Wilberforce University

Cryogenics Python Coding Calculations For Nitrogen & Helium Loss

Anthony Brown, Supervisor: Joaquim Creus-Prats, Fermilab, Batavia, IL, USA

Abstract

Helium and Nitrogen are two significant chemical elements that are used to advance cryogenic scientific breakthroughs. Helium is a colorless, odorless, and tasteless gas that becomes liquid at (−452 °F). Nitrogen is a gas at its standard conditions; It becomes solid once temperatures reach below (-346 °F), and it boils and becomes a gas once temperatures reach above (-320 °F). During this research project I used a web-based interactive computing platform called Jupyter Lab to modify various python programs that calculate, and display helium losses and nitrogen usage from three cryo plants.

INTRODUCTION

Cryogenics is the science that addresses the production and effects of very low temperatures. The principles behind cryogenic technology involve controlling the rate of cooling. In some cases, this involves using mechanical refrigerants or even liquid nitrogen to cool materials down rapidly into sub-zero temperatures.

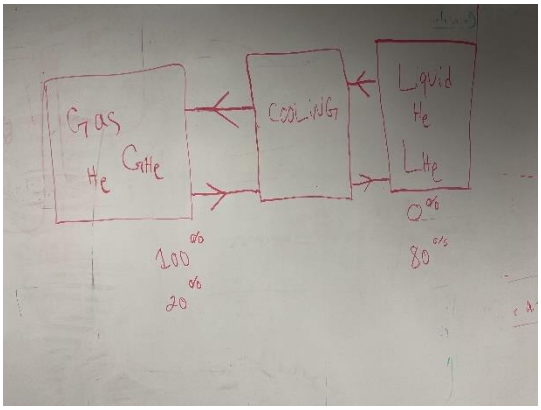

Figure 1: This image above is an example diagram of Cryogenic science that shows the gaseous state of helium having to go through a low temperature cooling phase that then converts the gas into a liquid.

Purpose/Goal Of Research

The purpose of this research project is to modify existing python codes to fetch live or historical data from any set of sensors that have data stored in ACNET D44 database. We are interested in developing one significant python program that calculate and generate the helium inventory, the nitrogen inventory, and report for three cryo plants known as CMTF, NML, and MESON. The development of this code is to further machine intelligence and efficiency.

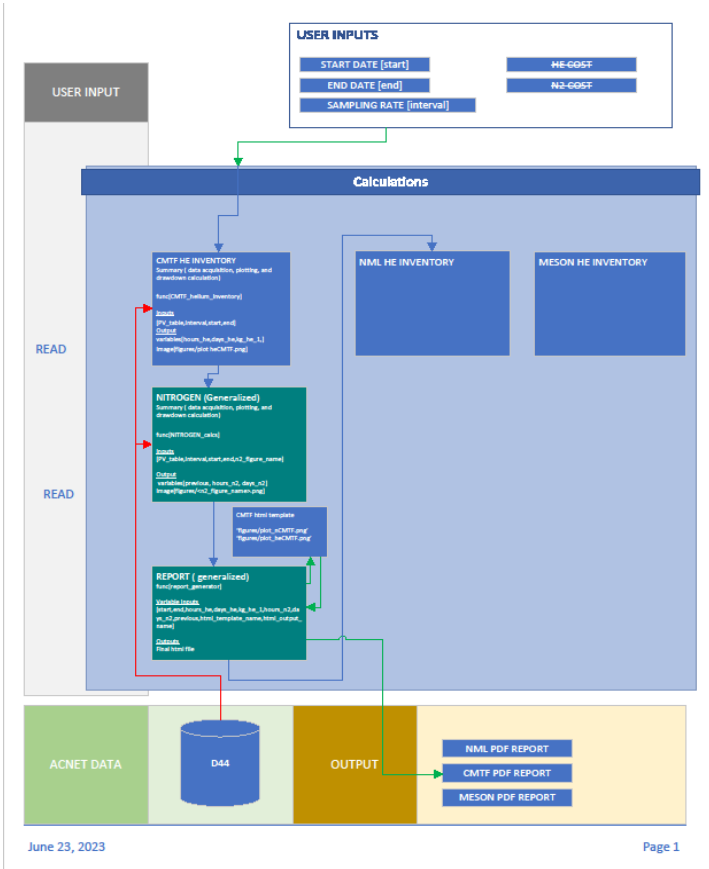

Figure 2: This image to the left is a detailed diagram that shows the order in which the sensor data

from each cryo plant will be calculated in the python code. The python script calculates CMTF's Helium Inventory, Nitrogen Inventory, then it displays the report. This process is then done for the NML and MESON cryo plants.

Characteristics of Cryo Plant Sensors

Cryogenic plant sensors are installed along the cryogenic plant to measure temperature, pressure, and fill level. Each cryo plant has various sensors with readings every second.

Nitrogen inventory			
	CONSTANTS	Units	
Nitrogen TANK max internal volume		10000 gallons	
Nitrogen TANK max level	N/A (measured in gallons)		
	ACNET DEVICE	Units	
Nitrogen TANK level	Z:CTVLND	gallons	
Gaseous Helium inventory			
	CONSTANTS	Units	
Red storage tank internal volume		30000 gallons	
	ACNET DEVICE	Units	
GHe Helium Red Storage Tank Pressure	Z:CTPRST	psig	
	CONSTANTS	Units	
Brown storage tank internal volume		30000 gallons	
	ACNET DEVICE	Units	
GHe Helium Brown Storage Tank Pressure	Z:CTPBST	psig	
Temp sensor	'M:OUTTMP'	Fahrenheit	
Warm/Cold Helium inventory			
	CONSTANTS	Units	
Device internal volume		42.3 gallons	Used to caculate inventory
Device max level		100 %	Used to caculate inventory
	ACNET DEVICE	Units	
Device level gage	Z:CTLL11	%	Used to caculate inventory
Device + pressure gage	Z:CTTIEX	psig	Used to calculate density at atmospheric conditions
Device - pressure gage	n/a		Used to calculate density at 2K conditions (if applicable)
Device temperature gage	Z:CTTXCO	K	Used to calculate density
	CONSTANTS	Units	
Device internal volume			Used to caculate inventory
Device internal volume			Used to caculate inventory
Device max level		100 %	Used to caculate inventory
	ACNET DEVICE	Units	
Device level gage	Z:HSLLS1	%	Used to caculate inventory
Device + pressure gage	Z:HSPAS6	psia	Used to calculate density at atmospheric conditions
Device - pressure gage	Z:HSPTS6	Torr	Used to calculate density at 2K conditions (if applicable)
Device temperature gage	Z:HSTXS6	K	Used to calculate density
Device internal volume	TBD	gallons	Used to caculate inventory
Device max level			Used to caculate inventory
	ACNET DEVICE	Units	
Device level gage			Used to caculate inventory
Device + pressure gage			Used to calculate density at atmospheric conditions
Device - pressure gage			Used to calculate density at 2K conditions (if applicable)
Device temperature gage			Used to calculate density

Figure 3: This image to the left is an excel chart that provide the sensor names and units for both the helium and nitrogen tanks in the MESON cryo plant.

MESON Nitrogen Tank Sensor: "Z: CTVLND"
MESON Helium Tank Sensors: "Z: CTPRST, Z: CTPBST, Z: CTLL11, Z: HSLLS1"

NML Nitrogen Tank Sensor: "N: 1DPDW"
NML Helium Tank Sensors: "N: 1PT540, N: 3LL90, N: C1LLHD, N: C2LLHD, N: SLLDEW, N: NLLDEW, C: 2PT43"

CMTF Nitrogen Tank Sensor: "C: 6LLDWN"

CMTF Helium Tank Sensors: "C: 5LL80H, C: 3LL191, T: CLL301, P: HB6LT041, P: S1LL10H, N: 1PT580, N: 1PT570, N: 1PT560, N: 1PT550"

Characteristics of The Accelerator Control Network (ACNET)

The Accelerator Control Network also known as ACNET is a system of computers that monitors, controls and stores process data in the accelerator complex. It is interfaced to users through consoles in the MCR and elsewhere. ACNET played a significant role in this research project as the origin of our data requests as well as a verification tool. This application is also used to set each sensor to a fixed interval number that allows all data to be collected at that specific moment in time.

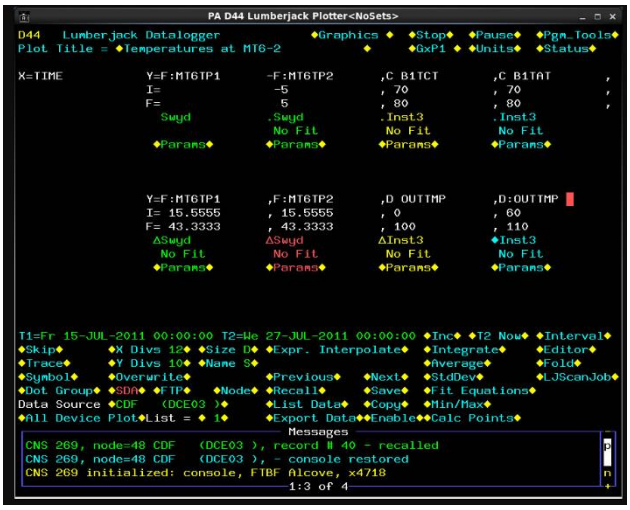

Figure 4: This image shows a visual representation of how ACNET D44 page is set up. First you would input your sensor values where the white text is located. Second, you would synch time intervals to each sensor. For instance, synching time intervals means all values would capture data 30 seconds at a time. Then once all your data is inputted, you check your graph to see if any data was captured by the sensors.

MESON's Helium & Nitrogen Python Script

```
[1]: from libraries import cryo_daq  # Importing the cryo_daq library
     from datetime import datetime  # Importing the datetime module
     import pandas as pd  # Importing the pandas library
     from matplotlib import pyplot as plt, dates as mdates, collections as c  # Importing varia
     import numpy as np  # Importing the numpy library

     # Start and End Data range to be called
     start='19.06.2023 08:30:00,00'  # Setting the start date and time
     end='26.06.2023 08:30:00,00'  # Setting the end date and time

     # Calculating the Helium inventory at Meson testing area

     # Sensors devices at Meson testing area
     PV_table=['Z:CTPRST','Z:CTPBST','Z:CTLL11','Z:HSLLS1','M:OUTTMP']

     # We are using 30 second intervals to track the data
     interval='30000'  # Setting the interval to 30 seconds (converted to milliseconds)

     datalogger=cryo_daq(PV_table,start,end,interval)  # Creating a cryo_daq object with specif

     # Calculate the Liquid Volume In Meson Inventory
     #print(datalogger)
     datalogger['InventoryResult'] = 0  # Initializing 'InventoryResult' key in datalogger obje
     def calculate_liquid_volume(LHe_density, conversion_factor, percent_filled, datalogger):
         # Calculate the liquid volume
         volume = LHe_density * conversion_factor * percent_filled / 100 * datalogger  # Calcul
         return volume

     LHe_density = 150  # Density of the liquid in kg/m^3
     conversion_factor = 0.00378541  # Conversion factor from gallons to cubic meters
     percent_filled = 42.3  # Volume in gallons for Volume 1 (Mark III dewar)


     # Calculating helium inventory for Volume 1 Dewar
     datalogger['Volume1']= calculate_liquid_volume(LHe_density, conversion_factor, percent_fil

     datalogger['Z:HSLLS1']  # Accessing the 'Z:HSLLS1' data from the datalogger object
     datalogger['InventoryResult'] = 0  # Initializing 'InventoryResult' key in datalogger obje
     def calculate_liquid_volume(LHe_density, conversion_factor, percent_filled, datalogger):

         # Calculate the liquid volume
         volume = LHe_density * conversion_factor * percent_filled / 100 * datalogger  # Calcul
         return volume

     LHe_density = 150  # Density of the liquid in kg/m^3
     conversion_factor = 0.00378541  # Conversion factor from gallons to cubic meters
     percent_filled = 42.3  # Volume 1 Dewar in gallons
```

Figure 5: This code above calculates the helium inventory at the MESON testing area using data collected from various sensors in its Helium cryoplant. The script contains start and end data ranges which specifies data from certain time periods throughout the day. The 'PV_table' variable contains the names of the sensor devices at the MESON testing area. These sensors measures various parameters within the system. This formula "volume = LHe_density * conversion_factor * percent_filled / 100 * datalogger" was then added to the script to calculate the amount of liquid Helium present.

```
[7]: import pandas as pd
     from matplotlib import pyplot as plt, dates as mdates, collections as c
     import numpy as np

     PV_table=['Z:CTVLND'] #sensor in the Liquid Nitrogen tank at the Meson testing area
     interval='30000' # Seconds are converted into milliseconds

     datalogger_N2 = cryo_daq(PV_table, start, end, interval)

     # Conversion from gallons to cubic feet
     def convert_gallons_to_cubic_feet(gallons):
         return gallons * 93.11

     # Calculate nitrogen cost
     def calculate_nitrogen_cost(gallons):
         cubic_feet = convert_gallons_to_cubic_feet(gallons)
         cost = cubic_feet * 0.53/100  # $0.53/HCF
         return cost

     # Retrieve nitrogen quantity in gallons from previous calculation
     gallons = 42.0  # Replace with your actual value

     # Convert gallons to cubic feet
     cubic_feet = convert_gallons_to_cubic_feet(gallons)
     cubic_feet_rounded = round(cubic_feet, 2)

     # Calculate nitrogen cost
     cost = calculate_nitrogen_cost(gallons)
     cost_formatted = "${:,.2f}".format(cost)

     plt.rcParams["figure.figsize"] = [20, 10]
     plt.rcParams["figure.autolayout"] = True

     d = datalogger_N2.index.values
     y = datalogger_N2['Z:CTVLND']
     s = pd.Series(y, index=d)

     fig, ax = plt.subplots()

     xval = mdates.date2num(s.index.to_pydatetime())
     p = np.array([xval, s.values]).T.reshape(-1, 1, 2)
     s = np.concatenate([p[:-1], p[1:]], axis=1)
     lc = c.LineCollection(s, cmap="gray")
     #lc.set_array(xval)

     ax.add_collection(lc)
     ax.xaxis.set_major_locator(mdates.MonthLocator())
     ax.xaxis.set_minor_locator(mdates.DayLocator())
```

Figure 6: This code above calculates the Nitrogen inventory at the MESON testing area. The script also generates a plot thst shows Liquid Nitrogen loss over a certain period of time. The variable 'interval' is set to '30000', which means that data is collected at 30 second intervals and converted into milliseconds. This function 'convert_gallons_to_cubic_feet' takes a volume in gallons as input and returns the equivalent volume in cubic feet, using a conversion factor of 93.11. The variable gallons is set to 42.0. The code then calculates the equivalent volume in cubic feet using the 'convert_gallons_to_cubic_feet' function and rounds it to two decimal places, storing it in the cubic_feet_rounded variable. This conversion factor was added to make the data easier to read and comprehend.

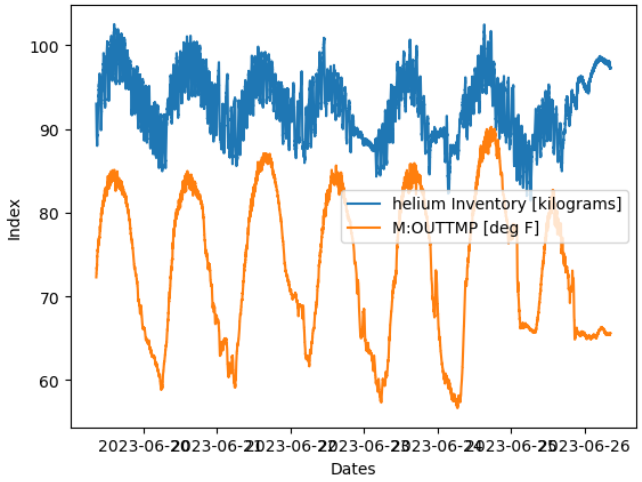

Figure 7: This graph represents the Helium inventory at the MESON testing area from June 19, 2023 to June 26, 2023. The fluctuation in between points expresses that as outdoor temperature increases, this causes the gas inventory density to decrease rising the storage pressure. Heat can cause helium to expand, and cold air can cause helium to shrink, this is why the helium inventory and the outdoor temperature lines on the graph are almost identical.

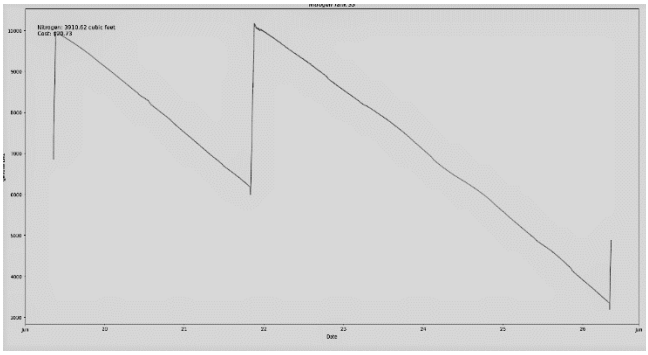

Figure 8: This graph represents the Nitrogen inventory at the MESON testing area from June 19, 2023, to June 26, 2023. These lines represent the liquid level decrease in the Nitrogen tank over a seven-day period. The “spikes” that are seen in this graph represent the refilling of the Nitrogen tank.

NML’s Helium & Nitrogen Python Script

```
PV_table=['N:1PT540','N:3LL90','N:C1LLHD','N:C2LLHD','N:SLLDEW','N:NLLDEW','C:2PT43','M:OUTTMP'] #stamps must be there a
interval='30000' # samlpling rate in milliseconds 1000ms = 1 second
#define start and end date

#start='27.03.2023 08:30:00,00' #DAY MONTH YEAR!!!!
#end='03.04.2023 08:30:00,00'   #DAY MONTH YEAR!!!!

datalogger=cryo_daq(PV_table,start,end,interval)
```

```
[ ]: print(datalogger)
```

```
[ ]:
## ENGINEERING CALCULATIONS #####
Tank5_vol= 30000*3.785/1000 #m3 TC verified
CC1_vol= 0.023 #m3 TC verified
CC2_vol= 0.023 #m3 TC verified
CM2_vol= 0.017*8 #m3 TC verified
end_cap= 0.024 #m3 TC verified -- not implemented
N_dew_vol=0.130 #m3 TC verified
S_dew_vol=0.46 #m3 TC verified
LHe_density=150 #kg/m3
R=8.31
M=4
#rho=PM/RT


#calculate GHe storage tank density for each timestamp based on external temp reading
for date in datalogger.index:
    datalogger.at[date,'GHe-storage-denisty']=(datalogger.at[date,'N:1PT540']*6.894)*M/((datalogger.at[date,'M:OUTTMP']-


## column to column math operations
datalogger['inventory']=Tank5_vol*datalogger['GHe-storage-denisty']   \
    +(datalogger['N:3LL90']*CM2_vol/100+datalogger['N:C1LLHD']*CC1_vol/10+datalogger['N:C2LLHD']*CC2_vol/10   \
     +datalogger['N:SLLDEW']*S_dew_vol/40+datalogger['N:NLLDEW']*N_dew_vol/100)*LHe_density
```

```
[ ]: print(datalogger)
```

```
[ ]: ### PRINTING PLOTS####

#define variables to be plotted
PV1_plot='inventory'
```

Figure 9: This python script calculates the inventory of helium based on various measurements at each timestamp. It performs column-to-column math operations using the specified volumes and densities to determine the helium inventory. The ‘PV_table’ contains the names of process variables that will be used to collect data. The ‘Interval’ specifies the sampling rate in milliseconds. This code creates a ‘cryo_daq’ object called ‘datalogger’ by passing the specified ‘PV_table’, ‘start’, ‘end’, and ‘interval’ parameters to the ‘cryo_daq’ constructor. The ‘cryo_daq’ object handles the data acquisition from the specified sensors during the given time range. The for loop in this script calculates the density of the Gaseous Helium (GHe) storage tank for each timestamp based on the pressure reading from 'N:1PT540' and the external temperature reading from 'M:OUTTMP'. The formula uses the Ideal Gas Law to compute the density.

```python
import pandas as pd
from matplotlib import pyplot as plt, dates as mdates, collections as c
import numpy as np


#define process variable and date time
PV_table=['N:1DPDW'] #stamps must be there all the time
interval='30000' # samlpling rate in milliseconds 1000ms = 1 second
#define start and end date
#start='27.03.2023 08:30:00,00' #DAY MONTH YEAR!!!!
#end='03.04.2023 08:30:00,00'   #DAY MONTH YEAR!!!!

datalogger_N2=cryo_daq(PV_table,start,end,interval)

plt.rcParams["figure.figsize"] = [20, 10]
plt.rcParams["figure.autolayout"] = True

d =  datalogger_N2.index.values
y = datalogger_N2['N:1DPDW']
s = pd.Series(y, index=d)

fig, ax = plt.subplots()

xval = mdates.date2num(s.index.to_pydatetime())
p = np.array([xval, s.values]).T.reshape(-1, 1, 2)
s = np.concatenate([p[:-1], p[1:]], axis=1)
lc = c.LineCollection(s, cmap="gray")
#lc.set_array(xval)

ax.add_collection(lc)
ax.xaxis.set_major_locator(mdates.MonthLocator())
ax.xaxis.set_minor_locator(mdates.DayLocator())

m = mdates.DateFormatter("%b")
ax.xaxis.set_major_formatter(m)
dat = mdates.DateFormatter("%d")
ax.xaxis.set_minor_formatter(dat)
ax.autoscale_view()

ax.set_xlabel('Date')
ax.set_ylabel('gallons LN2')
ax.set_title('Tank 45 level', fontstyle='italic')

# Show the plot
plt.savefig('figures/plot n2.png')
plt.show()
```

Figure 10: This python script displays a graph that represents the Nitrogen inventory in the NML cryo plant. The code generates a time-series plot showing the level of liquid nitrogen in Tank 45 over the specified time range. The x-axis represents the dates, and the y-axis represents the level of LN2 in gallons. The plot provides insights into the LN2 level variations over time. The code creates a Line Collection ‘lc’ to plot the data as a line graph. It converts the timestamp index ‘s.index’ to numerical values using ‘mdates.date2num’. The ‘LineCollection’ is created by combining adjacent data points for smoother visualization. The ‘lc’ Line Collection is added to the ’ax’ (axes) of the plot. The x-axis major locator is set to display month intervals, and the minor locator is set to display day intervals. The plot is saved as a PNG image file in the "figures" directory with the filename 'plot n2.png'. The ‘plt.show()’ function displays the plot on the screen, and ‘plt.close()’ closes the figure.

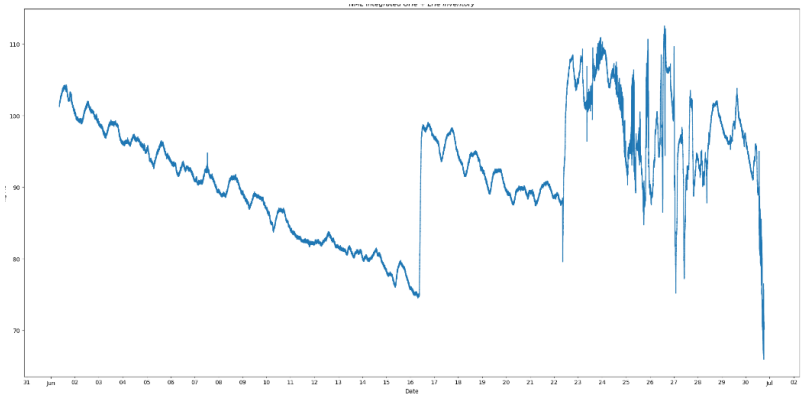

Figure 11: This graph represents the density of the Helium tank in a 30-day time period. The tank is constantly losing inventory and it has significant drop off on June 30, 2023 and July 30, 2023.

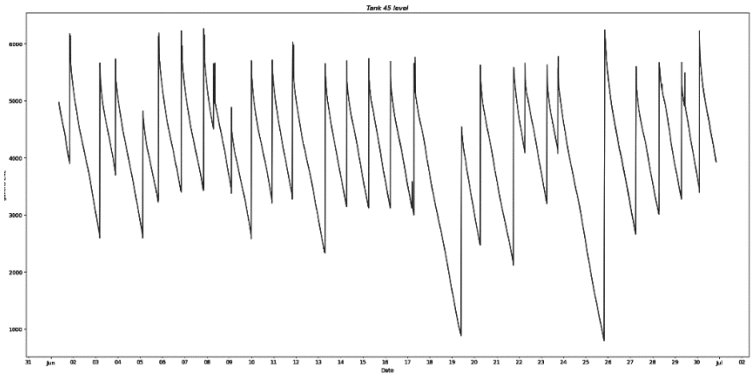

Figure 12: This graph represents the volume of the Nitrogen tank. It maintains a fill level of between 3,000 – 6,000 gallons during a 30-day time period. The “spikes” in this image also represents the refilling of the tank.

```python
import pandas as pd
from matplotlib import pyplot as plt, dates as mdates, collections as c
import numpy as np

#define process variable and date time
PV_table=['C:6LLDWN'] #stamps must be there all the time
interval='30000' # sampling rate in milliseconds (1000ms = 1 second)
#define start and end date
#start='27.03.2023 08:30:00,00' #DAY MONTH YEAR!!!!
#end='03.04.2023 08:30:00,00'   #DAY MONTH YEAR!!!!

datalogger_N2 = cryo_daq(PV_table, start, end, interval)

# Conversion from gallons to cubic feet
def convert_gallons_to_cubic_feet(gallons):
    return gallons * 93.11

# Calculate nitrogen cost
def calculate_nitrogen_cost(gallons):
    cubic_feet = convert_gallons_to_cubic_feet(gallons)
    cost = cubic_feet * 0.53/100  # $0.53/HCF
    return cost

# Retrieve nitrogen quantity in gallons from previous calculation
gallons = 42.0  # Replace with your actual value

# Convert gallons to cubic feet
cubic_feet = convert_gallons_to_cubic_feet(gallons)
cubic_feet_rounded = round(cubic_feet, 2)

# Calculate nitrogen cost
cost = calculate_nitrogen_cost(gallons)
cost_formatted = "${:,.2f}".format(cost)

plt.rcParams["figure.figsize"] = [20, 10]
plt.rcParams["figure.autolayout"] = True

d = datalogger_N2.index.values
y = datalogger_N2['C:6LLDWN']
s = pd.Series(y, index=d)

fig, ax = plt.subplots()

xval = mdates.date2num(s.index.to_pydatetime())
p = np.array([xval, s.values]).T.reshape(-1, 1, 2)
s = np.concatenate([p[:-1], p[1:]], axis=1)
lc = c.LineCollection(s, cmap="gray")
#lc.set_array(xval)
```

CMTF's Helium & Nitrogen Python Script

Figure 14: The image on the bottom left is the Nitrogen python script in the CMTF testing area. This code is generalized to successfully run and gather data from any Nitrogen tank sensor. I inputted 'PV_table' sensor: 'C:6LLDWN' to collect data from the tank in 30 second intervals.

```python
#Calculate the Liquid Volume in Inventory
#print(datalogger)
datalogger['InventoryResult'] = 0
def calculate_liquid_volume(LHe_density, conversion_factor, percent_filled, datalogger):
    # Calculate the liquid volume
    volume = LHe_density * conversion_factor * percent_filled / 100 * datalogger
    return volume

LHe_density = 150  # Density of the liquid in kg/m^3
conversion_factor = 0.001  # Conversion factor from gallons to cubic meters
percent_filled = 3000  # volume 1 Dewar in gallons


datalogger['VolumeN']= calculate_liquid_volume(LHe_density, conversion_factor, percent_

datalogger['C:3LL191']
datalogger['InventoryResult'] = 0
def calculate_liquid_volume(LHe_density, conversion_factor, percent_filled, datalogger):
    # Calculate the liquid volume
    volume = LHe_density * conversion_factor * percent_filled / 100 * datalogger
    return volume

LHe_density = 150  # Density of the liquid in kg/m^3
conversion_factor = 0.001  # Conversion factor from liters to cubic meters
percent_filled = 3000  # volume 1 Dewar in gallons

datalogger['VolumeN+1'] = calculate_liquid_volume(LHe_density, conversion_factor, perce

R=8.31
M=4

for date in datalogger.index:
```

Figure 13: This is a snippet of the Helium python script in the CMTF testing area. The code performs data processing, cleaning, and calculations to estimate the inventory of liquid helium based on various process variables and their measurements. The function 'calculate_liquid_volume' takes LHe_density, conversion_factor, percent_filled, and datalogger as inputs and calculates the liquid volume based on these values. The code also contained multiple loops to clean the data. Values were set to zero for specific variables when they were less than 0.5.

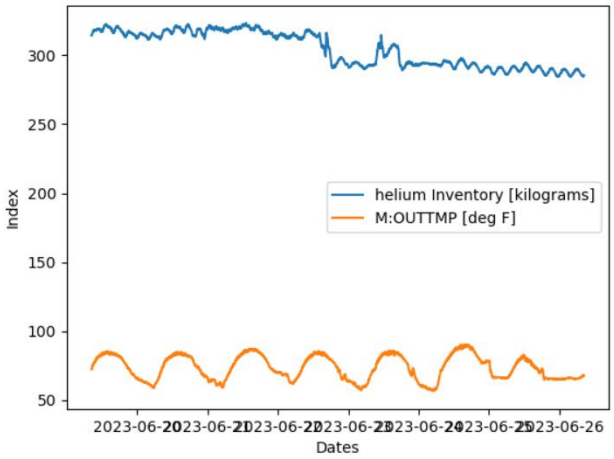

Figure 15: This graph represents the Helium inventory at the CMTF testing area from June 19, 2023, to June 26, 2023. Like stated before, there are small vertical alterations in the line graph because hot outdoor temperatures expands helium and colder outdoor temperatures shrinks helium.

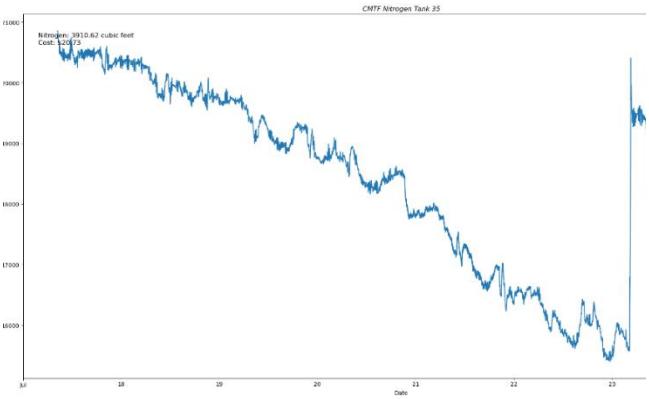

Figure 16: This graph shows the liquid Nitrogen usage in the CMTF cryo plant. The Nitrogen tank is having inventory used until July 23rd then it got refilled July 24th.

Final Helium & Nitrogen Loss Python Script

```
#Testing the NML_helium_inventory function

PV_table = ['N:1PT540', 'N:3LL90', 'N:C1LLHD', 'N:C2LLHD', 'N:SLLDEW', 'N:NLLDEW',

hours_he, days_he, kg_he_1 = NML_helium_inventory(start, end, PV_table, interval)

#Testing the NML_nitrogen_inventory function

PV_table = ['N:1DPDW']
n2_figure_name='figures/plot n2.png'

previous, hours_n2, days_n2 = Nitrogen_Inventory(PV_table, interval, start, end, r


#run report for NML Inventory Here
html_template_name = 'NML template.html'
html_output_name = 'NML report.html'

generalized_html_report(start, end, hours_he, days_he, kg_he_1, hours_n2, days_n2,

print('END NML')
#### END NML#####
```

Figure 17: This is a generalized code that performs analysis and generates a report related to Helium and Nitrogen inventories for a specific time period.

Order In Which The Script Displays Its Output:

CMTF Helium Inventory →CMTF Nitrogen Inventory

CMTF report → NML Helium Inventory → NML Nitrogen Inventory →NML report → MESON Helium Inventory → MESON Nitrogen Inventory → MESON report

Conclusion

This script gathers data, plots and estimates losses for the Helium and Nitrogen inventories of CMTF, NML, and MESON. This script was made to gather data from various cryo plants faster and more efficient. Instead of having different persons manually retrieving D44 ACNET data for each cryo plant, this code was generalized and designed to input any sensory data from any testing site and generate its report.

ACKNOWLEDGEMENTS

I would like to thank SIST committee for giving me the opportunity to complete my second internship here at Fermilab. I deeply appreciate all the help given from those I contacted for assistance, I'm glad that you helped and encouraged my growth as a scientist and coding developer. I also want to thank my supervisor Joaquim Creus-Prats for his assistance and advice with this project. This has been my first time onsite since COVID prevented us from working at the lab last summer, and I had amazing experience. I hope I will get the chance to return as a SIST Intern next summer as well.

REFERENCES

[1] *System Inventory Data Meson and NML Example*. fermicloud-my.sharepoint.com/:x:/r/personal/anthonyb_services_fnal_gov/_layouts/15/Doc.aspx?sourcedoc=%7BD377881C-8035-4DEF-AA38-E334F9D3B810%7D&file=Sprint%201%20-%20System%20INVENTORY%20data%20Meson%20and%20NML%20example.xlsx&action=default&mobileredirect=true.

[2] *System Inventory Data CMTF Example*. fermicloud-my.sharepoint.com/:x:/r/personal/jprats_services_fnal_gov/_layouts/15/Doc.aspx?sourcedoc=%7BB7CE63EE-

DF5B-4F88-BE24-98AC6708795B%7D&file=export_data-frame_cmtf.xlsx&action=default&mobileredirect=true.

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.