2022
9(2). 91-100

Journal of
MATERIALS NANOSCIENCE

Development of an automatic gas sampling cum injection unit and a graphical user interface of a feature extraction tool box based on PYTHON for sensor array data analysis

Debaayus Swain[#,1], Kunal Gupta[#,1], A. Sree Rama Murthy[#,2,3,*], V. Jayaraman[2,3]

[1]*Birla Institute of Technology and Science, Pilani, India - 333 031*

[2]*Indira Gandhi Centre for Atomic Research, Kalpakkam, India - 603 102*

[3]*Homi Bhabha National Institute, Kalpakkam - 603 102*

Supplementary Information

S1. Creation of Data Matrices

a) Data matrix (Type 1)

The data matrix of type 1 that consists of all the features of a particular sensor calculated by iterating the loop for all the pois and independently calling all the feature functions and storing them in the data frame subsequently providing option to user to store in separate *csv* files as per the code below:

```
def matrix_type1(df,list_poi,gap,points,sensor_name):
    sensor_list=df.columns.tolist()
    sensor=sensor_list.index(sensor_name)
    num_poi=len(list_poi)
    list_poi.sort(reverse=False)
    num_rows = df.shape[0]
    points = points // gap
    poi = list_poi[0]//gap
    if num_poi==1:
        next = num_rows-((list_poi[0])//gap)
    else:
        next = (list_poi[1]-list_poi[0])//gap
    sens = find_sensitivity(sensor,poi,df,next)
    recslope = recovery_slope(sensor,poi,df,next,gap)
    resslope = response_slope(sensor,poi,df,next,gap)
```

```
    restime = response_time(sensor,poi,df,next,gap)
    rectime = recovery_time(sensor,poi,df,next,gap)
    area = integral_area(sensor,poi,df,points,gap)
    Ratio = 1-sens
    sens *=100
    features = [[sens,recslope,resslope,rectime,restime,area,Ratio]]
    df1 = pd.DataFrame(features,columns=['Response(in %)','Recovery Slope', 'Response Slope', 'Reco
very Time', 'Response Time', 'Integral Area','Ratio'])
    for i in range(1,num_poi):
        poi = list_poi[i]//gap
        if i+1==num_poi:
            next = num_rows-((list_poi[i])//gap)
        else:
            next = (list_poi[i+1]-list_poi[i])//gap
        sens = find_sensitivity(sensor,poi,df,next)
        recslope = recovery_slope(sensor,poi,df,next,gap)
        resslope = response_slope(sensor,poi,df,next,gap)
        restime = response_time(sensor,poi,df,next,gap)
        rectime = recovery_time(sensor,poi,df,next,gap)
        area = integral_area(sensor,poi,df,points,gap)
        Ratio = 1-sens
        sens *= 100
        features = [[sens,recslope,resslope,restime,rectime,area,Ratio]]
        df1 = df1.append(pd.DataFrame(features,columns=['Response(in %)','Recovery Slope', 'Respons
e Slope', 'Recovery Time', 'Response Time', 'Integral Area','Ratio']),ignore_index=True)
    index_col = []
    for i in range (1,num_poi+1):
        index_col.append('Signal '+str(i))
    df1.insert(loc=0,column='Signal',value=index_col)
    df1.set_index("Signal",inplace=True)
    return df1
```

b) Data matrix (Type 2)

The data matrix type 2 is the table of a chosen feature for all the sensors. The elements of the data frame are computed by iterating all the sensors followed by the signals associated with that feature. The code for this type of matrix is given below.

```
def matrix_type2(feature,df,list_poi,gap,total_sensor, dat_col, points):
    num_rows=df.shape[0]
    num_poi=len(list_poi)
    start=int(dat_col)-1
    column_name = df.columns[start: start+total_sensor].tolist()
    df1 = pd.DataFrame(columns=column_name)
    if feature=='Response(in %)':
        for i in range(num_poi):
            poi = list_poi[i]//gap
            if i+1==num_poi:
                next = num_rows-((list_poi[i])//gap)
            else:
                next = (list_poi[i+1]-list_poi[i])//gap
            sens=[]
            for i in range (start,start+total_sensor):
                sens.append(find_sensitivity(i,poi,df,next)*100)
            df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
    elif feature=='Recovery Slope':
        for i in range(num_poi):
            poi = list_poi[i]//gap
            if i+1==num_poi:
                next = num_rows-((list_poi[i])//gap)
            else:
                next = (list_poi[i+1]-list_poi[i])//gap
            sens=[]
            for i in range (start,start+total_sensor):
                sens.append(response_slope(i,poi,df,next,gap))
            df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
```

```
elif feature=='Response Slope':
    for i in range(num_poi):
        poi = list_poi[i]//gap
        if i+1==num_poi:
            next = num_rows-((list_poi[i])//gap)
        else:
            next = (list_poi[i+1]-list_poi[i])//gap
        sens=[]
        for i in range (start,start+total_sensor):
            sens.append(recovery_slope(i,poi,df,next,gap))
        df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
elif feature=='Recovery Time':
    for i in range(num_poi):
        poi = list_poi[i]//gap
        if i+1==num_poi:
            next = num_rows-((list_poi[i])//gap)
        else:
            next = (list_poi[i+1]-list_poi[i])//gap
        sens=[]
        for i in range (start,start+total_sensor):
            sens.append(response_time(i,poi,df,next,gap))
        df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
elif feature=='Response Time':
    for i in range(num_poi):
        poi = list_poi[i]//gap
        if i+1==num_poi:
            next = num_rows-((list_poi[i])//gap)
        else:
            next = (list_poi[i+1]-list_poi[i])//gap
        sens=[]
        for i in range (start,start+total_sensor):
            sens.append(recovery_time(i,poi,df,next,gap))
        df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
elif feature=='Integral Area':
```

```
    points = points // gap
    for i in range(num_poi):
        poi = list_poi[i]//gap
        if i+1==num_poi:
            next = num_rows-((list_poi[i])//gap)
        else:
            next = (list_poi[i+1]-list_poi[i])//gap
        sens=[]
        for i in range (start,start+total_sensor):
            sens.append(integral_area(i,poi,df,points,gap))
        df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
elif feature=='Ratio':
    for i in range(num_poi):
        poi = list_poi[i]//gap
        if i+1==num_poi:
            next = num_rows-((list_poi[i])//gap)
        else:
            next = (list_poi[i+1]-list_poi[i])//gap
        sens=[]
        for i in range (start,start+total_sensor):
            sens.append(1-find_sensitivity(i,poi,df,next))
        df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
index_col = []
for i in range (1,num_poi+1):
    index_col.append('Signal '+str(i))
df1.insert(loc=0,column='Signal',value=index_col)
df1.set_index("Signal",inplace=True)
return df1
```

S2. Python codes for response and recovery time calculations

```
def response_time(sensor,poi,df,next,gap):
    ser1=df.iloc[:,sensor]
    baseline =base_line(poi,ser1)
    ser2=df.iloc[poi-1:poi+next-1,sensor]
    index_tip = tip(ser2,poi,next)
    delR = (baseline-ser2[index_tip])*0.90
    R90 = ser2[poi] -delR
    index1=poi
    index1min = abs(ser2[poi]-R90)
    for i in range(poi,index_tip+1):
        index1min = min(index1min,abs(ser2[i]-R90))
        if index1min == abs(ser2[i]-R90):
            index1 = i
    index2 = index1-1
    time1 = index1 * gap
    time2 = index2 * gap
    time = ((((R90-ser2[index2])/(ser2[index1]-ser2[index2]))*(time1-time2))+time2)-(poi*gap)
    return time
```

```
def recovery_time(sensor,poi,df,next,gap):
    ser1=df.iloc[:,sensor]
    baseline =base_line(poi+next,ser1)
    ser2=df.iloc[poi-1:poi+next-1,sensor]
    index_tip = tip(ser2,poi,next)
    delR = (baseline-ser2[index_tip])*0.90
    R90 = ser2[index_tip]+delR
    index1 = index_tip
    index1min = abs(ser2[index_tip]-R90)
    for i in range(index_tip,poi+next):
        index1min = min(index1min,abs(ser2[i]-R90))
        if index1min == abs(ser2[i]-R90):
            index1 = i
    index2 = index1-1
```

```
time1 = index1 * gap
time2 = index2 * gap
time = ((((R90-ser2[index2])/(ser2[index1]-ser2[index2]))*(time1-time2))+time2)-(index_tip*gap)
return time
```

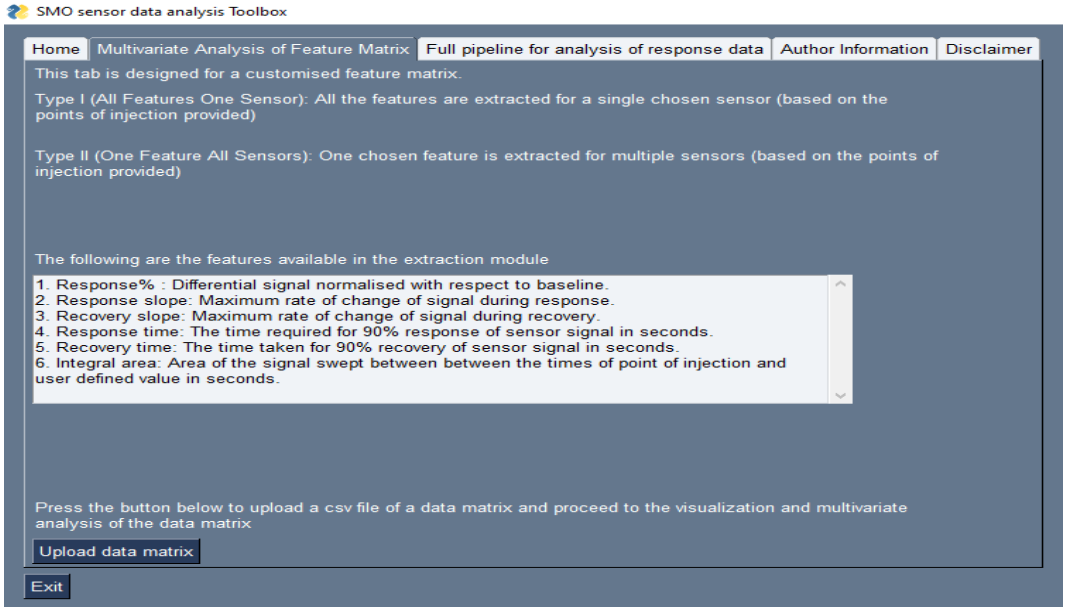


Fig. S1 The descriptors of Multivariate Analysis of Feature Matrix tab

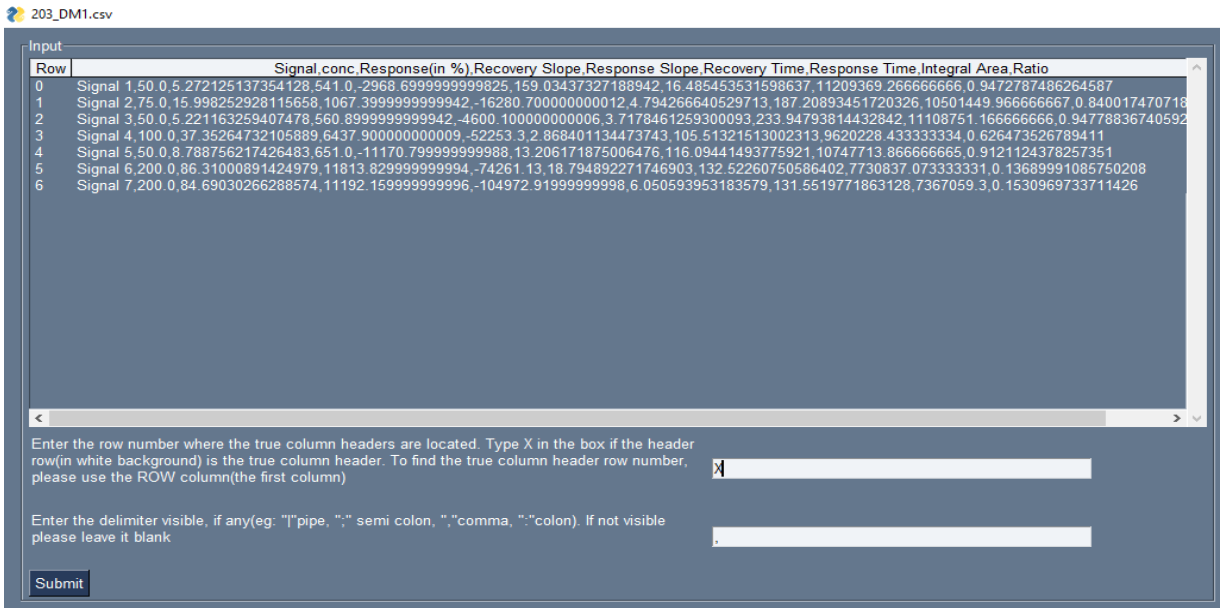


Fig. S2 Screenshot depicting the typical view of the uploaded data

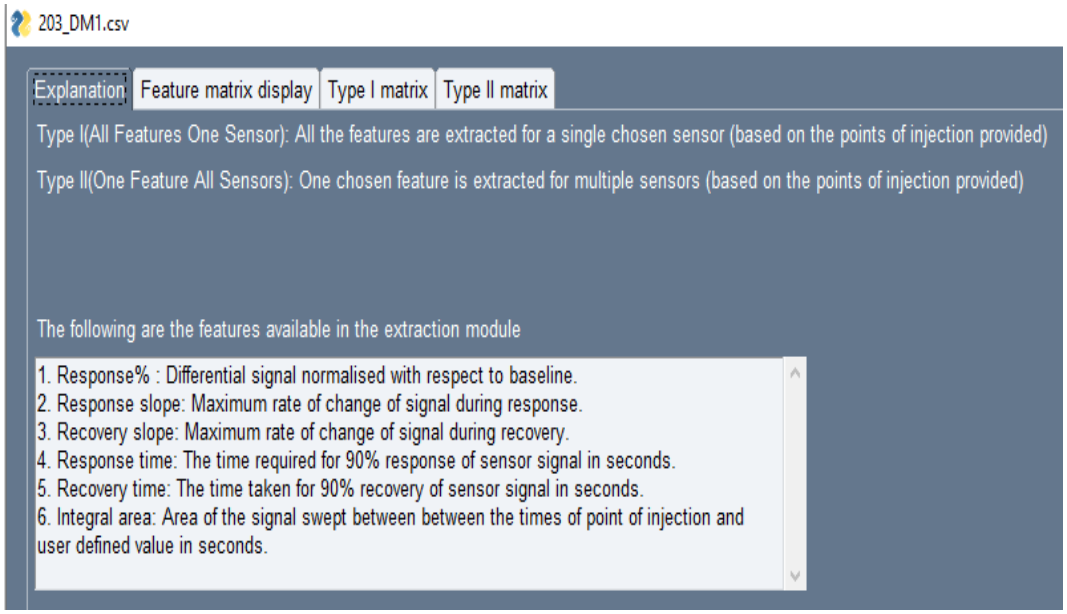


Fig. S3 Explanation tab with a short summary of the type of matrix and the features of the signals which can be visualized

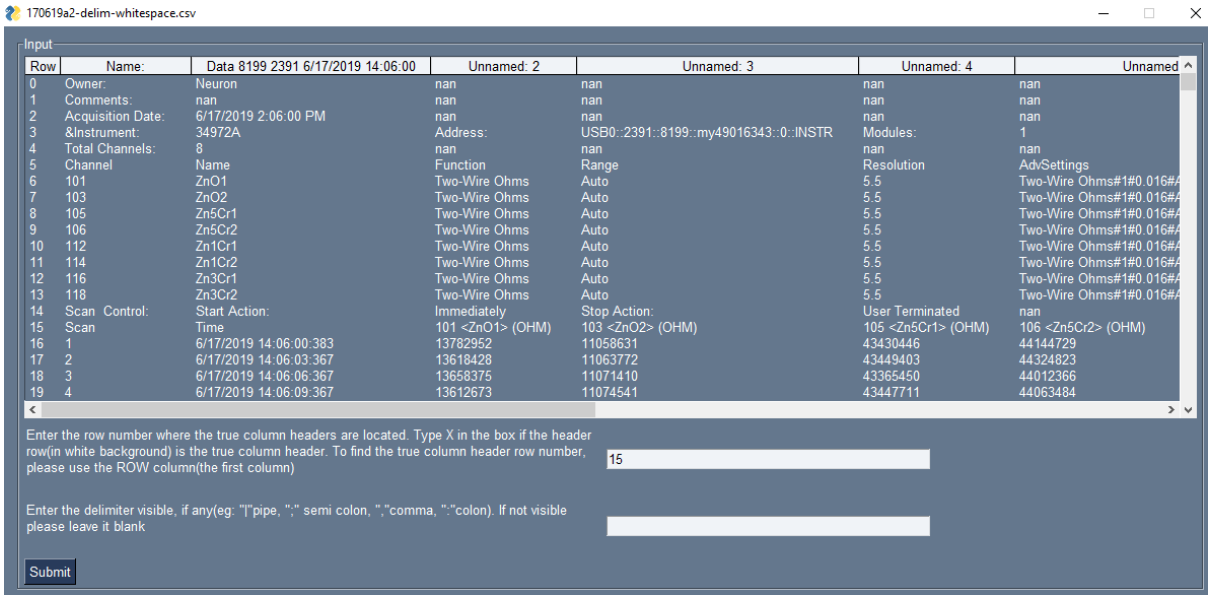


Fig. S4 In this typical example, user must enter 15 as the true headers to skip 15 rows and leave delimiter option blank for formatting the multiple sensors time series data

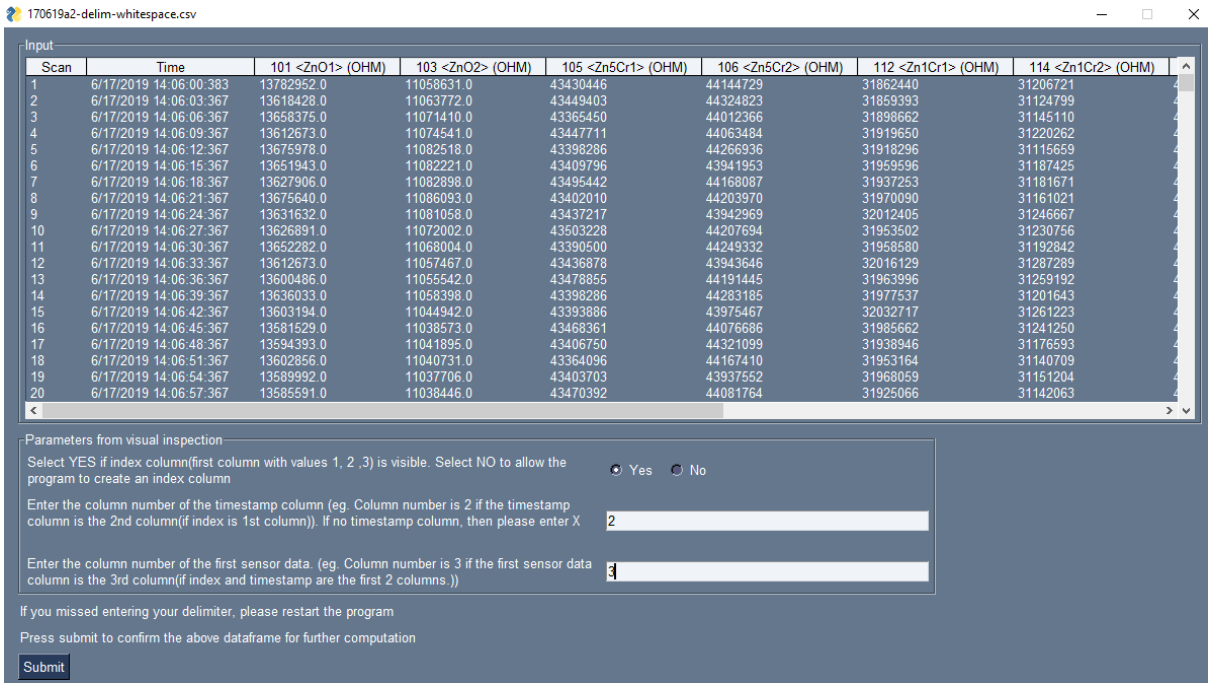
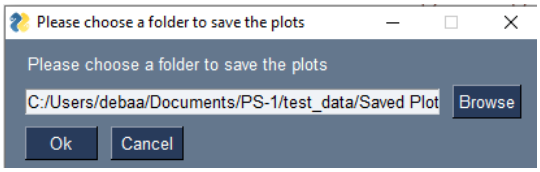
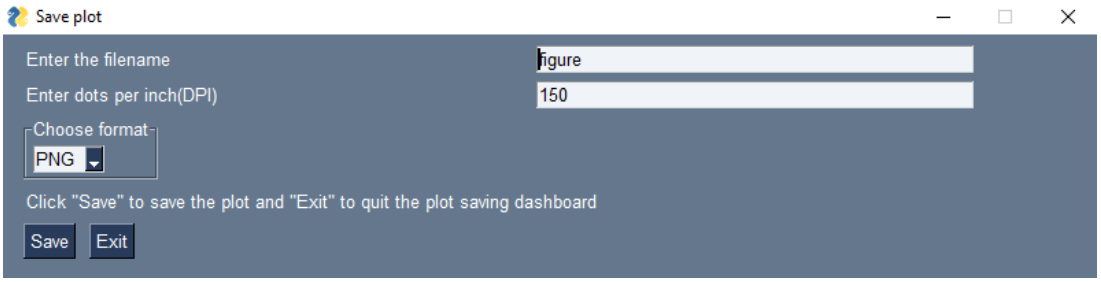


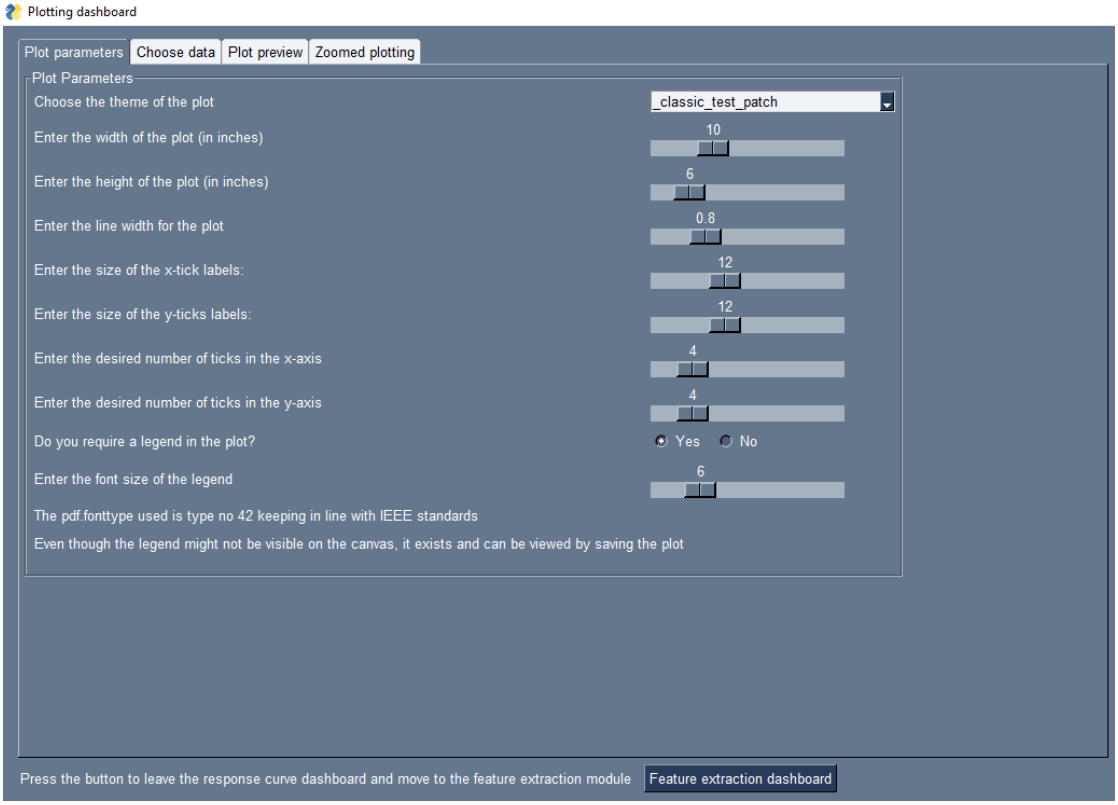
Fig. S5 Input screen for visual inspection with three entry options viz., index column, time stamp column and column number of 1st sensor's data



(a)



(b)



(c)

Fig. S6 Options for (a) saving the plot, (b) choosing the format and resolution (DPI) and (c) customized parameter selection for visualisation of the plots in plotting dashboard

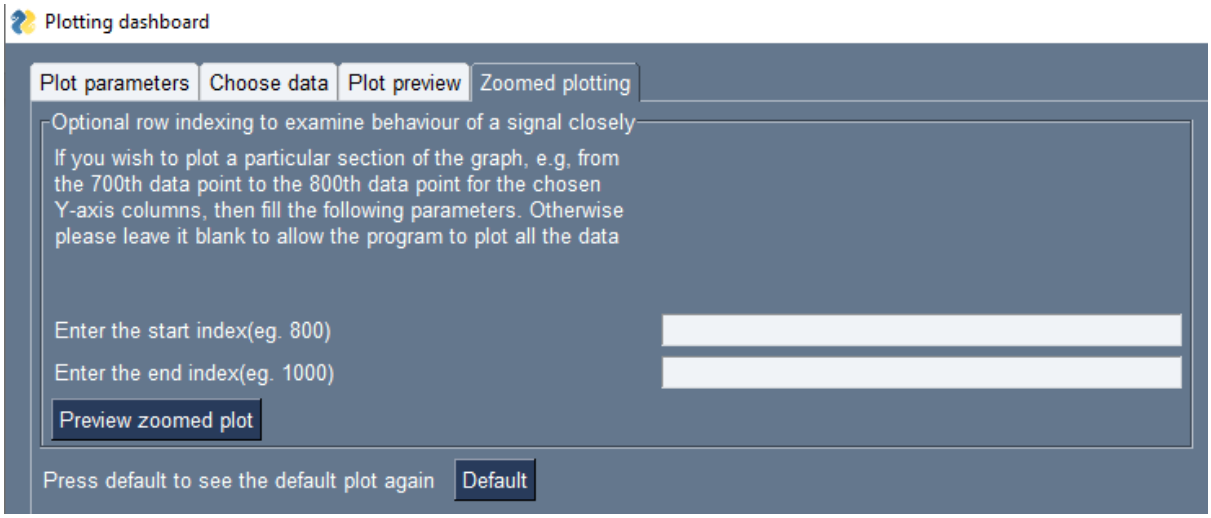


Fig. S7 Zoomed plotting option with user-selected section of the time series plot

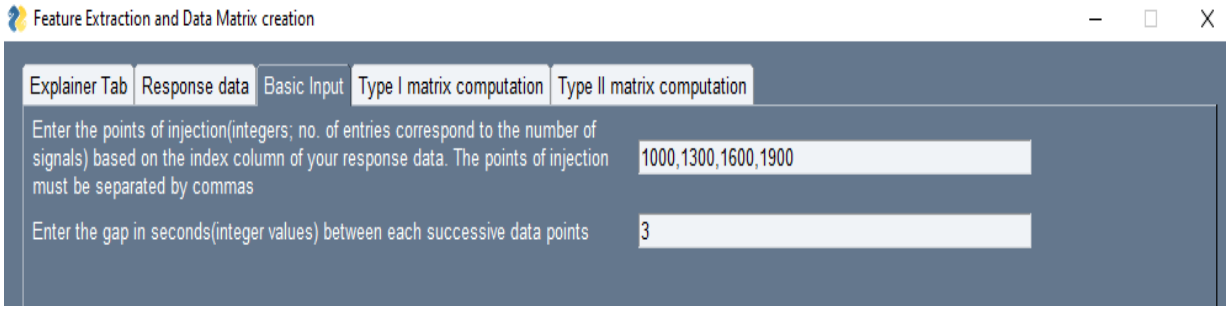


Figure S8 Basic input tab for pois and successive time gap between the time series data

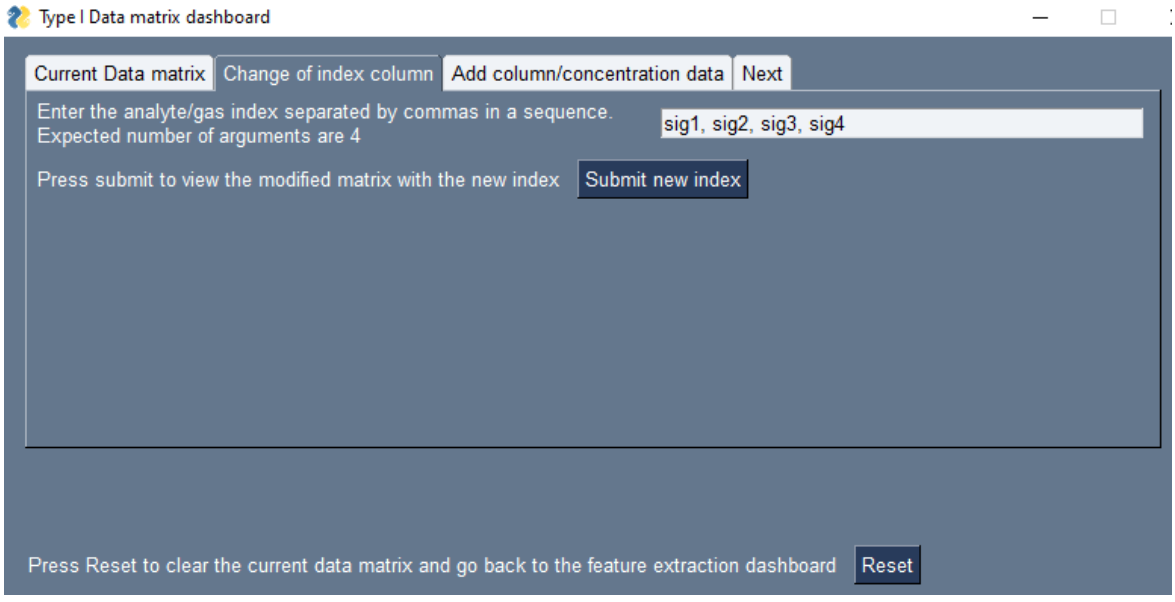
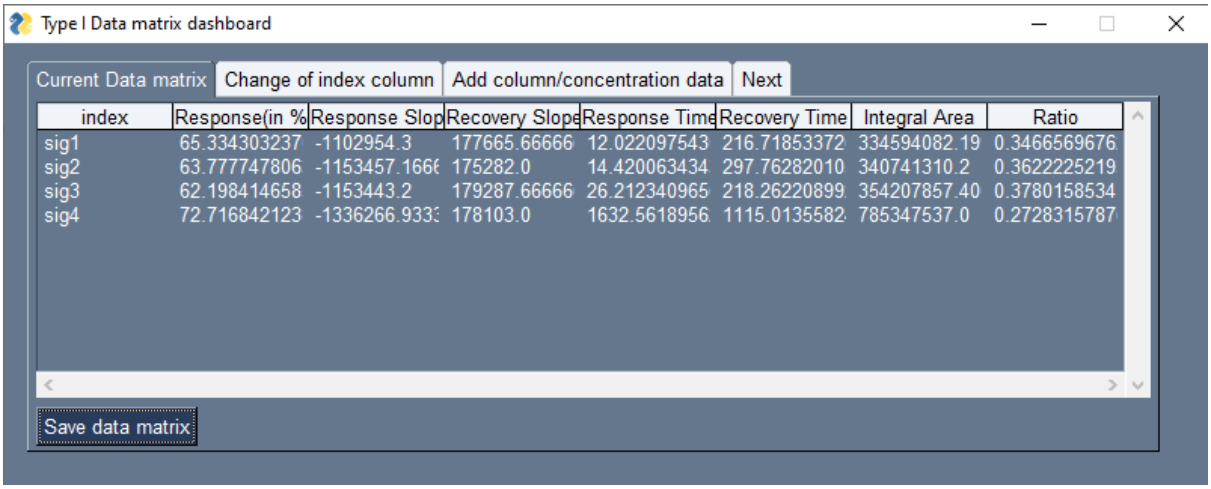
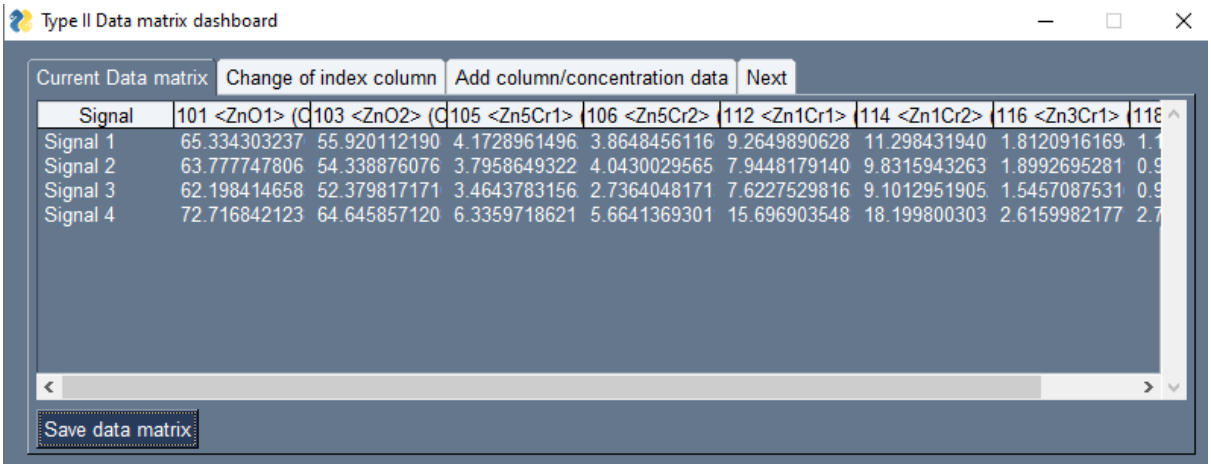


Fig. S9 Option to change the index column names as per the user's choice



(a)



(b)

Fig. S10 Tab showing the results with changed indices in (a) type I and (b) type II data matrices

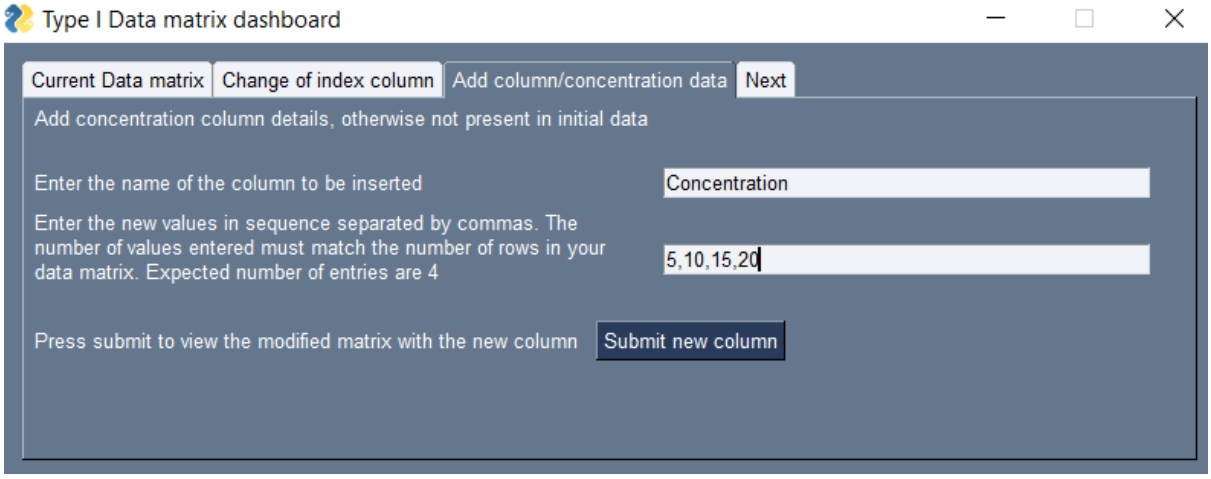


Fig. S11 Option to add columnar data of concentrations to the current data matrix

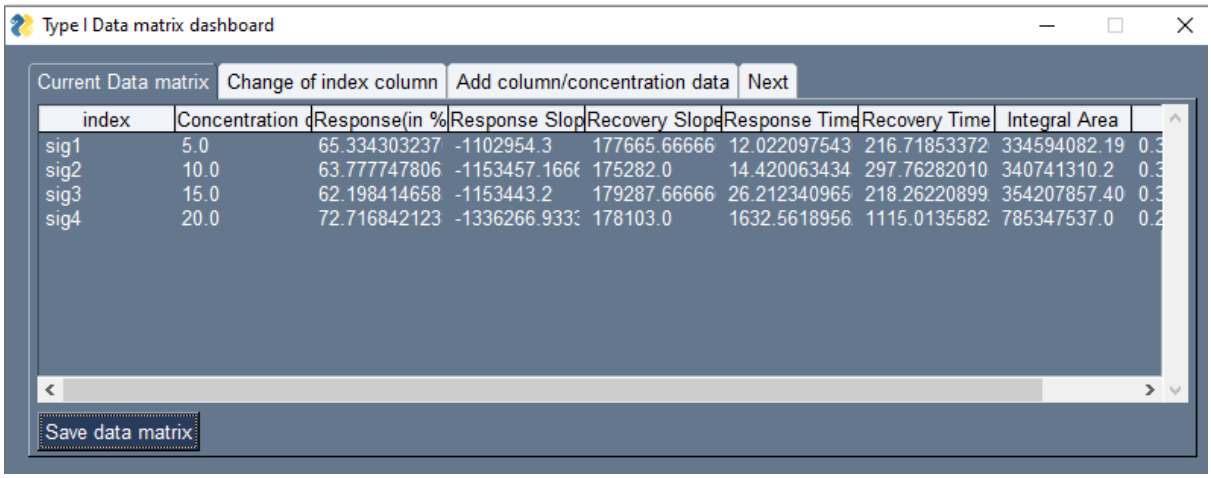


Fig. S12 Visualization of the modified type 1 data matrix

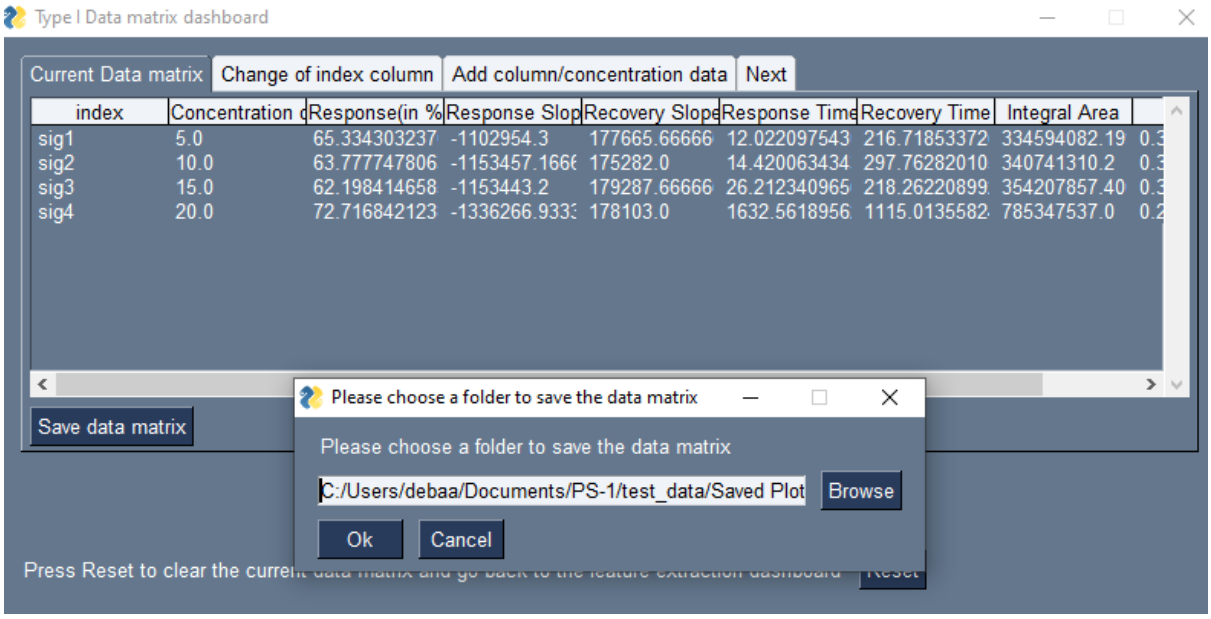


Fig. S13 Selection of a location to save the modified data matrix

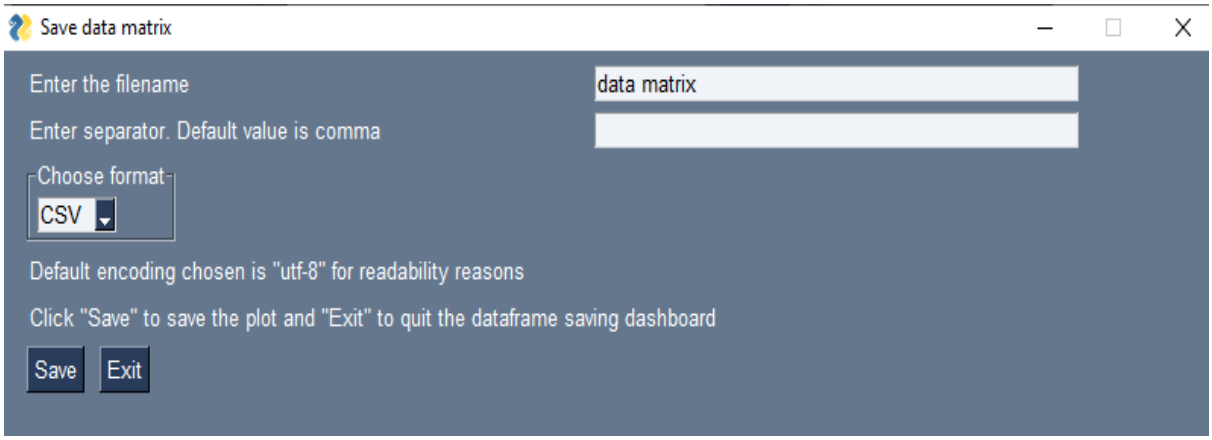


Figure S14 Screenshot of exercising the choice of format of the modified data matrix while saving

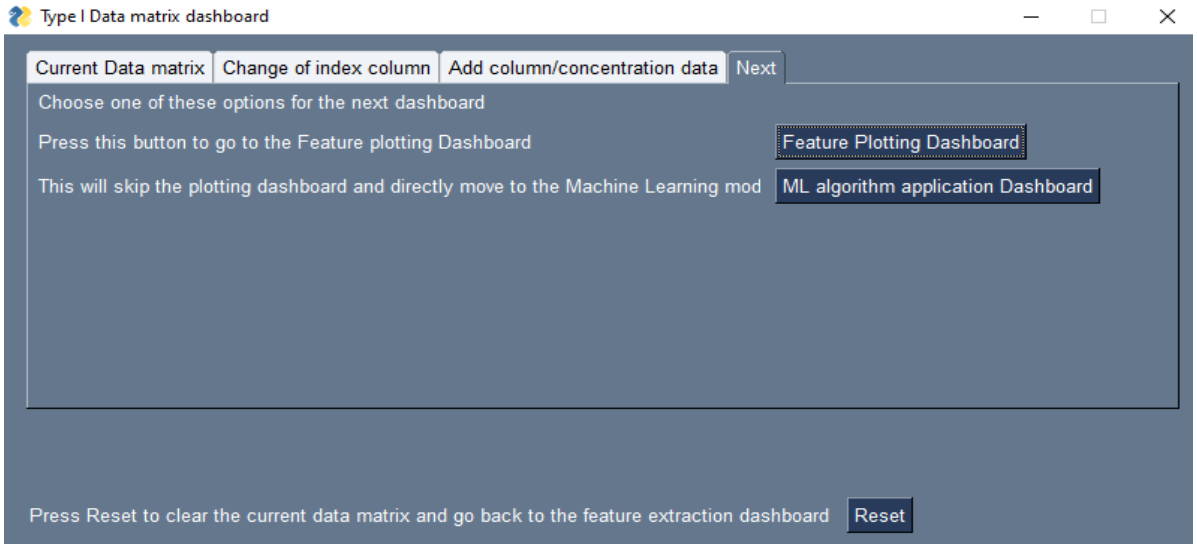


Figure S15 Screenshot of 'Next' tab to proceed either for feature visualization or for resetting the user-defined parameters

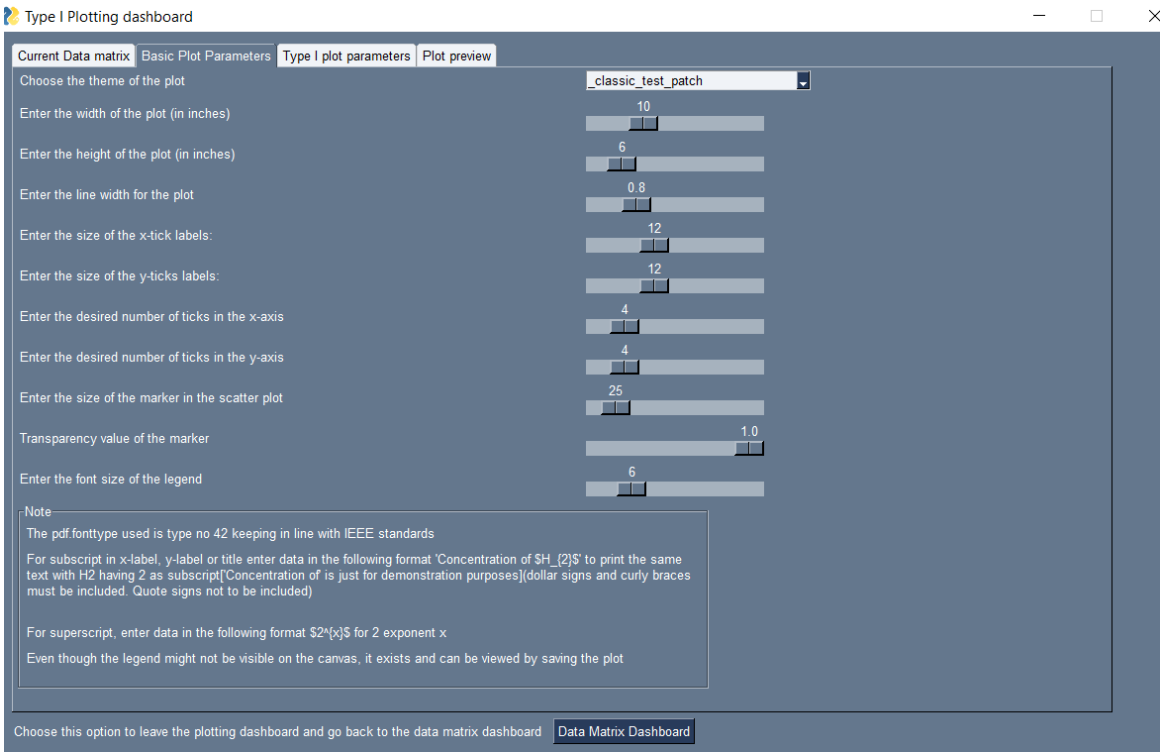


Fig. S2 Basic plot parameters tab to format the visualization details

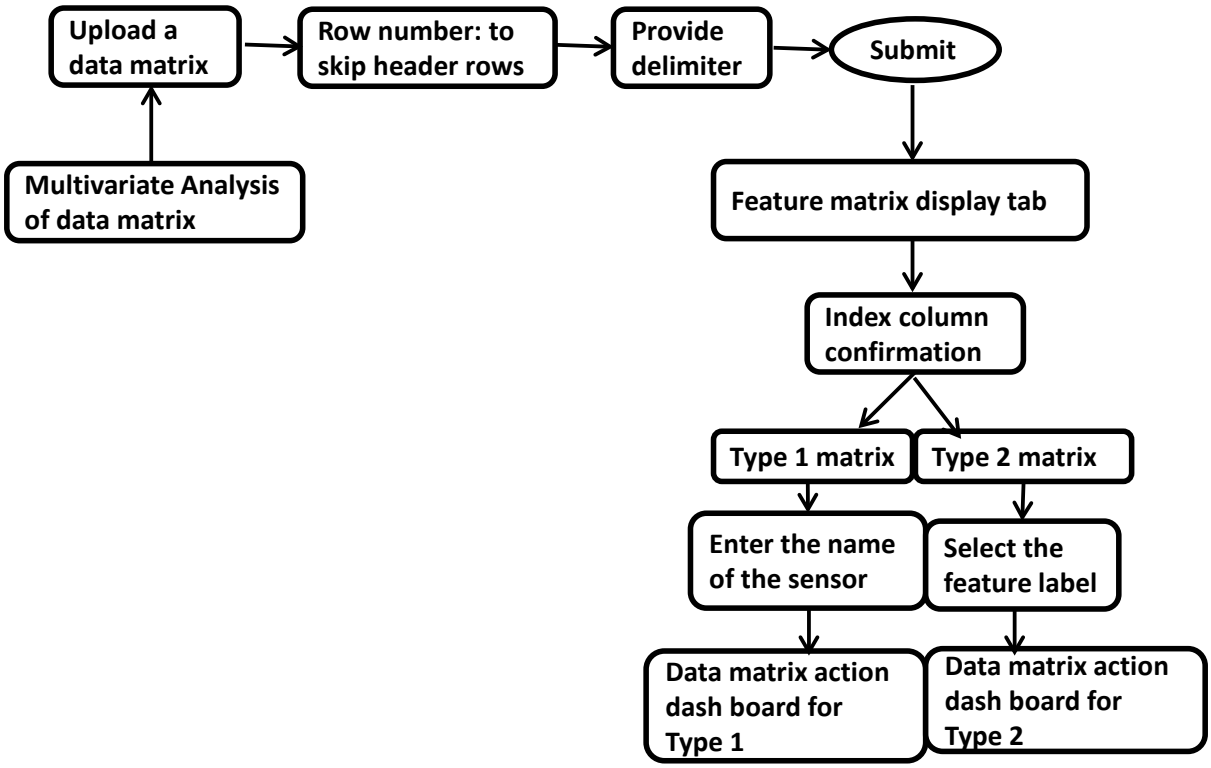
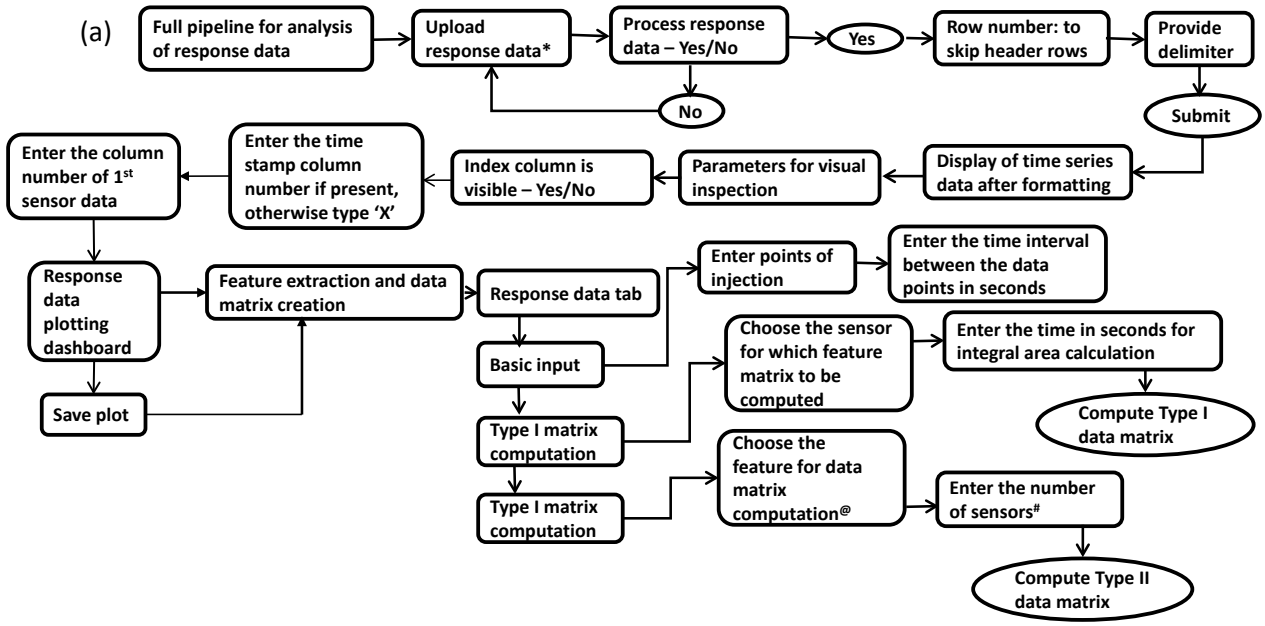
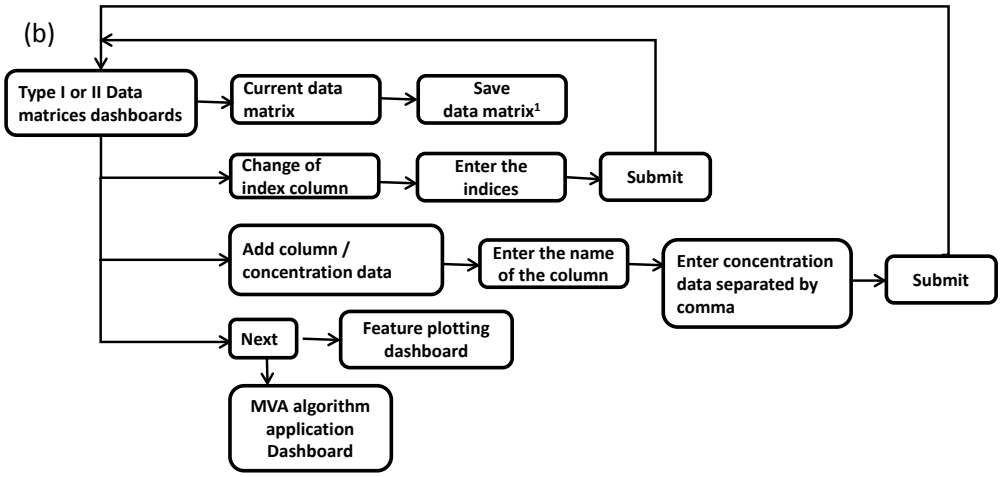


Fig. S17 A design flow sheet for multivariate analysis of data matrix



*** Response data should contain only one 'time' column with multiple sensor responses with common points of injection (POI)**

@for integral area calculation provide time in seconds; #Do not consider empty columns



[1]One can opt *.csv or *.xlsx or *.txt formats with chosen separator like , or |, etc.

Fig. S3A design flow sheet showing (a) Full pipeline for analysis of response data, (b) the process flow sheet of Type I and Type II data matrices.