

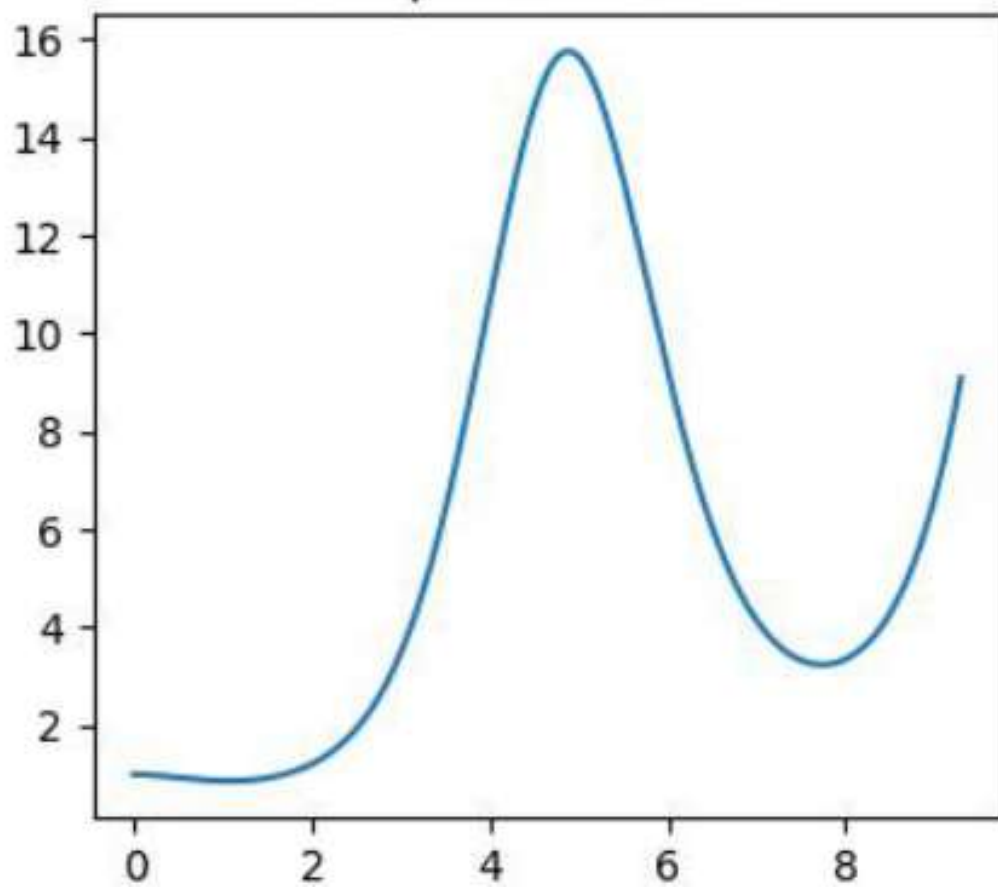
# Differential equations assignment

[github.com/gavaz10/de\\_ass](https://github.com/gavaz10/de_ass)

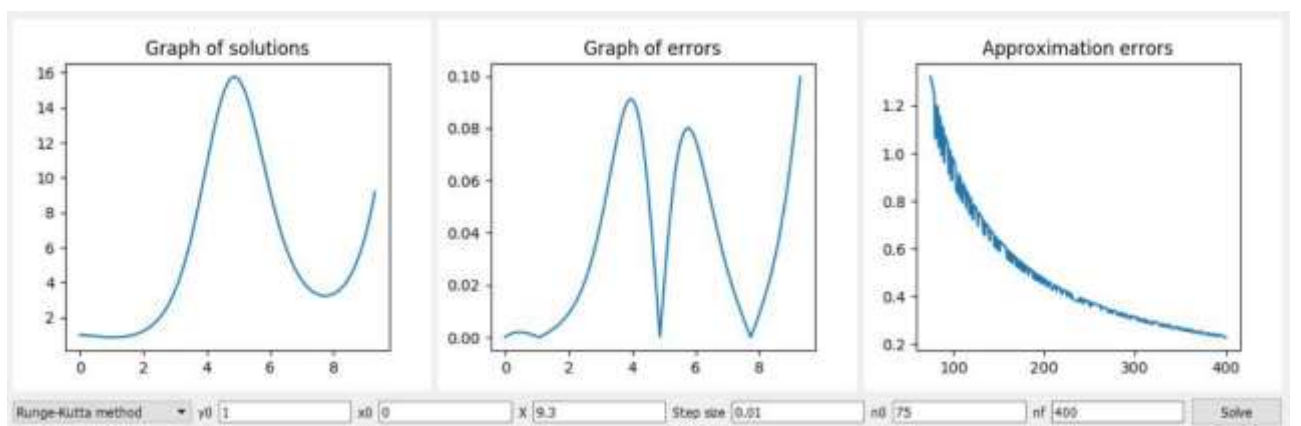
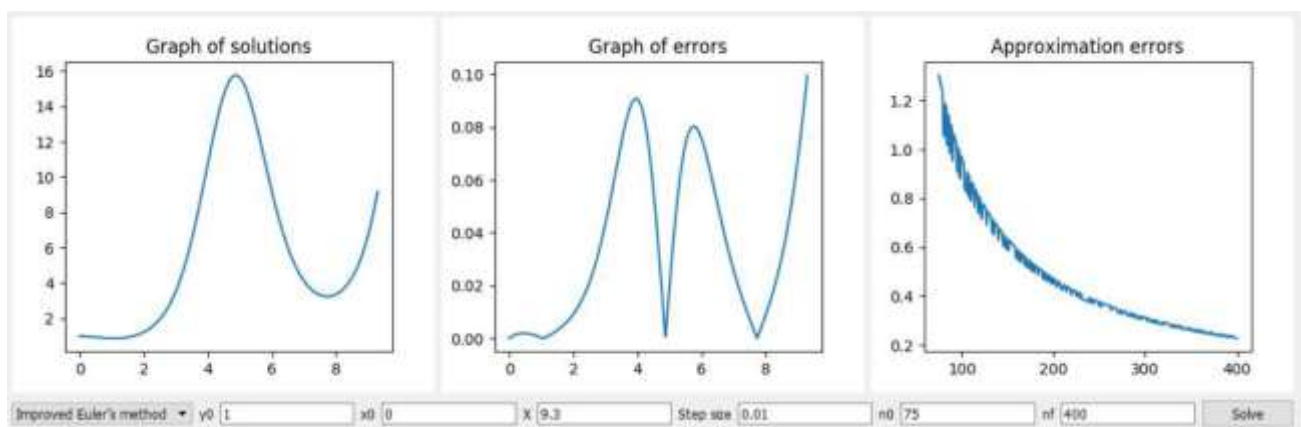
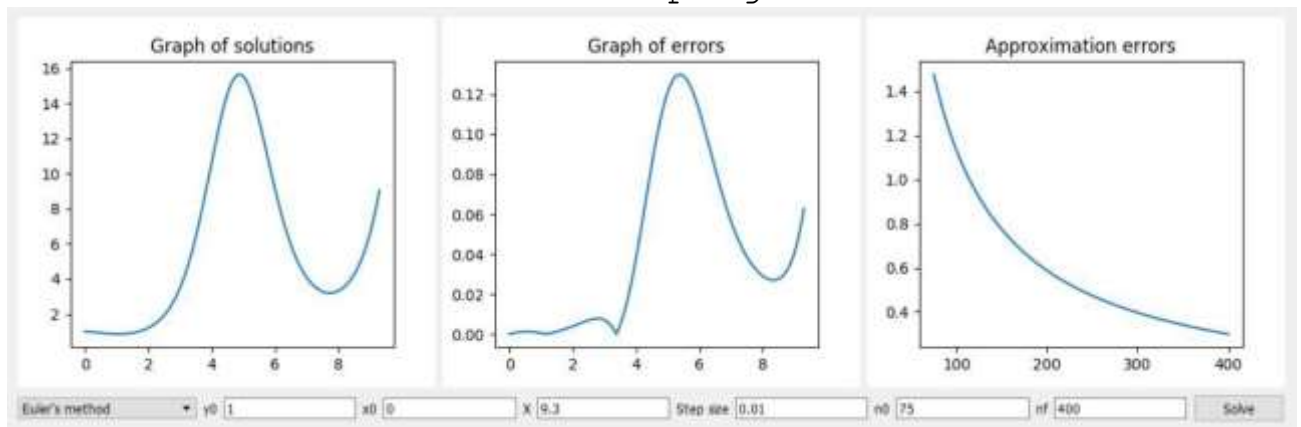
Done by  
Gayaz Kamaletdinov b17-06

Exact solution to  $e^{-\sin(x)} - y \cos(x)$

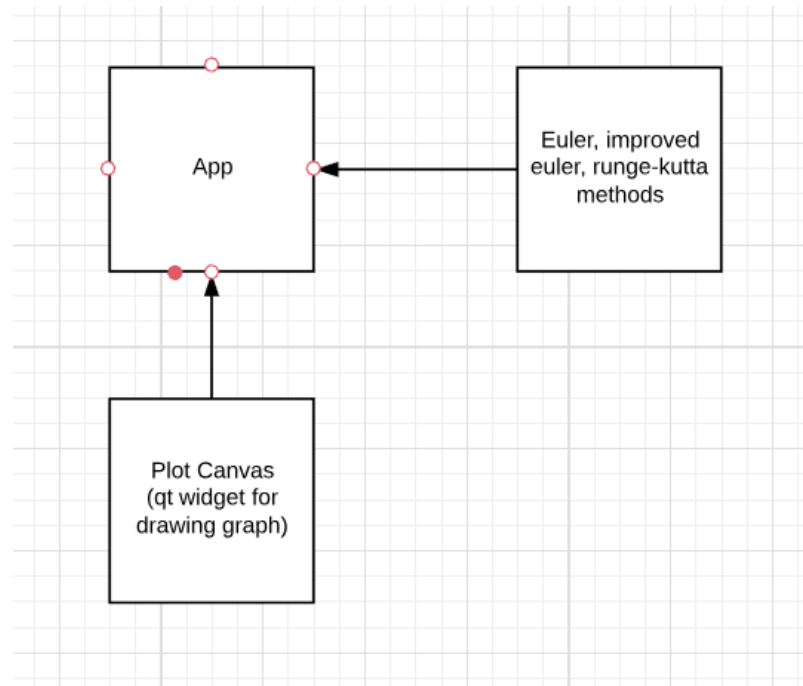
Graph of solutions



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



## Structure



main.py

```

v e
v x_graph
v y_graph
v y_graph_error
v x_appr
v y_appr
f runge_kutta(f, y0, x0, X, h, ap=False)
f improved_euler(f, y0, x0, X, h, ap=False)
f euler(f, y0, x0, X, h, ap=False)
f exact_solution(y0, x0, X, h, ap=False)
f calc_error(num_ans, x0, y0, x)
f func(x, y)
v App(QWidget)
  m __init__(self)
  m on_click(self)
  m initUI(self)
  f appr_errors
  f errors
  f graph
  f method_box
  f n0
  f nf
  f step
  f x
  f x0
  f y0
v PlotCanvas(FigureCanvas)
  m __init__(self, parent=None, width=5, height=4, dpi=100, text="lul")
  m plot(self, x, y)
  f axes
  f fig
  f text
```

## Interesting parts of source code

```
16
17 def runge_kutta(f, y0, x0, X, h, ap=False):
18     x, y = x0, y0
19     error_max = -math.inf
20     while x <= X:
21         k1 = f(x, y)
22         k2 = f(x + 0.5 * h, y + 0.5 * h * k1)
23         k3 = f(x + 0.5 * h, y + 0.5 * h * k2)
24         k4 = f(x + h, y + h * k3)
25         y += h * (k1/6 + k2/3 + k3/3 + k4/6)
26         if not ap:
27             x_graph.append(x)
28             y_graph.append(y)
29         else:
30             error_max = max(error_max, calc_error(y, x0, y0, x))
31         x += h
32     if ap:
33         x_appr.append((X-x0)/h)
34         y_appr.append(error_max)
35
```

```
36
37 def improved_euler(f, y0, x0, X, h, ap=False):
38     t, y = x0, y0
39     error_max = -math.inf
40     while t <= X:
41         k1 = f(t, y)
42         k2 = f(t + h, y + h * k1)
43         y += h * (k1 + k2) / 2
44         if not ap:
45             x_graph.append(t)
46             y_graph.append(y)
47         else:
48             error_max = max(error_max, calc_error(y, x0, y0, t))
49         t += h
50     if ap:
51         x_appr.append((X-x0)/h)
52         y_appr.append(error_max)
53
```

```

54
55 def euler(f, y0, x0, X, h, ap=False):
56     t, y = x0, y0
57     error_max = -math.inf
58     while t <= X:
59         y += h * f(t, y)
60         if not ap:
61             x_graph.append(t)
62             y_graph.append(y)
63         else:
64             error_max = max(error_max, calc_error(y, x0, y0, t))
65             t += h
66     if ap:
67         x_appr.append((X-x0)/h)
68         y_appr.append(error_max)
69

```

```

114
115
116
117
118
119
120
121
122
123 for z in range(len(x_graph)):
124     i = x_graph[z]
125     j = y_graph[z]
126     y_graph_error.append(calc_error(j, x0, y0, i))
127 for i in range(n0, nf + 1):
128     if self.method_box.currentIndex() == 0:
129         euler(func, y0, x0, x, (x - x0)/i, ap=True)
130     elif self.method_box.currentIndex() == 1:
131         improved_euler(func, y0, x0, x, (x - x0)/i, ap=True)
132     elif self.method_box.currentIndex() == 2:
133         runge_kutta(func, y0, x0, x, (x - x0)/i, ap=True)
134     elif self.method_box.currentIndex() == 3:
135         exact_solution(y0, x0, x, (x - x0)/i, ap=True)
136 self.graph.plot(x_graph, y_graph)
137 self.errors.plot(x_graph, y_graph_error)
138 self.appr_errors.plot(x_appr, y_appr)
139

```