# skfda.preprocessing.registration.invert_warping

**skfda.preprocessing.registration.invert_warping**(*fdatagrid, *, eval_points=None*) [source]

Compute the inverse of a diffeomorphism.

Let $\gamma : [a, b] \to [a, b]$ be a function strictly increasing, calculates the corresponding inverse $\gamma^{-1} : [a, b] \to [a, b]$ such that $\gamma^{-1} \circ \gamma = \gamma \circ \gamma^{-1} = \gamma_{id}$.

Uses a PCHIP interpolator to compute approximately the inverse.

**Parameters:**

- **fdatagrid** (`FDataGrid`) – Functions to be inverted.
- **eval_points** – (array_like, optional): Set of points where the functions are interpolated to obtain the inverse, by default uses the sample points of the fdatagrid.

**Returns:** Inverse of the original functions.

**Return type:** `FDataGrid`

**Raises:** `ValueError` – If the functions are not strictly increasing or are multidimensional.

## Examples

```
>>> import numpy as np
>>> from skfda import FDataGrid
>>> from skfda.preprocessing.registration import invert_warping
```

We will construct the warping $\gamma : [0, 1] rightarrow[0, 1]$ wich maps t to t^3.

```
>>> t = np.linspace(0, 1)
>>> gamma = FDataGrid(t**3, t)
>>> gamma
FDataGrid(...)
```

We will compute the inverse.

```
>>> inverse = invert_warping(gamma)
>>> inverse
FDataGrid(...)
```

The result of the composition should be approximately the identity function .

```
>>> identity = gamma.compose(inverse)
>>> identity([0, 0.25, 0.5, 0.75, 1]).round(3)
array([[ 0.  ,  0.25,  0.5 ,  0.75,  1.  ]])
```