MET CS 767 - Project 4

Name: Ryan Kophs
Date: 12 December 2016

1 Tasks completed

- Implemented the fuel cell optimization equation presented in [1] using a genetic algorithm heuristic
- Implemented the fuel cell optimization equation presented in [1] using the heuristic presented in [2].
- Constructed a web application interface to configure and execute multiple runs of either algorithm implementation.

2 Web Interface

The web interface may be accessed via public link or by compiling locally. It is recommended to use the public link, as this has the most up to date code.

2.1 Publicly

Accessed the application at `https://rkophs.github.io/rkophs_CS_767_project_4`

2.2 Locally

The web application may also be compiled locally if running locally is preferred (although the public link has the most updated code). To compile and run:

```
$ cd /path/to/project/
$ yarn
$ webpack
$ python -m SimpleHTTPServer
```

Access the application at `http://localhost:8000`

2.3 Settings

Settings may be expanded by clicking on the settings link. It provides the ability to set all the fuel cell constants, cell parameter bounds, and the observed actual fuel cell parameters. It also provides links to set the parameters specific to the genetic algorithm and the algorithm proposed by [2]. Settings may be supressed by toggling the settings link.

2.4 Results

Results are displayed per execution run underneath the settings section. Switch between execution results by clicking the respective run links on the right. A graph shows a series of lines - each being the best solution obtained for a different generation/iteration of the algorithm. The observed actual fuel cell stack is highlighted in green, and the best solution found the algorithm that reduces the sum of squared error between the solution and the observed actual stack is highlighted in red.

The minimal sum of squared error is displayed above the graph. Below the graph are the results breakdowns per generation/iteration, and the configurations used for the run, defined in settings.

3 Implementation

The webapplication is built using React [3] and and Redux [4]. All code pertaining specfically to the fuel cell problem splace and machine learning algorithms resides in `/app/machineLearning/`. This directory is broken up as follows:

- `/app/machineLearning/GeneticAlgorithm.js`: General genetic algorithm that can be applied to any problem space.
- `/app/machineLearning/GeneticIndividual.js`: A factory method that constructs a single solution to any problem space by accepting **Crossover**, **Mutate**, **Fitness**, **Cost**, **Solution** call back function in the the problem space.
- `/app/machineLearning/JPSAlgorithm.js`: General algorithm proposed in [2] that can be applied to any problem space.
- `/app/machineLearning/FuelCell.js`: The individual calculations for a fuel cell potential given input constants and parameters declared in [1].
- `/app/machineLearning/FuelCellML.js`: A factory method that constructs a single individual solution pertaining to the fuel cell stack potential. Implements the **Crossover**, **Mutate**, **Fitness**, **Cost**, **Solution** call backs used by `/app/machineLearning/GeneticIndividual.js`.

4 Inputs and Solution

4.1 Inputs

All inputs are defined as the defaults on the settings page of the application. The are as follows:
Cell Parameter Bounds:

	x_1	x_2	x_3	x_4	λ	$R_c(\Omega)$	$B(V)$
Lower	-1.19969	0.001	0.000036	-0.00026	10	0.0001	0.0136
Upper	-0.8532	0.005	0.000098	-0.0000954	24	0.0008	0.5

Observed (actual) Fuel Cell Parameters:

	x_1	x_2	x_3	x_4	λ	$R_c(\Omega)$	$B(V)$
Actual	-0.944957	0.00301801	0.00007401	-0.000188	23	0.0001	0.02914489

Fuel Cell Constants:

$A(cm^2)$	N_s	RH_a	$l(cm)$	RH_c	$\rho_a(atm)$	$\rho_c(atm)$	$T(K)$	$i_{limit,den}(A/cm^2)$
27	24	1	0.0127	1	2.96077	4.93462	353.15	0.86

Genetic Algorithm Parameters:

Noise	Max Current (A)	Generations	Child rate	Mutation Rate	Population	PRG Seed
0.3	20	100	2	0.1	400	123

Algorithm Proposed in [2] Parameters:

Noise	Max Current (A)	Iterations	α	Mutation Rate	Population	PRG Seed	$evalFrac$	Target Cost
0.3	20	1000	0.999	0.2	400	123	0.1	1.2

4.2 Solution

With the inputs defined above, the Genetic Algorithm resulted in a sum of squares of 1.2454399. The following shows its resulting current and potential relationships:

Current (A)	Calculated Potential (V)	Observed (Actual) Potential (V)
1	23.455987642743438	23.564554482390516
2	22.266071132995584	22.24605976795055
3	21.533698896349215	21.671246947536986
4	20.987287939879614	21.1274826128611
5	20.541583158530862	20.229085854654382
6	20.158468583948707	20.43843255959356
7	19.81746715049399	19.23365640902375
8	19.506201648853413	19.267538832399143
9	19.216518975308425	19.66066261240516
10	18.942660124538627	18.867660801177234
11	18.680293079265184	18.656749539340336
12	18.42594897604348	18.36686923581295
13	18.176656990658813	18.36421709805783
14	17.929673317380036	18.05773556826612
15	17.682238373686456	17.562141060717035
16	17.431303326370443	17.21126480052054
17	17.173144739941858	17.11364785799601
18	16.902710752432355	17.205571871654968
19	16.61232303013411	16.62344467263446
20	16.288644917967673	16.006013706044644

With the inputs defined above, the algorithm proposed in [2] resulted in a sum of squares of 1.6155517. The following shows its resulting current and potential relationships:

Current (A)	Calculated Potential (V)	Observed (Actual) Potential (V)
1	23.190057959619537	23.564554482390516
2	22.04300667240566	22.24605976795055
3	21.333125053735824	21.671246947536986
4	20.800950224227797	21.1274826128611
5	20.365049620298592	20.229085854654382
6	19.989034849618445	20.43843255959356
7	19.653377419970667	19.23365640902375
8	19.34629680813374	19.267538832399143
9	19.060060210369855	19.66066261240516
10	18.789237915618017	18.867660801177234
11	18.52978464625933	18.656749539340336
12	18.27850850774464	18.36686923581295
13	18.032733401523934	18.36421709805783
14	17.790057172527675	18.05773556826612
15	17.548146598597707	17.562141060717035
16	17.304520676517086	17.21126480052054
17	17.05626049663695	17.11364785799601
18	16.799531650260327	17.205571871654968
19	16.52864945927416	16.62344467263446
20	16.23390805690545	16.006013706044644

5 Resources

1. Chakraborty, U. K., Abbott, T. E., & Das, S. K. (2012). PEM fuel cell modeling using differential evolution. Energy, 40(1), 387-399. doi:10.1016/j.energy.2012.01.039

2. Chakraborty, U. (2012). A new stochastic algorithm for proton exchange membrane fuel cell stack design optimization. Journal of Power Sources, 216, 530-541. doi:10.1016/j.jpowsour.2012.06.040

3. https://facebook.github.io/react/

4. http://redux.js.org/

5. https://en.wikipedia.org/wiki/Genetic_algorithm

6. https://en.wikipedia.org/wiki/Fitness_proportionate_selection