Coupling a Reduced Order Building Energy Model to UrbanSim

Ralph T. Muehleisen[1], Joshua Bergerson[2]
[1]Energy Systems Division, Argonne National Laboratory, Argonne, IL, USA
[2]Global Security Sciences Division, Argonne National Laboratory, Argonne, IL, USA

Abstract

This paper describes the coupling of a real estate / urban planning simulation model, UrbanSim, to a reduced order building energy model, ISOmodel, for estimating energy use and greenhouse gas emissions for a large urban region and allow building energy use and associated costs to be included in urban growth projections. Projected real-estate information from UrbanSim, including building type, location, size, and number of occupants, is extracted from UrbanSim and used to generate inputs to the reduced order building energy model. ISOmodel is then used to estimate monthly gas and electric building energy use and associated carbon footprints in several end use categories. The scripts to couple the two tools consist primarily of Python code that makes extensive use of the Pandas data analysis library.

Introduction

Over the next several decades, several billion people are expected to migrate into cities, creating unprecedented increases in demand for food, water, energy, shelter, transportation, healthcare, education, and other services and infrastructure (World Bank 2017). These increases will be unsustainable without the accumulation of new knowledge to understand the relationships between these systems and development of new technologies and tools to better design our future urban centers, retrofit existing ones, and operate both more efficiently. Some of the key tools will be models that help decision makers understand the complex connections between people, the infrastructure, and the environment.

UrbanSim (Waddell et al., 2002) is a state-of-the-art platform for simulation of real estate markets for support of planning and analysis of urban developments. UrbanSim models the interactions of land use, transportation, the economy, and environment but currently does not include any simulation of building energy or its related carbon footprint and costs. As a result, the impacts of building energy use on real estate decisions, namely energy costs and energy related changes in real estate value, are not included in the simulation. Nor can the urban planners directly use UrbanSim simulations to assess expected changes in energy use and carbon footprint as they investigate changes in the real estate market.

Researchers at Argonne National Laboratory have developed a reduced order building energy model (BEM) based on the ISO 13790 and related standards, heretofore called the ISOmodel. Argonne's ISOmodel has been used for urban building studies (Guzowski et al. 2012), has been incorporated as an alternative energy engine for OpenStudio (Muehleisen et al. 2013), and is used in the Argonne Commercial Building Agent Model (CoBAM), a technology market adoption model (Zhao et al. 2011, Muehleisen et al. 2016). This energy model captures much of the basic physics of building energy use and runs extremely fast, allowing it to be used as building energy model in agent based simulations or in uncertainty and sensitivity analyses where the energy simulation must be run hundreds of thousands to hundreds of millions of times in a complete simulation. This BEM is available as both a part of the OpenStudio package (Guglielmetti et al. 2011) and as a standalone executable (ANL 2017a).

This paper describes the coupling of the Argonne ISOmodel to UrbanSim including the data extracted from UrbanSim for energy modeling, the methodology used for generating input files for the ISOmodel from the UrbanSim data, the running of the ISOmodel, and aggregation of data from the ISOmodel. An example of use of the code to analyse UrbanSim data from the city of San Francisco is shown.

UrbanSim

UrbanSim is an open source model system for analyzing urban development with an emphasis on modeling real estate markets. UrbanSim was originally written in Java more than two decades ago. It was modularized and implemented in Python about a decade ago making extensive use of several numerical libraries. Most recently it has been rewritten again to make use of the Pandas data analysis library (McKinney 2010) to help make the capability more widely accessible to urban planners and modelers (Waddell 2016). Pandas is an open source set of data structure and data analysis tools for Python (both Python V2.7 and V3.5) that has become one of the most popular set of tools for use in data science in Python.

https://doi.org/10.26868/25222708.2017.136

UrbanSim is a yearly simulation that utilizes six main modules that are run in sequence as shown in Figure 1 below.

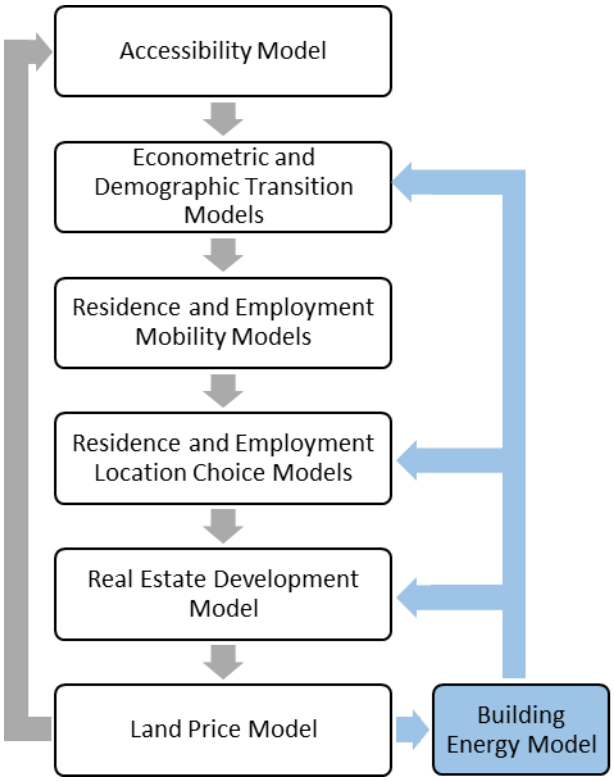

Figure 1: UrbanSim flow model (gray) augmented with an external building energy model (blue).

The *accessibility model* combines data from external transit models and land use data to estimate accessibility between locations within the region under study. It is used as an input to the other modules as well as any external transit models.

The *economic and demographic transition model* estimates the number of new households and jobs that will be added region in the study in the year.

The *residence and employment mobility model* simulates the decision of households and jobs to change location within the region of study in the year.

The *residential and employment location choice models* simulate the location choice decisions taken by the households and jobs that change.

The *real estate development model* simulates the actions of real estate developers to develop sections of the city including location and the type of development.

The *land price model* simulates changes in the real estate market, balancing real estate supply and demand.

At the end of the yearly simulation, UrbanSim has generated a new set of real estate and occupant data which are used as inputs along with other external data, for the next year of simulation.

Building energy use data from the ISOmodel, combined with localized fuel costs, can be used to estimate energy related building costs. Those costs, along with the raw building energy use can then be fed back into UrbanSim and can be used by the simulation as additional inputs within the various models.

Coupling UrbanSim to the ISOmodel

The ISOmodel is coupled to UrbanSim through a set of Python scripts which extract data from the UrbanSim data frames, create the necessary input files for the ISOmodel, run the ISOmodel, and import the results back into UrbanSim to add to the UrbanSim data frames. The scripts make extensive use of the Pandas Data Analysis library. The scripts are written in Python V2.7 for compatibility with UrbanSim which currently also uses Python V2.7. A flow diagram of the scripts is shown in Figure 2 below.

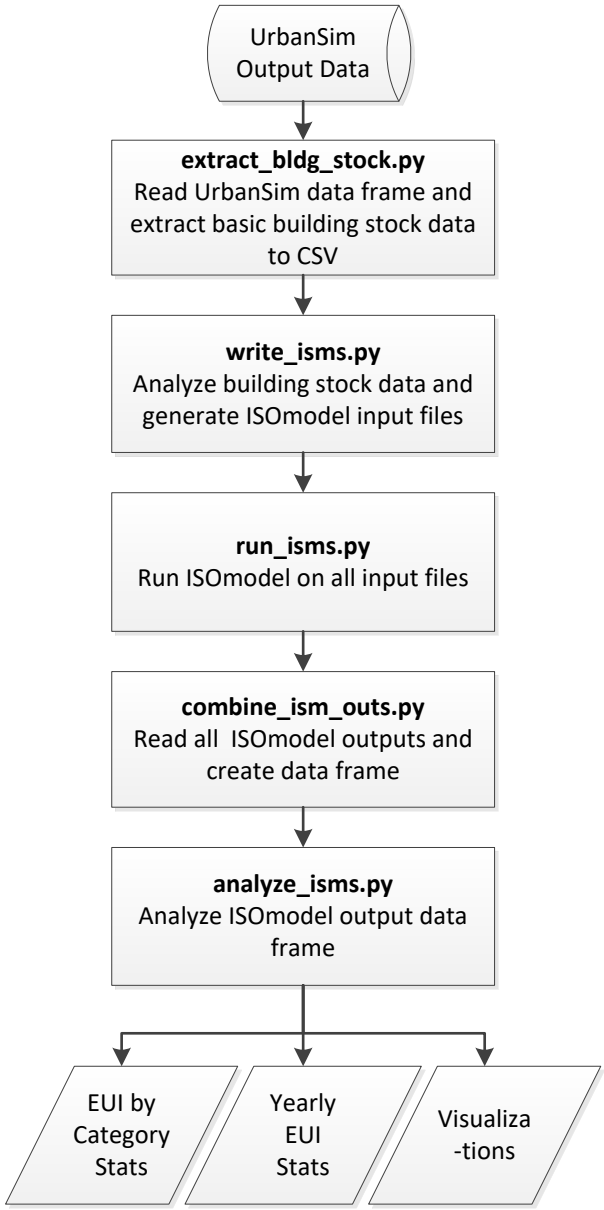

Figure 2: Flow diagram showing the order and basic operations of the major Python scripts.

Data Extraction

The first Python script developed for this project, *extract_bldg_stock.py*, is used to access and extract the building information from the UrbanSim HDF5 data file that houses most of the UrbanSim input data and run results. This script creates two data frames: one for storing all the buildings and a second for storing households listed in the UrbanSim HDF5 file.

The script then totals the number of people who are listed to be in each building and creates a new data frame that merges the information from the household and buildings data frames. This data frame is then saved to disk in a CSV format for conversion into ISOmodel BEM inputs. These data are saved in CSV instead of HDF5 format for ease of review and edit by the user using standard text editors or Excel. The data that are extract during this process include:

- building_id: identifier for individual buildings
- parcel_id: identifier for the building lot
- residential_units: number of residential units in building
- non_residential_units: number of non-residential units in building
- building_area: total floor area of the building in sqft
- stories: height of the building in stories
- building_type_id: numeric identifier of the overall building type (used for categorizing the building type and selecting default BEM inputs)
- year_built: year of building construction (used in choosing default BEM inputs)
- household_count: number of households residing in building
- num_people: total number of people in the building

Generating BEM Input Files

The second Python script, *write_isms.py*, analyzes the extracted building data table and generates input files for the BEM from the extracted data. For each building in the table, the Python script will generate and populate a default building energy model based upon the building type and year built. The mapping of building type and year of build to default BEM model input values is one of the most important steps in the process. Unfortunately, UrbanSim does not have a standardized way of mapping the actual building type to the number assigned to building_type_id as stored in the HDF5 file. This mapping is left to the choice of the UrbanSim modeler. This means that users of the BEM scripts described here must carefully confirm the mapping of the actual building type to the building_type_id values used in UrbanSim to ensure that the correct BEM default values are correctly selected.

Correctly identifying the building type, vintage, and areas when selecting the correct BEM default values is most important when modeling existing building stock and making policy decisions related to renovation of that existing building stock, because misidentification could lead to a very inaccurate energy model.

For recent construction and new buildings to be constructed a good set of default values could be derived from modeling a minimum code compliant building for that city. A small fraction of the new buildings could be modeled as higher performance buildings with more energy efficient envelope features and HVAC design, based on statistics of what fraction of buildings obtain performance ratings such as LEED, Energy Star, or Passive House.

For existing buildings, creating a set of building models that accurately model the building energy use is a daunting task for a large city, but methodologies have been developed to make good use of a variety of data sources and ease the data collection burden including CHEERI (Guzowski et al., 2012), CityBES (Hong et al. 2016) and UBEM (Cerezo Davila et al. 2016). These methods also assign default building models and values based upon the size, type, and vintage of the building

Table 1: Default ISOmodel BEM parameters set by the building type and vintage selection

Element	Parameters
Wall and Roofs	Thermal Resistance, Thermal Mass, Solar Absorptivity, Thermal Emissivity
Windows and Skylights	Assembly "U" Value, Solar Heat Gain Coefficient, Shading Coefficient, Window to Wall Ratio, Skylight to Roof Ratio
Lighting	Interior Lighting Power Density, Exterior Total Lighting Power, Lighting Schedules, Lighting Controls
Interior Loads	Gas and Electric Interior Load Power Densities, Heat Gain Per Person, Interior Load Schedules
HVAC	Thermostat Set Points and Schedules, HVAC System Type, Fuel, and Efficiency, Ventilation Rate (per person and floor area), Hot Water Heater Fuel and Efficiency, Fan and Pump Efficiency, HVAC Control
Other	Window to Wall Ratio, Skylight to Roof Ratio, Enclosure Aspect Ratio, Building Floor-to-Floor Height, Hot Water Use Per Person, Infiltration Rate

The default building model selected is used to set the upper and lower limits and mode of a triangle distribution for many of the BEM inputs as shown in Table 1 above. The floor area and building height (in stories) data extracted from UrbanSim are used with the default aspect, window-to-wall, and skylight-to-roof, ratio to generate sizes of walls, windows, and roofs. The floor area data and occupancy extracted from UrbanSim are used with default

parameters to generate BEM inputs related to occupant density, ventilation rate, and hot water demand. HVAC system and lighting and interior load inputs are determined by the building type.

For each building extracted from the UrbanSim simulation, random input values are generated from the distribution ranges as described above to create unique ISOmodel building energy model input files for each building. These ISOmodel input files are simple text files and can be easily read by and edited by the user.

The generated input files are stored on disk in a user defined folder location. The current scripts utilize a single weather data model. If additional weather data are available, the script could be modified to select the weather data for the closest weather station. Such a modification would require a change to the data extraction script in order to extract the building parcel location and select the most appropriate weather data file.

Running the BEM Model

A third Python script, *run_isms.py*, is used to look at the contents of the ISOmodel input file folder and run the standalone ISOmodel program for each of the input files and selected weather files, generating an associated energy use output CSV file placed in a user selected location.

The energy use output file is a CSV file consisting of 12 rows and 12 columns (not counting the header/label rows and columns) with the energy use as an energy use intensity (EUI) in kWh/m^2. Each row represents a month and the columns have estimates of the following energy end use:

- Electricity used for Heating and Cooling
- Electricity used for Interior and Exterior lights
- Electricity used for Electric Fans and Pumps
- Electricity used by interior electrical equipment
- Electricity used for domestic hot water heat.
- Gas used for Heating and Cooling
- Gas used by miscellaneous interior equipment
- Gas used for domestic hot water heating

Post-processing the BEM outputs

A fourth script, *combine_ism_outs.py*, loads the output data from the individual ISOmodel runs into Python and used to create a data frame with the whole building and end use monthly EUIs for each building. This data frame is saved to a new HDF5 file.

A fifth script, *analyze_ism_outs.py*, will read the data frame into Python, do some quality control checking, aggregate monthly data into yearly total EUIs for each building, and analyze the results for statistical information including mean, median, standard deviation, and quantile percentages.

Quality control checking consists of looking for data with abnormally high or low yearly EUI which indicate errors in the input file generation process created by incorrect extraction of data from UrbanSim or incorrect data in the UrbanSim model. As an example, during development we found a residential building with an EUI of 2000 kWh/m^2. In sorting through the input data, we found that the building was a multistory residential building with 10 households but with an area of only 95 sqft. Further investigation found that this was a problem with the actual UrbanSim data and not the extraction by our scripts. These are the sort of data that must be double checked by the user.

This script could be easily augmented to provide additional analysis such as conversion of site-to-source energy use and the associated carbon footprint. Because of the regional variations in electricity generation sources and transmission and distribution system efficiencies, local, or at least regional source-to-site and greenhouse gas emission conversion factors would need to be provided by the modeler.

Pandas Data Analysis Library

The Python code used in the project makes extensive use of Pandas data frames and the data frame operators defined in Pandas. This library has great support for a variety of common data types and operations common to statistical analysis of large data sets. The routines are fast, handle missing data well, and handle both two and three-dimensional data sets. Code written using Pandas can be extremely compact, easy to read, and easy to maintain.

For example, the code required to generate a table of quantile statistics for a data series of building total yearly EUI named bldg_yr_eui would look like:

```
levels = [0.05, 0.25,0.05, 0.75 ,0.95]
y = bldg_yr_eui.quantile(levels)
```

In addition to statistical functions, Pandas makes use of the Matplotlib (Hunter 2007) to provide easy access to many visualization functions. The code required to generate a probability normalized histogram plot of the same bldg_yr_eui discussed above would look like:

```
bldg_yr_eui.plot.hist(normed=True)
```

The use of Pandas thus creates a set of Python scripts that are easy to read, maintain, and modify.

Speed of Model Execution

It is important that the speed of the building energy modeling be sufficiently fast that urban planners and policy makers can adjust and run scenarios by hand or utilize an optimization “wrapper” that will minimize or maximize some model output quantity. This need for speed precludes the use of a detailed BEM such as DOE2 or EnergyPlus unless the modeling was done using cloud computing or high performance computing (HPC). Even then, the computing and software system would need to be set up very carefully in order to ensure that data transfer to/from the cloud or HPC was fast enough and enough processing power was utilized that the turnaround time for the simulation of hundreds of thousands of BEM was reasonable. This is one of the reasons for using the reduced order ISOmodel in the coupling to UrbanSim. The ISOmodel runs extremely fast, on the 10 milliseconds

for each building model. The largest fraction of that time currently binvolves reading the weather file and BEM input file from disk and writing output data to disk in readable CSV format.

To further improve the speed for this particular integration the C++ ISOmodel could be modified to utilize a fast binary file format such as HDF5 (already used natively by UrbanSim) or Feathers (McKinney 2016), a new and very fast data frame storage format designed for interchange between Python, R, and C++. Then the entire set of BEM inputs could be passed to the ISOmodel as a single binary file and the ISOmodel would read it and the weather data file only once, loop through the entire array of inputs, and generate a single binary output file with all the output data.

Model Validation

One of the most important aspects of developing such a modeling mechanism is validation of the model. When using UrbanSim to generate predictions of future real-estate development, there are no data to use for calibration. However, UrbanSim does use some existing data about a city as the starting point for the future predictions and those existing data can be analyzed as one type of validation of the model.

The two main components of this exercise have been previously validated. UrbanSim validation can be found in Waddell et al. 2002. Validation of the ISOmodel through comparison to EnergyPlus can be found in Guzowski et al. 2014. Because there is no interaction between the models yet, (i.e. the energy use computed with the ISOmodel is not yet fed back into UrbanSim as an input that affects building prices), there was no compelling need for validation of the combined model. Validation of the combined model is planned for the future using some of the building energy data used by the aforementioned CityBES project along with one of the more detailed San Francisco UrbanSim models.

Example: San Francisco

The UrbanSim developers have an example data set developed from real data for the city of San Francisco. These data are available for download from the UrbanSim GitHub repository. This is not a complete set of the San Francisco building stock and data have been anonymized to avoid privacy issues and thus these data are not appropriate for a validation. However, the data are a fair statistical representation of the building stock of San Francisco fairly well and can be used to illustrate the coupling of the models.

The data set includes over 113,000 individual buildings, most of which are residential or primarily residential. The data set defines 14 unique building types: three residential, two office, one hotel, one school, three industrial, two retail, one mixed residential, and one mixed office, but details about the individual types are lacking, so the exact mapping of a building type in UrbanSim to a specific building type and default set of inputs (i.e. a specific reference building) has great uncertainty. As a result, for this example, the authors have used a set of default data from the DOE reference building set rather than a set of reference buildings specifically derived for San Francisco because the purpose of the exercise was to show the utility of the integration rather than generate true actionable information for an urban planner.

Once the data generated by UrbanSim or the ISOmodel have been read into Pandas data frames in Python, manipulation of those data is fairly simple. The data frame generated by the ISOmodel includes the set of monthly EUI for all the end use categories for all 113,000 buildings. Using Pandas, the total yearly EUI for each building, summed over all categories, can be obtained with a single command.

An example of the statistical information that is easily obtained from the simulation is shown in Table 2. Simple statistics including mean, standard deviation, min, and max in additional to quartile statistics are obtained with the two Pandas data frame commands mentioned earlier.

Table 2: Statistics generated by project scripts from the UrbanSim San Francisco example data.

Statistic	EUI (kWh/m^2)
Mean	304
Standard Deviation	92
Min	72
5% Quartile	175
25% Quartile	242
50% Quartile	294
75% Quartile	354
95% Quartile	465
Max	998

Another typical analysis a researcher or planner may want to do is to generate a histogram of the building energy use intensities (EUIs). Again, using Pandas, a probability normalized histogram can be generated and plotted like the one shown in Figure 3 in a single line of Python code as described earlier.

When the data frame containing the individual building EUIs is read back into UrbanSim and added to the existing data tables of the program, the 3D visualization capabilities that are being developed in conjunction with UrbanSim will be able to be used for better visualization of building energy data. Figure 4 below is an example of the sort of false color plot that could be generated from UrbanSim. Such a plot could be generated where the color is related to the EUI or total energy use of the building. The color in Figure 4 is related to the year of construction.

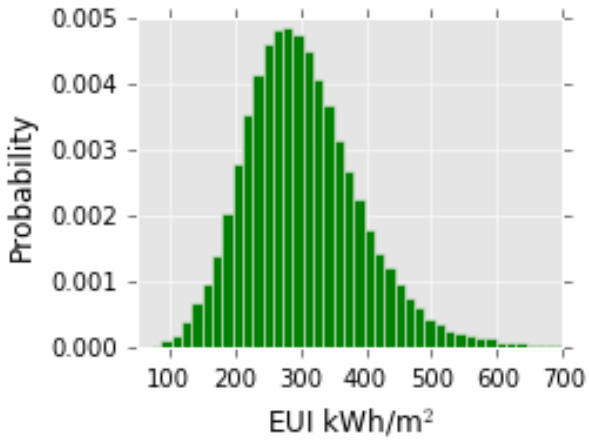

Figure 3: Probability histogram from the output of the San Francisco example data supplied with UrbanSim.

As the UrbanSim simulation is run and the data set evolves, the change in the total city EUI, or categorical EUI or even building-by-building EUI could be computed.

Figure 4: False color plot showing projection of building related data on a 3D map of San Francisco.

Future Work

The Python scripts for extracting building data, generating BEM input files, running the BEM, reading the BEM outputs and doing basic analysis will soon be released as an open source software package on the Argonne National Laboratory GitHub site (ANL 2017b). But, there is still much work to be done to further integrate the ISOmodel with UrbanSim. Perhaps the most important follow-on work is to make the modifications to UrbanSim to utilize the energy use data and the associated energy costs in the decision logic of UrbanSim, particularly the Residence and Employment Location Module and the Real Estate Development Modules.

The authors plan on modifying the ISOmodel C++ code to accept an energy model passed as a memory object from Python and to return computation results as a memory object to Python to speed the simulations significantly.

As the 3D visualization capabilities of UrbanSim are extended, another important future extension is to ensure that the data generated by the BEM are added back to UrbanSim in such a manner as to make it accessible by the UrbanSim 3D visualization modules. A third important follow-on project would be to implement data exchange between UrbanSim and the ISOmodel using data frames and a fast file format such as HDF5 or Feather as described earlier rather than individual input and output files utilizing CSV format.

Conclusion

This paper has discussed the first steps at integration of the Argonne ISOmodel, a reduced order building energy model, with the UrbanSim urban dynamic modeling software. UrbanSim is used to generate information about the changing building stock including locations, types, and occupancy of buildings within a city. Python scripts are used to extract the useful building information from UrbanSim and generate inputs the ISOmodel BEM and run the energy model on a building-by-building basis. The resulting BEM outputs are saved and read back into Python data frames where the data can be aggregated and statistically analyzed and put back into UrbanSim for further use by UrbanSim.

Acknowledgements

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government

The authors also wish to thank the developers of UrbanSim and Pandas for their outstanding contributions to the research community and for their ongoing development and free release of UrbanSim and Pandas.

References

ANL. 2017a. "Argonne National Laboratory GitHub UrbanSim-Isomodel Repository", Accessed May 9. https://github.com/Argonne-National-Laboratory/ISOmodel

ANL. 2017b. "Argonne National Laboratory GitHub UrbanSim-Isomodel Repository", Accessed May 9. https://github.com/Argonne-National-Laboratory/urbansim-isomodel

Cerezo Davila, Carlos, Christoph F. Reinhart, and Jamie L. Bemis. 2016. "Modeling Boston: A Workflow for the Efficient Generation and Maintenance of Urban Building Energy Models from Existing Geospatial Datasets." *Energy* 117, Part 1 (December): 237–50.

Chu, S. and A. Majumdar. 2012. Opportunities and challenges for a sustainable energy future. *Nature 488*, 294–303.

Guglielmetti, Rob, Dan Macumber, and Nicholas Long. 2011. "OpenStudio: An Open Source Integrated

Analysis Platform." In *Proceedings of the 12th Conference of International Building Performance Simulation Association*.

Guzowski, Leah B, D. J Graziano, Y. Heo, and R. T Muehleisen. 2012. "Testing a Streamlined Project Evaluation Tool for Risk-Conscious Decision Making: The Chicago Loop Energy Efficiency Retrofit Initiative." In 2012 ACEEE Summer Study on Energy Efficiency in Buildings, 139–51. ACEEE.

Guzowski, Leah B, Ralph T Muehleisen, Yeonsook Heo, and Diane J. Graziano. 2014. "Comparative Analysis for the Chicago Energy Retrofit Project." ANL Report: ANL/DIS-14/2. Argonne National Laboratory.

Hong, Tianzhen, Yixing Chen, Sang Hoon Lee, and Mary Ann Piette. 2016. "CityBES: A Web-Based Platform to Support City-Scale Building Energy Efficiency." In *5th International Workshop on Urban Computation (UrbComp 2016)*.

Hunter, J. D. 2007. "Matplotlib: A 2D Graphics Environment." *Computing in Science Engineering* 9 (3): 90–95.

McKinney, Wes. 2010. "Data Structures for Statistical Computing in Python." In *Proceedings of the 9th Python in Science Conference*, 445:51–56.

McKinney, Wes. 2016. "Feather: fast, interoperable binary data frame storage for Python, R, and more powered by Apache Arrow". Last accessed 14-Dec-2016. https://github.com/wesm/feather .

Muehleisen, Ralph T, Brian Craig, Daniel Macumber, Elaine Hale, and Jason Turner. 2014. "Integration of the CEN/ISO Monthly Building Energy Model into OpenStudio." In ACEEE Summer Study on Energy Efficiency in Buildings. 247-259. ACEEE.

Muehleisen, Ralph T., Joshua Bergerson, Nicholson Collier, Diane J. Graziano, and Eric Tatara. 2016. "Agent Based Technology Adoption Model for Program Planning and Design." In *ACEEE Summer Study on Energy Efficiency in Buildings 2016*.

Waddell, Paul., A. Borning, M. Noth, N. Freier, M. Becke, and G. Ulfarsson. 2002. "UrbanSim: A Simulation System for Land Use and Transportation." Networks and Spatial Economics 3 (43–67).

Waddell, Paul. A. 2016. "UrbanSim Readme". Last Accessed 14-Dec-2016. https://github.com/UDST/urbansim/blob/master/README.rst

World Bank. 2017. "Urban Development Overview." Accessed May 1. http://www.worldbank.org/en/topic/urbandevelopment/overview

Zhao, Fei, Ignacio J. Martinez-Moyano, and Godfried Augenbroe. 2011. "Agent-Based Modeling of Commercial Building Stocks for Policy Support." In Building Simulation 2011, 2385–92. IBPSA.