Pore Network Modeling Software Interface Development

Harold Day, Jeff T. Gostick*

Department of Chemical Engineering, McGill University, Montréal, QC, H3A 2B2, Canada
*Supervising Professor | Phone: (514) 398-4301 | Fax: (514) 398-6678 | E-mail: jeff.gostick@mcgill.ca

INTRODUCTION

Polymer electrolyte membrane fuel cells (PEMFCs) **[Fig 1.]** are a promising alternative to combustion engines for automotive applications due to their ability to provide high power for long durations with short recharge times. Current efforts in the field are directed towards increasing their power density to lower costs and enhance commercial viability.

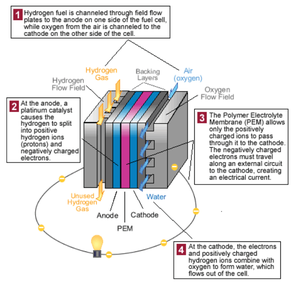

FIG 1. Basic diagram of main elements involved in the operation of PEMFCs.
Source: www.fueleconomy.gov/feg/fc_pics/fuel_cell_still.gif

This project is a subset of research aiming to develop more accurate pore network models to aid in the creation sophisticated designs that overcome existing issues centered around water retention in the porous electrode: If the levels are too high they inhibit transport in the gas layer, if they are too low the membrane becomes desiccated when operating at high temperatures.

One of the ways in which the pore materials for the fuel cell electrode are characterized is via air-water capillary pressure curves. Researchers can experimentally obtain this data via a Mercury Intrusion Porosimetry (MIP) setup. **[Fig 2.]**

Conventional modeling approaches based on parallel capillary tubes fail to reflect the properties and behaviour of materials used, requiring unrealistic contact angle hysteresis (over 80°) to explain observations. Pore Network Models (PNMs), from the most basic to the most sophisticated, generally fare much better when subjected to invasion simulation.

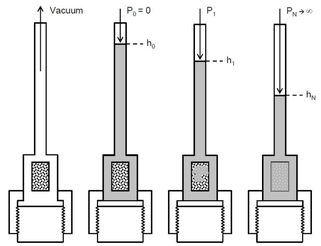

FIG 2. Diagram illustrating the essentials of a Mercury Intrusion Porosimetry setup.
Source: Jeff T. Gostick

OBJECTIVES

This project is an effort to verify the viability of an open-source, operating-system-independent programming language as a main platform for developing the modeling framework and deploying it for academic and industrial use.

- Port existing PNM Matlab code to Python
- Review and augment algorithms as necessary
- Design and implement a graphical user interface (GUI)
- Emphasise making code accessible to non-expert users and produce documentation geared to augmenting code in the future

PORE NETWORK MODELING

Due to the inexistence of relevant Python code, we started by porting over the most basic version of the PNM generation algorithm:

1. Generate a matrix of random numbers following a Weibull distribution to represent pore size
2. Obtain information about the pore structure from assumptions made about pore shape
3. Obtain description of structure in terms of threshold pressures according to the capillary tube pressure (Washburn) formula
4. Perform invasion simulation across predetermined pressure range using image processing algorithm **[Fig 3.]**
   a. Determine which cells are accessible at each injection pressure
   b. Label cell clusters according to proximity
   c. Discard clusters not connected to injection layer
   d. Quantify degree of invasion in terms of volume
   e. Output simulated capillary pressure plot

Image analysis algorithms have been extensively developed and optimized in Python libraries. We used `numpy.ndarray.nd_image` which means that the ported code is elegant, very efficient and fast.

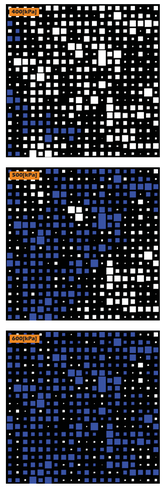

FIG 3. Hinton diagrams displaying 2D invasion simulation of 20x20 model in the 400-600(kPa) range.

MATHEMATICAL OPTIMIZATION

An application **[Fig 4.]** was developed to allow users to quickly and easily manage and visualize sets of data, rapidly fit generic Van Genuchten and Brooks-Corey curves, and recover optimal parameters.

Simultaneous Perturbation Stochastic Approximation (SPSA) **[Fig 5.]** is the method best suited to matching modeling parameters to experimental data. The current implementation of the algorithm can easily recover generating parameters of artificial data from basis-less initial guesses thanks to performance-improving tweaks discovered in relevant literature.

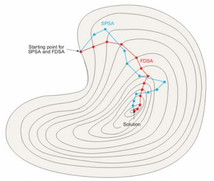

Figure 5. Example of relative search paths for SPSA and FDSA in $p = 2$ problem. Deviations of SPSA from FDSA average out in reaching a solution in the same number of iterations; FDSA, nearly follows the gradient descent path (perpendicular to level curves) in the long run.
Source: J. Spall, "An Overview of the Simultaneous Perturbation Method for Efficient Optimization", Johns Hopkins APL Technical Digest, Vol. 19, No. 4 (1998)

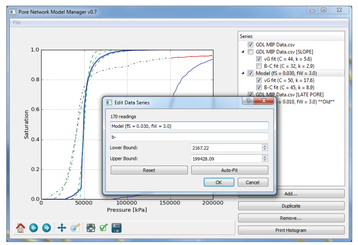

FIG 4. Version 0.7 of the "Pore Network Model Manager" application. Supported features include: Import and Export of both individual data sets and entire sessions, slicing of data points to home in on areas of interest (including automatic bound fitting) and generation of industry-standard pore size histograms from capillary data or model.

Unfortunately, when fitting real data, the relative simplicity of the PNM prevents a snug curve-fit. No unique combination of parameters definitively minimize the objective function, leading to numerous local minima.

A semi-automatic manual fitting GUI was developed, showcasing the benefits of direct user interaction and feedback it provides. **[Fig 6.]**

PYTHON & GUI DESIGN

Substantial effort was put towards researching the options available in terms of GUI development. Python offers several design packages, of which three were seriously considered for this project: `Tkinter`, `wxPython` and `PyQt4` (chosen).

Additionally there are a number of plotting libraries available. The decision to go with `matplotlib` over `PyQwt` came down mainly to the `Matlab`-friendly syntax style of the former.

Features such as dynamic variable declaration and extensive event handling (`Qt`'s `SLOT` and `SIGNAL` mechanism is a great example) make the programming experience resemble the classical physical engineering design process more than ever before, meaning that engineers outside of software should have no problem getting a handle on coding.

Designing the widget layout of a GUI with non-expert users in mind required analysis of existing soft guidelines such as the `OK|Cancel` vs. `Save|Discard` ideologies.

The result is a vindication of Python as a solid modeling platform, and software that is portable, customizable, easily expandable and very computationally efficient.

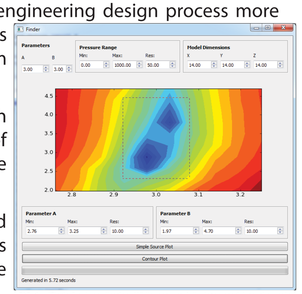

FIG 6. "Finder" sub-application. While not yet fully integrated into the main program, it allows users to make the process of finding best fit parameters for data interactive and intuitive, with visual validation provided immediately. Eventually it will be obsoleted (only relevant for 2-parameters) but hot fix tools such as this one enhance productivity immensely.

FUTURE WORK

- Enhance validity of basic code
  - Correlate pore sizes
  - Model surface roughness effects
  - Simulate late-pore filling
- Port over more sophisticated, Voronoi-cell based model code
- Further study SPSA algorithms to attain better understanding of its parameters in the context of the problem at hand, and convey information to user effectively

ACKNOWLEDGEMENTS

This work was supported via funding provided by the ***Summer Undergraduate Research in Engineering (SURE)*** program of McGill University and ***The Automotive Fuel Cell Cooperation (AFCC)***.

Mark Summerfield's ***"Rapid GUI Programming with Python and Qt"*** and the community at ***StackOverflow.com*** were invaluable aides in the software development process and recommended to any aspiring Python programmer.