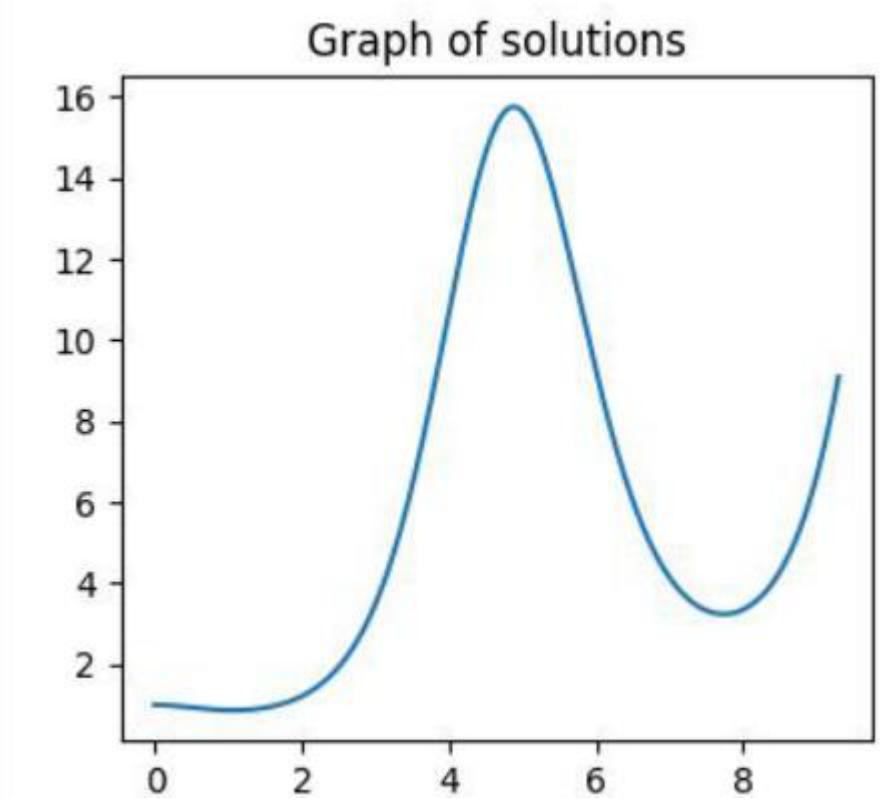
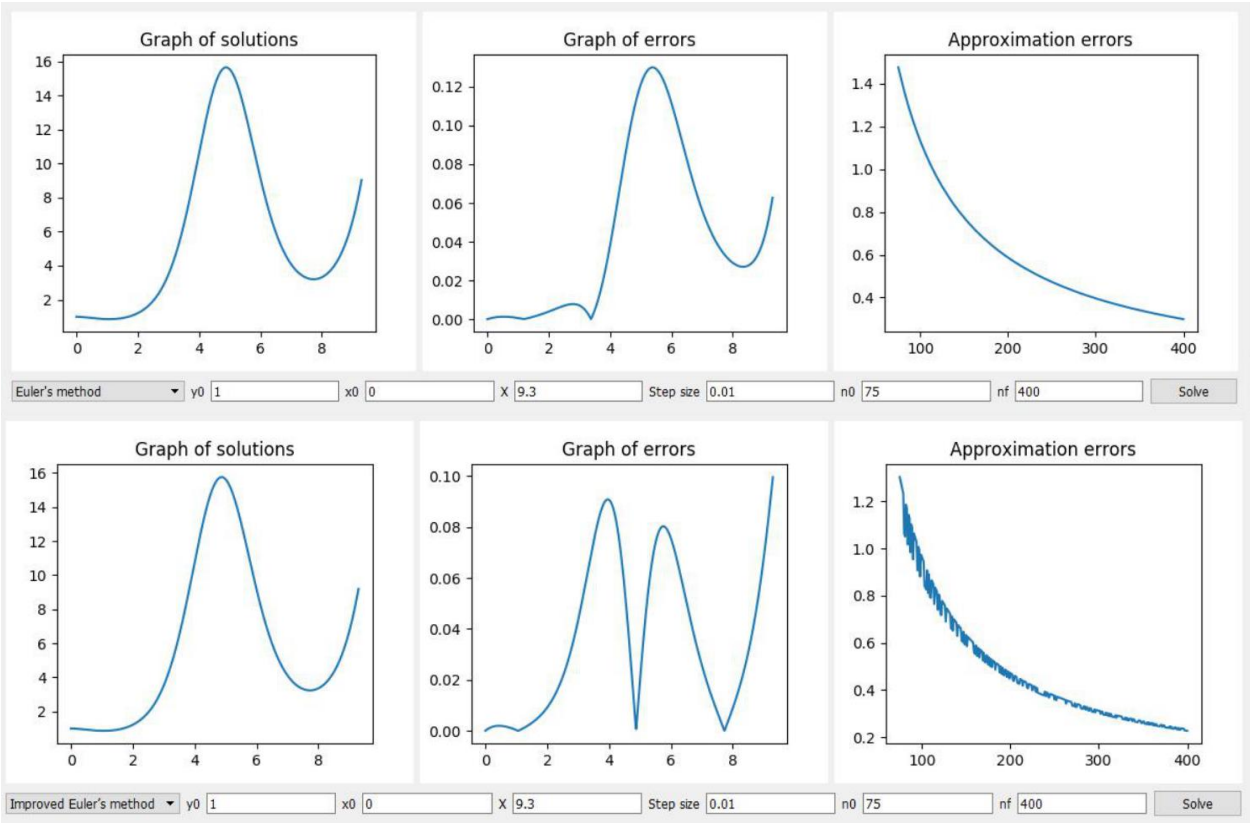
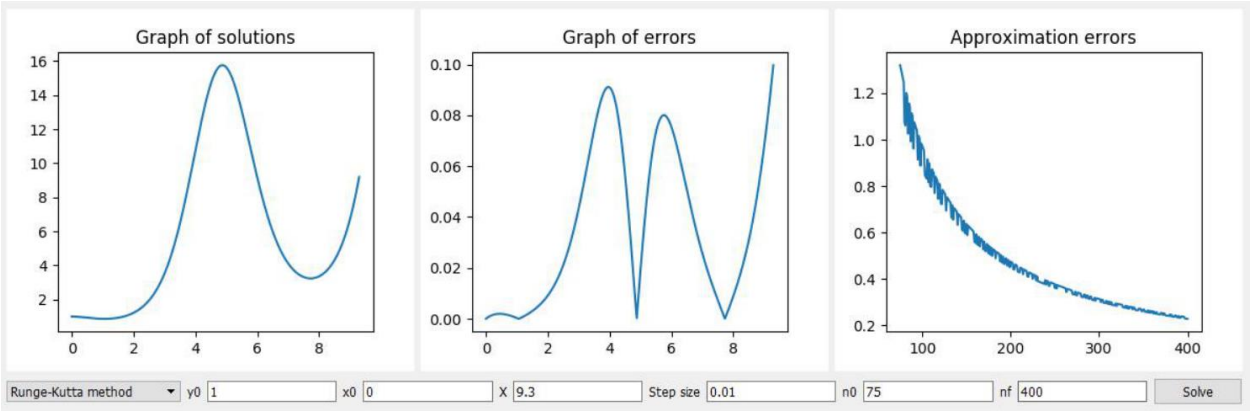
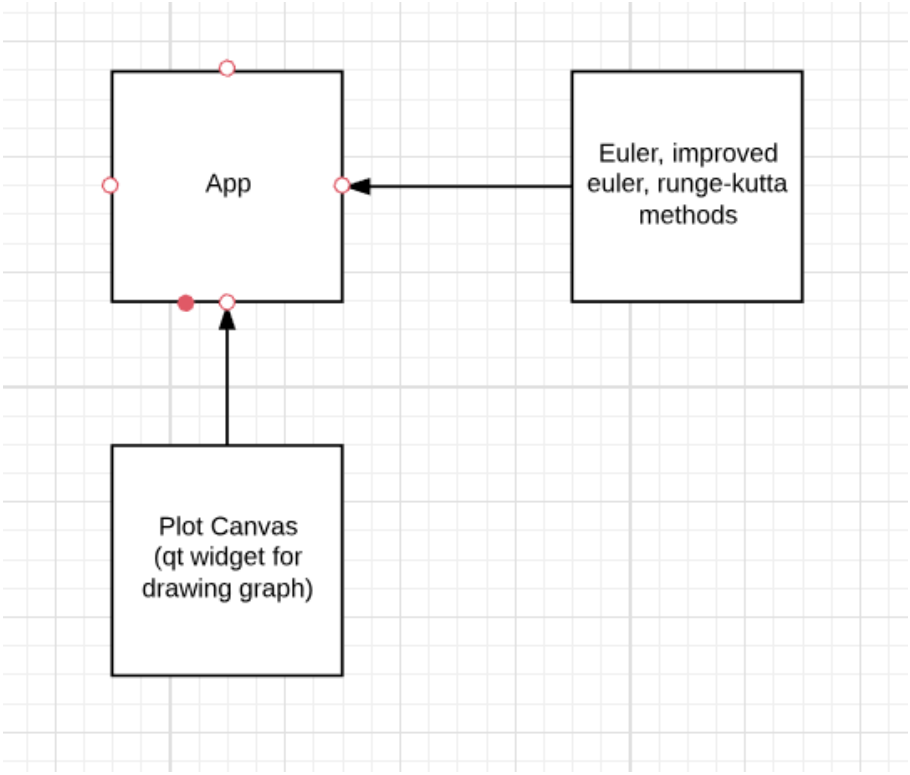
Exact solution to e^(-sin(x)) - y*cos(x)



Results of euler, improved euler, runge-kutta method in program







Interesting parts of source code

```python
def runge_kutta(f, y0, x0, X, h, ap=False):
    x, y = x0, y0
    error_max = -math.inf
    while x <= X:
        k1 = f(x, y)
        k2 = f(x + 0.5 * h, y + 0.5 * h * k1)
        k3 = f(x + 0.5 * h, y + 0.5 * h * k2)
        k4 = f(x + h, y + h * k3)
        y += h * (k1/6 + k2/3 + k3/3 + k4/6)
        if not ap:
            x_graph.append(x)
            y_graph.append(y)
        else:
            error_max = max(error_max, calc_error(y, x0, y0, x))
        x += h
    if ap:
        x_appr.append((X-x0)/h)
        y_appr.append(error_max)
```

```python
def improved_euler(f, y0, x0, X, h, ap=False):
    t,y = x0,y0
    error_max = -math.inf
    while t <= X:
        k1 = f(t, y)
        k2 = f(t + h, y + h * k1)
        y += h*(k1 + k2)/2
        if not ap:
            x_graph.append(t)
            y_graph.append(y)
        else:
            error_max = max(error_max, calc_error(y, x0, y0, t))
        t += h
    if ap:
        x_appr.append((X-x0)/h)
        y_appr.append(error_max)
```

```python
def euler(f, y0, x0, X, h, ap=False):
    t,y = x0,y0
    error_max = -math.inf
    while t <= X:
        y += h * f(t,y)
        if not ap:
            x_graph.append(t)
            y_graph.append(y)
        else:
            error_max = max(error_max, calc_error(y, x0, y0, t))
        t += h
    if ap:
        x_appr.append((X-x0)/h)
        y_appr.append(error_max)
```

```
        nf = int(self.nf.text())
        if self.method_box.currentIndex() == 0:
            euler(func, y0, x0, x, step)
        elif self.method_box.currentIndex() == 1:
            improved_euler(func, y0, x0, x, step)
        elif self.method_box.currentIndex() == 2:
            runge_kutta(func, y0, x0, x, step)
        elif self.method_box.currentIndex() == 3:
            exact_solution(y0, x0, x, step)
        for z in range(len(x_graph)):
            i = x_graph[z]
            j = y_graph[z]
            y_graph_error.append(calc_error(j, x0, y0, i))
        for i in range(n0, nf + 1):
            if self.method_box.currentIndex() == 0:
                euler(func, y0, x0, x, (x - x0)/i, ap=True)
            elif self.method_box.currentIndex() == 1:
                improved_euler(func, y0, x0, x, (x - x0)/i, ap=True)
            elif self.method_box.currentIndex() == 2:
                runge_kutta(func, y0, x0, x, (x - x0)/i, ap=True)
            elif self.method_box.currentIndex() == 3:
                exact_solution(y0, x0, x, (x - x0)/i, ap=True)
        self.graph.plot(x_graph, y_graph)
        self.errors.plot(x_graph, y_graph_error)
        self.appr_errors.plot(x_appr, y_appr)
```