MathWorks Math Modeling Challenge
A program of siam

SAMPLE PAPER: EXCELLENT

The pre-triage judges were impressed with the executive summary, which makes a good first impression. The team provides a brief overview of the problem, recommendations based on their results, and they provide a good sense about how they approached the problem.

The team made use of an ODE for the first part of the problem, and they gave specific guidance about how they approximated their final model. It was not clear, though, how they used the data to approximate the ambient temperature at any given time during the day other than to say they used the hourly data provided to them. The team provided good insights into the robustness of their model. They used a "Sobol analysis" but did not provide a citation in the text for their approach. They did make good use of citations and references throughout the rest of their paper.

For the second question the team made use of a Seasonal Auto-Regressive Integrated Moving Average with Exogenous Variables model (SARIMAX). They did not provide details as to whether or not the data is consistent with the assumptions of the approach. The team used linear regression to model the future temperature trends as well as population trends. The team did a good job of presenting their results, but little analysis on how well the regression matched the data was provided.

For the third question, the team defines a "heat index." The team states that they use the definition used by NOAA but do not include a citation, which is unfortunate as there are other definitions of the heat index. The team made direct use of their model from the first question to obtain their results. This was one of the few entries in which the team incorporated their previous work into the methods used to address the third question.

Hot Button Issue: *Staying Cool as the World Heats Up*

EXECUTIVE SUMMARY

Dear Birmingham local authorities,

As global temperatures rise, understanding the effect of heatwaves on urban city-spaces and residential areas is becoming increasingly important. This understanding is integral for protecting vulnerable populations, ensuring sustainable energy management, and informing effective policy decisions to mitigate the adverse impacts of extreme heat events.

We first predicted the fluctuation of internal temperatures over a 24-hour period for 4 different dwelling-types in Birmingham. We employed a thermal Ordinary Differential Equation (ODE) model to capture the fundamental heat transfer process within a building incoporating factors like solar radiation, convection and thermal properties of the dwellings. We predict that the maisonnette will have the highest peak temperature, with the flat, the semi-detached house and the terraced house following respectively. We also show that there is a lag time associated with the correlation of the outside ambient temperature and the internal temperature fluctuation. This lag is due to the thermal inertia of the building materials, showing the importance of considering building design and materials in mitigating heatwave impacts. We utilized Sobol sensitivity analysis to calculate the robustness of our model, and find out which parameters have the greatest impact on final model output – we show that specific heat capacity and shadyness are by far the greatest factors, accounting for 48% and 29% of the models output respectively. This suggests that improving the specific heat capacity of building materials, and increasing the shade on buildings are two key areas for intervention. More detailed analysis is shown in Figure 1, illustrating the temporal dynamics of indoor temperature changes across different dwelling types.

We then used an Seasonal Auto-Regressive Integrated Moving Average Model (SARIMAX) to model the predicted peak demand to account for during summer months. We find that peak demand is consistently highest in June and that over a period of 20 years there is an increase in the peak demand that will need to be accounted for. This increase is likely driven by long term increases in domestic energy consumption by consumers, with low sensitivities to our exogenous maximum temperature data implying air conditioning's role in the energy mix, on the demand side, will remain limited. This is despite assumed long term increases in maximum temperatures due to climate change. However, this overall increase in demand is accompanied with a sustained increase in population, showing that our model predictions for per capita energy use will remain somewhat consistent, likely due to increases in energy efficiency coinciding with increased reasons for demand.

For the development of heatwave risk metrics, we used our model from the first question calibrated on a heatwave in 2022 to find maximum heat indices (humidity adjusted temperatures) for different dwelling classes. By weighting dwelling type for each ward alongside a suitable heuristic, we were able to produce an aggregated per capita maximum heat index value for each ward. By extension, this provides a way to quantify the risk of average heat stroke risk by ward through the combination of this "feels-like" metric with homogenous internal body temperature. We previously established that grid stability and demand is a wholly separate issue in question 2, so we decided to focus on this problem entirely through a public health perspective. The primary benefit of our approach is that it can be augmented with existing weather forecast solutions to provide live predictions that still capture the relative risks of different wards due to their dwelling types.

We believe that our models can be used by both home-owners and policymakers to help reduce risk and inform decisions oriented to cool down cities and residential areas during periods of heatwave. Our models could also be used to inform energy management going forwards. We provide specific factors to target through informed policy to aid the many individuals across the UK who may be vulnerable during a period of heatwave.

Contents

Hot Button Issue: *Staying Cool as the World Heats Up*

1 Question 1: Hot to Go

1.1 Defining the Problem

The first problem requires us to construct a model to predict the internal temperature of any non-air-conditioned dwelling during a 24-hour heatwave period. We have modelled this against the July 2022 heatwave in Birmingham.

1.2 Assumptions

Heat loss through conduction and convection is the primary mechanism affecting indoor temperatures

- **Justification:** We assume that the radiative effects of Boltzmann's law are negligible in the temperature domain studied.

No dwellings have integrated air conditioning units.

- **Justification:** Less than 5% of the population of the UK has air conditioning in their home [7].

Indoor air mixing is uniform and therefore all rooms have the same temperature.

- **Justification:** Internal temperature gradients would be too sensitive and specific to building types to robustly model via a first-order differential equation.

The initial indoor temperature for all buildings is 20°C.

- **Justification:** Values of room temperature are generally cited as 18-21°C so we can assume that an initial value of 20°C represents a comfortable internal temperature before any external heating has an effect. [3]

The heat transfer due to wind speed follows the convection coefficient formula

- **Justification:** The formula is empirically derived and for the sake of model elegance no external factors for heat transfer are included.

No structural or physical degradation affect insulation performance over time

- **Justification:** Insulation materials can degrade due to moisture infiltration, settling, and material breakdown, but we assume that building regulation and retrofitting provide somewhat continual maintenance and some degree of consistency.

The effect of doors and windows have the same convection effect in all properties that we modelled.

- **Justification:** The absence of granular architectural data for the dwellings provided leads us to assume that the effects of windows and doors have a similar net convective effect. Additionally, our model directly incorporates an empirical convective formula which indirectly accounts for airflow around doors and windows.

Wind speed and convection maintain a linear relationship throughout the 24-hour period.

- **Justification:** Within the range of wind speeds modelled, non-linear effects such as turbulent eddies and boundary layer separation are rare and therefore linearity can be assumed.

All buildings studied have the same heat capacity.

- **Justification:** We assume thermal mass is similar for each dwelling-type.

1.3 The Model

1.3.1 Model development

We chose to model diurnal internal temperatures during a heatwave with a thermal Ordinary Differential Equation (ODE). Approaching this problem with an ODE allows us to generate predictions grounded in empirical physical truth as it captures the fundamental heat transfer processes occurring in a building such as heat gain from solar radiation and heat loss to the environment, and convection. Additionally, it handles time-varying inputs such as outdoor temperature and solar radiation which allows for accurate and elegant modelling of internal temperature fluctuation.

We considered the usage of other models such as random forest and gradient boosting models. These models could suitably capture the complex relationship, but would lack the physical basis that an ODE allows for. Additionally, more data would be required to train an accurate model and the risk of overfitting would mean that further measures would be required to maintain model accuracy and generalism.

1.3.2 Model execution

We used the provided dataset to parametrize our ODE. Specifically, we used the hourly heatwave temperatures and data on dwelling types and their respective unit sizes.

We used the `solarpy` library to accurately predict solar beam irradiance with high-granularity anywhere on earth. We used the `scipy.odeint` package to numerically integrate a system of ordinary differential equations using the lsoda method [9][5].

Symbol	Variable	Unit	Home 1	Home 2	Home 3	Home 4
u	Thermal transmittance	$W/m^2 \cdot K$	2.1	0.31	0.74	1.8
A	Heat loss area	m^2	139	74	59	96
C_p	Heat capacity	J/K	12.6×10^6	12.6×10^6	12.6×10^6	12.6×10^6
T_{in}	Indoor temperature	°C	20	20	20	20
$U_a(t, A, u)$	Heat transfer coefficient	$J/(h \cdot K)$	$1,051,740$	$82,584$	$157,212$	$622,080$
t	Time	hours	0-23	0-23	0-23	0-23
s	Shadiness factor	n/a	0.7	0.5	0.5	0.7

Table 1: Model parameters for different home types

We collated empirical heat transfer coefficient data [4], specific heat capacity data [2]
As such, the differential equation we used to model the problem is provided below:

$$\frac{dT_{in}}{dt} = \frac{1}{C_p}\left[-U_a(t, A, u)(T_{in} - T_{amb}(t)) + Q_{solar}(t, A) \cdot s\right] \tag{1}$$

1.4 Results

Using our thermal ODE model, we predicted internal dwelling temperature discretized in one-hour timesteps during the 24-hour heatwave in Birmingham on July 19th, 2022. Figure 1 shows how internal temperature in different dwelling types changes with regard to ambient outside temperature. We therefore predict that the

'maisonnette' will have the highest peak temperature, with the flat, the semi-detached house and the terraced house following respectively.

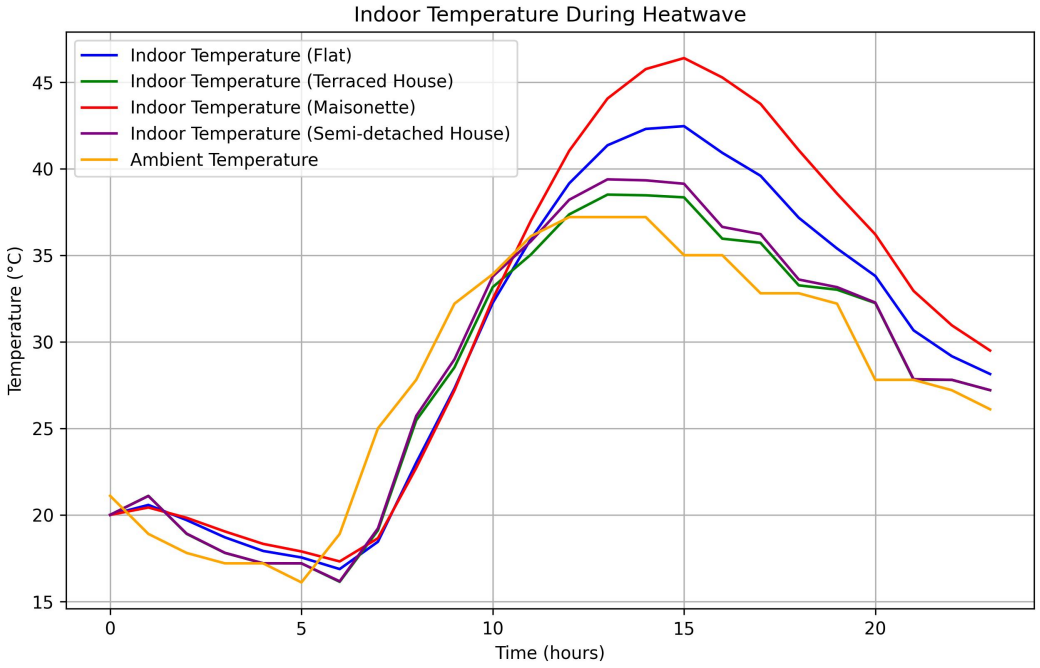

Figure 1: Internal dwelling temperature plotted against ambient outdoor temperature.

1.5 Sensitivity analysis

We utilize Sobol analysis (variance-based sensitivity analysis) to quantitatively measure the robustness of our model.

Parameter	S_1 **(First-Order)**	S_1 **Conf (Confidence)**	S_T **(Total)**	S_T **Conf (Confidence)**
C_p	0.48274667	0.17215788	0.5230437	0.12683454
A	0.20129002	0.1149062	0.2386194	0.10131119
s	0.2953626	0.09787687	0.33772019	0.10466462
U Value	0.03176134	0.05759425	0.04948288	0.02911131

Table 2: First-Order and Total Sobol' Indices with Confidence Intervals

From the first-order Sobol' indices (S_1), we can see that C_p (specific heat capacity) has the largest individual effect on the model's output, with an S_1 value of 0.4827, followed by Shadyness (0.2954) and Area (0.2013). This indicates that the value for specific heat capacity affects the model's output by 48%. This indicates that changes in the specific heat capacity have the most significant impact on the variation in indoor temperature.

The total Sobol' indices (S_t) consider both the direct effects and the interactions between parameters. Similar to the first-order indices, C_p contributes the most to the total variance (0.5230), followed by Shadyness (0.3377) and Area (0.2386). The confidence intervals for the total indices also suggest a strong contribution of C_p to the variance.

Parameter Pair	S_2 (Second-Order)	S_2 Conf (Confidence)
(C_p, A)	0.00808149	0.30484754
(C_p, s)	0.05786142	0.35510765
(C_p, U_a)	0.05334028	0.37256475
(A, s)	-0.00380012	0.18256259
(A, U_a)	-0.03893358	0.14545158
(s, U_a)	-0.03959983	0.18491372

Table 3: Second-Order Sobol' Indices with Confidence Intervals

The second-order Sobol' indices show interactions between parameters affecting indoor temperature during a heatwave. The strongest interaction is between C_p (specific heat capacity) and Shadyness, followed by the combination of C_p and U_a (thermal conductivity). These interactions suggest that both the material properties and shading play an important role in temperature regulation. In contrast to this, the interaction between Area and Shadyness, and Area and U_a, have much smaller or even negative effects. This indicates limited impact when considered together.

1.6 Discussion

The data shows that indoor temperature in different dwelling-types rises with less variance but as the ambient temperature reaches peak temperature, all dwelling types rise correspondingly with a lag factor. Each dwelling type is differentiated by its variance in corresponding peak internal temperatures. In our Sobol analysis, we determine specific heat capacity and shadyness to be two key driving factors in reducing internal building temperature. As such, we recommend intervention in the construction of new buildings to be targeted towards improving those two areas.

1.7 Strengths and weaknesses

Our ODE approach allows us to dynamically model time-varying effects and gain a mechanistic understanding of the driving features behind internal dwelling temperature fluctation. This is particularily important for a model predicting internal temperatures as a mechanistic understanding can help home-owners create the necessary mitigations to keep their home cool during heat-waves. We use Sobol analysis to critically analyse the robustness and sensitivity of our model, which is a significant strength as it explores the entire parameter space as opposed to local sensitivity about a point. This analysis would be critical if we were to extend the temporal domain of our modelling, as small inaccuracies and sensitivities would quickly amplify.

The model does not account for uncertainty or inaccuracy in the weather data, particularly in the solar irradiation predicted values. Additionally, our model neglects thermal mass distribution and uses an over-simplified model of a building. We assume uniform temperature and that the building is a singular thermal mass. Future iterations of our model might take into account specific geographical granularity in the ODE, and differing heat capacities contingent on more precise information on the composition of a buildings by building material.

2 Question 2: Power Hungry

2.1 Defining the Problem

The second question required us to predict the peak demand that Birmingham's power grid should be prepared to handle during summer months. We have also predicted a longer-term trend for peak energy demand.

2.2 Assumptions

The data exhibits seasonality with a periodicity of 12 months.

- **Justification:** Our model assumes that the seasonality of our data is 12-months. This is a long enough time to capture intra-year seasonal variations, particulary peak demand during the summer months.

The population projection is a reasonable adjustment factor.

- **Justification:** Our model assumes that our population projections influence consumption in an invariant way across all periods.

The forecast period is not affected by outliers or unusual events.

- **Justification:** We assume that there are no outliers or 'black-swan' events during our forecast period.

The exogenous variable (temperature) are not highly collinear.

- **Justification:** If temperature and population projections are highly correlated, this could lead to multicollinearity. This would affect the stability and interpretability of the model.

The forecast horizon is short enough that the model's dynamics remain relevant.

- **Justification:** We assume that the dynamics of the model remain stable and predictive over the maximum forecast horizon of 20 years.

Exogenous variables have a measurable and consistent effect on the power consumption

- **Justification:** We assume that social dynamics with regard to power consumption and temperature remains constant.

Population growth increases linearly with respect to a linear regression model

- **Justification:** We assume that popululation grows linearly over the forecast horizon.

Intra-year seasonal trends can be interpolated and predicted accurately, and do not change over the forecast horizon.

- **Justification:** We assume that the intra-year seasonal trend remains constant across both the years from 2012 and also into the prediction horizon.

2.3 The Model

2.3.1 Model development

We chose a Seasonal Auto-Regressive Integrated Moving Average with Exogenous Variables (SARIMAX) model [1] to predict the future peak demand required during summer months. This model is highly suitable as it allows seasonable variation to be accurately accounted for when predicting for summer months specifically.

We considered making use of other models, such as a Long-Short-Term-Memory (LSTM) model but we deemed this inappropriate as similar to machine learning approaches in question 1, much more data would be needed and overfitting could seriously impact model accuracy and performance.

2.3.2 Model execution

We used the provided data [6] for the maximum temperature and Varbes [8] for yearly population data. Additionally, the provided data for 2012 energy consumption formed the basis for our initial seasonal variation. We then interpolated this variation with the other national consumption values to enrich the initially seasonally-sparse dataset. We calculated ratios for each month and from there assumed the monthly variation in 2012 continues to the other years. The SARIMAX equation is as follows:

$$y_t = \alpha + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \sum_{k=1}^{m} \beta_k x_{t-k} + \sum_{l=1}^{s} \gamma_l y_{t-12l} + \sum_{m=1}^{s} \delta_m \epsilon_{t-12m} + \epsilon_t \tag{2}$$

Symbol	Variable	Unit
y_t	Target (dependent) variable - energy consumption	Unit of target variable (e.g., kWh)
α	Intercept (constant)	kWh
ϕ_i	Autoregressive coefficients for past values of y	Dimensionless
ϵ_t	Error term (residuals)	kWh
θ_j	Moving average coefficients for past error terms	Dimensionless
x_{t-k}	Exogenous variable (e.g., temperature)	°C
β_k	Coefficients for exogenous variables (temperature)	Dimensionless
γ_l	Seasonal autoregressive coefficients for seasonal component	Dimensionless
δ_m	Seasonal moving average coefficients for seasonal component	Dimensionless
ϵ_t	Error term (random noise or residual)	kWh

Table 4: Symbols, Variables, and Units for ARIMAX Model

Additionally, we use a linear regression model to predict both future population [8] and future maximum temperatures [6] per year.

$$\text{TempC}_t = \beta_0 + \beta_1 \cdot t + \epsilon_t \tag{3}$$

Symbol	Variable	Unit
Population_t	Predicted population at year t	People
β_0	Intercept	People
t	Time	Year
ϵ_t	Error term	People

Table 5: Symbols, Variables, and Units for Linear Regression Model

$$\text{Population}_t = \beta_0 + \beta_1 \cdot t + \epsilon_t \tag{4}$$

Symbol	Variable	Unit
TempC_t	Predicted temperature at year t	°C
β_0	Intercept	Celsius (°C)
t	Time	Year
ϵ_t	Error term	Celsius (°C)

Table 6: Symbols, Variables, and Units for Linear Regression Model

We used the `sklearn` library for the linear regression of temperature and population, and the `statsmodels` library for the ARIMAX functionality.

2.4 Results

Figure 3 shows a linear increase in overall energy consumption in the 20 year forecast window. Table 7 below shows the predicted peak demand for the summer months in the coming 5 years, showing the precise seasonal variation between the summer months.

Table 7: Monthly Data for Summer Months (June, July, August) Over Five Years

Month	2023 (kWH)	2024 (kWH)	2025 (kWH)	2026 (kWH)	2027 (kWH)
June	2.685×10^8	2.622×10^8	2.649×10^8	2.596×10^8	2.635×10^8
July	2.786×10^8	2.723×10^8	2.753×10^8	2.701×10^8	2.742×10^8
August	2.739×10^8	2.677×10^8	2.706×10^8	2.654×10^8	2.695×10^8

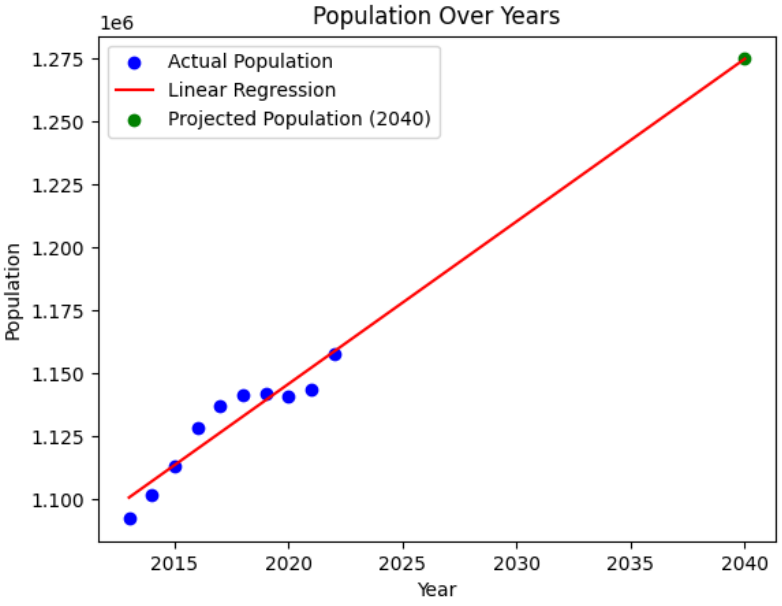

Figure 2: Population growth as we have predicted it to linearly grow.

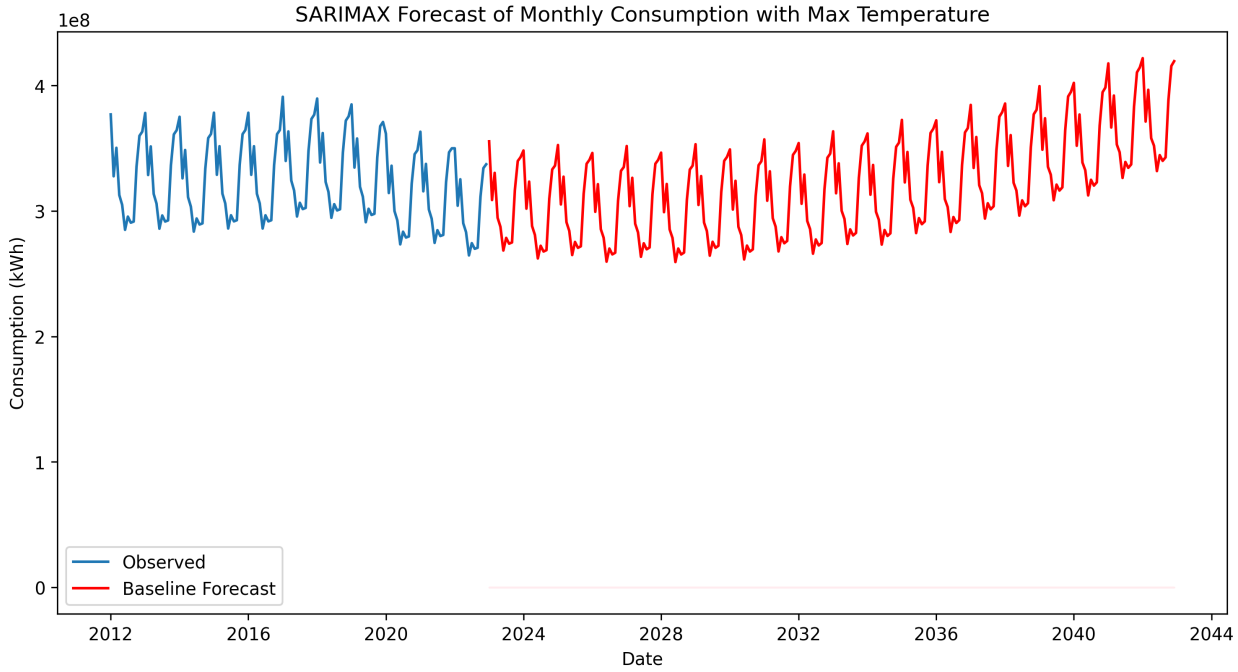

Figure 3: Total energy consumption for Birmingham - historical values and projections

2.5 Discussion

We found that our model suitably projects the long term growth in the demand of energy in Birmingham, capturing both seasonal variations, the expected medium term dip in demand as well as the expected long term increase. Our model predicts that June has consistently the highest temperatures, which is a crucial data point to account for in potential future energy management.

2.6 Sensitivity Analysis

For sensitivity analysis, we used a One At a Time (OAT) method which involved the use of a perturbation of $\pm 10\%$ for our maximum temperature exogenous data. We found that there was very little sensitivity with regard to max temperature, which makes sense as grid load is unlikely to vary by much due to the lack of AC in Birmingham and the UK.

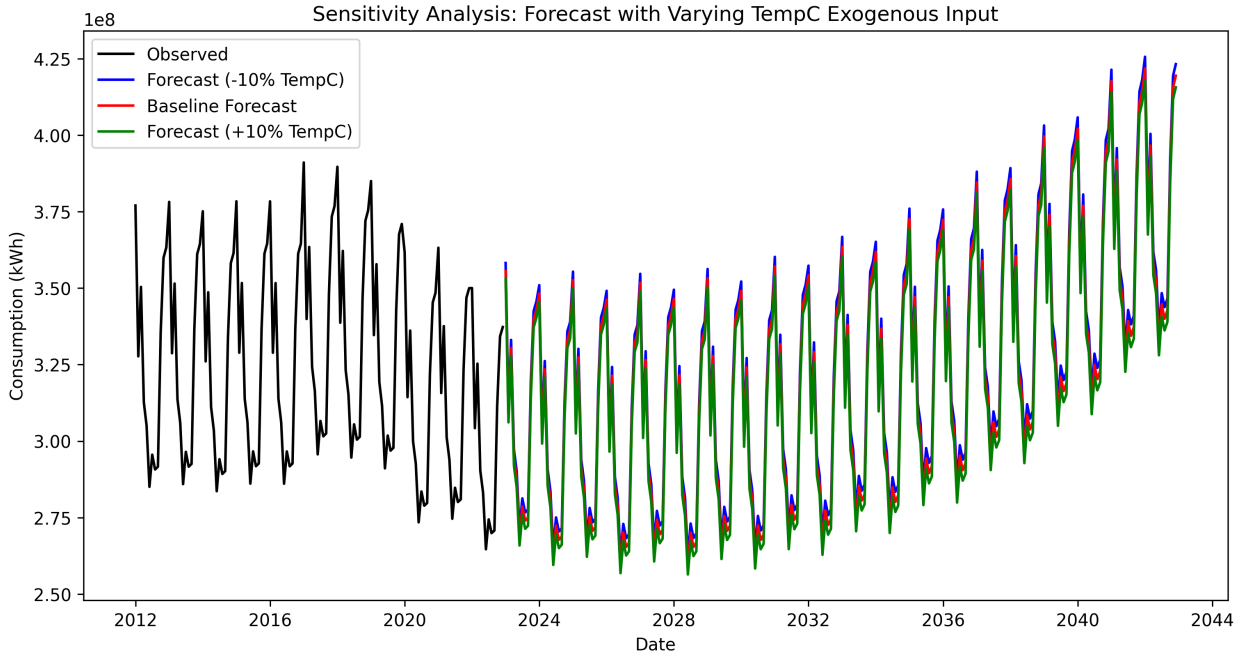

Figure 4: Time series projections with perturbed exogenous variables

Date	Change
2030-01-01	±0.88%
2035-01-01	±0.90%
2040-01-01	±0.91%

2.7 Strengths and weaknesses

Our model captures seasonal and trend components of the datasets, which is a significant strength. Additionally, it allows for the incorporation of exogenous variables which can play a critical role in ensuring prediction accuracy and relevancy. We also account for population growth adjustments which can enhance our model's real-world applicability.

However, we demonstrate that the exogenous variables we chose have little effect on the final model output through OAT sensitivity analysis. To improve our model for future use, we would include a much wider range of exogenous variables that have a more causal effect on eventual energy consumption. Finally, we would collate a more data-rich time series for monthly energy consumption rather than just interpolating.

3 Question 3: Beat the Heat

3.1 Defining the Problem

The third problem requires us to construct a vulnerability metric with municipal granularity which can be used by the authorities to allocate resources to minimize the effects of a heat wave or a power grid failure.

3.2 Assumptions

To simplify the problem, we make the following assumptions:

The heat index is primarily a function of temperature and relative humidity.

- **Justification:** This is a standard assumption employed by the National Weather Service (NWS) Heat Index, derived by Lu and Romps (2022).

The heat index is a suitable gauge for risk associated with heatwaves.

- **Justification:** The primary effects of heatwaves are associated with public health risks, as risks to grid stability are negligible due to air conditioning uptake in the UK [**?**]. If heat indices exceed the internal body temperature of a human, the risk of heatstroke is introduced.

We can use our results from Question 1 to develop profiles for different dwelling classes.

- **Justification:** While our heat index can be re-based on any arbitrary heatwave and associated prior weather forecast to develop indices for a live, evolving threat, choosing a previous heatwave ensures that the intrinsic relative dynamics of each municipal district remain invariant.

The population is uniformly affected by heat exposure without accounting for individual physiological differences.

- **Justification:** The assumption of population homogeneity is a standard statistical simplification, further reinforced by the fact that internal body temperature is generally uniform.

Heat stroke risk corresponds with heat indices.

- **Justification:** This assumption relates to how the indices are interpreted. However, we can reasonably discard irrational behaviors—such as voluntary exercise in heatwave conditions—when considering public health risks during a crisis.

Inheritance of dwelling class is suitable from Question 1.

- **Justification:** We define:
  - 1- and 2-bedroom houses as semi-detached houses built in 1940.
  - 3-, 4-, and 5+ bedroom houses as terraced homes built in 1915.
  - 1- and 2-bedroom flats as flats built in the 1980s.
  - 3-, 4-, and 5+ bedroom flats as 2010 maisonettes.

  These classifications are based on bedroom numbers aligning with previously established dwelling classes from Question 1. This assumption is valid because the variance associated with an arbitrary house within its given dwelling class is symmetric. Thus, on aggregate, these variances cancel out, ensuring that per capita heat indices remain appropriate.

3.3 The Model

After determining the importance of the heat index in calculating the vulnerability score, we incorporate the model used in Question 1 to provide the temperatures experienced indoors during heatwaves and find the maximum heat index experienced in each class of dwelling. The `pythermalcomfort` library was used to calculate this heat index based on temperature and relative humidity, using the following equation:

$$HI = c_1 + c_2T + c_3RH + c_4T \cdot RH + c_5T^2 + c_6RH^2 + c_7T^2 \cdot RH + c_8T \cdot RH^2 + c_9T^2 \cdot RH^2 \tag{5}$$

The parameters for this equation are defined in the following table:
The empirical coefficients c_1 through c_9 are derived from NOAA's heat index formulation.

Once the peak heat index of each class of house was found, assumptions were made to approximate the number of each type of dwelling in each region of Birmingham. By calculating the number of rooms and

Symbol	Variable	Unit
HI	Heat Index	°C
T	Temperature	°C
RH	Relative Humidity	%
c_1-c_9	Empirical coefficients	-

Table 8: Parameters for the Heat Index equation

using that figure to calculate the population living in each class of house, a "per capita heat index" can be calculated for each district, which is directly related to the vulnerability score:

$$\frac{1}{P_T}\sum_{i=1}^{n} HI_i \cdot P_i \tag{6}$$

The parameters for this equation are defined in the following table:

Symbol	Variable	Unit
P_T	Total population of the region	-
HI_i	Maximum heat index for house class i	°C
P_i	Population in house class i	-
n	Number of dwelling types	-

Table 9: Parameters for the per capita heat index equation

This formula has been applied to calculate the per capita heat index, where $n = 4$ for each of the types of dwellings.

3.4 Results

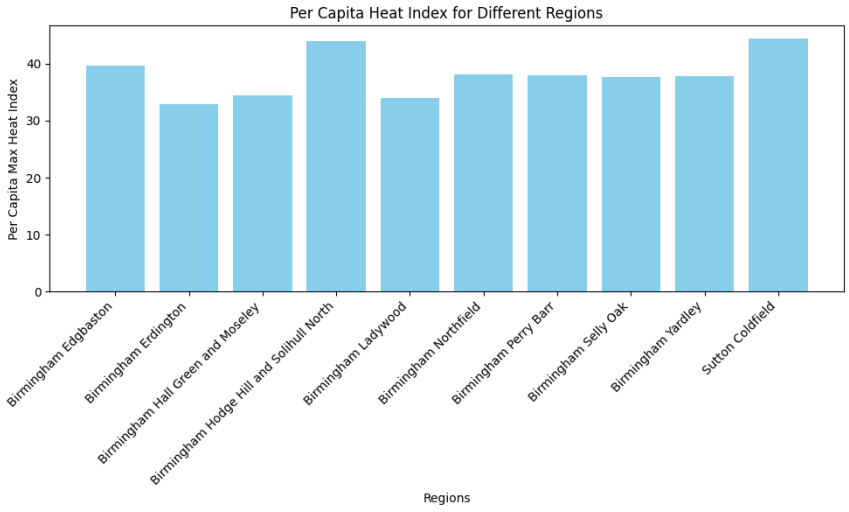

Figure 5: Bar chart of the per capita heat indexes.

The bar chart demonstrates the variations in heat vulnerability across Birmingham's districts, with Sutton Coldfield and Birmingham Hodge Hill and Solihull North having the highest per capita heat indexes, implying their greater risk of heat-related health impacts. Birmingham Edgbaston, Northfield, and Perry Barr are relatively vulnerable areas, while Birmingham Edington and Hall Green and Moseley have lower per capita heat indexes, suggesting these region's higher resilience to extreme heat. The variation between these regions

however is quite small which could suggest that heat vulnerability is a widespread issue across Birmingham rather than just being in a few higher risk areas. This emphasises the need for strategies to deal with these area's vulnerabilities.

3.5 Discussion

These results highlight the importance of humidity in determining perceived temperature. The model suggests that even moderate temperature increases can lead to dangerous heat index values when humidity is high. Future work should explore urban heat island effects and individual physiological differences.

3.5.1 Using the vulnerability score to mitigate heat-related risks

Regions that have a high per capita heat index correspond to where residents are at a greater risk of heat-related illnesses, and so targeted intervention strategies are essential.

To significantly reduce the impact of heatwaves in Birmingham, authorities can make use of our calculated vulnerability scores when creating their emergency response and resource allocation strategies. Here is our proposed approach for doing so:

By mapping out vulnerability scores across different districts, authorities can identify high-risk areas that require immediate intervention. A general heuristic for this would be to compare the heat index to typical body temperatures but inputs from public health experts would likely be needed. In these districts, the following measures can be prioritized: Local public buildings such as community centres, libraries, and schools can be converted into cooling centres where residents can find relief from extreme heat. Moreover, districts with the highest vulnerability scores should be given water stations, portable fans, and cooling kits. Furthermore, awareness campaigns and education on heat-related illnesses and preventative methods would go a long way to further mitigate the risks of days of extreme heat.

In addition, the vulnerability index needs to be continuously updated based on real-time weather forecasts and population variations. This can be achieved through making use of live temperature and humidity data that can refine our heat index calculations hence improving the accuracy of the vulnerability score whilst enabling dynamic risk assessments. Residents in high-risk areas could also receive SMS or app-based alerts forecasting incoming heatwaves whilst informing them of safety measures to take. Finally, long-term strategies should include increasing green spaces, implementing reflective roofing materials, and improving building insulation to reduce indoor heat retention.

By incorporating these approaches, Birmingham can effectively use these vulnerability scores to minimize the impact of heat-related health risks and ensure that resources can be distributed appropriately during extreme weather events. The vulnerability score can be further improved by focusing on refining the model by incorporating additional socio-economic factors, urban heat island effects, and physiological variations among the population.

3.6 Strengths and weaknesses

Our model accurately determines per capita head consumption from the composition of dwelling class per ward, which is a significant strength. Additionally, by leveraging our existing, sophisticated model from question 1, we are able to achieve similar levels of versatility and we can thus inherit existing uncertainty and sensitivity measures.

However, we do also inherit existing issues from question 1 and more approaches. One such inherited issue is the lack of quantitative empirical data. Deploying any such models would necessitate more effective calibration to real world typical specific heat capacities of buildings. Another could be the lack of uncertainty quantification with weather data, relying on augmentation with pre-existing solutions for weather prediction. A new shortcoming introduced downstream of our internal ODE model would be the assumption per capita heat indices are obtained from people staying at home. People may venture outside where it is more or less shaded, or go to public places like parks or shopping centres. Collation with more data rich land use information would subsequently yield quite beneficial results, particularly in the city centrer.

4 Conclusion

We examined the effect of heatwaves on cities and residential areas. Specifically, we modelled the internal temperature fluctuation for 4 different dwelling types in Birmingham during the June 2022 heatwave. The model predicted that maisonnettes would have the highest peak internal temperature, and that all housing types would have a thermal inertia causing a lag between the peak ambient temperature and the peak internal temperature.

We also modelled the increase in peak demand capacity for a forecast horizon of 20 years, particularly examining the seasonality around the summer months. We conclude that June is consistently the month that has the highest energy consumption. Additionally, we establish that the maximum demand increases over the period of 20 years we predicted for.

Finally, we use predictions from our examination of internal temperature fluctuation in buildings (question 1) to build a risk model that calculates the heat index in various municipalities of Birmingham. This risk model will allow health officials to provide targeted support to communities

Our research could be improved by incorporating more granular architectural data, accounting for thermal mass distribution of buildings and addressing uncertainties in weather data such as solar irradiation. The SARIMAX model could benefit from the introduction of more relevant exogenous variables. For our risk model, we could incorporate more diverse prediction factors.

5 Acknowledgements

Large language models (LLMs) were utilised for purely administrative tasks, such as assisting with data processing code. No primary contributions architecturally or textually were made using LLMs.

References

[1] Fahad Alharbi and Dénes Csala. A seasonal autoregressive integrated moving average with exogenous factors (sarimax) forecasting model-based time series approach. *Inventions*, 7:94, 10 2022. doi:10.3390/inventions7040094.

[2] Protons for Breakfast. Estimating the heat capacity of my house, 2022. Accessed: 2025-03-01. URL: https://protonsforbreakfast.wordpress.com/2022/12/19/estimating-the-heat-capacity-of-my-house/.

[3] British Gas. What's the ideal home temperature? *British Gas*, 2023. Accessed: 2025-03-01. URL: https://www.britishgas.co.uk/the-source/no-place-like-home/whats-the-ideal-home-temperature.html.

[4] David Glew, Mark Collett, Martin Fletcher, Adam Hardy, Beth Jones, Dominic Miles-Shenton, Kate Morland, Jim Parker, Kambiz Rakhshanbabanari, Felix Thomas, and Christopher Tsang. Deep report 4.00 brick material properties. Report DEEP 4.00, Leeds Beckett University (LBU), UK, 2024. Prepared for DESNZ. URL: https://assets.publishing.service.gov.uk/media/6717bd6b38149ce9d09e383f/4._DEEP_Materials_Report.pdf.

[5] Sven Hoyer and Ryan Brown. Solarpy: A python package for solar irradiance prediction, 2021. Accessed: 2025-03-01. URL: https://solarpy.readthedocs.io/.

[6] MathWorks. M3 2025 accompanying data sheet, 2025. Data sheet. URL: https://m3challenge.siam.org/897bjhb54cgfc/.

[7] Marshall Shepherd. Most homes in the united kingdom have no air conditioning—unprecedented heat looms. *Forbes*, 2022. Accessed: 2025-03-01. URL: https://www.forbes.com/sites/marshallshepherd/2022/07/15/most-homes-in-the-united-kingdom-have-no-air-conditioningunprecedented-heat-looms.

[8] Varbes. Varbes property analytics, 2019. Accessed: 2025-03-01. URL: https://www.varbes.com/.

[9] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, et al. *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, 2020. Accessed: 2025-03-01. URL: https://www.scipy.org.

A Code Appendix

```

#arimax.py

import numpy as np
import pandas as pd
from statsmodels.tsa.statespace.sarimax import SARIMAX
import matplotlib.pyplot as plt
import q2_data_parser

data = pd.read_csv('data/q2newdata.csv', header=0)

exog_data = pd.read_csv('data/q2_maxtemp_data.csv', header=None, names=['Year', 'Date', 'TempF', 'TempC'])
exog_data['Year'] = exog_data['Year'].astype(int)
exog_data.set_index('Year', inplace=True)

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
monthly_data = data.melt(id_vars=['Year'], value_vars=months, var_name='Month', value_name='Consumption')
monthly_data['Date'] = pd.to_datetime(monthly_data['Year'].astype(str) + '-' + monthly_data['Month'], format='%Y-%b')
monthly_data = monthly_data.sort_values('Date').set_index('Date')
monthly_data['Year'] = monthly_data.index.year
monthly_data = monthly_data.merge(exog_data[['TempC']], left_on='Year', right_index=True, how='left')
monthly_data = monthly_data.drop('Year', axis=1)

model = SARIMAX(monthly_data['Consumption'], exog=monthly_data['TempC'], order=(3, 3, 3), seasonal_order=(1, 1, 1, 12))
model_fit = model.fit()
forecast_steps = 240
future_dates = pd.date_range(start=monthly_data.index[-1] + pd.DateOffset(months=1), periods=forecast_steps, freq='M')
future_exog = pd.DataFrame({
    'TempC': [q2_data_parser.project_max_temperature(date.year) for date in future_dates]
}, index=future_dates)

forecast_result = model_fit.get_forecast(steps=forecast_steps, exog=future_exog)
forecast_baseline = forecast_result.predicted_mean
confidence_intervals = forecast_result.conf_int()

updated_baseline_forecast = []
for date, consumption in forecast_baseline.items():
    year = date.year
    updated_baseline_forecast.append([date, consumption * q2_data_parser.project_population(year)])
forecast_baseline = pd.Series(dict(updated_baseline_forecast))

total_consumption = []
for date, consumption in zip(monthly_data.index, monthly_data['Consumption']):
    year = date.year
    total_consumption.append(consumption * q2_data_parser.project_population(year))
monthly_data['Consumption'] = total_consumption

print("------ Forecasting Monthly Consumption with Max Temperature using SARIMAX ------")
print("Confidence intervals for the baseline forecast:")
print(confidence_intervals)

plt.figure(figsize=(12, 6))
plt.plot(monthly_data.index, monthly_data['Consumption'], label='Observed')
plt.plot(forecast_baseline.index, forecast_baseline, color='red', label='Baseline Forecast')
plt.fill_between(forecast_baseline.index,
                 confidence_intervals.iloc[:, 0],
                 confidence_intervals.iloc[:, 1],
```

```
                color='pink', alpha=0.3)
plt.title('SARIMAX Forecast of Monthly Consumption with Max Temperature')
plt.xlabel('Date')
plt.ylabel('Consumption (kWh)')
plt.legend()
plt.savefig('monthly_forecast_plot_with_exog_q2.png', dpi=300, bbox_inches='tight')

print("Forecast for the next 12 months (baseline):")
print(forecast_baseline.head(60))

# ---------------- Sensitivity Analysis ----------------
sensitivity_factors = [0.9, 1.0, 1.1]
sensitivity_forecasts = {}

for factor in sensitivity_factors:
    scenario_exog = future_exog.copy()
    scenario_exog['TempC'] = scenario_exog['TempC'] * factor

    scenario_forecast_result = model_fit.get_forecast(steps=forecast_steps, exog=
        scenario_exog)
    scenario_forecast = scenario_forecast_result.predicted_mean

    scenario_updated_forecast = []
    for date, consumption in scenario_forecast.items():
        year = date.year
        scenario_updated_forecast.append([date, consumption * q2_data_parser.
            project_population(year)])
    sensitivity_forecasts[factor] = pd.Series(dict(scenario_updated_forecast))

plt.figure(figsize=(12, 6))
plt.plot(monthly_data.index, monthly_data['Consumption'], label='Observed', color='black')
colors = {0.9: 'blue', 1.0: 'red', 1.1: 'green'}
labels = {0.9: 'Forecast (-10% TempC)', 1.0: 'Baseline Forecast', 1.1: 'Forecast (+10% TempC
    )'}
for factor, forecast_series in sensitivity_forecasts.items():
    plt.plot(forecast_series.index, forecast_series, color=colors[factor], label=labels[
        factor])
plt.title('Sensitivity Analysis: Forecast with Varying TempC Exogenous Input')
plt.xlabel('Date')
plt.ylabel('Consumption (kWh)')
plt.legend()
plt.savefig('sensitivity_analysis_forecast_q2.png', dpi=300, bbox_inches='tight')
print("Sensitivity Analysis Forecast (first 12 months):")
for factor, series in sensitivity_forecasts.items():
    print(f"\nScenario factor {factor}:")
    print(series.head(12))
```

```

    #data_parser.py

    import pandas as pd

class DataParser:
    def __init__(self, path, columns, process_func=lambda x: x):
        self.path = path
        self.data = pd.read_csv(self.path)
        self.data = process_func(self.data, columns)

    def get_data_from_column(self, id, column):
        return self.data[column].values[id]

def process_data(data, columns):
    data = data[columns]
    data = data.dropna()
    return data

def load_data(columns, path):
    return DataParser(path, columns, process_func=process_data)
```

```
columns = ['Time', 'Temperature (celsius)', 'Wind Speed (miles per hour)', 'Humidity (%)']
```

```
#model_3.py
    from pythermalcomfort.models import heat_index_lu
import ode
import pandas as pd
import numpy as np
from scipy.integrate import odeint
from solarpy import irradiance_on_plane
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
from SALib.analyze import sobol
from SALib.sample import saltelli
import data_parser

# Read the CSV file and print its head
data = pd.read_csv('data/q3data.csv')
print(data.head())


# 1915 style house
t_1915 = odeint(ode.model, ode.T_in_0, ode.t_hours, args=(ode.Cp_1915, ode.
    heat_loss_area_1915, ode.shadyness_1915, ode.Ua_1915))
heat_indices_1915 = []
for a, x in enumerate(t_1915):
    humidity = float(ode.parser.data['Humidity (%)'].values[a])
    heat_indices_1915.append(heat_index_lu(tdb=x[0], rh=humidity).hi)
max_1915 = max(heat_indices_1915)
print("Max HI for 1915 style house", max_1915)

# 1940 style house (this is the style we will perform sensitivity analysis on)
t_1940 = odeint(ode.model, ode.T_in_0, ode.t_hours, args=(ode.Cp_1940, ode.
    heat_loss_area_1940, ode.shadyness_1940, ode.Ua_1940))
heat_indices_1940 = []
for a, x in enumerate(t_1940):
    humidity = float(ode.parser.data['Humidity (%)'].values[a])
    heat_indices_1940.append(heat_index_lu(tdb=x[0], rh=humidity).hi)
max_1940 = max(heat_indices_1940)
print("Max HI for 1940 style house", max_1940)

# 1980 style house
t_1980 = odeint(ode.model, ode.T_in_0, ode.t_hours, args=(ode.Cp_1980, ode.
    heat_loss_area_1980, ode.shadyness_1980, ode.Ua_1980))
heat_indices_1980 = []
for a, x in enumerate(t_1980):
    humidity = float(ode.parser.data['Humidity (%)'].values[a])
    heat_indices_1980.append(heat_index_lu(tdb=x[0], rh=humidity).hi)
max_1980 = max(heat_indices_1980)
print("Max HI for 1980 style house", max_1980)

# 2010 style house
t_2010 = odeint(ode.model, ode.T_in_0, ode.t_hours, args=(ode.Cp_2010, ode.
    heat_loss_area_2010, ode.shadyness_2010, ode.Ua_2010))
heat_indices_2010 = []
for a, x in enumerate(t_2010):
    humidity = float(ode.parser.data['Humidity (%)'].values[a])
    heat_indices_2010.append(heat_index_lu(tdb=x[0], rh=humidity).hi)
max_2010 = max(heat_indices_2010)
print("Max HI for 2010 style house", max_2010)

csv2DArray = [
    list(map(float, line.strip().strip(",").split(",")[1:]))
    for line in open("data/q3data.csv").readlines()[1:]
]

houseColumns = [2, 3, 2, 3]

```

```
regions = []  # list of dictionaries
for row in csv2DArray:
    iStart = 0
    populationPerType = []
    for ind in houseColumns:
        iEnd = iStart + ind
        count = sum([row[iStart + i] * (((iStart + i) % 5) + 1) for i in range(ind)])
        populationPerType.append(count * row[-2])
        iStart = iEnd
    official_total_population = row[-1]
    regions.append({
        'pop_per_type': populationPerType,
        'official_total': official_total_population
    })

per_capita_heat_index = []
for region in regions:
    pop_per_type = region['pop_per_type']
    total_heat_index = (max_1940 * pop_per_type[0] +
                        max_1915 * pop_per_type[1] +
                        max_1980 * pop_per_type[2] +
                        max_2010 * pop_per_type[3])
    per_capita_heat_index.append(total_heat_index / region['official_total'])
    print("Raw housing counts per type:", pop_per_type)
    print("Official total population:", region['official_total'])
    print("Total heat index for region:", total_heat_index)
    print("Per capita heat index for region:", per_capita_heat_index, "\n")

regions_list = [
    "Birmingham Edgbaston",
    "Birmingham Erdington",
    "Birmingham Hall Green and Moseley",
    "Birmingham Hodge Hill and Solihull North",
    "Birmingham Ladywood",
    "Birmingham Northfield",
    "Birmingham Perry Barr",
    "Birmingham Selly Oak",
    "Birmingham Yardley",
    "Sutton Coldfield"
]

plt.figure(figsize=(10, 6))
plt.bar(regions_list, per_capita_heat_index, color='skyblue')
plt.xlabel('Regions')
plt.ylabel('Per Capita Max Heat Index')
plt.title('Per Capita Heat Index for Different Regions')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.savefig('q3_bar_chart.png')


# ----------------------------------------------------------------------
def model_max_hi(params):
    Cp, heat_loss_area, shadyness, Ua = params
    t_model = odeint(ode.model, ode.T_in_0, ode.t_hours, args=(Cp, heat_loss_area, shadyness
        , Ua))
    hi_values = []
    for a, x in enumerate(t_model):
        humidity = float(ode.parser.data['Humidity (%)'].values[a])
        hi = heat_index_lu(tdb=x[0], rh=humidity).hi
        hi_values.append(hi)
    return max(hi_values)

nominal_params = [ode.Cp_1940, ode.heat_loss_area_1940, ode.shadyness_1940, ode.Ua_1940]
problem = {
    'num_vars': 4,
    'names': ['Cp', 'heat_loss_area', 'shadyness', 'Ua'],
    'bounds': [[0.8 * param, 1.2 * param] for param in nominal_params]
```

```
}

param_values = saltelli.sample(problem, 128, calc_second_order=True)

Y = np.array([model_max_hi(params) for params in param_values])

# Perform Sobol sensitivity analysis on the model output.
Si = sobol.analyze(problem, Y, print_to_console=True)

print("\nSensitivity Analysis Results for 1940 Style House:")
print(Si)

def model_max_hi_1915(params):
    Cp, heat_loss_area, shadyness, Ua = params
    t_model = odeint(ode.model, ode.T_in_0, ode.t_hours, args=(Cp, heat_loss_area, shadyness, Ua))
    hi_values = []
    for a, x in enumerate(t_model):
        humidity = float(ode.parser.data['Humidity (%)'].values[a])
        hi_values.append(heat_index_lu(tdb=x[0], rh=humidity).hi)
    return max(hi_values)

nominal_params_1915 = [ode.Cp_1915, ode.heat_loss_area_1915, ode.shadyness_1915, ode.Ua_1915]
problem_1915 = {
    'num_vars': 4,
    'names': ['Cp', 'heat_loss_area', 'shadyness', 'Ua'],
    'bounds': [[0.8 * p, 1.2 * p] for p in nominal_params_1915]
}
param_values_1915 = saltelli.sample(problem_1915, 128, calc_second_order=True)
Y_1915 = np.array([model_max_hi_1915(params) for params in param_values_1915])
Si_1915 = sobol.analyze(problem_1915, Y_1915, print_to_console=True)
print("\nSensitivity Analysis Results for 1915 Style House:")
print(Si_1915)

# Sensitivity Analysis for the 1980 Style House
def model_max_hi_1980(params):
    Cp, heat_loss_area, shadyness, Ua = params
    t_model = odeint(ode.model, ode.T_in_0, ode.t_hours, args=(Cp, heat_loss_area, shadyness, Ua))
    hi_values = []
    for a, x in enumerate(t_model):
        humidity = float(ode.parser.data['Humidity (%)'].values[a])
        hi_values.append(heat_index_lu(tdb=x[0], rh=humidity).hi)
    return max(hi_values)

nominal_params_1980 = [ode.Cp_1980, ode.heat_loss_area_1980, ode.shadyness_1980, ode.Ua_1980]
problem_1980 = {
    'num_vars': 4,
    'names': ['Cp', 'heat_loss_area', 'shadyness', 'Ua'],
    'bounds': [[0.8 * p, 1.2 * p] for p in nominal_params_1980]
}
param_values_1980 = saltelli.sample(problem_1980, 128, calc_second_order=True)
Y_1980 = np.array([model_max_hi_1980(params) for params in param_values_1980])
Si_1980 = sobol.analyze(problem_1980, Y_1980, print_to_console=True)
print("\nSensitivity Analysis Results for 1980 Style House:")
print(Si_1980)

# Sensitivity Analysis for the 2010 Style House
def model_max_hi_2010(params):
    Cp, heat_loss_area, shadyness, Ua = params
    t_model = odeint(ode.model, ode.T_in_0, ode.t_hours, args=(Cp, heat_loss_area, shadyness, Ua))
    hi_values = []
    for a, x in enumerate(t_model):
        humidity = float(ode.parser.data['Humidity (%)'].values[a])
        hi_values.append(heat_index_lu(tdb=x[0], rh=humidity).hi)
```

```
    return max(hi_values)

nominal_params_2010 = [ode.Cp_2010, ode.heat_loss_area_2010, ode.shadyness_2010, ode.Ua_2010]
problem_2010 = {
    'num_vars': 4,
    'names': ['Cp', 'heat_loss_area', 'shadyness', 'Ua'],
    'bounds': [[0.8 * p, 1.2 * p] for p in nominal_params_2010]
}
param_values_2010 = saltelli.sample(problem_2010, 128, calc_second_order=True)
Y_2010 = np.array([model_max_hi_2010(params) for params in param_values_2010])
Si_2010 = sobol.analyze(problem_2010, Y_2010, print_to_console=True)
print("\nSensitivity Analysis Results for 2010 Style House:")
print(Si_2010)
```

```
#ode.py
    import numpy as np
from scipy.integrate import odeint
from solarpy import irradiance_on_plane
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
from SALib.analyze import sobol
from SALib.sample import saltelli
import data_parser

# ------------------------------------------------------------
# 1) Define building properties in consistent units
# ------------------------------------------------------------

# 1915 Terraced House
u_value_1915 = 2.1  # W/m   K (empirical value for solid brick walls)
heat_loss_area_1915 = 139  # m   (approximate area based on size)
Cp_1915 = 3.5 * 3600000
Ua_1915 = u_value_1915 * heat_loss_area_1915  # [J/(h K)]
shadyness_1915 = 0.7

# 2010 Maisonette
u_value_2010 = 0.31  # W/m   K (empirical value for insulated walls)
heat_loss_area_2010 = 74  # m   (approximate area based on size)
Cp_2010 = 3.5 * 3600000
Ua_2010 = u_value_2010 * heat_loss_area_2010
shadyness_2010 = 0.5

# 1980 Flat
heat_loss_area_1980 = 59  # m   (approximate area based on size)
u_value_1980 = 0.74  # W/m   K (effective conductivity for cavity wall)
Cp_1980 = 3.5 * 3600000
Ua_1980 = u_value_1980 * heat_loss_area_1980
shadyness_1980 = 0.5

# 1940 Semi-detached House
u_value_1940 = 1.8  # W/m   K (empirical value for transition-era solid walls)
heat_loss_area_1940 = 96  # m   (approximate area based on size)
Cp_1940 = 3.5 * 3600000
Ua_1940 = u_value_1940 * heat_loss_area_1940
shadyness_1940 = 0.7


# # 1980 Flat
# wall_mass_1980 = 3600
# specific_heat_capacity_brick_1980 = 980
# heat_loss_area_1980 = 59
# u_value_1980 = 1.7 * heat_loss_area_1980
# shadyness_1980 = 2/3
# # Cp_1980 = wall_mass_1980 * specific_heat_capacity_brick_1980
# Cp_1980 = 3.5 * 3600000
# Ua_1980 = float(u_value_1980 * heat_loss_area_1980 * 3600)

```

```
transmittance = 0.7

# ------------------------------------------------------------
# 2) Ambient temperature function
# ------------------------------------------------------------

parser = data_parser.load_data(data_parser.columns, "data/q1data.csv")

def T_amb(t):
    idx = min(int(t), 23)
    return float(parser.get_data_from_column(idx, 'Temperature (celsius)'))

# ------------------------------------------------------------
# 3) Solar gain function
# ------------------------------------------------------------

# solarpy.irradiance_on_plane(...) returns W/m   (i.e., J/s  m  ).
# We multiply by cross_section_area (m  ) to get W,
# then by 3600 to get J/h.
latitude = 52.4862
altitude = 0
vnorm = np.array([0, 0, -1])  # Horizontal plane (pointing up)

def Q_solar(t, cross_section_area):
    start_time = datetime(2022, 7, 19, 0, 0)
    current_time = start_time + timedelta(hours=int(t))
    irradiance = irradiance_on_plane(vnorm, altitude, current_time, latitude)  # [W/m  ]
    return float(irradiance * cross_section_area) * 3600 * transmittance

# ------------------------------------------------------------
# 4) U_a function
# ------------------------------------------------------------

def get_windspeed(t):
    idx = min(int(t), 23)
    return float(parser.get_data_from_column(idx, 'Wind Speed (miles per hour)'))

def U_a(t, heat_loss_area, u_value):
    k_wind = 0.15  # Empirical convection coefficient [W/(m   K) per m/s]
    wind_speed = get_windspeed(t) * 0.447  # Convert mph to m/s
    convective_U = k_wind * wind_speed * heat_loss_area * 3600  # Convert to J/(h  K)
    return (u_value * heat_loss_area * 3600) + convective_U


# ------------------------------------------------------------
# 5) ODE model
# ------------------------------------------------------------

# T_in is indoor temperature ( C ).
# t is time in hours.
# C_p is building heat capacity in [J/K].
# U_a is building conduction in [J/(h  K)].
# Q_solar(t, area) returns solar gain in [J/h].
#
# net energy flow [J/h] / heat capacity [J/K] => dT/dt [K/h].
def model(T_in, t, C_p, area, shadyness, u_value):
    conduction_loss = U_a(t, area, u_value) * (T_in - T_amb(t))   # [J/h]
    solar_gain = Q_solar(t, area) * shadyness           # [J/h]
    dTdt = (-conduction_loss + solar_gain) / C_p        # [K/h]
    return dTdt
# ------------------------------------------------------------
# 6) Solve
# ------------------------------------------------------------

T_in_0 = 20  # Initial indoor temperature ( C )
t_hours = np.linspace(0, 23, 24)  # 24 hours, in hours
T_in_1980 = odeint(model, T_in_0, t_hours, args=(Cp_1980, heat_loss_area_1980,
    shadyness_1980, Ua_1980))
```

```
T_in_1915 = odeint(model, T_in_0, t_hours, args=(Cp_1915, heat_loss_area_1915,
    shadyness_1915, Ua_1915))
T_in_2010 = odeint(model, T_in_0, t_hours, args=(Cp_2010, heat_loss_area_2010,
    shadyness_2010, Ua_2010))
T_in_1940 = odeint(model, T_in_0, t_hours, args=(Cp_1940, heat_loss_area_1940,
    shadyness_1940, Ua_1940))

T_amb_values = [T_amb(th) for th in t_hours]

# ------------------------------------------------------------
# 7) Plot results
# ------------------------------------------------------------

plt.figure(figsize=(10,6), dpi=500)
plt.plot(t_hours, T_in_1980, label='Indoor Temperature (Flat)', color='blue')
plt.plot(t_hours, T_in_1915, label='Indoor Temperature (Terraced House)', color='green')
plt.plot(t_hours, T_in_2010, label='Indoor Temperature (Maisonette)', color='red')
plt.plot(t_hours, T_in_1940, label='Indoor Temperature (Semi-detached House)', color='purple
    ')
plt.plot(t_hours, T_amb_values, label='Ambient Temperature', color='orange')
plt.xlabel('Time (hours)')
plt.ylabel('Temperature ( C )')
plt.title('Indoor Temperature During Heatwave')
plt.legend()
plt.grid(True)
plt.savefig("indoor_temperature_model.jpg", dpi=300)
```

```
#ode_sobol.py
    import numpy as np
from scipy.integrate import odeint
from solarpy import irradiance_on_plane
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
from SALib.analyze import sobol
from SALib.sample import saltelli
import data_parser

# ------------------------------------------------------------
# 1) Building & simulation data (using the 1980 flat as an example)
# ------------------------------------------------------------

transmittance = 0.7
parser = data_parser.load_data(data_parser.columns, "data/q1data.csv")
latitude = 52.4862
altitude = 0
vnorm = np.array([0, 0, -1])  # Horizontal plane (pointing up)

def T_amb(t):
    """Return the ambient temperature at hour t ( C )."""
    idx = min(int(t), 23)
    return float(parser.get_data_from_column(idx, 'Temperature (celsius)'))

def get_windspeed(t):
    """Return the wind speed at hour t (in m/s)."""
    idx = min(int(t), 23)
    return float(parser.get_data_from_column(idx, 'Wind Speed (miles per hour)'))

# ------------------------------------------------------------
# 2) Redefine functions to accept uncertain parameters
# ------------------------------------------------------------

def U_a_sobol(t, area, u_value):
    """
    Calculate the overall conduction coefficient U_a at time t.
    Combines a constant conduction term (scaled by u_value and area) with a convective term.
    """
    k_wind = 0.15  # Empirical convection coefficient [W/( m   K ) per m/s]
    wind_speed = get_windspeed(t) * 0.447  # Convert mph to m/s
```

```
    convective_U = k_wind * wind_speed * area * 3600  # [J/(h K)]
    return (u_value * area * 3600) + convective_U

def Q_solar_sobol(t, area, transmittance):
    """
    Calculate the solar gain at time t.
    The function computes irradiance on a horizontal plane (vnorm) and scales it by area, time, and transmittance.
    """
    start_time = datetime(2022, 7, 19, 0, 0)
    current_time = start_time + timedelta(hours=int(t))
    irradiance = irradiance_on_plane(vnorm, altitude, current_time, latitude)  # [W/m ]
    return float(irradiance * area) * 3600 * transmittance  # [J/h]

def model_sobol(T_in, t, Cp, area, shadyness, u_value):
    """ODE model for the indoor temperature."""
    conduction_loss = U_a_sobol(t, area, u_value) * (T_in - T_amb(t))   # [J/h]
    solar_gain = Q_solar_sobol(t, area, transmittance) * shadyness           # [J/h]
    dTdt = (-conduction_loss + solar_gain) / Cp        # [K/h]
    return dTdt

def simulate_indoor_temperature(params):
    """
    Runs the simulation for one set of parameters and returns the final indoor temperature.
    """
    Cp, area, shadyness, u_value = params
    T_in_0 = 20  # Initial indoor temperature ( C ) - same as original ODE
    t_hours = np.linspace(0, 23, 24)  # 24-hour period
    T_in = odeint(model_sobol, T_in_0, t_hours, args=(Cp, area, shadyness, u_value))
    return T_in[-1]  # Return temperature at the final time


problem = {
    'num_vars': 4,
    'names': ['Cp', 'area', 'shadyness', 'u_value'],
    'bounds': [
        [1e7, 5e7],    # Cp: Building heat capacity (J/K) - Adjusted bounds
        [50, 150],       # area: Heat loss / cross-sectional area (m  ) - Adjusted bounds
        [0.3, 0.9],     # shadyness: Solar gain modifier (dimensionless) - Adjusted bounds
        [0.5, 2.5]      # u_value: Conduction coefficient (W/m   K) - Adjusted bounds
    ]
}

# ------------------------------------------------------------
# 4) Generate samples and run the model
# ------------------------------------------------------------

param_values = saltelli.sample(problem, 128)

Y = np.zeros([param_values.shape[0]])

for i, params in enumerate(param_values):
    Y[i] = simulate_indoor_temperature(params)

# ------------------------------------------------------------
# 5) Perform Sobol sensitivity analysis
# ------------------------------------------------------------

Si = sobol.analyze(problem, Y, print_to_console=True)
print("Sobol Sensitivity Indices:")
print(Si)
```

```
#q2_data_parser.py
    import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

def project_population(target_year, should_plot=False):
```

```
    population_data = [
        (2022, 1157603),
        (2021, 1143285),
        (2020, 1140525),
        (2019, 1141816),
        (2018, 1141374),
        (2017, 1137123),
        (2016, 1128077),
        (2015, 1112950),
        (2014, 1101521),
        (2013, 1092190)
    ]

    years = np.array([year for year, _ in population_data]).reshape(-1, 1)
    populations = np.array([population for _, population in population_data])

    model = LinearRegression()
    model.fit(years, populations)

    extended_years = np.arange(years.min()-1, target_year + 1).reshape(-1, 1)
    predicted_populations = model.predict(extended_years)

    target_population = model.predict(np.array([[target_year]]))[0]

    if should_plot:
        plt.scatter(years, populations, color='blue', label='Actual Population')
        plt.plot(extended_years, predicted_populations, color='red', label='Linear
            Regression')
        plt.scatter([target_year], [target_population], color='green', label=f'Projected
            Population ({target_year})')
        plt.xlabel('Year')
        plt.ylabel('Population')
        plt.title('Population Over Years')
        plt.legend()
        plt.savefig('population_trend.png')

    return int(target_population)

def project_max_temperature(target_year, should_plot=False):
    temperature_data = [
        (2022, 37.2),
        (2021, 30),
        (2020, 33.9),
        (2019, 33.9),
        (2018, 31.1),
        (2017, 30),
        (2016, 31.1),
        (2015, 32.2),
        (2014, 27.8),
        (2013, 31.1)
    ]

    years = np.array([year for year, _ in temperature_data]).reshape(-1, 1)
    temperatures = np.array([temperature for _, temperature in temperature_data])

    model = LinearRegression()
    model.fit(years, temperatures)

    extended_years = np.arange(years.min(), target_year + 1).reshape(-1, 1)
    predicted_temperatures = model.predict(extended_years)
    target_temperature = model.predict(np.array([[target_year]]))[0]

    if should_plot:
        plt.scatter(years, temperatures, color='blue', label='Actual Temperature')
        plt.plot(extended_years, predicted_temperatures, color='red', label='Linear
            Regression')
        plt.scatter([target_year], [target_temperature], color='green', label=f'Projected
            Temperature ({target_year})')
```

```
71          plt.plot([years.min(), target_year], [predicted_temperatures[0], target_temperature
                ], color='red', linestyle='--')
72          plt.xlabel('Year')
73          plt.ylabel('Temperature␣( C )')
74          plt.title('Max␣Temperature␣Over␣Years')
75          plt.legend()
76          plt.savefig('temperature_trend.png')
77
78      return target_temperature
79
80  # project_max_temperature(2040, True)
```