**Introduction**

This report discusses the implementation and evaluation of numerical methods for root finding and integration. For Task 1, the Bisection and Newton-Raphson methods were implemented to find the roots of two given functions. These results were compared with the roots obtained using scipy.optimize.root(). For Task 2, the Midpoint Approximation and Trapezium Rule were used for numerical integration, with results compared to the analytical solutions.

**Task 1: Root Finding**

**Function 1: f(x)=x2−x−1f(x) = x^2 - x - 1f(x)=x2−x−1**

* **Bisection Method**:
  + Root 1: 1.618034361.618034361.61803436, Iterations: 20
  + Root 2: −0.61803436-0.61803436−0.61803436, Iterations: 20
* **Newton-Raphson Method**:
  + Root 1: 1.618033991.618033991.61803399, Iterations: 5
  + Root 2: −0.61803399-0.61803399−0.61803399, Iterations: 6
* **Scipy Method**:
  + Root 1: 1.618033991.618033991.61803399
  + Root 2: −0.61803399-0.61803399−0.61803399

**Function 2: f(x)=x3−x2−2x+1f(x) = x^3 - x^2 - 2x + 1f(x)=x3−x2−2x+1**

* **Bisection Method**:
  + Root 1: −1.24697971-1.24697971−1.24697971, Interval: [−2,−1][-2, -1][−2,−1], Iterations: 21
  + Root 2: 0.445041660.445041660.44504166, Interval: [0,1][0, 1][0,1], Iterations: 21
  + Root 3: 1.801938061.801938061.80193806, Interval: [1,2][1, 2][1,2], Iterations: 21
* **Newton-Raphson Method**:
  + Root 1: −1.24697960-1.24697960−1.24697960, Initial Guess: −1-1−1, Iterations: 5
  + Root 2: 0.445041820.445041820.44504182, Initial Guess: 111, Iterations: 4
  + Root 3: 1.801937741.801937741.80193774, Initial Guess: 2.52.52.5, Iterations: 4
* **Scipy Method**:
  + Root 1: −1.24697960-1.24697960−1.24697960
  + Root 2: 0.445041870.445041870.44504187
  + Root 3: 1.801937741.801937741.80193774

**Task 2: Numerical Integration**

**Function 1: f(x)=xx2+1f(x) = \frac{x}{x^2 + 1}f(x)=x2+1x​ over [0, 5]**

* **Midpoint Approximation**:
  + N=10N=10N=10: 1.64041.64041.6404
  + N=30N=30N=30: 1.63031.63031.6303
  + N=50N=50N=50: 1.62951.62951.6295
  + N=100N=100N=100: 1.62921.62921.6292
  + N=500N=500N=500: 1.62911.62911.6291
* **Trapezium Rule**:
  + N=10N=10N=10: 1.60691.60691.6069
  + N=30N=30N=30: 1.62661.62661.6266
  + N=50N=50N=50: 1.62821.62821.6282
  + N=100N=100N=100: 1.62881.62881.6288
  + N=500N=500N=500: 1.62901.62901.6290
* **Analytical Result**: 1.62901.62901.6290
* **Error**:
  + Midpoint Approximation (N=500): 4×10−64 \times 10^{-6}4×10−6
  + Trapezium Rule (N=500): 9×10−69 \times 10^{-6}9×10−6

**Function 2: f(x)=exf(x) = e^xf(x)=ex over [0, 5]**

* **Midpoint Approximation**:
  + N=10N=10N=10: 145.8887145.8887145.8887
  + N=30N=30N=30: 147.2427147.2427147.2427
  + N=50N=50N=50: 147.3518147.3518147.3518
  + N=100N=100N=100: 147.3978147.3978147.3978
  + N=500N=500N=500: 147.4125147.4125147.4125
* **Trapezium Rule**:
  + N=10N=10N=10: 150.4715150.4715150.4715
  + N=30N=30N=30: 147.7542147.7542147.7542
  + N=50N=50N=50: 147.5360147.5360147.5360
  + N=100N=100N=100: 147.4439147.4439147.4439
  + N=500N=500N=500: 147.4144147.4144147.4144
* **Analytical Result**: 147.4132147.4132147.4132
* **Error**:
  + Midpoint Approximation (N=500): 6×10−46 \times 10^{-4}6×10−4
  + Trapezium Rule (N=500): 1.2×10−31.2 \times 10^{-3}1.2×10−3

**Conclusion**

The implemented methods closely matched the reference results, with minor deviations that diminished as the number of iterations increased. The comparison shows that the Newton-Raphson method converges faster than the Bisection method, and the Midpoint Approximation offers a slightly better accuracy than the Trapezium Rule for the given functions.

**Graphs**

* **Task 1**: [Include the graph of f(x)=x2−x−1f(x) = x^2 - x - 1f(x)=x2−x−1 and f(x)=x3−x2−2x+1f(x) = x^3 - x^2 - 2x + 1f(x)=x3−x2−2x+1]
* **Task 2**: [Include the graph of f(x)=xx2+1f(x) = \frac{x}{x^2 + 1}f(x)=x2+1x​ and f(x)=exf(x) = e^xf(x)=ex]