denavit_hartenberg_modified

April 15, 2022

```
[ ]: import sympy as sym
     from sympy import sin, cos, Matrix, S, solveset, tan, atan, acos, asin, atan2,␣
      ↪sqrt
     from sympy.solvers.solveset import nonlinsolve
     import plotly.express as px
     import plotly.graph_objects as go
     import numpy as np
     import pandas as pd
     import time
```

0.0.1 Symbol declaration

```
[ ]: a = sym.symbols('a_0 a_1 a_2 a_3 a_4 a_5 a_6')
     d = sym.symbols('d_0 d_1 d_2 d_3 d_4 d_5 d_6')
     th = sym.symbols('theta_0 theta_1 theta_2 theta_3 theta_4 theta_5 theta_6')
     T = sym.symbols('T T^0_1 T^1_2 T^2_3 T^3_4 T^4_5 T^5_6')
     T0 = sym.symbols('T T T^0_2 T^0_3 T^0_4 T^0_5 T^0_6')
```

0.0.2 Substitutions for concise matrix representations

```
[ ]: c = {
       1: sym.Symbol('c_1'),
       2: sym.Symbol('c_2'),
       3: sym.Symbol('c_3'),
       4: sym.Symbol('c_4'),
       5: sym.Symbol('c_5'),
       6: sym.Symbol('c_6'),
       12: sym.Symbol('c_12'),
       13: sym.Symbol('c_13'),
       14: sym.Symbol('c_14'),
       15: sym.Symbol('c_15'),
       34: sym.Symbol('c_34'),
       45: sym.Symbol('c_45'),
       345: sym.Symbol('c_345'),
       }
     s = {
       1: sym.Symbol('s_1'),
       2: sym.Symbol('s_2'),
       3: sym.Symbol('s_3'),
       4: sym.Symbol('s_4'),
       5: sym.Symbol('s_5'),
       6: sym.Symbol('s_6'),
       12: sym.Symbol('s_12'),
       13: sym.Symbol('s_13'),
       14: sym.Symbol('s_14'),
       15: sym.Symbol('s_15'),
       34: sym.Symbol('s_34'),
       45: sym.Symbol('s_45'),
       345: sym.Symbol('s_345'),
       }

     substitutions = [
       (cos(th[1]), c[1]),
       (cos(th[2]), c[2]),
       (cos(th[3]), c[3]),
       (cos(th[4]), c[4]),
       (cos(th[5]), c[5]),
       (cos(th[6]), c[6]),
       (sin(th[1]), s[1]),
       (sin(th[2]), s[2]),
       (sin(th[3]), s[3]),
       (sin(th[4]), s[4]),
       (sin(th[5]), s[5]),
       (sin(th[6]), s[6]),
       (sin(th[1] + th[2]), s[12]),
       (cos(th[1] + th[2]), c[12]),
       (sin(th[3] + th[4]), s[34]),
       (cos(th[3] + th[4]), c[34]),
       (sin(th[4] + th[5]), s[45]),
       (cos(th[4] + th[5]), c[45]),
       (sin(th[3] + th[4] + th[5]), s[345]),
       (cos(th[3] + th[4] + th[5]), c[345]),
     ]
```

0.0.3 Transformation matrices

[ ]:
```
T_0_1 = Matrix(
  [
    [cos(th[1]), -sin(th[1]), 0, 0],
    [sin(th[1]),  cos(th[1]), 0, 0],
    [         0,           0, 1, 0],
    [         0,           0, 0, 1]
  ]
)
```

```
T_1_2 = Matrix(
  [
    [cos(th[2]), -sin(th[2]), 0, a[1]],
    [sin(th[2]),  cos(th[2]), 0,    0],
    [         0,           0, 1,    0],
    [         0,           0, 0,    1]
  ]
)
T_2_3 = Matrix(
  [
    [cos(th[3]), -sin(th[3]),  0, 0],
    [         0,           0, -1, 0],
    [sin(th[3]),  cos(th[3]),  0, 0],
    [         0,           0,  0, 1]
  ]
)
T_3_4 = Matrix(
  [
    [ cos(th[4]), -sin(th[4]), 0, a[3]],
    [          0,           0, 1,    0],
    [-sin(th[4]), -cos(th[4]), 0,    0],
    [          0,           0, 0,    1]
  ]
)
T_4_5 = Matrix(
  [
    [cos(th[5]), -sin(th[5]), 0, a[4]],
    [sin(th[5]),  cos(th[5]), 0,    0],
    [         0,           0, 1,    0],
    [         0,           0, 0,    1]
  ]
)
T_5_6 = Matrix(
  [
    [1, 0, 0, a[5]],
    [0, 1, 0,    0],
    [0, 0, 1,    0],
    [0, 0, 0,    1]
  ]
)

# Display equations + LaTex code
display(sym.Eq(T[1], T_0_1.subs(substitutions), evaluate=False))
# sym.print_latex(sym.Eq(T[1], T_0_1.subs(substitutions), evaluate=False))
display(sym.Eq(T[2], T_1_2.subs(substitutions), evaluate=False))
# sym.print_latex(sym.Eq(T[2], T_1_2.subs(substitutions), evaluate=False))
display(sym.Eq(T[3], T_2_3.subs(substitutions), evaluate=False))
```

```
# sym.print_latex(sym.Eq(T[3], T_2_3.subs(substitutions), evaluate=False))
display(sym.Eq(T[4], T_3_4.subs(substitutions), evaluate=False))
# sym.print_latex(sym.Eq(T[4], T_3_4.subs(substitutions), evaluate=False))
display(sym.Eq(T[5], T_4_5.subs(substitutions), evaluate=False))
# sym.print_latex(sym.Eq(T[5], T_4_5.subs(substitutions), evaluate=False))
display(sym.Eq(T[6], T_5_6.subs(substitutions), evaluate=False))
# sym.print_latex(sym.Eq(T[6], T_5_6.subs(substitutions), evaluate=False))
```

$$T_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & a_1 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} c_3 & -s_3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^3 = \begin{bmatrix} c_4 & -s_4 & 0 & a_3 \\ 0 & 0 & 1 & 0 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^4 = \begin{bmatrix} c_5 & -s_5 & 0 & a_4 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6^5 = \begin{bmatrix} 1 & 0 & 0 & a_5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

0.0.4 Derived transformation matrices

[ ]:
```
# Multiplication and trigonometric simplification
T_0_2 = sym.trigsimp(T_0_1 * T_1_2)
T_0_3 = sym.trigsimp(T_0_2 * T_2_3)
T_0_4 = sym.trigsimp(T_0_3 * T_3_4)
T_0_5 = sym.trigsimp(T_0_4 * T_4_5)
T_0_6 = sym.trigsimp(T_0_5 * T_5_6)

# Display + LaTeX code
display(sym.Eq(T0[2], T_0_2.subs(substitutions), evaluate=False))
# sym.print_latex(sym.Eq(T0[2], T_0_2.subs(substitutions), evaluate=False))
```

```
display(sym.Eq(T0[3], T_0_3.subs(substitutions), evaluate=False))
# sym.print_latex(sym.Eq(T0[3], T_0_3.subs(substitutions), evaluate=False))
display(sym.Eq(T0[4], T_0_4.subs(substitutions), evaluate=False))
# sym.print_latex(sym.Eq(T0[4], T_0_4.subs(substitutions), evaluate=False))
display(sym.Eq(T0[5], T_0_5.subs(substitutions), evaluate=False))
# sym.print_latex(sym.Eq(T0[5], T_0_5.subs(substitutions), evaluate=False))
display(sym.Eq(T0[6], T_0_6.subs(substitutions), evaluate=False))
# sym.print_latex(sym.Eq(T0[6], T_0_6.subs(substitutions), evaluate=False))
```

$$T_2^0 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1c_1 \\ s_{12} & c_{12} & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^0 = \begin{bmatrix} c_{12}c_3 & -c_{12}s_3 & s_{12} & a_1c_1 \\ c_3s_{12} & -s_{12}s_3 & -c_{12} & a_1s_1 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^0 = \begin{bmatrix} c_{12}c_3c_4 - s_{12}s_4 & -c_{12}c_3s_4 - c_4s_{12} & -c_{12}s_3 & a_1c_1 + a_3c_{12}c_3 \\ c_{12}s_4 + c_3c_4s_{12} & c_{12}c_4 - c_3s_{12}s_4 & -s_{12}s_3 & a_1s_1 + a_3c_3s_{12} \\ c_4s_3 & -s_3s_4 & c_3 & a_3s_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^0 = \begin{bmatrix} c_{12}c_3c_{45} - s_{12}s_{45} & -c_{12}c_3s_{45} - c_{45}s_{12} & -c_{12}s_3 & a_1c_1 + a_3c_{12}c_3 + a_4\left(c_{12}c_3c_4 - s_{12}s_4\right) \\ c_{12}s_{45} + c_3c_{45}s_{12} & c_{12}c_{45} - c_3s_{12}s_{45} & -s_{12}s_3 & a_1s_1 + a_3c_3s_{12} + a_4\left(c_{12}s_4 + c_3c_4s_{12}\right) \\ c_{45}s_3 & -s_3s_{45} & c_3 & s_3\left(a_3 + a_4c_4\right) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6^0 = \left[\begin{matrix} c_{12}c_3c_{45} - s_{12}s_{45} & -c_{12}c_3s_{45} - c_{45}s_{12} & -c_{12}s_3 & a_1c_1 + a_3c_{12}c_3 + a_4\left(c_{12}c_3c_4 - s_{12}s_4\right) + a_5\left(c_{12}c_3c_{45} - s_{12}s_{45} \right. \\ c_{12}s_{45} + c_3c_{45}s_{12} & c_{12}c_{45} - c_3s_{12}s_{45} & -s_{12}s_3 & a_1s_1 + a_3c_3s_{12} + a_4\left(c_{12}s_4 + c_3c_4s_{12}\right) + a_5\left(c_{12}s_{45} + c_3c_{45}s_{12} \right. \\ c_{45}s_3 & -s_3s_{45} & c_3 & s_3\left(a_3 + a_4c_4 + a_5c_{45}\right) \\ 0 & 0 & 0 & 1 \end{matrix}\right.$$

0.0.5 Translations in 3D

[ ]:
```
# Extracting translation vector from matrix
T_0_1_tran = T_0_1[0:3, 3]
T_0_2_tran = T_0_2[0:3, 3]
T_0_3_tran = T_0_3[0:3, 3]
T_0_4_tran = T_0_4[0:3, 3]
T_0_5_tran = T_0_5[0:3, 3]
T_0_6_tran = T_0_6[0:3, 3]
T_0_3_tran
```

[ ]:

$$\begin{bmatrix} a_1\cos\left(\theta_1\right) \\ a_1\sin\left(\theta_1\right) \\ 0 \end{bmatrix}$$

Lambdify translations and transformations for numpy arrays

```
[ ]: # Lambdifying translations
     # T_0_1_np = sym.lambdify((a, d, th), T_0_1_tran, modules='numpy')
     # T_0_2_np = sym.lambdify((a, d, th), T_0_2_tran, modules='numpy')
     # T_0_3_np = sym.lambdify((a, d, th), T_0_3_tran, modules='numpy')
     # T_0_4_np = sym.lambdify((a, d, th), T_0_4_tran, modules='numpy')
     # T_0_5_np = sym.lambdify((a, d, th), T_0_5_tran, modules='numpy')
     # T_0_6_np = sym.lambdify((a, d, th), T_0_6_tran, modules='numpy')
     T_0_1_np = sym.lambdify((a, th), T_0_1_tran, modules='numpy')
     T_0_2_np = sym.lambdify((a, th), T_0_2_tran, modules='numpy')
     T_0_3_np = sym.lambdify((a, th), T_0_3_tran, modules='numpy')
     T_0_4_np = sym.lambdify((a, th), T_0_4_tran, modules='numpy')
     T_0_5_np = sym.lambdify((a, th), T_0_5_tran, modules='numpy')
     T_0_6_np = sym.lambdify((a, th), T_0_6_tran, modules='numpy')

     # Lambdifying end effector transformation
     T_0_6_full_np = sym.lambdify((a, th), T_0_6, modules='numpy')

     # Grid spaced at 1° for visualization of 3 angles
     th_mcp_aa, th_mcp_fe, th_cmc_fe = np.radians(np.mgrid[-15:15:31j, -10:90:101j,␣
      ↪0:5:6j])
     # th_mcp_aa, th_mcp_fe = np.radians(np.mgrid[-0:-0:2j, 90:90:2j])
     # th_cmc_fe = np.zeros_like(th_mcp_aa)

     # Beta values measured from bones
     # beta = [0, np.radians(2), np.radians(0), np.radians(7), np.radians(-3)]
     # Beta values = 0 for comparison
     beta = [0, 0, 0, 0, 0]

     # Modified DH parameters - theta
     th_values = np.array([
       np.zeros_like(th_mcp_aa), # th_0
       th_cmc_fe - beta[1], # th_1
       th_mcp_fe + beta[2], # th_2
       th_mcp_aa, # th_3
       # th_values for circular grasping
       # 0.75 * th_mcp_fe + beta[3], # th_4
       # 0.5 * th_mcp_fe + beta[4], # th_5
       # # th_values for prismatic grasping
       0.66666667 * th_mcp_fe + beta[3], # th_4
       0.33333333 * th_mcp_fe + beta[4], # th_5
       np.zeros_like(th_mcp_aa), # th_6
     ])

     # Link lengths measured from bones
     L = [0, 6.34, 4.26, 2.51, 1.80]
     a_values = np.array([
       np.zeros_like(th_mcp_aa), #a_0
```

```
  np.full_like(th_mcp_aa, L[1]), # a_1
  np.zeros_like(th_mcp_aa), # a_2
  np.full_like(th_mcp_aa, L[2]), # a_3
  np.full_like(th_mcp_aa, L[3]), # a_4
  np.full_like(th_mcp_aa, L[4]), # a_5
  np.zeros_like(th_mcp_aa), # a_6
])
# d_values = np.array([
#   np.zeros_like(th_mcp_aa), # d_0
#   np.zeros_like(th_mcp_aa), # d_1
#   np.zeros_like(th_mcp_aa), # d_2
#   np.zeros_like(th_mcp_aa), # d_3
#   np.zeros_like(th_mcp_aa), # d_4
#   np.zeros_like(th_mcp_aa), # d_5
#   np.zeros_like(th_mcp_aa), # d_6
# ])
# th_values.shape, a_values.shape, d_values.shape
# th_mcp_aa.shape, th_mcp_fe.shape, th_cmc_fe.shape
```

Calculate x, y and z coordinates of every joint

```
[ ]: # x1, y1, z1 = T_0_1_np(a_values, d_values, th_values)
x1, y1, z1 = T_0_1_np(a_values, th_values)
x1, y1, z1 = x1[0], y1[0], z1[0]
x0, y0, z0 = np.zeros_like(x1), np.zeros_like(y1), np.zeros_like(z1)
# x2, y2, z2 = T_0_2_np(a_values, d_values, th_values)
x2, y2, z2 = T_0_2_np(a_values, th_values)
x2, y2, z2 = x2[0], y2[0], z2[0]
# x3, y3, z3 = T_0_3_np(a_values, d_values, th_values)
x3, y3, z3 = T_0_3_np(a_values, th_values)
x3, y3, z3 = x3[0], y3[0], z3[0]
# x4, y4, z4 = T_0_4_np(a_values, d_values, th_values)
x4, y4, z4 = T_0_4_np(a_values, th_values)
x4, y4, z4 = x4[0], y4[0], z4[0]
# x5, y5, z5 = T_0_5_np(a_values, d_values, th_values)
x5, y5, z5 = T_0_5_np(a_values, th_values)
x5, y5, z5 = x5[0], y5[0], z5[0]
# x6, y6, z6 = T_0_6_np(a_values, d_values, th_values)
x6, y6, z6 = T_0_6_np(a_values, th_values)
x6, y6, z6 = x6[0], y6[0], z6[0]

# some vectors are all zeros but wrong dimension
# z0.shape
z0, z1, z2, z3, x0, x1, y0, y1 = [np.zeros_like(z4)] * 8

df_coord = pd.DataFrame(data={
  'x': np.concatenate((
    x0.flatten(), x1.flatten(), x2.flatten(), x3.flatten(), x4.flatten(),
    x5.flatten(), x6.flatten()
    )),
  'y': np.concatenate((
    y0.flatten(), y1.flatten(), y2.flatten(), y3.flatten(), y4.flatten(),
    y5.flatten(), y6.flatten()
    )),
  'z': np.concatenate((
    z0.flatten(), z1.flatten(), z2.flatten(), z3.flatten(), z4.flatten(),
    z5.flatten(), z6.flatten()
    )),
  'joint': np.array([len(x1.flatten()) * [i] for i in range(7)]).flatten(),
  'th_mcp_fe': np.tile(th_mcp_fe.flatten(), 7),
  'th_mcp_aa': np.tile(th_mcp_aa.flatten(), 7),
  'th_cmc_fe': np.tile(th_cmc_fe.flatten(), 7),
  'th_values': th_values.flatten(),
  'a_values': a_values.flatten(),
  # 'd_values': d_values.flatten(),
  })
# Save as circular/prismatic
# df_coord.to_csv('DH_modified_index_circular.csv')
# df_coord.to_csv('DH_modified_index_prismatic.csv')
# Save as circular/prismatic with beta = 0 for comparison
# df_coord.to_csv('DH_modified_index_circular_no_beta.csv')
df_coord.to_csv('DH_modified_index_prismatic_no_beta.csv')
```

[ ]:
```
df_coord['z'] = df_coord['z'] * -1
df_coord['size'] = 2
fig = go.Figure(
  data=go.Scatter3d(
    x=df_coord['z'],
    y=df_coord['y'],
    z=df_coord['x'],
    marker=dict(
      size=2,
      # color=df_coord['joint'],
      color=np.degrees(df_coord['th_mcp_fe']),
      colorscale='viridis_r',
      showscale=True,
      colorbar=dict(title='theta_mpc_fe')
    ),
    line=dict(
      width=0
    ),
    mode='markers',
  )
)
```

```
fig.update_yaxes(
   scaleanchor = "x",
   scaleratio = 1,
 )
fig.update_layout(scene_aspectmode='auto')
fig.update_layout(
  scene = dict(
    zaxis = dict(nticks=20, range=[0,18],),
    yaxis = dict(nticks=20, range=[-8,10],),
    xaxis = dict(nticks=20, range=[-9,9],),),
  height=700,
  width=600,
  margin=dict(r=5, l=5, b=5, t=5))
fig.show()
```

Further simplify matrices by substituting circular and prismatic constraints

[ ]:
```
# Finger ROM angles
th_f = {
  'CMC': sym.Symbol('theta_CMC,FE', nonegative=True, real=True),
  'MCP_FE': sym.Symbol('theta_MCP,FE', real=True),
  'MCP_AA': sym.Symbol('theta_MCP,AA', real=True),
  'PIP': sym.Symbol('theta_PIP,FE', real=True),
  'DIP': sym.Symbol('theta_DIP,FE', real=True),
  }
# Finger beta angles (curvature)
beta_s = {
  1: sym.Symbol('beta_1', real=True),
  2: sym.Symbol('beta_2', real=True),
  3: sym.Symbol('beta_3', real=True),
  4: sym.Symbol('beta_4', real=True),
  }
# Frame to frame lengths
L_s = {
  1: sym.Symbol('L_1', positive=True, real=True),
  2: sym.Symbol('L_2', positive=True, real=True),
  3: sym.Symbol('L_3', positive=True, real=True),
  4: sym.Symbol('L_4', positive=True, real=True),
  }

# Symbol for linear dependency
K = {
  'PIP': sym.Symbol('K_PIP', nonnegative=True, real=True),
  'DIP': sym.Symbol('K_DIP', nonnegative=True, real=True),
  }
# Substitutions for generic grasp type
subs_grasp = [
  (th[1], th_f['CMC'] - beta_s[1]),
  (th[2], th_f['MCP_FE'] + beta_s[2]),
  (th[3], th_f['MCP_AA']),
  (th[4], th_f['PIP'] + beta_s[3]),
  (th[5], th_f['DIP'] + beta_s[4]),
  (a[1], L_s[1]),
  (a[3], L_s[2]),
  (a[4], L_s[3]),
  (a[5], L_s[4]),
  (th_f['PIP'], K['PIP'] * th_f['MCP_FE']),
  (th_f['DIP'], K['DIP'] * th_f['MCP_FE']),
]

T_0_2_circ = sym.simplify(T_0_2.subs(subs_grasp))
# T_0_2_prism = sym.trigsimp(T_0_2.subs(subs_prismatic))
T_0_3_circ = sym.simplify(T_0_3.subs(subs_grasp))
# T_0_3_prism = sym.trigsimp(T_0_3.subs(subs_prismatic))
T_0_4_circ = sym.simplify(T_0_4.subs(subs_grasp))
# T_0_4_prism = sym.trigsimp(T_0_4.subs(subs_prismatic))
T_0_5_circ = sym.simplify(T_0_5.subs(subs_grasp))
# T_0_5_prism = sym.trigsimp(T_0_5.subs(subs_prismatic))
T_0_6_circ = sym.simplify(T_0_6.subs(subs_grasp))
# T_0_6_prism = sym.trigsimp(T_0_6.subs(subs_prismatic))
display(sym.Eq(T0[2], T_0_2_circ, evaluate=False))
# display(sym.Eq(T0[2], T_0_2_prism, evaluate=False))
# sym.print_latex(sym.Eq(T0[2], T_0_2_circ, evaluate=False))
display(sym.Eq(T0[3], T_0_3_circ, evaluate=False))
# display(sym.Eq(T0[3], T_0_3_prism, evaluate=False))
# sym.print_latex(sym.Eq(T0[3], T_0_3_circ, evaluate=False))
display(sym.Eq(T0[4], T_0_4_circ, evaluate=False))
# sym.print_latex(sym.Eq(T0[4], T_0_4_circ, evaluate=False))
# display(sym.Eq(T0[4], T_0_4_prism, evaluate=False))
# sym.print_latex(sym.Eq(T0[4], T_0_4_prism, evaluate=False))
display(sym.Eq(T0[5], T_0_5_circ, evaluate=False))
# sym.print_latex(sym.Eq(T0[5], T_0_5_circ, evaluate=False))
# display(sym.Eq(T0[5], T_0_5_prism, evaluate=False))
# sym.print_latex(sym.Eq(T0[5], T_0_5_prism, evaluate=False))
display(sym.Eq(T0[6], T_0_6_circ, evaluate=False))
# sym.print_latex(sym.Eq(T0[6], T_0_6_circ, evaluate=False))
# display(sym.Eq(T0[6], T_0_6_prism, evaluate=False))
# sym.print_latex(sym.Eq(T0[6], T_0_6_prism, evaluate=False))
```

$$T_2^0 = \left[\begin{matrix}\cos\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) & -\sin\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) & 0 & L_1 \cos\left(\beta_1 - \theta_{CMC,F}\right. \\ \sin\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) & \cos\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) & 0 & -L_1 \sin\left(\beta_1 - \theta_{CMC,F}\right. \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1\end{matrix}\right.$$

$$T_3^0 = \begin{bmatrix} \cos\left(\theta_{MCP,AA}\right)\cos\left(-\beta_1+\beta_2+\theta_{CMC,FE}+\theta_{MCP,FE}\right) & -\sin\left(\theta_{MCP,AA}\right)\cos\left(-\beta_1+\beta_2+\theta_{CMC,FE}+\theta_{MCP,F} \\ \sin\left(-\beta_1+\beta_2+\theta_{CMC,FE}+\theta_{MCP,FE}\right)\cos\left(\theta_{MCP,AA}\right) & -\sin\left(\theta_{MCP,AA}\right)\sin\left(-\beta_1+\beta_2+\theta_{CMC,FE}+\theta_{MCP,F} \\ \sin\left(\theta_{MCP,AA}\right) & \cos\left(\theta_{MCP,AA}\right) \\ 0 & 0 \end{bmatrix}$$

$$T_4^0 = \begin{bmatrix} -\sin\left(K_{PIP}\theta_{MCP,FE}+\beta_3\right)\sin\left(-\beta_1+\beta_2+\theta_{CMC,FE}+\theta_{MCP,FE}\right)+\cos\left(\theta_{MCP,AA}\right)\cos\left(K_{PIP}\theta_{MCP,FE}+\beta_3 \\ \sin\left(K_{PIP}\theta_{MCP,FE}+\beta_3\right)\cos\left(-\beta_1+\beta_2+\theta_{CMC,FE}+\theta_{MCP,FE}\right)+\sin\left(-\beta_1+\beta_2+\theta_{CMC,FE}+\theta_{MCP,FE}\right)c \\ \sin\left(\theta_{MCP,AA}\right)\cos\left(K_{PIP}\theta_{MCP,FE}+\beta_3\right) \\ 0 \end{bmatrix}$$

$$T_5^0 = \begin{bmatrix} -\sin\left(-\beta_1+\beta_2+\theta_{CMC,FE}+\theta_{MCP,FE}\right)\sin\left(K_{DIP}\theta_{MCP,FE}+K_{PIP}\theta_{MCP,FE}+\beta_3+\beta_4\right)+\cos\left(\theta_{MCP,AA}\right)c \\ \sin\left(-\beta_1+\beta_2+\theta_{CMC,FE}+\theta_{MCP,FE}\right)\cos\left(\theta_{MCP,AA}\right)\cos\left(K_{DIP}\theta_{MCP,FE}+K_{PIP}\theta_{MCP,FE}+\beta_3+\beta_4\right)+\sin \\ \sin\left(\theta_{MCP,AA}\right)\cos\left(K_{DIP}\theta_{MCP,FE}+K_{PIP} \\ 0 \end{bmatrix}$$

$$T_6^0 = \begin{bmatrix} -\sin\left(-\beta_1+\beta_2+\theta_{CMC,FE}+\theta_{MCP,FE}\right)\sin\left(K_{DIP}\theta_{MCP,FE}+K_{PIP}\theta_{MCP,FE}+\beta_3+\beta_4\right)+\cos\left(\theta_{MCP,AA}\right)c \\ \sin\left(-\beta_1+\beta_2+\theta_{CMC,FE}+\theta_{MCP,FE}\right)\cos\left(\theta_{MCP,AA}\right)\cos\left(K_{DIP}\theta_{MCP,FE}+K_{PIP}\theta_{MCP,FE}+\beta_3+\beta_4\right)+\sin \\ \sin\left(\theta_{MCP,AA}\right)\cos\left(K_{DIP}\theta_{MCP,FE}+K_{PIP} \\ 0 \end{bmatrix}$$

Inverse kinematics - TIP transformation matrix printed as columns

[ ]:
```
display(T0[6])
display('First column')
display(T_0_6_circ[0:3,0])
# sym.print_latex(T_0_6_circ[0:3,0])
display('Second column')
display(T_0_6_circ[0:3,1])
# sym.print_latex(T_0_6_circ[0:3,1])
display('Third column')
display(T_0_6_circ[0:3,2])
# sym.print_latex(T_0_6_circ[0:3,2])
display('Fourth column')
display(T_0_6_circ[0:3,3])
# sym.print_latex(T_0_6_circ[0:3,3])

# Symbols for positions in 3D space px, py and pz from T_0_6
p = {
  'x': sym.Symbol('p_x', real=True),
  'y': sym.Symbol('p_y', real=True),
  'z': sym.Symbol('p_z', real=True),
  2: sym.Symbol('P^0_2'),
  3: sym.Symbol('P^0_3'),
  4: sym.Symbol('P^0_4'),
  5: sym.Symbol('P^0_5'),
  6: sym.Symbol('P^0_6'),
  }
# Symbols for rotations
r = {
  11: sym.Symbol('r_11', real=True),
  12: sym.Symbol('r_12', real=True),
  13: sym.Symbol('r_13', real=True),
  21: sym.Symbol('r_21', real=True),
  22: sym.Symbol('r_22', real=True),
  23: sym.Symbol('r_23', real=True),
  31: sym.Symbol('r_31', real=True),
  32: sym.Symbol('r_32', real=True),
  33: sym.Symbol('r_33', real=True, positive=True), # always in [0.96, 1] for␣
↪MCP_AA [-15, - 15]
  }

# Symbols for composite angles substitutions
u = {
  'CMC': sym.Symbol('u_CMC', real=True),
  'CMC_MCP': sym.Symbol('u_CMC,MCP', real=True),
  'MCP_PIP': sym.Symbol('u_MCP,PIP', real=True),
  'MCP_DIP_PIP': sym.Symbol('u_MCP,DIP,PIP', real=True),
  'MCP_AA': sym.Symbol('u_MCP,AA', real=True),
  'half_DIP': sym.Symbol('u_half,DIP', real=True),
  'half_PIP': sym.Symbol('u_half,PIP', real=True),
  'half_CMC': sym.Symbol('u_half,CMC', real=True),
  'half_MCP': sym.Symbol('u_half,MCP', real=True),
  'tan_DIP': sym.Symbol('u_tan,DIP', real=True),
  'tan_PIP': sym.Symbol('u_tan,PIP', real=True),
  'tan_CMC': sym.Symbol('u_tan,CMC', real=True),
  'tan_MCP': sym.Symbol('u_tan,MCP', real=True),
  'sin_AA': sym.Symbol('u_sin,AA', real=True),
  'half_MCP_PIP': sym.Symbol('u_half,MCP,PIP', real=True),
  'temp': sym.Symbol('u_temp', real=True)
}

# Substitutions for K_DIP and K_PIP
subs_half_angle = [
  (K['DIP']*th_f['MCP_FE']/2, u['half_DIP']),
  (K['PIP']*th_f['MCP_FE']/2, u['half_PIP']),
  (th_f['MCP_FE']/2, u['half_MCP']),
  (th_f['CMC']/2, u['half_CMC']),
]

subs_tan_angle = [
  (tan(u['half_DIP']), u['tan_DIP']),
  (tan(u['half_PIP']), u['tan_PIP']),
  (tan(u['half_MCP']), u['tan_MCP']),
  (tan(u['half_CMC']), u['tan_CMC']),
]
```

```
# Composite angles substitutions
subs_reduce_angles = [
  (K['PIP']*th_f['MCP_FE'] + beta_s[3], u['MCP_PIP']),
  (K['DIP']*th_f['MCP_FE'] + beta_s[4] + u['MCP_PIP'], u['MCP_DIP_PIP']),
  (-th_f['CMC'] + beta_s[1], u['CMC']),
  (th_f['MCP_FE'] + beta_s[2] - u['CMC'], u['CMC_MCP']),
]

# Function for duplicating expression and returning both +- acos expressions
def sign_acos(expr):
  # New expression with all filpped acos signs
  expr2 = expr.replace(acos, lambda args: -1*acos(args))
  return set([expr, expr2])

print('Composite angles substitutions:')
for i in subs_reduce_angles:
    display(sym.Eq(i[1], i[0]))
    # sym.print_latex(sym.Eq(i[1], i[0]))

# Simplification substitutions during calculations
subs_simpl = [
  (r[31]**2 + r[32]**2, 1 - r[33]**2),
  (r[13]**2 + r[23]**2, 1 - r[33]**2),
]
# Simplification substitutions after calculations
r_subs = sym.Symbol('r_s', positive=True, real=True)
subs_simpl_after = [
  (1 - r[33]**2, r_subs)
]

print('Simplification substitutions during calculations:')
for i in subs_simpl:
    display(sym.Eq(i[1], i[0]))
    # sym.print_latex(sym.Eq(i[1], i[0]))

print('Simplification substitutions for concise representation:')
for i in subs_simpl_after:
    display(sym.Eq(i[1], i[0]))
    # sym.print_latex(sym.Eq(i[1], i[0]))

print('Position vectors')
for i, vec in zip(
  (p[2], p[3], p[4], p[5], p[6]),
  (T_0_2_circ, T_0_3_circ, T_0_4_circ, T_0_5_circ, T_0_6_circ)):
  # display(sym.Eq(i, vec[0:3, 3].subs(subs_reduce_angles)))
  display(sym.Eq(i, vec[0:3, 3].subs(subs_reduce_angles), evaluate=False))
  # sym.print_latex(sym.Eq(i, vec[0:3, 3].subs(subs_reduce_angles),␣
↪evaluate=False))

print('\nTIP 6D transform')
T_0_6_reduced = sym.simplify(T_0_6_circ.subs(subs_reduce_angles))
display(T_0_6_reduced)
# sym.print_latex(T_0_6_reduced)
```

T_6^0

```
'First column'
```

$$\left[\begin{matrix}- \sin\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) \sin\left(K_{DIP}\theta_{MCP,FE} + K_{PIP}\theta_{MCP,FE} + \beta_3 + \beta_4\right) + \cos\left(\theta_{MCP,AA}\right) \cos\left(-\beta\right. \\ \sin\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) \cos\left(\theta_{MCP,AA}\right) \cos\left(K_{DIP}\theta_{MCP,FE} + K_{PIP}\theta_{MCP,FE} + \beta_3 + \beta_4\right) + \sin\left(K_{DI}\right. \\ \sin\left(\theta_{MCP,AA}\right) \cos\left(K_{DIP}\theta_{MCP,FE} + K_{PIP}\theta_{MCP,}\right.\end{matrix}\right.$$

```
'Second column'
```

$$\left[\begin{matrix}- \sin\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) \cos\left(K_{DIP}\theta_{MCP,FE} + K_{PIP}\theta_{MCP,FE} + \beta_3 + \beta_4\right) - \sin\left(K_{DIP}\theta_{MCP,FE} + \right. \\ - \sin\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) \sin\left(K_{DIP}\theta_{MCP,FE} + K_{PIP}\theta_{MCP,FE} + \beta_3 + \beta_4\right) \cos\left(\theta_{MCP,AA}\right) + \cos\left(-\beta\right. \\ - \sin\left(\theta_{MCP,AA}\right) \sin\left(K_{DIP}\theta_{MCP,FE} + K_{PIP}\theta_{MC}\right.\end{matrix}\right.$$

```
'Third column'
```

$$\begin{bmatrix}- \sin\left(\theta_{MCP,AA}\right) \cos\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) \\ - \sin\left(\theta_{MCP,AA}\right) \sin\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) \\ \cos\left(\theta_{MCP,AA}\right)\end{bmatrix}$$

```
'Fourth column'
```

$$\left[\begin{matrix}L_1 \cos\left(\beta_1 - \theta_{CMC,FE}\right) + L_2 \cos\left(\theta_{MCP,AA}\right) \cos\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) - L_3 \left(\sin\left(K_{PIP}\theta_{MCP,FE} + \beta_3\right)\right. \\ -L_1 \sin\left(\beta_1 - \theta_{CMC,FE}\right) + L_2 \sin\left(-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE}\right) \cos\left(\theta_{MCP,AA}\right) + L_3 \left(\sin\left(K_{PIP}\theta_{MCP,FE} + \beta_3\right.\right. \\ \end{matrix}\right.$$

```
Composite angles substitutions:
```

$u_{MCP,PIP} = K_{PIP}\theta_{MCP,FE} + \beta_3$

$u_{MCP,DIP,PIP} = K_{DIP}\theta_{MCP,FE} + \beta_4 + u_{MCP,PIP}$

$u_{CMC} = \beta_1 - \theta_{CMC,FE}$

$u_{CMC,MCP} = \beta_2 + \theta_{MCP,FE} - u_{CMC}$

```
Simplification substitutions during calculations:
```

$1 - r_{33}^2 = r_{31}^2 + r_{32}^2$

$1 - r_{33}^2 = r_{13}^2 + r_{23}^2$

```
Simplification substitutions for concise representation:
```

$r_s = 1 - r_{33}^2$

```
Position vectors
```

$$P_{2}^{0} = \left[\begin{matrix}L_{1} \cos{\left(u_{CMC} \right)}\\- L_{1} \sin{\left(u_{CMC} \right)}\\0\end{matrix}\right]$$

$$P_{3}^{0} = \left[\begin{matrix}L_{1} \cos{\left(u_{CMC} \right)}\\- L_{1} \sin{\left(u_{CMC} \right)}\\0\end{matrix}\right]$$

$$P_{4}^{0} = \left[\begin{matrix}L_{1} \cos{\left(u_{CMC} \right)} + L_{2} \cos{\left(\theta_{MCP,AA} \right)} \cos{\left(u_{CMC,MCP} \right)}\\- L_{1} \sin{\left(u_{CMC} \right)} + L_{2} \sin{\left(u_{CMC,MCP} \right)} \cos{\left(\theta_{MCP,AA} \right)}\\L_{2} \sin{\left(\theta_{MCP,AA} \right)}\end{matrix}\right]$$

$$P_{5}^{0} = \left[\begin{matrix}L_{1} \cos{\left(u_{CMC} \right)} + L_{2} \cos{\left(\theta_{MCP,AA} \right)} \cos{\left(u_{CMC,MCP} \right)} - L_{3} \left(\sin{\left(u_{CMC,MCP} \right)} \sin{\left(u_{MCP,PIP} \right)} - \cos{\left(\theta_{MCP,AA} \right)}\\- L_{1} \sin{\left(u_{CMC} \right)} + L_{2} \sin{\left(u_{CMC,MCP} \right)} \cos{\left(\theta_{MCP,AA} \right)} + L_{3} \left(\sin{\left(u_{CMC,MCP} \right)} \cos{\left(\theta_{MCP,AA} \right)} \cos{\left(u_{MCP,PIP} \right)} -\\\left(L_{2} + L_{3} \cos{\left(u_{MCP,PIP} \right)}\right) \sin{\left(\theta_{MCP,AA} \right)}\end{matrix}\right.$$

$$P_{6}^{0} = \left[\begin{matrix}L_{1} \cos{\left(u_{CMC} \right)} + L_{2} \cos{\left(\theta_{MCP,AA} \right)} \cos{\left(u_{CMC,MCP} \right)} - L_{3} \left(\sin{\left(u_{CMC,MCP} \right)} \sin{\left(u_{MCP,PIP} \right)} - \cos{\left(\theta_{MCP,AA} \right)}\\- L_{1} \sin{\left(u_{CMC} \right)} + L_{2} \sin{\left(u_{CMC,MCP} \right)} \cos{\left(\theta_{MCP,AA} \right)} + L_{3} \left(\sin{\left(u_{CMC,MCP} \right)} \cos{\left(\theta_{MCP,AA} \right)} \cos{\left(u_{MCP,PIP} \right)} -\\\left(L_{2} + L_{3} \cos{\left(u_{MCP,\right.}\right.}\end{matrix}\right.$$

```
TIP 6D transform
```

$$\left[\begin{matrix}- \sin{\left(u_{CMC,MCP} \right)} \sin{\left(u_{MCP,DIP,PIP} \right)} + \cos{\left(\theta_{MCP,AA} \right)} \cos{\left(u_{CMC,MCP} \right)} \cos{\left(u_{MCP,DIP,PIP} \right)} & - \sin{\left(u_{CMC,MCP} \right)}\\\sin{\left(u_{CMC,MCP} \right)} \cos{\left(\theta_{MCP,AA} \right)} \cos{\left(u_{MCP,DIP,PIP} \right)} + \sin{\left(u_{MCP,DIP,PIP} \right)} \cos{\left(u_{CMC,MCP} \right)} & - \sin{\left(u_{CMC,MCP} \right)}\\\sin{\left(\theta_{MCP,AA} \right)} \cos{\left(u_{MCP,DIP,PIP} \right)} & \\0 & \end{matrix}\right.$$

0.0.6 Inverse kinematics - Algebraic solution - Overdefined equation system

Solving th__MCP__AA without substitutions - trivial from third column third row

```
[ ]: eq1 = sym.Eq(r[33], sym.simplify(T_0_6_reduced[2, 2]))
     print('Equation')
     display(eq1)
     # sym.print_latex(eq1)

     th_final = dict()
     th_final[th_f['MCP_AA']] = list()
     th_sol = sym.solve(
       eq1,
       th_f['MCP_AA'],
       domain=S.Reals,
       dict=True, minimal=False, simplify=True, implicit=False
       )

     print('Solution')
     i = sym.simplify(th_sol[1][th_f['MCP_AA']])
     # acos returns between [-pi, pi]
```

```
i = sign_acos(i)
for j in i:
  # Same as atan2
  j = j.rewrite(atan)
  display(sym.Eq(th_f['MCP_AA'], j))
  th_final[th_f['MCP_AA']].append(j)
  # sym.print_latex(sym.Eq(th_f['MCP_AA'], j))
```

```
Equation
```

$r_{33} = \cos\left(\theta_{MCP,AA}\right)$

```
Solution
```

$$\theta_{MCP,AA} = \operatorname{atan}\left(\frac{\sqrt{1 - r_{33}^2}}{r_{33}}\right)$$

$$\theta_{MCP,AA} = -\operatorname{atan}\left(\frac{\sqrt{1 - r_{33}^2}}{r_{33}}\right)$$

Solving u_MCP_DIP_PIP with substitutions - from second and third column, third row

```
[ ]: # Substitute sin th_mcp_aa
subs_sin_AA = [(sin(th_f['MCP_AA']), u['sin_AA'])]
eq2 = sym.Eq(-r[32] + r[31], (-T_0_6_reduced[2, 1] + T_0_6_reduced[2, 0]).subs(
  subs_sin_AA
))
# eq2 = sym.Eq(r[32], sym.simplify(T_0_6_reduced[2, 1]))

print('Equation')
display(eq2)
# sym.print_latex(eq2.subs(subs_simpl))

th_sol = sym.solve(
  eq2,
  u['MCP_DIP_PIP'],
  domain=S.Reals,
  dict=True, minimal=False, simplify=True, implicit=False
  )
# Keep only solution in domain [-pi, pi]
print('Solution')
subs_solve = dict()
subs_solve['mcp_dip_pip'] = []
# 2 solutions for equation type: a cos   + b sin   = c
# If solutions are missing, add/subtract 2*pi (tan period is pi)
for sol in th_sol:
  i = sym.simplify(sol[u['MCP_DIP_PIP']]).subs([(j, i) for i, j in subs_sin_AA])
  display(sym.Eq(u['MCP_DIP_PIP'], i))
  subs_solve['mcp_dip_pip'].append(i)
  # sym.print_latex(sym.Eq(u['MCP_DIP_PIP'], th_sol[1][u['MCP_DIP_PIP']]))

free_symbols = set()
print('Free symbols')
for eq in subs_solve['mcp_dip_pip']:
  free_symbols = free_symbols | eq.free_symbols
print(free_symbols)
```

```
Equation
```

$r_{31} - r_{32} = u_{sin,AA} \sin\left(u_{MCP,DIP,PIP}\right) + u_{sin,AA} \cos\left(u_{MCP,DIP,PIP}\right)$

```
Solution
```

$$u_{MCP,DIP,PIP} = 2 \operatorname{atan}\left(\frac{-\sqrt{-r_{31}^{2} + 2 r_{31} r_{32} - r_{32}^{2} + 2 \sin^{2}\left(\theta_{MCP,AA}\right)} + \sin\left(\theta_{MCP,AA}\right)}{r_{31} - r_{32} + \sin\left(\theta_{MCP,AA}\right)}\right)$$

$$u_{MCP,DIP,PIP} = 2 \operatorname{atan}\left(\frac{\sqrt{-r_{31}^{2} + 2 r_{31} r_{32} - r_{32}^{2} + 2 \sin^{2}\left(\theta_{MCP,AA}\right)} + \sin\left(\theta_{MCP,AA}\right)}{r_{31} - r_{32} + \sin\left(\theta_{MCP,AA}\right)}\right)$$

```
Free symbols
{r_32, theta_MCP,AA, r_31}
```

Solving u__CMC__MCP with substitutions - from r__11, r__12, r__13, r__21, r__22, r__23

```
[ ]: # eq3 = sym.Eq(r[23], sym.simplify(T_0_6_reduced[1, 2]))
eq3 = sym.Eq(-r[23] - r[13], (-T_0_6_reduced[1, 2] - T_0_6_reduced[0, 2]).
 ↪subs(subs_sin_AA))
# eq4 did not produce more solutions
# eq4 = sym.simplify((T_0_6_reduced[0, 1] + T_0_6_reduced[1, 0]) /␣
 ↪(T_0_6_reduced[0, 0] - T_0_6_reduced[1, 1]))
# eq4 = sym.Eq((r[12] + r[21]) / (r[11] - r[22]), eq4.subs(
#   [(u['MCP_DIP_PIP'], subs_solve['mcp_dip_pip'][0])]
#    ))
print('Equation')
display(eq3)
# display(eq4)
# sym.print_latex(eq3.subs(subs_simpl))

th_sol = sym.solve(
  eq3,
  u['CMC_MCP'],
  domain=S.Reals,
  dict=True, minimal=False, simplify=True, implicit=False
  )
# Keep only solution in domain [-pi, pi]
print('Solution')
```

```
subs_solve['cmc_mcp'] = []
# 2 solutions for equation type: a cos   + b sin   = c
# If solutions are missing, add/subtract 2*pi (tan period is pi)
for sol in th_sol:
  i = sym.simplify(sol[u['CMC_MCP']]).subs([(j, i) for i, j in subs_sin_AA])
  display(sym.Eq(u['CMC_MCP'], i))
  subs_solve['cmc_mcp'].append(i)
# sym.print_latex(sym.Eq(u['CMC_MCP'], th_sol[1][u['CMC_MCP']]))

# th_sol = sym.solve(
#   eq4,
#   u['CMC_MCP'],
#   domain=S.Reals,
#   dict=True, minimal=False, simplify=True, implicit=False
#   )
# Keep only solution in domain [-pi/2, pi/2]
# display(sym.Eq(u['CMC_MCP'], th_sol[0][u['CMC_MCP']]))
# display(sym.Eq(u['CMC_MCP'], th_sol[1][u['CMC_MCP']]))
# sym.print_latex(sym.Eq(u['CMC_MCP'], th_sol[1][u['CMC_MCP']]))

# subs_solve['cmc_mcp'].append(th_sol[0][u['CMC_MCP']])

free_symbols = set()
print('Free symbols')
for eq in subs_solve['cmc_mcp']:
  free_symbols = free_symbols | eq.free_symbols
print(free_symbols)
```

```
Equation
```

$$-r_{13} - r_{23} = u_{sin,AA} \sin\left(u_{CMC,MCP}\right) + u_{sin,AA} \cos\left(u_{CMC,MCP}\right)$$

```
Solution
```

$$u_{CMC,MCP} = -2\operatorname{atan}\left(\frac{-\sqrt{-r_{13}^2 - 2r_{13}r_{23} - r_{23}^2 + 2\sin^2\left(\theta_{MCP,AA}\right)} + \sin\left(\theta_{MCP,AA}\right)}{r_{13} + r_{23} - \sin\left(\theta_{MCP,AA}\right)}\right)$$

$$u_{CMC,MCP} = -2\operatorname{atan}\left(\frac{\sqrt{-r_{13}^2 - 2r_{13}r_{23} - r_{23}^2 + 2\sin^2\left(\theta_{MCP,AA}\right)} + \sin\left(\theta_{MCP,AA}\right)}{r_{13} + r_{23} - \sin\left(\theta_{MCP,AA}\right)}\right)$$

```
Free symbols
{r_13, theta_MCP,AA, r_23}
```

Solving u__MCP__PIP with substitutions - from fourth column third row

```
[ ]: """
     subs_cos_mcp_dip_pip = [
       (cos(u['MCP_DIP_PIP']), u['temp'])
     ]
     # Change all sin/cos to substitutions
     eq4 = sym.Eq(p['z'], (T_0_6_reduced[2, 3])).subs(subs_sin_AA).
      ↪subs(subs_cos_mcp_dip_pip)
     # Rewrite using tan for polynomial
     eq4 = eq4.rewrite(tan)
     print('Equation')
     display(eq4)
     # sym.print_latex(eq4.subs(subs_simpl))

     th_sol = sym.solve(
       eq4,
       u['MCP_PIP'],
       domain=S.Reals,
       dict=True, minimal=False, simplify=True, implicit=False
       )
     # Keep only solution in domain [-pi, pi]
     # If solutions are missing, add/subtract 2*pi (tan period is pi)
     print('Solution')
     subs_solve['mcp_pip'] = []
     # Rewrite using atan and insert substitutions for temp and sin_AA
     th_sol[0][u['MCP_PIP']] = th_sol[0][u['MCP_PIP']].rewrite(atan).subs(
       [(j, i) for i, j in subs_sin_AA]).subs(
         [(j, i) for i, j in subs_cos_mcp_dip_pip])
     subs_solve['mcp_pip'].append(th_sol[0][u['MCP_PIP']])
     th_sol[1][u['MCP_PIP']] = th_sol[1][u['MCP_PIP']].rewrite(atan).subs(
       [(j, i) for i, j in subs_sin_AA]).subs(
         [(j, i) for i, j in subs_cos_mcp_dip_pip])
     subs_solve['mcp_pip'].append(th_sol[1][u['MCP_PIP']])
     display(sym.Eq(u['MCP_PIP'], th_sol[0][u['MCP_PIP']]))
     display(sym.Eq(u['MCP_PIP'], th_sol[1][u['MCP_PIP']]))
     # sym.print_latex(sym.Eq(u['MCP_PIP'], th_sol[1][u['MCP_PIP']]))

     free_symbols = set()
     print('Free symbols')
     for eq in subs_solve['mcp_pip']:
       free_symbols = free_symbols | eq.free_symbols
     print(free_symbols)
     """
```

[ ]:
```
"\nsubs_cos_mcp_dip_pip = [\n  (cos(u['MCP_DIP_PIP']), u['temp'])\n]\n# Change
all sin/cos to substitutions\neq4 = sym.Eq(p['z'], (T_0_6_reduced[2,
3])).subs(subs_sin_AA).subs(subs_cos_mcp_dip_pip)\n# Rewrite using tan for
polynomial\neq4 = eq4.rewrite(tan)\nprint('Equation')\ndisplay(eq4)\n#
sym.print_latex(eq4.subs(subs_simpl))\n\nth_sol = sym.solve(\n  eq4,\n
u['MCP_PIP'],\n  domain=S.Reals,\n  dict=True, minimal=False, simplify=True,
implicit=False\n  )\n# Keep only solution in domain [-pi, pi]\n# If solutions
```

```
are missing, add/subtract 2*pi (tan period is
pi)\nprint('Solution')\nsubs_solve['mcp_pip'] = []\n# Rewrite using atan and
insert substitutions for temp and sin_AA \nth_sol[0][u['MCP_PIP']] =
th_sol[0][u['MCP_PIP']].rewrite(atan).subs(\n  [(j, i) for i, j in
subs_sin_AA]).subs(\n    [(j, i) for i, j in subs_cos_mcp_dip_pip])\nsubs_solve[
'mcp_pip'].append(th_sol[0][u['MCP_PIP']])\nth_sol[1][u['MCP_PIP']] =
th_sol[1][u['MCP_PIP']].rewrite(atan).subs(\n  [(j, i) for i, j in
subs_sin_AA]).subs(\n    [(j, i) for i, j in subs_cos_mcp_dip_pip])\nsubs_solve[
'mcp_pip'].append(th_sol[1][u['MCP_PIP']])\ndisplay(sym.Eq(u['MCP_PIP'],
th_sol[0][u['MCP_PIP']]))\ndisplay(sym.Eq(u['MCP_PIP'],
th_sol[1][u['MCP_PIP']]))\n# sym.print_latex(sym.Eq(u['MCP_PIP'],
th_sol[1][u['MCP_PIP']]))\n\nfree_symbols = set()\nprint('Free symbols')\nfor eq
in subs_solve['mcp_pip']:\n  free_symbols = free_symbols |
eq.free_symbols\nprint(free_symbols)\n"
```

Solving u_CMC with substitutions - from fourth column, first and second row

[ ]:
```
"""
eq5 = sym.Eq(p['y'], sym.simplify(T_0_6_reduced[1, 3].subs(
  [(u['MCP_DIP_PIP'], subs_solve['mcp_dip_pip'][0])]
)).subs(subs_simpl))
eq5 = sym.simplify(eq5)
eq6 = sym.Eq(p['x'], sym.simplify(T_0_6_reduced[0, 3].subs(
  [(u['MCP_DIP_PIP'], subs_solve['mcp_dip_pip'][0])]
)).subs(subs_simpl))
eq6 = sym.simplify(eq6)

print('Equations')
display(eq5)
# sym.print_latex(eq5.subs(subs_simpl))
display(eq6)
# sym.print_latex(eq6.subs(subs_simpl))

# First solution
subs_solve['cmc'] = []
th_sol = sym.solve(
  eq5,
  u['CMC'],
  domain=S.Reals,
  dict=True, minimal=False, simplify=True, implicit=False
  )
# Keep only solution in domain [-pi/2, pi/2]
print('Solutions')
th_sol[0][u['CMC']] = th_sol[0][u['CMC']]
th_sol[1][u['CMC']] = th_sol[1][u['CMC']]
display(sym.Eq(u['CMC'], th_sol[0][u['CMC']]))
display(sym.Eq(u['CMC'], th_sol[1][u['CMC']]))
```

```
subs_solve['cmc'].extend([th_sol[0][u['CMC']], th_sol[1][u['CMC']]])
# sym.print_latex(sym.Eq(u['CMC'], th_sol[1][u['CMC']]))

# Second solution
th_sol = sym.solve(
  eq6,
  u['CMC'],
  domain=S.Reals,
  dict=True, minimal=False, simplify=True, implicit=False
  )
# Keep only solution in domain [-pi/2, pi/2]
j = th_sol[1][u['CMC']]
# acos returns between [-pi, pi]
j = sign_acos(j)
subs_solve['cmc'].extend(j)
for jj in j:
  display(sym.Eq(u['CMC'], jj))
  # sym.print_latex(sym.Eq(th_f['MCP_AA'], j))

free_symbols = set()
print('Free symbols')
for eq in subs_solve['cmc']:
  free_symbols = free_symbols | eq.free_symbols
print(free_symbols)
"""
```

```
[ ]: "\neq5 = sym.Eq(p['y'], sym.simplify(T_0_6_reduced[1, 3].subs(\n
     [(u['MCP_DIP_PIP'], subs_solve['mcp_dip_pip'][0])]\n)).subs(subs_simpl))\neq5 =
     sym.simplify(eq5)\neq6 = sym.Eq(p['x'], sym.simplify(T_0_6_reduced[0, 3].subs(\n
     [(u['MCP_DIP_PIP'], subs_solve['mcp_dip_pip'][0])]\n)).subs(subs_simpl))\neq6 =
     sym.simplify(eq6)\n\nprint('Equations')\ndisplay(eq5)\n#
     sym.print_latex(eq5.subs(subs_simpl))\ndisplay(eq6)\n#
     sym.print_latex(eq6.subs(subs_simpl))\n\n# First solution\nsubs_solve['cmc'] =
     []\nth_sol = sym.solve(\n  eq5,\n  u['CMC'],\n  domain=S.Reals,\n  dict=True,
     minimal=False, simplify=True, implicit=False\n  )\n# Keep only solution in
     domain [-pi/2, pi/2]\nprint('Solutions')\nth_sol[0][u['CMC']] =
     th_sol[0][u['CMC']]\nth_sol[1][u['CMC']] =
     th_sol[1][u['CMC']]\ndisplay(sym.Eq(u['CMC'],
     th_sol[0][u['CMC']]))\ndisplay(sym.Eq(u['CMC'],
     th_sol[1][u['CMC']]))\nsubs_solve['cmc'].extend([th_sol[0][u['CMC']],
     th_sol[1][u['CMC']]])\n# sym.print_latex(sym.Eq(u['CMC'],
     th_sol[1][u['CMC']]))\n\n# Second solution\nth_sol = sym.solve(\n  eq6,\n
     u['CMC'],\n  domain=S.Reals,\n  dict=True, minimal=False, simplify=True,
     implicit=False\n  )\n# Keep only solution in domain [-pi/2, pi/2]\nj =
     th_sol[1][u['CMC']]\n# acos returns between [-pi, pi]\nj =
     sign_acos(j)\nsubs_solve['cmc'].extend(j)\nfor jj in j:\n
     display(sym.Eq(u['CMC'], jj))\n  # sym.print_latex(sym.Eq(th_f['MCP_AA'],
     j))\n\nfree_symbols = set()\nprint('Free symbols')\nfor eq in
     subs_solve['cmc']:\n  free_symbols = free_symbols |
     eq.free_symbols\nprint(free_symbols)\n"
```

Solving finger angles with known substitutions

```
[ ]: # generating list of equations for solving
     eqan = list()

     for i, j in subs_reduce_angles:
       ii = (i - j)
       eqan.append(ii)
       display(sym.Eq(eqan[-1], 0))
```

$$K_{PIP}\theta_{MCP,FE} + \beta_3 - u_{MCP,PIP} = 0$$

$$K_{DIP}\theta_{MCP,FE} + \beta_4 - u_{MCP,DIP,PIP} + u_{MCP,PIP} = 0$$

$$\beta_1 - \theta_{CMC,FE} - u_{CMC} = 0$$

$$\beta_2 + \theta_{MCP,FE} - u_{CMC} - u_{CMC,MCP} = 0$$

Solving th_mcp_fe

```
[ ]: th_final[th_f['MCP_FE']] = list()
     print('Equations:')
     for eq in [eqan[1]+eqan[0]]: #[eqan[0], eqan[1], eqan[1]+eqan[0], eqan[3]]:
       th_sol = sym.solve(
         eq,
         th_f['MCP_FE'],
         domain=S.Reals,
         dict=False, minimal=False, simplify=True, implicit=False
         )
       display(sym.Eq(eq, 0))
       i = th_sol[0]
       th_final[th_f['MCP_FE']].append(i)
       # if j not in th_final[th_f['MCP_FE']]:
       #   # acos returns between [-pi, pi]
       #   j = sign_acos(j)
       #   th_final[th_f['MCP_FE']].extend(j)

     # sym.simplify(th_sol)
     free_symbols = set()
     print('Solutions:')
     for key, value in th_final.items():
       for i in value:
         if key == th_f['MCP_FE']:
           display(sym.Eq(key, i))
           free_symbols = free_symbols | i.free_symbols
```

```
    # sym.print_latex(sym.Eq(key, i))

print('Free symbols:')
print(free_symbols)
```

```
Equations:
```

$$K_{DIP}\theta_{MCP,FE} + K_{PIP}\theta_{MCP,FE} + \beta_3 + \beta_4 - u_{MCP,DIP,PIP} = 0$$

```
Solutions:
```

$$\theta_{MCP,FE} = \frac{-\beta_3 - \beta_4 + u_{MCP,DIP,PIP}}{K_{DIP} + K_{PIP}}$$

```
Free symbols:
{K_PIP, u_MCP,DIP,PIP, beta_4, beta_3, K_DIP}
```

Solving th_cmc_fe

[ ]:
```
th_final[th_f['CMC']] = list()
print('Equations:')
for eq in [eqan[3] - eqan[2]]: #[eqan[2], eqan[3] - eqan[2]]:
  th_sol = sym.solve(
    eq,
    th_f['CMC'],
    domain=S.Reals,
    dict=False, minimal=False, simplify=True, implicit=False
    )
  display(sym.Eq(eq, 0))
  i = th_sol[0]
  # acos returns between [-pi, pi]
  # i = sign_acos(i)
  # j = sym.simplify(th_sol[0].subs(subs_solve2).subs(subs_simpl).
↪subs(subs_simpl_after)).subs(subs_simpl)
  th_final[th_f['CMC']].append(i)
  # if j not in th_final[th_f['CMC']]:
  #   # acos returns between [-pi, pi]
  #   j = sign_acos(j)
  #   th_final[th_f['CMC']].extend(j)

# Print with code
free_symbols = set()
print('Solutions:')
for key, value in th_final.items():
  if key == th_f['CMC']:
    for i in value:
      display(sym.Eq(key, i))
      free_symbols = free_symbols | i.free_symbols
      # sym.print_latex(sym.Eq(key, i))
```

```
print('Free symbols:')
print(free_symbols)
```

```
Equations:
```

$$-\beta_1 + \beta_2 + \theta_{CMC,FE} + \theta_{MCP,FE} - u_{CMC,MCP} = 0$$

```
Solutions:
```

$$\theta_{CMC,FE} = \beta_1 - \beta_2 - \theta_{MCP,FE} + u_{CMC,MCP}$$

```
Free symbols:
{beta_2, theta_MCP,FE, beta_1, u_CMC,MCP}
```

```
[ ]: # Remapping sympy expressions keys in dictionary
     newnames = {
       th_f['MCP_AA']: 'mcp_aa', th_f['MCP_FE']: 'mcp_fe', th_f['CMC']: 'cmc'
     }
     th_sol_remapped = {newnames[key]: value for key, value in th_final.items()}
     th_sol_remapped
```

```
[ ]: {'mcp_aa': [atan(sqrt(1 - r_33**2)/r_33), -atan(sqrt(1 - r_33**2)/r_33)],
      'mcp_fe': [(-beta_3 - beta_4 + u_MCP,DIP,PIP)/(K_DIP + K_PIP)],
      'cmc': [beta_1 - beta_2 - theta_MCP,FE + u_CMC,MCP]}
```

```
[ ]: # Lambdifying expressions for use with numpy arrays
     grasp_parameters_s = [K['PIP'], K['DIP']]
     finger_lengths_s = [L_s[1], L_s[2], L_s[3], L_s[4]]
     finger_inclinations_s = [beta_s[1], beta_s[2], beta_s[3], beta_s[4]]
     position_vector_s = [p['x'], p['y'], p['z']]
     rotation_matrix_s = [
       r[11], r[12], r[13], r[21], r[22], r[23], r[31], r[32], r[33]
       ]

     subs_solve['mcp_dip_pip'] = sym.lambdify(
       [rotation_matrix_s, th_f['MCP_AA']], subs_solve['mcp_dip_pip'],␣
      ↪modules='numpy'
     )

     subs_solve['cmc_mcp'] = sym.lambdify(
       [rotation_matrix_s, th_f['MCP_AA']], subs_solve['cmc_mcp'], modules='numpy'
     )

     # subs_solve['mcp_pip'] = sym.lambdify(
     #   [finger_lengths_s, position_vector_s, rotation_matrix_s, th_f['MCP_AA'],
     #    u['MCP_DIP_PIP']],
     #   subs_solve['mcp_pip'], modules='numpy'
     # )
```

```python
# subs_solve['cmc'] = sym.lambdify(
#   [finger_lengths_s, position_vector_s, rotation_matrix_s, u['MCP_PIP'],
#    th_f['MCP_AA'], u['CMC_MCP']],
#   subs_solve['cmc'], modules='numpy'
# )
print(subs_solve)

th_sol_remapped['mcp_aa'] = sym.lambdify(
  [rotation_matrix_s], th_sol_remapped['mcp_aa'], modules='numpy'
  )
th_sol_remapped['mcp_fe'] = sym.lambdify(
  [grasp_parameters_s, finger_inclinations_s,
   u['MCP_DIP_PIP']],
  th_sol_remapped['mcp_fe'],
  modules='numpy'
  )
th_sol_remapped['cmc'] = sym.lambdify(
  [finger_inclinations_s, u['CMC_MCP'], th_f['MCP_FE']],
  th_sol_remapped['cmc'], modules='numpy'
  )
print(th_sol_remapped)

# Lambdify position vector
pos_vec_np = sym.lambdify(
  [grasp_parameters_s, finger_lengths_s, finger_inclinations_s,
   th_f['MCP_AA'], th_f['MCP_FE'], th_f['CMC']],
  T_0_6_circ[0:3, 3],
  modules='numpy'
  )
print(pos_vec_np)

# Lambdify jacobian of position vector for faster least_squares calculation
# Matrix m x n, (m - function dimension, n - number of angles)
pos_vec_jac_np = sym.lambdify(
  [grasp_parameters_s, finger_lengths_s, finger_inclinations_s,
   th_f['MCP_AA'], th_f['MCP_FE'], th_f['CMC']],
  # Partial derivatives per each angle and stack in matrix
  sym.Matrix.hstack(
    sym.diff(T_0_6_circ[0:3, 3], th_f['MCP_AA']),
    sym.diff(T_0_6_circ[0:3, 3], th_f['MCP_FE']),
    sym.diff(T_0_6_circ[0:3, 3], th_f['CMC'])
    ),
  modules='numpy'
  )
print(pos_vec_jac_np)
```

```
{'mcp_dip_pip': <function _lambdifygenerated at 0x7f348ee6f670>, 'cmc_mcp':
```

```
<function _lambdifygenerated at 0x7f348ee6f8b0>}
{'mcp_aa': <function _lambdifygenerated at 0x7f348ee6f5e0>, 'mcp_fe': <function
_lambdifygenerated at 0x7f348ee47940>, 'cmc': <function _lambdifygenerated at
0x7f3490e819d0>}
<function _lambdifygenerated at 0x7f3490e81280>
<function _lambdifygenerated at 0x7f348ee6a4c0>
```

0.1 Inverse kinematics verification

[ ]:
```
#TODO: r_33 must be slightly lower than 1, else make r_33 = 1 - eps and r_31␣
↪and r_13 = sqrt(2*eps-eps**2)
import numpy as np
import itertools
class FingerIKSolve(object):
  def __init__(
    self, equations_dictionary, substitutions_exprs, rtol=1e-1, atol=1e-6,
    round_dec=6
    ):
    """Return solutions to Index, Middle, Ring and Little finger Inverse␣
↪Kinematics

    Args:
        equations_dictionary (dict): dictionary of list of IK equations for
        theta_MCP,AA, theta_MCP,FE and theta_CMC,FE
        substitutions_exprs (dict): dictionary of list of substitution␣
↪expressions
        for mcp_dip_pip, cmc_mcp, mcp_pip, cmc
        rtol (float): relative tolerance for validating solutions
        atol (float): absolute tolerance for validating solutions
        round_dec (int): number of decimals for selecting unique float values,
        no actual rounding is performed
    """
    self._rtol = rtol
    self._atol = atol
    self._eqs = equations_dictionary
    self._subs = substitutions_exprs
    self._round = round_dec
    # Default constraints for I, M, R, L fingers
    self._cons_mcp_aa = np.radians([-15, 15])
    self._cons_mcp_fe = np.radians([-10, 90])
    # Default constraints for I finger
    self._cons_cmc = np.radians([0, 5])
    # Circular graspign default K_PIP and K_DIP
    self._grasp_parameters = np.array([0.75, 0.5])

  def add_joint_constraints(self, mcp_aa, mcp_fe, cmc):
    """Three joint constraints
    Args:
        mcp_aa (list): [min, max]
        mcp_fe (list): [min, max]
        cmc (list): [min, max]
    """
    if all((len(mcp_aa) == 2, len(mcp_fe) == 2, len(cmc) == 2)):
      self._cons_mcp_aa = np.radians(mcp_aa)
      self._cons_mcp_fe = np.radians(mcp_fe)
      self._cons_cmc = np.radians(cmc)
    else:
      raise Exception('Too many or too few joint constraints defined')

  def add_grasp_parameters(self, k_pip, k_dip):
    """K_PIP and K_DIP joint dependency parameters

    Args:
        k_pip (float)): linear coupling between DIP to MCP FE joint
        k_dip (float): linear coupling between DIP to MCP FE joint
    """
    self._grasp_parameters = [k_pip, k_dip]

  def add_finger_params(self, finger_lengths, finger_inclinations):
    """Define 4 length and 4 inclination parameters

    Args:
        finger_lengths (list): list containing lengths L1, L2, L3, L4
        finger_inclinations (list): list containing inclinations beta1, beta2,
        beta3, beta4 in degrees
    """
    if all((len(finger_lengths) == 4, len(finger_inclinations) == 4)):
      self._finger_lengths = finger_lengths
      self._finger_inclinations = np.radians(finger_inclinations)
      # Precalculate dh a values:
      self.dh_a_values()
    else:
      raise Exception('Too many or too few finger parameters defined')

  def add_transformation_matrix_expression(self, trans_expression):
    """Lambdified sympy expression for calculating forward kinematics

    Args:
        trans_expression (function): lambdified sympy expression with a and␣
↪theta
        parameters for forward kinematics
    """
    self._fk = trans_expression
```

```
 def add_position_vector_expression(self, position_vector_expression):
   """Lambdified sympy expression for calculating position vector

   Args:
       position_vector_expression (function): lambdified sympy vector with␣
↪grasp parameters,
       finger lengths, finger inclinations and three finger angles
   """
   self._pos_vec = position_vector_expression

 def add_position_vector_jacobian_expression(self, position_vector_jac_expr):
   """Lambdified sympy expression for calculating jacobian of position vector.

   Args:
       position_vector_jac_expr (function): lambdified sympy 3 x 3 matrix of␣
↪partial
       derivatives (p_x, p_y, p_z per th_mcp_aa, th_mcp_fe, th_cmc_fe),
       inputs are grasp parameters, finger lengths, finger inclinations and␣
↪three finger angles
   """
   self._pos_vec_jac = position_vector_jac_expr

 def dh_th_values(self, th_mcp_aa, th_mcp_fe, th_cmc_fe):
   """Return modified Denavit-Hartenberg parameter theta

   Args:
       th_mcp_aa (float): angle in radians
       th_mcp_fe (float): angle in radians
       th_cmc_fe (float): angle in radians
   """
   return np.fromiter([
   0,
   th_cmc_fe - self._finger_inclinations[0], # th_1
   th_mcp_fe + self._finger_inclinations[1], # th_2
   th_mcp_aa, # th_3
   self._grasp_parameters[0] * th_mcp_fe + self._finger_inclinations[2], # th_4
   self._grasp_parameters[1] * th_mcp_fe + self._finger_inclinations[3], # th_5
   0 # th_6
   ], np.float64)

 def dh_a_values(self):
   """Return modified Denavit-Hartenberg parameter a
   """
   self._dh_a_values = np.array([
     0, #a_0
     self._finger_lengths[0], # a_1
```

```
      0, # a_2
      self._finger_lengths[1], # a_3
      self._finger_lengths[2], # a_4
      self._finger_lengths[3], # a_5
      0, # a_6
    ])

  def calculate_FK(self, th_mcp_aa, th_mcp_fe, th_cmc_fe):
    """Return numpy array with transformation matrix values
    """
    return self._fk(
        self._dh_a_values, self.dh_th_values(th_mcp_aa, th_mcp_fe, th_cmc_fe)
        )

  def calculate_pos_vector(self, th_mcp_aa, th_mcp_fe, th_cmc_fe):
    """Return position vector using only three finger angles and finger and␣
↪grasp
    parameters

    Args:
        th_mcp_aa (float)
        th_mcp_fe (float)
        th_cmc_fe (float)
    """
    return np.fromiter(
      self._pos_vec(
        self._grasp_parameters, self._finger_lengths, self._finger_inclinations,
        th_mcp_aa, th_mcp_fe, th_cmc_fe
        ),
      np.float64)

  def calculate_pos_vector_jac(self, th_mcp_aa, th_mcp_fe, th_cmc_fe):
    """Return position vector jacobian using only three finger angles and finger
    and grasp parameters

    Args:
        th_mcp_aa (float)
        th_mcp_fe (float)
        th_cmc_fe (float)
    """
    return self._pos_vec_jac(
      self._grasp_parameters, self._finger_lengths, self._finger_inclinations,
      th_mcp_aa, th_mcp_fe, th_cmc_fe
    )

  def calculate_IK(self, transformation_matrix):
    """return IK solution (three angles defining finger pose)
    Args:
        transformation_matrix (np.array): 4 x 4 transformation matrix defining␣
↪TIP position
    """
    # Segmenting transformation matrix
    self._trans_matrix = transformation_matrix
    position_vector = transformation_matrix[0:3, 3]
    # print(f'Position vector: {position_vector}')
    rotation_matrix = [
      transformation_matrix[i[0]][i[1]] for i in (
        [0, 0], [0, 1], [0, 2], [1, 0], [1, 1], [1, 2], [2, 0], [2, 1], [2, 2]
        )
      ]
    # Tracking number of evaluations
    self.num_eval = 0
    # print(f'Rotations: {rotation_matrix}')
    # print(f'Grasp parameters: {self._grasp_parameters}')
    # print(f'Finger lengths: {self._finger_lengths}')
    # print(f'Finger inclinations: {self._finger_inclinations}')
    # Calculate both theta_mcp_aa values
    mcp_aa = np.fromiter(self._eqs['mcp_aa'](rotation_matrix), np.float64)
    self.num_eval += mcp_aa.size
    # Select unique values if rounded on self._round decimals (no actual␣
↪rounding)
    mcp_aa = mcp_aa[np.unique(np.around(mcp_aa, decimals=self._round),␣
↪return_index=True)[1]]
    # Drop nan values
    mcp_aa = mcp_aa[~np.isnan(mcp_aa)]
    # Check joint limits conditions and keep valid solutions
    mcp_aa = mcp_aa[(self._cons_mcp_aa[0] <= mcp_aa + self._atol) * (mcp_aa -␣
↪self._atol <= self._cons_mcp_aa[1])]
    # If no solution exists
    if not mcp_aa.size: return []
    # If mcp_aa equals zero, division by zero is returned, add atol
    if np.allclose(mcp_aa, np.zeros_like(mcp_aa), rtol = 0, atol = self.
↪_atol**2):
      mcp_aa = np.fromiter([self._atol**2], np.float64)
    # print(f'MCP_AA: {np.degrees(mcp_aa)}')
    # print(f'MCP_AA: {mcp_aa}')

    # Calculate all substitution expressions
    mcp_dip_pip = []
    for mcp_aa_i in mcp_aa:
      mcp_dip_pip_i = self._subs['mcp_dip_pip'](rotation_matrix, mcp_aa_i)
      mcp_dip_pip.extend(mcp_dip_pip_i)
    # Convert to numpy array
    mcp_dip_pip = np.fromiter(mcp_dip_pip, np.float64)
    self.num_eval += mcp_dip_pip.size
    # Select unique values if rounded on self._round decimals (no actual␣
↪rounding)
    mcp_dip_pip = mcp_dip_pip[np.unique(np.around(mcp_dip_pip, decimals=self.
↪_round), return_index=True)[1]]
    # Drop nan values
    mcp_dip_pip  = mcp_dip_pip[~np.isnan(mcp_dip_pip)]
    # If no solution exists
    if not mcp_dip_pip.size: return []

    cmc_mcp = []
    for mcp_aa_i in mcp_aa:
      cmc_mcp_i = self._subs['cmc_mcp'](rotation_matrix, mcp_aa_i)
      cmc_mcp.extend(cmc_mcp_i)
    # Convert to numpy array
    cmc_mcp = np.fromiter(cmc_mcp, np.float64)
    self.num_eval += cmc_mcp.size
    # Select unique values if rounded on self._round decimals (no actual␣
↪rounding)
    cmc_mcp = cmc_mcp[np.unique(np.around(cmc_mcp, decimals=self._round),␣
↪return_index=True)[1]]
    # Drop nan values
    cmc_mcp = cmc_mcp[~np.isnan(cmc_mcp)]
    # If no solution exists
    if not cmc_mcp.size: return []

    # mcp_pip = np.fromiter(
    #   self._subs['mcp_pip'](self._finger_lengths, position_vector,␣
↪rotation_matrix),
    #   np.float64
    #   )
    # self.num_eval += mcp_pip.size
    # # Select unique values if rounded on self._round decimals (no actual␣
↪rounding)
    # mcp_pip = mcp_pip[np.unique(np.around(mcp_pip, decimals=self._round),␣
↪return_index=True)[1]]
    # # Drop nan values
    # mcp_pip  = mcp_pip[~np.isnan(mcp_pip)]

    # cmc = []
    # for mcp_aa_i, mcp_pip_i, cmc_mcp_i in itertools.product(mcp_aa, mcp_pip,␣
↪cmc_mcp):
    #   cmc_i = self._subs['cmc'](
    #     self._finger_lengths, position_vector, rotation_matrix, mcp_pip_i,
    #     mcp_aa_i, cmc_mcp_i
    #     )
    #   cmc.extend(cmc_i)
    # # Convert to numpy array
    # cmc = np.fromiter(cmc, np.float64)
    # self.num_eval += cmc.size
    # # Select unique values if rounded on self._round decimals (no actual␣
↪rounding)
    # cmc = cmc[np.unique(np.around(cmc, decimals=self._round),␣
↪return_index=True)[1]]
    # # Drop nan values
    # cmc  = cmc[~np.isnan(cmc)]

    # Calculate all theta_mcp_fe values
    mcp_fe = []
    for mcp_dip_pip_i in mcp_dip_pip:
      mcp_fe_i = self._eqs['mcp_fe'](
        self._grasp_parameters, self._finger_inclinations, mcp_dip_pip_i
        )
      mcp_fe.extend(mcp_fe_i)
    # Convert to numpy array
    mcp_fe = np.fromiter(mcp_fe, np.float64)
    self.num_eval += mcp_fe.size
    # Select unique values if rounded on self._round decimals (no actual␣
↪rounding)
    mcp_fe = mcp_fe[np.unique(np.around(mcp_fe, decimals=self._round),␣
↪return_index=True)[1]]
    # Drop nan values
    mcp_fe = mcp_fe[~np.isnan(mcp_fe)]
    # print(f'MCP_FE: {np.degrees(mcp_fe)}')

    # Calculate all theta_cmc values
    cmc_fe = []
    for cmc_mcp_i, mcp_fe_i in itertools.product(cmc_mcp, mcp_fe):
      cmc_fe_i = self._eqs['cmc'](
        self._finger_inclinations, cmc_mcp_i, mcp_fe_i
        )
      cmc_fe.extend(cmc_fe_i)
    # Convert to numpy array
    cmc_fe = np.fromiter(cmc_fe, np.float64)
    self.num_eval += cmc_fe.size
    # Select unique values if rounded on self._round decimals (no actual␣
↪rounding)
    cmc_fe = cmc_fe[np.unique(np.around(cmc_fe, decimals=self._round),␣
↪return_index=True)[1]]
    # Drop nan values
    cmc_fe = cmc_fe[~np.isnan(cmc_fe)]
    # Check joint limits conditions and keep valid solutions
    mcp_fe = mcp_fe[(self._cons_mcp_fe[0] <= mcp_fe + self._atol) * (mcp_fe -␣
↪self._atol <= self._cons_mcp_fe[1])]
    # If no solution exists
    if not mcp_fe.size: return []
    # Check joint limits conditions and keep valid solutions
    # print(f'CMC_FE: {np.degrees(cmc_fe)}')
    cmc_fe = cmc_fe[(self._cons_cmc[0] <= cmc_fe + self._atol) * (cmc_fe - self.
↪_atol <= self._cons_cmc[1])]
    # print(f'CMC_FE: {np.degrees(cmc_fe)}')
    # If no solution exists
    if not cmc_fe.size: return []
    # Some solutions exist - validate Cartesian product and return valid␣
↪solutions
    return self.validate_IK_solutions(itertools.product(mcp_aa, mcp_fe, cmc_fe))

  def validate_IK_solutions(self, ik_solutions):
    solutions = []
    for mcp_aa, mcp_fe, cmc_fe in ik_solutions:
      validation_matrix = self.calculate_FK(mcp_aa, mcp_fe, cmc_fe)
      # print(validation_matrix)
      if np.allclose(
        validation_matrix, self._trans_matrix, rtol = self._rtol, atol = self.
↪_atol
        ):
        solutions.append([mcp_aa, mcp_fe, cmc_fe])
        # print('Transformation matrix after IK')
        # print(validation_matrix)
        # print('IK solution')
        # print(f'MCP_AA: {np.degrees(mcp_aa)}, MCP_FE: {np.degrees(mcp_fe)}, '
        #       f'CMC_FE: {np.degrees(cmc_fe)}')
    return np.degrees(solutions)
```

Set up analytical solver - one example + timing

```
[ ]: solver = FingerIKSolve(th_sol_remapped, subs_solve, rtol=1e-2, round_dec=6)
     solver.add_joint_constraints(mcp_aa=[-15, 15], mcp_fe=[-10, 90], cmc=[0, 5])
     solver.add_grasp_parameters(k_pip=0.75, k_dip=0.5)
     solver.add_finger_params(
       finger_lengths=[6.34, 4.26, 2.51, 1.80], finger_inclinations=[2, 0, 7, -3]
       )
     solver.add_transformation_matrix_expression(T_0_6_full_np)
     solver.add_position_vector_expression(pos_vec_np)
     solver.add_position_vector_jacobian_expression(pos_vec_jac_np)
     # Works for th_mcp_AA as small as 1e-5
```

```
initial_matrix = solver.calculate_FK(np.radians(12.0001), np.radians(30.
↪0003250), np.radians(3.000100))
print('Initial transformation matrix')
print(initial_matrix)
print('-------')
tic = time.perf_counter()
n_iter = 1000
for i in range(n_iter):
  j = solver.calculate_IK(initial_matrix)
print(f'Time elapsed for 1 solution: {(time.perf_counter() - tic)/n_iter}')
print(j)
```

```
Initial transformation matrix
[[ 0.28666307 -0.94130942 -0.17821577 11.6217857 ]
 [ 0.94529176  0.30814991 -0.10708464  6.11834624]
 [ 0.15571695 -0.13776869  0.97814724  1.62020661]
 [ 0.          0.          0.          1.        ]]
-------
Time elapsed for 1 solution: 0.0008628023229998689
[[12.0001   30.000325  3.0001   ]]
```

Non-linear least-squares optimization method setup - residual function + timing

[ ]:
```
from scipy.optimize import least_squares

def residual_function(theta_values, initial_matrix, solver):
  """Return flatten residuals by subtracting transformation matrices

  Args:
      theta_values (numpy array): th_mcp_aa, th_mcp_fe, th_cmc_fe in radians
  """
  result_fk = solver.calculate_FK(*theta_values)
  # Actual - temporary value
  return (initial_matrix - result_fk).flatten()

tic = time.perf_counter()
n_iter = 100
for i in range(n_iter):
  ls_sol = least_squares(
    fun=residual_function,
    x0=np.radians([0, 0, 0]), # initial guess
    bounds=([solver._cons_mcp_aa[0], solver._cons_mcp_fe[0], solver.
↪_cons_cmc[0]],
              [solver._cons_mcp_aa[1], solver._cons_mcp_fe[1], solver.
↪_cons_cmc[1]]),
    xtol=1e-5, ftol=None, gtol=None,
    kwargs={'initial_matrix': initial_matrix,
            'solver': solver}
    )
print(f'Time elapsed for 1 solution: {(time.perf_counter() - tic)/n_iter}')
print(np.degrees(ls_sol.x))
print(f'Function evaluations: {ls_sol.nfev}')
print(f'Function evaluations for jacobian: {ls_sol.njev}')
```

```
Time elapsed for 1 solution: 0.02862790446999952
[12.0001   30.000325  3.0001  ]
Function evaluations: 38
Function evaluations for jacobian: 38
```

Non-linear least-squares optimization method setup - only position vector -> residual function + analytical jacobian + timing

[ ]:
```
def residual_function_pos_vec(theta_values, initial_pos, solver):
  """Return residuals by subtracting initial and resulting position vector

  Args:
      theta_values (1D array): th_mcp_aa, th_mcp_fe, th_cmc_fe in radians
      initial_pos (1D array): targeted positon vector
      solver (object): solver object
  """
  return initial_pos - solver.calculate_pos_vector(*theta_values)

def residual_function_pos_vec_jac(theta_values, initial_pos, solver):
  """Return negative value of position vector Jacobian. Jacobian of residuals is
  negative value of partial derivation of position vector per each of three␣
 ↪finger angles
  (initial_pos_vector is constant).
  Args:
      theta_values (1D array): th_mcp_aa, th_mcp_fe, th_cmc_fe in radians
      initial_pos: not used (for compatibility)
      solver (object): solver object
  """
  return -solver.calculate_pos_vector_jac(*theta_values)

tic = time.perf_counter()
n_iter = 100
for i in range(n_iter):
  ls_sol = least_squares(
    fun=residual_function_pos_vec,
    jac=residual_function_pos_vec_jac, # analytical Jacobian - small speedup
    x0=np.radians([0, 0, 0]), # initial guess
    bounds=([solver._cons_mcp_aa[0], solver._cons_mcp_fe[0], solver.
 ↪_cons_cmc[0]],
               [solver._cons_mcp_aa[1], solver._cons_mcp_fe[1], solver.
 ↪_cons_cmc[1]]),
    xtol=1e-5, ftol=None, gtol=None,
    kwargs={'initial_pos': initial_matrix[0:3, 3],
            'solver': solver}
    )
print(f'Time elapsed for 1 solution: {(time.perf_counter() - tic)/n_iter}')
np.degrees(ls_sol.x)
print(np.degrees(ls_sol.x))
print(f'Function evaluations: {ls_sol.nfev}')
print(f'Function evaluations for jacobian: {ls_sol.njev}')
```

```
Time elapsed for 1 solution: 0.018314731570001186
[12.0001   30.000325  3.0001   ]
Function evaluations: 38
Function evaluations for jacobian: 38
```

Result validation over a grid of values - quality of solutions where solutions exist

```
[ ]: # For least squares, position vector part of matrix (faster computation) - some␣
     ↪wrong solutions
     cart_product = 0
     solved_IK = 0
     wrong_IK = 0
     solved_ls = 0
     wrong_ls = 0
     fun_eval_ls = 0
     fun_eval = 0
     no_solution = dict()
     no_solution['count'] = 0
     no_solution['mcp_aa'] = []
     no_solution['mcp_fe'] = []
     no_solution['cmc_fe'] = []
     no_solution_ls = dict()
     no_solution_ls['count'] = 0
     no_solution_ls['mcp_aa'] = []
     no_solution_ls['mcp_fe'] = []
     no_solution_ls['cmc_fe'] = []
     for starting_ang in itertools.product(np.linspace(-14.326, 14.936, 30), np.
     ↪linspace(-9.254, 89.824, 30), np.linspace(0.125, 4.976, 30)):
       cart_product += 1
       initial_matrix = solver.calculate_FK(*np.radians(starting_ang))
       solution_ang = solver.calculate_IK(initial_matrix)
       # Count number of function evaluations for analytical solution
       fun_eval += solver.num_eval
       ls_sol = least_squares(
         # fun=residual_function,
         fun=residual_function_pos_vec,
         jac=residual_function_pos_vec_jac,
         x0=np.radians([0, 40, 3]), # initial guess
```

```
    bounds=([solver._cons_mcp_aa[0], solver._cons_mcp_fe[0], solver.
↪_cons_cmc[0]],
              [solver._cons_mcp_aa[1], solver._cons_mcp_fe[1], solver.
↪_cons_cmc[1]]),
    xtol=1e-5, ftol=None, gtol=None,
    kwargs={
      # 'initial_matrix': initial_matrix,
      'initial_pos': initial_matrix[0:3, 3],
      'solver': solver
      }
    )
  # Count number of function evaluations (x12 since entire transformation␣
↪matrix is evaluated)
  # fun_eval_ls += (ls_sol.nfev + ls_sol.njev) * 12
  # Count number of function evaluations (x3 since entire position vector is␣
↪evaluated)
  fun_eval_ls += (ls_sol.nfev + ls_sol.njev) * 3
  if ls_sol.success:
    if np.allclose(starting_ang, np.degrees(ls_sol.x), rtol=1e-2, atol=1e-6):
      solved_ls += 1
    else:
      wrong_ls += 1
      print(f'Wrong least_squares solution for {starting_ang} : {np.
↪degrees(ls_sol.x)}')
  else:
    no_solution_ls['count'] += 1
    no_solution_ls['mcp_aa'].append(starting_ang[0])
    no_solution_ls['mcp_fe'].append(starting_ang[1])
    no_solution_ls['cmc_fe'].append(starting_ang[2])
    print(f'No least_squares solution for {starting_ang}')
  # Check if solution empty
  if not solution_ang.size:
    no_solution['count'] += 1
    no_solution['mcp_aa'].append(starting_ang[0])
    no_solution['mcp_fe'].append(starting_ang[1])
    no_solution['cmc_fe'].append(starting_ang[2])
    print(f'No analytical solution for {starting_ang}')
  for sol in solution_ang:
    if np.allclose(starting_ang, sol, rtol=1e-2, atol=1e-6):
      solved_IK += 1
      break
    else:
      wrong_IK += 1
      print(f'Wrong analytical solution for {starting_ang} : {sol}')
print(f"Tried: {cart_product}")
print(f"Analytical approach - Solved: {solved_IK}, No solution:␣
↪{no_solution['count']}, Wrong solution: {wrong_IK}, Function evaluations:␣
↪{fun_eval}")
print(f"Least squares approach - Solved: {solved_ls}, No solution:␣
↪{no_solution_ls['count']} Wrong solution: {wrong_ls}, Function evaluations:␣
↪{fun_eval_ls}")
no_solution.pop('count')
no_solution_ls.pop('count')
no_solution = pd.DataFrame(no_solution)
no_solution_ls = pd.DataFrame(no_solution_ls)
print('Analytical approach - descriptive statistics for no solutions')
display(no_solution.describe())
print('Least squares approach - descriptive statistics for no solutions')
display(no_solution_ls.describe())
# tic = time.perf_counter()
# for i in range(100):
#   solver.calculate_IK(initial_matrix)
# print((time.perf_counter() - tic)/100)
```

```
Wrong least_squares solution for (-14.326, -5.8375172413793095,
3.1359655172413796) : [-1.43637459e+01 -2.14353014e+00  4.00586859e-14]
Wrong least_squares solution for (-14.326, -5.8375172413793095,
3.303241379310345) : [-1.43645980e+01 -1.94629312e+00  4.92888007e-14]
Wrong least_squares solution for (-14.326, -5.8375172413793095,
3.4705172413793104) : [-1.43653319e+01 -1.74906423e+00  2.91402953e-14]
Wrong least_squares solution for (-14.326, -5.8375172413793095,
3.637793103448276) : [-1.43659477e+01 -1.55184767e+00  1.19052805e-14]
Wrong least_squares solution for (-14.326, -5.8375172413793095,
3.8050689655172416) : [-1.43664453e+01 -1.35464757e+00  3.73374814e-15]
Wrong least_squares solution for (-14.326, -5.8375172413793095,
3.972344827586207) : [-1.43668255e+01 -1.15747852e+00  2.54418117e-07]
Wrong least_squares solution for (-14.326, -5.8375172413793095,
4.139620689655173) : [-1.43670865e+01 -9.60321063e-01  1.10115467e-07]
Wrong least_squares solution for (-14.326, -5.8375172413793095,
4.306896551724138) : [-1.43672299e+01 -7.63192873e-01  3.98002469e-08]
Wrong least_squares solution for (-14.326, -5.8375172413793095,
4.474172413793104) : [-1.43672554e+01 -5.66097942e-01  5.73398965e-09]
Wrong least_squares solution for (-14.326, -5.8375172413793095,
4.641448275862069) : [-1.43671633e+01 -3.69039968e-01  2.70456447e-13]
Wrong least_squares solution for (-14.326, -5.8375172413793095,
4.808724137931034) : [-1.43669534e+01 -1.72023159e-01  1.67840526e-13]
Wrong least_squares solution for (-14.326, -5.8375172413793095, 4.976) :
[-1.43666252e+01  2.49497994e-02  2.21335163e-12]
Wrong least_squares solution for (-14.326, -2.4210344827586203, 0.125) :
[-14.32427114  -2.71441331   0.37407901]
Wrong analytical solution for (-14.326, 24.9108275862069, 4.139620689655173) :
[-14.326       24.91082759   4.03872414]
```

```
Wrong analytical solution for (-14.326, 45.409724137931036, 1.6304827586206898)
: [-14.326       45.40972414   1.55006897]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
3.1359655172413796) : [-1.33521178e+01 -2.14512954e+00  4.95051708e-14]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
3.303241379310345) : [-1.33529126e+01 -1.94797769e+00  5.53873847e-14]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
3.4705172413793104) : [-1.33535975e+01 -1.75083389e+00  3.05832559e-14]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
3.637793103448276) : [-1.33541724e+01 -1.55370232e+00  1.17454947e-14]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
3.8050689655172416) : [-1.33546374e+01 -1.35658713e+00  3.50074096e-15]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
3.972344827586207) : [-1.33549927e+01 -1.15949239e+00  8.73598467e-16]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
4.139620689655173) : [-1.33552401e+01 -9.62429595e-01  1.07768233e-07]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
4.306896551724138) : [-1.33553752e+01 -7.65386005e-01  4.16722534e-08]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
4.474172413793104) : [-1.33554008e+01 -5.68375540e-01  9.95317583e-09]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
4.641448275862069) : [-1.33553168e+01 -3.71401973e-01  2.37418960e-13]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
4.808724137931034) : [-1.33551233e+01 -1.74469295e-01  1.48392283e-13]
Wrong least_squares solution for (-13.31696551724138, -5.8375172413793095,
4.976) : [-1.33548198e+01  2.24196762e-02  6.63296276e-13]
Wrong least_squares solution for (-13.31696551724138, -2.4210344827586203,
0.125) : [-13.31542486  -2.70247856   0.36404989]
Wrong analytical solution for (-13.31696551724138, 24.9108275862069,
4.139620689655173) : [-13.31696552  24.91082759   4.03872414]
Wrong analytical solution for (-13.31696551724138, 45.409724137931036,
1.6304827586206898) : [-13.31696552  45.40972414   1.55006897]
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
3.1359655172413796) : [-1.23404755e+01 -2.14660597e+00  5.93733921e-14]
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
3.303241379310345) : [-1.23412124e+01 -1.94953276e+00  6.31311092e-14]
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
3.4705172413793104) : [-1.23418476e+01 -1.75246750e+00  3.30186005e-14]
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
3.637793103448276) : [-1.23423811e+01 -1.55541439e+00  1.20453731e-14]
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
3.8050689655172416) : [-1.23428131e+01 -1.35837757e+00  3.43734415e-15]
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
3.972344827586207) : [-1.23431434e+01 -1.16136110e+00  8.31373863e-16]
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
4.139620689655173) : [-1.23433741e+01 -9.64376375e-01  1.03521987e-07]
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
4.306896551724138) : [-1.23435009e+01 -7.67410827e-01  4.01032728e-08]
```

```
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
4.474172413793104) : [-1.23435264e+01 -5.70478285e-01  9.72189740e-09]
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
4.641448275862069) : [-1.23434505e+01 -3.73582507e-01  2.31376372e-13]
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
4.808724137931034) : [-1.23432734e+01 -1.76727459e-01  1.46597320e-13]
Wrong least_squares solution for (-12.30793103448276, -5.8375172413793095,
4.976) : [-1.23429945e+01  2.00840623e-02  2.87941210e-14]
Wrong least_squares solution for (-12.30793103448276, -2.4210344827586203,
0.125) : [-12.30656379  -2.69144773   0.35477251]
Wrong analytical solution for (-12.30793103448276, 24.9108275862069,
4.139620689655173) : [-12.30793103  24.91082759   4.03872414]
Wrong analytical solution for (-12.30793103448276, 45.409724137931036,
1.6304827586206898) : [-12.30793103  45.40972414   1.55006897]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
3.1359655172413796) : [-1.13288200e+01 -2.14796085e+00  6.85162303e-14]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
3.303241379310345) : [-1.13294985e+01 -1.95095982e+00  7.04850735e-14]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
3.4705172413793104) : [-1.13300835e+01 -1.75396663e+00  3.53738034e-14]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
3.637793103448276) : [-1.13305751e+01 -1.55698553e+00  1.24164066e-14]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
3.8050689655172416) : [-1.13309735e+01 -1.36002062e+00  3.43279907e-15]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
3.972344827586207) : [-1.13312785e+01 -1.16307598e+00  8.12436707e-16]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
4.139620689655173) : [-1.13314919e+01 -9.66162954e-01  1.01208499e-07]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
4.306896551724138) : [-1.13316099e+01 -7.69269018e-01  3.93139267e-08]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
4.474172413793104) : [-1.13316349e+01 -5.72407981e-01  9.66971535e-09]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
4.641448275862069) : [-1.13315668e+01 -3.75583583e-01  2.29304615e-13]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
4.808724137931034) : [-1.13314056e+01 -1.78799770e-01  1.46858265e-13]
Wrong least_squares solution for (-11.298896551724138, -5.8375172413793095,
4.976) : [-1.13311514e+01  1.79394103e-02  9.12168715e-14]
Wrong least_squares solution for (-11.298896551724138, -2.4210344827586203,
0.125) : [-11.29768914  -2.68131333   0.34624244]
Wrong analytical solution for (-11.298896551724138, 24.9108275862069,
4.139620689655173) : [-11.29889655  24.91082759   4.03872414]
Wrong analytical solution for (-11.298896551724138, 45.409724137931036,
1.6304827586206898) : [-11.29889655  45.40972414   1.55006897]
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
3.1359655172413796) : [-1.03171524e+01 -2.14919511e+00  3.01937610e-09]
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
3.303241379310345) : [-1.03177720e+01 -1.95226025e+00  7.75371727e-14]
```

```
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
3.4705172413793104) : [-1.03183063e+01 -1.75533275e+00  3.76316940e-14]
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
3.637793103448276) : [-1.03187556e+01 -1.55841726e+00  1.28059350e-14]
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
3.8050689655172416) : [-1.03191199e+01 -1.36151789e+00  3.45320084e-15]
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
3.972344827586207) : [-1.03193991e+01 -1.16463871e+00  8.03792381e-16]
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
4.139620689655173) : [-1.03195933e+01 -9.67783740e-01  1.68689540e-16]
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
4.306896551724138) : [-1.03197037e+01 -7.70962358e-01  3.88776048e-08]
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
4.474172413793104) : [-1.03197277e+01 -5.74166479e-01  9.69482354e-09]
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
4.641448275862069) : [-1.03196669e+01 -3.77407126e-01  2.28740711e-13]
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
4.808724137931034) : [-1.03195213e+01 -1.80688227e-01  1.47795673e-13]
Wrong least_squares solution for (-10.289862068965519, -5.8375172413793095,
4.976) : [-1.03192910e+01  1.59861734e-02  9.31058039e-14]
Wrong least_squares solution for (-10.289862068965519, -2.4210344827586203,
0.125) : [-10.28880211  -2.6720685    0.33845558]
Wrong analytical solution for (-10.289862068965519, 24.9108275862069,
4.139620689655173) : [-10.28986207  24.91082759   4.03872414]
Wrong analytical solution for (-10.289862068965519, 28.327310344827588,
0.7941034482758621) : [-10.28986207  28.32731034   0.55127586]
Wrong analytical solution for (-10.289862068965519, 45.409724137931036,
1.6304827586206898) : [-10.28986207  45.40972414   1.55006897]
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
3.1359655172413796) : [-9.30547386e+00 -2.15031072e+00  1.79551519e-09]
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
3.303241379310345) : [-9.30603402e+00 -1.95343532e+00  8.42307082e-14]
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
3.4705172413793104) : [-9.30651727e+00 -1.75656717e+00  3.97755006e-14]
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
3.637793103448276) : [-9.30692376e+00 -1.55971097e+00  1.31981676e-14]
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
3.8050689655172416) : [-9.30725351e+00 -1.36287083e+00  3.48816697e-15]
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
3.972344827586207) : [-9.30750652e+00 -1.16605080e+00  8.01330217e-16]
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
4.139620689655173) : [-9.30768282e+00 -9.69254888e-01  1.67268882e-16]
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
4.306896551724138) : [-9.30778348e+00 -7.72492473e-01  3.86460120e-08]
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
4.474172413793104) : [-9.30780623e+00 -5.75755471e-01  9.75654616e-09]
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
4.641448275862069) : [-9.30775242e+00 -3.79054893e-01  2.28991907e-13]
```

```
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
4.808724137931034) : [-9.30762208e+00 -1.82394651e-01  1.49036262e-13]
Wrong least_squares solution for (-9.280827586206897, -5.8375172413793095,
4.976) : [-9.30741531e+00  1.42212135e-02  9.50005560e-14]
Wrong least_squares solution for (-9.280827586206897, -2.4210344827586203,
0.125) : [-9.27990386 -2.6637069   0.3314081 ]
Wrong analytical solution for (-9.280827586206897, 24.9108275862069,
4.139620689655173) : [-9.28082759 24.91082759  4.03872414]
Wrong analytical solution for (-9.280827586206897, 28.327310344827588,
0.7941034482758621) : [-9.28082759 28.32731034  0.55127586]
Wrong analytical solution for (-9.280827586206897, 45.409724137931036,
1.6304827586206898) : [-9.28082759 45.40972414  1.55006897]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
3.1359655172413796) : [-8.29378541e+00 -2.15130838e+00  7.54224838e-10]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
3.303241379310345) : [-8.29428575e+00 -1.95448616e+00  9.04813040e-14]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
3.4705172413793104) : [-8.29471749e+00 -1.75767108e+00  4.17874856e-14]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
3.637793103448276) : [-8.29508079e+00 -1.56086790e+00  1.35852018e-14]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
3.8050689655172416) : [-8.29537566e+00 -1.36408073e+00  3.53291720e-15]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
3.972344827586207) : [-8.29560211e+00 -1.16731359e+00  8.03162989e-16]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
4.139620689655173) : [-8.29576015e+00 -9.70570505e-01  1.66949281e-16]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
4.306896551724138) : [-8.29584980e+00 -7.73855465e-01  3.01064609e-17]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
4.474172413793104) : [-8.29587187e+00 -5.77176482e-01  9.83229063e-09]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
4.641448275862069) : [-8.29582472e+00 -3.80528465e-01  2.29797406e-13]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
4.808724137931034) : [-8.29570934e+00 -1.83920679e-01  1.50448121e-13]
Wrong least_squares solution for (-8.271793103448276, -5.8375172413793095,
4.976) : [-8.29552580e+00  1.26428363e-02  9.68424774e-14]
Wrong least_squares solution for (-8.271793103448276, -2.4210344827586203,
0.125) : [-8.27099554 -2.65622294  0.32509667]
Wrong analytical solution for (-8.271793103448276, 24.9108275862069,
4.139620689655173) : [-8.2717931  24.91082759  4.03872414]
Wrong analytical solution for (-8.271793103448276, 28.327310344827588,
0.7941034482758621) : [-8.2717931  28.32731034  0.55127586]
Wrong analytical solution for (-8.271793103448276, 45.409724137931036,
1.6304827586206898) : [-8.2717931  45.40972414  1.55006897]
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
3.1359655172413796) : [-7.28208812e+00 -2.15218904e+00  3.55775217e-11]
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
3.303241379310345) : [-7.28252826e+00 -1.95541377e+00  9.62194478e-14]
```

```
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
3.4705172413793104) : [-7.28290814e+00 -1.75864555e+00  4.36469886e-14]
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
3.637793103448276) : [-7.28322790e+00 -1.56188917e+00  1.39574715e-14]
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
3.8050689655172416) : [-7.28348756e+00 -1.36514875e+00  3.58289659e-15]
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
3.972344827586207) : [-7.28368711e+00 -1.16842831e+00  8.07728354e-16]
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
4.139620689655173) : [-7.28382658e+00 -9.71731853e-01  1.67334333e-16]
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
4.306896551724138) : [-7.28390598e+00 -7.75063369e-01  3.02353902e-17]
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
4.474172413793104) : [-7.28392535e+00 -5.78426858e-01  3.22939604e-13]
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
4.641448275862069) : [-7.28388526e+00 -3.81829252e-01  2.30934840e-13]
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
4.808724137931034) : [-7.28378456e+00 -1.85267772e-01  1.51918323e-13]
Wrong least_squares solution for (-7.262758620689656, -5.8375172413793095,
4.976) : [-7.28362398e+00  1.12495324e-02  9.85801087e-14]
Wrong least_squares solution for (-7.262758620689656, -2.4210344827586203,
0.125) : [-7.2620783  -2.64961155  0.31951823]
Wrong analytical solution for (-7.262758620689656, 24.9108275862069,
4.139620689655173) : [-7.26275862 24.91082759  4.03872414]
Wrong analytical solution for (-7.262758620689656, 28.327310344827588,
0.7941034482758621) : [-7.26275862 28.32731034  0.55127586]
Wrong analytical solution for (-7.262758620689656, 45.409724137931036,
1.6304827586206898) : [-7.26275862 45.40972414  1.55006897]
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
3.1359655172413796) : [-6.27038308e+00 -2.15295357e+00  1.86711453e-10]
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
3.303241379310345) : [-6.27076268e+00 -1.95621904e+00  1.01380857e-13]
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
3.4705172413793104) : [-6.27109039e+00 -1.75949150e+00  4.53321657e-14]
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
3.637793103448276) : [-6.27136631e+00 -1.56277575e+00  1.43057208e-14]
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
3.8050689655172416) : [-6.27159046e+00 -1.36607592e+00  3.63429818e-15]
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
3.972344827586207) : [-6.27176284e+00 -1.16939602e+00  8.13830231e-16]
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
4.139620689655173) : [-6.27188346e+00 -9.72740040e-01  1.68136097e-16]
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
4.306896551724138) : [-6.27195234e+00 -7.76111973e-01  3.04246520e-17]
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
4.474172413793104) : [-6.27196951e+00 -5.79515812e-01  2.12024833e-13]
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
4.641448275862069) : [-6.27193500e+00 -3.82955541e-01  5.11140882e-16]
```

```
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
4.808724137931034) : [-6.27184920e+00 -1.86437213e-01  1.53358109e-13]
Wrong least_squares solution for (-6.2537241379310355, -5.8375172413793095,
4.976) : [-6.27171137e+00  1.00399738e-02  1.00171999e-13]
Wrong least_squares solution for (-6.2537241379310355, -2.4210344827586203,
0.125) : [-6.25315324 -2.64386828  0.31467008]
Wrong analytical solution for (-6.2537241379310355, 24.9108275862069,
4.139620689655173) : [-6.25372414 24.91082759  4.03872414]
Wrong analytical solution for (-6.2537241379310355, 28.327310344827588,
0.7941034482758621) : [-6.25372414 28.32731034  0.55127586]
Wrong analytical solution for (-6.2537241379310355, 45.409724137931036,
1.6304827586206898) : [-6.25372414 45.40972414  1.55006897]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
3.1359655172413796) : [-5.25867136e+00 -2.15360267e+00  1.27264691e-10]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
3.303241379310345) : [-5.25899014e+00 -1.95690275e+00  1.05877910e-13]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
3.4705172413793104) : [-5.25926540e+00 -1.76020973e+00  4.68206136e-14]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
3.637793103448276) : [-5.25949721e+00 -1.56352849e+00  1.46242594e-14]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
3.8050689655172416) : [-5.25968560e+00 -1.36686312e+00  3.68524555e-15]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
3.972344827586207) : [-5.25983056e+00 -1.17021764e+00  8.20892689e-16]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
4.139620689655173) : [-5.25993210e+00 -9.73596027e-01  1.69212366e-16]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
4.306896551724138) : [-5.25999023e+00 -7.77002276e-01  3.06472496e-17]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
4.474172413793104) : [-5.26000498e+00 -5.80440373e-01  1.34640226e-13]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
4.641448275862069) : [-5.25997637e+00 -3.83914299e-01  5.14709100e-16]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
4.808724137931034) : [-5.25990475e+00 -1.87430113e-01  1.54733282e-13]
Wrong least_squares solution for (-5.244689655172415, -5.8375172413793095,
4.976) : [-5.25978947e+00  9.01301110e-03  1.01602597e-13]
Wrong least_squares solution for (-5.244689655172415, -2.4210344827586203,
0.125) : [-5.24422149 -2.63898927  0.31054989]
Wrong analytical solution for (-5.244689655172415, 24.9108275862069,
4.139620689655173) : [-5.24468966 24.91082759  4.03872414]
Wrong analytical solution for (-5.244689655172415, 28.327310344827588,
0.7941034482758621) : [-5.24468966 28.32731034  0.55127586]
Wrong analytical solution for (-5.244689655172415, 45.409724137931036,
1.6304827586206898) : [-5.24468966 45.40972414  1.55006897]
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
3.1359655172413796) : [-4.24695402e+00 -2.15413698e+00  4.04745122e-15]
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
3.303241379310345) : [-4.24721176e+00 -1.95746553e+00  1.09644828e-13]
```

```
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
3.4705172413793104) : [-4.24743434e+00 -1.76080095e+00  4.80909938e-14]
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
3.637793103448276) : [-4.24762184e+00 -1.56414810e+00  1.49062618e-14]
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
3.8050689655172416) : [-4.24777425e+00 -1.36751110e+00  3.73352065e-15]
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
3.972344827586207) : [-4.24789158e+00 -1.17089395e+00  8.28298943e-16]
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
4.139620689655173) : [-4.24797383e+00 -9.74300627e-01  1.70423590e-16]
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
4.306896551724138) : [-4.24802102e+00 -7.77735124e-01  3.08792193e-17]
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
4.474172413793104) : [-4.24803317e+00 -5.81201420e-01  8.87439553e-14]
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
4.641448275862069) : [-4.24801029e+00 -3.84703496e-01  5.18430015e-16]
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
4.808724137931034) : [-4.24795242e+00 -1.88245349e-01  2.53950577e-16]
Wrong least_squares solution for (-4.235655172413795, -5.8375172413793095,
4.976) : [-4.24785976e+00  8.16767088e-03  1.02851763e-13]
Wrong least_squares solution for (-4.235655172413795, -2.4210344827586203,
0.125) : [-4.23528414 -2.63497122  0.30715567]
Wrong analytical solution for (-4.235655172413795, 24.9108275862069,
4.139620689655173) : [-4.23565517 24.91082759  4.03872414]
Wrong analytical solution for (-4.235655172413795, 28.327310344827588,
0.7941034482758621) : [-4.23565517 28.32731034  0.55127586]
Wrong analytical solution for (-4.235655172413795, 45.409724137931036,
1.6304827586206898) : [-4.23565517 45.40972414  1.55006897]
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
3.1359655172413796) : [-3.23523214e+00 -2.15455699e+00  2.83423452e-15]
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
3.303241379310345) : [-3.23542865e+00 -1.95790793e+00  1.12639157e-13]
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
3.4705172413793104) : [-3.23559838e+00 -1.76126569e+00  4.91237537e-14]
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
3.637793103448276) : [-3.23574138e+00 -1.56463517e+00  1.51440505e-14]
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
3.8050689655172416) : [-3.23585764e+00 -1.36802047e+00  3.77661958e-15]
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
3.972344827586207) : [-3.23594718e+00 -1.17142559e+00  8.35391970e-16]
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
4.139620689655173) : [-3.23600999e+00 -9.74854507e-01  1.71630062e-16]
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
4.306896551724138) : [-3.23604608e+00 -7.78311209e-01  3.10985290e-17]
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
4.474172413793104) : [-3.23605547e+00 -5.81799674e-01  6.87862569e-14]
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
4.641448275862069) : [-3.23603818e+00 -3.85323878e-01  5.21997527e-16]
```

```
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
4.808724137931034) : [-3.23599422e+00 -1.88887817e-01  2.56151547e-16]
Wrong least_squares solution for (-3.2266206896551743, -5.8375172413793095,
4.976) : [-3.23592363e+00  7.50454159e-03  1.26464630e-16]
Wrong least_squares solution for (-3.2266206896551743, -2.4210344827586203,
0.125) : [-3.22634227 -2.63181146  0.30448577]
Wrong analytical solution for (-3.2266206896551743, 24.9108275862069,
4.139620689655173) : [-3.22662069 24.91082759  4.03872414]
Wrong analytical solution for (-3.2266206896551743, 28.327310344827588,
0.7941034482758621) : [-3.22662069 28.32731034  0.55127586]
Wrong analytical solution for (-3.2266206896551743, 45.409724137931036,
1.6304827586206898) : [-3.22662069 45.40972414  1.55006897]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
3.1359655172413796) : [-2.22350681e+00 -2.15486311e+00  1.79787653e-15]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
3.303241379310345) : [-2.22364195e+00 -1.95823036e+00  1.14837367e-13]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
3.4705172413793104) : [-2.22375869e+00 -1.76160441e+00  4.99012872e-14]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
3.637793103448276) : [-2.22385705e+00 -1.56499016e+00  1.53296123e-14]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
3.8050689655172416) : [-2.22393704e+00 -1.36839171e+00  3.81192827e-15]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
3.972344827586207) : [-2.22399865e+00 -1.17181306e+00  8.41509914e-16]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
4.139620689655173) : [-2.22404189e+00 -9.75258192e-01  1.72696746e-16]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
4.306896551724138) : [-2.22406677e+00 -7.78731078e-01  3.12851367e-17]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
4.474172413793104) : [-2.22407330e+00 -5.82235700e-01  6.71512477e-14]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
4.641448275862069) : [-2.22406148e+00 -3.85776032e-01  5.25091836e-16]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
4.808724137931034) : [-2.22403133e+00 -1.89356068e-01  2.57962625e-16]
Wrong least_squares solution for (-2.217586206896552, -5.8375172413793095,
4.976) : [-2.22398288e+00  7.02022658e-03  1.27696461e-16]
Wrong least_squares solution for (-2.217586206896552, -2.4210344827586203,
0.125) : [-2.21739697 -2.62950785  0.30253891]
Wrong analytical solution for (-2.217586206896552, 24.9108275862069,
4.139620689655173) : [-2.21758621 24.91082759  4.03872414]
Wrong analytical solution for (-2.217586206896552, 28.327310344827588,
0.7941034482758621) : [-2.21758621 28.32731034  0.55127586]
Wrong analytical solution for (-2.217586206896552, 45.409724137931036,
1.6304827586206898) : [-2.21758621 45.40972414  1.55006897]
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
3.1359655172413796) : [-1.21177910e+00 -2.15505562e+00  9.09547116e-16]
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
3.303241379310345) : [-1.21185274e+00 -1.95843238e+00  1.47135029e-16]
```

```
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
3.4705172413793104) : [-1.21191641e+00 -1.76181686e+00  3.64528748e-17]
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
3.637793103448276) : [-1.21197006e+00 -1.56521341e+00  1.54546535e-14]
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
3.8050689655172416) : [-1.21201368e+00 -1.36862518e+00  3.83676289e-15]
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
3.972344827586207) : [-1.21204728e+00 -1.17205674e+00  8.45990965e-16]
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
4.139620689655173) : [-1.21207088e+00 -9.75512062e-01  1.73491981e-16]
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
4.306896551724138) : [-1.21208446e+00 -7.78995127e-01  3.14202273e-17]
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
4.474172413793104) : [-1.21208804e+00 -5.82509909e-01  7.50415590e-14]
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
4.641448275862069) : [-1.21208162e+00 -3.86060384e-01  5.27379811e-16]
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
4.808724137931034) : [-1.21206522e+00 -1.89650544e-01  2.59247148e-16]
Wrong least_squares solution for (-1.2085517241379318, -5.8375172413793095,
4.976) : [-1.21203883e+00  6.71564855e-03  1.28532033e-16]
Wrong least_squares solution for (-1.2085517241379318, -2.4210344827586203,
0.125) : [-1.20844932 -2.62805886  0.30131415]
Wrong analytical solution for (-1.2085517241379318, 24.9108275862069,
4.139620689655173) : [-1.20855172 24.91082759  4.03872414]
Wrong analytical solution for (-1.2085517241379318, 28.327310344827588,
0.7941034482758621) : [-1.20855172 28.32731034  0.55127586]
Wrong analytical solution for (-1.2085517241379318, 45.409724137931036,
1.6304827586206898) : [-1.20855172 45.40972414  1.55006897]
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
3.1359655172413796) : [-2.00050091e-01 -2.15513471e+00  7.82179077e-16]
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
3.303241379310345) : [-2.00062251e-01 -1.95851568e+00  1.48457371e-16]
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
3.4705172413793104) : [-2.00072763e-01 -1.76190437e+00  3.67279725e-17]
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
3.637793103448276) : [-2.00081621e-01 -1.56530478e+00  5.64484035e-18]
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
3.8050689655172416) : [-2.00088825e-01 -1.36872090e+00  6.65818090e-19]
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
3.972344827586207) : [-2.00094375e-01 -1.17215685e+00  8.48114167e-16]
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
4.139620689655173) : [-2.00098272e-01 -9.75616358e-01  1.73874107e-16]
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
4.306896551724138) : [-2.00100516e-01 -7.79103604e-01  3.14836819e-17]
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
4.474172413793104) : [-2.00101109e-01 -5.82622561e-01  8.22957126e-14]
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
4.641448275862069) : [-2.00100051e-01 -3.86177202e-01  5.28477653e-16]
```

```
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
4.808724137931034) : [-2.00097344e-01 -1.89771521e-01  2.59843306e-16]
Wrong least_squares solution for (-0.19951724137931137, -5.8375172413793095,
4.976) : [-2.00092990e-01  6.59052058e-03  1.28904209e-16]
Wrong least_squares solution for (-0.19951724137931137, -2.4210344827586203,
0.125) : [-0.19950038 -2.62746352  0.30081089]
Wrong analytical solution for (-0.19951724137931137, 24.9108275862069,
4.139620689655173) : [-0.19951724 24.91082759  4.03872414]
Wrong analytical solution for (-0.19951724137931137, 28.327310344827588,
0.7941034482758621) : [-0.19951724 28.32731035  0.55127586]
Wrong analytical solution for (-0.19951724137931137, 45.409724137931036,
1.6304827586206898) : [-0.19951724 45.40972414  1.55006923]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095,
3.1359655172413796) : [ 8.11679128e-01 -2.15510045e+00  7.79177523e-16]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095,
3.303241379310345) : [ 8.11728463e-01 -1.95847960e+00  1.47883346e-16]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095,
3.4705172413793104) : [ 8.11771111e-01 -1.76186646e+00  3.66077113e-17]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095,
3.637793103448276) : [ 8.11807047e-01 -1.56526505e+00  5.62951871e-18]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095,
3.8050689655172416) : [ 8.11836275e-01 -1.36867954e+00  3.84305880e-15]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095,
3.972344827586207) : [ 8.11858789e-01 -1.17211348e+00  8.47157236e-16]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095,
4.139620689655173) : [ 8.11874596e-01 -9.75571178e-01  1.73701285e-16]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095,
4.306896551724138) : [ 8.11883699e-01 -7.79056613e-01  3.14551389e-17]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095,
4.474172413793104) : [ 8.11886101e-01 -5.82573762e-01  7.86259969e-14]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095,
4.641448275862069) : [ 8.11881806e-01 -3.86126598e-01  5.27980978e-16]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095,
4.808724137931034) : [ 8.11870820e-01 -1.89719115e-01  2.59575831e-16]
Wrong least_squares solution for (0.809517241379309, -5.8375172413793095, 4.976)
: [8.11853149e-01 6.64472483e-03 1.28739046e-16]
Wrong least_squares solution for (0.809517241379309, -2.4210344827586203, 0.125)
: [ 0.80944876 -2.62772142  0.3010289 ]
Wrong analytical solution for (0.809517241379309, 24.9108275862069,
4.139620689655173) : [ 0.80951724 24.91082759  4.03872414]
Wrong analytical solution for (0.809517241379309, 28.327310344827588,
0.7941034482758621) : [ 0.80951724 28.32731034  0.55127586]
Wrong analytical solution for (0.809517241379309, 45.409724137931036,
1.6304827586206898) : [ 0.80951724 45.40972414  1.55006895]
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
3.1359655172413796) : [ 1.82340748e+00 -2.15495281e+00  1.43051515e-15]
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
3.303241379310345) : [ 1.82351833e+00 -1.95832484e+00  1.15484248e-13]
```

```
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
3.4705172413793104) : [ 1.82361408e+00 -1.76170367e+00  5.01346983e-14]
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
3.637793103448276) : [ 1.82369476e+00 -1.56509418e+00  1.53867776e-14]
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
3.8050689655172416) : [ 1.82376037e+00 -1.36850049e+00  3.82315618e-15]
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
3.972344827586207) : [ 1.82381092e+00 -1.17192660e+00  8.43515389e-16]
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
4.139620689655173) : [ 1.82384640e+00 -9.75376480e-01  1.73051111e-16]
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
4.306896551724138) : [ 1.82386682e+00 -7.78854108e-01  3.13457773e-17]
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
4.474172413793104) : [ 1.82387219e+00 -5.82363464e-01  6.96574239e-14]
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
4.641448275862069) : [ 1.82386251e+00 -3.85908522e-01  5.26112593e-16]
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
4.808724137931034) : [ 1.82383781e+00 -1.89493275e-01  2.58541703e-16]
Wrong least_squares solution for (1.8185517241379294, -5.8375172413793095,
4.976) : [1.82379809e+00 6.87831235e-03 1.28077615e-16]
Wrong least_squares solution for (1.8185517241379294, -2.4210344827586203,
0.125) : [ 1.81839705 -2.62883274  0.30196829]
Wrong analytical solution for (1.8185517241379294, 24.9108275862069,
4.139620689655173) : [ 1.81855172 24.91082759  4.03872414]
Wrong analytical solution for (1.8185517241379294, 28.327310344827588,
0.7941034482758621) : [ 1.81855172 28.32731034  0.55127586]
Wrong analytical solution for (1.8185517241379294, 45.409724137931036,
1.6304827586206898) : [ 1.81855172 45.40972414  1.55006896]
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
3.1359655172413796) : [ 2.83513388e+00 -2.15469164e+00  2.40482289e-15]
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
3.303241379310345) : [ 2.83530614e+00 -1.95804976e+00  1.13604453e-13]
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
3.4705172413793104) : [ 2.83545493e+00 -1.76141469e+00  4.94626743e-14]
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
3.637793103448276) : [ 2.83558028e+00 -1.56479132e+00  1.52241274e-14]
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
3.8050689655172416) : [ 2.83568222e+00 -1.36818377e+00  3.79166072e-15]
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
3.972344827586207) : [ 2.83576072e+00 -1.17159603e+00  8.37964420e-16]
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
4.139620689655173) : [ 2.83581581e+00 -9.75032078e-01  1.72075933e-16]
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
4.306896551724138) : [ 2.83584748e+00 -7.78495899e-01  3.11772918e-17]
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
4.474172413793104) : [ 2.83585575e+00 -5.81991471e-01  6.64163218e-14]
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
4.641448275862069) : [ 2.83584063e+00 -3.85522769e-01  5.23295747e-16]
```

```
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
4.808724137931034) : [ 2.83580215e+00 -1.89093790e-01  2.56921658e-16]
Wrong least_squares solution for (2.8275862068965516, -5.8375172413793095,
4.976) : [2.83574032e+00 7.29150314e-03 1.26995547e-16]
Wrong least_squares solution for (2.8275862068965516, -2.4210344827586203,
0.125) : [ 2.82734341 -2.63079823  0.30362949]
Wrong analytical solution for (2.8275862068965516, 24.9108275862069,
4.139620689655173) : [ 2.82758621 24.91082759  4.03872414]
Wrong analytical solution for (2.8275862068965516, 28.327310344827588,
0.7941034482758621) : [ 2.82758621 28.32731034  0.55127586]
Wrong analytical solution for (2.8275862068965516, 45.409724137931036,
1.6304827586206898) : [ 2.82758621 45.40972414  1.55006897]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
3.1359655172413796) : [ 3.84685726e+00 -2.15431672e+00  3.54477668e-15]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
3.303241379310345) : [ 3.84709080e+00 -1.95765485e+00  1.10922988e-13]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
3.4705172413793104) : [ 3.84729250e+00 -1.76099983e+00  4.85288336e-14]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
3.637793103448276) : [ 3.84746242e+00 -1.56435653e+00  1.50060249e-14]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
3.8050689655172416) : [ 3.84760055e+00 -1.36772907e+00  3.75132761e-15]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
3.972344827586207) : [ 3.84770691e+00 -1.17112145e+00  8.31178316e-16]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
4.139620689655173) : [ 3.84778149e+00 -9.74537649e-01  1.70908916e-16]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
4.306896551724138) : [ 3.84782431e+00 -7.77981648e-01  3.09686162e-17]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
4.474172413793104) : [ 3.84783538e+00 -5.81457432e-01  7.81414992e-14]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
4.641448275862069) : [ 3.84781473e+00 -3.84968976e-01  5.19876683e-16]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
4.808724137931034) : [ 3.84776237e+00 -1.88520281e-01  2.54859310e-16]
Wrong least_squares solution for (3.8366206896551702, -5.8375172413793095,
4.976) : [3.84767835e+00 7.88468665e-03 1.25551039e-16]
Wrong least_squares solution for (3.8366206896551702, -2.4210344827586203,
0.125) : [ 3.83628676 -2.63361919  0.30601332]
Wrong analytical solution for (3.8366206896551702, 24.9108275862069,
4.139620689655173) : [ 3.83662069 24.91082759  4.03872414]
Wrong analytical solution for (3.8366206896551702, 28.327310344827588,
0.7941034482758621) : [ 3.83662069 28.32731034  0.55127586]
Wrong analytical solution for (3.8366206896551702, 45.409724137931036,
1.6304827586206898) : [ 3.83662069 45.40972414  1.55006897]
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095,
3.1359655172413796) : [ 4.85857652e+00 -2.15382766e+00  4.46016405e-11]
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095,
3.303241379310345) : [ 4.85887119e+00 -1.95713973e+00  1.07457707e-13]
```

```
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095,
3.4705172413793104) : [ 4.85912564e+00 -1.76045869e+00  4.73501993e-14]
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095,
3.637793103448276) : [ 4.85933995e+00 -1.56378940e+00  1.47405627e-14]
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095,
3.8050689655172416) : [ 4.85951413e+00 -1.36713597e+00  3.70479100e-15]
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095,
3.972344827586207) : [ 4.85964819e+00 -1.17050242e+00  8.23816612e-16]
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095,
4.139620689655173) : [ 4.85974212e+00 -9.73892725e-01  1.69683059e-16]
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095,
4.306896551724138) : [ 4.85979595e+00 -7.77310869e-01  3.07391347e-17]
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095,
4.474172413793104) : [ 4.85980969e+00 -5.80760840e-01  1.12973077e-13]
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095,
4.641448275862069) : [ 4.85978337e+00 -3.84246620e-01  5.16178887e-16]
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095,
4.808724137931034) : [ 4.85971702e+00 -1.87772209e-01  2.52484842e-16]
Wrong least_squares solution for (4.845655172413792, -5.8375172413793095, 4.976)
: [4.85961088e+00 8.65704933e-03 1.02119587e-13]
Wrong least_squares solution for (4.845655172413792, -2.4210344827586203, 0.125)
: [ 4.84522603 -2.63729753  0.30912093]
Wrong analytical solution for (4.845655172413792, 24.9108275862069,
4.139620689655173) : [ 4.84565517 24.91082759  4.03872414]
Wrong analytical solution for (4.845655172413792, 28.327310344827588,
0.7941034482758621) : [ 4.84565517 28.32731034  0.55127586]
Wrong analytical solution for (4.845655172413792, 45.409724137931036,
1.6304827586206898) : [ 4.84565517 45.40972414  1.55006897]
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095,
3.1359655172413796) : [ 5.87029059e+00 -2.15322402e+00  1.88657695e-10]
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095,
3.303241379310345) : [ 5.87064617e+00 -1.95650391e+00  1.03242890e-13]
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095,
3.4705172413793104) : [ 5.87095316e+00 -1.75979076e+00  4.59455255e-14]
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095,
3.637793103448276) : [ 5.87121167e+00 -1.56308939e+00  1.44355952e-14]
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095,
3.8050689655172416) : [ 5.87142171e+00 -1.36640391e+00  3.65461474e-15]
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095,
3.972344827586207) : [ 5.87158327e+00 -1.16973835e+00  8.16541213e-16]
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095,
4.139620689655173) : [ 5.87169637e+00 -9.73096695e-01  1.68536611e-16]
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095,
4.306896551724138) : [ 5.87176103e+00 -7.76482926e-01  3.05100619e-17]
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095,
4.474172413793104) : [ 5.87177726e+00 -5.79901039e-01  1.77431178e-13]
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095,
4.641448275862069) : [ 5.87174511e+00 -3.83355016e-01  5.12515291e-16]
```

```
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095,
4.808724137931034) : [ 5.87166494e+00 -1.86850914e-01  1.53911766e-13]
Wrong least_squares solution for (5.854689655172411, -5.8375172413793095, 4.976)
: [5.87153605e+00 9.61208067e-03 1.00758009e-13]
Wrong least_squares solution for (5.854689655172411, -2.4210344827586203, 0.125)
: [ 5.85416012 -2.6418357   0.31295381]
Wrong analytical solution for (5.854689655172411, 24.9108275862069,
4.139620689655173) : [ 5.85468966 24.91082759  4.03872414]
Wrong analytical solution for (5.854689655172411, 28.327310344827588,
0.7941034482758621) : [ 5.85468966 28.32731034  0.55127586]
Wrong analytical solution for (5.854689655172411, 45.409724137931036,
1.6304827586206898) : [ 5.85468966 45.40972414  1.55006897]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095,
3.1359655172413796) : [ 6.88199840e+00 -2.15250523e+00  1.20208978e-10]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095,
3.303241379310345) : [ 6.88241464e+00 -1.95574680e+00  9.83347691e-14]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095,
3.4705172413793104) : [ 6.88277392e+00 -1.75899540e+00  4.43355083e-14]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095,
3.637793103448276) : [ 6.88307638e+00 -1.56225583e+00  1.40983597e-14]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095,
3.8050689655172416) : [ 6.88332202e+00 -1.36553220e+00  3.60315476e-15]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095,
3.972344827586207) : [ 6.88351087e+00 -1.16882852e+00  8.09990231e-16]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095,
4.139620689655173) : [ 6.88364291e+00 -9.72148804e-01  1.67609870e-16]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095,
4.306896551724138) : [ 6.88371818e+00 -7.75497035e-01  3.03046505e-17]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095,
4.474172413793104) : [ 6.88373671e+00 -5.78877211e-01  2.75131151e-13]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095,
4.641448275862069) : [ 6.88369904e+00 -3.82296266e-01  2.31432245e-13]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095,
4.808724137931034) : [ 6.88360407e+00 -1.85751411e-01  1.52493225e-13]
Wrong least_squares solution for (6.863724137931033, -5.8375172413793095, 4.976)
: [6.88345252e+00 1.07493020e-02 9.92279082e-14]
Wrong least_squares solution for (6.863724137931033, -2.4210344827586203, 0.125)
: [ 6.86308796 -2.64723677  0.31751382]
Wrong analytical solution for (6.863724137931033, 24.9108275862069,
4.139620689655173) : [ 6.86372414 24.91082759  4.03872414]
Wrong analytical solution for (6.863724137931033, 28.327310344827588,
0.7941034482758621) : [ 6.86372414 28.32731034  0.55127586]
Wrong analytical solution for (6.863724137931033, 45.409724137931036,
1.6304827586206898) : [ 6.86372414 45.40972414  1.55006897]
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095,
3.1359655172413796) : [ 7.89369888e+00 -2.15167058e+00  3.93469083e-10]
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095,
3.303241379310345) : [ 7.89417546e+00 -1.95486767e+00  9.28139576e-14]
```

```
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095,
3.4705172413793104) : [ 7.89458673e+00 -1.75807187e+00  4.25422506e-14]
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095,
3.637793103448276) : [ 7.89493285e+00 -1.56128793e+00  1.37348718e-14]
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095,
3.8050689655172416) : [ 7.89521384e+00 -1.36451999e+00  3.55235497e-15]
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095,
3.972344827586207) : [ 7.89542969e+00 -1.16777206e+00  8.04734288e-16]
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095,
4.139620689655173) : [ 7.89558042e+00 -9.71048149e-01  1.67039093e-16]
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095,
4.306896551724138) : [ 7.89566605e+00 -7.74352257e-01  3.01478713e-17]
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095,
4.474172413793104) : [ 7.89568735e+00 -5.77692393e-01  9.86311625e-09]
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095,
4.641448275862069) : [ 7.89564276e+00 -3.81063457e-01  2.30221075e-13]
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095,
4.808724137931034) : [ 7.89553321e+00 -1.84474717e-01  1.51029466e-13]
Wrong least_squares solution for (7.872758620689652, -5.8375172413793095, 4.976)
: [7.89535878e+00 1.20697929e-02 9.75453143e-14]
Wrong least_squares solution for (7.872758620689652, -2.4210344827586203, 0.125)
: [ 7.87200843 -2.65350435  0.32280315]
Wrong analytical solution for (7.872758620689652, 24.9108275862069,
4.139620689655173) : [ 7.87275862 24.91082759  4.03872414]
Wrong analytical solution for (7.872758620689652, 28.327310344827588,
0.7941034482758621) : [ 7.87275862 28.32731034  0.55127586]
Wrong analytical solution for (7.872758620689652, 45.409724137931036,
1.6304827586206898) : [ 7.87275862 45.40972414  1.55006897]
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095,
3.1359655172413796) : [ 8.90539096e+00 -2.15071929e+00  1.36154580e-09]
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095,
3.303241379310345) : [ 8.90592751e+00 -1.95386568e+00  8.67596749e-14]
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095,
3.4705172413793104) : [ 8.90639044e+00 -1.75701926e+00  4.05880275e-14]
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095,
3.637793103448276) : [ 8.90677989e+00 -1.56018477e+00  1.33523618e-14]
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095,
3.8050689655172416) : [ 8.90709589e+00 -1.36336633e+00  3.50495798e-15]
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095,
3.972344827586207) : [ 8.90733844e+00 -1.16656796e+00  8.01636098e-16]
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095,
4.139620689655173) : [ 8.90750755e+00 -9.69793682e-01  1.67035479e-16]
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095,
4.306896551724138) : [ 8.90760326e+00 -7.73047502e-01  3.00784096e-17]
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095,
4.474172413793104) : [ 8.90762642e+00 -5.76337427e-01  9.78567988e-09]
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095,
4.641448275862069) : [ 8.90757528e+00 -3.79658374e-01  2.29257559e-13]
```

```
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095,
4.808724137931034) : [ 8.90745089e+00 -1.83019615e-01  1.49580902e-13]
Wrong least_squares solution for (8.881793103448274, -5.8375172413793095, 4.976)
: [8.90725334e+00 1.35748105e-02 9.57383423e-14]
Wrong least_squares solution for (8.881793103448274, -2.4210344827586203, 0.125)
: [ 8.88092041 -2.66064266  0.32882437]
Wrong analytical solution for (8.881793103448274, 24.9108275862069,
4.139620689655173) : [ 8.8817931  24.91082759  4.03872414]
Wrong analytical solution for (8.881793103448274, 28.327310344827588,
0.7941034482758621) : [ 8.8817931  28.32731034  0.55127586]
Wrong analytical solution for (8.881793103448274, 45.409724137931036,
1.6304827586206898) : [ 8.8817931  45.40972414  1.55006897]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095,
3.1359655172413796) : [ 9.91707355e+00 -2.14965045e+00  2.51421177e-09]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095,
3.303241379310345) : [ 9.91766968e+00 -1.95273987e+00  8.02321998e-14]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095,
3.4705172413793104) : [ 9.91818387e+00 -1.75583659e+00  3.84940222e-14]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095,
3.637793103448276) : [ 9.91861629e+00 -1.55894530e+00  1.29610966e-14]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095,
3.8050689655172416) : [ 9.91896694e+00 -1.36207011e+00  3.46554886e-15]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095,
3.972344827586207) : [ 9.91923584e+00 -1.16521507e+00  8.02190512e-16]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095,
4.139620689655173) : [ 9.91942300e+00 -9.68384201e-01  1.67965656e-16]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095,
4.306896551724138) : [ 9.91952958e+00 -7.71586886e-01  3.87661912e-08]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095,
4.474172413793104) : [ 9.91955316e+00 -5.74815038e-01  9.71641024e-09]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095,
4.641448275862069) : [ 9.91949517e+00 -3.78079674e-01  2.28757706e-13]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095,
4.808724137931034) : [ 9.91935564e+00 -1.81384717e-01  1.48257673e-13]
Wrong least_squares solution for (9.890827586206896, -5.8375172413793095, 4.976)
: [9.91913468e+00 1.52657919e-02 9.38577773e-14]
Wrong least_squares solution for (9.890827586206896, -2.4210344827586203, 0.125)
: [ 9.88982278 -2.66865654  0.33558037]
Wrong analytical solution for (9.890827586206896, 24.9108275862069,
4.139620689655173) : [ 9.89082759 24.91082759  4.03872414]
Wrong analytical solution for (9.890827586206896, 28.327310344827588,
0.7941034482758621) : [ 9.89082759 28.32731034  0.55127586]
Wrong analytical solution for (9.890827586206896, 45.409724137931036,
1.6304827586206898) : [ 9.89082759 45.40972414  1.55006897]
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
3.1359655172413796) : [ 1.09287456e+01 -2.14846342e+00  7.19768286e-14]
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
3.303241379310345) : [ 1.09294008e+01 -1.95148915e+00  7.33104199e-14]
```

```
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
3.4705172413793104) : [ 1.09299659e+01 -1.75452270e+00  3.62793958e-14]
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
3.637793103448276) : [ 1.09304408e+01 -1.55756830e+00  1.25697817e-14]
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
3.8050689655172416) : [ 1.09308257e+01 -1.36063008e+00  3.43884246e-15]
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
3.972344827586207) : [ 1.09311206e+01 -1.16371209e+00  8.08150302e-16]
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
4.139620689655173) : [ 1.09313254e+01 -9.66818353e-01  1.70296622e-16]
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
4.306896551724138) : [ 1.09314415e+01 -7.69958281e-01  3.91110807e-08]
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
4.474172413793104) : [ 1.09314662e+01 -5.73123766e-01  9.67320838e-09]
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
4.641448275862069) : [ 1.09314010e+01 -3.76325844e-01  2.28965978e-13]
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
4.808724137931034) : [ 1.09312460e+01 -1.79568454e-01  1.47183254e-13]
Wrong least_squares solution for (10.899862068965515, -5.8375172413793095,
4.976) : [1.09310013e+01 1.71443580e-02 9.19588470e-14]
Wrong least_squares solution for (10.899862068965515, -2.4210344827586203,
0.125) : [10.89871438 -2.67755137  0.34307441]
Wrong analytical solution for (10.899862068965515, 24.9108275862069,
4.139620689655173) : [10.89986207 24.91082759  4.03872414]
Wrong analytical solution for (10.899862068965515, 28.327310344827588,
0.7941034482758621) : [10.89986207 28.32731034  0.55127586]
Wrong analytical solution for (10.899862068965515, 45.409724137931036,
1.6304827586206898) : [10.89986207 45.40972414  1.55006897]
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
3.1359655172413796) : [ 1.19404060e+01 -2.14715622e+00  6.30631967e-14]
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
3.303241379310345) : [ 1.19411199e+01 -1.95011233e+00  6.60781262e-14]
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
3.4705172413793104) : [ 1.19417353e+01 -1.75307634e+00  3.39604090e-14]
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
3.637793103448276) : [ 1.19422523e+01 -1.55605247e+00  1.21875574e-14]
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
3.8050689655172416) : [ 1.19426710e+01 -1.35904486e+00  3.43115677e-15]
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
3.972344827586207) : [ 1.19429914e+01 -1.16205756e+00  8.22135738e-16]
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
4.139620689655173) : [ 1.19432153e+01 -9.65101951e-01  1.02458313e-07]
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
4.306896551724138) : [ 1.19433387e+01 -7.68165486e-01  3.97323056e-08]
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
4.474172413793104) : [ 1.19433640e+01 -5.71261985e-01  9.68797409e-09]
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
4.641448275862069) : [ 1.19432913e+01 -3.74395197e-01  2.30290208e-13]
```

```
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
4.808724137931034) : [ 1.19431205e+01 -1.77569080e-01  1.46572987e-13]
Wrong least_squares solution for (11.908896551724137, -5.8375172413793095,
4.976) : [1.19428517e+01 1.92123164e-02 9.01420512e-14]
Wrong least_squares solution for (11.908896551724137, -2.4210344827586203,
0.125) : [11.90759407 -2.6873332   0.3513101 ]
Wrong analytical solution for (11.908896551724137, 24.9108275862069,
4.139620689655173) : [11.90889655 24.91082759  4.03872414]
Wrong analytical solution for (11.908896551724137, 45.409724137931036,
1.6304827586206898) : [11.90889655 45.40972414  1.55006897]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
3.1359655172413796) : [ 1.29520537e+01 -2.14572803e+00  5.35296029e-14]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
3.303241379310345) : [ 1.29528256e+01 -1.94860805e+00  5.85150628e-14]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
3.4705172413793104) : [ 1.29534909e+01 -1.75149609e+00  3.15538967e-14]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
3.637793103448276) : [ 1.29540496e+01 -1.55439633e+00  1.18474796e-14]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
3.8050689655172416) : [ 1.29545016e+01 -1.35731290e+00  3.46346277e-15]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
3.972344827586207) : [ 1.29548470e+01 -1.16024989e+00  8.52135782e-16]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
4.139620689655173) : [ 1.29550879e+01 -9.63218735e-01  1.05704455e-07]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
4.306896551724138) : [ 1.29552198e+01 -7.66206781e-01  4.08954425e-08]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
4.474172413793104) : [ 1.29552454e+01 -5.69227905e-01  9.82670393e-09]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
4.641448275862069) : [ 1.29551646e+01 -3.72285872e-01  2.34140979e-13]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
4.808724137931034) : [ 1.29549776e+01 -1.75384663e-01  1.47199373e-13]
Wrong least_squares solution for (12.917931034482756, -5.8375172413793095,
4.976) : [1.29546839e+01 2.14729104e-02 3.42557351e-13]
Wrong least_squares solution for (12.917931034482756, -2.4210344827586203,
0.125) : [12.91646065 -2.69800866  0.36029143]
Wrong analytical solution for (12.917931034482756, 24.9108275862069,
4.139620689655173) : [12.91793103 24.91082759  4.03872414]
Wrong analytical solution for (12.917931034482756, 45.409724137931036,
1.6304827586206898) : [12.91793103 45.40972414  1.55006897]
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
3.1359655172413796) : [ 1.39636875e+01 -2.14417742e+00  4.27792117e-14]
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
3.303241379310345) : [ 1.39645171e+01 -1.94697487e+00  5.03688525e-14]
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
3.4705172413793104) : [ 1.39652316e+01 -1.74978042e+00  2.91365060e-14]
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
3.637793103448276) : [ 1.39658313e+01 -1.55259826e+00  1.17048069e-14]
```

```
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
3.8050689655172416) : [ 1.39663161e+01 -1.35543253e+00  3.63008713e-15]
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
3.972344827586207) : [ 1.39666891e+01 -1.15829740e+00  2.56159789e-07]
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
4.139620689655173) : [ 1.39669435e+01 -9.61174265e-01  1.12924958e-07]
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
4.306896551724138) : [ 1.39670834e+01 -7.64080340e-01  4.36307157e-08]
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
4.474172413793104) : [ 1.39671088e+01 -5.67019612e-01  1.02334724e-08]
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
4.641448275862069) : [ 1.39670196e+01 -3.69995865e-01  2.49046533e-13]
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
4.808724137931034) : [ 1.39668160e+01 -1.73013112e-01  1.54207169e-13]
Wrong least_squares solution for (13.926965517241378, -5.8375172413793095,
4.976) : [1.39664974e+01 2.39258282e-02 1.38032614e-12]
Wrong least_squares solution for (13.926965517241378, -2.4210344827586203,
0.125) : [13.92531293 -2.70958503  0.37002273]
Wrong analytical solution for (13.926965517241378, 24.9108275862069,
4.139620689655173) : [13.92696552 24.91082759  4.03872414]
Wrong analytical solution for (13.926965517241378, 45.409724137931036,
1.6304827586206898) : [13.92696552 45.40972414  1.55006897]
Wrong least_squares solution for (14.936, -5.8375172413793095,
3.1359655172413796) : [ 1.49753063e+01 -2.14250269e+00  8.89711262e-12]
Wrong least_squares solution for (14.936, -5.8375172413793095,
3.303241379310345) : [ 1.49761927e+01 -1.94521081e+00  5.63831682e-15]
Wrong least_squares solution for (14.936, -5.8375172413793095,
3.4705172413793104) : [ 1.49769560e+01 -1.74792731e+00  6.27300140e-16]
Wrong least_squares solution for (14.936, -5.8375172413793095,
3.637793103448276) : [ 1.49775962e+01 -1.55065625e+00  1.18567207e-16]
Wrong least_squares solution for (14.936, -5.8375172413793095,
3.8050689655172416) : [ 1.49781133e+01 -1.35340173e+00  2.72791843e-17]
Wrong least_squares solution for (14.936, -5.8375172413793095,
3.972344827586207) : [ 1.49785073e+01 -1.15616782e+00  5.45435756e-18]
Wrong least_squares solution for (14.936, -5.8375172413793095,
4.139620689655173) : [ 1.49787783e+01 -9.58958599e-01  6.31850209e-19]
Wrong least_squares solution for (14.936, -5.8375172413793095,
4.306896551724138) : [ 1.49789263e+01 -7.61778143e-01  1.16598231e-20]
Wrong least_squares solution for (14.936, -5.8375172413793095,
4.474172413793104) : [ 1.49789513e+01 -5.64630520e-01  5.01297334e-22]
Wrong least_squares solution for (14.936, -5.8375172413793095,
4.641448275862069) : [ 1.49788536e+01 -3.67519794e-01  7.13732883e-20]
Wrong least_squares solution for (14.936, -5.8375172413793095,
4.808724137931034) : [ 1.49786331e+01 -1.70450025e-01  7.44834129e-20]
Wrong least_squares solution for (14.936, -5.8375172413793095, 4.976) :
[1.49782900e+01 2.65747360e-02 1.11254408e-16]
Wrong least_squares solution for (14.936, -2.4210344827586203, 0.125) :
[14.93414967 -2.72207015  0.38050863]
```

```
Wrong least_squares solution for (14.936, -2.4210344827586203,
0.2922758620689655) : [14.93414967 -2.72207015  0.54778449]
Wrong least_squares solution for (14.936, -2.4210344827586203,
0.45955172413793105) : [14.93414967 -2.72207015  0.71506036]
Wrong least_squares solution for (14.936, -2.4210344827586203,
0.6268275862068966) : [14.93414967 -2.72207015  0.88233622]
Wrong least_squares solution for (14.936, -2.4210344827586203,
0.7941034482758621) : [14.93414967 -2.72207015  1.04961208]
Wrong least_squares solution for (14.936, -2.4210344827586203,
0.9613793103448276) : [14.93414967 -2.72207016  1.21688795]
Wrong least_squares solution for (14.936, -2.4210344827586203,
1.4632068965517242) : [14.93414967 -2.72207015  1.71871553]
Wrong least_squares solution for (14.936, -2.4210344827586203,
1.6304827586206898) : [14.93414967 -2.72207015  1.88599139]
Wrong analytical solution for (14.936, 24.9108275862069, 4.139620689655173) :
[14.936      24.91082759  4.03872414]
Wrong analytical solution for (14.936, 45.409724137931036, 1.6304827586206898) :
[14.936      45.40972414  1.55006897]
Tried: 27000
Analytical approach - Solved: 27000, No solution: 0, Wrong solution: 82,
Function evaluations: 810000
Least squares approach - Solved: 26603, No solution: 0 Wrong solution: 397,
Function evaluations: 1564701
Analytical approach - descriptive statistics for no solutions

       mcp_aa  mcp_fe  cmc_fe
count     0.0     0.0     0.0
mean      NaN     NaN     NaN
std       NaN     NaN     NaN
min       NaN     NaN     NaN
25%       NaN     NaN     NaN
50%       NaN     NaN     NaN
75%       NaN     NaN     NaN
max       NaN     NaN     NaN


Least squares approach - descriptive statistics for no solutions

       mcp_aa  mcp_fe  cmc_fe
count     0.0     0.0     0.0
mean      NaN     NaN     NaN
std       NaN     NaN     NaN
min       NaN     NaN     NaN
25%       NaN     NaN     NaN
50%       NaN     NaN     NaN
75%       NaN     NaN     NaN
max       NaN     NaN     NaN


Tried: 27000 Analytical approach - Solved: 27000, No solution: 0, Wrong
```

solution: 82, Function evaluations: 810000 Wrong solutions - alternative solution in addition to first accurate one (CMC FE off by ~ 0.2°) Least squares approach (entire matrix) - Solved: 27000, No solution: 0 Wrong solution: 0, Function evaluations: 6140976 Least squares approach (position vector) - Solved: 26603, No solution: 0 Wrong solution: 397, Function evaluations: 1564701 Wrong solutions - always for MCP FE -5.8375172413793095° and -2.4210344827586203°, CMC FE varies - possible alternative solutions since 3D positions are relatively close for small CMC and MCP FE angles. Analytical approach - descriptive statistics for no solutions