Accurate Automation of Deadweight Force Standard Machine Based on Artificial Intelligent Model at NMCC-SASO-KSA

Hamad A. AlGhamdi*, Abdulelah A. Binown, Ahmed A. AlMatrawi

Saudi Standards, Metrology & Quality Org. – National Measurements & Calibration Center (SASO-NMCC)
Riyadh, Kingdom of Saudi Arabia
*h.gamdi@saso.gov.sa , a.binown@saso.gov.sa , a.matrawi@saso.gov.sa

Abstract

Accurate force measurements are critical in a various fields including metrology, material testing, and quality control. In this study, we propose an automated process for precise force measurements using Dead weight standard machines up to 1kN. This method leverages a combination of hardware and software components to achieve reliable results. The key steps in the automation process include the calibration Setup, which consists of a carefully calibrated force transducer used as a reference, which is selected based on its traceability to national standards. Automated Loading System (ALS) is the second item of the process, in which a motorized system places a known weights on the force transducer. The loading process was controlled to minimize disturbances and ensure accurate readings. The third step is Data Acquisition and Analysis, which includes a high-precision amplifier that collect force data during loading, process the acquired data, compensate for environmental factors (e.g., temperature and humidity) based on LabVIEW, and apply statistical analysis techniques to estimate the force values using python script based on classification machine learning. The full script is open source for all reader to apply in their different fields. The proposed automation process enhances measurement efficiency, reduces, cost, time, human errors, and ensures consistent results. This contributes to the advancement of force metrology and facilitates reliable force measurements in various applications.

Keywords:

LabVIEW, Python, Force, Machine, Digitization, Automation, Load cell, calibration.

1. Introduction

The National Metrology and Calibration Center (NMCC) at the Saudi Standards Metrology & Quality Organization (SASO), is committed to enhancing, advancing, and conserving the national standards of force, unit of Newton. The force laboratory also offers testing and calibration services to diverse sectors such as steel, concrete, military industrials, and scientific research, underscoring the pivotal role of the National Metrology Institute (NMI) in upholding a robust quality infrastructure. As defined by the Bureau International des Poids et Mesures (BIPM), metrology is the science of measurement, and it's vital for measurements of physical quantities like force to be conducted with a dependable stabilization system. This is of particular interest to researchers, scientists, equipment developers, and manufacturers who are invested in accurately measuring various substances. The NMI's metrology field primarily aims for high accuracy and precision in force lab from 200 N up to 5 MN [1,2,3,4].

The digital era has seen remarkable progress in big data, cloud computing, artificial intelligence (AI), and machine learning (ML). These technologies, including ML, deep learning, and neural networks, have been successfully integrated into metrology. The surge in computing and storage capabilities, coupled with increased data transfer speeds and the widespread availability of versatile sensors, has opened up new avenues for enhancing metrology tools, such as the Python programming language. Python is a versatile language used for machine learning, web development, desktop applications, and complex mathematical computations. Its simple syntax and ease of use make it an ideal choice. Python is particularly suited for ML, AI, descriptive and inferential statistics, and advanced mathematics, offering excellent results and accuracy with minimal resources and time. It boasts an extensive library system tailored for specific tasks; for advanced mathematics, libraries focusing on algebra, matrices, mathematics, visualization, exploratory data analysis (EDA), statistics, data pipelines, and data containers are utilized. One of our challenges is to digitalize the measurements of force and automate the process completely. In next sections we will introduce the three main steps in details.

2. Measurement system

The system of measurement consists of several connected equipment that works together in harmonized techniques as shown in the figure 1.

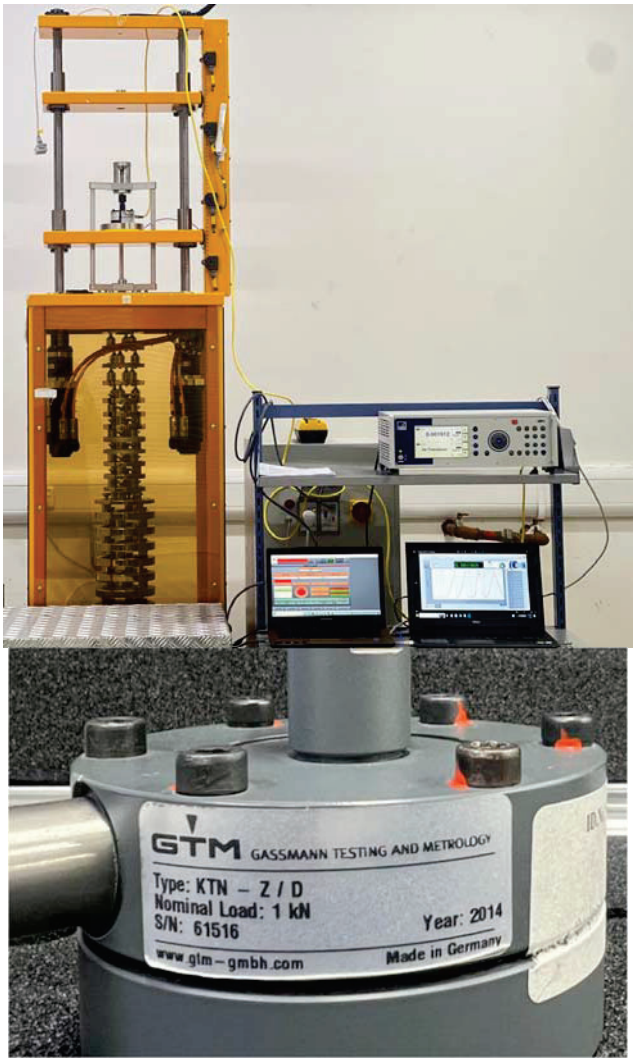


Figure 1. Force calibration Machine

2.1. Machine

Automated Loading System (ALS) is a motorized system places a known weights on the force transducer. The loading process was controlled to minimize disturbances and ensure accurate readings. Deadweight force machine code NMCC-MA-FO-003 has been used with several steps loading system [3].

Deadweight Force Standard Machines (DWMs) operate by using deadweights to measure force accurately. The weights are applied to a mechanical system to adjust the force. Weights are gradually added using precise motors, which automate the process of weight increment. The weights are either moved or adjusted by the motor to apply known amounts of force. Each weight represents a fixed value, such as 10 N or 50 N. The weights are added in successive steps to achieve the desired force. The calibration process with DWMs involves a series of precise steps to adjust and document the resulting force. During this process, 325 steps of commands (such as weight adjustments or motor modifications) are executed, while 81 precise readings are recorded, representing the measured force during calibration. Transducer calibration is carried out according to the ISO 376 standard, the transducer is connected to an indicator reading device, and then a known force (N) is applied using the DWMs. The resulting values are compared with reference values for accuracy. In order to convert the output signal from mV/V to Newton using a third-degree equation, it is essential to account for the relationship between the measured voltage (in mV/V) and the force (in Newton). A third-degree polynomial equation is commonly used when the relationship between voltage and force is nonlinear and requires precise correction to provide more accurate results, equation 1.

$$F = A\left(\frac{mV}{V}\right)^3 + B\left(\frac{mV}{V}\right)^2 + C\left(\frac{mV}{V}\right) + D \qquad 1$$

Where:

- F is the required force in Newtons (N).
- mV/V is the transducer output.
- A, B, C, D are constants determined through transducer calibration.

2.2. Measuring System

The measuring standard device used to acquire the reading data is DMP41 SN. 820669101. The DMP41 is a digital precision measuring amplifier known for its high accuracy and precision. It's often used in metrology specially in force measuring systems of strain gauge transducers, as well as transducers linked to weight, force, pressure, or torque. The Accuracy class is 0.0005 with resolution 6 digits and Long-term stability: ±5 ppm. DMP41 equipped with connectivity options: Ethernet, RS-232, USB.

3. Modeling and Algorithms

The automation model based on two main communicated programs, LabVIEW for automatic acquiring the data continuously from the DPM41 and store the collected pipeline of data to csv file and the second one is Python script that is coded with machine learning techniques based on classification and supported vectors.

3.1. LabVIEW

LabVIEW (**L**aboratory **V**irtual **I**nstrument **E**ngineering **W**orkbench) is a graphical programming environment created, it's extensively utilized for applications involving testing, measurement, and control, enabling quick access to hardware and data insights. The program consists of two main parts, the front Pannel (figure 2) and block diagram (figure 3).

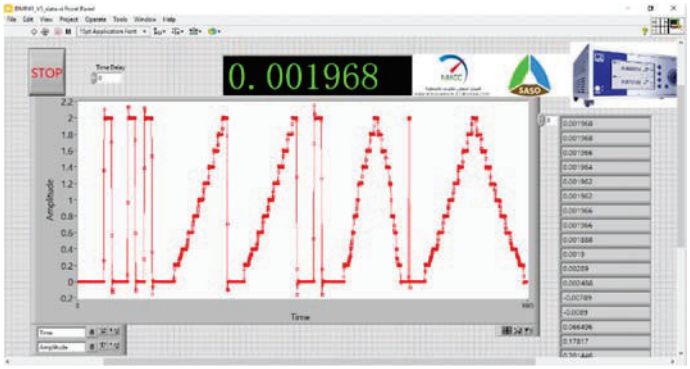


Figure 2. GUI Front Pannel

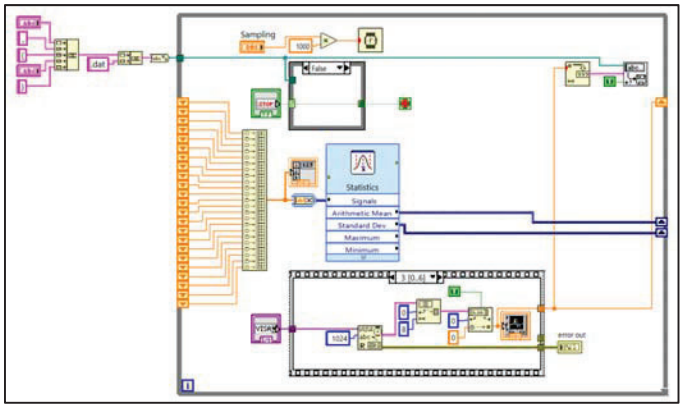


Figure 3. Block Diagram

The front panel consists of an indicator of live measurement, displaying chart for plotting the all acquired data and finally the last 50 consecutive readings with its mean, mode, median and standard deviation. The block diagram is the core of the program, it is start with LAN connection through VISA from DMP41 to the string reading (VISA-Read). The used code for this purpose is "MeAs?". The reading string handled with spitted technique again to separate the variable number into readable numeric. Finally, the numeric is acquired to csv file, connected to display and indicator in the front panel.

3.2. Python

Python is a versatile programming language known for its readability and simplicity. It is widely used for web development, data analysis, artificial intelligence, scientific computing, and more. Python is a popular choice for developing artificial intelligence (AI) applications due to its simplicity and the extensive libraries available for AI and machine learning [**5-13**]. We decided to use pythonic language in order to overcome the streamline pipeline of data that contains unwanted-polluted data (figure 4a) and scattered noise outliers (figure 4b). The challenge is how to extract only a specific step with its standard deviation and mean. This value is coming from a huge scattered data that reach in some cases to more than ten thousand of data.

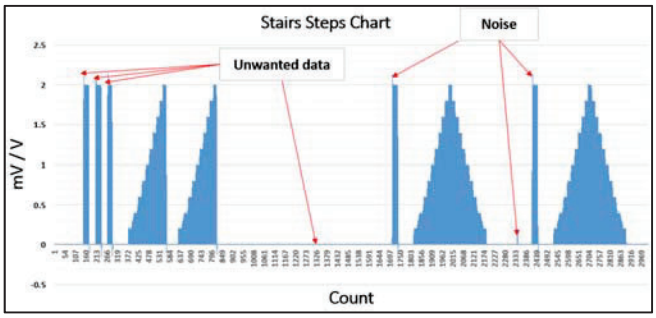


Figure 4.a Polluted data

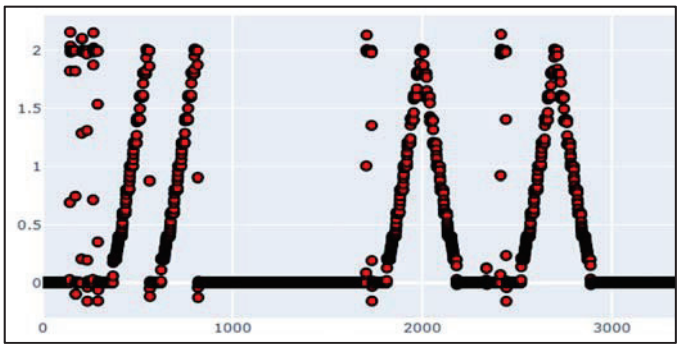


Figure 4.b Scattered data with outliers

The python script work under classification of data technique which is consider one of the strongest tools in machine learning. The full open script source in (Annex A),

4. Results

The calibration procedure begins with preloading the force-proving instrument three times to its maximum force in the direction in which the calibration forces will be applied. If the direction of loading is changed, the maximum force must also be applied three times

in the new direction. Before applying the calibration forces, the force transducer is loaded to its maximum capacity three times, with each preload lasting between 30 to 60 seconds. After each preload, readings corresponding to no force are recorded after waiting at least 30 seconds for the force to be completely removed. A minimum of 3 minutes should pass between subsequent measurement series to allow stabilization.

In the calibration procedure, the zero signal is noted before starting. The calibration is performed by applying two series of calibration forces with increasing values, without disturbing the device. Then, at least two additional series of calibration forces with both increasing and decreasing values are applied at specific positions. These additional series are carried out after changing the angle or position of the force-proving instrument, typically at positions 0^0, 120^0, and 240^0, figure 5. The maximum force is applied each time the position changes. The time interval between successive loadings should be uniform, and no readings should be taken within 30 seconds after a force change to ensure accurate measurements, figure 5.

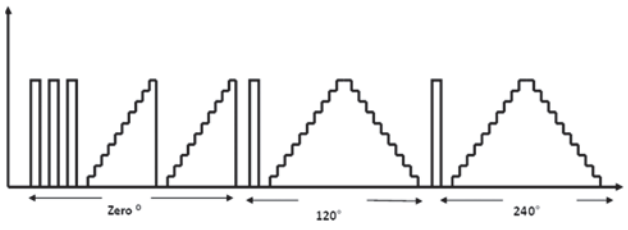


Figure 5.a Theoretical Scheme

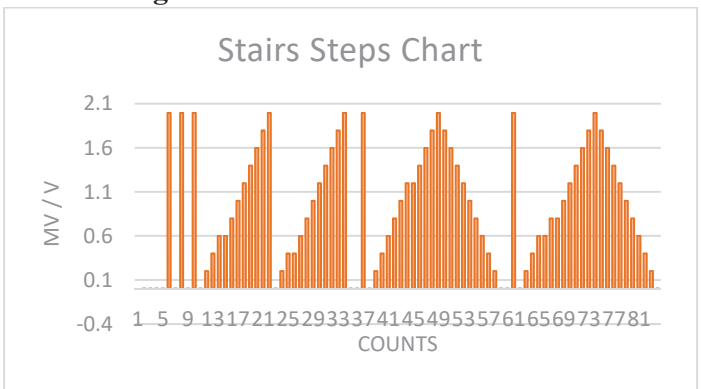


Figure 5.b clean data after classification

5. Conclusion

Machine learning model has been built in NMCC-SASO for accurate atomate the deadweights machine, the algorithm starts with acquire, handle, and stored the data as a phase one using LabVIEW. The second phase is coming with python code for selecting the proper data and extract only around 81 readings to link with the digital certificate in the soon future. The algorithm shows good performance for data ware during transition in the long series pipeline.

6. Reference

1. ISO 376:2011. "Metallic materials — Calibration of force-proving instruments used for the verification of uniaxial testing machines." International Organization for Standardization (ISO), 2011.
2. ASTM E4-16. "Standard Practices for Force Verification of Testing Machines." ASTM International, 2016. DOI: 10.1520/E0004-16
3. BIPM (Bureau International des Poids et Mesures). "International System of Units (SI)." International Bureau of Weights and Measures (BIPM), 9th edition, 2019. Available at: https://www.bipm.org/en/si
4. JCGM 100:2008 GUM. "Guide to the Expression of Uncertainty in Measurement." Joint Committee for Guides in Metrology, 2008. Available at: https://www.bipm.org/en/publications/guide
5. Jupyter Documentation (2024) retrieved from https://docs.jupyter.org/en/latest/
6. NumPy Documentation (2024) retrieved from https://numpy.org/news/
7. Pandas Documentation (2024) retrieved from https://pandas.pydata.org/docs/
8. Matplotlib Documentation (2024) retrieved from https://matplotlib.org/
9. Plotly Documentation (2024) retrieved from https://plotly.com/
10. SciPy Documentation (2024) retrieved from https://scipy.org/

11. Seaborn Documentation (2024) retrieved from https://seaborn.pydata.org/
12. statsmodels Documentation (2024) retrieved from https://www.statsmodels.org/stable/index.html
13. scikit-learn Documentation (2024) retrieved from https://scikit-learn.org/stable/

Annex-A Python Script

Import the required Liberaries

```
from dash import Dash, dcc, html, Input, Output
import plotly.graph_objects as go
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
```

Read the stored data ¶

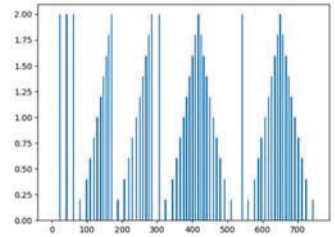
```
df = pd.read_excel(r't.xlsx')#, index_col='Date')
#df=df.dropna()
#df=df.drop(df.tail(1).index)
log = 0
```

```
app.layout = html.Div(
    [dcc.Graph(figure=fig, id='scatter-chart')]
)

@app.callback(
    Output('scatter-chart', 'figure'),
    Input('scatter-chart', 'relayoutData'),
    prevent_initial_call=True,
)
def update_title(relayout_data):
    try:
        x_min, x_max = relayout_data['xaxis.range[0]'],
        relayout_data['xaxis.range[1]']
        y_min, y_max = relayout_data['yaxis.range[0]'],
        relayout_data['yaxis.range[1]']
    except:
        avg = np.mean(y)
        min_val = np.min(y)
        max_val = np.max(y)
        y_pad = (max_val-min_val)/16
        std_val = np.std(y)
        n       = len(y)
        unc_val = std_val/np.sqrt(n)
        fig.update_layout(
            xaxis_range=[min(x), max(x)],
            yaxis_range=[min(y)-y_pad, max(y)+y_pad],
            title=f'Average: {avg:.8f},StD: {std_val:.8f}
            , n:{n}, unc: {unc_val:.8f}, Min: {min_val:.8f}
        , Max: {max_val:.8f}')
        return fig

    df_subset = df[df['x'].between(x_min,x_max)
    & df['y'].between(y_min,y_max)]
    avg     = df_subset['y'].mean()
    min_val = df_subset['y'].min()
    max_val = df_subset['y'].max()
    std_val = df_subset['y'].std()
    n       = len(df_subset['y'])
    unc_val = std_val/np.sqrt(n)
    #print(df_subset['y'].to_list())
    df_subset['y'].to_excel('out.xlsx',index=False, header=True)
    df_subset['y'].to_csv("out")
    fig.update_layout(
        xaxis_range=[x_min, x_max],
        yaxis_range=[y_min, y_max],
        title=f'Average: {avg:.8f},StD: {std_val:.8f}, n:{n}
        , unc: {unc_val:.8f}, Min: {min_val:.8f}, Max: {max_val:.8f}',)
    return fig

if __name__ == '__main__':
    app.run_server(debug=True,port=8051)
```



Criteria for filtering

```
#maximum = float(input('please enter Max Level {}'))
#minmum = float(input('please enter Min Level {}'))
criteria = float(input('Please enter the criteria {}'))
```

```
Please enter the criteria {} 0.0001
```

```
#df=pd.read_excel('t.xlsx')
#df = df[df.y>minmum]
#df = df[df.y<maximum]
```

```
data = df[df.columns[0]].to_list()
len(data)
```

```
6765
```

```
#Y=[np.mean(i) for i,j in zip(data, data[1::]) if np.abs(i-j)<0.1]
#plt.scatter(y=data,x=np.linspace(0,len(data),len(data)))
```

```
out = [] for j in range(0,len(data)-1): out[data[j]] = [[data[j],data[j+1]] if (np.abs(data[j+1]-data[j]))
```

```
tst=[]
for i in range (0, len(data)-1):
    if np.abs(data[i+1]-data[i])<criteria:
        tst.append(data[i])
    else:
        tst.append(0)

#plt.bar(np.linspace(0,len(tst),len(tst)),tst)
```

```
a = tst
split_value = 0

result = []
start = 0  # Initialize starting index for slicing

# Iterate through list to find
# indices of the split value
for i, value in enumerate(a):
    if value == split_value:

        # Add the sublist from start to the current index
        result.append(a[start:i])

        # Update start to next index after the split value
        start = i + 1

# Add the last sublist if there are remaining elements
if start < len(a):
    result.append(a[start:])
for i in range(0, len(result)):
    if result[i]==[]:
        result[i] = 0
    else: result[i] = np.mean(result[i])
result;
```

```
plt.bar(np.linspace(0,len(result),len(result)),result,width=4)
plt.show()
```