Modeling Integrated Community Energy and Harvesting Systems from Databases using OpenModelica

Data Transfer from MySQL Databases into OpenModelica Models

James LeMoine, Vick Lakhian, Jim Cotton

Database information

MIES

Integrated Community Energy Systems – A Whole Systems Approach

Thermal Energy Generation | Electrical Generation | Thermal Energy Storage | Electrical Storage

THE VISION EMC Integrated Community Energy and Harvesting System

ENERGY MANAGEMENT CENTRE

McMaster University ENGINEERING

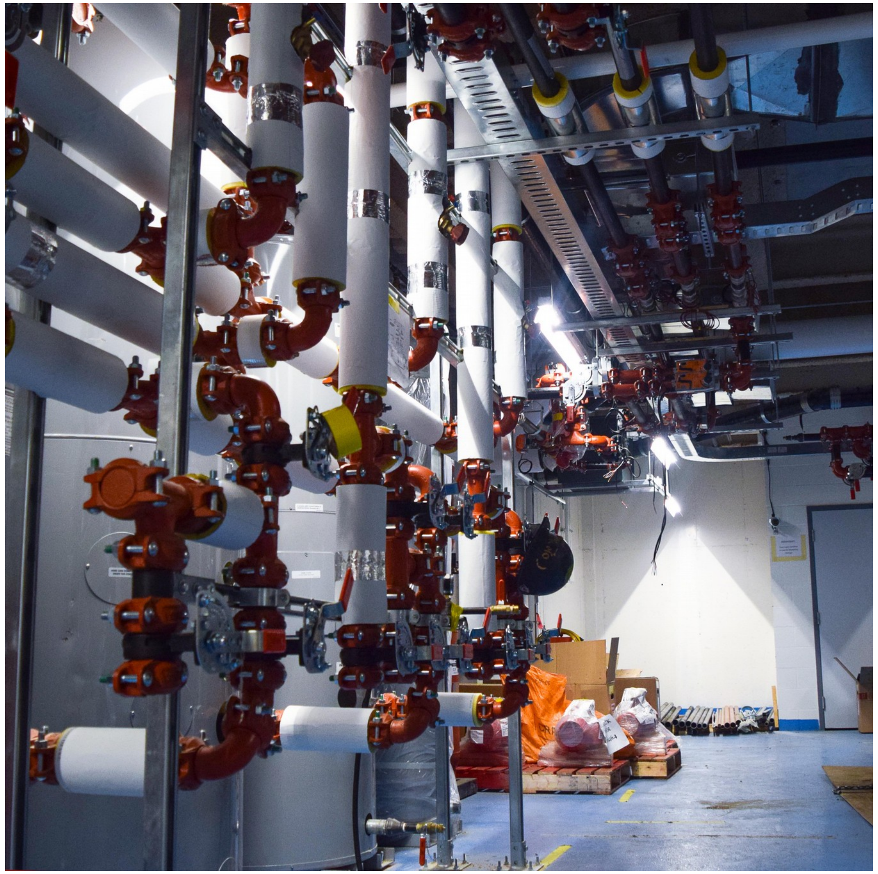
- Energy Transfer Station
- Community Buildings - Arena, Pool, Community Centre
- Electricity Generation
- Electricity Storage
- Thermal Energy Generation
- Thermal Energy Storage
- Underground Geothermal Seasonal Energy Storage

The EMC will plug into various components of the community. As well as those shown at right, other sample building types can include:

Grocery Stores, Box Stores

Apartments, Condos, Restaurants

ICE Harvest Facility



Overview of project

MySQL®

OpenModelica

python™

Overview of project

Python connects to and requests information stored in MySQL database

Overview of project

MySQL sends data requested and is stored in Python

Overview of project

Python sends parameters to a Modelica model

Overview of project

MIES

MySQL

OpenModelica

python™

Modelica simulates and returns the simulation results back to python

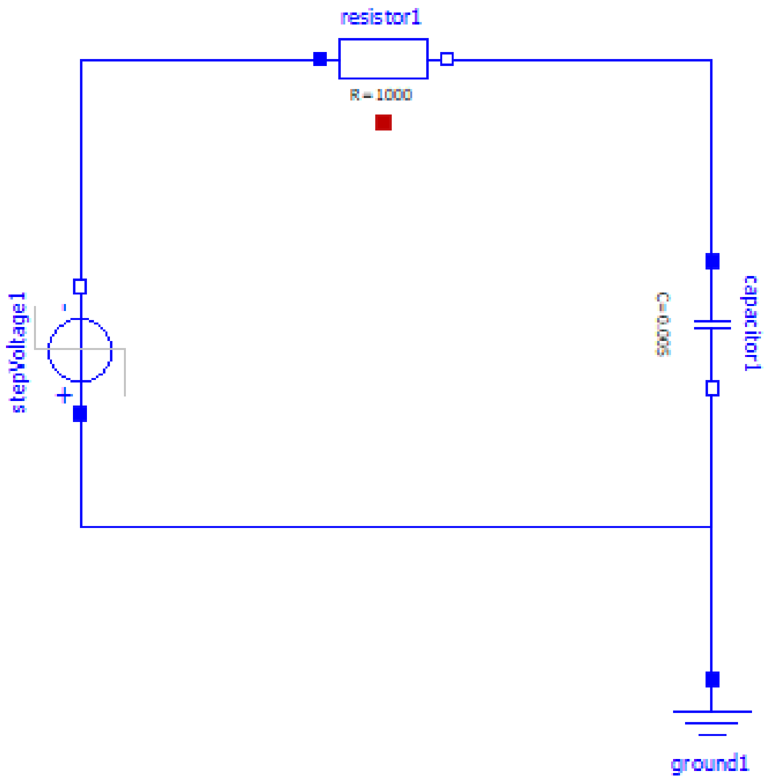
Overview of project

MIES

- The data then can be arranged with other python libraries:
  - MySQL
  - Excel
  - Text file
- Can be saved in graphical form:
  - Matplotlib
  - Any other Python graphics library

Modelica Model

MIES



- Simple RC circuit connected to a step voltage was used
- Any model can replace this simple model with a different database of parameters easily

Database information

	Voltage	Resistance	Capacitance
Case 1	120	10	0.005
Case 2	120	100	0.005
Case 3	120	300	0.005

- A MySQL database with the rows above is on the server that will be accessed
- These cases will show the difference that resistance has in the model

Output software

matplotlib

- Using matplotlib to show results graphically
- To display:
  - Voltage across the resistor vs time
  - Current across the resistor vs time

Python to MySQL data transfer

```
import mysql.connector

## Connecting to MySQL Server
cnx = mysql.connector.connect( user='User' , password='Password' ,
                               host='IpAdress' , database='Database')

## Creating a new cursor object
cursor = cnx.cursor()

## Defining the query
query = ("SELECT V, R, C FROM Circuits"
         )

## Executing the query and readying the server to send the data
cursor.execute(query)

## Taking the data from the database, test is now an array with rows of output data
testVariables = []

for (V, R, C) in cursor:
    testVariables = testVariables + [(V, R, C)]

## Closing the cursor and connection
cursor.close()
cnx.close()
```

Connecting to your database

```
import mysql.connector

## Connecting to MySQL Server
cnx = mysql.connector.connect( user='username' , password='password' ,
                               host='hostIPaddress' , database='database')
```

- Accesses a user with reading privileges to the database
- Connects to the server holding the test parameters
- Specifies the database where the tables of test parameters are held

Executing a search of the database

```
## Creating a new cursor object
cursor = cnx.cursor()

## Defining the query
query = ("SELECT V, R, C FROM Circuits"
         )

## Executing the query and readying the server to send the data
cursor.execute(query)
```

- Creates a cursor object that will perform the query and hold the results
- Defines the query to be sent to the database
  - This query can use expressions to specify the results desired
- Executes the query and prepares the database to send the results

Taking the parameters out of the database

```
## Taking the data from the database, test is now an array with rows of output data
testVariables = []

for (V, R, C) in cursor:
    testVariables = testVariables + [(V, R, C)]

## Closing the cursor and connection
cursor.close()
cnx.close()
```

- The parameters are then stored in variable
- The MySQL connection is then closed

Python and OpenModelica data transfer

MIES

```
from OMPython import ModelicaSystem
import matplotlib.pyplot as plt

mod = ModelicaSystem("C:/Users/user/Desktop/JamesLemoine/Modelica Models/RC.mo" , "RC" )

runTime = 20

listOfxVar = ['Voltage','Current']
listOfTitles = ['Voltage vs. Time', 'Current vs. Time']

fig, axs = plt.subplots(2,3)
fig.subplots_adjust(hspace=1)

for i in range(len(testVariables)):
    mod.setParameters( **{"stepVoltage1.V" : testVariables[i][0],
                          "resistor1.R" : testVariables[i][1],
                          "capacitor1.C" : testVariables[i][2] } )

    mod.setSimulationOptions(stopTime = runTime)

    mod.simulate()


    resistorV = mod.getSolutions('resistor1.n.v')
    resistorI = mod.getSolutions('resistor1.i')
    time = mod.getSolutions('time')

    column = [ resistorV , resistorI ]


    for j in range(2):
        xVar = column[j]
        axs[j][i].plot( time , xVar)
        axs[j][i].grid(True)
        axs[j][i].set_xlabel('Time (sec)')
        axs[j][i].set_ylabel( listOfxVar[j] )
        axs[j][i].set_title( listOfTitles[j] + " Case " + str(i+1) )

plt.show()
```

Loading the model

```
from OMPython import ModelicaSystem

mod = ModelicaSystem("C:/Users/user/Desktop/JamesLemoine/Modelica Models/RC.mo" , "RC" )

runTime = 20
```

- The specific model can be loaded by giving the directory it was saved and the model name
- This also loads the Modelica Standard library
- Run time is stored as 20 seconds
  - This can be done later to change the run time of each set of parameters

Preparing for outputting data

```
listOfxVar = ['Voltage (V)','Current (A)']
listOfTitles = ['Voltage vs. Time', 'Current vs. Time']

fig, axs = plt.subplots(2,3)
fig.subplots_adjust(hspace=1)
```

- Setting up for the final data manipulation
- Setting up the axis and chart titles
- Making subplots to show all graphs on one figure

Simulating model

```
for i in range(len(testVariables)):
    mod.setParameters( **{"stepVoltage1.V" : testVariables[i][0],
                          "resistor1.R" : testVariables[i][1],
                          "capacitor1.C" : testVariables[i][2] } )

    mod.setSimulationOptions(stopTime = runTime)

    mod.simulate()
```

- Inputting the parameters saved in the array into the Modelica model
- Changing the runtime to the time set before
- Simulating each model

Retrieving solutions

MIES

* Still under previous for loop

```
resistorV = mod.getSolutions('resistor1.n.v')
resistorI = mod.getSolutions('resistor1.i')
time = mod.getSolutions('time')

column = [ resistorV , resistorI ]
```

- After each model is simulated the solutions are taken out and an array is made
- From this point the data may be output into a spreadsheet

Plotting solutions

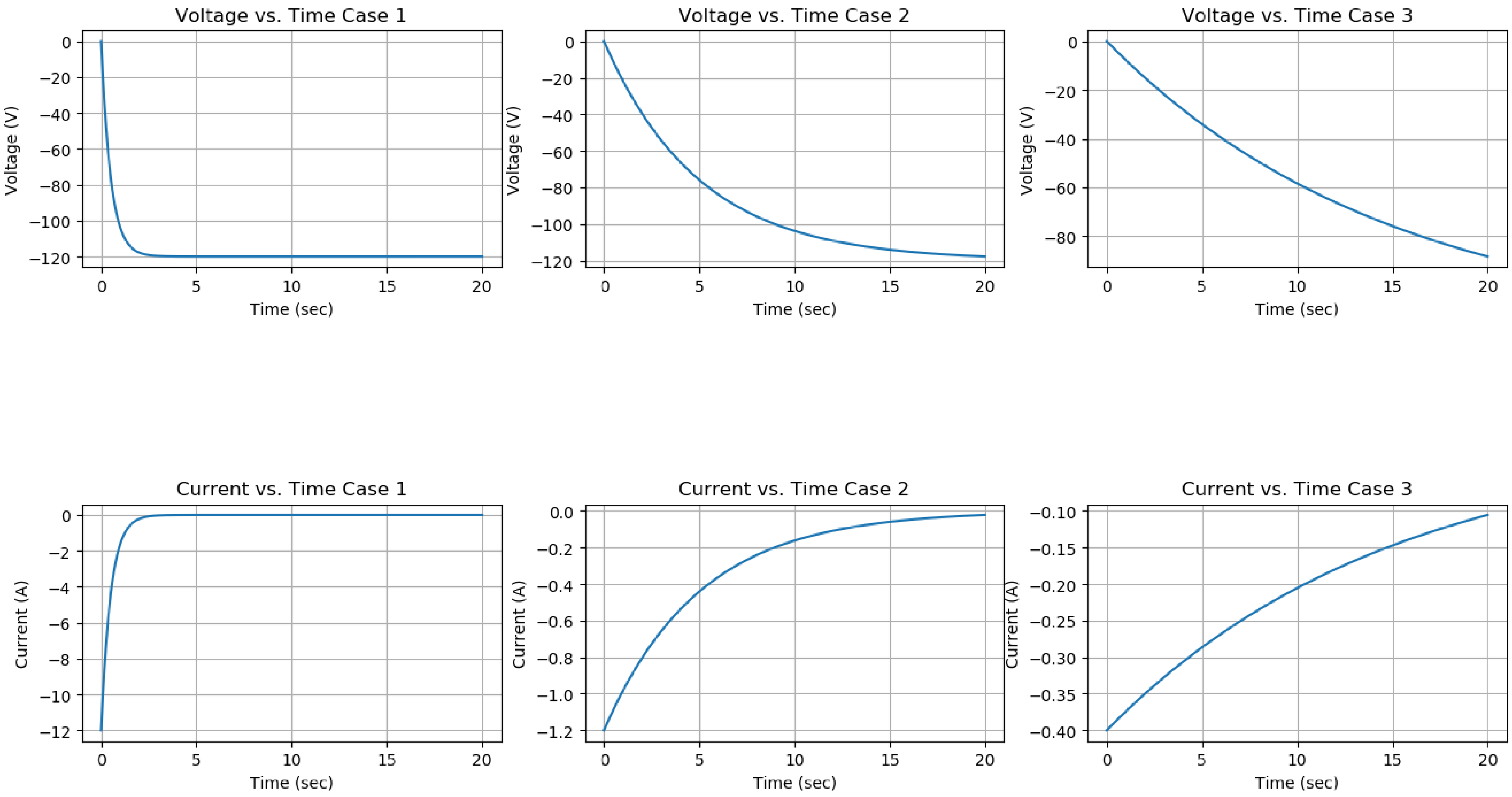
* Still under the for loop

```
    for j in range(2):
        xVar = column[j]
        axs[j][i].plot( time , xVar)
        axs[j][i].grid(True)
        axs[j][i].set_xlabel('Time (sec)')
        axs[j][i].set_ylabel( listOfxVar[j] )
        axs[j][i].set_title( listOfTitles[j] + " Case " + str(i+1) )

plt.show()
```

- Here the solutions are plotted
- A column of the subplot is given for each case
- Runs two times per simulation to make both subplots that are desired
- After both loops the final figure is shown

Output graphical window

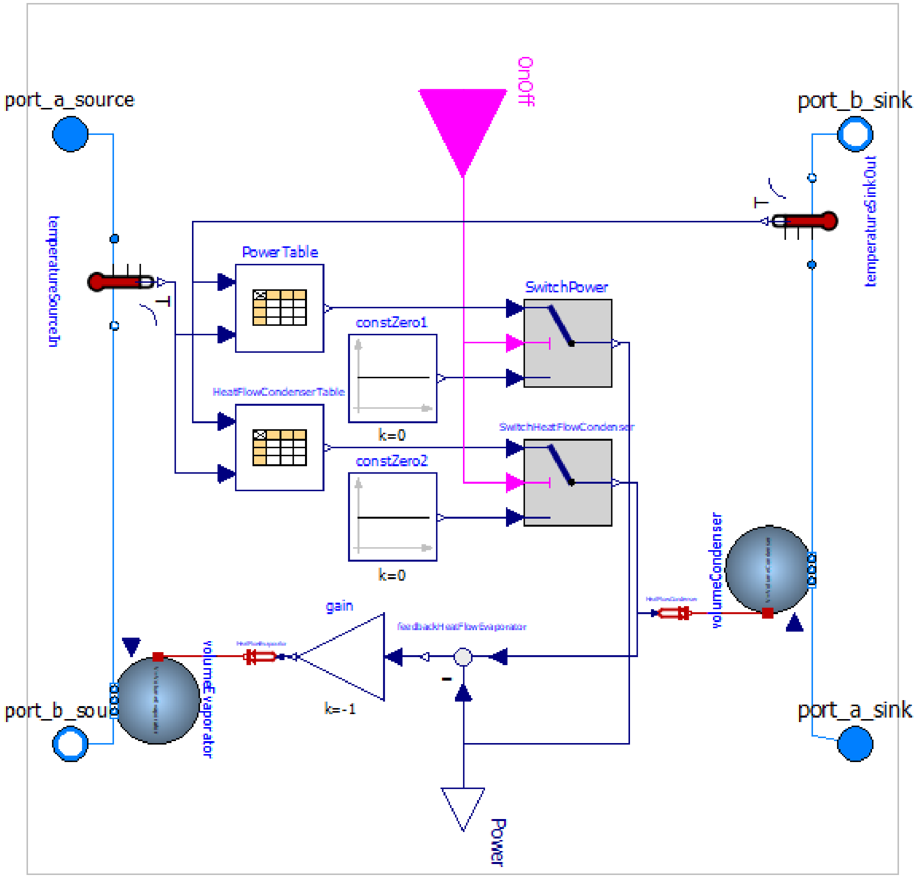


Future goals

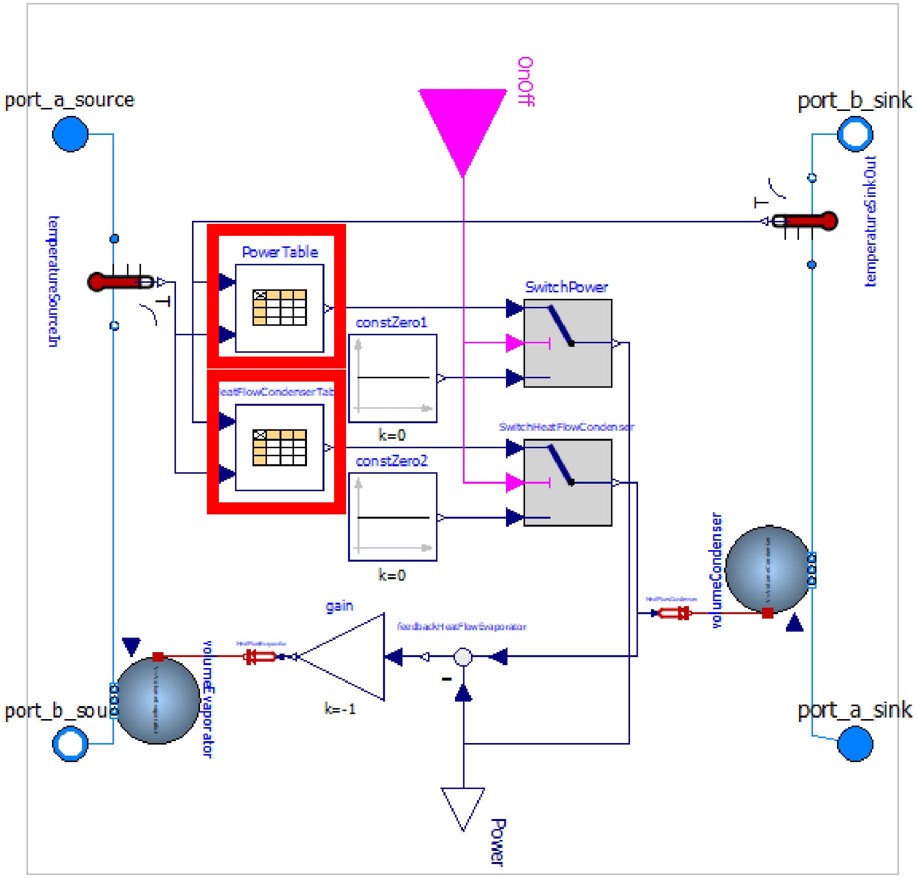
MIES

- The project is to be adapted for models of HVAC machinery
- Efficiency tables given from the manufacturers will be transferred into the Modelica models

Simple heat pump model from AixLib



Simple heat pump model from AixLib



Thank you