DEPARTMENT OF PHYSICS

St. Xavier's College, Mumbai

SYBSC SEMESTER IV: 2024-2025

TITLE:

COMPUTING DIVERGENCE OF ARBITRARY VECTOR FIELD USING PYTHON PROGRAMING.

NAME	UID	ROLL NO.
Diksha Pandit	2302160	226
Amsha Siddiqui	2302359	235

INDEX

ABSTRACT:

The project "Computing Divergence of Arbitrary Vector Field Using Python Programing" is based on the concept of divergence. The vector (V) is taken arbitrarily. We are computing the divergence of a vector field at an arbitrarily chosen point (x,y,z) using python. These computations are verified with the help of manual calculations. Its application is based on the differential form of the Gauss law equating divergence of an electric field to charge density over electrical permittivity. The charge density (ρ) is calculated using a given values of permittivity of free space (ε_0) and divergence of electric field (∇•E) at point (x,y,z).

INTRODUCTION:

The vector taken in this project is $x^5\hat{x} + y^3\hat{y} + z^2y\hat{z}$. It is chosen such as to best demonstrate the numerical partial differentiation method used. This method would be functional on any other vector as well. We take the partial derivative of $\hat{x}$, $\hat{y}$ and $\hat{z}$ components of the vector field with respect to δx, δy and δz. These partial derivatives are added together to find the divergence of the vector field. The program used to find these partial derivatives uses the method of numerical differentiation. A function (part_derv) is defined similarly to the ∇ operator as it differentiates a multi-variable function with respect to one variable. We have used the 'central method' of differentiation in our project. Once the divergence of V has been obtained, we can use the same method to find the divergence of an electric field. Then we apply the Gauss law to find the value of charge density at point (x,y,z).

THEORY:

Divergence of a vector V:

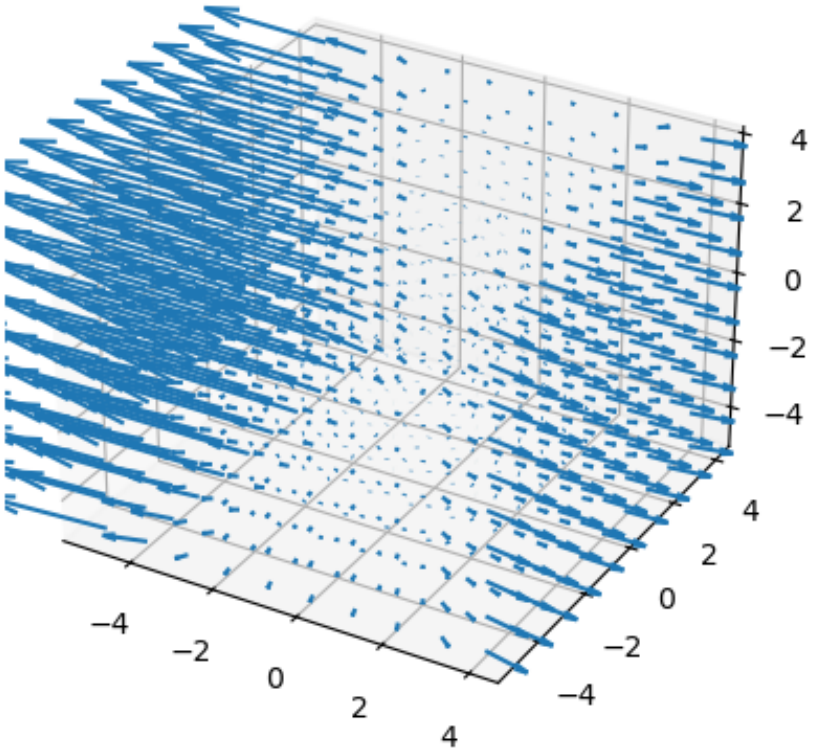
$$\nabla\bullet \mathrm{V} = \frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z}$$

Calculation of divergence analytically:

$\mathrm{V} = x^5\hat{x} + y^3\hat{y} + z^2\mathrm{y}\hat{z}$

Using the following code, we can see that the vector field has a non-zero divergence.

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
ax = fig.add_subplot(projection = '3d')

x,y,z = np.meshgrid(np.arange(-5,5,1),np.arange(-
5,5,1),np.arange(-5,5,1))

i = x**5
j = y**3
k = (z**2)*y

ax.quiver(x,y,z,i,j,k, length=0.001)

plt.show()
```



$\nabla \bullet V = 5x^4 + 3y^2 + 2zy$

At (x,y,z) = (1,2,3), $\nabla \bullet V = 5+12+12 = 29$

ALGORITHM:

1) Define a function for partial differentiation as part_derv.
2) Write components of the field vector in terms of three functions (funci, funcj, funck)
3) Define position of the (x,y,z) as x1, y1 and z1.
4) Define df_dx_i as part_derv function of funci.

   Define df_dy_j as part_derv function of funcj.

   Define df_dz_k as part_derv function of funck.
5) Print df_dx_i, df_dy_j, df_dz_k at (x1,y1,z1).
6) Define div as sum of df_dx_i, df_dy_j and df_dz_k.
7) Print the div as divergence of the vector.

PROGRAM CODE:

1) At h = 0.0001:

```
import numpy as np
def part_derv(f,x,y,z,method='central', h=1e-4):
    df_dx = (f(x+h,y,z)-f(x-h,y,z))/(2*h)
    df_dy = (f(x,y+h,z)-f(x,y-h,z))/(2*h)
    df_dz = (f(x,y,z+h)-f(x,y,z-h))/(2*h)
    return df_dx, df_dy, df_dz

def funci(x,y,z):
    return x**5

def funcj(x,y,z):
    return y**3

def funck(x,y,z):
    return (z**2)*y

x1= 1
y1= 2
z1= 3

df_dx_i, df_dy_i, df_dz_i =
part_derv(funci,x1,y1,z1,method='central',h=0.0001)
```

```
df_dx_j, df_dy_j, df_dz_j =
part_derv(funcj,x1,y1,z1,method='central',h=0.0001)
df_dx_k, df_dy_k, df_dz_k =
part_derv(funck,x1,y1,z1,method='central',h=0.0001)

print('\u03B4f/\u03B4x at (x,y,z)=',(x1,y1,z1),'is', df_dx_i)
print('\u03B4f/\u03B4y at (x,y,z)=',(x1,y1,z1),'is', df_dy_j)
print('\u03B4f/\u03B4z at (x,y,z)=',(x1,y1,z1),'is', df_dz_k)

div= df_dx_i + df_dy_j + df_dz_k
print('Divergence of Electric field=',div)
```

```
δf/δx at (x,y,z)= (1, 2, 3) is 5.000000099999458
δf/δy at (x,y,z)= (1, 2, 3) is 12.00000001000845
δf/δz at (x,y,z)= (1, 2, 3) is 12.000000000025324
Divergence of Electric field= 29.00000011003323
```

2) At h = 0.01:

```
import numpy as np
def part_derv(f,x,y,z,method='central', h=1e-4):
    df_dx = (f(x+h,y,z)-f(x-h,y,z))/(2*h)
    df_dy = (f(x,y+h,z)-f(x,y-h,z))/(2*h)
    df_dz = (f(x,y,z+h)-f(x,y,z-h))/(2*h)
    return df_dx, df_dy, df_dz

def funci(x,y,z):
    return x**5

def funcj(x,y,z):
    return y**3

def funck(x,y,z):
    return (z**2)*y

x1= 1
y1= 2
z1= 3

df_dx_i, df_dy_i, df_dz_i =
part_derv(funci,x1,y1,z1,method='central',h=0.01)
df_dx_j, df_dy_j, df_dz_j =
part_derv(funcj,x1,y1,z1,method='central',h=0.01)
```

```
df_dx_k, df_dy_k, df_dz_k =
part_derv(funck,x1,y1,z1,method='central',h=0.01)

print('\u03B4f/\u03B4x at (x,y,z)=',(x1,y1,z1),'is', df_dx_i)
print('\u03B4f/\u03B4y at (x,y,z)=',(x1,y1,z1),'is', df_dy_j)
print('\u03B4f/\u03B4z at (x,y,z)=',(x1,y1,z1),'is', df_dz_k)

div= df_dx_i + df_dy_j + df_dz_k
print('Divergence of Electric field=',div)
```

```
δf/δx at (x,y,z)= (1, 2, 3) is 5.0010000010000012
δf/δy at (x,y,z)= (1, 2, 3) is 12.000099999999847
δf/δz at (x,y,z)= (1, 2, 3) is 11.999999999999744
Divergence of Electric field= 29.001100009999604
```

3) At h = 1:

```
import numpy as np
def part_derv(f,x,y,z,method='central', h=1e-4):
    df_dx = (f(x+h,y,z)-f(x-h,y,z))/(2*h)
    df_dy = (f(x,y+h,z)-f(x,y-h,z))/(2*h)
    df_dz = (f(x,y,z+h)-f(x,y,z-h))/(2*h)
    return df_dx, df_dy, df_dz

def funci(x,y,z):
    return x**5

def funcj(x,y,z):
    return y**3

def funck(x,y,z):
    return (z**2)*y

x1= 1
y1= 2
z1= 3

df_dx_i, df_dy_i, df_dz_i =
part_derv(funci,x1,y1,z1,method='central',h=1)
df_dx_j, df_dy_j, df_dz_j =
part_derv(funcj,x1,y1,z1,method='central',h=1)
df_dx_k, df_dy_k, df_dz_k =
part_derv(funck,x1,y1,z1,method='central',h=1)
```

```
print('\u03B4f/\u03B4x at (x,y,z)=',(x1,y1,z1),'is', df_dx_i)
print('\u03B4f/\u03B4y at (x,y,z)=',(x1,y1,z1),'is', df_dy_j)
print('\u03B4f/\u03B4z at (x,y,z)=',(x1,y1,z1),'is', df_dz_k)

div= df_dx_i + df_dy_j + df_dz_k
print('Divergence of Electric field=',div)
```

```
δf/δx at (x,y,z)= (1, 2, 3) is 16.0
δf/δy at (x,y,z)= (1, 2, 3) is 13.0
δf/δz at (x,y,z)= (1, 2, 3) is 12.0
Divergence of Electric field= 41.0
```

By calculations, analytical answer = 29

ERROR ANALYSIS:

For $\frac{\partial f}{\partial x}$:

Sr. No.	Step Size	$\frac{\partial F}{\partial x}$	Analytical answer	Absolute error
1.	**0.0001**	**5.0000**	**5.0**	**0.0000**
2.	**0.01**	**5.00**	**5.0**	**0.00**
3.	**1**	**16.0**	**5.0**	**11.0**

For $\frac{\partial f}{\partial y}$:

Sr. No.	Step Size	$\frac{\partial F}{\partial y}$	Analytical answer	Absolute error
1.	**0.0001**	**12.0000**	**12.0**	**0.0000**
2.	**0.01**	**12.00**	**12.0**	**0.00**
3.	**1**	**13.0**	**12.0**	**1.0**

For $\frac{\partial f}{\partial z}$:

Sr. No.	Step Size	$\frac{\partial F}{\partial z}$	Analytical answer	Absolute error
1.	**0.0001**	**12.0000**	**12.0**	**0.0000**
2.	**0.01**	**11.99**	**12.0**	**0.01**
3.	**1**	**12.0**	**12.0**	**0.0**

For ∇•V:

Sr. No.	Step Size	$\frac{\partial F}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial F}{\partial z}$.	Analytical answer	Absolute error
1.	**0.0001**	**29.0000**	**29.0**	**0.0000**
2.	**0.01**	**29.00**	**29.0**	**0.00**
3.	**1**	**41.0**	**29.0**	**12.0**

As the value of h increases to 1, the terms with lower order show very accurate results but as the order of the terms increase, the results get very inaccurate.

Meanwhile, the lower values of h give fairly accurate results for lower and higher order terms.

APPLICATION:

We can use the above code to partially differentiate any function. Subsequently, we can use it to find divergence of an electric field such as: E = $2xy\hat{x} + z\hat{y} + z^2y\hat{z}$ at point (x,y,z) = (2,-1,3)

1)At h= 0.0001:

```
import numpy as np
def part_derv(f,x,y,z,method='central', h=1e-4):
    df_dx = (f(x+h,y,z)-f(x-h,y,z))/(2*h)
    df_dy = (f(x,y+h,z)-f(x,y-h,z))/(2*h)
    df_dz = (f(x,y,z+h)-f(x,y,z-h))/(2*h)
    return df_dx, df_dy, df_dz

def funci(x,y,z):
    return 2*x*y

def funcj(x,y,z):
    return z

def funck(x,y,z):
    return (z**2)*y

x1= 2
y1= -1
z1= 3

df_dx_i, df_dy_i, df_dz_i =
part_derv(funci,x1,y1,z1,method='central',h=0.0001)
df_dx_j, df_dy_j, df_dz_j =
part_derv(funcj,x1,y1,z1,method='central',h=0.0001)
df_dx_k, df_dy_k, df_dz_k =
part_derv(funck,x1,y1,z1,method='central',h=0.0001)

print('\u03B4f/\u03B4x at (x,y,z)=',(x1,y1,z1),'is', df_dx_i)
print('\u03B4f/\u03B4y at (x,y,z)=',(x1,y1,z1),'is', df_dy_j)
print('\u03B4f/\u03B4z at (x,y,z)=',(x1,y1,z1),'is', df_dz_k)

div= df_dx_i + df_dy_j + df_dz_k
print('Divergence of Electric field=',div)
```

```
δf/δx at (x,y,z)= (2, -1, 3) is -2.000000000002
δf/δy at (x,y,z)= (2, -1, 3) is 0.0
δf/δz at (x,y,z)= (2, -1, 3) is -6.000000000012662
Divergence of Electric field= -8.00000000001466
```

2)At h=0.01:

```
import numpy as np
def part_derv(f,x,y,z,method='central', h=1e-2):
    df_dx = (f(x+h,y,z)-f(x-h,y,z))/(2*h)
    df_dy = (f(x,y+h,z)-f(x,y-h,z))/(2*h)
    df_dz = (f(x,y,z+h)-f(x,y,z-h))/(2*h)
    return df_dx, df_dy, df_dz

def funci(x,y,z):
    return 2*x*y

def funcj(x,y,z):
    return z

def funck(x,y,z):
    return (z**2)*y

x1= 2
y1= -1
z1= 3

df_dx_i, df_dy_i, df_dz_i =
part_derv(funci,x1,y1,z1,method='central',h=0.01)
df_dx_j, df_dy_j, df_dz_j =
part_derv(funcj,x1,y1,z1,method='central',h=0.01)
df_dx_k, df_dy_k, df_dz_k =
part_derv(funck,x1,y1,z1,method='central',h=0.01)

print('\u03B4f/\u03B4x at (x,y,z)=',(x1,y1,z1),'is', df_dx_i)
print('\u03B4f/\u03B4y at (x,y,z)=',(x1,y1,z1),'is', df_dy_j)
print('\u03B4f/\u03B4z at (x,y,z)=',(x1,y1,z1),'is', df_dz_k)

div= df_dx_i + df_dy_j + df_dz_k
print('Divergence of Electric field=',div)
```

```
δf/δx at (x,y,z)= (2, -1, 3) is -1.9999999999999796
δf/δy at (x,y,z)= (2, -1, 3) is 0.0
δf/δz at (x,y,z)= (2, -1, 3) is -5.999999999999872
Divergence of Electric field= -7.999999999999852
```

3)At h=1:

```
import numpy as np
def part_derv(f,x,y,z,method='central', h=1):
    df_dx = (f(x+h,y,z)-f(x-h,y,z))/(2*h)
    df_dy = (f(x,y+h,z)-f(x,y-h,z))/(2*h)
    df_dz = (f(x,y,z+h)-f(x,y,z-h))/(2*h)
    return df_dx, df_dy, df_dz

def funci(x,y,z):
    return 2*x*y

def funcj(x,y,z):
    return z

def funck(x,y,z):
    return (z**2)*y

x1= 2
y1= -1
z1= 3

df_dx_i, df_dy_i, df_dz_i =
part_derv(funci,x1,y1,z1,method='central',h=1)
df_dx_j, df_dy_j, df_dz_j =
part_derv(funcj,x1,y1,z1,method='central',h=1)
df_dx_k, df_dy_k, df_dz_k =
part_derv(funck,x1,y1,z1,method='central',h=1)

print('\u03B4f/\u03B4x at (x,y,z)=',(x1,y1,z1),'is', df_dx_i)
print('\u03B4f/\u03B4y at (x,y,z)=',(x1,y1,z1),'is', df_dy_j)
print('\u03B4f/\u03B4z at (x,y,z)=',(x1,y1,z1),'is', df_dz_k)

div= df_dx_i + df_dy_j + df_dz_k
print('Divergence of Electric field=',div)
```

```
δf/δx at (x,y,z)= (2, -1, 3) is -2.0
δf/δy at (x,y,z)= (2, -1, 3) is 0.0
δf/δz at (x,y,z)= (2, -1, 3) is -6.0
Divergence of Electric field= -8.0
```

Gauss Law:

$\nabla \bullet E = \frac{\rho}{\varepsilon_0}$

where, E = electric field.

ρ = charge density

ε_0 = permittivity of free space

$\therefore \frac{\rho}{\varepsilon_0} = -8$

$\therefore \rho = -8*8.85*10^{-12} = 7.08*10^{-11}\ C/m^3$

CONCLUSION:

From the error analysis, we can see that the method used gets less accurate as the degree of the expression being differentiated increases. Increasing the value of h gives more accurate values for lower order terms but becomes very inaccurate as the order increases. At very small values of h, we get reasonably accurate values even for terms of higher order. Thus, we can say that the computed value of ∇•V is 29. The value of ∇•E is -8. The value of charge density is therefore $7.08*10^{-11}\ C/m^3$.

REFERENCES:

1) Edward M. Purcell and David J. Morin, Electricity and Magnetism, 3rd edition, published by Cambridge University Press, Chapter 2, Section 2.10, page no. 78.
2) Steven C. Chapra and Raymond P. Canale, Numerical Methods for Engineers, 7th edition, published by McGraw-Hill, Part 6, Chapter 23, Section 23.5, page no. 662.