# Math 342: Project 2

Connor Emmons

Documentation: The main Project 2 MatLab script and all required dependencies are located in the Project 2 folder found here: https://github.com/Connor-Lemons/Emmons-Math-342. No other resources used.

Problem 1: Arrange the parameters of RK4 in a Butcher Tableu

$$w_0 = \alpha \tag{1}$$

$$K_1 = hf(t_k, w_k) \tag{2}$$

$$K_2 = hf\left(t_k + \frac{h}{2}, w_k + \frac{1}{2}K_1\right) \tag{3}$$

$$K_3 = hf\left(t_k + \frac{h}{2}, w_k + \frac{1}{2}K_2\right) \tag{4}$$

$$K_4 = hf(t_{k+1}, w_k + K_3) \tag{5}$$

$$w_{k+1} = w_k + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) \tag{6}$$

The Butcher Tableu representation of this method is:

| | | | | |
|---|---|---|---|---|
| $0$ | $0$ | $0$ | $0$ | $0$ |
| $\frac{1}{2}$ | $\frac{1}{2}$ | $0$ | $0$ | $0$ |
| $\frac{1}{2}$ | $0$ | $\frac{1}{2}$ | $0$ | $0$ |
| $1$ | $0$ | $0$ | $1$ | $0$ |
| | $\frac{1}{6}$ | $\frac{1}{3}$ | $\frac{1}{3}$ | $\frac{1}{6}$ |

Problem 2: For the three RK methods shown below, given the Butcher Tableau representation, give the equations which define those methods.

Ralston Method

| | | |
|---|---|---|
| $0$ | | |
| $\frac{2}{3}$ | $\frac{2}{3}$ | |
| | $\frac{1}{4}$ | $\frac{3}{4}$ |

Heun Third-Order Method

| | | | |
|---|---|---|---|
| $0$ | | | |
| $\frac{1}{3}$ | $\frac{1}{3}$ | | |
| $\frac{2}{3}$ | $0$ | $\frac{2}{3}$ | |
| | $\frac{1}{4}$ | $0$ | $\frac{3}{4}$ |

Runge-Kutta 3/8 Rule

| | | | | |
|---|---|---|---|---|
| $0$ | | | | |
| $\frac{1}{3}$ | $\frac{1}{3}$ | | | |
| $\frac{2}{3}$ | $-\frac{1}{3}$ | $1$ | | |
| $1$ | $1$ | $-1$ | $1$ | |
| | $\frac{1}{8}$ | $\frac{3}{8}$ | $\frac{3}{8}$ | $\frac{1}{8}$ |

The equations for the Ralston Method are:

$$w_0 = \alpha \tag{1}$$

$$K_1 = hf(t_k, w_k) \tag{2}$$

$$K_2 = hf\left(t_k + \tfrac{2}{3}h, w_k + \tfrac{2}{3}K_1\right) \tag{3}$$

$$w_{k+1} = w_k + \tfrac{1}{4}(K_1 + 3K_2) \tag{4}$$

The equations for the Heun Third-Order Method are:

$$w_0 = \alpha \tag{5}$$

$$K_1 = hf(t_k, w_k) \tag{6}$$

$$K_2 = \mathrm{hf}\left(t_k + \tfrac{1}{3}h, w_k + \tfrac{1}{3}K_1\right) \tag{7}$$

$$K_3 = hf\left(t_k + \tfrac{2}{3}h, w_k + \tfrac{2}{3}K_2\right) \tag{8}$$

$$w_{k+1} = w_k + \tfrac{1}{4}(K_1 + 3K_3) \tag{9}$$

The equations for the Runga-Kutta 3/8 Rule are:

$$w_0 = \alpha \tag{10}$$

$$K_1 = hf(t_k, w_k) \tag{11}$$

$$K_2 = hf\left(t_k + \frac{1}{3}h, w_k + \frac{1}{3}K_1\right) \quad (12)$$

$$K_3 = hf\left(t_k + \frac{2}{3}h, w_k - \frac{1}{3}K_1 + K_2\right) \quad (13)$$

$$K_4 = hf(t_k + h, w_k + K_1 - K_2 + K_3) \quad (14)$$

$$w_{k+1} = w_k + \frac{1}{8}(K_1 + 3K_2 + 3K_3 + K_4) \quad (15)$$

Problem 3: Implement the three methods discussed in Problem 2 in MatLab.

Code can be found in the GitHub page or attached at the end of this document.

https://github.com/Connor-Lemons/Emmons-Math-342/tree/main/Project%202

Problem 4: Implement the Runge-Kutta-Fehlberg method according to Algorithm 5.3.

Code can be found in the GitHub page or attached at the end of this document.

https://github.com/Connor-Lemons/Emmons-Math-342/tree/main/Project%202

Problem 5: Rewrite the equation $y'(t) = kx(t)y(t)$, where $x(t)$ is the number of susceptible (non-infective) individuals and y(t) is the number of infective individuals, as a function solely of $y(t)$ and $m$, which represents the total population.

Note that the the sum of the number of non-infective individuals and the number of infective individuals is the size of the population. This gives:

$$m = x(t) + y(t) \quad (1)$$

Solving for $x(t)$ gives:

$$x(t) = m - y(t) \quad (2)$$

Making this substitution in the original equation gives:

$$y'(t) = k\big(m - y(t)\big)y(t) \quad (3)$$

$$y'(t) = kmy(t) - ky(t)^2 \quad (4)$$

Problem 6: Use the methods described in Problems 1-4 to obtain various estimated solutions to equation (4) in Problem 5.

Parameters:

$$m = 100000 \quad (1)$$

$$y(0) = 1000 \quad (2)$$

$$k = 2 \times 10^{-6} \quad (3)$$

$$t_i = 0\ days \quad (4)$$

$$t_f = 30\ days \quad (5)$$

The results of these methods can be found in the Github page or attached at the end of this document.

https://github.com/Connor-Lemons/Emmons-Math-342/tree/main/Project%202

The final estimation of the number of infective individuals from each method are shown below. Note that all values are rounded to the nearest integer.

$h = 2$:

Ralston: $y(30) = 79319$

Heun: $y(30) = 80231$

RK38: $y(30) = 80289$

RK4: $y(30) = 80288$

$h = 1$:

Ralston: $y(30) = 80028$

Heun: $y(30) = 80287$

RK38: $y(30) = 80295$

RK4: $y(30) = 80295$

RKF45: $y(30) = 80296$

All estimations are relatively similar. The higher the order of the method, the more accurate it is likely to be. Running the methods with a smaller $h$ also likely produced more accurate results, with the most accurate method being the RKF45 method due to its high order and optimizing of $h$ to match a specified tolerance level.

# Project 2

```
% clear; clc;
```

## Problem 6

```
m = 100000;
y_0 = 1000;
k = 2e-6;
t_i = 0;
t_f = 30;

syms t y
f(t, y) = k*m*y - k*y^2;

N = 15;
[t_Ralston_case1, w_Ralston_case1] = Ralston(f, t_i, t_f, N, y_0);
[t_Heun_case1, w_Heun_case1] = Heun(f, t_i, t_f, N, y_0);
[t_RK38_case1, w_RK38_case1] = RK38(f, t_i, t_f, N, y_0);
[t_RK4_case1, w_RK4_case1] = RK4(f, t_i, t_f, N, y_0);

N = 30;
[t_Ralston_case2, w_Ralston_case2] = Ralston(f, t_i, t_f, N, y_0);
[t_Heun_case2, w_Heun_case2] = Heun(f, t_i, t_f, N, y_0);
[t_RK38_case2, w_RK38_case2] = RK38(f, t_i, t_f, N, y_0);
[t_RK4_case2, w_RK4_case2] = RK4(f, t_i, t_f, N, y_0);

TOL = 1e-2;
hmax = 2;
hmin = 1;
[t_RKF45, w_RKF45, h_RKF45] = RKF45(f, t_i, t_f, y_0, TOL, hmax, hmin);

tableGen("Ralston with h = 2", t_Ralston_case1, w_Ralston_case1);
```

```
Ralston with h = 2
i              t              w
1              0              1000
2              2              1473.406912
3              4              2166.349885
4              6              3175.36656
5              8              4633.507291
6              10             6717.612826
7              12             9649.73179
8              14             13683.81293
9              16             19064.89833
10             18             25949.91898
11             20             34296.9513
12             22             43768.44791
13             24             53729.28586
14             26             63393.46
15             28             72063.7384
16             30             79319.345
```

```
tableGen("Heun with h = 2", t_Heun_case1, w_Heun_case1);
```

```
Heun with h = 2
i               t               w
1               0               1000
2               2               1483.476947
3               4               2195.580913
4               6               3238.402916
5               8               4752.718476
6               10              6924.972447
7               12              9986.883079
8               14              14197.63672
9               16              19795.49586
10              18              26910.70818
11              20              35454.67349
12              22              45039.86068
13              24              55006.59604
14              26              64585.52021
15              28              73121.5041
16              30              80231.35432
```

```
tableGen("Runge-Kutta 3/8 with h = 2", t_RK38_case1, w_RK38_case1);
```

```
Runge-Kutta 3/8 with h = 2
i               t               w
1               0               1000
2               2               1484.439053
3               4               2198.351628
4               6               3244.307299
5               8               4763.694516
6               10              6943.611543
7               12              10016.22019
8               14              14240.50536
9               16              19853.42098
10              18              26982.77728
11              20              35537.29954
12              22              45127.89761
13              24              55094.73239
14              26              64668.54443
15              28              73194.14719
16              30              80289.29072
```

```
tableGen("Runga-Kutta 4th Order with h = 2", t_RK4_case1, w_RK4_case1);
```

```
Runga-Kutta 4th Order with h = 2
i               t               w
1               0               1000
2               2               1484.438695
3               4               2198.350324
4               6               3244.303729
5               8               4763.685843
6               10              6943.591958
7               12              10016.17847
8               14              14240.42138
9               16              19853.26202
10              18              26982.49677
11              20              35536.84283
12              22              45127.22092
13              24              55093.8364
14              26              64667.50379
15              28              73193.09644
16              30              80288.36099
```

```
tableGen("Ralston with h = 1", t_Ralston_case2, w_Ralston_case2);
```

```
Ralston with h = 1
i               t               w
1               0               1000
2               1               1217.377864
3               2               1481.316424
4               3               1801.457069
5               4               2189.279915
6               5               2658.379531
7               6               3224.745456
8               7               3907.026527
9               8               4726.746421
10              9               5708.422715
11              10              6879.523353
12              11              8270.174076
13              12              9912.511242
14              13              11839.56266
15              14              14083.54399
16              15              16673.49287
17              16              19632.24123
18              17              22972.85805
19              18              26694.877
20              19              30780.82839
21              20              35193.76506
22              21              39876.52283
23              22              44753.3098
24              23              49733.84182
25              24              54719.69593
26              25              59611.99746
27              26              64319.19117
28              27              68763.61985
29              28              72885.95858
30              29              76647.09778
31              30              80027.63253
```

```
tableGen("Heun with h = 1", t_Heun_case2, w_Heun_case2);
```

```
Heun with h = 1
i               t               w
1               0               1000
2               1               1218.641214
3               2               1484.370978
4               3               1806.986203
5               4               2198.157349
6               5               2671.708115
7               6               3243.898368
8               7               3933.687977
9               8               4762.947053
10              9               5756.562411
11              10              6942.371244
12              11              8350.832555
13              12              10014.32852
14              13              11965.97821
15              14              14237.85495
16              15              16858.53945
17              16              19850.02628
18              17              23224.14035
19              18              26978.80439
20              19              31094.69654
21              20              35532.98777
22              21              40234.87066
23              22              45123.41404
24              23              50107.88653
25              24              55090.16104
```

```
26              25              59972.29821
27              26              64664.09518
28              27              69089.39671
29              28              73190.29467
30              29              76928.86006
31              30              80286.57585
```

```
tableGen("Runge-Kutta 3/8 with h = 1", t_RK38_case2, w_RK38_case2);
```

```
Runge-Kutta 3/8 with h = 1
i               t               w
1               0               1000
2               1               1218.70194
3               2               1484.517232
4               3               1807.249738
5               4               2198.578231
6               5               2672.336127
7               6               3244.794336
8               7               3934.924828
9               8               4764.610314
10              9               5758.749788
11              10              6945.190825
12              11              8354.399151
13              12              10018.75774
14              13              11971.37805
15              14              14244.31532
16              15              16866.12036
17              16              19858.74664
18              17              23233.96968
19              18              26989.65969
20              19              31106.44588
21              20              35545.46022
22              21              40247.86982
23              22              45136.73109
24              23              50121.30925
25              24              55103.47766
26              25              59985.2982
27              26              64676.57114
28              27              69101.15071
29              28              73201.14968
30              29              76938.67369
31              30              80295.25235
```

```
tableGen("Runga-Kutta 4th Order with h = 1", t_RK4_case2, w_RK4_case2);
```

```
Runga-Kutta 4th Order with h = 1
i               t               w
1               0               1000
2               1               1218.701934
3               2               1484.517215
4               3               1807.249704
5               4               2198.578169
6               5               2672.336022
7               6               3244.794166
8               7               3934.924559
9               8               4764.609895
10              9               5758.749147
11              10              6945.189859
12              11              8354.397713
13              12              10018.75562
14              13              11971.37497
15              14              14244.3109
16              15              16866.11408
```

```
17              16              19858.73788
18              17              23233.95764
19              18              26989.64347
20              19              31106.42449
21              20              35545.43271
22              21              40247.83541
23              22              45136.68936
24              23              50121.26032
25              24              55103.42229
26              25              59985.23779
27              26              64676.5076
28              27              69101.08622
29              28              73201.08639
30              29              76938.61347
31              30              80295.1966
```

```
RKF45_print(t_RKF45, w_RKF45, h_RKF45);
```

```
Runga-Kutta-Fehlberg
i               t               w               h
1               0               1000            NaN
2               2               1484.53         2
3               3.98195         2190.866        1.98195
4               5.793882        3117.911        1.811933
5               7.454612        4293.501        1.66073
6               8.998258        5757.038        1.543646
7               10.45361        7555.156        1.455351
8               11.84445        9742.111        1.39084
9               13.19176        12381.92        1.347311
10              14.51569        15551.77        1.323927
11              15.83751        19347.29        1.321822
12              17.18223        23891.17        1.34472
13              18.583          29348.86        1.400773
14              20.09089        35964.1         1.507889
15              21.80482        44173.19        1.713933
16              23.80482        54137.12        2
17              25.80482        63780.81        2
18              27.80482        72429.42        2
19              29.31261        78030.12        1.507792
20              30              80296.02        0.6873852
```

```
function tableGen(name, t, w)

    fprintf("%s\n", name);
    fprintf("%-15s%-15s%-15s\n", "i", "t", "w");

    for j = 1:length(t)
        fprintf("%-15.7g%-15.7g%-15.10g\n", j, t(j), w(j));
    end

end

function RKF45_print(t_RKF45, w_RKF45, h_RKF45)

    fprintf("%s\n", "Runga-Kutta-Fehlberg");
    fprintf("%-15s%-15s%-15s%-15s\n", "i", "t", "w", "h");
    for j = 1:length(t_RKF45)
```

```
        fprintf("%-15.7g%-15.7g%-15.7g%-15.7g\n", j, t_RKF45(j), w_RKF45(j),
h_RKF45(j));
    end

end
```

```
function [t, w] = Ralston(f, a, b, N, alpha)

    syms t y

    h = (b - a)/N;
    t(1) = a;
    w(1) = alpha;

    for i = 1:N



        K1 = h*f(t(i), w(i));
        K2 = h*f(t(i) + (2/3)*h, w(i) + (2/3)*K1);

        w(i+1) = w(i) + (1/4)*(K1 + 3*K2);
        t(i+1) = a + i*h;

    end

end
```

```
function [t, w] = Heun(f, a, b, N, alpha)

    syms t y

    h = (b - a)/N;
    t(1) = a;
    w(1) = alpha;

    for i = 1:N



        K1 = h*f(t(i), w(i));
        K2 = h*f(t(i) + (1/3)*h, w(i) + (1/3)*K1);
        K3 = h*f(t(i) + (2/3)*h, w(i) + (2/3)*K2);

        w(i+1) = w(i) + (1/4)*(K1 + 3*K3);
        t(i+1) = a + i*h;

    end

end
```

```
function [t, w] = RK38(f, a, b, N, alpha)

    syms t y

    h = (b - a)/N;
    t(1) = a;
    w(1) = alpha;

    for i = 1:N



        K1 = h*f(t(i), w(i));
        K2 = h*f(t(i) + (1/3)*h, w(i) + (1/3)*K1);
        K3 = h*f(t(i) + (2/3)*h, w(i) - (1/3)*K1 + K2);
        K4 = h*f(t(i) + h, w(i) + K1 - K2 + K3);

        w(i+1) = w(i) + (1/8)*(K1 + 3*K2 + 3*K3 + K4);
        t(i+1) = a + i*h;

    end

end
```

```
function [t, w] = RK4(f, a, b, N, alpha)

    syms t y

    h = (b - a)/N;
    t(1) = a;
    w(1) = alpha;

    for i = 1:N



        K1 = h*f(t(i), w(i));
        K2 = h*f(t(i) + h/2, w(i) + K1/2);
        K3 = h*f(t(i) + h/2, w(i) + K2/2);
        K4 = h*f(t(i) + h, w(i) + K3);

        w(i+1) = w(i) + (K1 + 2*K2 + 2*K3 + K4)/6;
        t(i+1) = a + i*h;

    end

end
```

```
function [t, w, h] = RKF45(f, a, b, alpha, TOL, hmax, hmin)

    syms t y

    t(1) = a;
    w(1) = alpha;
    h(1) = hmax;
    h_temp = h(1);
    FLAG = 1;
    i = 1;

    while FLAG == 1

        iter = false;
        h(i) = h_temp;

        K1 = h(i)*f(t(i), w(i));
        K2 = h(i)*f(t(i) + (1/4)*h(i), w(i) + K1/4);
        K3 = h(i)*f(t(i) + (3/8)*h(i), w(i) + (3/32)*K1 + (9/32)*K2);
        K4 = h(i)*f(t(i) + (12/13)*h(i), w(i) + (1932/2197)*K1 - (7200/2197)*K2 + ↙
(7296/2197)*K3);
        K5 = h(i)*f(t(i) + h(i), w(i) + (439/216)*K1 - 8*K2 + (3680/513)*K3 - ↙
(845/4104)*K4);
        K6 = h(i)*f(t(i) + (1/2)*h(i), w(i) - (8/27)*K1 +2*K2 - (3544/2565)*K3 + ↙
(1859/4104)*K4 - (11/40)*K5);

        R = (1/h(i))*abs((1/360)*K1 - (128/4275)*K3 - (2197/75240)*K4 + (1/50)*K5 + ↙
(2/55)*K6);

        if R < TOL
            t(i+1) = t(i) + h(i);
            w(i+1) = w(i) + (25/216)*K1 + (1408/2565)*K3 + (2197/4104)*K4 - (1/5)*K5;
            iter = true;
        end

        delta = 0.84*(TOL/R)^(1/4);

        if delta <= 0.1
            h_temp = 0.1*h(i);
        elseif delta >= 4
            h_temp = 4*h(i);
        else
            h_temp = delta*h(i);
        end

        if h_temp > hmax
            h_temp = hmax;
        end
```

```
        if t(end) >= b
            FLAG = 0;
        elseif t(end) + h_temp > b
            h_temp = b - t(end);
        elseif h_temp < hmin
            disp("minimum h exceeded")
            return
        end

        if iter == true
            h(i+1) = h_temp;
            i = i + 1;
        end

    end
    h = [NaN h(1:end-1)];

end
```