

PNNL Cost-Performance Tool (DOE-OE sponsored) for Estimation of Capital Costs for Redox Flow Batteries. Battelle IPID 30401.

Copyright © 2013, Battelle Memorial Institute  
All rights reserved.

**Instructions for accessing, modifying and sharing redox flow battery cost estimator source code using Github. September 10, 2013**

Alasdair Crawford 509-438-3916 [Alasdair.crawford@pnnl.gov](mailto:Alasdair.crawford@pnnl.gov)

Vilayanur Viswanathan 509-372-4745 vilayanur.viswanathan@pnnl.gov

Pacific Northwest National Laboratory

PNNL has developed an interactive tool to estimate the capital cost for PNNL all vanadium Gen 1, Gen2, and iron-vanadium chemistries. This tool has also been adapted for organic electrolyte redox flow batteries. The Gen 2 all vanadium system incorporates the PNNL-developed electrolyte with high concentration and wide operating temperature range. The tool allows determination of the most cost-effective chemistries and the optimum operating conditions for power or energy intensive applications, providing a strategy for a redox flow battery management system (BMS). This tool was developed with funding from The U. S. Department of Energy, Office of Electricity Delivery and Energy Reliability/Energy Storage Program, under the leadership of Dr. Imre Gyuk.

**How to download code from github, modify, and run.** You do not need a github account to access and download code.

1. Go to <https://github.com/PNNL-OE-Redox-Flow-Battery-Cost-Tool/PNNL-OE-Redox-Flow-Battery-Cost-Tool>
2. In the lower right, select “download zip”. This will download a zip of the project.
   1. Unzip the file.   
      NOTE: If you simply want to use the GUI and do not need to look at the source code, please go to page 3.
3. Open Visual Studio. If you don’t have it, you can download it for free from Microsoft. It is recommended to download Visual Studio 2010 C++ express at this link: <http://www.microsoft.com/visualstudio/eng/downloads#d-2010-express>
4. Select “Open Project”
5. Navigate to BatGUI.sln in the folder you unzipped in step 3. Open this file.
6. In the solution explorer, expand BatGui, and go to Headerfiles and expand this selection.
7. Double click on Form1.h. This should preview what the program looks like.
8. Right click on Form1.h and select “view code”. This will allow you to look at the code.
9. Make desired modifications. (Detailed description of the source code layout is available in the document layout.doc at the repository).
10. Find the green triangle in the toolbar. To the right of it is a drop down menu that says “Debug”. Click on this window and change it to “Release”. If it already says “Release” you don’t need to do this step.
11. Click on the green triangle or press f5. This will compile and run the program. The may take 1-5 minutes depending on your system.
12. For further instruction on using the source code, please go to page 11.

**How to get a github account and submit proposed changes to code.**

Note, if you do not intend to share your work with the community, you can skip this section.

1. Go to github.com
2. Download for your system. For example, for Windows, download is at http://windows.github.com/
3. Once you have an account, click on Fork button to fork our repository at <https://github.com/PNNL-OE-Redox-Flow-Battery-Cost-Tool/PNNL-OE-Redox-Flow-Battery-Cost-Tool> This will make a copy of our code repository to your github account.
4. Open Github program from your computer. Using the github software, click on clone button to clone this repository to your desktop. This will download the code repository to your desktop.
5. If you need to make changes to code and share, make changes (the detailed description of source code layout is provided in layout.doc at the repository). Then using the github software, click “sync” to upload your changes from your computer to your github repository at the github.com website.
6. To request that your changes to the code be merged into the PNNL code repository, go to your repository on the github site and hit the pull request button. There will be a comment box provided for you to state details such as which part of the code tree (branch vs. master) you would like your version to be added. Also, within the source code, please provide ample comments describing the changes.
7. The program administrator will look at the proposed changes and accept or ask for further clarifications such as reason for change, request for more comments and other issues. The program admin may also want your change to be on a new branch. In this case the admin will close your pull request and open a new branch. The admin will send you instructions to send a new pull request to this newly created branch.
8. Github does automatic version control.
9. Further info on Github can be obtained at <http://help.github.com>. The administrator will also be available to answer questions. Questions may also be addressed to the emails provided above.



PNNL Cost-Performance Tool (DOE-OE sponsored) for Estimation of Capital Costs for Redox Flow Batteries. Battelle IPID 30401.

Copyright © 2013, Battelle Memorial Institute  
All rights reserved.

**Instructions for using redox flow battery cost estimator PNNLBatGui.exe**

**September 10, 2013 (updated 4/12/16)**

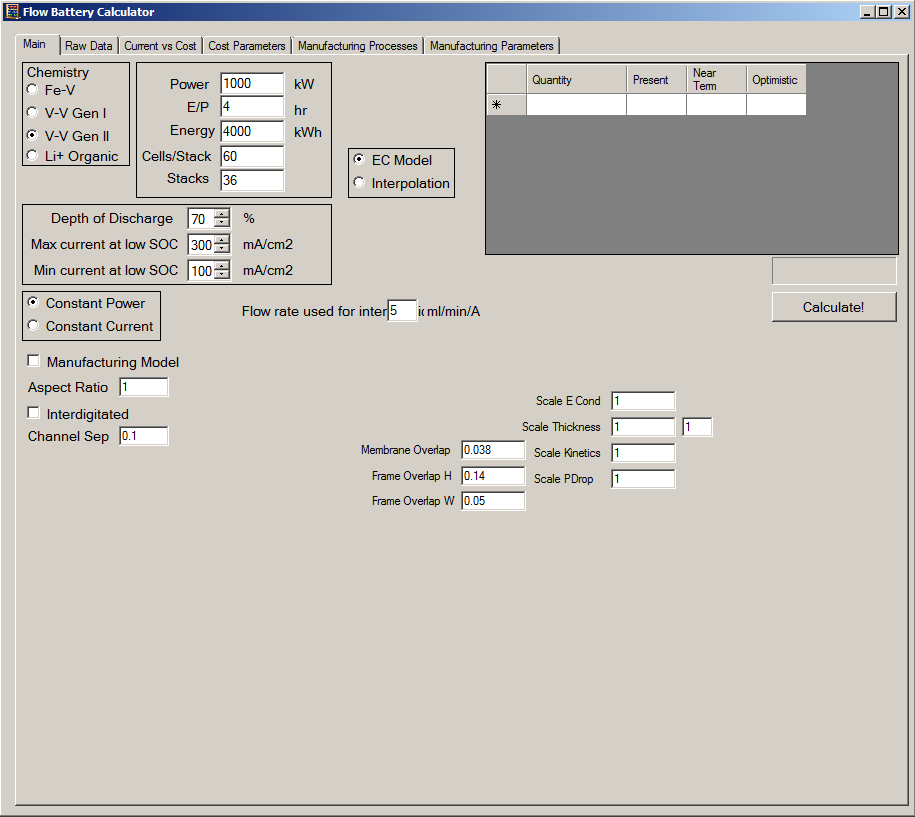
**This does not require using C++ - just uses GUI**

Alasdair Crawford 509-438-3916 [Alasdair.crawford@pnnl.gov](mailto:Alasdair.crawford@pnnl.gov)

Vilayanur Viswanathan 509-372-4745 vilayanur.viswanathan@pnnl.gov

Pacific Northwest National Laboratory

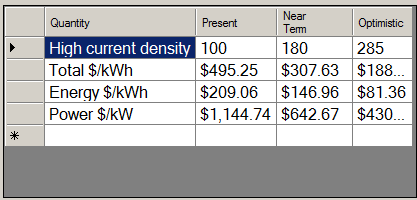
PNNL has developed an interactive tool to estimate the capital cost for PNNL all vanadium Gen 1, Gen2, and iron-vanadium chemistries. This tool has also been adapted for organic electrolyte redox flow batteries. The Gen 2 all vanadium system incorporates the PNNL-developed electrolyte with high concentration and wide operating temperature range. The tool allows determination of the most cost-effective chemistries and the optimum operating conditions for power or energy intensive applications, providing a strategy for a redox flow battery management system (BMS). This tool was developed with funding from The U. S. Department of Energy, Office of Electricity Delivery and Energy Reliability/Energy Storage Program, under the leadership of Dr. Imre Gyuk.

1. In chemistry pane, select the cell chemistry to be used.
2. In the battery design pane, select the required power, the E/P ratio, the required energy, the cells per stack, and the number of stacks desired.
3. Select the depth of discharge (DOD) and the current ranges to investigate.
4. Select EC model or interpolation. Interpolation requires a \*.csv file with voltage as a function of current and SOC. For an example of how to format your data, see example.csv in the zip file. Note that an arbitrary amount of data points can be entered. EC model uses built in electrochemical model to calculate voltage. If interpolating, a flow rate per ampin ml/min/A must be entered in the “Flow rate used for inter” box in the GUI.
5. Select constant power or constant current. The former reduces current at higher SOCs to provide a constant power for increased efficiency. The latter uses the current at low SOC for all SOCs.
6. If desired, the interdigitated model can be turned on.
7. Select calculate.

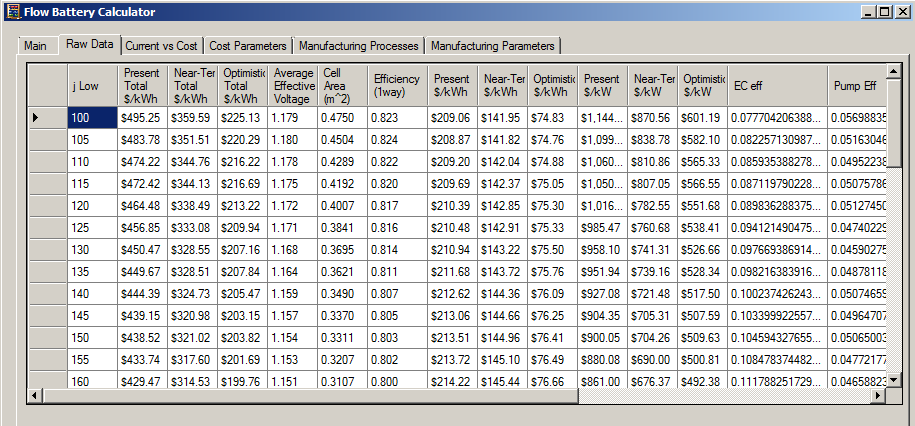
Please note that in some instances, the program will give an error message if, for the range of current density values selected, the current densities are too low, too high or both for the DOD selected. For example, let us assumesome of the current densities investigated are too low, while none are too high. The error message will state that the current density is too low. Increase the current density till you get stable results. Same applies if current density is too high. Decrease the current density till you get stable results.

In short, for a selecteted DOD, the stable current density range selected can be found by trial and error for your chemistry.

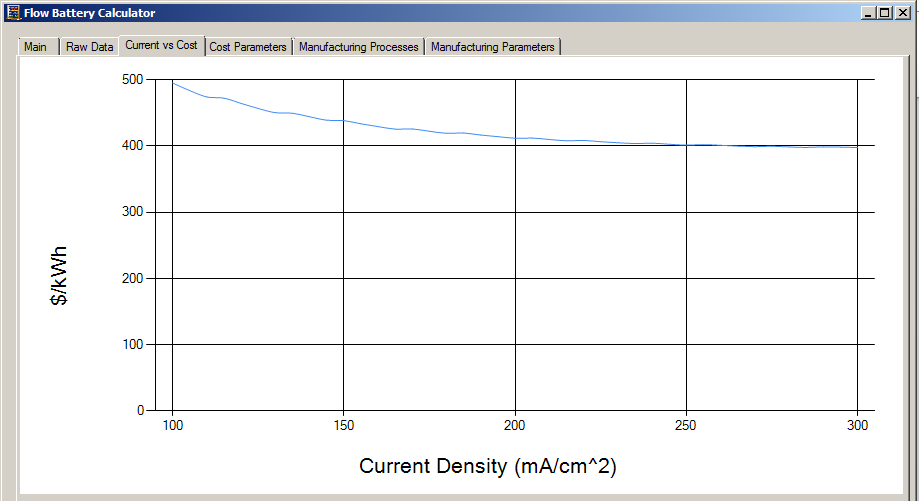
1. The results of the calculation are displayed in this pane. The costs and optimum current densities are returned.



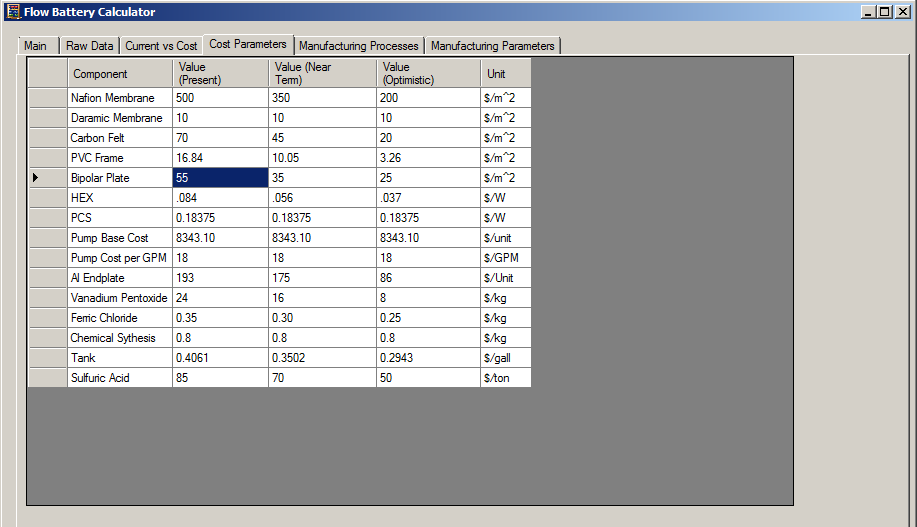
Advanced



1. The raw data tab can be used to look at each current density’s results in depth. This can be used to look at cell area, efficiency, efficiency loss each from pumping, shunt, and electrochemistry, and flow rates used. The cost trend as a function of current density can be used to set the appropriate current density range to be investigated for this DOD as shown below.



1. The current vs cost tab can be used to graphically look at how the total cost varied with current density. This graph can be useful in setting the appropriate current density range to be investigated.



1. The cost parameters tab can be used to choose the cost of each component. This allows choosing the latest cost information available based on material and or manufacturing improvents.

Some general tips:

As discussed, the following input parameters are selected by the user (the max flow rate button has been deactivated, so you can enter any number here):

* DOD
* Current density range.

The model will only output the cell operating conditions such as voltage and flow rate, and system parameters such as electrochemical, pumping and shunt current related losses that maximizes, Veffective (Veff) for each current density in the range selected, where Veff is the discharge voltage after accounting for pumping and shunt current losses. Hence, the user has to run the model by adjusting these parameters (DOD, current density range) till the operating conditions for minimum cost is obtained. For example, you can chose DOD = 50%, and current density range as 80 to 400 mA/cm2 for a given E/P ratio. The model will list the results in the raw data tab, stepping the current density in increments of 5 mA/cm2. For the DOD and current density range chosen, the ‘minimum” cost can be determined. By varying the DOD, the “minimum” cost can be determined for each DOD. After a few iterations, the DOD and current density that provides the maximum Veff can be determined. This is assumed to correspond to the minimum system cost.

**Instructions on how to use the PNNL-OE C++ Source Code.**

Outline: the following functions appear in the program in the same order they appear here.

From #pragma region to #pragma endregion is all setting up the GUI. It is advised not to alter this unless you are familiar with Windows forms.

**InitializeCostParameters** is used to initialize the component costs. It runs when the program is started. It fills out the component cost table with default values. Therefore it is rather simple to change these default values.

**populateSpline2V** takes a series of points (X,Y) and produces a matrix describing a cubic spline interpolation of these points.

**interpolateSplineV** takes an interpolation matrix (made by the previous function populateSpline2V) and the point to interpolate at, X, and returns a Y.

**interpolate2dSpline** takes a series of interpolation matrixes and a series of values and uses it to interpolate a value as a function of two variables, ie Z as a function of Y and X. in the program it is used to interpolate voltage as a function of SOC and current density.

**fileRead** reads a csv file and interprets it as a matrix and returns it. Used to read a matrix of V as function of current density and SOC and use cubic spline interpolation.

**Asinh** is the inverse hyperbolic sine, described here as it does not exist in cmath library.

**Voltage** is the electrochemical model, it returns a voltage as a function of SOC, cell area, flow rate, current density, aspect ratio of cell, cell chemistry, membrane used, and interpolation matrixes (optional). It currently contains two V-V electrochemical models, a Fe-V model, and a Li+ model. More may be added.

**CostTab** is used to read cost data from the component cost table.

**unitFunction** is an experimental function used for giving component costs as functions rather than constants.

**powerCost** calculates the stack and pump costs as a function of cell area, maximum flow rate, aspect ratio, cells per stack, power rating, system chemistry, and membrane type.

**energyCost** calculates the chemical and tank costs as a function of average effective voltage, depth of discharge, concentration, and system chemistry. The cost returned is per kWh.

**shuntLoss** calculates the shunt current power loss, in watts, as a function of cell voltage, area, channel dimensions, aspect ratio, and number of cells. It uses a finite different method to calculate this shunt power loss.

**pLoss** calculates the power loss from pumping electrolyte through the stack. Function of cell area, flow rate, aspect ratio, channel dimensions, number of channels. Uses darcy’s law to calculate pressure drop through cell, and hagen-poiseuille equation to calculate pressure drop through the cell channels.

**MarshalString** is used to convert managed strings to unmanaged strings – this does not need altered.

**slope** is a simple function used to calculate the slope between two points. Useful shortcut.

**calculate\_Click** is what executes when the calculate button is clicked. This can be thought of as the “main program”. This function will collect the information from the GUI (such as rated power, rated energy, system chemistry, etc) and use it to perform an optimization. After analysis it outputs the results in the results table.

**epBox\_TextChanged** is a simple function – when the energy/power ratio box is changed, the energy box is updated accordingly.

**energyBox\_TextChanged** is a simple function – when the energy box is changed, the energy/power ratio box is updated accordingly.