12.560

Harsha-EE25BTECH11026

September 20,2025

Question

A scalar function is given by $f(x, y) = x^2 + y^2$. Take $\hat{i}$ and $\hat{j}$ as the unit vectors along the x and y axes, respectively. At $(x, y) = (3, 4)$, the direction along which f increases the fastest is

1. $\frac{1}{5}\left(4\hat{i} - 3\hat{j}\right)$
2. $\frac{1}{5}\left(3\hat{i} - 4\hat{j}\right)$
3. $\frac{1}{5}\left(3\hat{i} + 4\hat{j}\right)$
4. $\frac{1}{5}\left(4\hat{i} + 3\hat{j}\right)$

Theoretical Solution:Approach-1

The direction vector along which the function $f(x, y)$ is given by the gradient direction vector of the function, which is given by

$$\nabla f(x, y) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \tag{1}$$

$$\therefore \nabla f(x, y) = \begin{pmatrix} 2x \\ 2y \end{pmatrix} \tag{2}$$

At $(x, y) = (3, 4)$,

$$\nabla f(3, 4) = \begin{pmatrix} 6 \\ 8 \end{pmatrix} \tag{3}$$

$$\implies \text{Direction vector: } \frac{1}{5}\begin{pmatrix} 3 \\ 4 \end{pmatrix} \tag{4}$$

C Code -Finding displacement matrix

```
#include<stdio.h>

void dir_vec(double x, double y, double *grad){
        grad[0]=2*x;
        grad[1]=2*y;
}
```

Python+C code

```python
import sympy as sp
import numpy as np
import ctypes
import matplotlib.pyplot as plt

lib = ctypes.CDLL("./libmain.so")

lib.dir_vec.argtypes = (ctypes.c_double,ctypes.c_double,np.
    ctypeslib.ndpointer(dtype=np.float64, ndim=1, flags="
    C_CONTIGUOUS"))

px, py = 3, 4
grad = np.empty(2, dtype=np.float64)
lib.dir_vec(px, py, grad)
```

Python+C code

```
norm_grad = np.linalg.norm(grad)
unit_grad = grad / norm_grad
unit_vec = sp.Matrix(unit_grad)
print("Unit vector along the direction of f:")
sp.pprint(unit_vec)

xx = np.linspace(-5, 5, 200)
yy = np.linspace(-5, 5, 200)
X, Y = np.meshgrid(xx, yy)
Z = X**2 + Y**2

plt.figure(figsize=(7,6))
contours = plt.contour(X, Y, Z, levels=20, cmap="viridis")
plt.clabel(contours, inline=True, fontsize=8)
```

Python+C code

```
plt.scatter(px, py, color="red", label="Point (3,4)")

plt.quiver(px, py, grad[0], grad[1],angles="xy", scale_units="xy", scale=1, color="blue", width=0.005,label="Full f at (3,4)")

plt.quiver(px, py, unit_grad[0], unit_grad[1],angles="xy", scale_units="xy", scale=1, color="green", width=0.005,label="Unit f at (3,4)")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("Gradient and Unit Gradient at (3,4) for f(x,y) = x + y")
plt.legend()
plt.axis("equal")
plt.savefig("/home/user/Matrix Theory: workspace/Matgeo_assignments/12.560/figs/Figure_1.png")
plt.show()
```

Python code

```
import sympy as sp
import matplotlib.pyplot as plt
import numpy as np

x, y = sp.symbols('x y')
f = x**2 + y**2

grad = sp.Matrix([sp.diff(f, v) for v in (x, y)])
px, py = 3, 4
grad_val = grad.subs({x: px, y: py})
norm_val = grad_val.norm()
unit_grad = grad_val / norm_val

print("Unit vector along the direction where f grows the fastest:")
sp.pprint(unit_grad)

grad_num = np.array([float(grad_val[0]), float(grad_val[1])])
unit_grad_num = np.array([float(unit_grad[0]), float(unit_grad
```

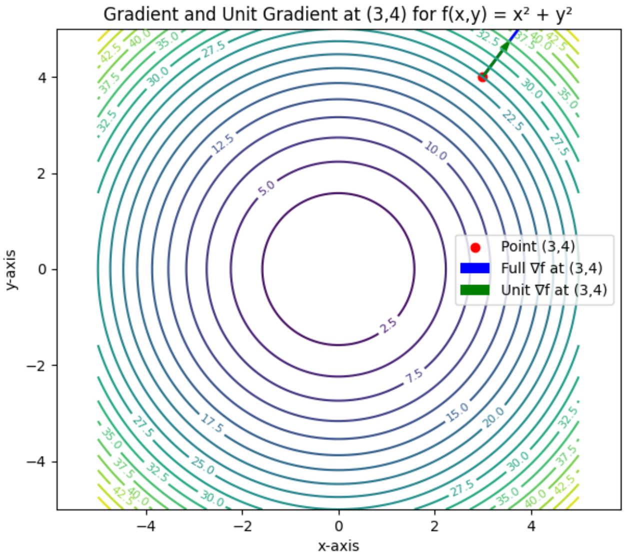
Python code

```
xx = np.linspace(-5, 5, 200)
yy = np.linspace(-5, 5, 200)
X, Y = np.meshgrid(xx, yy)
Z = X**2 + Y**2

plt.figure(figsize=(7,6))
contours = plt.contour(X, Y, Z, levels=20, cmap="viridis")
plt.clabel(contours, inline=True, fontsize=8)

plt.scatter(px, py, color="red", label="Point (3,4)")

plt.quiver(px, py, grad_num[0], grad_num[1], angles="xy",
    scale_units="xy", scale=1, color="blue", width=0.005,
label="Full f at (3,4)")
```

Python code

```
# Draw unit gradient vector
plt.quiver(px, py, unit_grad_num[0], unit_grad_num[1],
           angles="xy", scale_units="xy", scale=1, color="green",
               width=0.005,
           label="Unit f at (3,4)")

plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("Gradient and Unit Gradient at (3,4) for f(x,y) = x + y
    ")
plt.legend()
plt.axis("equal")
plt.savefig("/home/user/Matrix Theory: workspace/
    Matgeo_assignments/12.560/figs/Figure_1.png")
plt.show()
```

Plot



Gradient and Unit Gradient at (3,4) for f(x,y) = x² + y²
Point (3,4)
Full ∇f at (3,4)
Unit ∇f at (3,4)
x-axis
y-axis

As the point is given to be $\begin{pmatrix} 3 \\ 4 \end{pmatrix}$, it can be assumed that for the circle,

$$\mathbf{x}^\top \mathbf{V}\mathbf{x} = 3^2 + 4^2 = 25 \tag{5}$$

where $\mathbf{V} = \mathbf{I}$.
We can infer that the function will increse along the direction vector of normal at that point. The direction vector of normal is given by

$$\mathbf{n} = (\mathbf{V}\mathbf{q} + \mathbf{u}) \tag{6}$$

where, $\mathbf{q}$ is the point of contact.

$$\therefore \mathbf{n} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \end{pmatrix} \tag{7}$$

C Code -Finding displacement matrix

```c
#include <stdio.h>

void normal_vector(double x, double y, double *result) {
    result[0] = x;
    result[1] = y;
}
```

Python+C code

```
import ctypes
import sympy as sp
import matplotlib.pyplot as plt
import numpy as np

lib = ctypes.CDLL("./libnormal.so")

lib.normal_vector.argtypes = (ctypes.c_double, ctypes.c_double,
                              np.ctypeslib.ndpointer(dtype=np.
                                  float64, ndim=1, flags="
                                  C_CONTIGUOUS"))

x0, y0 = 3.0, 4.0
result = np.zeros(2, dtype=np.float64)
lib.normal_vector(x0, y0, result)
normal_vec = sp.Matrix(result)
print("Normal direction vector:")
sp.pprint(normal_vec)
```

Python+C code

```
theta = np.linspace(0, 2*np.pi, 400)
circle_x = 5 * np.cos(theta)
circle_y = 5 * np.sin(theta)

plt.plot(circle_x, circle_y, label='Circle: x^2+y^2=25')
plt.plot(x0, y0, 'ro', label='Point (3,4)')
plt.quiver(x0, y0, result[0], result[1], angles='xy', scale_units
    ='xy', scale=1,
           color='g', label='Normal vector')

plt.gca().set_aspect('equal', adjustable='box')
plt.axhline(0, color='k', linewidth=0.5)
plt.axvline(0, color='k', linewidth=0.5)
plt.legend()
plt.grid(True)
plt.savefig("/home/user/Matrix Theory: workspace/
    Matgeo_assignments/12.560/figs/Figure_2.png")
plt.show()
```

Python+C code

```
plt.scatter(px, py, color="red", label="Point (3,4)")

plt.quiver(px, py, grad[0], grad[1],angles="xy", scale_units="xy"
    , scale=1, color="blue", width=0.005,label="Full f at (3,4)")

plt.quiver(px, py, unit_grad[0], unit_grad[1],angles="xy",
    scale_units="xy", scale=1, color="green", width=0.005,label="
    Unit f at (3,4)")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("Gradient and Unit Gradient at (3,4) for f(x,y) = x + y
    ")
plt.legend()
plt.axis("equal")
plt.savefig("/home/user/Matrix Theory: workspace/
    Matgeo_assignments/12.560/figs/Figure_1.png")
plt.show()
```

Python code

```
import sympy as sp
import matplotlib.pyplot as plt
import numpy as np

x, y = sp.symbols('x y')
expr = x**2 + y**2 - 25

grad = sp.Matrix([sp.diff(expr, x), sp.diff(expr, y)])
q = {x: 3, y: 4}
normal_vec = grad.subs(q)
print("Normal direction vector:")
sp.pprint(normal_vec) # pretty print as column vector
```

Python code

```
# Circle parameters
theta = np.linspace(0, 2*np.pi, 400)
circle_x = 5 * np.cos(theta)
circle_y = 5 * np.sin(theta)
plt.plot(circle_x, circle_y, label='Circle: x^2+y^2=25')
plt.plot(3, 4, 'ro', label='Point (3,4)')

plt.quiver(3, 4, float(normal_vec[0]), float(normal_vec[1]),
           angles='xy', scale_units='xy', scale=1, color='g',
               label='Normal vector')

plt.gca().set_aspect('equal', adjustable='box')
plt.axhline(0, color='k', linewidth=0.5)
plt.axvline(0, color='k', linewidth=0.5)
plt.legend()
plt.grid(True)
plt.savefig("/home/user/Matrix Theory: workspace/
    Matgeo_assignments/12.560/figs/Figure_2.png")
plt.show()
```

Plot

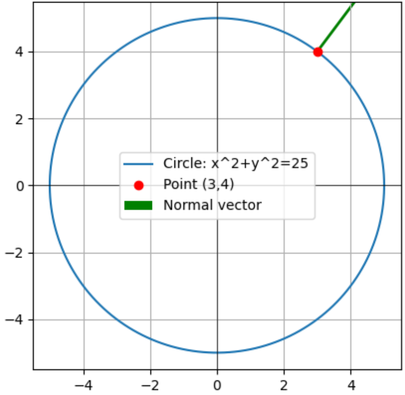


Figure: Graph for approach-2