Euler's Technique

Kaylin Shanahan 2023

This is a python programme which solves a differential equation, uses Euler's technique to determine the value of the function at a value of x and displays the solution on a graph.

The slope of a function is described by the differential equation:

$$\frac{dy}{dx} = 4x + 3$$

The initial value of the function is $y = 1$ when $x = 0$:

$$y(0) = 1$$

Using Euler's Technique with an initial step value of $\Delta x = 0.5$, we can determine the value of the function (y) at $x = 2$.

- Input the initial conditions
- Determine the number of steps required
- Initialise x and y values
- Calculate the slope using the differential equation above
- Output the x and y values
- Output a plot of the numerical and analytical solutions on a graph

In [24]:
```
# Solve the differential equation dy/dx = 4x + 3, with initial condition y(0) = 1
# Use Euler's technique to determine y(2)

import numpy as np

# Set the initial conditions
x_init = 0
y_init = 1
```

```python
# Determine the number of steps required
x_max = 2
delta_x = 0.5
N = int((x_max - x_init)/delta_x)

# Set up NumPy arrays to hold the x and y values and initialise
X = np.zeros(N + 1)
Y = np.zeros(N + 1)
X[0] = x_init
Y[0] = y_init
```

In [25]:

```python
# Use a for loop to step along the x-axis
for i in range(N):
    slope = 4 * X[i] + 3              # Calculate the slope at the start of interval
    Y[i+1] = Y[i] + slope * delta_x   # Estimate y at the end of the interval
    X[i+1] = X[i] + delta_x           # Calculate x at the end of the interval

# Print the x and y values
for i in range(N+1):
    print("i ={0:3}, x ={1:6.3f}, y ={2:6.3f}".format(i, X[i], Y[i]))
```

```
i =  0, x = 0.000, y = 1.000
i =  1, x = 0.500, y = 2.500
i =  2, x = 1.000, y = 5.000
i =  3, x = 1.500, y = 8.500
i =  4, x = 2.000, y =13.000
```

In [26]:

```python
import matplotlib.pyplot as plt

# Plot the numerical solution as points
plt.plot(X, Y, "ro")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()

# Plot the analytical solution as a line
X_101 = np.linspace(0, 2, 101)
Y_analytic = (2) * X_101 ** 2 + (3) * X_101 + 1
plt.plot(X_101, Y_analytic)
plt.show()
```

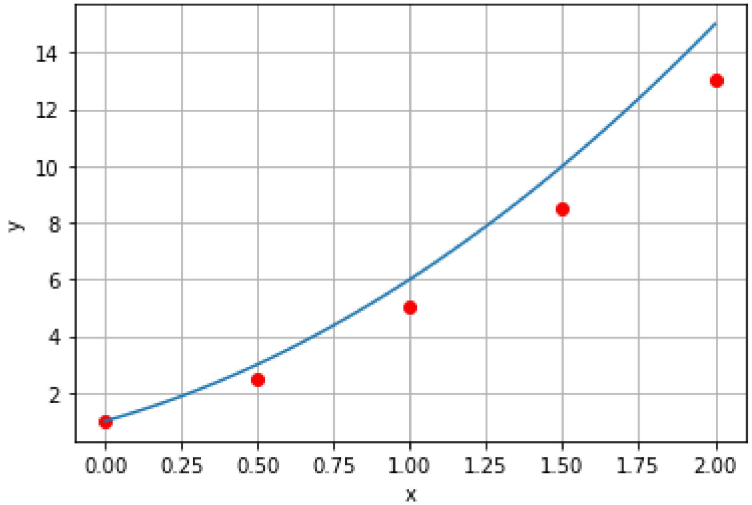

Table of Δx and $y(2)$ Values

Δx	$Y(2)$
0.5	13.000
0.05	14.800
0.005	14.980