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

Image analysis algorithms have been extensively developed and optimized in Python libraries. We used numpy, ndImage, and PyQt4's Qwt to come down mainly to the Matlab-friendly syntax style of the former.

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.

Due to the existence 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

Image analysis algorithms have been extensively developed and optimized in Python libraries. We used numpy, ndImage, and PyQt4's Qwt to come down mainly to the Matlab-friendly syntax style of the former.

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.

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

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 Poresimetry (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.

An application [Fig 4.] was developed to allow users to quickly and easily manage and visualize sets of data, rapidly fit generic Van Gouchten 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.

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.]

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 aids in the software development process and recommended to any aspiring Python programmer.