameters, can be visualised using the function

```
plot_initial_gridded_estimates_1cov(DATA,type),
```

where `type` is a text field, set as either `'pdf'`, `'thresh'`, `'sigma'` or `'xi'`. An example application for the North Sea data, described in Section 3, is provided in the script `Step_1_initial_gridded_estimates_1cov.m`. Examples of the visualisations are shown in Figure 1.

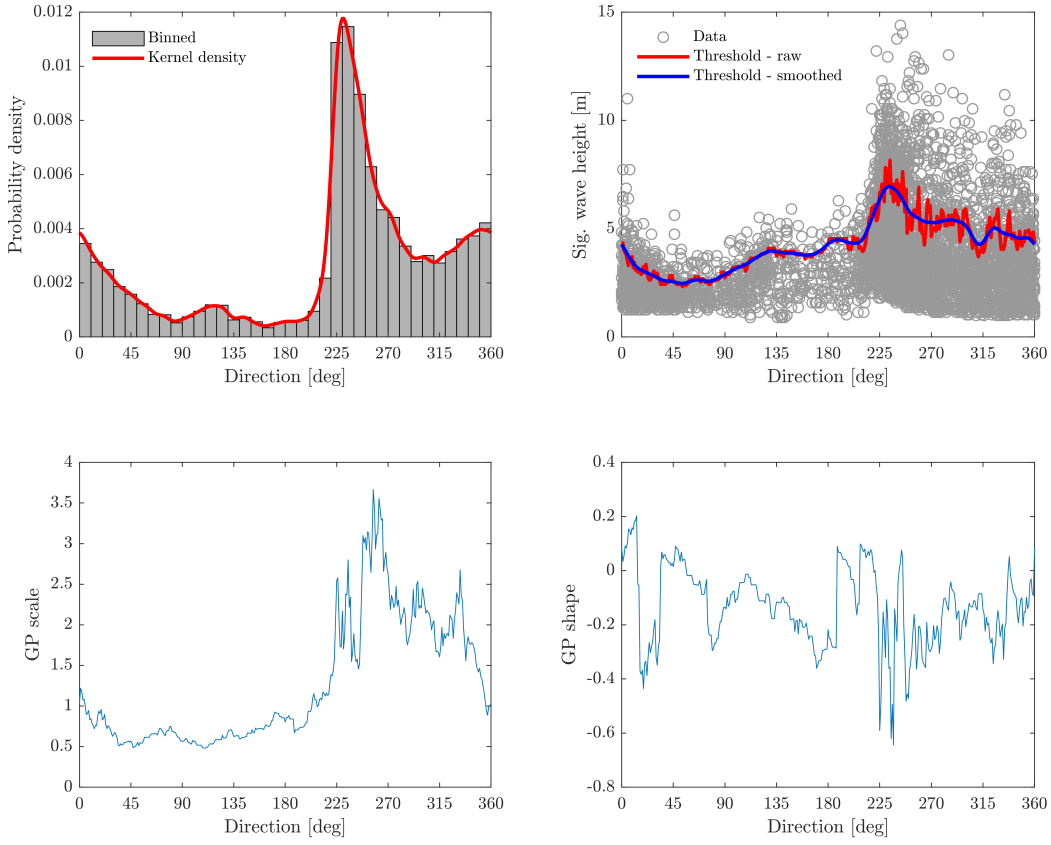

Figure 1: *Estimates of covariate density (top left), non-stationary GP threshold (top right), and GP scale (bottom left) and shape parameters (bottom right), for the North Sea dataset.*

4.1.2 Two-covariate code

The two-covariate code follows a similar format to the one-covariate code. The input data to the code is stored in a MATLAB structure with the following fields:

- `DATA.X` = $[N_{obs} \times 2]$ array of covariate data
- `DATA.Y` = $[N_{obs} \times 1]$ array of response data
- `DATA.Nyears` = length of dataset in years
- `DATA.name.dataset` = name of dataset (text)
- `DATA.name.X1` = name of 1st covariate (text)
- `DATA.name.X2` = name of 2nd covariate (text)
- `DATA.name.Y` = name of response (text)
- `DATA.unit.X1` = unit of 1st covariate (text)
- `DATA.unit.X2` = unit of 2nd covariate (text)
- `DATA.unit.Y` = unit of response (text)

The covariate density, extreme value threshold, and initial estimates of the GP scale and shape parameters are estimated in the same way as for the 1-covariate code. The settings for the estimation are input as follows

- `DATA.grid.x1` = $[1 \times N_{grid,1}]$ array of grid points for 1st covariate in $[0, 360]$
- `DATA.grid.x2` = $[1 \times N_{grid,2}]$ array of grid points for 2nd covariate in $[0, 360]$
- `DATA.grid.Nnearest` = number of nearest points, C, for threshold and GP parameter estimates
- `DATA.grid.KD_bw` = bandwidth for kernel density estimate of covariate PDF
- `DATA.grid.Thresh_bw` = bandwidth for smoothing threshold estimate
- `DATA.grid.sigma_bw` = bandwidth for smoothing GP scale estimate
- `DATA.thresh_NEP` = threshold non-exceedance probability = $1 - \zeta$

The `DATA` structure is then passed to the function `set_initial_gridded_estimates_2cov`:

```
DATA=set_initial_gridded_estimates_2cov(DATA);
```

This function adds the following fields to the `DATA` structure:

- `DATA.grid.pdf` $= [N_{grid,2} \times N_{grid,1}]$ array of kernel density estimates of covariate PDF, $f_{\mathbf{X}}(\mathbf{x})$
- `DATA.grid.thresh_raw` $= [N_{grid,2} \times N_{grid,1}]$ array of initial estimates of non-stationary threshold $u(\mathbf{x})$
- `DATA.grid.thresh` $= [N_{grid,2} \times N_{grid,1}]$ array of smoothed estimates of non-stationary threshold $u(\mathbf{x})$
- `DATA.grid.sigma_raw` $= [N_{grid,2} \times N_{grid,1}]$ array of initial estimates of GP scale parameter $\sigma(\mathbf{x})$
- `DATA.grid.sigma` $= [N_{grid,2} \times N_{grid,1}]$ array of smoothed estimates of GP scale parameter $\sigma(\mathbf{x})$
- `DATA.grid.xi` $= [N_{grid,2} \times N_{grid,1}]$ array of initial estimates of GP scale parameter $\xi(\mathbf{x})$
- `DATA.exceedance.Z` $= [N_{exc} \times 1]$ array of threshold exceedances $y_i - u(\mathbf{x}_i)$, $i = 1, ..., N_{exc}$, where $N_{exc} < N_{obs}$ is the number of observations exceeding the threshold
- `DATA.exceedance.X` $= [N_{exc} \times 2]$ array of covariate values, $\mathbf{x}_i$, corresponding to threshold exceedances, $i = 1, ..., N_{exc}$
- `DATA.exceedance.thresh` $= [N_{exc} \times 1]$ array of non-stationary threshold values $u(\mathbf{x}_i)$, $i = 1, ..., N_{exc}$, corresponding to threshold exceedances

The gridded estimates of the covariate density, non-stationary threshold and initial estimates of the GP parameters, can be visualised using the function

```
plot_initial_gridded_estimates_2cov(DATA,type),
```

where `type` is a text field, set as either `'pdf'`, `'thresh'`, `'sigma'` or `'xi'`. An example application for the North Sea data, described in Section 3, is provided in the script `Step_1_initial_gridded_estimates_2cov.m`. Examples of the visualisations are shown in Figure 2.

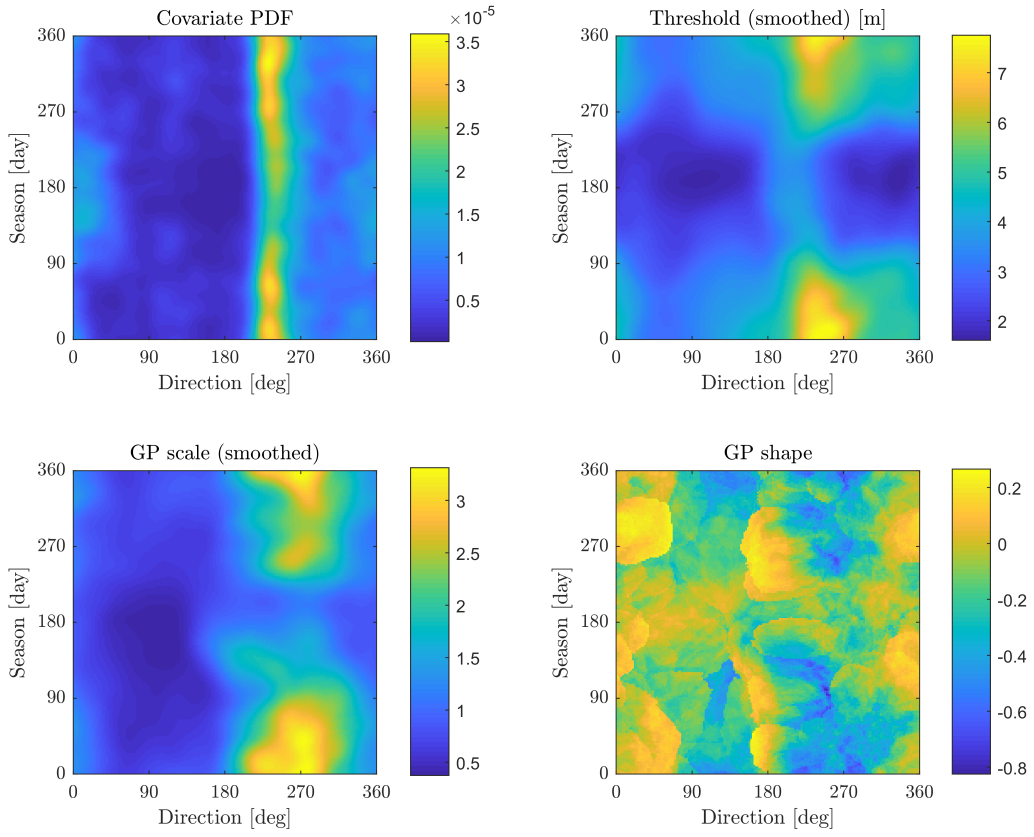

Figure 2: *Estimates of covariate density (top left), non-stationary GP threshold (top right), and GP scale (bottom left) and shape parameters (bottom right), for the North Sea dataset.*

4.2 Step 2: Setting node positions and triangulation

A critical precursor to successful inference of the GP scale and shape parameters, is reasonable node placement on the covariate domain, $\mathcal{D}$. It is recommended that nodes are placed at local maxima or minima, or turning points of the initial gridded estimates of the GP scale and shape parameters, calculated in Step 1. This section describes how nodes should be located for the one- and two-covariate codes.

4.2.1 One-covariate code

In the one-covariate code, the node locations are specified using the command

```
[DATA, GRID]=create_grid(DATA,nodes);
```

where `DATA` is the structure created in Step 1, and `nodes` is a $1 \times K$ array of values of the covariate, where nodes are located. This function adds the following fields to the `DATA` structure:

- `DATA.exceedance.bin_num` = $[N_{exc} \times 1]$ array of bin numbers for each exceedance
- `DATA.exceedance.nearest_node` = $[N_{exc} \times 1]$ array of indices of the nearest node to each exceedance

The `GRID` structure contains the fields

- `GRID.nodes.position` = $[1 \times (K+1)]$ array of node locations (last node location is equal to location of first node +360)
- `GRID.nodes.number` = $[1 \times (K+1)]$ array of node numbers
- `GRID.Nnodes` = number of nodes, K

Initial estimates of the GP scale and shape parameters at the node locations are estimated using a Voronoi partition of $\mathcal{D}$, with bins corresponding to the set of points closest to each node. Maximum likelihood estimates (MLEs) are calculated under the assumptions of either fixed or variable GP shape. The initial estimates at the nodes are set using the following command

```
DATA=set_initial_GP_voronoi_1cov(DATA);
```

This function adds the following fields to the `DATA` structure:

- `DATA.voronoi.xi_const.sigma` = $[K \times 1]$ array of initial estimates of GP scale parameter at node locations, under assumption of constant shape parameter

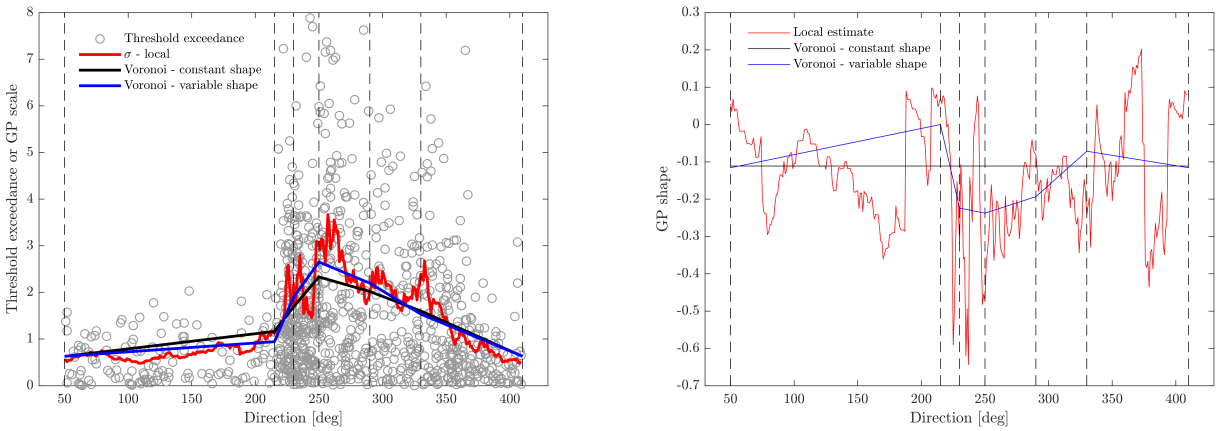

Figure 3: *Comparisons of the initial estimates of the GP scale and shape parameters at the node locations (dashed lines), with the initial gridded estimates, for the North Sea dataset.*

- `DATA.voronoi.xi_const.xi` = initial estimate of constant GP shape parameter
- `DATA.voronoi.xi_vary.sigma` = $[K \times 1]$ array of initial estimates of GP scale parameter at node locations, under assumption of variable shape parameter
- `DATA.voronoi.xi_vary.xi` = $[K \times 1]$ array of initial estimates of GP shape parameter at node locations, under assumption of variable shape parameter

The initial estimates of the GP parameters at the node locations can be compared to the initial gridded estimate, calculated in Step 1, using the command

```
plot_grid_checks_1cov(DATA,GRID)
```

Examples of the outputs for the North Sea data are shown in Figure 3.

4.2.2 Two-covariate code

Node placement in 2D presents a greater challenge than in 1D. Two approaches to node placement are available in the present implementation. The first approach, referred to as a "regular grid", is defined in terms of two sets of marginal node locations (on $[0, 360)$ here), used together to produce a regular lattice on $\mathcal{D} = [0, 360) \times [0, 360)$. The first node location in each set is also used to locate an additional "wrapping node", located at the location of the first node plus 360, to accommodate periodicity on $\mathcal{D}$. The sets of marginal nodes are then used to create a regular rectangular grid of nodes on $\mathcal{D}$. Extra nodes are then added at the centres of each rectangle created, producing a (periodic) triangulation of $\mathcal{D}$. The regular grid is attractive because it is specified in terms of sets of marginal nodes which are relatively straightforward to locate.

The second approach to node specification, referred to as an "irregular grid", permits user-specified node location on $\mathcal{D}$. This approach is obviously more flexible, and potentially more parsimonious than the regular grid, but requires judicious choice of node locations. Once the irregular node locations are specified, "wrapping nodes" are created, with co-ordinates shifted by ± 360 relative to those of the specified nodes in each covariate dimension. A Delaunay triangulation of $\mathcal{D}$ is then created using the specified and wrapping nodes.

For the two-covariate code, the node locations and triangulation is specified using the command:

```
[DATA, GRID]=create_grid(DATA,regular,nodes);
```

where

- `regular` $= [1 \times 1]$ logical variable, set as 'true' to create a regular grid or 'false' to create an irregular grid
- `nodes` = structure with fields
  - `nodes.x1` $= [K_1 \times 1]$ array
  - `nodes.x2` $= [K_2 \times 1]$ array

In the case of a regular grid, the fields `nodes.x1` and `nodes.x2` specify the marginal node locations, and can have different lengths. In the case of an irregular grid, these fields specify the coordinates of the nodes, so that $K_1 = K_2 := K$. The function `create_grid` updates the `DATA` structure with the following fields:

- `DATA.exceedance.bin_num` $= [N_{exc} \times 1]$ array of bin numbers for each exceedance, where some bins may be copies of others, with nodes shifted ± 360
- `DATA.exceedance.bin_num_unique` $= [N_{exc} \times 1]$ array of bin numbers for each exceedance, where all bins are unique
- `DATA.exceedance.nearest_node` $= [N_{exc} \times 1]$ array of indices of the nearest node to each exceedance

The `GRID` structure contains the fields

- `GRID.isregular` $= [1 \times 1]$ logical variable, 'true' for a regular grid, or 'false' for an irregular grid.
- `GRID.Ntri_unique` $= M$, number of unique triangles or 'bins'
- `GRID.Ntri_repeated` $= M_r$, number of triangles, including repeats
- `GRID.Nnode_unique` $= K$, number of unique nodes
- `GRID.Nnode_repeated` $= K_r$, number of nodes, including repeats

- `GRID.interp_coef` $= [3 \times 3 \times M_r]$, array of interpolation coefficients for each bin
- `GRID.nodes` structure with fields
  - `GRID.nodes.position` $= [K \times 2]$ array of unique node locations
  - `GRID.nodes.position_repeated` $= [K_r \times 2]$ array of node locations, including repeats
  - `GRID.nodes.number` $= [K_r \times 1]$ array of indices of unique nodes, to which each repeated node corresponds
- `GRID.tri` structure defining the triangulation of the nodes, with fields
  - `GRID.tri.indices` $= [M_r \times 3]$ array of node indices for each triangle
  - `GRID.tri.number` $= [M_r \times 1]$ array of unique bin numbers $1, ..., M$ corresponding to bin, including repeats

Examples of regular and irregular 2D grids are shown in Figure 4. Unique nodes are shown as red dots; all other nodes are repeats, included for ease of comprehension. Threshold exceedances are shown in blue. For ease of illustration on the regular grid, locations of observations with x-coordinate $\leq 120°$ and/or y-coordinate ≤ 20 days, where $(120°, 20\text{days})$ corresponds to the node location nearest the origin) have been shifted by $+360$. For ease of use of the irregular grid, the triangulation is extended by repeating node locations ± 360 in each covariate dimension.

Initial estimates of the GP scale and shape parameters at the node locations are estimated using a Voronoi partition of $\mathcal{D}$, in the same way as for the 1D code. The initial estimates at the nodes are set using the following command

```
DATA=set_initial_GP_voronoi_2cov(DATA);
```

This function adds the updates the `DATA` structure in the same way as the 1D code. The initial estimates of the GP parameters at the node locations can be compared to the initial gridded estimate, calculated in Step 1, using the command

```
plot_grid_checks_2cov(DATA,GRID)
```

Examples of regular and irregular 2D grids are shown in Figure 4. As for 1D covariates, irregular nodes in two dimensions should be placed where the local gradient of a model parameter with covariate is expected to change, i.e. at local turning points. Node locations can be selected by inspection, by comparing against the local estimates of GP scale. An example is shown in Figure 5 for the North Sea data. The surface shows initial local GP scale estimates from a local stationary GP fit to the $C = 50$ nearest observations to each grid point on $\mathcal{D}$, smoothed using Gaussian kernel, with common bandwidth of 10 degrees or days. The black grid lines emanating from nodes represent the irregular grid triangulation, and a piecewise-linear representation for GP scale. Node placement can be adjusted manually until reasonable agreement is obtained between the

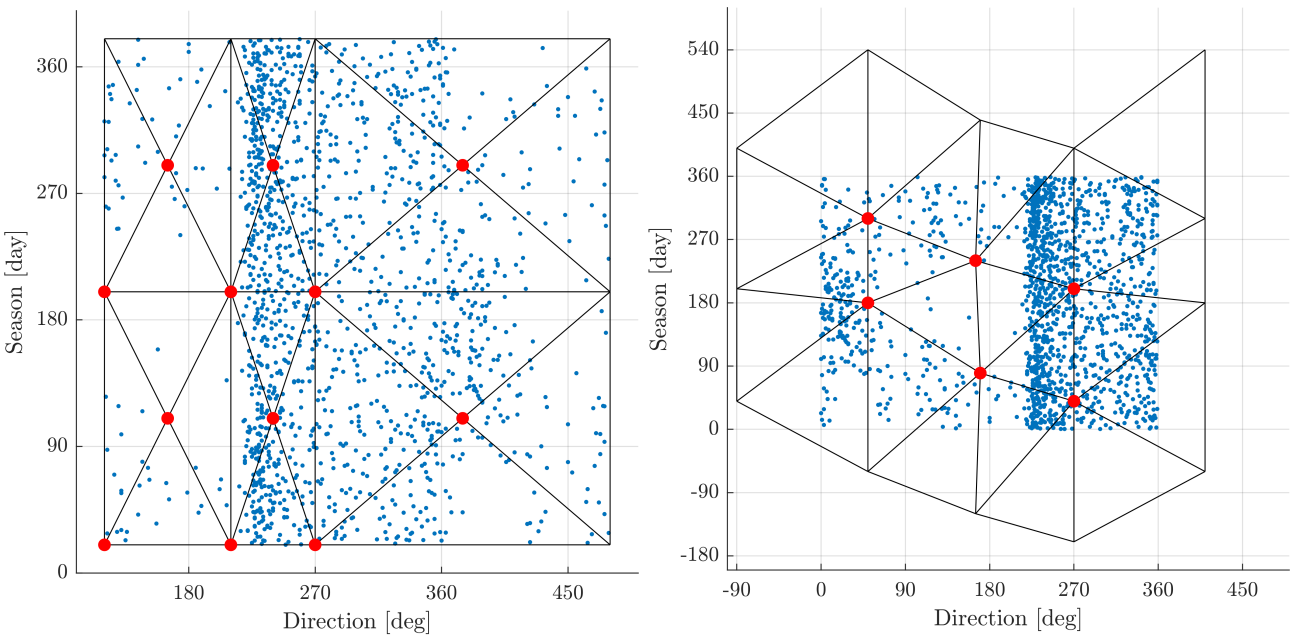

Figure 4: *Examples of regular and irregular 2D grids (left and right respectively) on extended covariate domain. Unique nodes are shown as red dots; all other nodes are repeats, included for ease of comprehension. Threshold exceedances are shown in blue. For ease of illustration on the regular grid, locations of observations with x-coordinate* $\leq 120°$ *and/or y-coordinate* ≤ 20 *days, where* $(120°, 20 days)$ *corresponds to the node location nearest the origin) have been shifted by* $+360$*. For ease of use of the irregular grid, the triangulation is extended by repeating node locations* ± 360 *in each covariate dimension.*

local estimates of scale, and that suggested by the piecewise linear triangulation. Comparisons of the initial local estimates of the scale and the starting values for the optimisation for a 3×2 regular grid are shown in Figure 6.

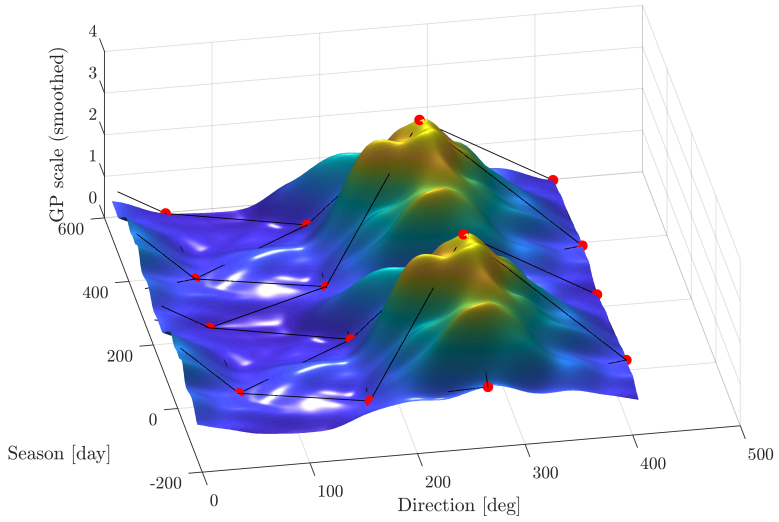

Figure 5: *Surface: Gaussian-kernel-smoothed initial local GP scale estimates. Grid lines between nodes (black lines) visualise a piecewise-linear scale representation. Reasonable agreement between actual value of scale, and that suggested by the piecewise linear triangulation, suggests good node placement. An extended covariate domain is used to aid interpretation.*

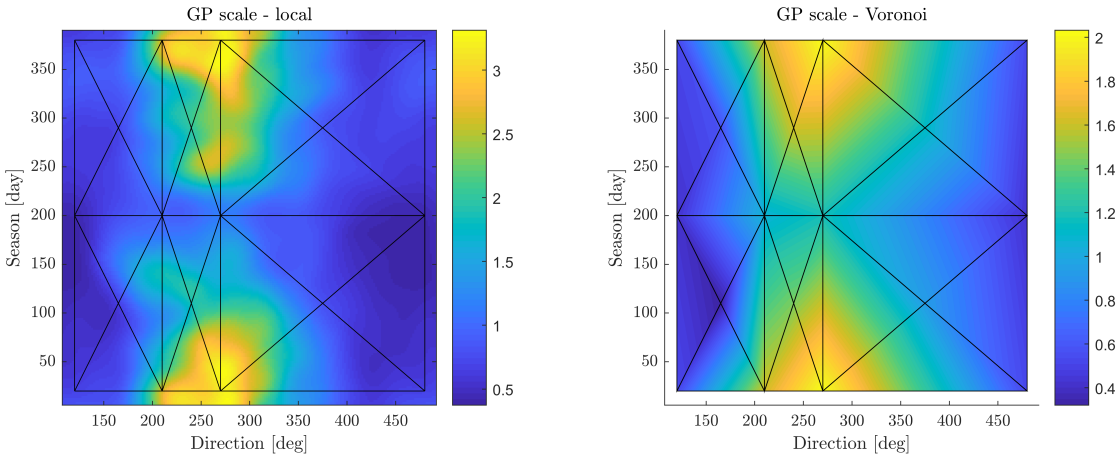

Figure 6: *Comparison of initial local estimates of GP scale (left), and starting values for optimisation (right) based on the Voronoi partition of the covariate domain, with respect to a regular* 3×2 *grid. Grid lines are overlaid on the initial local estimates for comparison.*

4.3 Step 3: Cross validation

Estimates of the GP scale and shape parameters at the nodes are found by maximising the sample likelihood, penalised for parameter roughness on $\mathcal{D}$. Optimal roughness penalties are estimated by cross-validation.

Consider a sample $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ of N conditionally-independent observations, to be used to estimate the model in (2.3). We denote the vector of GP parameters at the node locations as $\theta = (\sigma(\mathbf{n}_1), \ldots, \sigma(\mathbf{n}_K), \xi(\mathbf{n}_1), \ldots, \xi(\mathbf{n}_K))$, where $\mathbf{n}_i$ is the coordinates of the i^{th} node. The sample likelihood for $\mathcal{S}$ under the PPL model is

$$\mathcal{L}(\theta \,|\, \mathcal{S}) = \prod_{i \in \mathcal{I}} f_{\text{GP}}(y_i \,|\, u(\mathbf{x}_i), \sigma(\mathbf{x}_i), \xi(\mathbf{x}_i)) \tag{4.1}$$

where $\mathcal{I}$ is the index set of exceedances $\{i : y_i > u(\mathbf{x}_i)\}$, and for $\mathbf{x} \in B_m$, the m^{th} bin, parameters $\sigma(\mathbf{x})$ and $\xi(\mathbf{x})$ take piecewise-linear representations parameterised in terms of the corresponding bin nodes. The sample negative log-likelihood is denoted $\ell(\theta \,|\, \mathcal{S}) = -\log(\mathcal{L}(\theta \,|\, \mathcal{S}))$.

Roughness penalisation is used to smooth the variation of parameters on $\mathcal{D}$ for optimal predictive performance. This is quantified using the gradient of the parameter function in each covariate dimension, in each bin. The penalised negative log-likelihood becomes

$$\ell^*(\theta \,|\, \boldsymbol{\lambda}, \mathcal{S}) = \ell(\theta \,|\, \mathcal{S}) + \sum_{d=1}^{D} \sum_{m=1}^{M} \left(\lambda_{\sigma,d} \left| \frac{\partial \sigma(\mathbf{x})}{\partial x_d} \right|_{\mathbf{x} \in B_m} + \lambda_{\xi,d} \left| \frac{\partial \xi(\mathbf{x})}{\partial x_d} \right|_{\mathbf{x} \in B_m} \right) \tag{4.2}$$

where $\boldsymbol{\lambda} = (\lambda_{\sigma,1}, \ldots, \lambda_{\sigma,D}, \lambda_{\xi,1}, \ldots, \lambda_{\xi,D})$ is the vector of roughness penalties. For given $\boldsymbol{\lambda}$, constrained non-linear optimisation is used to minimise $\ell^*(\theta \,|\, \lambda, \mathcal{S})$, under the constraint $-0.5 \leq \xi(\mathbf{x}) < 0$, reasonable for environmental variables (with $\xi = -0.5$ corresponding to linear decrease in GP density with y, and $\xi < 0$ ensuring a finite upper bound for the distribution of Y). A large value of λ for some parameter-covariate pair imposes a large penalty on the corresponding parameter gradient, and hence drives smoother solutions for that parameter-covariate pair.

Section 4.4 describes how the parameter set θ is estimated, given roughness coefficients $\boldsymbol{\lambda}$. The value of $\boldsymbol{\lambda}$ is selected to maximise the predictive performance of the model using a G-fold cross-validation scheme. Cross-validation groups $\mathcal{S}_g$, $g = 1, 2, ..., G$ are formed by random partition of the sample $\mathcal{S}$ into G groups of approximately equal size.

The setup for the cross-validation is specified using the following structure:

- `CV.setup.Ngroups` = number of cross-validation groups, G
- `CV.setup.Npenalty` = number of values of lambda used per parameter and per dimension, L
- `CV.setup.Nrepeats` = number of repeats of cross-validation process (different random partitions), R
- `CV.setup.penLB` = lower bound on $\log_{10}$ scale for λ range (i.e. lowest value is 10^{penLB})
- `CV.setup.penUB` = upper bound on $\log_{10}$ scale for λ range (i.e. highest value is 10^{penUB})

- `CV.setup.penscale` = number of roughness penalties for GP scale parameter. `penscale=1` gives a common penalty for roughness in both dimensions (default option in 1D). `penscale=2` allows differing roughness penalties for scale for each covariate
- `CV.setup.penshape` = number of roughness penalties for GP shape parameter. `penshape=0` sets constant GP shape. `penshape=1` gives a common penalty for roughness in both dimensions (default option in 1D). `penshape=2` allows differing roughness penalties for shape for each covariate
- `CV.options` = options for constrained optimisation solver `fmnicon`, set using MATLAB `optimoptions` function, e.g. `options = optimoptions(@fmincon, 'Display', 'notify', 'MaxFunctionEvaluations', 1e5)`

The `CV` structure is the same for both the one- and two-covariate codes. The cross-validation procedure is run using the commands

```
CV = crossval_1cov(DATA,GRID,CV);
```

for the one-covariate code or

```
CV = crossval_2cov(DATA,GRID,CV);
```

for the two-covariate code. The `CV` structure is updated with the following fields:

- `CV.groups` = $[N_{exc} \times R]$ array of cross-validation groups for each threshold exceedance and each repeat.
- `CV.results` structure containing results of cross-validation, with fields
  - `CV.results.lambda` = $[L \times 1]$ array of values of λ considered for each parameter
  - `CV.results.predictive_likelihood` = $[R \times L \times L]$ array of values of predictive likelihood, corresponding to calculations for each repeat and each value of λ tried for each parameter/dimension
  - `CV.results.predictive_likelihood_min` = minimum value of mean predictive likelihood over R repeats
  - `CV.results.predictive_likelihood_min_jackknife_range` = uncertainty in minimum value of mean predictive likelihood, defined as range of jackknife estimates over R repeats
  - `CV.results.lambda_optimal` = value of λ resulting in minimum predictive likelihood
  - `CV.results.predictive_likelihood_accepted` = minimum value of mean predictive likelihood plus uncertainty from jackknife range
  - `CV.results.lambda_accepted` = all combinations of penalties that result in predictive likelihood below the accepted value

The results of the cross-validation procedure can be visualised using the commands

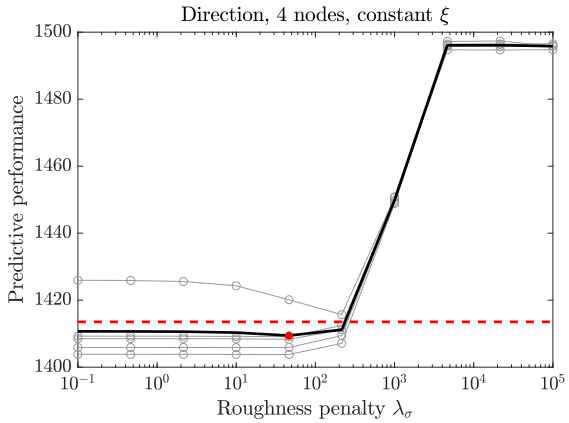

Figure 7: *Cross-validation results for 1D model. Predictive performance (negative log-likelihood) against roughness coefficient λ_σ, for directional model with $K = 4$ nodes. Grey lines indicate results for $R = 5$ replicate runs, and the solid black line is the mean over replicates. The dashed red line shows the 'accepted' value of the predictive performance. Red dot indicates minimum of the mean predictive performance.*

```
plot_cross_validation_results_1cov(CV)
```

or

```
plot_cross_validation_results_2cov(CV)
```

Examples of the outputs of the cross-validation procedure are shown in Figure 7.

4.4 Step 4: Model fitting and diagnostics

Once the optimal penalty coefficients have been estimated, using the cross-validation procedure, the parameter estimates at the nodes are estimated by minimising the sample penalised negative log likelihood (4.2). The model is fitted using the commands

```
param_nodes=fit_PPL_model_1cov(DATA,GRID,CV,Nboot);
```

or

```
param_nodes=fit_PPL_model_2cov(DATA,GRID,CV,Nboot);
```

The variable `Nboot` is the number of bootstrap trials for which the model is to be estimated. In the present implementation of the code, only the set of exceedances is resampled, so the uncertainty in the model fit is conditional on the estimated threshold and covariate density. The output `param_nodes` is an $K \times 2 \times Nboot$ array of GP scale and shape parameter estimates at the node locations, with the first entry in the second dimension of the array, corresponding to scale and the second corresponding to shape.

The parameter estimates can be visualised using the commands

```
plot_parameter_estimates_1cov(DATA,GRID,param_nodes,const_shape)
```

or

```
plot_parameter_estimates_2cov(DATA,GRID,param_nodes,const_shape)
```

The variable `const_shape` is a logical parameter, to indicate whether the GP shape is assumed constant (true) or not (false). Examples of the outputs for the 1D code are shown in Figure 8.

The conditional quantiles for the model can be plotted using the commands

```
plot_PPL_quantile_1cov(DATA,GRID,param_nodes,P))
```

or

```
plot_PPL_quantile_2cov(DATA,GRID,param_nodes,P))
```

For the 1D code, the variable `P` is an array of conditional probability values in $(1-\zeta, 1)$, whereas for the 2D code, `P` takes a single value in $(1-\zeta, 1)$. Examples of the outputs for the 1D code and 2D codes are shown in Figure 9 and Figure 10.

Threshold exceedances can be simulated from the model using the commands

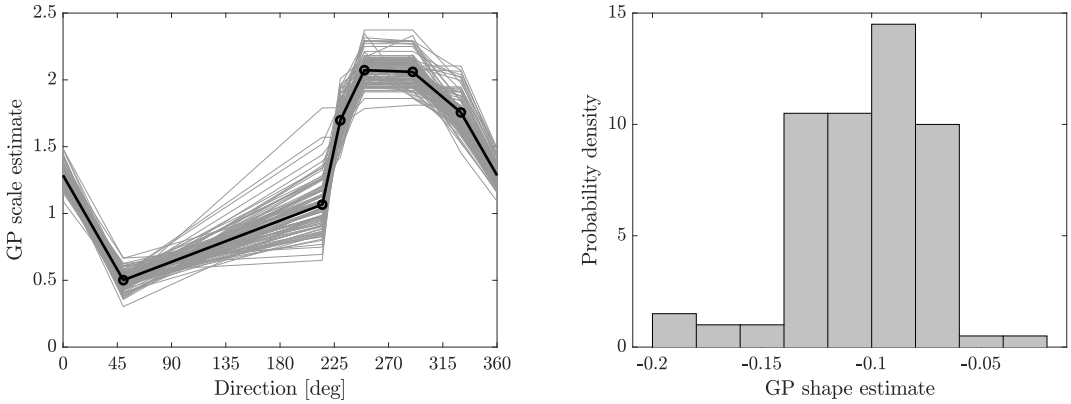

Figure 8: *Estimates of GP scale and shape parameters for 1D directional model with $K = 6$ nodes and constant shape parameter. Grey lines show estimates from each of 100 bootstrap resamples of the original sample. Bold lines show bootstrap means.*

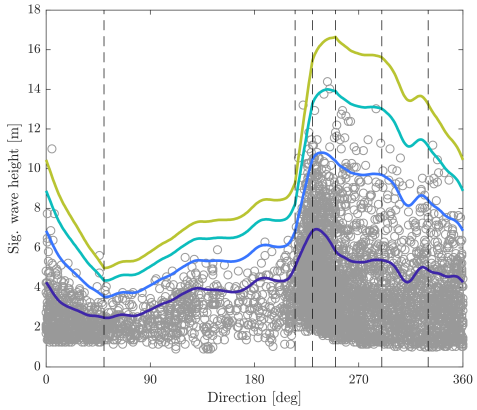

Figure 9: *Conditional quantiles of H_s for 1D directional model with $K = 6$ nodes. Coloured lines show quantiles at non-exceedance probabilities of 0.8 (corresponding to threshold, u), 0.9, 0.99 and 0.999. Circles are original observations. Vertical lines show node locations.*

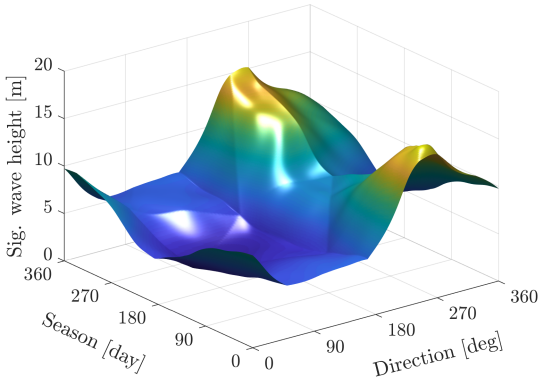

Figure 10: *Directional-seasonal conditional quantile of H_s for non-exceedance probability 0.99, for irregular 6-node grid and constant shape.*

```
DATA_sim=simulate_model_1cov(DATA,GRID,param_nodes,Npoints);
```

or

```
DATA_sim=simulate_model_1cov(DATA,GRID,param_nodes,Npoints);
```

where `Npoints` is the number of threshold exceedances to simulate. The output structure has fields

- `DATA_sim.X` = $[Npoints \times Nboot]$ array of covariate values for 1D case or = $[Npoints \times 2 \times Nboot]$ array for 2D case
- `DATA_sim.Y` = $[Npoints \times Nboot]$ array of response values
- `DATA_sim.Z` = $[Npoints \times Nboot]$ array of exceedance values ($= Y - U$)
- `DATA_sim.U` = $[Npoints \times Nboot]$ array of threshold values
- `DATA_sim.bin_num` = $[Npoints \times Nboot]$ array of bin numbers for each data point

Exceedance probabilities from the model and observations can be compared by bin, using the commands

```
plot_exceedance_by_bin_1cov(DATA,DATA_sim,GRID,xrange,yrange)
```

```
plot_exceedance_by_bin_2cov(DATA,DATA_sim,GRID,xrange,yrange)
```

where `xrange` and `yrange` are optional inputs to set the x- and y-axis limits on each plot. Alternatively, exceedance probabilities can be compared by month or by directional sector, using the commands

```
plot_exceedance_by_sector_1cov(DATA,DATA_sim,type,xrange,yrange)
```

```
plot_exceedance_by_sector_2cov(DATA,DATA_sim,xrange,yrange)
```

where for the 1D code, the input variable `type` is set as either `'season'` or `'direction'`. An example of a comparison of exceedance probabilities by directional sector for the 1D code is shown in Figure 11.

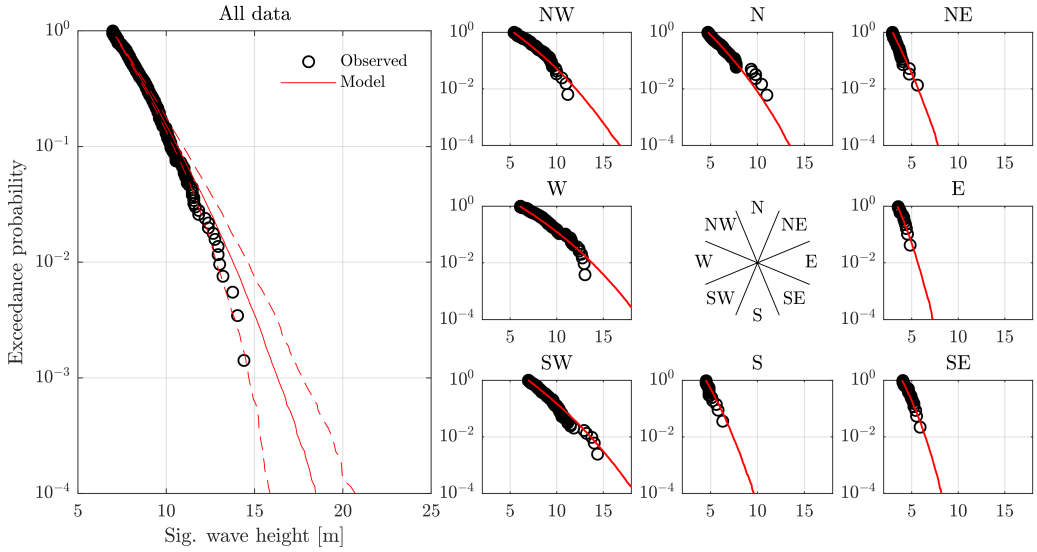

Figure 11: *Omnidirectional and directional exceedance probabilities for the 1D directional model with $K = 6$ nodes. Empirical estimates in black, and model-based estimates in red. Dashed lines on omnidirectional exceedance probability plot represent 95% uncertainty bands based on 100 bootstrap trials.*

5 List of abbreviations

- CDF - cumulative distribution function
- CV - cross validation
- GP - generalised Pareto
- MLE - maximum likelihood estimate
- PDF - probability density function
- POT - peaks-over-threshold
- PPL - penalised piecewise-linear

6 Nomenclature

Roman:

- $\mathbf{X} = (X_1, ..., X_D)$ random variable representing covariate data
- Y = random variable representing response data
- C = number of nearest observations for initial locally-stationary estimates of parameters
- B_m = covariate bin with index m
- D = number of covariate dimensions
- H_s = significant wave height
- M = number of covariate bins
- K = number of nodes
- N = number of observations
- u = threshold parameter for the generalised Pareto model

Greek:

- σ = scale parameter for the generalised Pareto model
- ξ = shape parameter for the generalised Pareto model
- ζ = threshold exceedance probability
- $\lambda_\sigma, \lambda_\xi$ = roughness penalty on scale and shape respectively

Script:

- $\mathcal{D} = [0, 360)^D$ = covariate domain
- $\mathcal{L}$ = sample likelihood
- $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ = sample of conditionally-independent observations
- ℓ = sample negative log-likelihood
- ℓ^* = sample penalised negative log-likelihood

References

[1] A. M. Barlow, E. Mackay, E. Eastoe, and P. Jonathan, *A penalised piecewise-linear model for non-stationary extreme value analysis of peaks over threshold*, 2022. arXiv: 2201.03915 [stat.ME].

[2] M. Reistad, O. Breivik, H. Haakenstad, O. J. Aarnes, B. R. Furevik, and J.-R. Bidlot, "A high-resolution hindcast of wind and waves for the North Sea, the Norwegian Sea, and the Barents Sea.," *J. Geophys. Res.*, vol. 116, pp. 1–18, C5 2011.

[3] K. C. Ewans and P. Jonathan, "The effect of directionality on northern North Sea extreme wave design criteria," *Journal of Offshore Mechanics and Arctic Engineering*, vol. 130, 041604:1–041604:8, 2008.