Documentation Dewar Monitoring C++ Implementation

1. Quantile based change detection

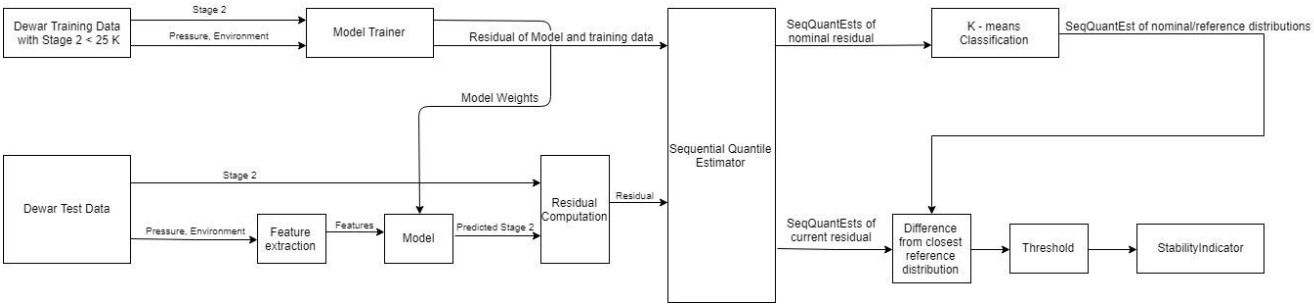
- The *Dewar Quantile Based Change Detection* scheme works by applying a threshold on the distance between the probability distribution of Stage 2 residual and the closest nominal distribution at every time sample in order to obtain an estimate for the state of the dewar cryogenic system.

- The Stage 2 residual is defined to be the difference between the true and the predicted Stage 2 value. The measured values, obtained from the installed sensor are considered to be true Stage 2 values while the predicted stage 2 values are obtained from a trained model.

- The algorithm at each time sample computes the deviation of the probability distribution (PD) of the computed residual from a set of pre-derived nominal/reference distributions. If the deviation exceeds beyond a pre-defined threshold, then an alarm signal is excited.

- The probability distribution (PD) of the residual is computed using the algorithm of *Sequential Quantile Estimator* which is obtained from [1]. The sequential estimator block depends on information from only one previous sample.

2. Flow Chart



- The model is derived by running a levenberg marquardt optimization to fit a polynomial over a *nominal* Stage 2 temperature data set. Nominal Stage 2 was defined to be all the Stage 2 readings that fall below 25 K.

- The feature space for the model polynomial is derived from Pressure and Environment temperature measurements.

- After the model is trained for least rms difference, a *nominal residual* data set is obtained by subtracting the predicted (obtained from model) and true stage 2. Keep note that as this is done for the nominal Stage 2 data set, the computed residual data set can be considered to be ideally generated from all probability distributions that themselves can be termed as *nominal distributions*.

- The *nominal distributions* reflect a stable state of the dewar system. One can expect a significant deviation of the PD of dewar from this set of nominal distributions if the dewar is approaching an unstable state at that given time instant.

- By setting a threshold on this deviation, one can gauge the stability condition of dewar.

3. Classes

DewarHandler.cpp

The code subject to the dewar quantile processing is organized within the DewarHandler class. This class defines certain properties that are associated with the quantile processing stage and member functions that update these properties. The DewarHandler class is instantiated in the main routine and is invoked for update for each time sample.

Properties

The following are the declared dewar properties with WINDOWSIZE = 1000 and NOOFQUANTS = 11 :

```
double S2;      // Stores current value of Stage 2 temperature
double P;       // Stores current value of Pressure
double ET;      // Stores current value of Environment temperature

double S2_series[WINDOWSIZE];  // Stores Stage 2 values for complete WINDOWSIZE, initialized with
predefined values
double P_series[WINDOWSIZE];   // Stores Pressure values for complete WINDOWSIZE, initialized with
predefined values
double ET_series[WINDOWSIZE];  // Stores Environment values for complete WINDOWSIZE, initialized
with predefined values

float f1_series[WINDOWSIZE];   // Stores normalized values of Pressure for complete WINDOWSIZE,
initialized with zero
float f2_series[WINDOWSIZE];   // Stores normalized values of Environment temperature for complete
WINDOWSIZE, initialized with zero

double resS2AndModel_series[WINDOWSIZE]; // Stores values of residual between Stage 2 and model for
complete WINDOWSIZE, initialized with zero

float eqn_series[WINDOWSIZE];  // Stores values of model for complete WINDOWSIZE, initialized with
zero
float dist_series[WINDOWSIZE]; // Stores values of distance of current PD with closest reference PD
for complete WINDOWSIZE, initialized with zero
float testQuantity;            // Stores cuurent value of distance of current PD with closest
reference PD

bool alarms_series[WINDOWSIZE];// Stores values of alarms for complete WINDOWSIZE, initialized with
false (no alarms)

int stabilityLevel;            // Stores current value of Stability Level (10 = Stable, 5 =
Semistable or 1 = Unstable)
char* stabilityLabel;          // Stores current value of Stability Label ('Stable', 'Semistable' or
'Unstable')

int sampleIndex;               // Stores sample index

struct SeqQuantEst{
        int n[NOOFQUANTS];
        int m[NOOFQUANTS];
        float seqQuantEst[NOOFQUANTS];
};
SeqQuantEst dewarQuantEst;     // Stores current sequential qunatile estimate (11 quantiles in 0.01
to 0.99)

Dislin g, r;    // Objects of DISLIN library to plot Dewar Data Monitoring Window and Dewar
Probability Distributions Window

struct Features{
        float f1;       // Normalized pressure
```

```
        float f2;       // Normalized environment temperature
        float f4;       // Normalized distance of environment temperature from threshold
        float f5;       // Normalized environment temperature without seasonal variation
        float y0;       // Stage 2 temperature
};
Features features;      // Stores the current values of normalized features for the model
```

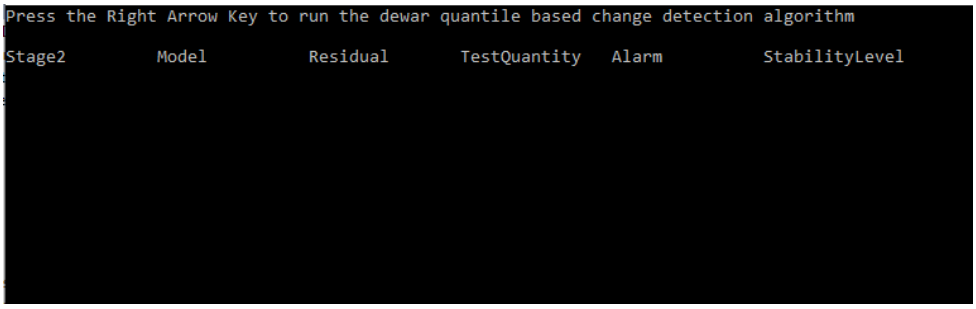
Member Functions

Following are the primary member functions of the DewarHandler class which are supported by some auxiliary functions that are not declared here:

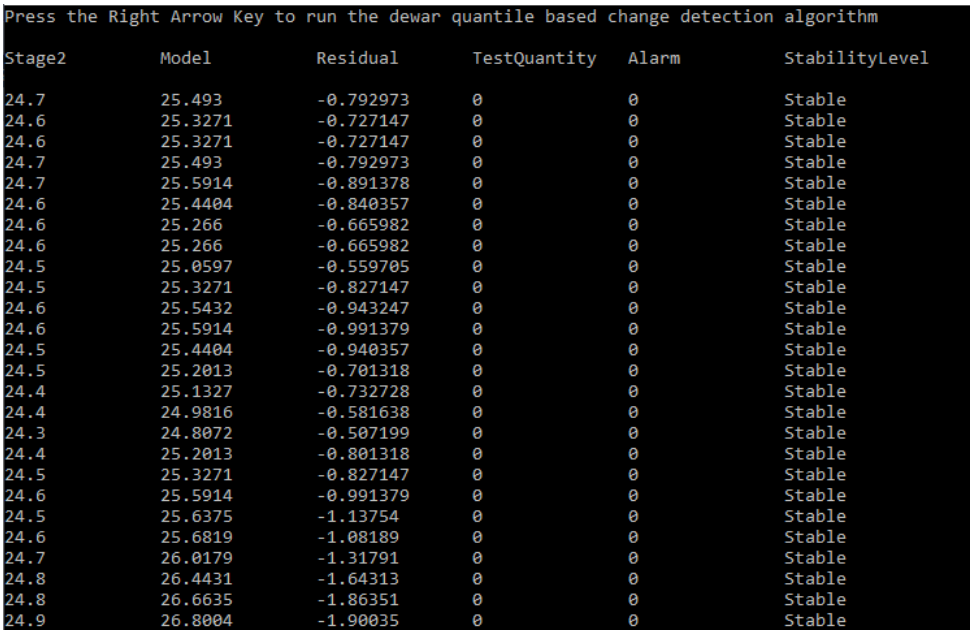
```
void initDewar();       // Initializes dewar properties

void updateDewarObj(); // Updates dewar object , called at every time sample in the main routine

void updateData();      // Updates dewar data properties, called in updateDewarObj()
void updateStabilityLevel(); // Updates dewar stability level and label, called in updateDewarObj()
void updateDataPlot();       // Updates Dewar Data Monitoring Window, called in updateDewarObj()
void updateResPlot();  // Updates Dewar Dewar Probability Distributions Window, called in
updateDewarObj()

void plotData(double series1[], char* color1, float series2[], char* color2, double series3[], char*
color3, float series4[], char* color4, bool series5[], char* color5);// Plots dewar data
properties, called in updateDataPlot()
void plotResData(double tau[], double centroid1[], double centroid2[], double centroid3[], double
centroid4[], double centroid5[], float seqQuantEst[]);  // Plots dewar probaility distribution and
reference distributions, called in updateResPlot()

void normalizeFeatures();      // Normalizes features, called in updateData()
void seqQuantEstimator(double res, double quantEst, int n, int m, double tau, int N, int
indexOfQuant);  // Estimates dewar probability distribution (quantile estimate), called in
updateData()
```

4. Dewar C++ Project Outputs

After building and running the DewarQuantileChangeDetection VS project for a Solution Platform of x64 bit, the following console output should appear:



On pressing the Right Arrow Key, the dewar object processing takes place and the results are displayed as the following:

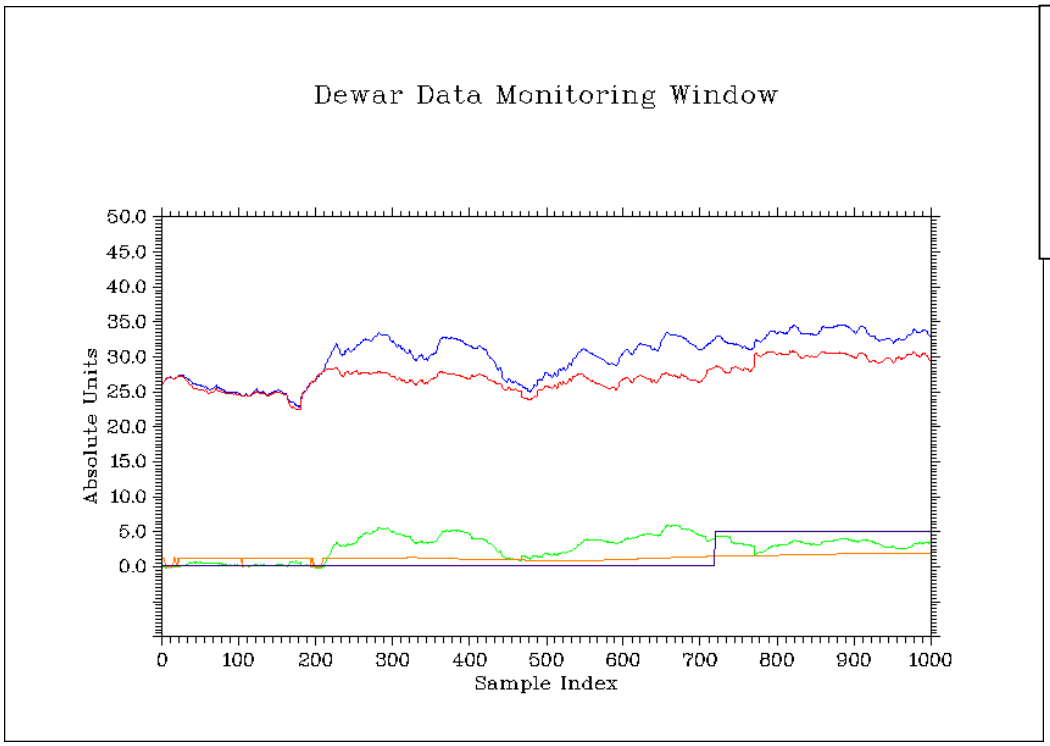


Stage2: Current value of stage 2 temperature (K)
Model: Current value trained model
Residual: Algebraic difference between Stage2 and Model, dewar processing only excited if this is greater than 0 i.e Stage 2 > Model
TestQuantity: Distance of probability distribution of dewar from closest reference distribution. Threshold for alarm is set to be 1.5
Alarm: Alarm signal, takes value of either 0 (no alarm) or 1 (alarm)
StabilityLevel: Indicates stability state of dewar

The Dewar Data Monitoring Window and Dewar Probability Distribution Window are generated at each time sample and saved as DewarDataPlot.png and Distribution Plot.png in the project folder.

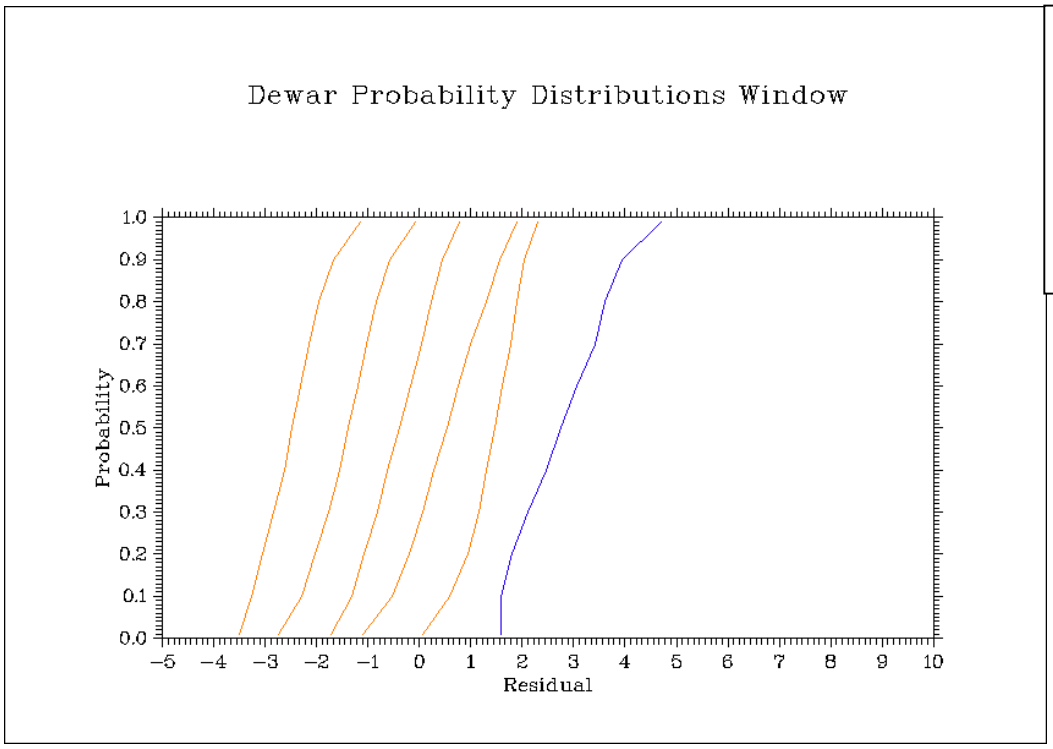
DISLIN Output Plots

The DISLIN C++ library is used to plot the *Dewar Data Monitoring Window* and the *Dewar Probability Distributions Window.* These windows provide a visual basis for analysis of dewar stability state.



Legend

Blue: Stage 2
Red: Model
Green: Residual
Orange: Distance Series
Gray: Alarm



Legend

Orange: Reference probability distributions

Red: Dewar probability distribution

5. References

[1] Daniel Jung, Erik Frisk, MattiasKrysander, 'Residual change detection using low-complexity sequential quantile estimation'