import math

import numpy as np

import matplotlib.pyplot as plt

from scipy import optimize

import pylab

import scipy

import sympy

from scipy.optimize import fsolve

from mpl\_toolkits.mplot3d import Axes3D

from sympy import symbols, diff

'''核心计算表达式：利润=销售额-成本1-成本2

P = x \* y - C1 - C2

P = x \* y - 0.16 \* x \*\* 2 - (-100) - 0.2 \* x - 50'''

fig = plt.figure()

plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定字体 KaiTi（楷体）

plt.rcParams['axes.unicode\_minus'] = False

plt.subplot(231)

a = -0.2059264703592322

b = 13.153235286675047

x = np.linspace(25,40,1000)

y0 = a\*x+b

#设销售量与售价的关系为未知，建立P的表达式

def func(x,y):

return x \* y - 0.16 \* x \*\* 2 - (-100) - 0.2 \* x - 50

#对比原来的销售量与售价的关系对应的P

def func0(x):

return (-0.2059264703592322\*x+13.153235286675047)\*x

for yi in range(5,15):

P = func(x,yi)

plt.plot(x,P,'--')

P0 = func0(x)

plt.plot(x,P0,'b-')

plt.xlabel('价格（元）')

plt.ylabel('盈利（万元）')

'''plt.legend([])'''

#由图可知原来的关系在yi对应10--11,先粗略求出一直大于原函数保持盈利的近似临界位置

tr = np.linspace(10,11,1000)

sign = 1

for i in tr:

P = func(x,i)

k = 0

for j in range(0,1000) :

if P[j]>=P0[j] :

k+=1

if k==1000 :

print("i1 = ",i )

sign=0

if sign==0 :

break

plt.subplot(232)

ym = func(x,i)

plt.plot(x,ym,'--')

plt.plot(x,P0,'b-')

plt.xlabel('价格（元）')

plt.ylabel('盈利（万元）')

'''plt.legend([])'''

tr = np.linspace(10,11,1000)

n = 1

t = 0

temp = []

for i in tr :

P = func(x,i)

k = 0

sign = 0

for j in range(0,1000) :

if P[j]>P0[j] :

k+=1

if k>=n :

t = i

temp.append(i)

sign = 1

break

if sign==0 :

temp.append(t)

if temp==1 :

n+=1

temp1 = np.array(temp)

'''plt.subplot(223)

plt.plot(x,temp1,'b-')'''

tr = np.linspace(10.10,10.30,1000)

sign = 0

for i in tr :

P = func(x,i)

k=0

for j in range(0,1000) :

if P[j]>=P0[j] :

k+=1

if k==1 :

print("i2 = ",i)

sign = 1

break

if sign ==1 :

break

y1 = func(x,i)

plt.subplot(233)

plt.plot(x,y1,'--')

plt.plot(x,P0,'b-')

plt.xlabel('价格（元）')

plt.ylabel('盈利（万元）')

'''plt.legend([])'''

'''x0 = np.linspace(25,40,16)

plt.subplot(224)

temp = []

tr = np.linspace(10.20,10.30,1000)

args=np.polyfit([x0],y0,1)

args

yhat=np.polyval(args,x0)

for m in tr :

f6=lambda x0:x0\*(args[0]\*x+args[1])-(x0\*m-0.16\*x0\*\*2-(-100)-0.2\*x0-50)

k=fsolve(f6,[1])

if k>=25 and k<=40 :

temp.append(a)

else:

temp.append(0)

plt.plot(temp,tr,'r-')'''

'''tr = np.linspace(10,11,1000)

for i in tr :'''

#P=P0时交点，得出x与y的关系

plt.subplot(234)

def func1(x):

return (0.16+a) \* x + (-100)/x + (0.2+b) + 50/x

xl = np.linspace(25,40,1000)

yl = func1(xl)

plt.plot(xl,yl,'r-')

plt.subplot(235)

#误差分析

'''x0 = [i for i in range (25,41,1)]

y0 = a\*x+b

k1 = symbols('k1')

args=np.polyfit(x0,y0,1)

args

f1 = 25\*(k1\*25 + args[1])

diff1 = diff(f1,k1)

def f2(k2) :

return diff\*k2/(25\*(k2\*25+args[1]))

print(f2(0.2))'''

plt.subplots\_adjust(wspace=0.4,hspace=0.4)

# 子图之间间距调整subplots\_adjust(left,bottom,right,top,wspace,hspace)作用：调整子图间距。参数：left--图形到左边缘距离；bottom--图形到下边缘距离；right--图形到右边缘距离；top--图形到上边缘距离；wspace--各子图间横向间距；hspace--各子图间纵向间距；

plt.show()