end-term-practical

June 13, 2024

Find an equation for the tangent plane to the surface 2 + 2 = − 1 at the point (1, -3, 2).

[220]:
```
'''Importing liberaries and modules '''
import sympy as sp
from sympy import symbols, diff, pretty_print, factorial, exp
from sympy.vector import *
```

[221]:
```
#defining symbols
x, y, z = sp.symbols("x y z")
```

[222]:
```
C = CoordSys3D('')
```

[223]:
```
#creating function
f = C.z-C.x*C.z**2-C.x**2*C.y
f
```

[223]: $-\mathbf{x}^2\mathbf{y} - \mathbf{x}\mathbf{z}^2 + \mathbf{z}$

Equation of Tangent Plane $(r - a).(\nabla f_A) = 0$

[224]:
```
g=gradient(f)
g
```

[224]: $(-2\mathbf{x}\mathbf{y} - \mathbf{z}^2)\hat{\mathbf{i}} + (-\mathbf{x}^2)\hat{\mathbf{j}} + (-2\mathbf{x}\mathbf{z} + 1)\hat{\mathbf{k}}$

[225]:
```
# substitution the values
m=g.subs(C.x, 1).subs(C.y, -3).subs(C.z, 2)
m
```

[225]: $(2)\hat{\mathbf{i}} - \hat{\mathbf{j}} + (-3)\hat{\mathbf{k}}$

[226]:
```
a=C.i-3*C.j+2*C.k
a
```

[226]: $\hat{\mathbf{i}} + (-3)\hat{\mathbf{j}} + (2)\hat{\mathbf{k}}$

[227]:
```
r=C.x*C.i+C.y*C.j+C.z*C.k
r
```

[227]: $(\mathbf{x})\hat{\mathbf{i}} + (\mathbf{y})\hat{\mathbf{j}} + (\mathbf{z})\hat{\mathbf{k}}$

[228]:
```
(r-a).dot(m)
```

[228]: $2\mathbf{x} - \mathbf{y} - 3\mathbf{z} + 1$

Find the maxima and minima for the following function (,) = 3 2 − 2 + 3

[229]:
```
from sympy import symbols, diff, solve, hessian, Matrix
```

[230]:
```
# Define the function
f1 = 3*x**2 - y**2 + x**3
```

[231]:
```
# Compute the first partial derivatives
f_x = diff(f1, x)
print(f_x)
f_y = diff(f1, y)
print(f_y)
```

```
3*x**2 + 6*x
-2*y
```

[232]:
```
critical_points = solve((f_x, f_y), (x, y))
print(f"Critical points: {critical_points}")
```

```
Critical points: [(-2, 0), (0, 0)]
```

[233]:
```
# Compute the second partial derivatives
f_xx = diff(f_x, x)
f_yy = diff(f_y, y)
f_xy = diff(f_x, y)
f_yx = diff(f_y, x)

print(f"f_xx: {f_xx} , ff_yy : {f_yy} , f_xy : {f_xy} ,f_yx:{f_yx} ")
```

```
f_xx: 6*x + 6 , ff_yy : -2 , f_xy : 0 ,f_yx:0
```

[234]:
```
# creting hessian matrix
hessian_matrix = sp.Matrix(2,2,[f_xx , f_xy , f_yx ,f_yy])
hessian_matrix
```

[234]: $\begin{bmatrix} 6x+6 & 0 \\ 0 & -2 \end{bmatrix}$

[235]:
```
# Evaluate the Hessian at each critical point
import sympy
for point in critical_points:
    Hessian_point = hessian_matrix.subs({x: point[0], y: point[1]})
```

```
    r = f_xx.subs({x: point[0], y: point[1]})
    if sp.det(Hessian_point) > 0 :
        if r >0 :
               print(f"the {Hessian_point} is maxima ")
        elif r <0 :
               print(f"the {Hessian_point} is minima ")
```

```
the Matrix([[-6, 0], [0, -2]]) is minima
```

Expand in ascending power of (− 1) by using Taylor's theorem.

[236]:
```
import numpy as np
import math
x, y, pi = symbols("x y pi")

h =x-1
n = 10
x0 = (sp.solve(h,x))[0]

func = sp.log(x)
result = func.subs(x, x0)
```

[237]:
```
for i in range(1, n):
    result += diff(func, x, i).subs(x, x0) * ((x - x0)**i)/(factorial(i))
result
pretty_print(result)
```

```
           9          8          7          6          5          4          3
    (x - 1)    (x - 1)    (x - 1)    (x - 1)    (x - 1)    (x - 1)    (x - 1)
x +        -          +          -          +          -          +
       9          8          7          6          5          4          3

          2
   (x - 1)
 -        - 1
      2
```