Package: CellODE (via r-universe)

May 26, 2026

Type Package

Title Cellular Dynamics Inference Using Neural ODE

Version 1.0.0

Description An R implementation for single-cell trajectory inference using Variational Autoencoder (VAE) and Neural Ordinary Differential Equations (Neural ODE). CellODE automatically infers cellular dynamics from single-cell RNA sequencing data, providing pseudotime estimation, latent space representation, and vector field analysis. The package is designed for seamless integration with Seurat objects and supports both Seurat V4 and V5.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports torch (>= 0.9.0), R6, Matrix, Rcpp, RcppArmadillo, ggplot2, coro, methods

Suggests Seurat (>= 4.0.0), testthat (>= 3.0.0), knitr, rmarkdown, pkgdown

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

URL https://zaoqu-liu.github.io/CellODE/, https://github.com/Zaoqu-Liu/CellODE

BugReports https://github.com/Zaoqu-Liu/CellODE/issues

Config/testthat/edition 3

Repository https://zaoqu-liu.r-universe.dev

Date/Publication 2026-01-25 18:08:27 UTC

RemoteUrl https://github.com/Zaoqu-Liu/CellODE

RemoteRef main

RemoteSha d163c8cf8510a34abb4af5e2b783163b9387ad3d

Contents

`CellODE-package` *CellODE: Cellular Dynamics Inference Using Neural ODE*

Description

CellODE is an R package for single-cell trajectory inference using Variational Autoencoder (VAE) and Neural Ordinary Differential Equations (Neural ODE). The package automatically infers cellular dynamics from single-cell RNA sequencing data, providing:

- Pseudotime estimation

- Latent space representation
- Vector field analysis
- Trajectory visualization

Details

CellODE is designed for seamless integration with Seurat objects (supporting both V4 and V5). The core model (TNODE) combines a VAE for dimensionality reduction with a Neural ODE for modeling continuous cellular dynamics.

The main workflow involves:

1. Creating a Trainer object with your Seurat data
2. Training the model using `trainer$train()`
3. Extracting pseudotime using `trainer$get_time()`
4. Getting latent representations using `trainer$get_latentsp()`
5. Visualizing results with `plot_pseudotime()` and `plot_vector_field()`

Author(s)

Zaoqu Liu <liuzaoqu@163.com>

References

Li, S. et al. (2023). scTour: A deep learning architecture for robust inference and accurate prediction of cellular dynamics. bioRxiv. https://doi.org/10.1101/2023.01.13.523988

See Also

Useful links:

- https://zaoqu-liu.github.io/CellODE/
- https://github.com/Zaoqu-Liu/CellODE
- Report bugs at https://github.com/Zaoqu-Liu/CellODE/issues

`cosine_similarity` *Calculate Cosine Similarity*

Description

Calculate cosine similarity between vector field and cell-neighbor latent state difference. This function matches scTour's cosine_similarity exactly. The calculation borrows the ideas from scvelo.

Uses optimized C++ implementation for performance on large datasets.

Usage

```
cosine_similarity(
  zs,
  vf,
  reverse = FALSE,
  n_neigh = 20,
  t = NULL,
  var_transform = FALSE,
  neighbor_indices = NULL
)
```

Arguments

zs	Latent space matrix (cells x latent_dim)
vf	Vector field matrix (cells x latent_dim)
reverse	Whether to reverse vector field direction (default: FALSE)
n_neigh	Number of neighbors (default: 20)
t	Time vector for time-aware neighbors (default: NULL)
var_transform	Variance-stabilizing transformation (default: FALSE)
neighbor_indices	Pre-computed neighbor indices matrix (optional)

Value

Sparse matrix of cosine similarities

---

data_utils *Data Utilities for CellODE*

Description

Functions for data preparation and handling

---

extract_expression *Extract Expression Matrix from Seurat Object*

Description

Extract expression matrix from Seurat object, supporting both V4 and V5.

Usage

```
extract_expression(seurat_obj, assay = "RNA", slot = "counts")
```

Arguments

seurat_obj	Seurat object
assay	Assay name (default: "RNA")
slot	Slot/layer name (default: "counts")

Value

Dense matrix (cells x genes)

---

l2_norm *L2 Norm*

Description

Calculate L2 norm along specified axis. Handles both dense and sparse matrices.

Usage

```
l2_norm(x, axis = -1)
```

Arguments

x	Input matrix
axis	Axis along which to compute norm (default: -1 for last axis)

Value

Vector of L2 norms

---

load_model *Load Trained Model*

Description

Load a trained CellODE model from file.

Usage

```
load_model(path, seurat_obj)
```

Arguments

path	Path to saved model (without extension)
seurat_obj	Seurat object for training data

Value

Trainer object with loaded model

log_nb *Log Probability of Negative Binomial Distribution*

Description

Calculate log probability under negative binomial distribution. Adapted from scvi-tools.

Usage

```
log_nb(x, mu, theta, eps = 1e-08)
```

Arguments

x	Observed counts
mu	Mean parameter
theta	Dispersion parameter
eps	Small constant for numerical stability

Value

Tensor of log probabilities

References

Gayoso et al. scvi-tools. https://github.com/YosefLab/scvi-tools

log_zinb *Log Probability of Zero-Inflated Negative Binomial Distribution*

Description

Calculate log probability under zero-inflated negative binomial. Adapted from scvi-tools.

Usage

```
log_zinb(x, mu, theta, pi, eps = 1e-08)
```

Arguments

x	Observed counts
mu	Mean parameter
theta	Dispersion parameter
pi	Dropout probability (logit scale)
eps	Small constant for numerical stability

Value

Tensor of log probabilities

References

Gayoso et al. scvi-tools. https://github.com/YosefLab/scvi-tools

---

MakeDataset *Create Dataset for Training*

Description

Create a torch dataset from expression matrix.

Usage

```
MakeDataset(X, loss_mode = "nb")
```

Arguments

X	Expression matrix (cells x genes)
loss_mode	Loss mode: "mse", "nb", or "zinb"

Value

torch::dataset object

---

nn_modules *Neural Network Modules for CellODE*

Description

VAE components and Latent ODE function using torch nn_module

normal_kl *KL Divergence between Two Normal Distributions*

Description

Calculate KL divergence between two normal distributions. This is the standard formulation from torchdiffeq.

Usage

```
normal_kl(mu1, lv1, mu2, lv2)
```

Arguments

mu1	Mean of first distribution (posterior)
lv1	Log variance of first distribution
mu2	Mean of second distribution (prior)
lv2	Log variance of second distribution

Value

Tensor of KL divergence values

References

RTQichen. torchdiffeq. https://github.com/rtqichen/torchdiffeq

ode_solver *Neural ODE Solver for CellODE*

Description

ODE solvers with support for gradient computation

odeint *Forward ODE Integration*

Description

Solve ODE forward in time. This is the main integration function that matches scTour's odeint behavior.

Usage

```
odeint(func, z0, t, method = "euler", options = list())
```

Arguments

func	ODE function (nn_module with forward(t, z) method)
z0	Initial state tensor
t	Time points tensor (must be sorted)
method	Integration method: "euler" (default, matches scTour) or "rk4"
options	List with optional step_size

Value

Tensor of states at each time point (n_times x n_latent)

Examples

```
## Not run:
states <- odeint(ode_func, z0, t, method = "euler")

## End(Not run)
```

plot_loss *Plot Training Loss Curves*

Description

Visualize training and validation loss curves from CellODE training.

Usage

```
plot_loss(
  trainer,
  smooth = FALSE,
  title = "Training Progress",
  colors = c(Training = "#2E86AB", Validation = "#E94F37")
)
```

Arguments

trainer	Trainer object with training log
smooth	Apply loess smoothing (default: FALSE)
title	Plot title (default: "Training Progress")
colors	Colors for train/validation lines

Value

ggplot2 object

Examples

```
## Not run:
plot_loss(trainer)

## End(Not run)
```

---

plot_pseudotime *Plot Pseudotime with Direction Arrows*

Description

Visualize pseudotime trajectory with direction arrows computed using grid-based vector field algorithm. Colors cells by pseudotime gradient and overlays arrows showing developmental direction.

Usage

```
plot_pseudotime(
  seurat_obj,
  time_key = "pseudotime",
  embedding_key = "umap",
  dims = c(1, 2),
  point_size = 0.8,
  point_palette = "Spectral",
  point_palcolor = NULL,
  point_alpha = 0.8,
  arrow_show = TRUE,
  arrow_density = 1.5,
  arrow_length = 0.12,
  arrow_width = 0.6,
  arrow_color = "grey20",
  arrow_angle = 25,
  arrow_type = c("open", "closed"),
  show_axes = TRUE,
  aspect_ratio = 1,
  title = "Pseudotime",
```

```
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  legend_position = "right",
  return_layer = FALSE,
  seed = 11
)
```

Arguments

seurat_obj	Seurat object with dimensionality reduction and pseudotime
time_key	Column name in metadata containing pseudotime (default: "pseudotime")
embedding_key	Embedding to use (default: "umap")
dims	Dimensions to plot (default: c(1, 2))
point_size	Point size (default: 0.8)
point_palette	Color palette (default: "Spectral")
point_palcolor	Custom color vector (overrides palette)
point_alpha	Point transparency (default: 0.8)
arrow_show	Show direction arrows (default: TRUE)
arrow_density	Arrow density (default: 1.5)
arrow_length	Arrow length multiplier (default: 0.12)
arrow_width	Arrow line width (default: 0.6)
arrow_color	Arrow color (default: "grey20")
arrow_angle	Arrow head angle (default: 25)
arrow_type	Arrow type: "open" or "closed" (default: "open")
show_axes	Show axes (default: TRUE)
aspect_ratio	Aspect ratio (default: 1)
title	Plot title (default: "Pseudotime")
subtitle	Plot subtitle (default: NULL)
xlab	X-axis label (default: auto)
ylab	Y-axis label (default: auto)
legend_position	Legend position (default: "right")
return_layer	Return layers for custom assembly (default: FALSE)
seed	Random seed (default: 11)

Value

ggplot2 object or list of layers

Examples

```
## Not run:
# Basic pseudotime visualization
plot_pseudotime(seurat_obj, time_key = "pseudotime")

# Custom styling
plot_pseudotime(seurat_obj, point_palette = "viridis",
                arrow_color = "darkred", arrow_density = 0.5)

## End(Not run)
```

---

plot_vector_field *Plot Vector Field with Direction Arrows*

Description

Visualize cellular dynamics vector field on 2D embedding using distance-weighted directional arrows. Computes direction vectors at grid centers by combining distance-based weights and pseudotime gradients from the latent space dynamics. Supports three rendering modes: raw (per-cell arrows), grid (smoothed arrows), and stream (streamlines for flow visualization).

Usage

```
plot_vector_field(
  seurat_obj,
  zs_key = "X_zs",
  vf_key = "X_VF",
  embedding_key = "umap",
  t_key = NULL,
  plot_type = c("grid", "raw", "stream"),
  reverse = FALSE,
  n_neigh = 20,
  var_transform = FALSE,
  scale = 10,
  self_transition = FALSE,
  point_size = 0.5,
  point_palette = "Spectral",
  point_palcolor = NULL,
  point_alpha = 0.8,
  arrow_density = 0.5,
  arrow_length = 0.15,
  arrow_width = 0.6,
  arrow_color = "grey20",
  arrow_angle = 25,
  arrow_head_size = 50,
  arrow_type = c("open", "closed"),
  smooth = 0.5,
```

```
  grid_density = 1,
  min_mass = 1,
  stream_n = 15,
  stream_L = 5,
  stream_color = NULL,
  stream_width = c(0.2, 0.8),
  color_by = NULL,
  show_axes = TRUE,
  aspect_ratio = 1,
  title = "Vector Field",
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  legend_position = "right",
  return_layer = FALSE,
  seed = 11
)
```

Arguments

seurat_obj	Seurat object containing embeddings and CellODE results
zs_key	Key for latent space in misc or reductions (default: "X_zs")
vf_key	Key for vector field in misc (default: "X_VF")
embedding_key	Embedding for visualization (default: "umap")
t_key	Key in metadata for pseudotime (default: NULL)
plot_type	Rendering mode: "raw", "grid", or "stream" (default: "grid")
reverse	Reverse vector field direction (default: FALSE)
n_neigh	Number of neighbors for similarity calculation (default: 20)
var_transform	Variance-stabilizing transformation (default: FALSE)
scale	Scale factor for cosine similarity (default: 10)
self_transition	Include self-transition (default: FALSE)
point_size	Size of background points (default: 0.5)
point_palette	Color palette for points (default: "Spectral")
point_palcolor	Custom color vector (overrides palette)
point_alpha	Point transparency (default: 0.8)
arrow_density	Proportion of arrows to display (default: 0.5)
arrow_length	Arrow length multiplier (default: 0.15)
arrow_width	Arrow line width (default: 0.6)
arrow_color	Arrow color (default: "grey20")
arrow_angle	Arrow head angle in degrees (default: 25)
arrow_head_size	Arrow head size scaling (default: 50)

arrow_type	Arrow head type: "open" or "closed" (default: "open")
smooth	Smoothing factor for grid (default: 0.5)
grid_density	Grid density factor (default: 1.0)
min_mass	Minimum mass threshold for grid points (default: 1)
stream_n	Number of streamlines (default: 15)
stream_L	Streamline length parameter (default: 5)
stream_color	Streamline color (default: NULL, uses gradient)
stream_width	Streamline width range (default: c(0.2, 0.8))
color_by	Variable to color points by (default: NULL)
show_axes	Show axes and labels (default: TRUE)
aspect_ratio	Plot aspect ratio (default: 1)
title	Plot title (default: "Vector Field")
subtitle	Plot subtitle (default: NULL)
xlab	X-axis label (default: auto)
ylab	Y-axis label (default: auto)
legend_position	Legend position (default: "right")
return_layer	Return layers for custom assembly (default: FALSE)
seed	Random seed (default: 11)

Value

ggplot2 object or list of layers if return_layer = TRUE

Examples

```
## Not run:
# Basic usage after CellODE training
plot_vector_field(seurat_obj, plot_type = "grid")

# Raw mode with custom colors
plot_vector_field(seurat_obj, plot_type = "raw",
                  arrow_color = "darkred", arrow_density = 0.3)

# Color points by pseudotime
plot_vector_field(seurat_obj, color_by = "pseudotime",
                  point_palette = "viridis")

## End(Not run)
```

plotting *Visualization Functions for CellODE*

Description

High-quality visualization for pseudotime and vector field

Author(s)

Zaoqu Liu

predict *Prediction Functions for CellODE*

Description

Functions for predicting pseudotime, latent space, and vector field

predict_latentsp *Predict Latent Space for Query Cells*

Description

Predict latent representations for query cells.

Usage

```
predict_latentsp(
  trainer,
  query_seurat,
  mode = "fine",
  alpha_z = 0.5,
  alpha_predz = 0.5,
  step_size = NULL,
  step_wise = FALSE,
  batch_size = NULL,
  assay = "RNA"
)
```

Arguments

trainer	Trained Trainer object
query_seurat	Query Seurat object
mode	Prediction mode: "fine" or "coarse" (default: "fine")
alpha_z	Weight for encoder-derived latent (default: 0.5)
alpha_predz	Weight for ODE-derived latent (default: 0.5)
step_size	Step size for integration (default: NULL)
step_wise	Use step-wise integration (default: FALSE)
batch_size	Batch size (default: NULL)
assay	Assay name (default: "RNA")

Value

List with mix_zs, zs, pred_zs matrices

---

predict_ltsp_from_time

Predict Latent Space from Time Points

Description

Predict latent space for query (unobserved) time intervals. Matches scTour's predict_ltsp_from_time function exactly.

Usage

```
predict_ltsp_from_time(
  trainer,
  t,
  reverse = FALSE,
  step_wise = TRUE,
  step_size = NULL,
  alpha_z = 0.5,
  alpha_predz = 0.5,
  k = 20,
  assay = "RNA"
)
```

Arguments

trainer	Trained Trainer object
t	Vector of query time points (values between 0 and 1)
reverse	Whether pseudotime was reversed (default: FALSE)

step_wise	Use step-wise integration (default: TRUE)
step_size	Step size for integration (default: NULL)
alpha_z	Weight for encoder-derived latent (default: 0.5)
alpha_predz	Weight for ODE-derived latent (default: 0.5)
k	Number of nearest neighbors in time space (default: 20)
assay	Assay name (default: "RNA")

Value

Matrix of predicted latent space

`predict_time` *Predict Pseudotime for Query Cells*

Description

Predict developmental pseudotime for query cells.

Usage

```
predict_time(trainer, query_seurat, reverse = FALSE, assay = "RNA")
```

Arguments

trainer	Trained Trainer object
query_seurat	Query Seurat object
reverse	Whether to reverse pseudotime (default: FALSE)
assay	Assay name (default: "RNA")

Value

Numeric vector of pseudotime values

predict_vector_field *Predict Vector Field for Query Cells*

Description

Predict vector field for query cells.

Usage

```
predict_vector_field(trainer, t, z)
```

Arguments

trainer	Trained Trainer object
t	Pseudotime vector
z	Latent space matrix

Value

Matrix of vector field

reverse_time *Reverse Pseudotime*

Description

Post-inference adjustment to reverse the pseudotime.

Usage

```
reverse_time(t)
```

Arguments

t	Pseudotime vector

Value

Reversed pseudotime (1 - t)

split_data *Split Data for Training and Validation*

Description

Split dataset into training and validation sets.

Usage

```
split_data(X, percent, val_frac = 0.1, loss_mode = "nb")
```

Arguments

X	Expression matrix (cells x genes)
percent	Percentage of cells for training
val_frac	Validation fraction from training set
loss_mode	Loss mode for data transformation

Value

List with train_data and val_data

split_index *Split Indices for Training and Validation*

Description

Split indices into training and validation sets.

Usage

```
split_index(n_cells, percent, val_frac = 0.1)
```

Arguments

n_cells	Total number of cells
percent	Percentage for training
val_frac	Validation fraction

Value

List with train_idx and val_idx

TNODE *Time Neural ODE (TNODE) Model*

Description

Complete model combining VAE and Neural ODE for cellular dynamics inference. This is the core model of CellODE.

Usage

```
TNODE(
  n_int,
  n_latent = 5L,
  n_ode_hidden = 25L,
  n_vae_hidden = 128L,
  batch_norm = FALSE,
  ode_method = "euler",
  step_size = NULL,
  alpha_recon_lec = 0.5,
  alpha_recon_lode = 0.5,
  alpha_kl = 1,
  loss_mode = "nb"
)
```

Arguments

n_int	Number of input features (genes)
n_latent	Dimensionality of latent space (default: 5)
n_ode_hidden	Hidden layer size for ODE function (default: 25)
n_vae_hidden	Hidden layer size for VAE (default: 128)
batch_norm	Whether to include BatchNorm layer (default: FALSE)
ode_method	ODE solver method (default: "euler")
step_size	Step size multiplier for integration (NULL for default)
alpha_recon_lec	Weight for encoder reconstruction loss (default: 0.5)
alpha_recon_lode	Weight for ODE reconstruction loss (default: 0.5)
alpha_kl	Weight for KL divergence (default: 1.0)
loss_mode	Loss function: "mse", "nb", or "zinb" (default: "nb")

Value

nn_module for TNODE model

train *Training Module for CellODE*

Description

Trainer class for model training and inference

Trainer *CellODE Trainer*

Description

R6 class for implementing the CellODE training process.

Public fields

`seurat_obj` Seurat object for training

`model` TNODE model

`optimizer` Adam optimizer

`device` Computation device

`log` Training log

`time_reverse` Whether to reverse time

`model_kwargs` Model configuration

Methods

Public methods:

- `Trainer$new()`
- `Trainer$train()`
- `Trainer$get_time()`
- `Trainer$get_vector_field()`
- `Trainer$get_latentsp()`
- `Trainer$save_model()`
- `Trainer$load_model()`
- `Trainer$clone()`

Method `new()`**:** Initialize Trainer

Usage:

```
Trainer$new(
  seurat_obj,
  assay = "RNA",
  slot = NULL,
  percent = NULL,
  n_latent = 5L,
  n_ode_hidden = 25L,
  n_vae_hidden = 128L,
  batch_norm = FALSE,
  ode_method = "euler",
  step_size = NULL,
  alpha_recon_lec = 0.5,
  alpha_recon_lode = 0.5,
  alpha_kl = 1,
  loss_mode = "nb",
  nepoch = NULL,
  batch_size = 1024L,
  drop_last = FALSE,
  lr = 0.001,
  wt_decay = 1e-06,
  eps = 0.01,
  random_state = 0L,
  val_frac = 0.1,
  use_gpu = TRUE
)
```

Arguments:

`seurat_obj` Seurat object with expression data

`assay` Assay to use (default: "RNA")

`slot` Slot to use (default: "counts" for nb/zinb, "data" for mse)

`percent` Percentage of cells for training (default: auto)

`n_latent` Latent space dimensions (default: 5)

`n_ode_hidden` ODE hidden layer size (default: 25)

`n_vae_hidden` VAE hidden layer size (default: 128)

`batch_norm` Use batch normalization (default: FALSE)

`ode_method` ODE solver (default: "euler")

`step_size` Step size multiplier (default: NULL)

`alpha_recon_lec` Encoder reconstruction weight (default: 0.5)

`alpha_recon_lode` ODE reconstruction weight (default: 0.5)

`alpha_kl` KL divergence weight (default: 1.0)

`loss_mode` Loss mode: "mse", "nb", "zinb" (default: "nb")

`nepoch` Number of epochs (default: auto)

`batch_size` Batch size (default: 1024)

`drop_last` Drop last incomplete batch (default: FALSE)

`lr` Learning rate (default: 1e-3)

`wt_decay` Weight decay (default: 1e-6)

`eps` Adam epsilon (default: 0.01)
`random_state` Random seed (default: 0)
`val_frac` Validation fraction (default: 0.1)
`use_gpu` Use GPU if available (default: TRUE)

Method `train()`**:** Train the model

Usage:

```
Trainer$train()
```

Method `get_time()`**:** Get pseudotime for all cells

Usage:

```
Trainer$get_time()
```

Returns: Numeric vector of pseudotime values

Method `get_vector_field()`**:** Get vector field

Usage:

```
Trainer$get_vector_field(t, z)
```

Arguments:

`t` Pseudotime vector
`z` Latent space matrix

Returns: Matrix of vector field

Method `get_latentsp()`**:** Get latent space representation

Usage:

```
Trainer$get_latentsp(
  alpha_z = 0.5,
  alpha_predz = 0.5,
  step_size = NULL,
  step_wise = FALSE,
  batch_size = NULL
)
```

Arguments:

`alpha_z` Weight for encoder-derived latent (default: 0.5)
`alpha_predz` Weight for ODE-derived latent (default: 0.5)
`step_size` Step size for integration (default: NULL)
`step_wise` Use step-wise integration (default: FALSE)
`batch_size` Batch size (default: NULL for all)

Returns: List with mix_zs, zs, pred_zs matrices

Method `save_model()`**:** Save trained model

Usage:

```
Trainer$save_model(path)
```

Arguments:

`path` File path (without extension)

Method `load_model()`**:** Load trained model

Usage:

```
Trainer$load_model(path)
```

Arguments:

`path` File path (without extension)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Trainer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`utils` *Utility Functions for CellODE*

Description

Mathematical utilities for loss computation and ODE solving

`vector_field` *Vector Field Analysis for CellODE*

Description

Functions for vector field computation and visualization

vector_field_embedding *Vector Field Embedding*

Description

Calculate weighted unitary displacement vectors under embedding. This function matches scTour's vector_field_embedding exactly. The calculation borrows the ideas from scvelo.

Usage

```
vector_field_embedding(T_mat, E, scale = 10, self_transition = FALSE)
```

Arguments

T_mat	Cosine similarity sparse matrix (from cosine_similarity)
E	Embedding matrix (cells x 2)
scale	Scale factor for cosine similarity (default: 10)
self_transition	Include self-transition (default: FALSE)

Value

Matrix of displacement vectors

vector_field_embedding_grid *Vector Field Embedding on Grid*

Description

Estimate displacement vectors on a grid. This function matches scTour's vector_field_embedding_grid exactly. The calculation borrows the ideas from scvelo.

Usage

```
vector_field_embedding_grid(E, V, smooth = 0.5, stream = FALSE, density = 1)
```

Arguments

E	Embedding matrix (cells x 2)
V	Displacement vectors (cells x 2)
smooth	Smoothing factor for Gaussian pdf (default: 0.5)
stream	Adjust for streamplot (default: FALSE)
density	Grid density (default: 1.0)

Value

List with E_grid and V_grid

---

zzz *Package Startup*

Description

Functions called on package load/attach

Index