Efficient Filtering and Fitting of Models Derived from Integro-Difference Equations

Evan Tate Paterson Hughes

Table of contents

: : : {.content-visible unless-format = “pdf”} Index : : :

```
# Fitting IDEM using ```jax_idem```
```

The primary use of the jax_idem package is to fit Integro-difference equation models to data.

Currently, the only supported way to do this is through maximum-likelihood estimation with the kalman/information filter and OPTAX.

Chapter 1

Simple example; synthetic simple data

We will start by simulating from a simple IDEM with only three time steps. We can quickly make a model using `gen_example_idem`

```
import jax
import jax.random as rand
import jax.numpy as jnp

jax.config.update('jax_enable_x64', False)

import matplotlib.pyplot as plt

import sys
import os
sys.path.append(os.path.abspath('../src/jaxidem'))
import idem
import utilities
import filter_smoother_functions as fsf

import importlib
importlib.reload(idem)
importlib.reload(utilities)
importlib.reload(fsf)
```

```
key = jax.random.PRNGKey(1)
keys = rand.split(key, 3)

process_basis = utilities.place_basis(nres=1, min_knot_num=5)
nbasis = process_basis.nbasis

m_0 = jnp.zeros(nbasis).at[20].set(10)
sigma2_0 = 0.0001

truemodel = idem.gen_example_idem(
    keys[0], k_spat_inv=True,
    process_basis=process_basis,
```