

Dataset Generator Documentation

Authors: Patrick Dixon and Alia Rezvi

Date: 01/09/23

Version: 1.2

Contents

| | |
|--|----|
| User Documentation..... | 4 |
| Introduction | 4 |
| Requirements..... | 5 |
| Local Application Installation..... | 5 |
| Windows | 6 |
| Mac/Linux | 6 |
| Hardware Issue | 6 |
| NetBeans Developer Application Installation..... | 6 |
| Application instructions | 7 |
| Load Configurations | 10 |
| Process Variables | 11 |
| Process Area Descriptions..... | 12 |
| Input Variables | 13 |
| Input Validations | 14 |
| State Variables | 15 |
| Output Variables | 16 |
| Lab Variables | 17 |
| Generate Data..... | 18 |
| Developer Documentation | 20 |
| Introduction | 20 |
| Process Configuration | 20 |
| Time Periods | 20 |
| Measurements..... | 21 |
| Input Configuration..... | 21 |
| State Configuration | 22 |

| | |
|--|----|
| Methods to adapt for different state variables | 23 |
| Methods to adapt for no state variables | 24 |
| Dynamics | 26 |
| Output Configuration | 26 |

User Documentation

Introduction

Imagine you are one of the following:

- A student or young engineer that wants to learn data analytics
- An instructor that wants to teach students and young engineers data analytics
- A vendor that wants to develop a solution using data analytics, test the solution, and demonstrate it
- An engineer for a manufacturer that wants to develop a methodology for performing data analytics using various techniques

In all of these cases, there is a fundamental requirement: process data. However, there are some challenges to obtaining process data:

- Unless you work for a manufacturer, you don't have an industrial process. Therefore, getting process data means asking a manufacturer to give you theirs. Manufacturers regard their data as intellectual property and are not looking to hand it out to others.
- Even if you have process data, it may not have sufficient excitation of the process to yield useful prediction models or analysis. Some industrial processes are single setpoint dominant; they run the same way all the time. In processes with grade changes, often the changes are occurring simultaneously and therefore are not decoupled. Coupled changes make it impossible to determine the independent magnitude and nature of these changes. To yield results that are useful, there must be sufficient independent movement of the variables of interest that significantly exceed the level of process noise.
- Typical datasets from an industrial process contain several months of data. If step tests are performed to provide decoupled responses, the time to conduct step tests for each variable of interest and wait for those responses to settle to steady state can represent an enormous amount of time. Obtaining a dataset from an industrial process can be a very significant investment of time.

If the use case is to perform data analytics or predictive modelling for an industrial process, of course the actual data needs to be used. The authors do not suggest actual process data can be replaced with artificially generated data. However, in the use cases considered in this paper, actual process data is not required.

The goals of this project were as followed:

- Provide a general-purpose tool that can be customized and configured to produce a dataset representative of an actual industrial process
- Reduce the time to generate that dataset from the months required for actual data to hours for simulated data
- Provide the tool as open source, which can be obtained, customized, and improved at no cost.

This guide will go through the instructions of how to operate the application. The dataset generated from this application is intended for the paper industry, which is why there may be specific requirements. The developer section of this documentation goes through what can be changed in the code to adapt.

The source of this documentation is the GitHub repository:

<https://github.com/Paramount10/dataset-generator>.

Requirements

This is a desktop application intended to be run on a computer/laptop device, not a mobile device.

Hardware

- Minimum 8 GB RAM available
- Minimum 2 GB storage available (sufficient to hold at least three generated datasets)

Software

- Java Development Kit (JDK) 8 or above installed

It is recommended to run this application on a machine with a high performing CPU and primary/secondary storage.

Local Application Installation

From the directory files, copy the 'generator' folder and place it in an accessible location that can be reached from the command line.

Windows

Open the command line and change the directory to the 'generator/main' path. Run the following two commands:

```
javac -cp ".;libraries\*" src\generator\*.java
```

```
java -Xmx6144m -cp "src;libraries\*" generator.Main
```

Alternatively, there is a 'run.bat' file in the 'generator/main' folder that holds these commands which can be run instead of executing the commands individually.

Mac/Linux

Open the command line and change the directory to the 'generator/main' path. Run the following two commands:

```
javac -cp ".:libraries/*" src/generator/*.java
```

```
java -Xmx6144m -cp "src:libraries/*" generator.Main
```

Hardware Issue

This application has been tested on machines with various hardware differences, but it was not possible to cover every case. If the hardware is too slow, the dataset generation may cause a memory error. In this unlikely event, the solution would be to increase the maximum heap size (by increments of 1024) when running the second command on the command line (if RAM permits) for example:

(On Windows)

```
java -Xmx7168m -cp "src;libraries\*" generator.Main
```

(On Mac/Linux)

```
java -Xmx7168m -cp "src:libraries/*" generator.Main
```

NetBeans Developer Application Installation

If you would like to edit the user interface of the application using the NetBeans GUI Editor, there is a NetBeans version of the project called 'NetBeans.zip'. Basic proficiency of NetBeans is assumed if using this version. Once you download and extract the main folder, you can import the project by

going to File -> Import Project -> From ZIP. Please note that this version of the project was not intended to be the final application, so this will include: less annotated comments, different directory structure and uneven code formatting. The GUI should be the same but the Form.java file will be different.

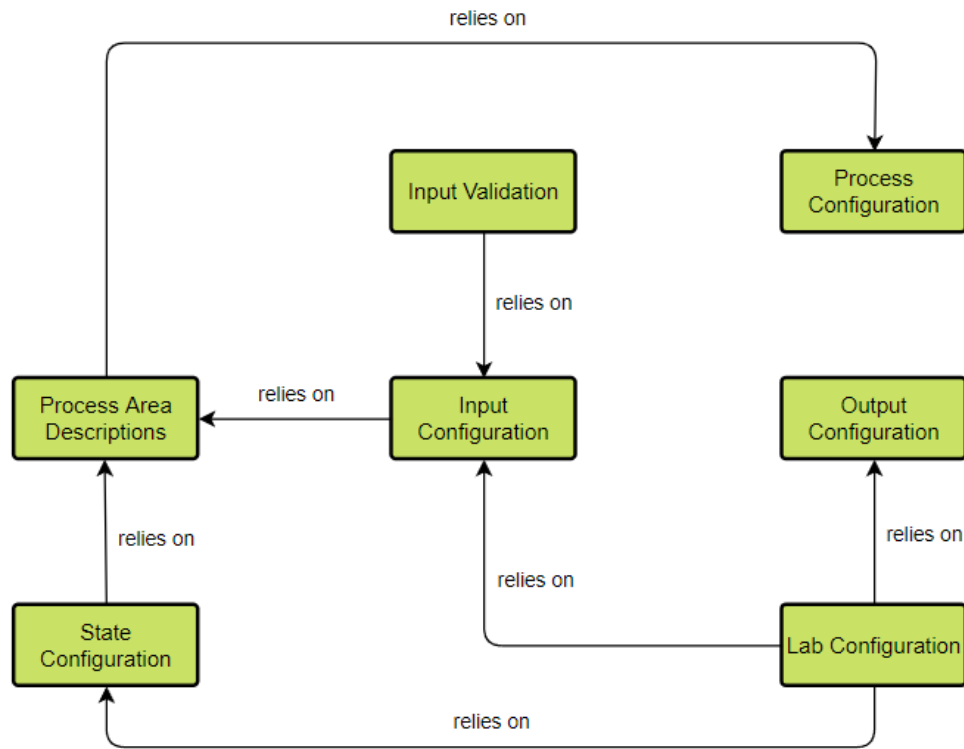
Application instructions

The application includes a sidebar with different buttons for pages and a main panel which includes the selected page. The first screen you will see is the home page, which includes the option to load configurations. Loading configurations are preferable as there are already configurations stored that you can test and adjust, but if you modified the application or would like to enter data manually, you can skip the section about loading configurations.

Throughout the application, there are important details to note when following this guide:

1. The data must be entered/submitted in sequential order. The sidebar shows the order in which the pages must be submitted.
2. Some pages rely on data from other pages. The sections below will mention what pages are affected by the current page.
3. Because of the relationship between some pages, if one page is changed, the affected pages will have to be re-submitted. If you didn't change a value on the page that affects other pages, you can just press the submit button again without changing the values on the other

related pages. There will be a pop-up box reminding you if a page needs to be re-submitted. The dependencies of the pages can be seen below:



4. All values that you enter will be trimmed of leading and trailing white space characters.
5. On pages with tables:
 - a. You can double-click on a cell to edit, and either press the Enter key on your keyboard or click on another cell to confirm the change.
 - b. You can re-size table headers by clicking and dragging on the header separators.
 - c. The 'Add row' button adds a row to the table based on the data in the text fields.
 - d. The 'Delete row' button deletes the row which has been selected (click on any cell in a row for it to be selected)
 - e. The 'Clear table' button clears the entire table.
6. In the 'Input Validations' page, the columns are created only once. If you change the amount of input configurations, you would need to exit the application and run it again. This can be something to work on for this application so the 'Input Validations' page can be accurately reflected once the input configurations change.

7. In the directory, under '...generator/main', the 'config' folder holds the CSV files with configurations that can be loaded into the application, further explained in the below section. The 'data' folder holds the dataset generated, which would be in the format of the current date and time.
8. There are limits to how much data this application can produce. The largest amount of data tested has been the datasets generated from the 'noise and sine' configuration explained in the next section. An increased amount of: input/state/output variables, input validation rows, uncoupled moves or a smaller value of: process configuration time periods, leads to more memory being used and Java may reach its limit of the heap size. In that case, heap size may be increased as described in the previous section which possibly could resolve the issue, but better hardware would also be required in that case. There may be a point at which the RAM limit is reached and the Java heap runs out of memory completely at which point, alternative solutions would have to be found.
9. Not all user validations are covered, so if you enter data out of sequence there may be a generation error. This can be a suggestion to work on for this application to cover all user validations.

Load Configurations

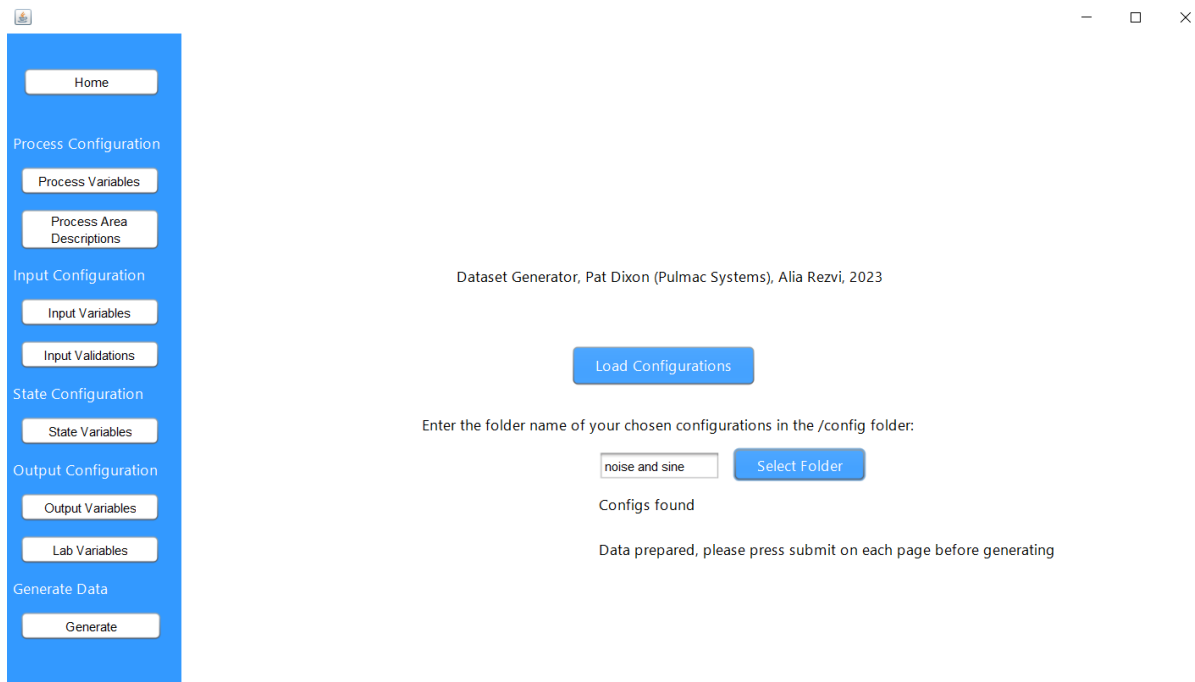
If you would like to load a set of configurations, click on the 'Load Configurations' button. A text field will appear asking for a folder name where the configurations are stored.

The default configuration folders are:

- 'plain' = No noise or sine waves added
- 'only noise' = Noise added but no sine waves
- 'noise and sine' = Noise and sine waves added

This guide will demonstrate the 'noise and sine' configuration.

Enter your chosen folder name and click on the 'Select Folder' button, which will check if the folder exists and loads the configurations if they are in the correct format:



The screenshot shows a web application interface for a 'Dataset Generator'. On the left is a blue sidebar with a menu containing: 'Home', 'Process Configuration' (with sub-items 'Process Variables' and 'Process Area Descriptions'), 'Input Configuration' (with sub-items 'Input Variables' and 'Input Validations'), 'State Configuration' (with sub-item 'State Variables'), 'Output Configuration' (with sub-items 'Output Variables' and 'Lab Variables'), and 'Generate Data' (with sub-item 'Generate'). The main content area has a title 'Dataset Generator, Pat Dixon (Pulmac Systems), Alia Rezvi, 2023'. Below the title is a blue button labeled 'Load Configurations'. Underneath this button is a text prompt 'Enter the folder name of your chosen configurations in the /config folder:'. A text input field contains the text 'noise and sine', followed by a blue button labeled 'Select Folder'. Below the input field, it says 'Configs found'. At the bottom of the main area, a message reads 'Data prepared, please press submit on each page before generating'.

To generate the dataset, follow the sections below which should now include the loaded data.

Process Variables

Process variables affect data throughout all configurations. These include:

- Start Date = The date at which the dataset starts at from 00:00. The process period is added to each date for every row in the final dataset.
- Process Period = A sample time.
- QCS Period = The time for the QCS variables (variables that are a subset of state variables), must be a multiple of the process period.
- Lab Period = The time for the output lab variables (must be a multiple of the process period).
- PulpEye Period = The time for the PulpEye variables (variables that are a subset of state and input), must be a multiple of the process period.
- Uncoupled Moves = The quantity of input steps.
- Trim = Used in state variable calculations.
- Draw = Used in state variable calculations.
- Added Settle = An added time period.

All process variables except 'Trim' and 'Draw' must be an integer.

If the configurations were loaded, this is what should appear as the default values. Press submit to confirm:

The screenshot shows a web application titled "Process Configuration". On the left is a blue sidebar with navigation buttons: "Home", "Process Configuration" (with sub-buttons "Process Variables" and "Process Area Descriptions"), "Input Configuration" (with sub-buttons "Input Variables" and "Input Validations"), "State Configuration" (with sub-button "State Variables"), "Output Configuration" (with sub-buttons "Output Variables" and "Lab Variables"), and "Generate Data" (with sub-button "Generate"). The main content area is titled "Process Configuration" and contains the following fields and values:

| Field | Value |
|----------------------|----------|
| Start Date | 01/01/21 |
| Format | mm/dd/yy |
| PulpEye Period (sec) | 300 |
| Process Period (sec) | 5 |
| Uncoupled Moves | 10 |
| QCS Period (sec) | 60 |
| Trim (ft) | 20.0 |
| Lab Period (sec) | 1200 |
| Draw | 1.1 |
| Added Settle (sec) | 10000 |

At the bottom of the main area, there is a blue "Submit" button and the text "Values submitted".

Process Area Descriptions

These descriptions consist of deadtime reels and lags which affect input and state dynamics. The descriptions can be applied to each input and state variable once submitted. The only condition here is that 'Deadtime Reel' converted into hours (multiplied by 60) must be a multiple of the 'Process Period'.

If the configurations were loaded, this is what should appear as the default values. Press submit to confirm:

Home

Process Configuration

Process Variables

Process Area Descriptions

Input Configuration

Input Variables

Input Validations

State Configuration

State Variables

Output Configuration

Output Variables

Lab Variables

Generate Data

Generate

Process Area Descriptions

| Name | Deadtime Reel | Lag 1 Reel | Lag 2 Reel |
|-----------------|---------------|------------|------------|
| AREA_HEADBOX | 0.5 | 1 | 0.5 |
| AREA_THICKSTOCK | 20 | 15 | 5 |
| AREA_THINSTOCK | 15 | 5 | 5 |
| AREA_REEL | 0 | 0 | 0 |

Name

Lag 1 Reel (min)

Add row

Delete row

Deadtime Reel (min)

Lag 2 Reel (min)

Submit

Clear table

Submitted successfully

Input Variables

These are the input variables that form the initial columns of the dataset. Specific requirements for variables are mentioned in the 'Developer Documentation' section of this document if no code changes are made.

The values for each variable consist of a: name, description, noise, lag, range, order, sine values. The noise is an additional value that gets entered into a random number generator and added to each cell value in the final dataset for all input variables. This can be set to 0 and will result in the same value for many cell values.

The sine period (must be larger than the process period) is used to create a degree value based on the ratio to the process period and the amplitude is multiplied by the final sine result. Amplitude can be set to 0 if no sine waves are desired.

If the configurations were loaded, this is what should appear as the default values. Press submit to confirm:

Home

Process Configuration

Process Variables

Process Area Descriptions

Input Configuration

Input Variables

Input Validations

State Configuration

State Variables

Output Configuration

Output Variables

Lab Variables

Generate Data

Generate

Input Configuration

| Name | Description | Noise | MV Lag | Max | Min | Sin Period | Sin Amplitude | Order |
|-----------------|---------------|-------|--------|------|-----|------------|---------------|-------|
| MV_WireSpeed | AREA_HEAD... | 1 | 5 | 1500 | 0 | 47 | 10 | 7 |
| MV_SWSpecifi... | AREA_THICK... | 0.01 | 5 | 3 | 0.5 | 10 | 2 | 3 |
| MV_HWSpecifi... | AREA_THICK... | 0.01 | 5 | 3 | 0.5 | 10 | 3 | 1 |
| MV_OCCSpe... | AREA_THICK... | 0.01 | 5 | 3 | 0.5 | 10 | 1 | 2 |
| MV_SWFlow | AREA_THICK... | 1 | 5 | 500 | 0 | 10 | 2 | 4 |
| MV_HWFlow | AREA_THICK... | 3 | 5 | 500 | 0 | 10 | 3 | 14 |
| MV_OCCFlow | AREA_THICK... | 2 | 5 | 1000 | 0 | 27 | 6 | 13 |
| MV_ThinStock... | AREA_THINS... | 4 | 5 | 2200 | 0 | 10 | 5 | 10 |
| MV_ThinStock... | AREA_THINS... | 0.01 | 30 | 1.5 | 0.5 | 10 | 3 | 11 |
| MV_FillerFlow | AREA_THINS... | 1 | 5 | 50 | 10 | 10 | 1 | 6 |
| MV_PressLoad | AREA_HEAD... | 2 | 15 | 1000 | 700 | 89 | 2 | 15 |
| MV_SteamPre... | AREA_HEAD... | 0.5 | 30 | 100 | 50 | 10 | 9 | 5 |
| MV_BlendChe... | AREA_THINS... | 0.5 | 30 | 75 | 25 | 10 | 1 | 16 |

Name

Noise

Description

MV Lag (sec)

Max

Min

Sin Period (sec)

Sin Amplitude

Add row

Delete row

Submit

Clear table

Submitted successfully

Input Validations

An additional set of coupled moves per input variable can be added. These are optional and this page can be skipped if desired. The data is added to the table manually by editing each cell, and an empty row can be added.

If the configurations were loaded, this is what should appear as the default values. Press submit to confirm:

Home

Process Configuration

Process Variables

Process Area Descriptions

Input Configuration

Input Variables

Input Validations

State Configuration

State Variables

Output Configuration

Output Variables

Lab Variables

Generate Data

Generate

Input Validation Configuration

| Row | MV_WireS... | MV_SWS... | MV_HWS... | MV_OCC... | MV_SWFL... | MV_HWFL... | MV_OCC... | MV_ThinS... | MV_ThinS... | MV_FillerF... | MV_F |
|-----|-------------|-----------|-----------|-----------|------------|------------|-----------|-------------|-------------|---------------|------|
| 1 | 0 | 0.5 | 0.5 | 0.5 | 0 | 0 | 0 | 0 | 0.5 | 10 | 700 |
| 2 | 750 | 1.75 | 1.75 | 1.75 | 250 | 250 | 500 | 1100 | 1 | | |
| 3 | | | | | | | | | | 30 | 850 |
| 4 | | | | | | | | | | | |
| 5 | | | | | | | | | | | |
| 6 | | | | | | | | | | | |
| 7 | | | | | | | | | | | |
| 8 | | | | | | | | | 1.5 | 50 | 1000 |
| 9 | | | | 3 | 500 | 500 | 1000 | 2200 | | | |
| 10 | 0 | | 0.5 | | 0 | | 0 | | 0.5 | | 700 |
| 11 | | 0.5 | | 0.5 | | 0 | | 0 | | 10 | |
| 12 | 1500 | 3 | 3 | 3 | 500 | 500 | 1000 | 2200 | 1.5 | 50 | 1000 |
| 13 | 750 | 1.75 | 1.75 | 1.75 | 250 | 250 | 500 | 1100 | 1 | 30 | 850 |
| 14 | | | | | | | | | | | |
| 15 | 1100 | | | | 300 | 300 | 500 | | | | |
| 16 | | 1 | 1 | | | | | | | | |
| 17 | | | | | | | | 2000 | 1 | | |
| 18 | | | | | 250 | | 900 | | | | 800 |
| 19 | | | | | | 250 | | | | 40 | |
| 20 | | | | | | | | | | | |

Submitted successfully

Add rowDelete rowClear tableSubmit

State Variables

These are the state variables that form the next set of columns in the final dataset. Specific requirements for variables are mentioned in the 'Developer Documentation' section of this document if no code changes are made.

The values for each variable consist of a: name, description, range and noise. The noise is an additional value that gets entered into a random number generator and added to each cell value in the final dataset for all state variables. This can be set to 0 and will result in the same value for many cell values.

If the configurations were loaded, this is what should appear as the default values. Press submit to confirm:

Home

Process Configuration

Process Variables

Process Area Descriptions

Input Configuration

Input Variables

Input Validations

State Configuration

State Variables

Output Configuration

Output Variables

Lab Variables

Generate Data

Generate

State Configuration

| Name | Description | Noise | Max | Min |
|--------------------|-----------------|-------|------|-----|
| MV_SWFreeness | AREA_THICKSTOCK | 2 | 1000 | 300 |
| MV_HWFreeness | AREA_THICKSTOCK | 1 | 1000 | 300 |
| MV_OCCFreeness | AREA_THICKSTOCK | 1 | 1000 | 300 |
| MV_SWPct | AREA_THICKSTOCK | 0 | 100 | 0 |
| MV_HWPct | AREA_THICKSTOCK | 0 | 100 | 0 |
| MV_OCCPct | AREA_THICKSTOCK | 0 | 100 | 0 |
| MV_HeadboxPressure | AREA_HEADBOX | 0.25 | 60 | 20 |
| MV_SliceOpening | AREA_HEADBOX | 0.1 | 10 | 5 |
| MV_MachineSpeed | AREA_HEADBOX | 1 | 1500 | 500 |
| QCS_BoneDryWeight | AREA_REEL | 0.3 | 50 | 5 |
| QCS_BasisWeight | AREA_REEL | 0.3 | 50 | 5 |
| QCS_Moisture | AREA_REEL | 0.1 | 10 | 5 |
| QCS_Caliper | AREA_REEL | 0.2 | 10 | 5 |

Name

Description

AREA_HEADBOX

Noise

Max

Min

Add row

Delete row

Submit

Clear table

Submitted successfully

Output Variables

These are the output variables that form the final set of columns in the final dataset. The specific data in these variables are expanded upon the in 'Lab Variables' page.

The values for each variable consist of a: name, description, range and noise. The noise is an additional value that gets entered into a random number generator and added to each cell value in the final dataset for all output variables. This can be set to 0 and will result in the same value for many cell values.

If the configurations were loaded, this is what should appear as the default values. Press submit to confirm:

Home

Process Configuration

Process Variables

Process Area Descriptions

Input Configuration

Input Variables

Input Validations

State Configuration

State Variables

Output Configuration

Output Variables

Lab Variables

Generate Data

Generate

Output Configuration

| Name | Description | Noise | Max | Min |
|----------------|-------------|-------|-----|-----|
| Lab_Tensile | AREA_REEL | 2 | 100 | 50 |
| Lab_Tear | AREA_REEL | 0.5 | 100 | 50 |
| Lab_Fold | AREA_REEL | 1 | 100 | 50 |
| Lab_Stiffness | AREA_REEL | 0.05 | 100 | 50 |
| Lab_Burst | AREA_REEL | 1.2 | 100 | 50 |
| Lab_RingCrush | AREA_REEL | 2 | 100 | 50 |
| Lab_Brightness | AREA_REEL | 1 | 100 | 50 |
| Lab_Opacity | AREA_REEL | 1 | 100 | 50 |

Name

Description

Noise

Max

Min

Add row

Delete row

Submit

Clear table

Submitted successfully

Lab Variables

These entries are used in the calculations of the output variables. This section requires the user to select an output variable and assign at least one input/state variable with graph configurations.

The values for each entry consist of a: an output name, input/state variable name, weight, asymptote, order, slope, model, direction, shape. For the gain variables (model, direction, shape), you can click on the question mark button beside the dropdown boxes to bring up a pop up window detailing what each value means.

The weight values in each output set must add up to 100, and the asymptote and slope values are optional.

If the configurations were loaded, this is what should appear as the default values. Press submit to confirm:

Home

Process Configuration

Process Variables

Process Area Descriptions

Input Configuration

Input Variables

Input Validations

State Configuration

State Variables

Output Configuration

Output Variables

Lab Variables

Generate Data

Generate

Output Lab Configuration

| Ouput | Variable | Weight | Asymptote | Order | Slope | Model | Direction | Shape |
|-------------|-----------------|--------|-----------|-------|-------|-------|-----------|-------|
| Lab_Tensile | MV_FillerFlow | 5 | | 2 | | 0 | 0 | 1 |
| Lab_Tensile | MV_PressLo... | 5 | | 2 | | 0 | 1 | 0 |
| Lab_Tensile | PulpEye_Ble... | 10 | | 1 | 7 | 1 | 1 | 1 |
| Lab_Tensile | PulpEye_Ble... | 5 | | 2 | | 0 | 0 | 1 |
| Lab_Tensile | PulpEye_Dirt... | 5 | | 2 | | 0 | 0 | 1 |
| Lab_Tensile | MV_SWPct | 5 | | 1 | 4 | 1 | 1 | 0 |
| Lab_Tensile | MV_OCCPct | 5 | | 2 | | 0 | 0 | 1 |
| Lab_Tensile | QCS_Basis... | 20 | | 1 | | 0 | 1 | 0 |
| Lab_Tensile | QCS_Moisture | 10 | | 2 | | 0 | 0 | 1 |
| Lab_Tensile | PulpEye_Ble... | 10 | | 2 | | 0 | 1 | 0 |
| Lab_Tensile | PulpEye_Ble... | 20 | | 2 | | 0 | 1 | 1 |
| Lab_Tear | MV_FillerFlow | 10 | | 2 | | 0 | 0 | 1 |
| Lab_Tear | MV_PressLo... | 10 | 600 | 2 | | 0 | 1 | 2 |
| Lab_Tear | PulpEye_Ble... | 5 | | 2 | | 0 | 1 | 1 |

Output Lab_Tensile

Variable MV_WireSpeed

Weight

Asymptote (can be empty)

Order 1

Slope (can be empty)

Gain Model 0 ?

Gain Direction 0 ?

Gain Shape 0 ?

Add row

Delete row

Submit

Clear table

Submitted successfully, ready to generate data

Generate Data

All the configurations have been submitted at this point, so the data is ready to be generated. If you entered configurations manually and missed any required variables, the names will show when you click in the Generate Data button. If not, the dataset generation should start as shown below:



Once the dataset has been generated, the time will be displayed and there will be an option to download the current configurations into a folder you can name in the 'config' directory. The

generated dataset will be found in the 'data' folder, inside the 'generator/main' directory, in a CSV file with the name of the current date and time.

Home

Process Configuration

Process Variables

Process Area Descriptions

Input Configuration

Input Variables

Input Validations

State Configuration

State Variables

Output Configuration

Output Variables

Lab Variables

Generate Data

Generate

Generate Data

Generate Data

Dataset generated in the /data folder

Time taken: 0 hours 4 minutes 49 seconds

Download Configurations

Please enter a folder name:

Select Folder

Developer Documentation

Introduction

This application is written for the paper industry, and the code includes specific calculations for variables that are entered. This section of the documentation will go through the methods to explain what is specific to this industry so you are able to configure to your own needs. Suggestions to improve the application are also mentioned.

There are three classes for this application located in the following files under 'generator/main/src': 'Main.java', 'Form.java' and 'Generator.java'. 'Main' initialises the 'Form' class. 'Form' creates the user interface and calls the 'Generator' class. 'Generator' consists of the data generation methods. 'Form' and 'Generator' are the classes that would most likely require adjusting for specific needs. The methods referred to in this chapter are in the 'Generator.java' file, unless otherwise stated. These files are fully commented, so it is worth going through if you plan to change the code.

One important note to remember however is that in 'Form.java', there is a check performed in the 'genButtonActionPerformed()' method where an array of required input and state variables are checked against the user entered variables to ensure the variables have been created. This array would need adjusting in the method if you are intending to add/remove any required variables.

It is advised to read through the 'User Documentation' section first to understand how the application runs.

A suggestion to increase the efficiency of this application could be to modularise the code, so the user could run only some of the generation methods instead of generating all the data again from the start.

Process Configuration

The following process variables are specific to the paper industry: QCS Period, PulpEye Period, Lab Period, Trim, Draw.

Time Periods

Additional time periods may be required for another industry with different names and properties. A suggestion to resolve this issue could be to: create a table to specify process periods where names can be assigned and the created periods can be assigned to input, state and output variables. The condition that time periods are a multiple of the Process Period must hold, however.

The specific period values are used to determine how often the rows of data should appear. The QCS and PulpEye periods are used in the 'createDataset()' method to determine which rows should be cleared from the related variables. The Lab Period is used in the 'calcLab()' method where the row clearing is performed earlier.

Measurements

The Trim and Draw variables are measurements to assist in specific industry state variable calculations. Trim is used in 'calcState()' to calculate 'MV_SliceOpening' and is also used in 'calcQCS()' to calculate 'QCS_BoneDryWeight'. Draw is used in 'calcState()' to calculate 'MV_MachineSpeed'.

Input Configuration

Process input variables are calculated in the 'createInputs()' method. Under the 'Input Variables' section of the user documentation, certain variable names are listed which are required for the calculations of some of the state variables.

The structure of the 'createInputs()' method is as follows:

- Calculation of steady state rows: required for the deadtimes.
- Calculation of staggered/uncoupled rows: using input steps.
- Calculation of custom/validation rows: user configured steps.

This structure is not industry specific. However if no changes are made to the state configuration methods mentioned in the following section, there are a set of variables which are required to be included in the input configuration or else the generation will not run:

- MV_WireSpeed
- MV_SWSpecificEnergy
- MV_HWSpecificEnergy
- MV_OCCSpecificEnergy
- MV_SWFlow
- MV_HWFlow
- MV_OCCFlow
- MV_ThinStockConsistency
- MV_ThinStockFlow
- MV_PressLoad
- MV_SteamPressure

- MV_JettoWire
- PulpEye_SWCrill
- PulpEye_HWCrill
- PulpEye_OCCCrill

One suggestion to add to the input configurations could be to include isolated steps. This could be achieved by including another set of rows before the validation rows, where an input is stepped all the way from lowest to highest with all other inputs at their average. Then that input is returned to the average and the next input is stepped.

Another suggestion could be to implement the sorting of order values in the input configuration page. There could be a button to automatically sort the order values, or the code could swap an order value when the user updates it from the dropdown box.

State Configuration

State variables are those that can be calculated based on engineering principles, such as mass or energy balance. Input variables are used in these calculations. The resulting state variable is dependent upon those inputs used in the calculations. They differ from outputs only in the respect that there are known calculations for deriving them, whereas output variables have known curves but not explicit equations.

An example of a state variable can be moisture ('QCS_Moisture' in the configurations/code). Inputs can tell us the amount of dry material and water, the amount of drainage and heat applied in drying, and the resulting moisture can be calculated from these inputs. We can also apply dynamics so that the relationship of the inputs to the resulting moisture incorporates deadtime and lags.

In this case, all the variables listed under the 'State Variables' section of the user documentation contribute to all the state variables required for this dataset for the paper industry. All the calculations in the 'calcState()' and 'calcQCS()' methods are unique to the industry and would need to be modified if your purposes are different.

If no changes are made to the state configuration methods mentioned in the next section, there are a set of variables which are required to be included in the state configuration or else the generation will not run:

- MV_SWFreeness
- MV_HWFreeness

- MV_OCCFreeness
- MV_HeadboxPressure
- MV_SliceOpening
- MV_MachineSpeed
- MV_SWPct
- MV_HWPct
- MV_OCCPct
- PulpEye_BlendFreeness
- PulpEye_BlendCrill
- QCS_BoneDryWeight
- QCS_BasisWeight
- QCS_Moisture
- QCS_Caliper

One suggestion to add to the state configurations could be to allow the user to enter state calculation on the user interface, which may be more convenient instead of having to change the code all the time.

The below section will go through what methods to change to configure unique or no state variables:

Methods to adapt for different state variables

'Generator.java':

- Firstly, you may need to change the variable fields of the class if you are not using all the process configuration values. This can be done by renaming/adding the variables fields of the class and changing the 'Generator()' constructor to initialise the variables. Any mention of those variables in any of the methods would also need to be changed.
- In the 'config' folders, the state variable configuration is called 'state.csv'. The rows of the file show the data required for each state variable. If you would like to adjust the data required, make note of the row numbers and change the variables in 'Generator.java' to match the new state table layout whenever the 'state' variable is used.
- 'calcState()' = This is the primary method that calculates a majority of state variables (including the associated helper method: 'stateSetup()'). All the calculations are unique, so the entire method can be modified to include your variables/calculations.

- 'calcQCS()' = This is another method that calculates some of the state variables (prefixed by 'QCS'). Again, this entire method can be modified to your set of specific variables if required. This method also shows an example of how dynamic values are used, so you can use it as a template to include dynamic input and state variables in your calculations.
- 'createDataset()' = This method clears some rows that are not required for the process periods. You can change variables that are assigned to the 'numRows' variable, as well as the conditional statements that check the prefix of the variable, for example:

```
int numRows;
if (name.contains("QCS")) {
    numRows = qcsPeriod;
}
```

'Form.java':

- If any of the process variables were changed in 'Generator.java', change the following methods to reflect it on the UI: ' initComponents()', 'load()', 'submitProcessButtonActionPerformed()', 'download()'.
- If the layout of the state variable data was changed (for example the state configuration table required different data for each entry), change the following methods to reflect it on the UI: ' initComponents()', 'load()', 'stateAddButtonActionPerformed()', 'stateSubmitButtonActionPerformed()', 'load()', 'download()'.
- 'genButtonActionPerformed()' = This method goes through a list of required input and state variables that will prohibit data generation unless the variables have been entered. The input variables currently included contribute to the calculations of some state variables. Change the array variable 'variables', where this list is present.
- 'run()' = This method runs the generation methods from a 'Generator' object. The line that calls the 'calcQCS()' method can be removed or adjusted if it is not required:

```
gen.calcQCS();
```

Methods to adapt for no state variables

'Generator.java':

- 'calcLab()' = This method calculates output variables but includes some lines that would need to be commented if you do not want to include state variables. This includes the lines:

```
List<Integer> stateNames = new ArrayList<>();
calcList(stateNames, numState, state);
and
```



```
for (int state : stateNames) {
    dynamicValues(j, state, false);
}
```

- 'gainModel()' = This method applies a check to see if variables are from the input or state table. The following block of code can be changed from this:

```
if (searchCol(varName, data) < sRow) {
    col = searchCol(varName, input);
    max = Double.parseDouble(input.get(8, col));
    min = Double.parseDouble(input.get(9, col));
}
else{
    col = searchCol(varName, state);
    max = Double.parseDouble(state.get(7, col));
    min = Double.parseDouble(state.get(8, col));
}
```

to this:

```
col = searchCol(varName, input);
max = Double.parseDouble(input.get(8, col));
min = Double.parseDouble(input.get(9, col));
```

'Form.java':

- 'inputSubmitButtonActionPerformed()' = In the final else statement, add the following lines to the end:

```
processDescSet = true;
stateSet = true;
```

- 'labButtonActionPerformed()' = Comment out these lines:

```
else if (state.isEmpty())
    dialog("Please fill out the state configurations", "Output
Error");
```

- 'genDataButtonActionPerformed()' = Comment out these lines:

```
else if (state.isEmpty())
    dialog("Please fill out the state configurations", "State Error");
```

- 'genButtonActionPerformed()' = Comment out this line:

```
listTextArea.append(var + "\n");
```

- 'run()' = Comment out these lines:

```
gen.calcState();
```

- And:

```
gen.calcQCS();
```

- As long as you don't press the submit button on the state configuration page, state variables won't be included/required after making these changes.

- If you do not adjust the output lab configuration to remove the state variables, there will be an error in the code so be aware that state variables must be completely removed from all configurations.

Dynamics

A subset of input and state variables are required to be dynamically moved after steady state rows and those new values form part of the calculations in the 'calcQCS()' and 'calcLab()' methods. The process area description values assigned to input and state variables are used in the 'dynamicValues()' method, which dynamically moves the variable for each row. The method is called on the row where the calculation takes place and are overwritten on the next row, so not all dynamic values need to be stored in memory. This is not specific to industry, however you may need to modify the code when determining which variables require dynamics.

Output Configuration

Output configurations are calculated in 'calcLab()', which takes the values of the lab configurations to calculate each output value. The methods 'gainModel()' and 'gainFunction()' are called to apply these calculations. The dynamic values of input and state variables that create the rows of the lab configurations are used.