你上传的这篇专利文档《CN107064054A》是关于**基于偏最小二乘法的光谱回归模型构建方法**，它主要涉及：

• 使用 **PLS 回归模型**（Partial Least Squares）来对光谱数据建模；

• **选择特征波段**来降低维度（变量选择）；

• 通过**权重计算、回归系数、VIP 值（变量重要性投影）**等方法来进行光谱数据分析；

• 实现对例如 CD、CG 值等化学性质的预测。

---

✅ **我能为你做的事情：**

🔧 **实现该专利方法的核心代码，包括：**

1. **PLS 模型构建**

2. **VIP 值计算**

3. **特征波段选择**

4. **使用选中的波段进行再训练与预测**

---

✅ **PLS + VIP + 特征选择代码（专利实现）**

下面是一个完整可用的代码逻辑，实现你想要复现的专利方法：

```
import numpy as np
from sklearn.cross_decomposition import PLSRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

# 假设已有数据
# X: (n_samples, n_bands), y: (n_samples, )
# 你可以是真实数据 X_real, y_real
X = ...
y = ...

# 1. 标准化
```

```
scaler_X = StandardScaler()
scaler_y = StandardScaler()
X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y.reshape(-1, 1)).ravel()

# 2. 构建 PLS 模型
n_components = 10  # 可调
pls = PLSRegression(n_components=n_components)
pls.fit(X_scaled, y_scaled)

# 3. 计算 VIP 值 (变量重要性投影)
def calculate_vip(pls, X, y):
    t = pls.x_scores_
    w = pls.x_weights_
    q = pls.y_loadings_
    p, h = w.shape
    s = np.diag(np.dot(t.T ** 2, q.T ** 2)).reshape(h, -1)
    Wnorm2 = np.sum(w ** 2, axis=0)
    vip = np.zeros((p,))
    for i in range(p):
        weight = np.array([(w[i, j] / Wnorm2[j]) ** 2 for j in range(h)])
        vip[i] = np.sqrt(p * (s.T @ weight))
    return vip

vip_scores = calculate_vip(pls, X_scaled, y_scaled)

# 4. 选择重要波段
vip_threshold = 1.0  # 通常用 1.0 作为选择阈值
important_idx = np.where(vip_scores >= vip_threshold)[0]
print("Selected important bands:", important_idx)

# 5. 使用选中的波段重新训练模型
X_selected = X[:, important_idx]
X_selected_scaled = scaler_X.fit_transform(X_selected)
pls_selected = PLSRegression(n_components=min(5, X_selected.shape[1]))
pls_selected.fit(X_selected_scaled, y_scaled)

# 6. 预测与评估
y_pred = pls_selected.predict(X_selected_scaled)
r2 = r2_score(y_scaled, y_pred)
print("R² after VIP band selection:", r2)

# 7. 可视化 VIP
plt.figure(figsize=(12, 4))
plt.plot(vip_scores)
plt.axhline(y=vip_threshold, color='r', linestyle='--')
plt.title("VIP Scores for Spectral Bands")
plt.xlabel("Band Index")
plt.ylabel("VIP Score")
plt.grid(True)
plt.show()
```

✅ **用途与下一步扩展**

你可以在这段代码基础上：

• 用真实数据 X_real 来执行（哪怕只有 10 条）；

• 用这套 VIP 波段选取的方式，在仿真数据中筛选类似模式；

• 用这套模型结果指导仿真数据的响应对齐、偏差校正等工作。

---

**如果你希望我进一步实现：**

• 用这套方法结合“仿真数据的响应模式分析”；

• 把真实和仿真在选中波段的 slope 可视化；

• 或者打包成 Jupyter Notebook；

我都可以继续为你做，告诉我你接下来想结合哪部分使用？是否用你前面提到的仿真数据来配合 VIP 波段做 slope 分析？

你上传的专利《CN104091089A》是另一篇关于**光谱数据建模与变量选择的专利**。根据文档内容，这篇专利主要描述了：

---

✅ **专利方法核心内容（CN104091089A）**

这篇专利提出一种光谱变量选择和建模方法，包括以下几个步骤：

1. **使用主成分分析（PCA）或偏最小二乘法（PLS）进行降维分析；**

2. **基于回归系数（regression coefficient）或 VIP 值（变量重要性）选择关键波段；**

3. **重新构建简化后的建模光谱集；**

4. **使用这些关键变量进行建模和预测，例如用 PLS 回归。**

---

## ✅ 和上一篇的区别

这篇专利强调了“**变量选择 + 再建模**”的组合思路，并提出可以使用如下方式做变量选择：

• 回归系数法（PLS 回归系数）

• VIP 法（变量投影重要性）

• 一阶导数/光谱变化率分析

• 噪声分析（选择信噪比高的波段）

## ✅ 复现代码实现（完整）

下面是一个完整的 Python 实现，涵盖：

• PLS 回归建模

• 基于**回归系数**做变量选择

• 基于 **VIP 值**做变量选择

• 简化光谱建模

```
import numpy as np
from sklearn.cross_decomposition import PLSRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

# 假设你有数据 X, y （如真实光谱或仿真数据）
X = ...
y = ...

# Step 1: 标准化
scaler_X = StandardScaler()
scaler_y = StandardScaler()
X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y.reshape(-1, 1)).ravel()
```

```
# Step 2: 建立初始 PLS 模型
n_components = min(10, X.shape[1])  # 不要超过样本数
pls = PLSRegression(n_components=n_components)
pls.fit(X_scaled, y_scaled)
y_pred = pls.predict(X_scaled)
print("原始模型 R²:", r2_score(y_scaled, y_pred))

# Step 3: 提取回归系数（coefficient）
coefs = np.abs(pls.coef_[:, 0])
top_k_by_coef = np.argsort(coefs)[-50:]  # 选前 50 个变量
print("回归系数筛选的波段索引:", top_k_by_coef)

# Step 4: 计算 VIP 值
def calculate_vip(pls, X, y):
    t = pls.x_scores_
    w = pls.x_weights_
    q = pls.y_loadings_
    p, h = w.shape
    s = np.diag(np.dot(t.T ** 2, q.T ** 2)).reshape(h, -1)
    Wnorm2 = np.sum(w ** 2, axis=0)
    vip = np.zeros((p,))
    for i in range(p):
        weight = np.array([(w[i, j] / Wnorm2[j]) ** 2 for j in range(h)])
        vip[i] = np.sqrt(p * (s.T @ weight))
    return vip

vip_scores = calculate_vip(pls, X_scaled, y_scaled)
top_k_by_vip = np.argsort(vip_scores)[-50:]
print("VIP 筛选的波段索引:", top_k_by_vip)

# Step 5: 可视化 VIP / 回归系数
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(coefs)
plt.title("PLS Regression Coefficients")
plt.xlabel("Band Index")
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(vip_scores)
plt.axhline(y=1.0, color='r', linestyle='--')
plt.title("VIP Scores")
plt.xlabel("Band Index")
plt.grid(True)
plt.tight_layout()
plt.show()

# Step 6: 用选定波段重新建模
X_selected = X[:, top_k_by_vip]  # or top_k_by_coef
X_selected_scaled = scaler_X.fit_transform(X_selected)

pls_selected = PLSRegression(n_components=min(5, X_selected.shape[1]))
pls_selected.fit(X_selected_scaled, y_scaled)
y_selected_pred = pls_selected.predict(X_selected_scaled)
```

```
print("选定变量后模型 R²:", r2_score(y_scaled, y_selected_pred))
```

---

✅ **可选扩展（根据专利思路）**

• 使用**一阶导数**来计算光谱变化率做变量选择；

• 使用**样本间标准差**判断噪声区域；

• 用更复杂模型（比如 Kernel PLS）建模重要变量。

---

✅ **接下来你可以做：**

| 目标 | 推荐做法 |
| --- | --- |
| 把真实数据和仿真数据都带入这个流程 | 分别跑一下 VIP 和 coef 选择，看是否选中一致的波段 |
| 把选中的波段用于仿真数据 slope 计算 | 结合前面的方法做响应共性分析 |
| 做可视化比对：仿真 vs 真实在选中波段上的行为 | 用 line plot 或 heatmap 展示 slope 差异 |

---

如果你需要我帮你：

• 把这两篇专利的方法整合成一个统一框架；

• 实现多个样本 + 波段的 VIP/slope 可视化；

• 对仿真数据进行 slope-based 分析或筛选；

我都可以继续写完整 pipeline 或 notebook。你想我现在继续整合成一个统一 demo 吗？还是你这边先试一试？