10.4.2

Harsha-EE25BTECH11026

September 11,2025

Question

Find the equations of tangent and normal to the curve $y = \frac{x-7}{(x-2)(x-3)}$ at the point where it cuts the X axis.

Theoretical Solution

It is given that the tangents and normal are drawn at the point where the curve intersects the X axis i.e, at $\mathbf{P} = \begin{pmatrix} 7 \\ 0 \end{pmatrix}$ Let the equations of tangent be,

$$\mathbf{n_1}^\top \mathbf{x} = c_1 \tag{1}$$

And equation of normal be,

$$\mathbf{n_2}^\top \mathbf{x} = c_2 \tag{2}$$

To solve for their direction vectors, we can use he Jacobian technique.

Theoretical Solution

It states that,

$$\mathbf{n} = \begin{pmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \end{pmatrix}_{\text{at point of tangency}} \tag{3}$$

where,

$$\mathbf{n} : \text{Direction vector of tangent}$$
$$\mathbf{F(x, y)} : \text{Function of the curve}$$

Equation of the curve can be modified and written as,

$$yx^2 - 5xy + 6y - x + 7 = 0 \tag{4}$$

$$\implies \mathbf{n} = \begin{pmatrix} 2xy - 5y - 1 \\ x^2 - 5x + 6 \end{pmatrix} \tag{5}$$

Theoretical Solution

Substituting the point $\mathbf{P}$,

$$\therefore \mathbf{n_1} = \begin{pmatrix} -1 \\ 20 \end{pmatrix} \tag{6}$$

$$\implies \mathbf{n_2} = \begin{pmatrix} 20 \\ 1 \end{pmatrix} \tag{7}$$

Substituing (6) and (7) in (1) and (2) respectively yields,

$$\begin{pmatrix} -1 & 20 \end{pmatrix} \mathbf{x} = c_1 \tag{8}$$

$$\begin{pmatrix} 20 & 1 \end{pmatrix} \mathbf{x} = c_2 \tag{9}$$

Substituting $\mathbf{P}$ in (8) and (9),

$$\implies c_1 = -7 \quad and \quad c_2 = 140 \tag{10}$$

Theoretical Solution

$$\therefore \text{Equation of tangent: } \begin{pmatrix} -1 & 20 \end{pmatrix} \mathbf{x} = -7 \quad (11)$$

$$\therefore \text{Equation of normal: } \begin{pmatrix} 20 & 1 \end{pmatrix} \mathbf{x} = 140 \quad (12)$$

C Code -Finding the intersection of conics

```c
#include <stdio.h>
#include <math.h>

// Function to evaluate the curve y = (x-7)/((x-2)(x-3))
double curve(double x) {
    double denom = (x - 2.0) * (x - 3.0);
    if (fabs(denom) < 1e-9) {
        return NAN; // return NaN at asymptotes
    }
    return (x - 7.0) / denom;
}

// Function to evaluate tangent line at (7,0)
double tangent(double x) {
    double m_tan = 1.0 / 20.0; // slope of tangent
    return m_tan * (x - 7.0);
}
```

C Code -Finding the intersection of conics

```c
// Function to evaluate normal line at (7,0)
double normal(double x) {
    double m_norm = -20.0; // slope of normal
    return m_norm * (x - 7.0);
}

// Function to print tangent & normal equations
void print_equations() {
    double m_tan = 1.0 / 20.0;
    double m_norm = -20.0;

    printf("Tangent equation: y = %.1f*(x - 7)\n", m_tan);
    printf("Normal equation: y = %.1f*(x - 7)\n", m_norm);
}
```

Python+C code

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

# Load C library
lib = ctypes.CDLL("./libcurve_solver.so")

# Function signatures
lib.curve.restype = ctypes.c_double
lib.curve.argtypes = [ctypes.c_double]
lib.tangent.restype = ctypes.c_double
lib.tangent.argtypes = [ctypes.c_double]
lib.normal.restype = ctypes.c_double
lib.normal.argtypes = [ctypes.c_double]
lib.print_equations.restype = None
```

Python+C code

```python
# Print equations once
lib.print_equations()

# Define helper wrappers for numpy arrays
def curve(x):
    return np.array([lib.curve(float(val)) for val in x])

def tangent(x):
    return np.array([lib.tangent(float(val)) for val in x])

def normal(x):
    return np.array([lib.normal(float(val)) for val in x])

# Split domain around asymptotes
x_vals1 = np.linspace(-2, 1.9, 200)
x_vals2 = np.linspace(2.1, 2.9, 200)
x_vals3 = np.linspace(3.1, 12, 200)
```

Python+C code

```
y_vals1 = curve(x_vals1)
y_vals2 = curve(x_vals2)
y_vals3 = curve(x_vals3)

# Tangent & normal
x0, y0 = 7, 0
x_line = np.linspace(4, 10, 200)
y_tan = tangent(x_line)
y_norm = normal(x_line)

plt.figure(figsize=(8,6))
plt.plot(x_vals1, y_vals1, 'b')
plt.plot(x_vals2, y_vals2, 'b')
```

Python+C code

```
plt.plot(x_vals3, y_vals3, 'b', label="Curve y=(x-7)/((x-2)(x-3))
    ")
plt.plot(x_line, y_tan, 'r--', label="Tangent at (7,0)")
plt.plot(x_line, y_norm, 'g--', label="Normal at (7,0)")
plt.scatter([x0], [y0], color='k', zorder=5)
plt.text(x0+0.2, y0+0.2, "(7,0)")
plt.axhline(0, color='gray', lw=1)
plt.axvline(0, color='gray', lw=1)
plt.ylim(-2, 2)
plt.xlim(-2, 12)
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.title("Curve with Tangent and Normal at (7,0)")
plt.grid(True)
plt.savefig("/home/user/Matrix Theory: workspace/
    Matgeo_assignments/10.4.2/figs/figure_1.png")
plt.show()
```

Python code

```
import numpy as np
import matplotlib.pyplot as plt
# Define the curve y = (x-7)/((x-2)(x-3))
def curve(x):
    denom = (x - 2) * (x - 3)
    mask = np.isclose(denom, 0) # True where denominator ~ 0
    y = (x - 7) / denom
    y[mask] = np.nan
    return y

x0, y0 = 7, 0
def grad_F(x, y):
    Fx = 2 * x * y - 5 * y - 1
    Fy = x**2 - 5*x + 6
    return np.array([Fx, Fy])

# Evaluate gradient at (7,0)
Fx, Fy = grad_F(x0, y0)
```

Python code

```
def tangent(x):
    return y0 + m_tan * (x - x0)

# Normal line: slope = -1/m_tan (perpendicular)
m_norm = -1 / m_tan
def normal(x):
    return y0 + m_norm * (x - x0)

# Print equations
def print_equations():
    print(f"Tangent equation: y = {m_tan:.1f}*(x - {x0}) + {y0}")
    print(f"Normal equation: y = {m_norm:.1f}*(x - {x0}) + {y0}")

print_equations()
```

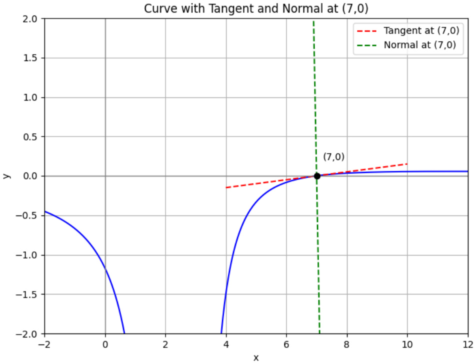
Python code

```
# Plot
x_vals1 = np.linspace(-2, 1.9, 200)
x_vals2 = np.linspace(2.1, 2.9, 200)
x_vals3 = np.linspace(3.1, 12, 200)
y_vals1 = curve(x_vals1)
y_vals2 = curve(x_vals2)
y_vals3 = curve(x_vals3)

plt.figure(figsize=(8,6))
# Curve
plt.plot(x_vals1, y_vals1, 'b')
plt.plot(x_vals2, y_vals2, 'b')
plt.plot(x_vals3, y_vals3, 'b')
# Tangent and normal (extended around x0)
x_line = np.linspace(4, 10, 200)
plt.plot(x_line, tangent(x_line), 'r--', label="Tangent at (7,0)"
    )
plt.plot(x_line, normal(x_line), 'g--', label="Normal at (7,0)")
```

Python code

```
# Mark the point of tangency
plt.scatter([x0], [y0], color='k', zorder=5)
plt.text(x0+0.2, y0+0.2, "(7,0)")

# Axes and formatting
plt.axhline(0, color='gray', lw=1)
plt.axvline(0, color='gray', lw=1)
plt.ylim(-2, 2)
plt.xlim(-2, 12)
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.title("Curve with Tangent and Normal at (7,0)")
plt.grid(True)
plt.savefig("/home/user/Matrix Theory: workspace/
    Matgeo_assignments/10.4.2/figs/Figure_1.png")
plt.show()
```

Plot



Curve with Tangent and Normal at (7,0)
Tangent at (7,0)
Normal at (7,0)
(7,0)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
−2
0
2
4
6
8
10
12
x