Scaling Comparisons for the filtering algorithms

Evan Tate Paterson Hughes

Table of contents

Currently, the Kalman filter, the Information filter, the Square-root filter, and the square-root-information filter are all implemented.

We will use a simulated data set, from cosine basis functions over 100 frequencies and invariant kernel. See figure (?) for the 6 time points of the simulated process.

We then filter this data, and time the implemented filters to compute the marginal log likelihood. We run the alogrithms with a sequence of model parameters that are close to the ones used to simulate the data, with added noise to avoid the memoisation the JAX uses.

We simulate $n = 300$ observations per time point in total, and redact observations in order to show the compute times for each n between 50 and 300. These are computed in 64 bit for stability, and the results are in figure (?)

Additionally, to demonstrate the square root filters stability, we also fix n and progressively increase n. The data is simulated from a high-dimensional process basis, and then filtered using models with a lower r, increasing in square between 5^2 and 15^2. The results are shown in figure (?).

```
import pandas as pd
import plotly.express as px

# Load the datasets
df64 = pd.read_csv('data/varying_n_laptop_64.csv')
df32 = pd.read_csv('data/varying_n_laptop_32.csv')

# Select and melt relevant columns from df64
df64_melted = df64.melt(
    id_vars="n",
    value_vars=df64.columns[3:7],
    var_name="Filter",
    value_name="Time"
)
df64_melted["Precision"] = "64-bit"

# Select and melt the two columns from df32
df32_melted = df32.melt(
    id_vars="n",
    value_vars=[df32.columns[4], df32.columns[6]],
    var_name="Filter",
    value_name="Time"
)
df32_melted["Precision"] = "32-bit"

# Combine both
df_combined = pd.concat([df64_melted, df32_melted], ignore_index=True)

# Plot using Plotly Express
fig = px.line(
    df_combined,
    x="n",
    y="Time",
    color="Filter",
    line_dash="Precision",
    labels={"n": "number of observations", "Time": "Time (seconds)"},
    title="Filters Comparison with Varying n"
)

fig.write_image("figure/scale_test_n.png")
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): text/html
```

```
import jax
import jax.numpy as jnp
import pandas as pd
import plotly.express as px

df = pd.read_csv('data/varying_n_64.csv')

fig = px.line(
    df,
    x="n",
    y=df.columns[3:7],
    labels={"n": "number of observations", "value": "Time (seconds)"},
    title="Average compute time Filters with Varying n (64 bit)"
)

# Show the plot
fig.write_image("figure/scale_test_n_64.png")
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

```
import jax
import jax.numpy as jnp
import pandas as pd
import plotly.express as px

df = pd.read_csv('data/varying_r_32.csv')

fig = px.line(
    df,
    x="r",
    y=df.columns[7:],
    labels={"n": "number of observations", "value": "Time (seconds)"},
    title="Compte Time of Filters Varying r (32 bit)"
)

# Show the plot
fig.write_image("figure/scale_test_r_32.png")
fig.show()
```

```
Unable to display output for mime type(s): text/html
```