Problem Set 4

Rippy

October 2020

Problem 1

(a)

Using separation of variables and unitizing the initial condition, we get the exact solution to be:

$$y(t) = \frac{1}{1+t}$$

(b)

Using the code below, I found the approximate value $y(1) = 0.46117$

```
f[t_] := -t^2;


y0 = 1;
n = 5;
h = 1.0 / n;
For[i = 0, i < n, i++,
 y2 = y0 + h * f[y0];

 Print[(i + 1) * h];
 Print[y2];
 y0 = y2;




]
0.2
0.8
0.4
0.672
0.6
0.581683
0.8
0.514012
1.
0.46117
```

(c)

The process was repeated for a step size of 0.1, and I found the approximate value $y(1) = 0.481713$

(d)

The error for (a) is 0.03883, and for (b) is 0.018287. This is about half (0.47) the error when halving the step side, which is consistent with the theory of the error being proportional to order h.

Problem 2

Using Taylor's method of order two in the code below, we find the approximate value of $y(1) = 0.506914$, which has an error of 0.006914 from the exact value.

```
y0 = 1;
y1[t_] := -t^2;
g[t_] := -2 * t;
n = 5;
h = 1.0 / n;
```

```
For[i = 0, i < n, i++,
 y2 = y0 + h * y1[y0] + 1/2 * h^2 * y1[y0] * g[y0];
 Print[(i + 1) * h];
 Print[y2];
 y0 = y2;


]
0.2
0.84
0.4
0.722588
0.6
0.633253
0.8
0.563209
1.
0.506914
```

Problem 3

Given:

$$\frac{dy}{dt} = (t+1)\sqrt{y}$$

We can use separation of variables,the initial condition, and integrate both sides, which gives us:

$$y(t) = (\frac{t^2}{4} + \frac{t}{2} + 1)^2$$

Which gives us an exact value $y(1) = 3.0625$

Problem 4

(a)

Using the midpoint method with $h = 0.5$, we find $y(1) = 2.99292$, which has an error of 0.06958.

(b)

Using the midpoint method with $h = 0.25$, we find $y(1) = 3.04307$, which has an error of 0.01943.

(c)

The results match what theory predicts. The error falls off on order h^2. The error from a step size of 0.5 to 0.25 is about 3.58 times less, which is in line with theory of about 4 times less.

Problem 5

Using the code below, we implement RK4. We find the approximate value $y(1) = 3.06246$.

```
f[t_, y_] := (t + 1) √y ;

ti = 0;
wi = 1;
n = 4;
h = 1 / n;
For[i = 0, i < n, i++,
 h = 0.25;
 k1 = f[ti, wi];
 k2 = f[ti + h/2, wi + h/2 * k1];
 k3 = f[ti + h/2, wi + h/2 * k2];
 k4 = f[ti + h, wi + h * k3];
 w2 = wi + h/6 (k1 + 2 * k2 + 2 * k3 + k4);
 wi = w2;
 ti = ti + h;
 Print[w2]
]

1.30102
1.72264
2.2971
3.06246
```

Problem 6

To derive b_0 and b_1, we must integrate

$$f(t_i, w_i)\frac{(t - t_{i+1})}{(t_i - t_{i+1})} + f(t_{i+1}, w_{i+1})\frac{(t - t_i)}{(t_{i+1} - t_i)}$$

We will then substitute to put this in terms of h, and since this is an implicit method, we will set the bounds of integration to be 0 to h. That leaves us to evaluate the following integral:

$$\int_0^h f(t_i, w_i)\frac{t - h}{-h} + f(t_{i+1}, w_{i+1})\frac{t - 0}{h}$$

Simplifying we get:

$$\frac{1}{h}\int_0^h f(t_i, w_i)(h - t) + f(t_{i+1}, w_{i+1})(t)$$

Integrating, we get:

$$h[\frac{1}{2}f(t_i, w_i) + \frac{1}{2}f(t_{i+1}, w_{i+1})]$$

Indicating that $b_0, b_1 = \frac{1}{2}$.

Problem 7

From RK4, we determine $w_1 = 1.30102$. Thus, we have w_0, w_1. The predictor corrector method goes as follows:
Take w_0, w_1 and put them into AB2, this will give us a rough w_2 value. We then take this rough value and w_1 and put them into the method of Problem 6, and it outputs a corrected w_2 value. The code is below. the value it predicts for $y(1)$ is 3.06987.

```
f[t_, y_] := (t + 1) * √y;



w0 = 1;
w1 = 1.30102;
h = 0.25;
For[i = 1, i < 4, i++,
 w2bad = w1 + h/2 * (3 * f[h * (i), w1] - f[h * (i - 1), w0]);
 w2good = w1 + h/2 * (f[h * (i + 1), w2bad] + f[h * i, w1]);

 w0 = w1;
 w1 = w2good;
 Print[w2good]
]
1.72448
2.30137
3.06987
```

Problem 8

For 4, the error was order h^2, for 5, the error was order h^4, and for 7, the error was order h^3. As far as the best method (in terms of error), RK4 reigns supreme, with the closest approximation, and quickest error drop off. This would be the method to go with if computations are cheap, and precision is most important. If computations are very expensive, midpoint is the best of the three, since it has the least computations of the three, with still a relatively small error. If you want a compromise, the predictor corrector method of 7 is a few more computations, but in turn has less error than the midpoint.