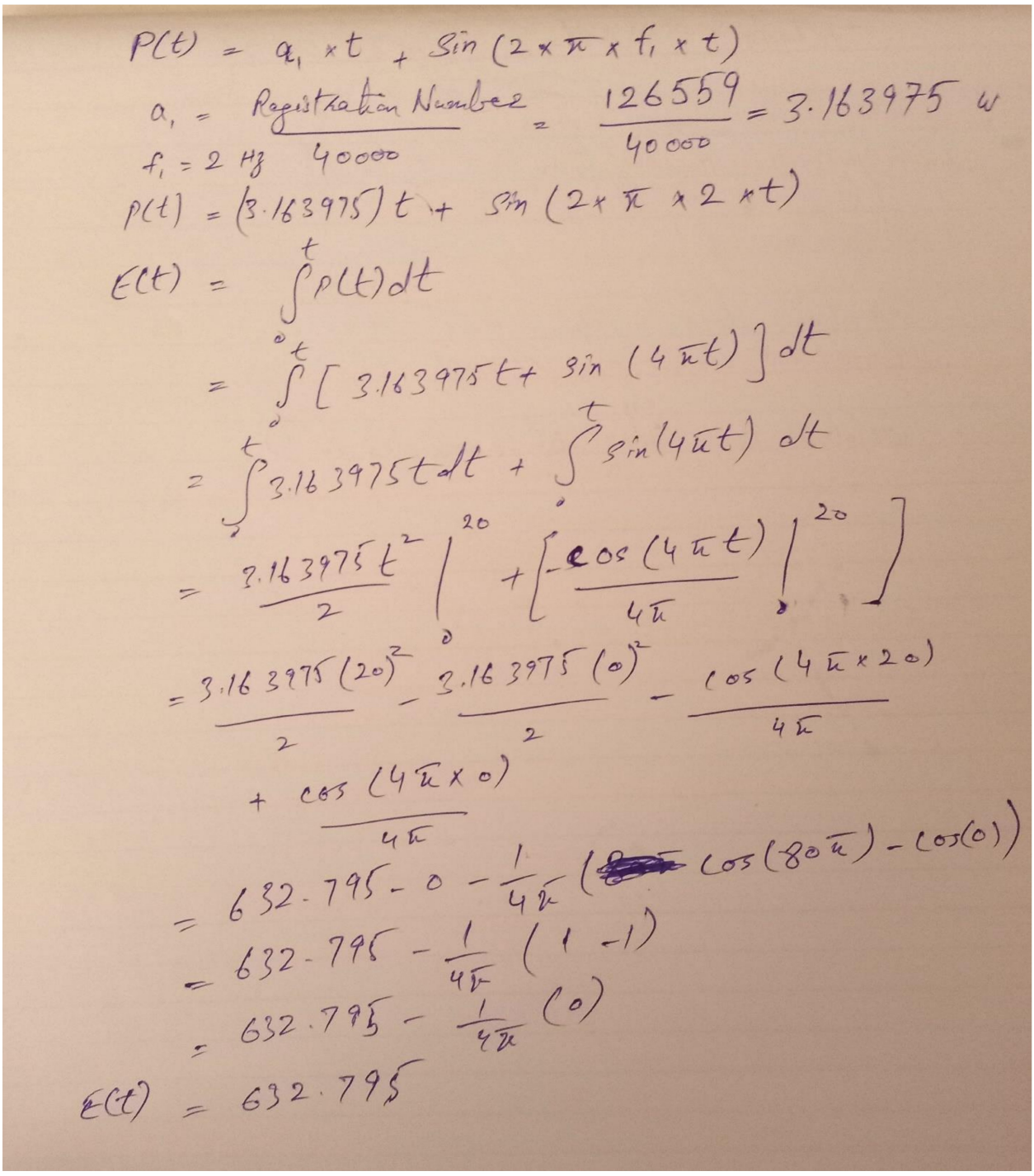
Exact Solution:



$$P(t) = a_1 \times t + \sin(2 \times \pi \times f_1 \times t)$$

$$a_1 = \frac{\text{Registration Number}}{40000} = \frac{126559}{40000} = 3.163975 \text{ w}$$

$$f_1 = 2 \text{ Hz}$$

$$P(t) = (3.163975)t + \sin(2 \times \pi \times 2 \times t)$$

$$E(t) = \int_0^t P(t)\,dt$$

$$= \int_0^t [3.163975t + \sin(4\pi t)]\,dt$$

$$= \int_0^t 3.163975t\,dt + \int_0^t \sin(4\pi t)\,dt$$

$$= \frac{3.163975t^2}{2}\Big|_0^{20} + \left[\frac{-\cos(4\pi t)}{4\pi}\Big|_0^{20}\right]$$

$$= \frac{3.163975(20)^2}{2} - \frac{3.163975(0)^2}{2} - \frac{\cos(4\pi \times 20)}{4\pi} + \frac{\cos(4\pi \times 0)}{4\pi}$$

$$= 632.795 - 0 - \frac{1}{4\pi}(\cos(80\pi) - \cos(0))$$

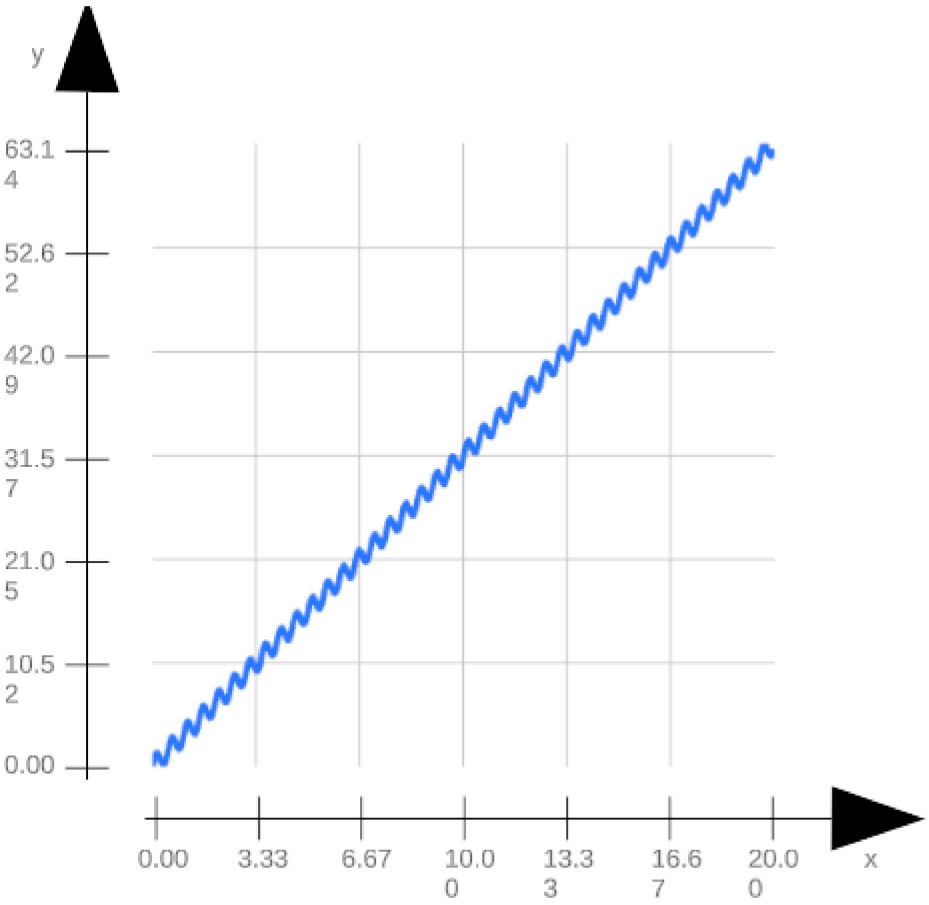
$$= 632.795 - \frac{1}{4\pi}(1 - 1)$$

$$= 632.795 - \frac{1}{4\pi}(0)$$

$$E(t) = 632.795$$

Function Plot:

Script: PlotFunction.cs


y
63.14
52.62
42.09
31.57
21.05
10.52
0.00
0.00
3.33
6.67
10.00
13.33
16.67
20.00
x

Nassi-Schneiderman diagrams:

Method1:

```
function AreaUnderCurve_ConstantHeight(numSteps) {
    E: deltaT = (t2 - t1) /numSteps
    E: area = 0
    i < numSteps
        E: x0 = t1 + i(deltaT)
        E: y0 = Get_Y_Value(x0)
        E: x1 = x0 + deltaT
        E: area += y0 * deltaT
        E: p1 = new Vector2(x0, 0)
        E: p2 = new Vector2(x0, y0)
        E: p3 = new Vector2(x1, y0)
        E: p4 = new Vector2(x1, 0)
        E: plotter.AddLine(p1, p2)
        E: plotter.AddLine(p2, p3)
        E: plotter.AddLine(p3, p4)
        E: plotter.AddLine(p4, p1)
    A: area
}
```

Method2:

```
function AreaUnderCurve_MeanValue(numSteps) {
    E: deltaT = (t2 - t1) /numSteps
    E: area = 0
    i < numSteps
        E: x0 = t1 + i(deltaT)
        E: y0 = Get_Y_Value(x0)
        E: x1 = x0 + deltaT
        E: y1 = Get_Y_Value(x1)
        E: meanValue = (y0 + y1) / 2
        E: area += meanValue * deltaT
        E: p1 = new Vector2(x0, 0)
        E: p2 = new Vector2(x0, meanValue)
        E: p3 = new Vector2(x1, meanValue)
        E: p4 = new Vector2(x1, 0)
        E: plotter.AddLine(p1, p2)
        E: plotter.AddLine(p2, p3)
        E: plotter.AddLine(p3, p4)
        E: plotter.AddLine(p4, p1)
    A: area
}
```

Method3:

```
function AreaUnderCurve_LinearInterpolation(numSteps) {
    E: deltaT = (t2 - t1) /numSteps
    E: area = 0
    i < numSteps
        E: x0 = t1 + i(deltaT)
        E: y0 = Get_Y_Value(x0)
        E: x1 = x0 + deltaT
        E: y1 = Get_Y_Value(x1)
        E:  length = (y0 + y1) / 2
        E: area += length * deltaT
        E: p1 = new Vector2(x0, 0)
        E: p2 = new Vector2(x0, y0)
        E: p3 = new Vector2(x1, y1)
        E: p4 = new Vector2(x1, 0)
        E: plotter.AddLine(p1, p2)
        E: plotter.AddLine(p2, p3)
        E: plotter.AddLine(p3, p4)
        E: plotter.AddLine(p4, p1)
    A: area
}
```

UML Diagram:

HomeWork01Class
+ trueValue: double + numSteps: integer - registrationNumber: double - a: double - f: double - t1: double - t2: double - x0: double - x1: double - y0: double - y1: double - deltaT: double - area: double - functionString: string - gameObject: GameObject - plotter: FunctionPlotter - p1: Vector2 - p2: Vector2 - p3: Vector2 - p4: Vector2
+ HomeWork01Class (registrationNumber: double, f: double, t1: double, t2: double, gameObject: GameObject) + PlotFunction (): void + Get_Y_Value (x: double): double + AreaUnderCurve_ConstantHeight (numSteps: integer): double + AreaUnderCurve_MeanValue (numSteps: integer): double + AreaUnderCurve_LinearInterpolation (numSteps: integer): double + AbsoluteError (trueValue: double, calculatedValue: double): double + RelativeError (trueValue: double, calculatedValue: double): double

Results of the numerical integration:

Script: AllMethods.cs

Note: The numerical integration was performed for 200 Steps.

Area for Method 1 : 629.631026860076

Area for Method 2 : 632.795002400502

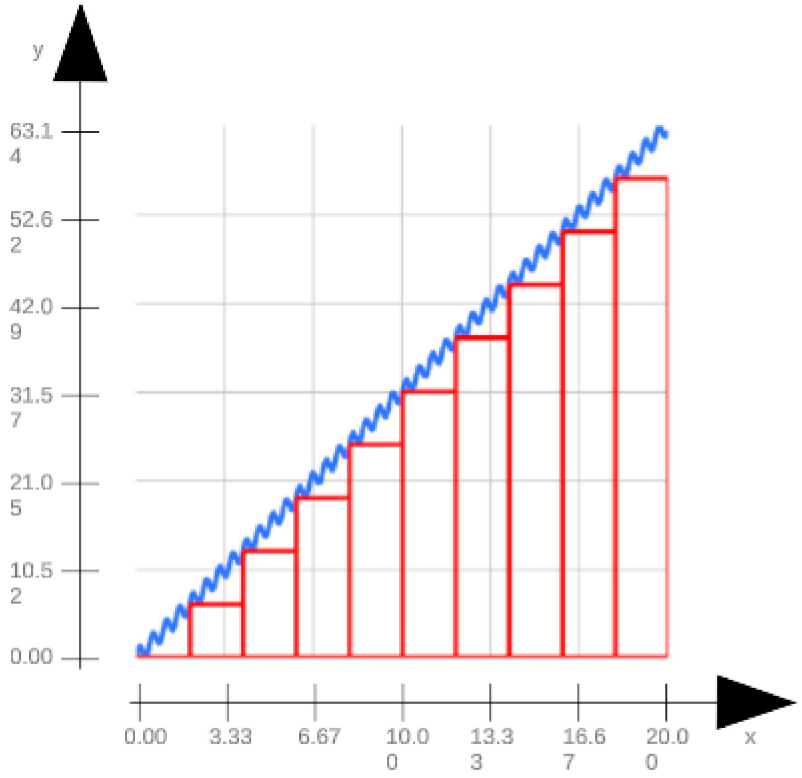
Area for Method 3 : 632.795002400502

Visualization of the integrals

Method 01 Visualization

Note: Step size was reduced to 10 to get better visualization

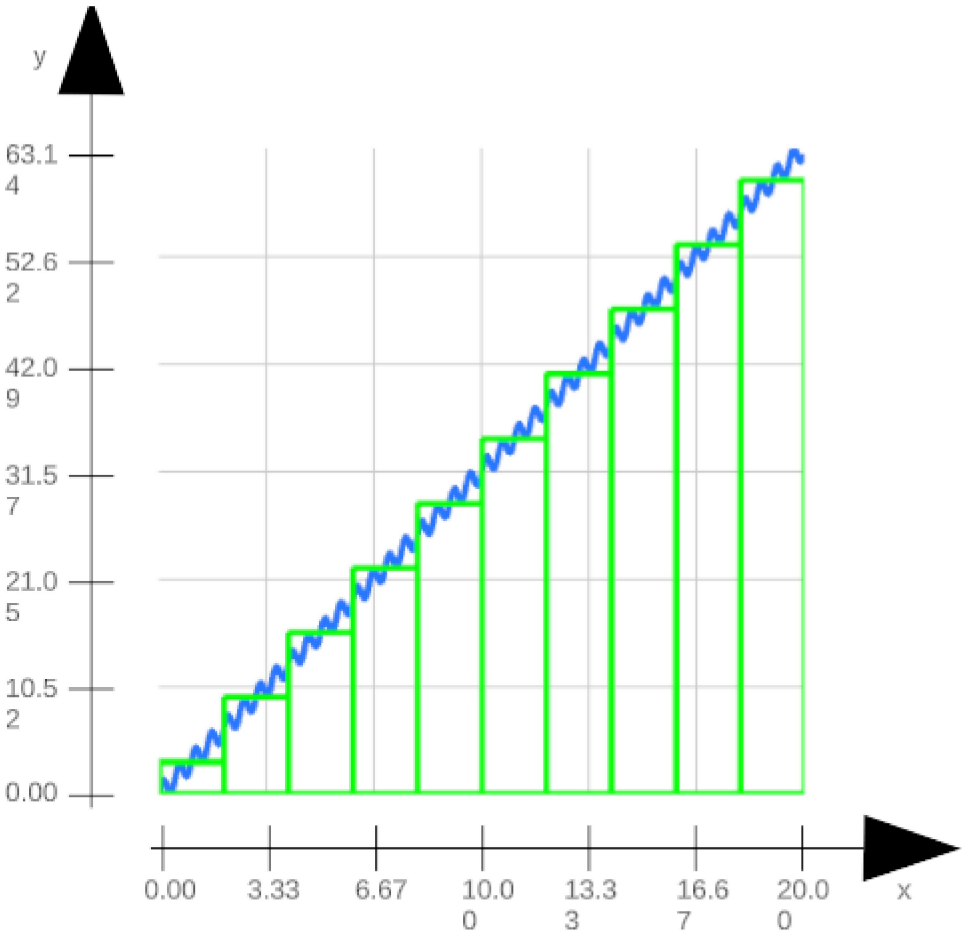
Script: Method01.cs



Method 2 Visualization

Note: Step size was reduced to 10 to get better visualization

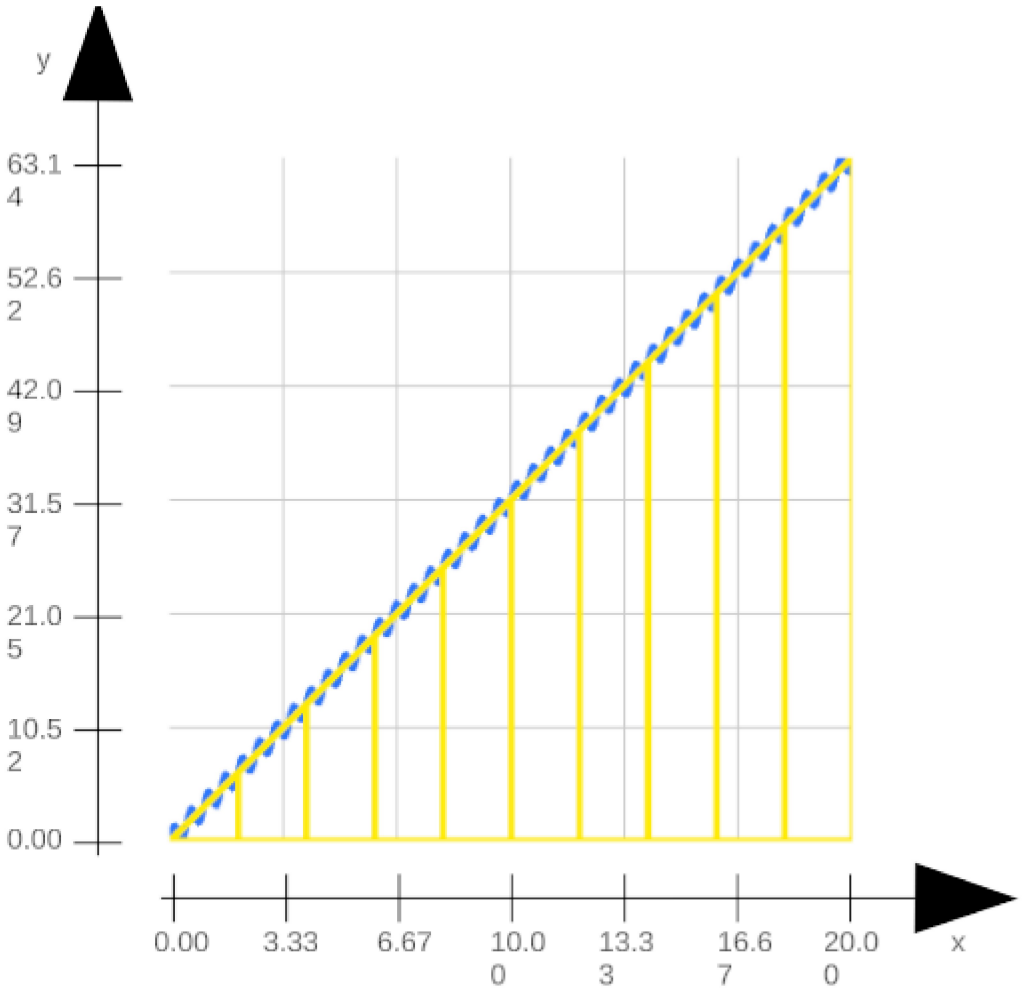
Script: Method02.cs



Method 3 Visualization

Note: Step size was reduced to 10 to get better visualization

Script: Method03.cs



Absolute and relative error of the numerical results

Script: AllMethods.cs

Note: The numerical integration was performed for 200 Steps.

Absolute Error for Method 1 : 3.16397313992445

Relative Error for Method 1 : 0.499999706054007

Absolute Error for Method 2 : 2.40050201227859E-06

Relative Error for Method 2 : 3.79349080235873E-07

Absolute Error for Method 3 : 2.40050201227859E-06

Relative Error for Method 3 : 3.79349080235873E-07