问题2

In [1]:
```
import math
import plotly
import numpy as np
import pandas as pd
import cufflinks as cf
from mitosheet import sheet
import plotly.express as px
import plotly.graph_objects as go

from IPython.core.interactiveshell import InteractiveShell
# InteractiveShell.ast_node_interactivity = 'all'
InteractiveShell.ast_node_interactivity = 'last'
```

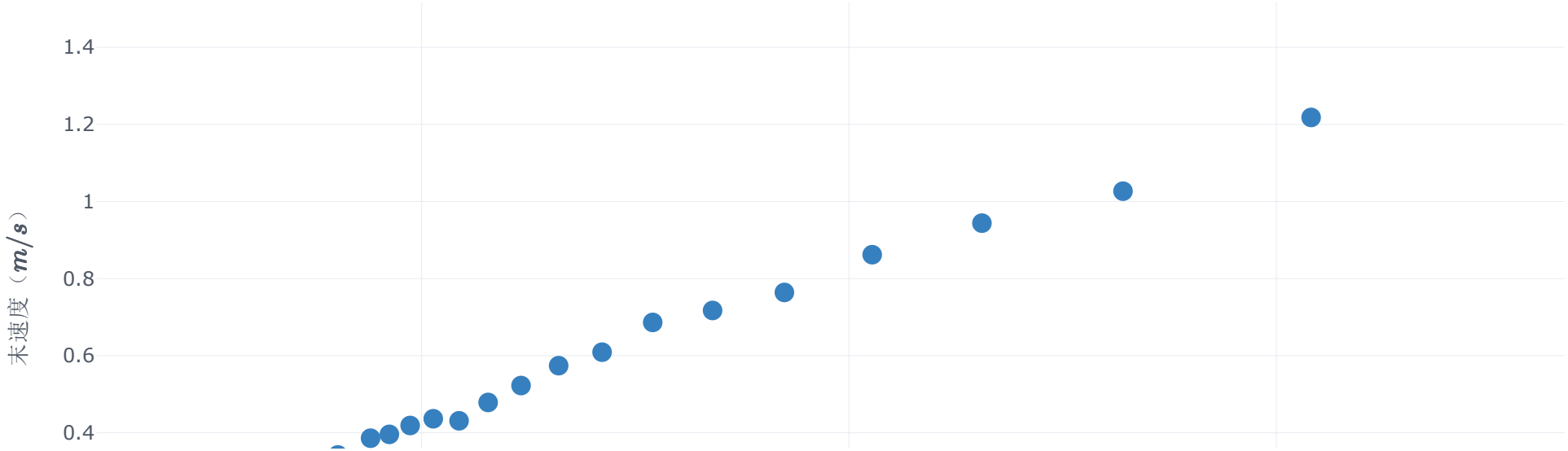
In [ ]:
```
```

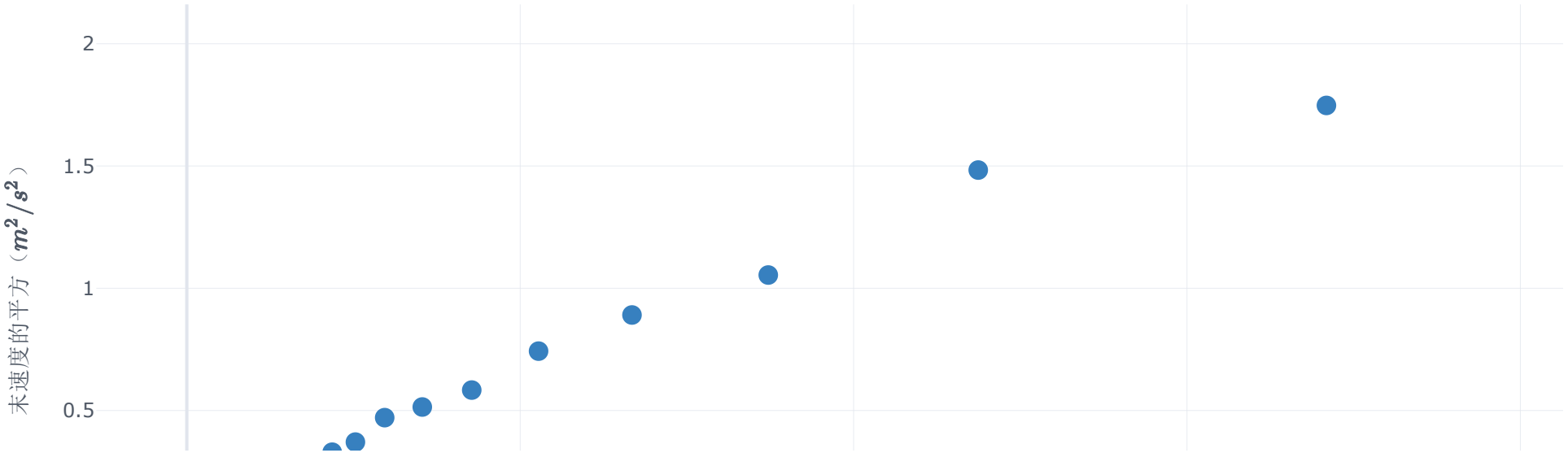
In [2]:
```
# 附件参数
copper_ball_material = 'copper'  # 小球材质
copper_ball_R = 10e-3     # m 小球半径
copper_ball_M = 35e-3     # kg 小球质量
mag_sen_rubber_L = 10e-2  # m 磁敏橡胶长

# 查找参数
mu1 = copper_Poisson_ratio = 0.3          # 铜的泊松比
mu2 = mag_sen_rubber_Poisson_ratio = 0.5  # 橡胶的泊松比
E1 = copper_elastic_modulus = 110000      # MPa 铜的弹性模量
E2 = mag_sen_rubber_elastic_modulus = 6   # MPa 橡胶的弹性模量
```

In [ ]:
```
```

In [3]:
```
# todo 速度
B_values = [0, 50, 100, 150, 200]
sheet_names = ["0 mT", "50 mT", "100 mT", "150 mT", "200mT（测试集）"]
data_v0_v1 = pd.read_excel("A题附件1.xlsx", sheet_name=sheet_names[0]).iloc[:, :2]
# data_v0_v1

data_v0_v1.iplot(
    kind='scatter',
    mode='markers',
```

```
    x="初速度（m/s）",
    y="末速度（m/s）",
    colors=['blue'],
    xTitle='$初速度（m/s）$',
    yTitle='$末速度（m/s）$',
#     size=5,
#     bestfit_colors=['pink'],
)

# todo 速度的平方
data_v02_v12 = data_v0_v1**2
data_v02_v12.rename(columns={"初速度（m/s）": "初速度的平方（m^2/s^2）", "末速度（m/s）": "末速度的平方（m^2/s^2）"}, inplace=True)
# data_v02_v12

data_v02_v12.iplot(
    kind='scatter',
    mode='markers',
    x="初速度的平方（m^2/s^2）",
    y="末速度的平方（m^2/s^2）",
    colors=['blue'],
    xTitle='$初速度的平方（m^2/s^2）$',
    yTitle='$末速度的平方（m^2/s^2）$',
#     size=5,
#     bestfit_colors=['pink'],
)

# todo 速度的平方之差 v0^2 - v1^2
v0_2_sub_v1_2 = pd.DataFrame(data_v02_v12.iloc[:, 0] - data_v02_v12.iloc[:, 1], columns=["初、末速度平方之差"])
# v0_2_sub_v1_2
```

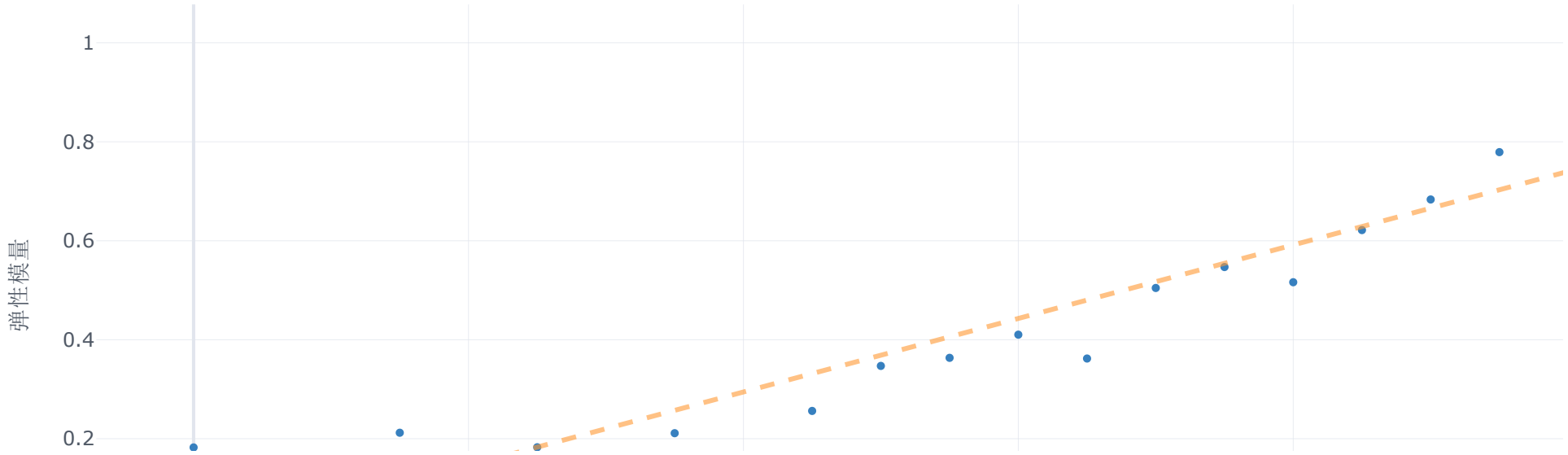

1.4
1.2
1
0.8
0.6
0.4
末速度（m/s）

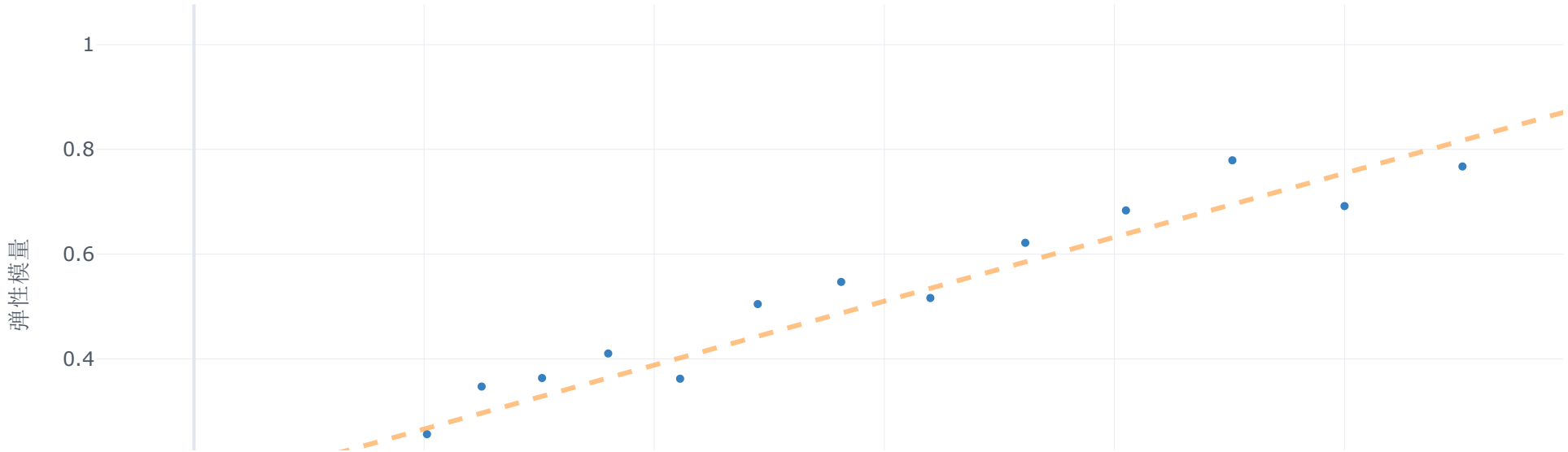


In [ ]:

In [4]:
```
# todo 磁感应强度 - 弹性模量
data_B_E = pd.read_csv("B-E.csv", index_col=0)
# data_B_E

data_B_E.iplot(
    kind='scatter',
    mode='markers',
    x='磁感应强度',
    y='弹性模量',
    bestfit=True,
    colors=['blue'],
    size=5,
```

```
    xTitle='$磁感应强度$',
    yTitle='$弹性模量$',
#     bestfit_colors=['pink'],
)

# todo 磁感应强度^2 - 弹性模量
data_B2_E = data_B_E.copy()
data_B2_E.iloc[:, 0] = data_B2_E.iloc[:, 0]**2
data_B2_E.rename(columns={"磁感应强度":"磁感应强度的平方"}, inplace=True)
# data_B2_E

data_B2_E.iplot(
    kind='scatter',
    mode='markers',
    x='磁感应强度的平方',
    y='弹性模量',
    bestfit=True,
    colors=['blue'],
    size=5,
    xTitle='$磁感应强度的平方$',
    yTitle='$弹性模量$',
)
```


1
0.8
0.6
0.4
0.2
弹性模量



In [ ]:

In [5]:
```python
import math
import latexify

@latexify.with_latex
def K(v_0, v_1, E=6):  # Latex
    return 202.13 * R / (2.44 * B**2 + 0.14 + E) - 219.6 * R**2 * math.sqrt((v_0**2 - v_1**2) / (L * (2.44 * B**2 + 0.14 + E)**(4/3)))
K
```

Out[5]:

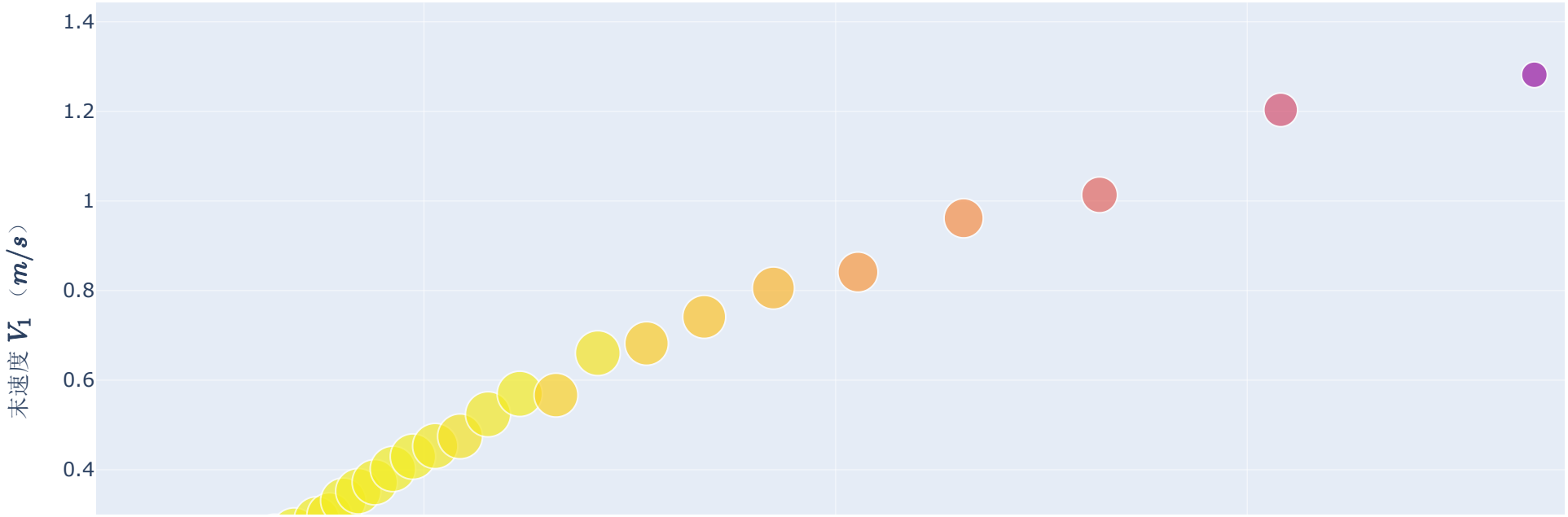
$$\mathrm{K}(v_0, v_1, E) \triangleq \frac{202.13R}{2.44B^2 + 0.14 + E} - 219.6R^2 \sqrt{\frac{v_0^2 - v_1^2}{L(2.44B^2 + 0.14 + E)^{\frac{4}{3}}}}$$

In [6]:
```python
def get_K(v_0, v_1, B, E=6, R=copper_ball_R, L=mag_sen_rubber_L):  # latex
    return 202.13 * R / (2.44 * B**2 + 0.14 + E) - 219.6 * R**2 * ((v_0**2 - v_1**2) / (L * (2.44 * B**2 + 0.14 + E)**(4/3)))**0.5
```

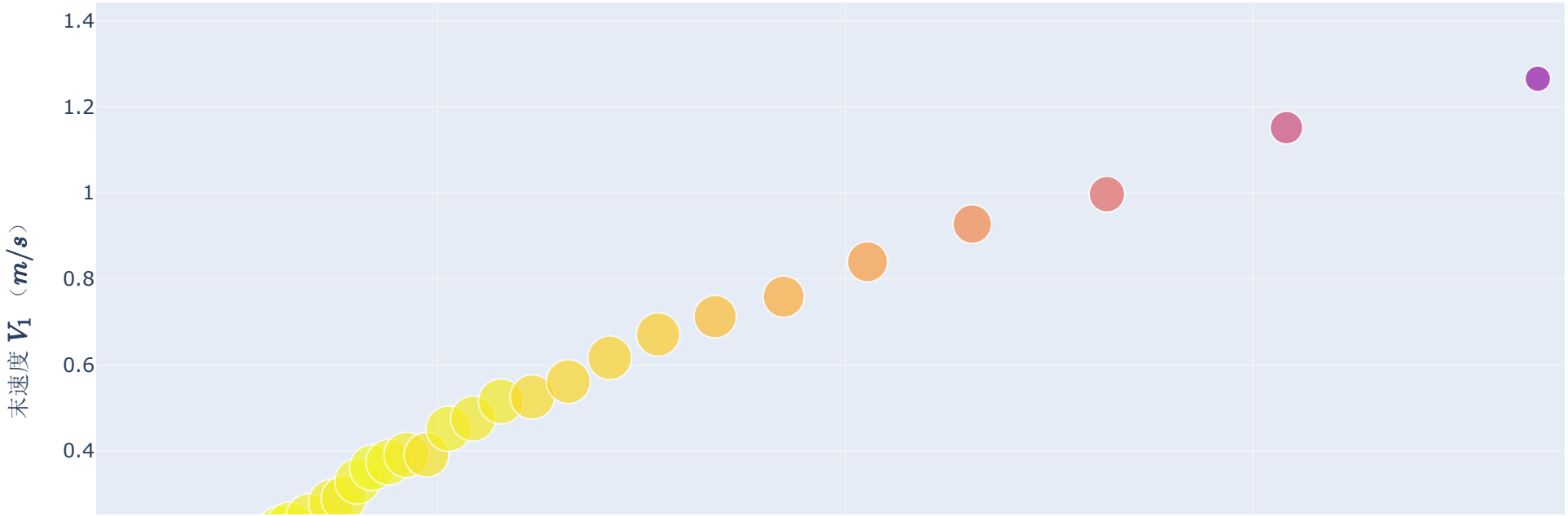
In [ ]:
```python
```

In [7]:
```python
B_values = [0, 0.050, 0.100, 0.150, 0.200]  # T
sheet_names = ["0 mT", "50 mT", "100 mT", "150 mT", "200mT（测试集）", ]
B_value_str = ["0mT", "50mT", "100mT", "150mT", "200mT", ]
cols = {0: r"$初速度~V_0~（m/s）$", 1: r"$末速度~V_1~（m/s）$", 2: r"滚动摩擦<br> 系数 K", 3: "K_norm"}
titles = [r"$B = 0 mT$", r"$B = 50 mT$", r"$B = 100 mT$", r"$B = 150 mT$", r"$B = 200 mT$", ]

for i in range(1, 5):
    B = B_values[i]
    sheet_name = sheet_names[i]
    data_v0_v1 = pd.read_excel("A题附件1.xlsx", sheet_name=sheet_names[i]).iloc[:, :2]
    K = pd.DataFrame(get_K(data_v0_v1.iloc[:, 0].values, data_v0_v1.iloc[:, 1].values, B))
    data_K = pd.concat([data_v0_v1, K], axis=1, ignore_index=True)
    data_K = pd.concat(
        [data_K, (data_K.iloc[:, 2] - min(data_K.iloc[:, 2])) / (max(data_K.iloc[:, 2]) - min(data_K.iloc[:, 2]))],
        axis=1,
        ignore_index=True,
    )
    data_K.rename(columns=cols, inplace=True)
#     data_K
    fig = px.scatter(
        data_K,
        x=cols[0],
        y=cols[1],
        labels=cols[2],
        size=cols[3],
        color=cols[2],
        hover_name=cols[2],
    )
    fig.update_layout(
        title=titles[i],
    )
#     fig.show()
    fig.write_image("v0-v1-K,B=%s.png" % B_value_str[i])
```

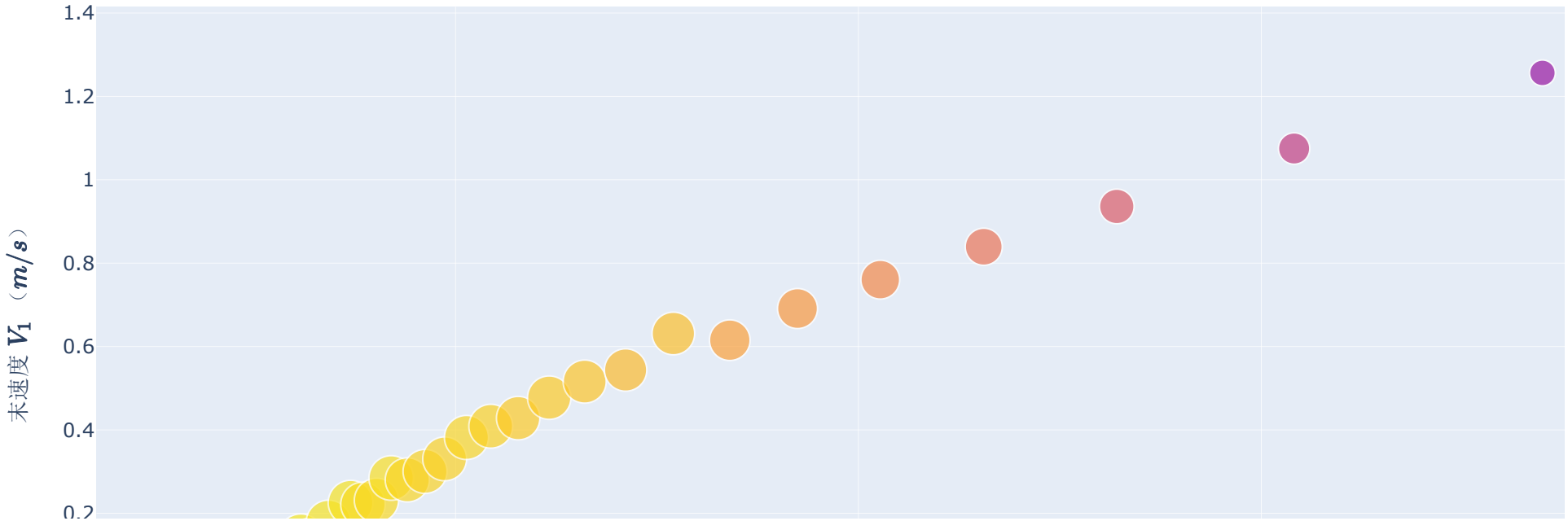
$B = 50mT$

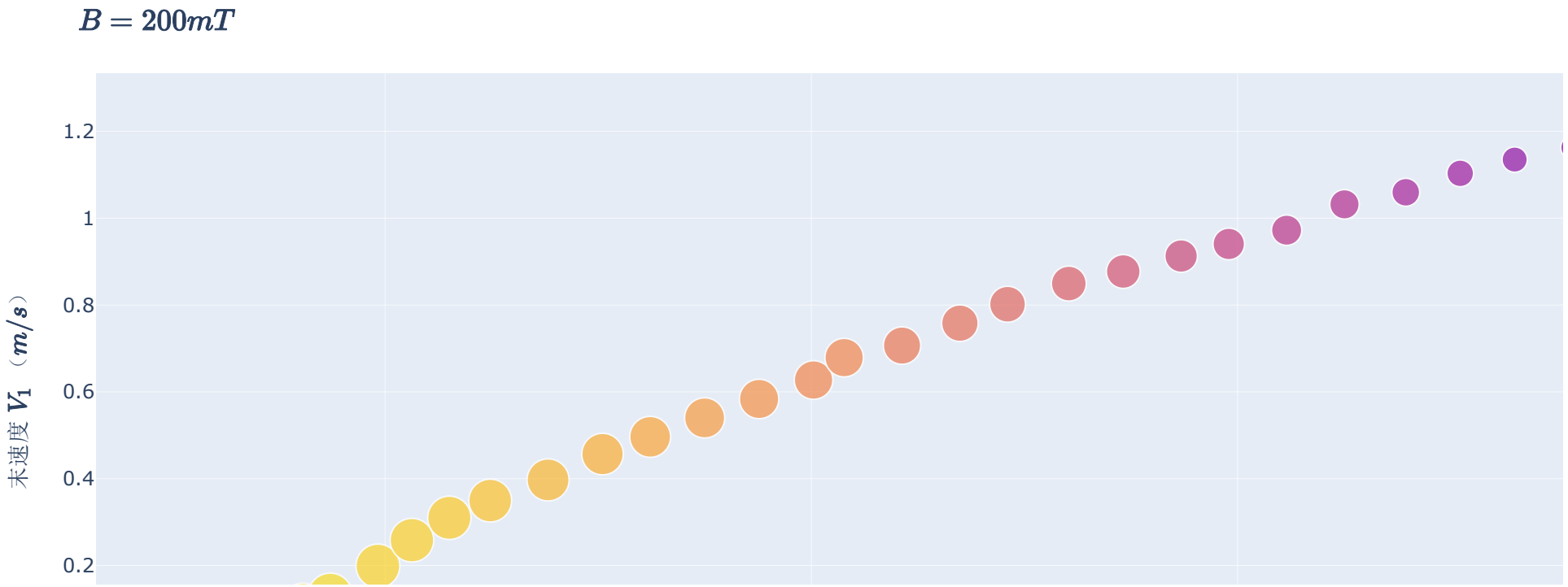


$B = 100mT$



$B = 150mT$



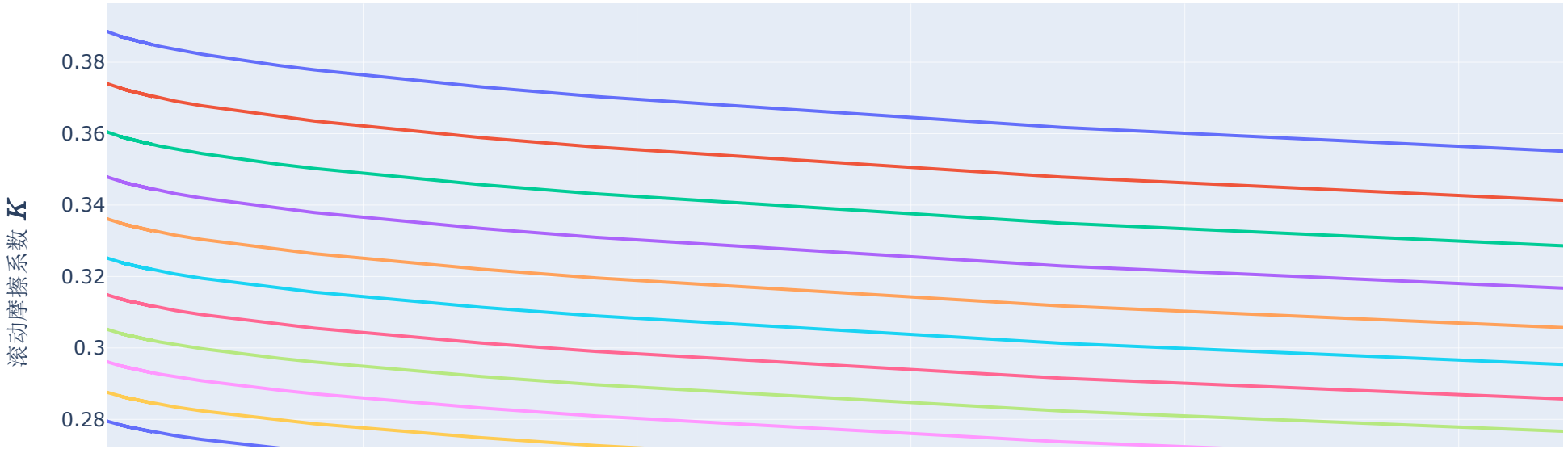


In [ ]:

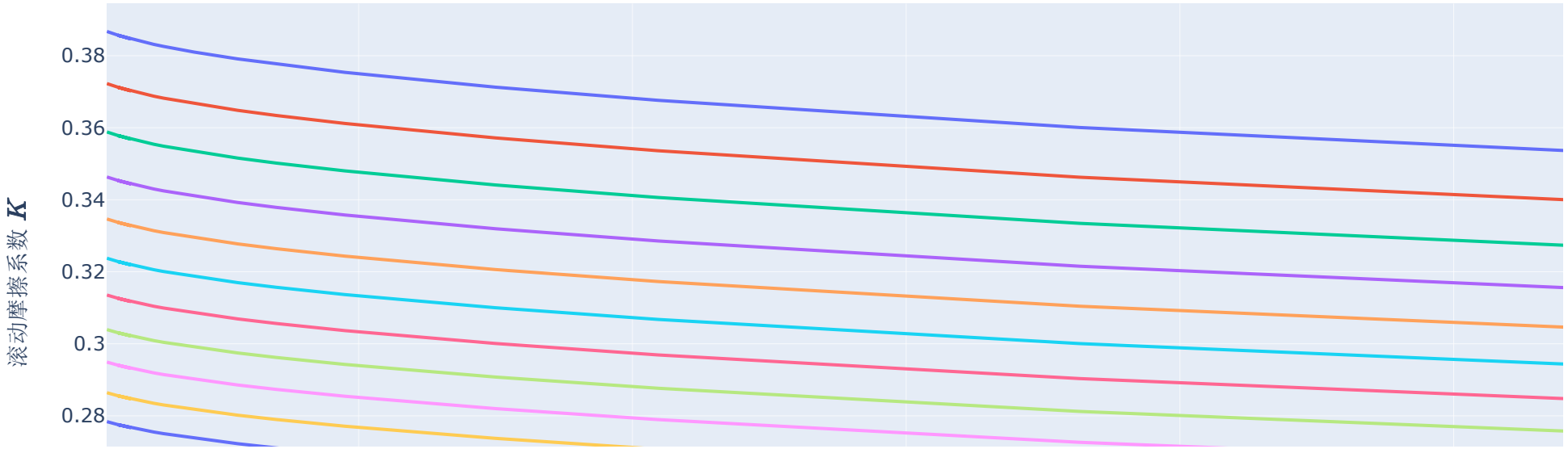
In [8]:
```
B_values = [0, 0.050, 0.100, 0.150, 0.200]  # T
sheet_names = ["0 mT", "50 mT", "100 mT", "150 mT", "200mT（测试集）", ]
B_value_str = ["0mT", "50mT", "100mT", "150mT", "200mT", ]
cols = {0: r"$初速度~V_0~（m/s）$", 1: r"$末速度~V_1~（m/s）$", 2: r"滚动摩擦<br> 系数 K", 3: "K_norm"}
titles = [r"$B = 0 mT$", r"$B = 50 mT$", r"$B = 100 mT$", r"$B = 150 mT$", r"$B = 200 mT$", ]

InteractiveShell.ast_node_interactivity = 'last_expr'

for i in range(1, 5):
    B = B_values[i]
    sheet_name = sheet_names[i]
    data_v0_v1 = pd.read_excel("A题附件1.xlsx", sheet_name=sheet_names[i]).iloc[:, :2]
```

```
    data_v02_v12 = data_v0_v1**2
    data = []
    for E in [j / 10 for j in range(50, 71, 2)]:
        K = pd.DataFrame(get_K(data_v0_v1.iloc[:, 0].values, data_v0_v1.iloc[:, 1].values, B, E=E))
        sub = pd.DataFrame(data_v02_v12.iloc[:, 0] - data_v02_v12.iloc[:, 1])
        data_xy = pd.concat([sub, K], ignore_index=True, axis=1)
        data_xy.rename(
            columns={
                0: r"$初、末速度平方之差 V_0^2 - V_1^2 (m^2 / s^2)$",
                1: r"$滚动摩擦系数 K$"},
            inplace=True,
        )
        trace = go.Scatter(
            x=data_xy.iloc[:, 0],
            y=data_xy.iloc[:, 1],
            mode='lines',
            name=fr'$E = {E}$',
        )
        data.append(trace)
    fig = go.Figure(data=data)
    fig.update_layout(
        title=titles[i],
        xaxis_title=r"$初、末速度平方之差 ~ V_0^2 - V_1^2 ~ (m^2 / s^2)$",
        yaxis_title=r"$滚动摩擦系数 ~ K$",
    )
#     fig.show()
    fig.write_image("sensitivity,B=%s.png" % B_value_str[i])
```

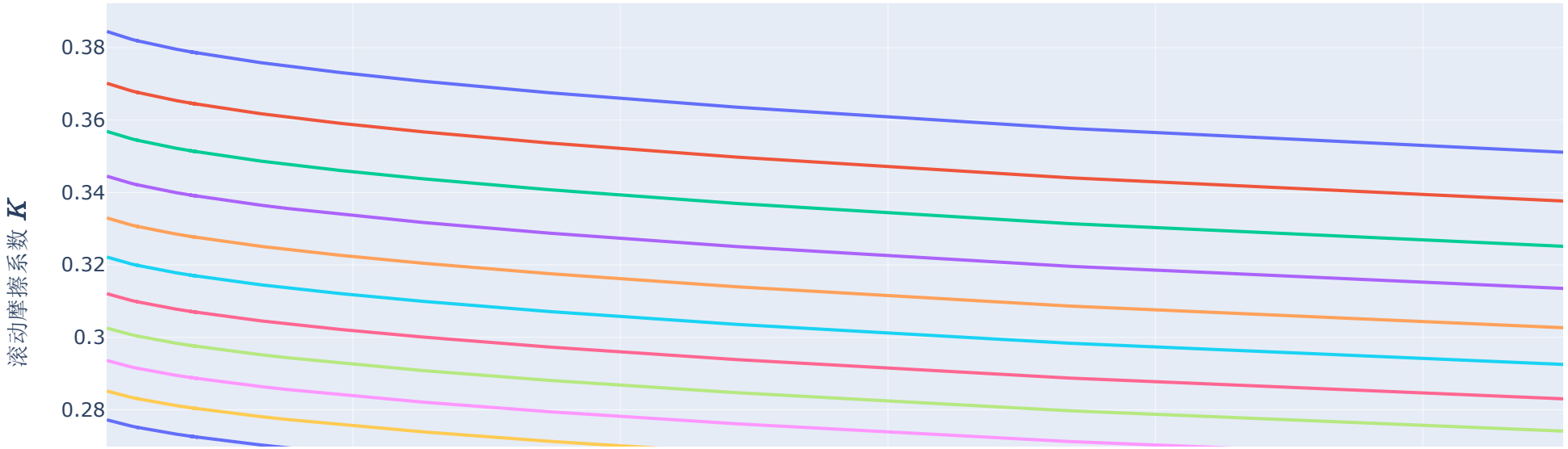
$B = 50mT$



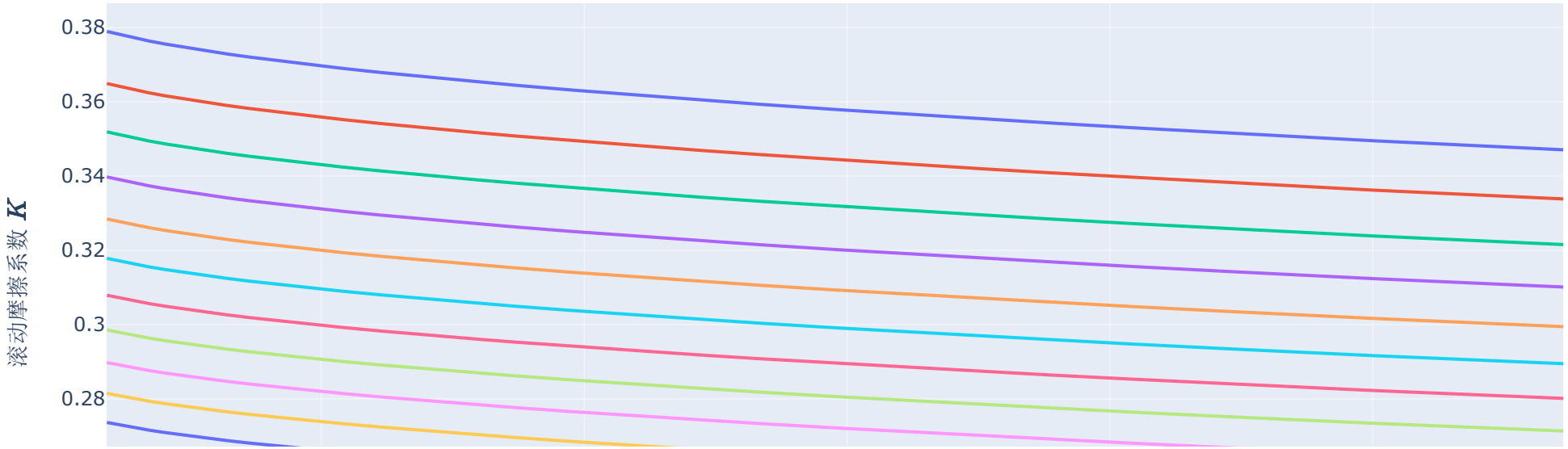
$B = 100mT$



$B = 150mT$



$B = 200mT$



In [ ]: