Synthetic Bedrock Generation

> **IMPORTANT POINTS**
>
> Using these beds in ISSM, requires the ice model to handle the periodic boundaries... need to figure out a way to match the edges.

To generate informed artificial beds for ISSM (Ice Sheet System Model) runs, the `synthetic_bed.py` script uses a **Spectral Synthesis (Inverse FFT)** approach. The idea behind the synthetic generator is based on "**recipes**" that encompass roughness statistics and anisotropy and statistical signatures for each analysed region. The parameters and recipes are based on the findings reported in Bedmap3 observational data analysis .

Method

The statistical analysis (Bedmap3 observational data analysis) used the Lomb-Scargle periodogram to fit a power law. However, this script generates synthetic terrain by creating noise that adheres to this power law using the function `generate_anisotropic_bed()` , this function uses instead a standard FFT, since the synthetic data is generated on a regular grid.

Initialising the Domain

The script defines a frequency space grid with a user adjustable resolution (in meters). Then it creates an array of random Gaussian noise (mean 0, std 1). This represents a "flat" spectrum (**a white noise canvas**).

Incorporating Anisotropy

Bedrock roughness is **anisotropic**—it looks different parallel to flow vs. perpendicular to flow. This naturally generates elongated, streamlined bedforms (drumlinoids/mega-scale glacial lineations) **aligned with the flow**, and bumpy ribbing and cross-channels **perpendicular to flow*[1]*.

The script uses a **Separable Spectral Filter** In order to mimic a real-world aerial flight line via a **marginal power spectrum**. The code validates this by averaging the 2D power spectrum along one axis to see if the 1D profile still matches the target power law exponent (β) [2] .

```
H(kx, ky) = 1 / (|kx|^(β_x/2) × |ky|^(β_y/2))
```

This guarantees that the X (flow) and Y (cross-flow) directions maintain independent statistical signatures

$$PSD(k_x, k_y) = |H|^2 \propto |k_x|^{-\beta_x} \times |k_y|^{-\beta_y}$$

The marginal spectrum is:

$$PSD(k_x) = \int PSD(k_x, k_y), dk_y \propto |k_x|^{-\beta_x}$$

The DC component and axis frequencies are set to zero to avoid division-by-zero singularities and to ensure the resulting surface has zero mean.

```
scaling_factor[0, 0] = 0
scaling_factor[0, :] = 0
scaling_factor[:, 0] = 0
```

Inverse Transform

Apply the Inverse FFT (IFFT) to convert the filtered spectrum back into the spatial domain (elevation).

Injecting Specific Bedforms (Dominant Wavelengths)

To move from only noise to a geologically informed bed that includes specific features like drumlins, ribbing, or large basin undulations, the script uses a function `inject_bedforms()` that takes the following configuration arguments to match the corresponding bedform recipes:

	Resolution	Wavelength `wavelength_m`	Amplitude `amplitude_m`	Orientation `angle_deg` $\rightarrow$ `angle_rad` = θ
Short Bedforms (e.g., ASB/PEL)	**150 m**	**300m - 800m**	**5m - 20m**	**0° (Flow parallel)**
Basin Undulations (e.g., ROSS)	**500 m**	**10km - 50km**	**50m - 100m**	**Variable**

The script uses a coordinate rotation to align bedrock features via a **2D oriented wave** that can point in any direction (flow-parallel or transverse).

$$\Phi = \frac{2\pi}{\lambda}(x_{\text{grid}} \cos \theta + y_{\text{grid}} \sin \theta),$$

$$Z_{\text{bed}} = A \sin \Phi$$

Normalise to target relief by Slope-Limited Amplitude Scaling.

The code uses 1st–99th percentile for relief calculation, which is important for robustness:

```
p_min, p_max = np.percentile(bed, [1, 99])
current_relief = p_max - p_min
```

Ice sheet models like ISSM require high resolution where slopes are steep to maintain stability.
The script does a binary search—once after the texture/anisotropy layer and a second time after the bedform injection—for a maximum acceptable scale of the vertical landscape until the steepest point satisfies the `max_slope_deg` limit.
The convergence criterion is

```
(abs(test_slope - max_slope_deg) < 0.5)
```

This approach keeps the **spectral signature (the "roughness")** perfectly intact and sacrificing **Vertical Relief** if needed.

Validation Check

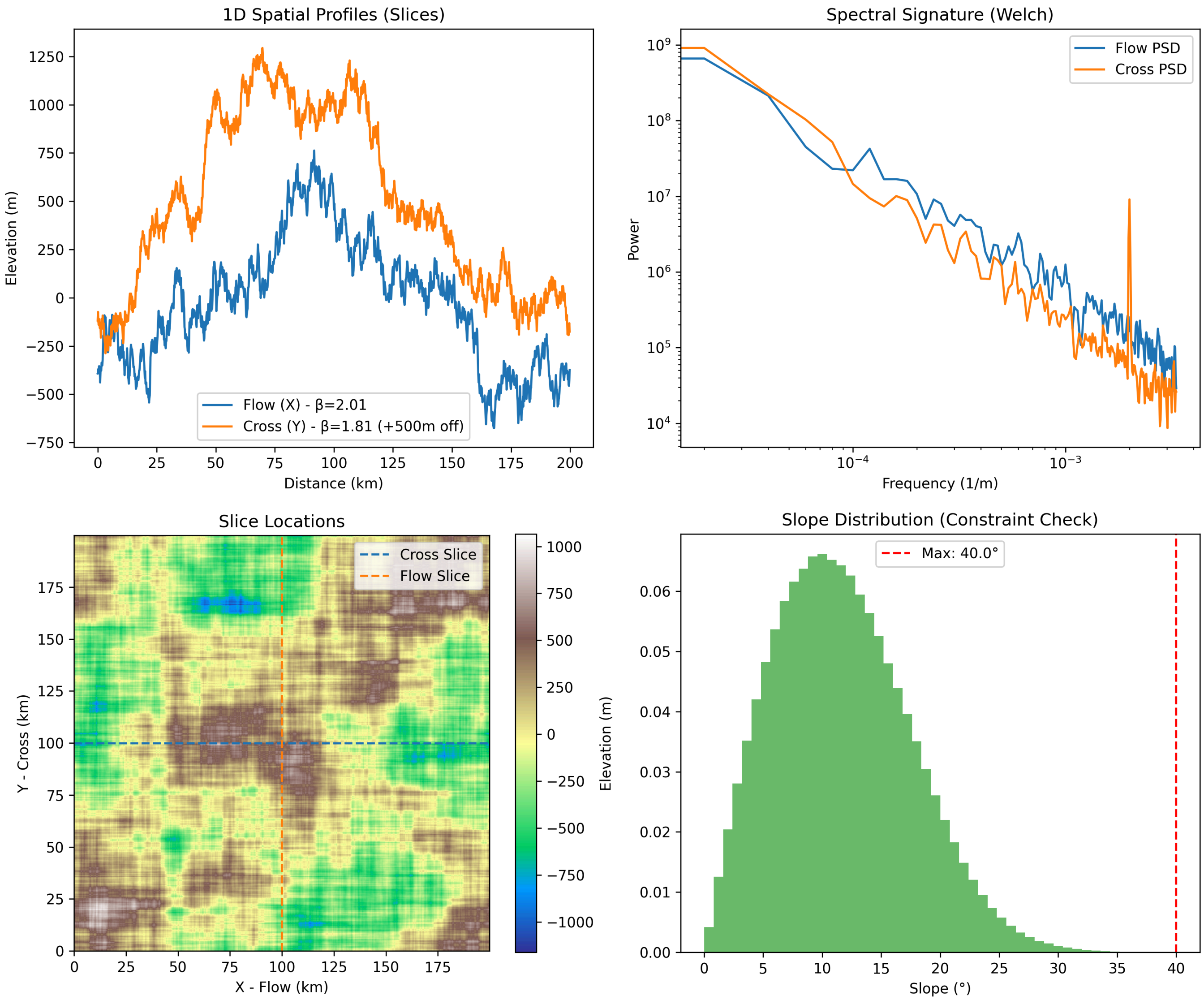
Following the logic of the Bedmap3 analysis, the `validate_anisotropy()` routine verifies that the generated bed has adequate texture values in two ways

1. It uses a **Standard FFT** for the `compute_marginal_beta` validation which measures the power β from a power spectrum given a user specified direction (this is a global average over the entire 2D FFT).
2. It extracts x and y profiles and calculates the spectral signature
3. Signature of a single profile using **Welch's method** for estimating the power spectral density (PSD).
   Welch's method [1](https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.welch.html#r34b375daf612-1) computes an estimate of the power spectral density by dividing the data into overlapping segments, computing a modified periodogram for each segment and averaging the periodograms.

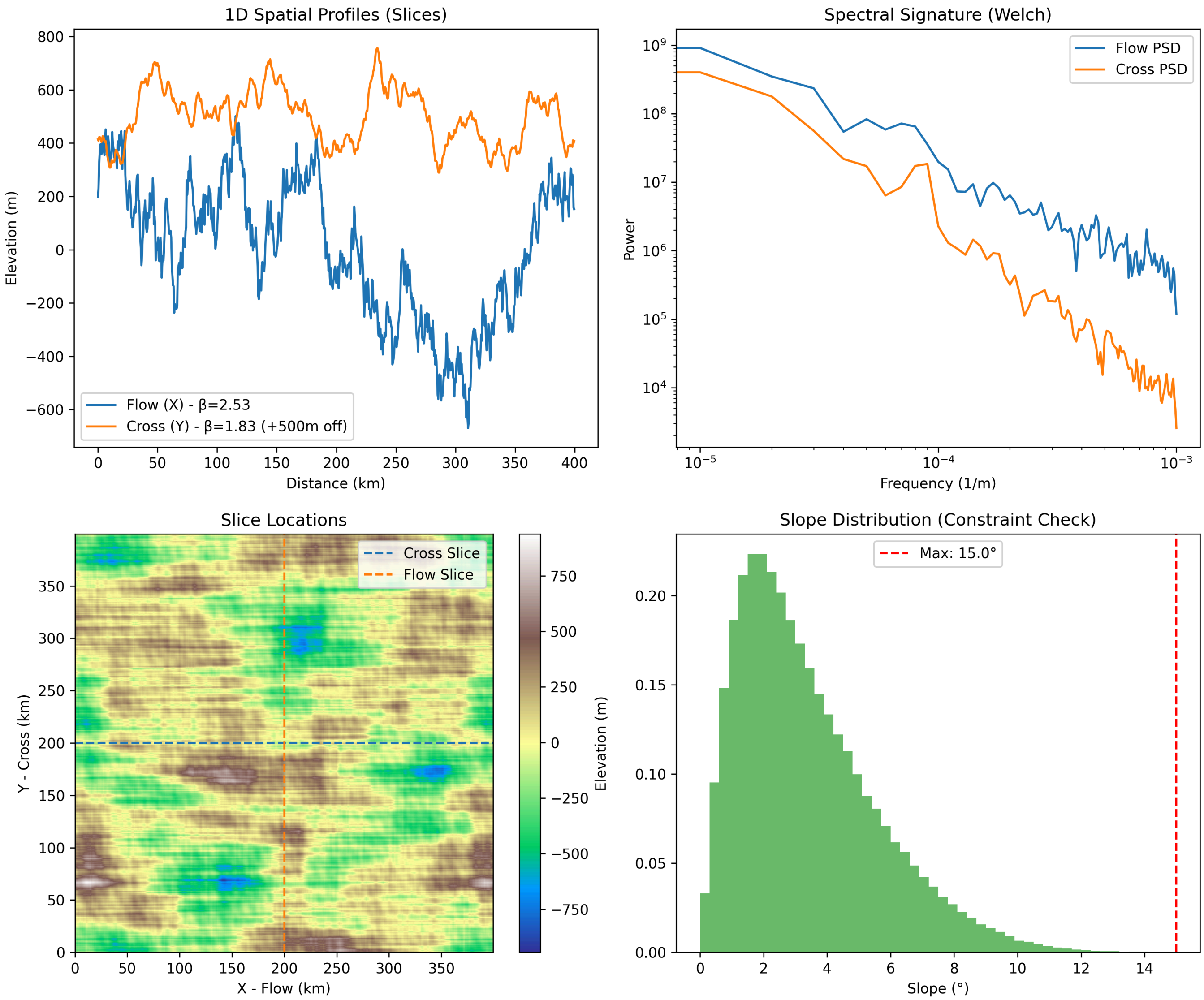
The validation function also utilises a log-log linear regression fit to asses the quality of the anisotropy check (R^2)

Results

PEL : HARD BED



ROSS: SOFT BED



Plot Interpretations

Ax1: Single dimension elevation profiles (for x (parallel to flow) in blue and y (cross to flow) orange)

Ax2: **Spectral signature: (PSD) via Welch's method**:
The frequency increases to the right. The x-axis runs from roughly 10^{-5} to 10^{-3} 1/m (i.e. ~100s of km to ~1km features), right side of the plot implies a higher spatial frequency, which corresponds to shorter wavelengths / smaller-scale features. The power decreases with increase in frequency for both regions, which is as expected. The slope of these fits is the spectral exponent β--- a measure of the bedrock roughness—the power separation between the plots represents the magnitude of the anisotropy of the bed. The direction parallel to the flow (blue) has higher power for both **PEL** and **ROSS** regions, as expected.
There's a noticeable feature in the **ROSS** Cross PSD (orange plot) around the 10^{-4} frequency. This could suggest a characteristic scale in perpendiculat flow direction (see **Ax3** comments below).

Ax3: Digital elevation model (DEM) of the generated bed. The flight lines in the flow and cross flow directions are indicated with a dash line.
PEL region: we see a high mountain terrain.
ROSS region: the elongated flow-parallel features (the brown ridges) that appear to have some quasi-periodic spacing (around 50 -100 km) in the cross flow direction. That could be mega-scale glacial lineations or similar flow-aligned bedforms.

Ax4: Cumulative distribution of all slopes in the DEM, all slopes are below the slope threshold for each region.

`Terminal` output

```
============================================================
GENERATING: ROSS
Resolution: 500m | Max allowed slope: 15.0°
Target β: flow=2.5, cross=1.8
```

```
==================================================

Base bed generated:
  Final Relief: 1100m (100% of target)
  Max slope: 15.0°

Injecting 2 bedform layers...
Bedforms were scaled to 37% to satisfy 15.0° limit.

=== ANISOTROPY VALIDATION ===
X (Flow):  β = 2.53 (target 2.5, R²=0.96)
Y (Cross): β = 1.83 (target 1.8, R²=0.91)

*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~

==================================================
GENERATING: PEL
Resolution: 150m | Max allowed slope: 40.0°
Target β: flow=2.0, cross=1.8
==================================================

Base bed generated:
  Final Relief: 1375m (69% of target)
  Max slope: 38.9°
Relief was reduced to satisfy slope constraint

Injecting 1 bedform layers...
Bedforms were scaled to 83% to satisfy 40.0° limit.

=== ANISOTROPY VALIDATION ===
X (Flow):  β = 2.01 (target 2.0, R²=0.96)
Y (Cross): β = 1.81 (target 1.8, R²=0.94)

*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~
```

Footnotes

---

1. By design the terrain has a lower β value perpendicular to the flow direction (Cross Flow). ↩
2. The Fast Fourier Transform (FFT) works with amplitudes not power, but since $\text{Power} \propto \text{Amplitude}^2$, the amplitude filter is applied as the square root of the power filter. ↩