Simulate Weather-Driven Infrastructure Electricity Demand

April 3, 2024

A temperature-driven model of infrastructure electricity load is constructed (coefficients in `models` have been trained on BT group plc. data 2016-2020). Applying the model to 40+ years of historic weather observations (MERRA2 reanalysis) results in a hypothetical time series with the weather variability that would have occurred for present infrastructure experiencing historic weather observations.

Day of the week effect (higher demand on Mondays vs. Sundays etc.) is incoprorated, and any further *residual* variability is introduced via a lag-1 autoregressive component.

This python notebook is an optional demonstration of the model, and can be used to generate new stochastic outputs (by resetting the random number generator seed). This notebook does the following:

- Read the reanalysis temperature timeseries
- Generate timeseries of infrastructure electricity load, based on temperature timeseries and model coefficients
- Generate stochastic realisations of infrastructure electricity load
- Save the new demand datasets in CSV format

If you benefit from this work, please consider adding a citation to our paper **Fallon et al. (2023), https://doi.org/10.1002/MET.2158**.

1 Jupyter notebook setup and configuration

[1]:
```
region = "GB" # choose from GB / cooling (London) / heating (North Scotland)
```

[2]:
```
# Number of realisations in the stochastic model
# note: a multiple of 7 ensures an equal number of Mondays/Tuesdays/etc.
realisations_size = 560
```

Import required python libraries

[3]:
```
# Access system functions (used to import local modules)
import sys

# Maths
import numpy as np

# Stats / read CSV files
import pandas as pd
```

```
# Plotting
import matplotlib.pyplot as plt

# Hide Warning Messages
import warnings
warnings.filterwarnings('ignore')
```

For reproducibility, pass a static seed `213885` for pseudo-random number generation

[4]:
```
# optional: fix the seed for RNG (or set to None for new seeds)
seed = {"GB": 213885, "cooling": 134788, "heating": 721447}[region]
rng = np.random.default_rng(seed)
```

Import local python files from `modules` folder:

[5]:
```
# make helper functions in `./modules/` available
sys.path.append("modules")

# plotting helper functions
from plotting import apply_axis_calendar, int_percent_formatter,␣
 ↪int_degC_formatter

# First-order autoregressive model and weekday_array reshaper
from model import AR1, extend_weekday_array, CDD, HDD
```

Better default matplotlib style (adjusts fontsize, plotting colours, etc.):

[6]:
```
# optional: set the plotting style
plt.style.use('seaborn-v0_8-notebook')
```

Use vector format plots (ensures high DPI render):

[7]:
```
# optional: use svg for compatibility saving notebook in different formats
%config InlineBackend.figure_format = 'svg'
```

2 Datasets

Equation 3, Fallon et al. (2023): the deterministic model of infrastructure electricity load is calculated as follows:

$$\mathcal{D}(T) = \mathcal{D}_0 + \alpha_{\text{HDD}} \times \text{HDD}(T) + \alpha_{\text{CDD}} \times \text{CDD}(T)^2$$

HDD and CDD are the Heating Degree Day and Cooling Degree Day functions, representing an uptick in infrastructure electricity demand in response to temperatures beyond an expressed threshold. The base level of demand (in the absence of temperature-induced increases) is $\mathcal{D}_0$.

`CSV` files are read using the `pandas` library: MERRA2 temperature timeseries in `temperature/`, and model coefficients in `models/`.

2.1 Temperature time series

The MERRA-2 GB/London/North-Scotland temperature daily mean temperature (2m) timeseries, degrees Celsius.

[8]:
```
temperature_region = {"GB": "GB", "cooling": "London", "heating":␣
↪"North_Scotland"}[region]
temperature = pd.read_csv(f"temperature/merra2-{temperature_region}-Land-daily.
↪csv", index_col="time")
```

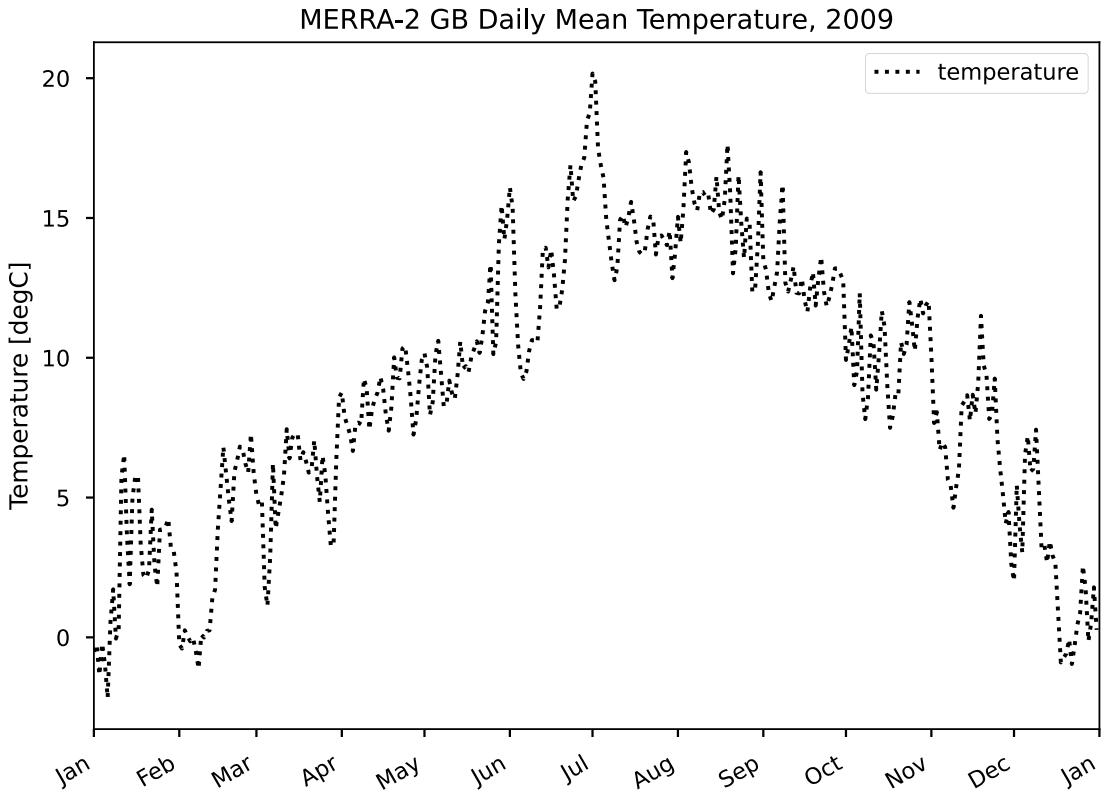
View the timeseries data:

[9]:
```
temperature
```

[9]:
```
            temperature
time
1980-01-01    -2.213271
1980-01-02    -2.846483
1980-01-03     2.220509
1980-01-04     4.068389
1980-01-05     4.160642
…                 …
2023-12-28     7.393188
2023-12-29     5.055420
2023-12-30     5.298340
2023-12-31     5.390900
2024-01-01     5.200836

[16072 rows x 1 columns]
```

Example plot of temperature timeseries spanning 2009.

[10]:
```
ax = temperature["2009":"2010"].plot(color="k", ls=":")

# Handle x-axis labels/ticks
apply_axis_calendar(ax)

ax.set_ylabel("Temperature [degC]")
ax.set_title(f"MERRA-2 {temperature_region} Daily Mean Temperature, 2009");
```



2.2 Model Coefficients

The coefficients for telecommunications infrastructure in the entire GB region, *heating* driven region, and *cooling* driven, from **Table 1, Fallon et al. (2023)**. These are stored in `models/`.

Import the model coefficients $\mathcal{D}_0$, ρ, σ, α_{HDD}, THDD, α_{CDD}, TCDD:

[11]:
```
# Read the GB (or cooling/heating) demand model coefficients
coefs = pd.read_csv(f"models/demand-coefs-{region}.csv", index_col="label").T
dd = pd.read_csv(f"models/demand-DD-{region}.csv", index_col="label").T
```

The constant offset coefficient ($\mathcal{D}_0$), and the autocorrelation coefficients (ρ and σ):

[12]:
```
coefs
```

[12]:
```
label        D0      rho     sigma
value -0.015803  0.75266  0.012481
```

The degree day coefficients (coupling coefficients α_{XDD} and threshold coefficients T_{XDD}):

[13]:
```
dd
```

[13]:
```
label            HDD       CDD
coupling    0.001541  0.001198
threshold   5.300000  9.100000
```

3 Infrastructure Electricity Load Model

3.1 Deterministic Demand (anomaly units)

Having imported the coefficients used in **Equation 3, Fallon et al., 2023** and defined the HDD and CDD functions, we may now calculate the deterministic demand timeseries:

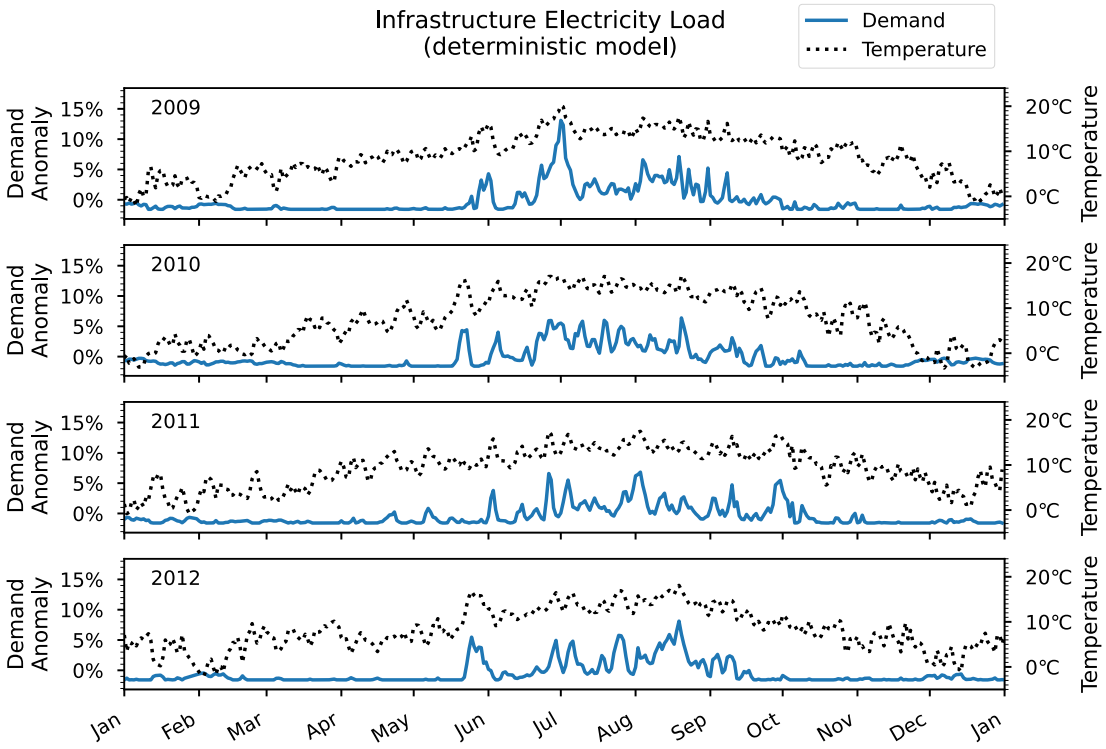
[14]:
```
demand_anom = float(coefs.D0) + dd.HDD.coupling * HDD(temperature, dd.HDD.
 ↪threshold) + dd.CDD.coupling * CDD(temperature, dd.CDD.threshold) ** 2
demand_anom.columns = ["demand (anom)"]
```

3.2 Example plot: Modelled Demand 2009-2012

Let's visualise this timeseries for a set of example years spanning 2009-2012.

[15]:
```
# Initialise the figure and axis
fig, axs = plt.subplots(4, sharex=True, sharey=True)

# Plot some example year timeseries of infrastructure electricity demand
for ax, year in zip(axs.flatten(), range(2009, 2022)):
    # Demand anomaly
    ax.text(0.03, 0.8, str(year), transform=ax.transAxes)
    ax.set_ylabel("Demand\nAnomaly")
    l1 = demand_anom[str(year):str(year+1)].plot(ax=ax, legend=False, zorder=5)
    # bug fix for legend - plot non existant line
    l2 = ax.plot(np.nan, ls=":", color="k")

    # Temperature
    ax2 = ax.twinx()
    ax2.set_ylim(-3, 23)
    ax2.set_yticks(np.arange(-5, 25, 1), minor=True)
    ax2.set_ylabel("Temperature")
    ax2.yaxis.set_major_formatter(int_degC_formatter)
    temperature[str(year):str(year+1)].plot(ax=ax2, legend=False, color="k",␣
 ↪ls=":", zorder=1)

# Handle x-axis labels/ticks
apply_axis_calendar(ax)

# Handle y-axis labels/ticks
yticks = {"GB": [0, 0.05, 0.1, 0.15], "cooling": [0, 0.1, 0.2], "heating": [-0.
 ↪05, 0, 0.05, 0.1, 0.15]}[region]
ax.set_yticks(yticks)
```

```
yticks_minor = {"GB": np.arange(-0.05, 0.2, 0.01), "cooling": np.arange(-0.1, 0.
↪3, 0.01), "heating": np.arange(-0.15, 0.2, 0.01)}[region]
ax.set_yticks(yticks_minor, minor=True)
ax.set_ylim(float(coefs.D0)*2, float(demand_anom.max()))
ax.yaxis.set_major_formatter(int_percent_formatter)

# Add a plot legend
labels = ["Demand", "Temperature"]
fig.legend([l1, l2], labels=labels, bbox_to_anchor=(0.9,1))

# Add a plot title
fig.suptitle("Infrastructure Electricity Load\n(deterministic model)");
```



Note that significant upticks in demand occur during the summer months (as high as 10-15% on some days). The demand induced in cold

4 Stochastic Model

A residual stochastic variability $\mathcal{D}^{(r)}_{\text{res}}(t)$ is subsequently introduced, with 1-day autocorrelated variability, generating realisations r of infrastructure demand:

$$\mathcal{D}^{(r)}(T(t)) = \mathcal{D}(T(t)) + \mathcal{D}^{(r)}_{\text{res}}(t)$$

This transfer function $\mathcal{D}^{(r)}(T(t))$ of reanalysis temperature data permits an assessment of weather-induced risk to reserve power systems for the period 1980 to 2020: one can simulate large upticks in demand responding to heating and cooling events, thus dimensioning the reserve capacity to the desired risk level.

We implement Equation 4, **Fallon et al. (2023)**, the first-order autoregressive term, as follows:

[16]:
```
# Daily weekday demand pattern (ie. higher energy use weekdays than weekends)
weekday_pattern = pd.read_csv(f"models/demand-weekday-{region}.csv",␣
 ↪index_col="weekday")

# The units of weekday_pattern can be changed from MW to other
# units of power, including kW, GW, etc.
# If using energy units (eg. kWh), remember to multiply results by
# the number of hours in a day (ie. x24)!
demand_unit = "MW"
```

4.1 Convert demand (anomaly units) into demand (normalised units)

See Figure 2 Fallon et al., 2023, however, we are applying the reverse process, converting anomaly units (panel e) into normalised units (panel d).

[17]:
```
# The variable `infrastructure_mean_demand_norm` is the evaluation
# of the longterm trend curve (panel d of Figure 2) at a desired
# time stamp (eg. mid timeseries, or latest date)
# This should be left as 1, unless simulating longterm trend
# changes in infrastructure electricity demand
infrastructure_mean_demand_norm = 1
```

[18]:
```
# Convert demand (anomaly units) into demand (normalised units)
demand_norm = demand_anom + infrastructure_mean_demand_norm
demand_norm.columns = ["demand (norm)"]
```

4.2 Calculation of stochastic demand (normalised units)

Residual error is the measurement error between the *observed* daily infrastructure demand and *determinsitic model* infrastructure demand.

Autocorrelation `rho` in this residual error can be calculated using the partial autocorrelation function. See Supporting Information Figure S2, **Fallon et al. (2023)**.

The standard deviation `sigma` is calculated by fitting a normal distribution to the error term, see Supporting Information Figure S2, **Fallon et al. (2023)**. Note that standard deviation is scaled to `sigma_z`, accounting for autocorrelation in the width of the distribution. See Equation 5, **Fallon et al. (2023)**.

[19]:
```
sigma_z = np.sqrt(1-coefs.rho**2) * coefs.sigma
```

[20]:
```
# Calculate residual variability component of infrastructure electricity demand
# First construct an array of normally distributed random values
```

```
# (centred around 0, array size = realisations size x timeseries size)
noise_array = rng.standard_normal((realisations_size, temperature.size)).T *␣
 ↪float(sigma_z)
# Then apply the first-order AR function (Equation 4, Fallon et al. 2023)
AR_array = pd.DataFrame(AR1(noise_array, float(coefs.rho)), index=temperature.
 ↪index)
```

[21]:
```
# Stochastic demand (normalised) = demand (normalised) + AR term
demand_stoch_norm = AR_array.add(demand_norm.values)
demand_stoch_norm.columns.name="realisation"
```

4.3 Calculation of stochastic demand (physical units)

Apply the weekday pattern of demand, converting the stochastic demand timeseries (normalised) into physical units

[22]:
```
# Convert weekday_pattern into a 7x7 grid, rotating the start day of the week
weekday_pattern_array = extend_weekday_array(weekday_pattern.values,␣
 ↪realisations_size, len(temperature)).T.squeeze()
```

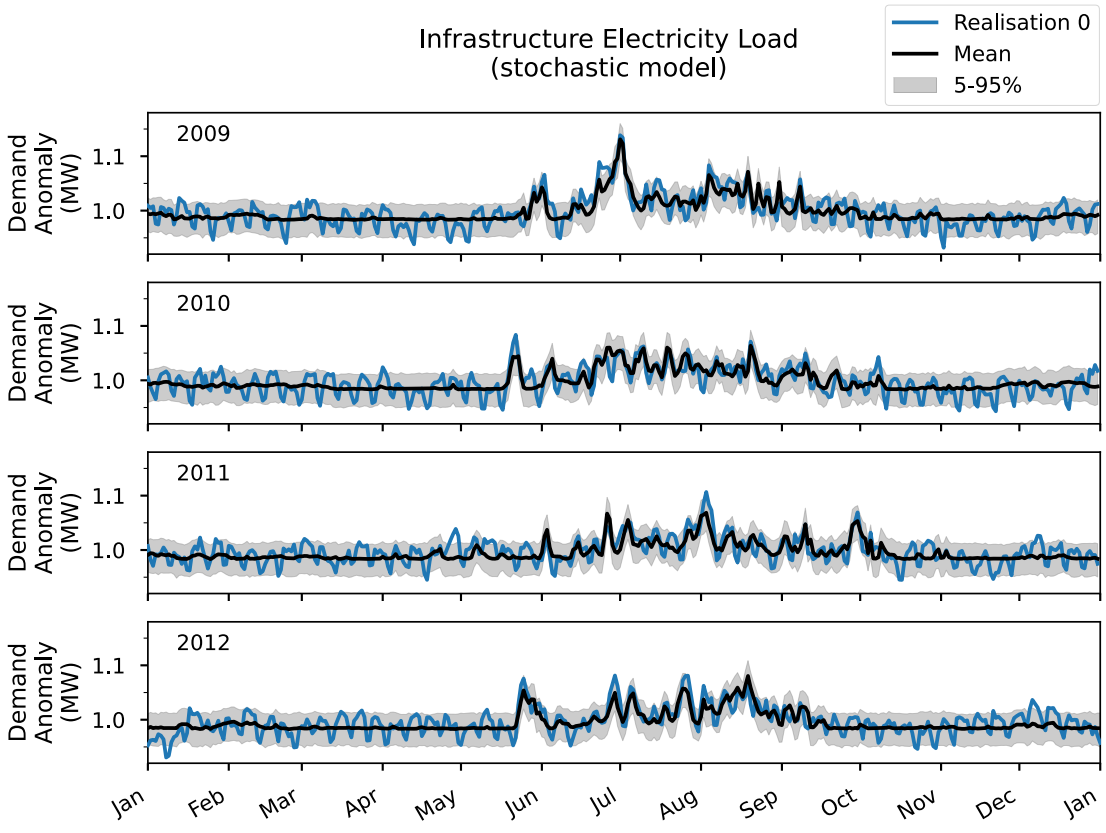
[23]:
```
# Multiply stochastic demand (normalised) by weekday pattern,
# obtaining physical units
demand_stoch = demand_stoch_norm * weekday_pattern_array
```

4.4 Plot Modelled Demand 2009-2012

Plot the mean and 5-95% range of all realisations of stochastic simulated timeseries of infrastructure electricity demand:

[24]:
```
fig, axs = plt.subplots(4, sharex=True, sharey=True)
realisation = 0

# plot some example year timeseries of infrastructure electricity demand
for ax, year in zip(axs.flatten(), range(2009, 2022)):
    series = demand_stoch[str(year):str(year+1)]

    x = np.arange(len(series))
    mean = series.mean(axis=1)
    q05 = series.quantile(0.05, axis=1)
    q95 = series.quantile(0.95, axis=1)

    l0 = series.T.loc[realisation].plot(ax=ax, legend=False, zorder=3,␣
 ↪color="tab:blue")
    l1 = mean.plot(ax=ax, legend=False, zorder=4, color="k")
    l2 = ax.fill_between(x, q05, q95, color="grey", alpha=0.4)

    ax.text(0.03, 0.8, str(year), transform=ax.transAxes)
    ax.set_ylabel(f"Demand\nAnomaly\n({demand_unit})")
```

```
# handle x-axis labels/ticks
apply_axis_calendar(ax)

# handle y-axis labels/ticks
ax.set_yticks( np.arange(1, 1.19, 0.1))
ax.set_yticks(np.arange(0.9, 1.3, 0.05), minor=True)
ax.set_ylim(0.92, 1.18)

labels = [f"Realisation {realisation}", "Mean", "5-95%"]
fig.legend([l0, l1, l2], labels=labels, bbox_to_anchor=(0.908,1.02))

# plot title
fig.suptitle("Infrastructure Electricity Load\n(stochastic model)");
```



5 Export datasets

Save output to CSV: Modelled (deterministic) demand

[25]:
```
# Export deterministic demand timeseries
demand_anom.to_csv(f"outputs/demand_{region}_anomaly_noweekday.csv")
demand_norm.to_csv(f"outputs/demand_{region}_normalised_noweekday.csv")
```

Save output to CSV: Modelled stochastic demand

[26]:
```
# Export stochastic demand timeseries
demand_stoch_norm.to_csv(f"outputs/
  ↪demand-stochastic_{region}_normalised_noweekday.csv")
demand_stoch.to_csv(f"outputs/demand-stochastic_{region}_{demand_unit}.csv")
```