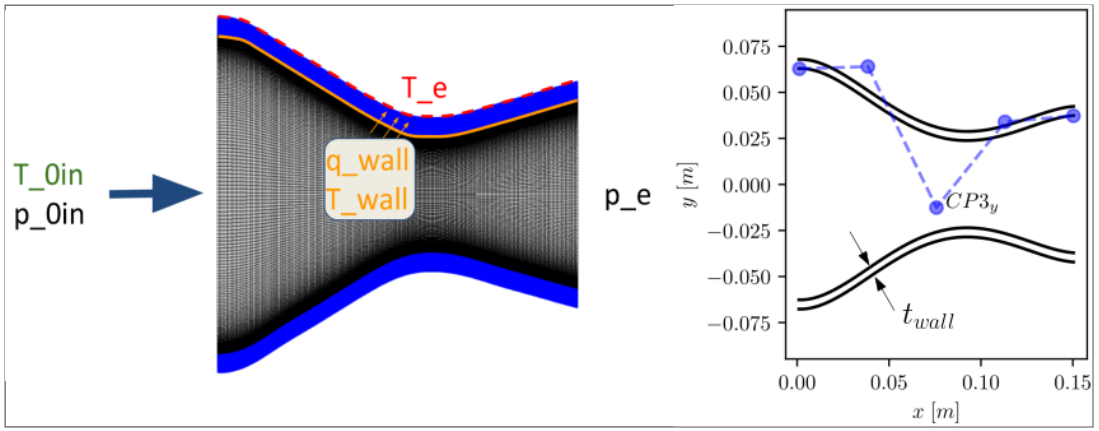
Proposal Study

This proposal employs a factorial design approach to construct a surrogate model. The aim is the reconstruction of 2D fields acquired through Reynolds Averaged Navier Stokes (RANS) solutions, involving conjugate heat transfer of the nozzle flow.

Nozzle Flow and Heat Transfer



Representation of Nozzle numerical domain and boundary conditions

Main Hypotesis

The central hypothesis poses that the factorial design could be effective in acquiring a model with the capacity to predict the principal components of a reduced-order model of fluid flow and heat transfer. This model, once established, can then be used to reconstruct complete two-dimensional flow fields from the designated independent variables.

Experimental Samples

To test this hypothesis, a series of numerical experiments will be conducted following the Central Composite Design (CCD)

- Inlet Total Temperature, T_{0in} [K]
- Inlet Total Pressure, p_{0in} [Pa]
- Wall Thickness, t_{wall} [m]
- Nozzle Shape Control Point, $CP3_y$ [m]
- External Nozzle Wall Temperature, T_e [K]

Central Composite Design

The range of variables were choosen in order to be within real experiments made by **(Back et al., 1964)** . Another factor taking into account to limit variable ranges is the numerical stability of conjugate heat transfer CFD.

In [1]:
```
from explann.doe import CentralCompositeDesign
       import pandas as pd

variables = {
    't_wall': [0.001, 0.010],               #[m]
    'CP3_y': [-0.01255805, 0.0],            #[m]
    'T_0in': [400.0, 650.0],                #[K]
    'p_0in': [4.0e5, 1.0e6],                #[Pa]
    'T_e': [290.0, 400.0],                  #[K]
}

# variable ranges
pd.DataFrame(variables)
```

Out[1]:

	t_wall	CP3_y	T_0in	p_0in	T_e
0	0.001	-0.012558	400.0	400000.0	290.0
1	0.010	0.000000	650.0	1000000.0	400.0

A CCD (Central Composite Desig) was choosen in order to be able to capture any high order interaction fo the design variables.

In [2]:
```
ccd = CentralCompositeDesign(
           variables=variables,
    center=(1,0),
    alpha='r',
    face='ccc',
)
ccd.doe
```

Out[2]:

Index	t_wall	CP3_y	T_0in	p_0in	T_e
1	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
2	1.000000	-1.000000	-1.000000	-1.000000	-1.000000
3	-1.000000	1.000000	-1.000000	-1.000000	-1.000000
4	1.000000	1.000000	-1.000000	-1.000000	-1.000000
5	-1.000000	-1.000000	1.000000	-1.000000	-1.000000
6	1.000000	-1.000000	1.000000	-1.000000	-1.000000
7	-1.000000	1.000000	1.000000	-1.000000	-1.000000
8	1.000000	1.000000	1.000000	-1.000000	-1.000000
9	-1.000000	-1.000000	-1.000000	1.000000	-1.000000
10	1.000000	-1.000000	-1.000000	1.000000	-1.000000
11	-1.000000	1.000000	-1.000000	1.000000	-1.000000
12	1.000000	1.000000	-1.000000	1.000000	-1.000000
13	-1.000000	-1.000000	1.000000	1.000000	-1.000000
14	1.000000	-1.000000	1.000000	1.000000	-1.000000
15	-1.000000	1.000000	1.000000	1.000000	-1.000000
16	1.000000	1.000000	1.000000	1.000000	-1.000000
17	-1.000000	-1.000000	-1.000000	-1.000000	1.000000
18	1.000000	-1.000000	-1.000000	-1.000000	1.000000
19	-1.000000	1.000000	-1.000000	-1.000000	1.000000
20	1.000000	1.000000	-1.000000	-1.000000	1.000000
21	-1.000000	-1.000000	1.000000	-1.000000	1.000000
22	1.000000	-1.000000	1.000000	-1.000000	1.000000
23	-1.000000	1.000000	1.000000	-1.000000	1.000000
24	1.000000	1.000000	1.000000	-1.000000	1.000000
25	-1.000000	-1.000000	-1.000000	1.000000	1.000000
26	1.000000	-1.000000	-1.000000	1.000000	1.000000

Index	t_wall	CP3_y	T_0in	p_0in	T_e
27	-1.000000	1.000000	-1.000000	1.000000	1.000000
28	1.000000	1.000000	-1.000000	1.000000	1.000000
29	-1.000000	-1.000000	1.000000	1.000000	1.000000
30	1.000000	-1.000000	1.000000	1.000000	1.000000
31	-1.000000	1.000000	1.000000	1.000000	1.000000
32	1.000000	1.000000	1.000000	1.000000	1.000000
33	0.000000	0.000000	0.000000	0.000000	0.000000
34	-2.378414	0.000000	0.000000	0.000000	0.000000
35	2.378414	0.000000	0.000000	0.000000	0.000000
36	0.000000	-2.378414	0.000000	0.000000	0.000000
37	0.000000	2.378414	0.000000	0.000000	0.000000
38	0.000000	0.000000	-2.378414	0.000000	0.000000
39	0.000000	0.000000	2.378414	0.000000	0.000000
40	0.000000	0.000000	0.000000	-2.378414	0.000000
41	0.000000	0.000000	0.000000	2.378414	0.000000
42	0.000000	0.000000	0.000000	0.000000	-2.378414
43	0.000000	0.000000	0.000000	0.000000	2.378414

In [3]:
```
ccd.levels
```

Out[3]:

Levels	t_wall	CP3_y	T_0in	p_0in	T_e
-2.378414	0.001000	-0.012558	400.000000	400000.000000	290.000000
-1.000000	0.003608	-0.008919	472.443974	573865.537712	321.875349
0.000000	0.005500	-0.006279	525.000000	700000.000000	345.000000
1.000000	0.007392	-0.003639	577.556026	826134.462288	368.124651

	t_wall	CP3_y	T_0in	p_0in	T_e
Levels					
2.378414	0.010000	0.000000	650.000000	1000000.000000	400.000000

This DoE variables was then used as boundary conditions for the CFD(Computational Fluid Dynamics) numerical experiments. The 43 runs accounts for accounts for 429.6MB of data stored in a `HDF5` file, from wich, the seleceted fields were choosen to build snapshot matrices

`'Heat_Flux_UPPER_WALL', 'Mach', 'Pressure', 'Temperature', 'Temperature_Solid', 'Temperature_Solid_INNERWALL'`

For each experiment a snapshot vector $\mathbf{s_i}$, corresponding to the concatation of fluid and solid domain 2D numerical solutions were obtained.

$$\mathbf{S_i}(T_{0in}, p_{0in}, t_{wall}, CP3_y) = \begin{bmatrix} \mathbf{q_{upperWall}} \\ \mathbf{p_{fluid}} \\ \mathbf{T_{fluid}} \\ \mathbf{M_{fluid}} \\ \mathbf{T_{solid}} \\ \mathbf{T_{solidInner}} \end{bmatrix}^{252840\times 1}$$

, the snapshots $\mathbf{S_i}$ were then concatenated to form the snapshot matrix $\mathbf{A}$ of the Design of Experimen (DoE).

$$\mathbf{A} = \begin{bmatrix} \mathbf{S_1^T} \\ \mathbf{S_2^T} \\ \vdots \\ \mathbf{S_N^T} \end{bmatrix}^{43 \times 252840}$$

In [4]:
```
import sys
sys.path.append('/home/ppiper/Dropbox/local/github/frog')

from frog.datahandler import HDF5Handler

hfdh = HDF5Handler(
    datapath='/home/ppiper/Dropbox/local/github/explann/data/experimental_planning_T0in_limit_clear/hf_transpose.h5',
    datasets = ['Heat_Flux_UPPER_WALL', 'Mach', 'Pressure', 'Temperature', 'Temperature_Solid', 'Temperature_Solid_INNERWALL'],
)
```

Dependent variables (Order Reduction)

The dependent variables are principal componentes of a POD(Proper Orthogonal Decompostion) of the snapshots matrix $\mathbf{A}$

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}}$$

by truncanting the diagonal matrix of singular values $\mathbf{\Sigma}$, the snapshot matrix can be represented in terms o a reduced basis $\tilde{\mathbf{U}}$ and principal components $\tilde{\mathbf{\Lambda}}$

$$\tilde{\mathbf{A}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Lambda}}$$

The data were then compressed using 10 principal components of the orthogonal projection, resulting in 24.3MB of data. An approximated 20-fold reduction in storage requirments.

In [5]:
```
from sklearn.pipeline import Pipeline
       from sklearn.decomposition import PCA
from frog.normalization import Normalization, DataHandlerNormalization
import pickle

rom = Pipeline([
    ('scaler', Normalization(bounds=[-1,1])),
    ('pca', PCA(n_components=10)),
])

rom.fit(hfdh.data)

results_pc = rom.transform(hfdh.data)

with open('/home/ppiper/Dropbox/local/github/explann/data/experimental_planning_T0in_limit_clear/rom.pkl','wb') as f:
    pickle.dump(rom,f)

with open('/home/ppiper/Dropbox/local/github/explann/data/experimental_planning_T0in_limit_clear/rom.pkl','rb') as f:
    rom = pickle.load(f)
```

The principal components are then joined to the CCD DoE to form full tables for factorial model construction.

In [6]:
```
import pandas as pd
results = {f'L{i}': results_pc[:,i] for i in range(results_pc.shape[1])}

ccd.append_results(results)
ccd.doe
```

Out[6]:

Index	t_wall	CP3_y	T_0in	p_0in	T_e	L0	L1	L2	L3	L4
1	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	47.655208	-2.574525	1.054864	0.116466	0.006886
2	1.000000	-1.000000	-1.000000	-1.000000	-1.000000	47.657308	-2.584284	1.705080	-0.057877	0.004462
3	-1.000000	1.000000	-1.000000	-1.000000	-1.000000	47.482853	2.523185	1.070312	0.115771	0.007093
4	1.000000	1.000000	-1.000000	-1.000000	-1.000000	47.484867	2.512161	1.748841	-0.031342	0.003471
5	-1.000000	-1.000000	1.000000	-1.000000	-1.000000	47.629272	-2.501821	-1.897897	0.122485	-0.010489
6	1.000000	-1.000000	1.000000	-1.000000	-1.000000	47.632879	-2.517772	-0.854666	-0.157822	-0.009886
7	-1.000000	1.000000	1.000000	-1.000000	-1.000000	47.456260	2.599557	-1.962213	0.058714	-0.004065
8	1.000000	1.000000	1.000000	-1.000000	-1.000000	47.459997	2.581783	-0.875525	-0.179893	-0.005059
9	-1.000000	-1.000000	-1.000000	1.000000	-1.000000	-47.418552	-3.711689	1.184190	0.148468	0.003752
10	1.000000	-1.000000	-1.000000	1.000000	-1.000000	-47.415844	-3.725914	2.135199	-0.085025	-0.001124
11	-1.000000	1.000000	-1.000000	1.000000	-1.000000	-47.666183	3.624649	1.288358	0.170554	-0.002338
12	1.000000	1.000000	-1.000000	1.000000	-1.000000	-47.663805	3.609438	2.268540	-0.022280	-0.010885
13	-1.000000	-1.000000	1.000000	1.000000	-1.000000	-47.451135	-3.618356	-2.496137	0.145204	0.006485
14	1.000000	-1.000000	1.000000	1.000000	-1.000000	-47.446400	-3.641930	-0.984372	-0.229059	0.005923
15	-1.000000	1.000000	1.000000	1.000000	-1.000000	-47.698985	3.721687	-2.469012	0.095408	0.006392
16	1.000000	1.000000	1.000000	1.000000	-1.000000	-47.694650	3.696741	-0.914726	-0.216737	0.001333
17	-1.000000	-1.000000	-1.000000	-1.000000	1.000000	47.655208	-2.574525	1.054864	0.116466	0.006886
18	1.000000	-1.000000	-1.000000	-1.000000	1.000000	47.657308	-2.584284	1.705080	-0.057877	0.004462
19	-1.000000	1.000000	-1.000000	-1.000000	1.000000	47.482853	2.523185	1.070312	0.115771	0.007093

Index	t_wall	CP3_y	T_0in	p_0in	T_e	L0	L1	L2	L3	L4
20	1.000000	1.000000	-1.000000	-1.000000	1.000000	47.484867	2.512161	1.748841	-0.031342	0.003471
21	-1.000000	-1.000000	1.000000	-1.000000	1.000000	47.629272	-2.501821	-1.897897	0.122485	-0.010489
22	1.000000	-1.000000	1.000000	-1.000000	1.000000	47.632879	-2.517772	-0.854666	-0.157822	-0.009886
23	-1.000000	1.000000	1.000000	-1.000000	1.000000	47.456260	2.599557	-1.962213	0.058714	-0.004065
24	1.000000	1.000000	1.000000	-1.000000	1.000000	47.459997	2.581783	-0.875525	-0.179893	-0.005059
25	-1.000000	-1.000000	-1.000000	1.000000	1.000000	-47.418552	-3.711689	1.184190	0.148468	0.003752
26	1.000000	-1.000000	-1.000000	1.000000	1.000000	-47.415844	-3.725914	2.135199	-0.085025	-0.001124
27	-1.000000	1.000000	-1.000000	1.000000	1.000000	-47.666183	3.624649	1.288358	0.170554	-0.002338
28	1.000000	1.000000	-1.000000	1.000000	1.000000	-47.663805	3.609438	2.268540	-0.022280	-0.010885
29	-1.000000	-1.000000	1.000000	1.000000	1.000000	-47.451135	-3.618356	-2.496137	0.145204	0.006485
30	1.000000	-1.000000	1.000000	1.000000	1.000000	-47.446400	-3.641930	-0.984372	-0.229059	0.005923
31	-1.000000	1.000000	1.000000	1.000000	1.000000	-47.698985	3.721687	-2.469012	0.095408	0.006392
32	1.000000	1.000000	1.000000	1.000000	1.000000	-47.694650	3.696741	-0.914726	-0.216737	0.001333
33	0.000000	0.000000	0.000000	0.000000	0.000000	0.025936	-0.038532	-0.035311	-0.020620	-0.031631
34	-2.378414	0.000000	0.000000	0.000000	0.000000	0.022612	-0.015342	-1.611487	0.417478	-0.036272
35	2.378414	0.000000	0.000000	0.000000	0.000000	0.029698	-0.056069	1.020713	-0.226146	-0.035953
36	0.000000	-2.378414	0.000000	0.000000	0.000000	0.137596	-7.218266	-0.059227	-0.014131	0.189392
37	0.000000	2.378414	0.000000	0.000000	0.000000	-0.381774	7.591583	0.028908	-0.014403	0.190904
38	0.000000	0.000000	-2.378414	0.000000	0.000000	0.059251	-0.135482	3.737966	0.092381	-0.044229
39	0.000000	0.000000	2.378414	0.000000	0.000000	-0.009178	0.056877	-3.682362	-0.139727	-0.048415
40	0.000000	0.000000	0.000000	-2.378414	0.000000	113.128140	-0.016841	0.340592	-0.011511	-0.054779
41	0.000000	0.000000	0.000000	2.378414	0.000000	-113.070333	-0.076683	0.329156	-0.028152	-0.069653
42	0.000000	0.000000	0.000000	0.000000	-2.378414	0.025936	-0.038532	-0.035311	-0.020620	-0.031631
43	0.000000	0.000000	0.000000	0.000000	2.378414	0.025936	-0.038532	-0.035311	-0.020620	-0.031631

Fitting Factorial Model

The CCD factorial model accounts for second order terms in each $\mathbf{L_i}$ componenent of the $\mathbf{\Lambda}$ matrix.

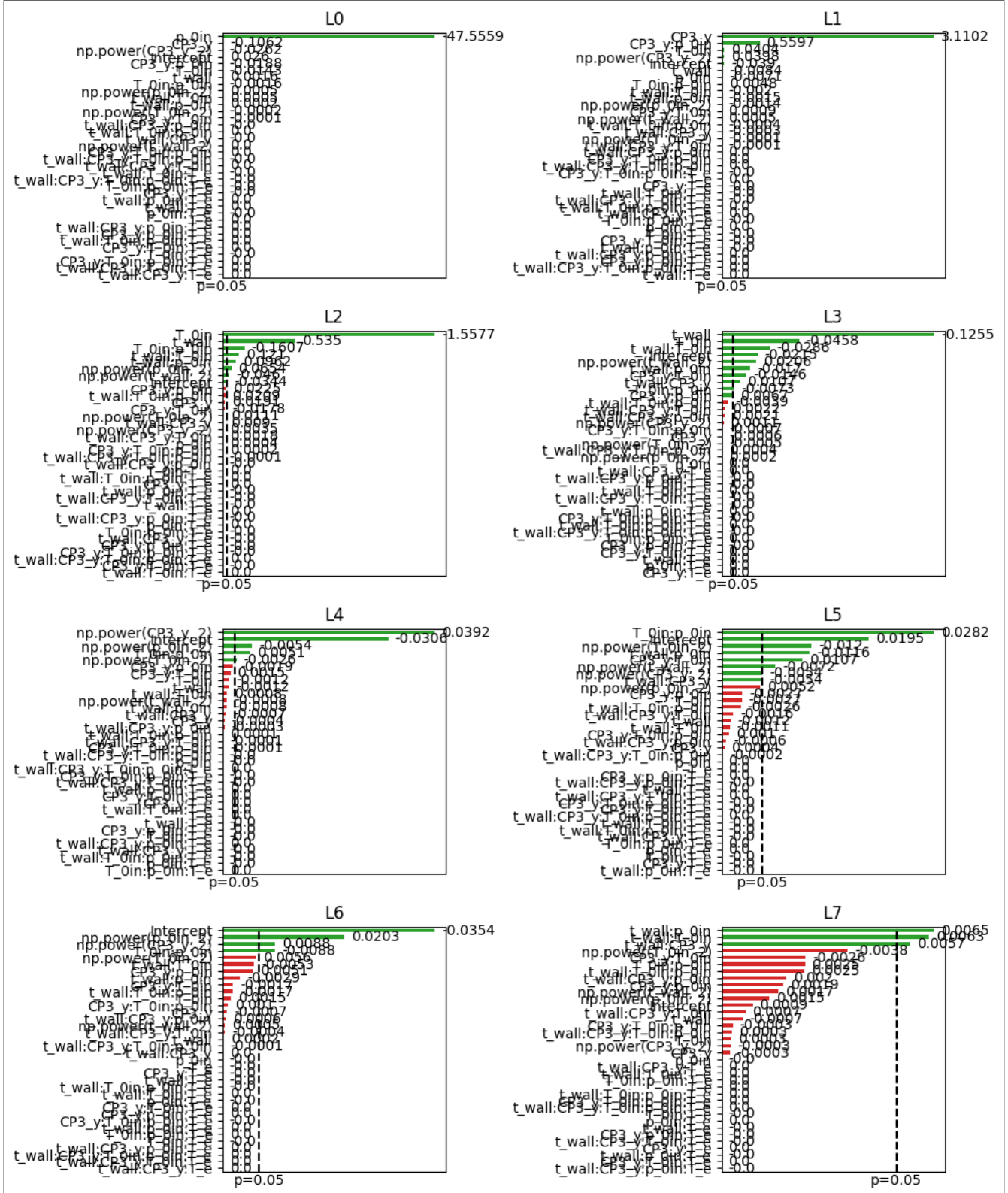
In [7]:
```
from explann.models import FactorialModel
       from explann.dataio import ImportString, ImportXLSX
from explann.plot import ParetoPlot
import matplotlib.pyplot as plt

expressions = {f'{key}': f'{key} ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_0in, 2) + np.power(p_0in,

fm_ccd = FactorialModel(
    data = ccd.doe,
    functions = expressions,
    levels = ccd.levels,
)
expressions
```

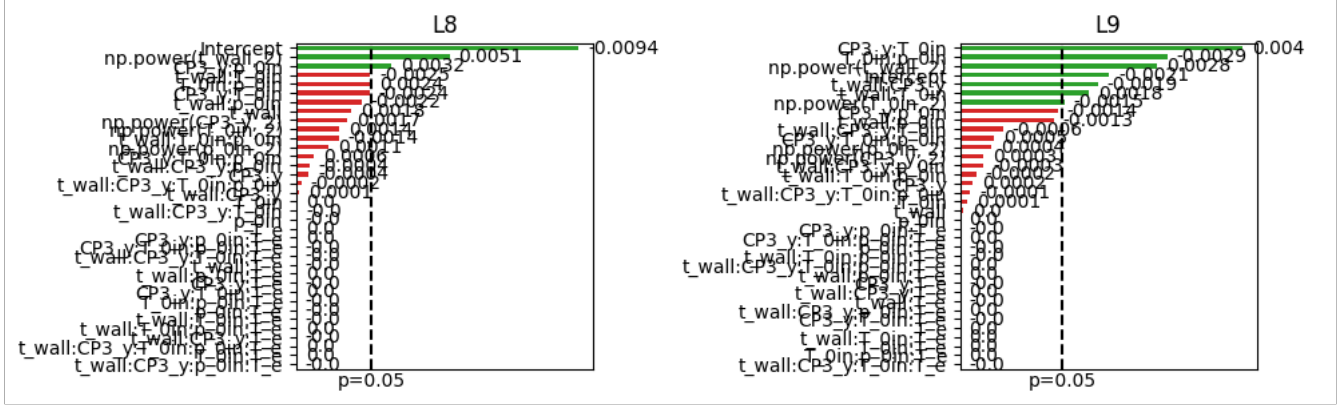
Out[7]:

```
{'L0': 'L0 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_0in, 2) + np.power(p_0in,
2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
 'L1': 'L1 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_0in, 2) + np.power(p_0in,
2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
 'L2': 'L2 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_0in, 2) + np.power(p_0in,
2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
 'L3': 'L3 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_0in, 2) + np.power(p_0in,
2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
 'L4': 'L4 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_0in, 2) + np.power(p_0in,
2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
 'L5': 'L5 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_0in, 2) + np.power(p_0in,
2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
 'L6': 'L6 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_0in, 2) + np.power(p_0in,
2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
 'L7': 'L7 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_0in, 2) + np.power(p_0in,
2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
 'L8': 'L8 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_0in, 2) + np.power(p_0in,
2) + np.power(CP3_y,2) + np.power(CP3_y, 2)',
 'L9': 'L9 ~ 1 + t_wall * CP3_y * T_0in * p_0in * T_e + np.power(t_wall, 2) + np.power(T_0in, 2) + np.power(p_0in,
2) + np.power(CP3_y,2) + np.power(CP3_y, 2)'}
```

As can be viewed in pareto plots, not all terms are significant, so we can retaing fewer terms.

In [8]:
```
from explann.plot import ParetoPlot
import matplotlib.pyplot as plt

fig, ax = plt.subplots(5,2, figsize=(10,15))
ax = ax.flatten()

pp = ParetoPlot(fm_ccd)

pp.plot(ax=ax)
plt.tight_layout()
```

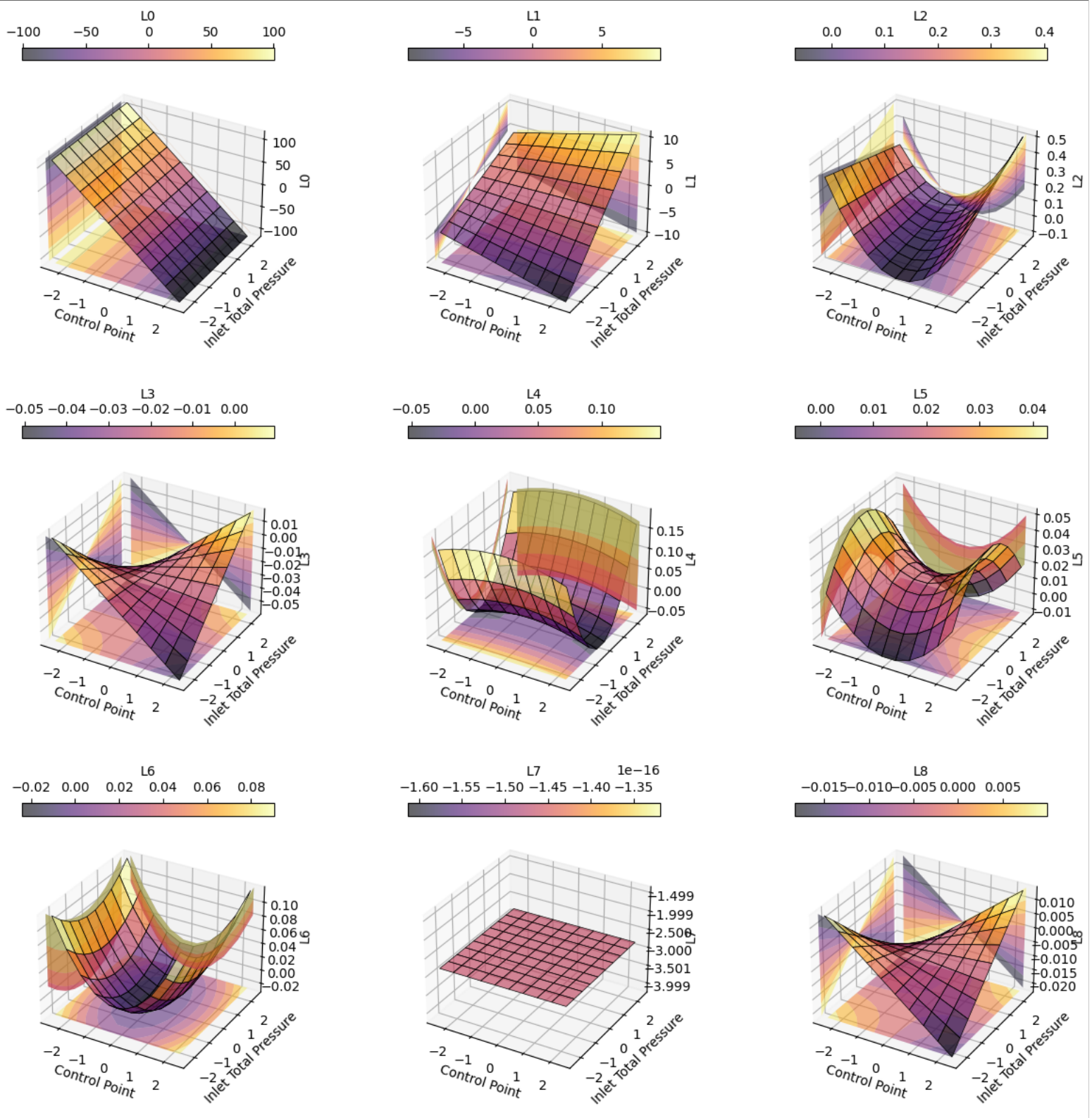

L0
p_0in 47.5559
CP3_y -0.1062
np.power(CP3_y, 2) 0.0262
Intercept 0.026
CP3_y:p_0in 0.0189
t_wall 0.0016
T_0in:p_0in 0.0006
np.power(p_0in, 2) 0.0006
t_wall:p_0in 0.0003
np.power(t_wall, 2) 0.0002
CP3_y:t_wall 0.0001
t_wall:T_0in 0.0
t_wall:CP3_y 0.0
np.power(CP3_y, 2):p_0in 0.0
t_wall:CP3_y:p_0in 0.0
t_wall:CP3_y:T_0in 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:CP3_y:T_0in:p_0in 0.0
CP3_y:T_0in 0.0
t_wall:p_0in 0.0
CP3_y:p_0in 0.0
p_0in:t_wall 0.0
t_wall:CP3_y:p_0in 0.0
CP3_y:T_0in:p_0in 0.0
t_wall:CP3_y:t_wall 0.0
p_0in:t_wall 0.0
CP3_y:t_wall 0.0
t_wall:CP3_y:t_wall 0.0
p=0.05
L1
CP3_y 3.1102
CP3_y:p_0in 0.5597
np.power(CP3_y, 2) 0.0404
Intercept 0.0398
t_wall 0.0054
T_0in:p_0in 0.0049
p_0in 0.0041
t_wall:p_0in 0.0015
np.power(p_0in, 2) 0.0009
CP3_y:t_wall 0.0005
np.power(t_wall, 2) 0.0004
t_wall:CP3_y 0.0003
np.power(CP3_y, 2):p_0in 0.0003
t_wall:CP3_y:p_0in 0.0001
CP3_y:T_0in 0.0
t_wall:CP3_y:T_0in 0.0
t_wall:CP3_y:T_0in:p_0in 0.0
p_0in:t_wall 0.0
CP3_y:p_0in 0.0
t_wall:T_0in 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:CP3_y:T_0in 0.0
t_wall:CP3_y:t_wall 0.0
CP3_y:t_wall 0.0
t_wall:p_0in 0.0
t_wall:CP3_y:p_0in 0.0
t_wall:CP3_y:T_0in:p_0in 0.0
t_wall:T_0in 0.0
p=0.05
L2
T_0in 1.5577
t_wall 0.535
T_0in:p_0in 0.1607
t_wall:p_0in 0.0392
np.power(p_0in, 2) 0.0362
np.power(t_wall, 2) 0.0344
Intercept 0.0225
t_wall:T_0in 0.0209
CP3_y 0.0178
CP3_y:T_0in 0.0111
np.power(T_0in, 2) 0.0095
t_wall:CP3_y 0.0095
np.power(CP3_y, 2) 0.0078
t_wall:CP3_y:T_0in 0.0018
CP3_y:p_0in 0.0001
t_wall:CP3_y:p_0in 0.0001
t_wall:CP3_y:t_wall 0.0
T_0in:p_0in 0.0
t_wall:T_0in:p_0in 0.0
t_wall:p_0in 0.0
t_wall:CP3_y:p_0in 0.0
T_0in:p_0in:t_wall 0.0
CP3_y:t_wall 0.0
t_wall:CP3_y:t_wall 0.0
CP3_y:T_0in:p_0in 0.0
t_wall:CP3_y:T_0in:p_0in 0.0
t_wall:T_0in:t_wall 0.0
p=0.05
L3
t_wall 0.1255
T_0in 0.0458
t_wall:T_0in 0.0286
Intercept 0.0253
np.power(t_wall, 2) 0.0206
CP3_y 0.0146
t_wall:CP3_y 0.0107
T_0in:p_0in 0.0073
CP3_y:T_0in 0.0067
t_wall:T_0in:p_0in 0.0039
np.power(T_0in, 2) 0.0021
t_wall:CP3_y:T_0in 0.0017
CP3_y:T_0in:p_0in 0.0006
np.power(T_0in, 2):p_0in 0.0005
t_wall:CP3_y:T_0in:p_0in 0.0004
np.power(p_0in, 2) 0.0002
t_wall:CP3_y:p_0in 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:p_0in 0.0
CP3_y:p_0in 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:CP3_y:T_0in 0.0
t_wall:CP3_y:t_wall 0.0
CP3_y:t_wall 0.0
t_wall:CP3_y:p_0in 0.0
t_wall:CP3_y:T_0in:p_0in 0.0
p=0.05
L4
np.power(CP3_y, 2) 0.0392
Intercept -0.0306
np.power(p_0in, 2) 0.0054
p_0in 0.0051
np.power(T_0in, 2) 0.0026
CP3_y:p_0in 0.0019
CP3_y:T_0in 0.0012
t_wall:p_0in 0.0012
t_wall 0.0012
np.power(t_wall, 2) 0.0008
t_wall:T_0in 0.0008
t_wall:CP3_y 0.0004
T_0in:p_0in 0.0003
t_wall:CP3_y:p_0in 0.0001
t_wall:CP3_y:T_0in 0.0001
t_wall:CP3_y:T_0in:p_0in 0.0
CP3_y:T_0in:p_0in 0.0
t_wall:CP3_y:t_wall 0.0
CP3_y:t_wall 0.0
t_wall:T_0in 0.0
t_wall:CP3_y:t_wall 0.0
CP3_y:p_0in 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:CP3_y:p_0in 0.0
t_wall:CP3_y:T_0in 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:CP3_y:T_0in:p_0in 0.0
T_0in:p_0in:t_wall 0.0
p=0.05
L5
T_0in:p_0in 0.0282
Intercept 0.0195
np.power(T_0in, 2) 0.012
t_wall:p_0in 0.0116
t_wall 0.0107
np.power(t_wall, 2) 0.0072
CP3_y:t_wall 0.0054
np.power(p_0in, 2) 0.0052
CP3_y:p_0in 0.0037
t_wall:T_0in 0.0026
t_wall:CP3_y 0.0015
t_wall:T_0in:p_0in 0.0011
CP3_y:T_0in 0.0006
t_wall:CP3_y:t_wall 0.0004
t_wall:CP3_y:T_0in 0.0002
t_wall:CP3_y:T_0in:p_0in 0.0
CP3_y:p_0in 0.0
t_wall:CP3_y:p_0in 0.0
t_wall:p_0in 0.0
CP3_y:t_wall 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:CP3_y:T_0in 0.0
t_wall:CP3_y:t_wall 0.0
CP3_y:T_0in:p_0in 0.0
t_wall:CP3_y:T_0in:p_0in 0.0
t_wall:T_0in 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:CP3_y:p_0in 0.0
t_wall:CP3_y:T_0in:p_0in 0.0
t_wall:T_0in:t_wall 0.0
p=0.05
L6
Intercept 0.0354
np.power(CP3_y, 2) 0.0203
np.power(T_0in, 2) 0.0088
T_0in:p_0in 0.0088
np.power(p_0in, 2) 0.0056
t_wall:p_0in 0.0053
t_wall 0.0051
CP3_y:p_0in 0.0029
CP3_y:t_wall 0.0017
t_wall:T_0in 0.0017
t_wall:T_0in:p_0in 0.0015
CP3_y:T_0in 0.0007
t_wall:CP3_y:p_0in 0.0006
np.power(t_wall, 2) 0.0005
t_wall:CP3_y:t_wall 0.0004
t_wall:CP3_y:T_0in 0.0001
t_wall:CP3_y:T_0in:p_0in 0.0
p_0in:t_wall 0.0
CP3_y:t_wall 0.0
t_wall:T_0in 0.0
t_wall:CP3_y:t_wall 0.0
CP3_y:p_0in 0.0
t_wall:CP3_y:t_wall 0.0
CP3_y:T_0in:p_0in 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:CP3_y:T_0in:p_0in 0.0
t_wall:CP3_y:p_0in 0.0
p=0.05
L7
t_wall:p_0in 0.0065
t_wall:T_0in 0.0063
CP3_y:t_wall 0.0057
np.power(T_0in, 2) 0.0038
CP3_y:T_0in 0.0026
t_wall:T_0in:p_0in 0.0023
t_wall:CP3_y 0.002
CP3_y:T_0in:p_0in 0.0019
np.power(t_wall, 2) 0.0017
Intercept 0.0015
t_wall:CP3_y:t_wall 0.0009
t_wall:CP3_y:T_0in 0.0007
t_wall:CP3_y:T_0in:p_0in 0.0003
np.power(CP3_y, 2) 0.0003
CP3_y:p_0in 0.0003
t_wall:CP3_y:p_0in 0.0
t_wall:T_0in 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:CP3_y:T_0in 0.0
t_wall:CP3_y:t_wall 0.0
t_wall:CP3_y:T_0in:p_0in 0.0
t_wall:T_0in:p_0in 0.0
t_wall:T_0in:t_wall 0.0
t_wall:CP3_y:p_0in 0.0
t_wall:CP3_y:T_0in:p_0in 0.0
p=0.05

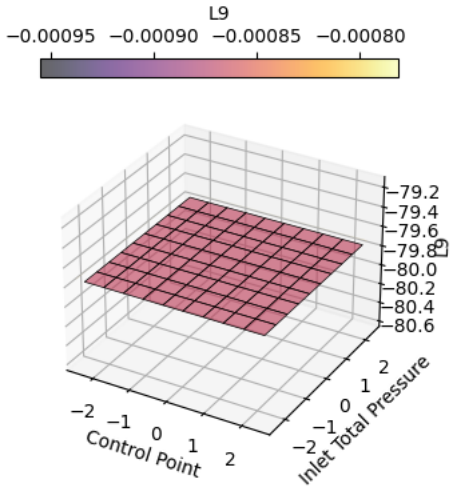


In [9]:
```
fm_ccd.get_significant_model_functions()
fm_ccd_full = fm_ccd
fm_ccd = fm_ccd.build_significant_models()
```

Resulting Model for principal components

In [10]:
```
from explann.plot import plot_surface

fig, ax = plt.subplots(4,3, figsize=(15,20),subplot_kw={"projection": "3d"})
ax = ax.flatten()

for i,axi in enumerate(ax):#fm_ccd.functions.items()):
    if i<len(fm_ccd.functions.items()):
        key = list(fm_ccd.functions.keys())[i]
        plot_surface(x='p_0in', y='CP3_y', z=key,
             model=fm_ccd,
             n_pts=10,
             ax=ax[i],
             other_params=dict(t_wall=0,T_0in=0,T_e=0),
             labels=dict(xlabel='Control Point', ylabel='Inlet Total Pressure', zlabel=key), cmap='inferno', scaled=False)
    else:
        ax[i].set_axis_off()
```


L0
L1
L2
L3
L4
L5
L6
L7
1e−16
L8
Control Point
Inlet Total Pressure


L9
−0.00095
−0.00090
−0.00085
−0.00080
−79.2
−79.4
−79.6
−79.8
−80.0
−80.2
−80.4
−80.6
L9
−2
−1
0
1
2
Control Point
−2
−1
0
1
2
Inlet Total Pressure

Test Model Accuracy

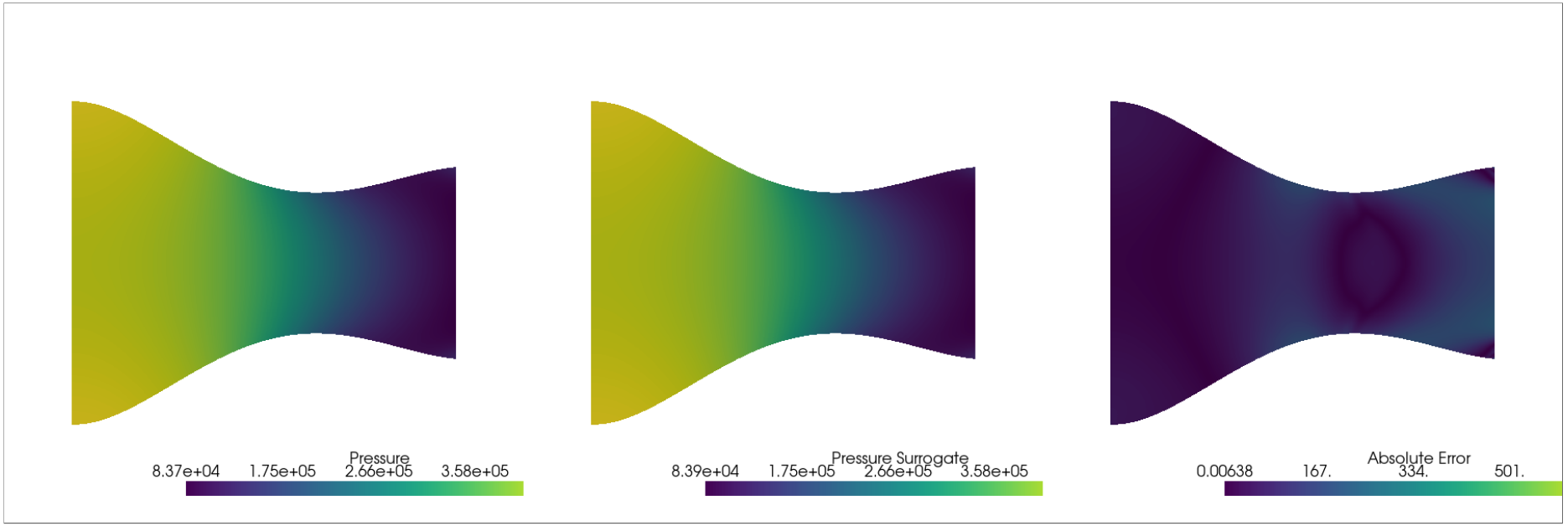
The model accuracy was tested by comparing flow reconstruction with full numerical solution for an unseeing set of independent variables

In [11]:
```
import numpy as np

def surrogate(model, variables, rom=None):
    independent_vars_coded = {}
    independent_vars = variables.copy()
    for var, value in independent_vars.items():
        independent_vars_coded[var] = np.interp(value,  model.levels[f'{var}'].values, model.levels.index.values)

    L_predicted = []
    for dependent in model.dependent_variables:
        L_predicted.append(model[dependent].predict(independent_vars_coded).item())

    L_predicted = np.array(L_predicted)

    if rom is not None:
        L_predicted = rom.inverse_transform(L_predicted)

    return L_predicted

surrogate_solution = surrogate(
    model=fm_ccd_full,
    variables={
        't_wall': 0.006044329228014826,
        'CP3_y': -0.001277584765410799,
        'T_0in': 616.0,
        'p_0in': 456429.13834591914,
        'T_e': 318.0
    },
    rom=rom)
```

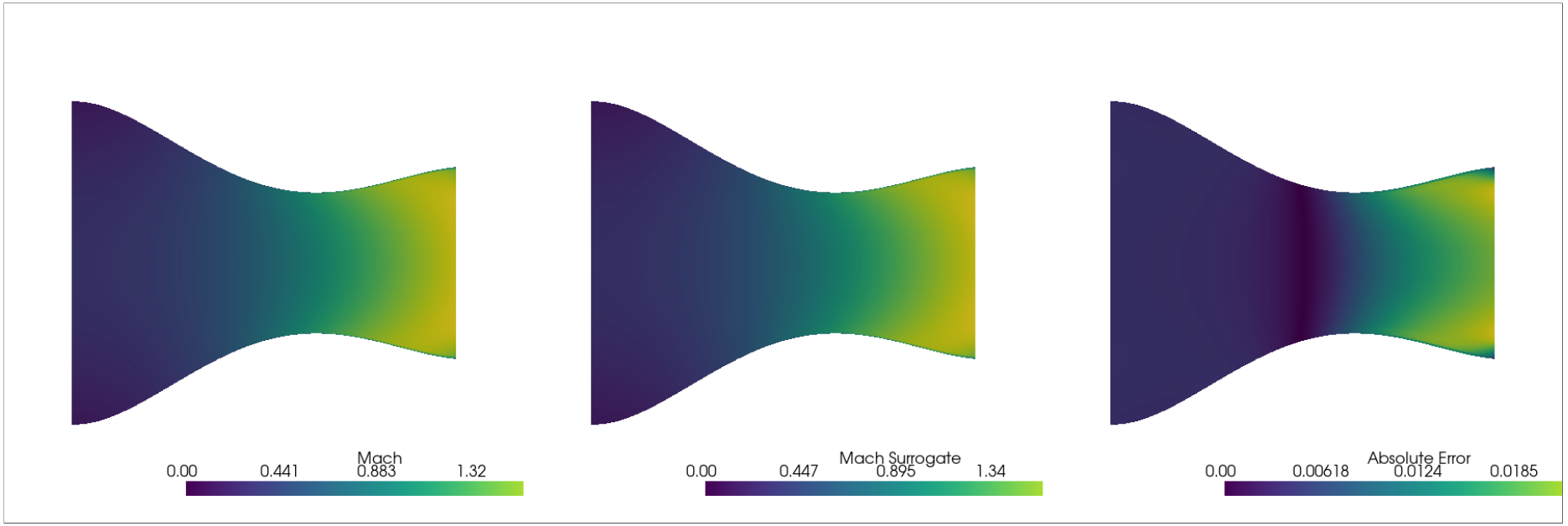
inverse_transform

In [12]:
```
from plot_2d import plot_property
```

In [13]:
```
plot_property(
            prop='Pressure',
    unseeing_path='/home/ppiper/Dropbox/local/github/explann/data/lhs/10/SU2/outputs/cht_setupSU2.vtm',
    surrogate_solution=surrogate_solution,
    hfdh=hfdh,
    solid=False)
```



In [14]:
```
plot_property(
            prop='Mach',
    unseeing_path='/home/ppiper/Dropbox/local/github/explann/data/lhs/10/SU2/outputs/cht_setupSU2.vtm',
    surrogate_solution=surrogate_solution,
    hfdh=hfdh,
    solid=False)
```



In [15]:
```
plot_property(
            prop='Temperature',
    unseeing_path='/home/ppiper/Dropbox/local/github/explann/data/lhs/10/SU2/outputs/cht_setupSU2.vtm',
    surrogate_solution=surrogate_solution,
    hfdh=hfdh,
    solid=True)
```