Differential Equation Estimations

Colton Williams

September 20, 2017

Part I
Function

$$y' = y - 4t^2 + 1$$

$$y(0) = 1.0$$

with t in the interval $[0, 1]$.

The exact solution is then

$$y(t) = 4t^2 + 8t - 6e^t + 7$$

with $y(1) = 2.690309$.

Part II
Approximations

1 Euler's Method

Given inputs a, b, N, α, we output approximations w of y at $(N + 1)$ values of t.

1. $h = (b - a)/N$
   $t = a$
   $w = \alpha$
   print (t, w)
2. For $i = 1, 2, ..., N$ do
   $w = w + hf(t, w)$
   $t = a + ih$
   print (t, w)
3. end algorithm

2 Modified Euler's Method

Given inputs a, b, N, α, we output approximations w of y at $(N+1)$ values of t.

1. $h = (b-a)/N$
   $t = a$
   $w_0 = \alpha$
   print (t, w)

2. For $i = 1, 2, ..., N$ do
   $w_{i+1} = w_i + \frac{h}{2}[f(t_i, w_i) + f(t_{i+1}, w_i + hf(t_i, w_i))]$
   $t = a + ih$
   print (t, w)

3. end algorithm

3 Runge-Kutta Method

Given inputs a, b, N, α, we output approximations w of y at $(N+1)$ values of t.

1. $h = (b-a)/N$
   $t = a$
   $w = \alpha$
   print (t, w)

2. For $i = 1, 2, ..., N$ do
   $K_1 = hf(t, w)$
   $K_2 = hf(t + h/2, w + K_1/2)$
   $K_3 = hf(t + h/2, w + K_2/2)$
   $K_4 = hf(t + h, w + K_3)$
   $w = w + (K_1 + 2K_2 + 2K_3 + K_4)/6$
   $t = a + ih$
   print (t, w)

3. end algorithm

Part III

Values

Euler

t_i	Exact	Estimation	Error
0.00	1.000000	1.000000	0.000000
0.05	1.102373	1.100000	0.002373
0.10	1.208974	1.204500	0.004474
0.15	1.318995	1.312725	0.006270
0.20	1.431583	1.423861	0.007722
0.25	1.545847	1.537045	0.008793
0.30	1.660847	1.651407	0.009440
0.35	1.775595	1.765977	0.009617
0.40	1.889052	1.897776	0.009276
0.45	2.000127	1.991765	0.008362
0.50	2.107672	2.100853	0.006819
0.55	2.210482	2.205896	0.004586
0.60	2.307287	2.305691	0.001596
0.65	2.396755	2.398975	0.002220
0.70	2.477484	2.484424	0.006940
0.75	2.548000	2.560645	0.012645
0.80	2.606754	2.626178	0.019423
0.85	2.652119	2.679486	0.027368
0.90	2.682381	2.718961	0.036579
0.95	2.695742	2.742909	0.47167
1.00	2.690309	2.749554	0.059245

Modified Euler

t_i	Exact	Estimation	Error
0.00	1.000000	1.000000	0.000000
0.05	1.102373	1.102250	0.000123
0.10	1.208974	1.208728	0.000247
0.15	1.318995	1.318625	0.000369
0.20	1.431583	1.431092	0.000491
0.25	1.545847	1.545236	0.000612
0.30	1.660847	1.660116	0.000731
0.35	1.775595	1.774747	0.000847
0.40	1.889052	1.888091	0.000961
0.45	2.000127	1.999055	0.001072
0.50	2.107672	2.106494	0.001178
0.55	2.210482	2.209202	0.001280
0.60	2.307287	2.305911	0.001376
0.65	2.396755	2.395289	0.001466
0.70	2.477484	2.475935	0.001548
0.75	2.548000	2.546377	0.001623
0.80	2.606754	2.605066	0.001688
0.85	2.652119	2.650376	0.001743
0.90	2.682381	2.680595	0.001786
0.95	2.695742	2.693926	0.001816
1.00	2.690309	2.688477	0.001832

Runge-Kutta

t_i	Exact	Estimation	Error
0.00	1.000000	1.000000	0.000000
0.05	1.102373	1.102373	0.000000
0.10	1.208974	1.208974	0.000000
0.15	1.318995	1.318995	0.000000
0.20	1.431583	1.431583	0.000000
0.25	1.545847	1.545847	0.000000
0.30	1.660847	1.660847	0.000000
0.35	1.775595	1.775595	0.000000
0.40	1.889052	1.889052	0.000000
0.45	2.000127	2.000127	0.000000
0.50	2.107672	2.107672	0.000000
0.55	2.210482	2.210482	0.000000
0.60	2.307287	2.307287	0.000000
0.65	2.396755	2.396755	0.000000
0.70	2.477484	2.477484	0.000000
0.75	2.548000	2.548000	0.000000
0.80	2.606754	2.606754	0.000000
0.85	2.652119	2.652119	0.000000
0.90	2.682381	2.682381	0.000000
0.95	2.695742	2.695742	0.000000
1.00	2.690309	2.690309	0.000000

Part IV
Code

```c
/*
        COLTON WILLIAMS 2017
        NUMERICAL ANALYSIS
        DIFFERENTIAL ESTIMATIONS
*/

// function used is f = y' = y - 4t^2 + 1 with y(0) = 1.0, on [0, 1]
// exact solution: y(t) = 4t^2 + 8t - 6e^t + 7, y(1.0) = 2.690309
// TODO allow passing of anonymous functions to estimations, not just f defined explicitly

#include <stdio.h>
#include <math.h>
```

```

double f(double t, double w)
{
        return w - 4 * t * t + 1;
}

double exact(double t)
{
        return 4 * t * t + 8 * t - (6 * exp(t)) + 7;
}

void euler(double a, double b, int steps, double init)
{
        double h = (b-a)/(double)steps;
        double t = a;
        double w = init;
        printf("(%f, %f)\t ERROR: %f\t\n", t, w, fabs(exact(t) - w));
        for (int i = 1; i <= steps; ++i)
        {
                w = w + h * f(t, w);
                t = a + i * h;
                printf("(%f, %f)\t ERROR: %f\t\n", t, w, fabs(exact(t) - w));
        }
}

void modified_euler(double a, double b, int steps, double init)
{
        double h = (b-a)/(double)steps;
        double t = a;
        double w = init;
        printf("(%f, %f)\t ERROR: %f\t\n", t, w, fabs(exact(t) - w));
        for (int i = 1; i <= steps; ++i)
        {
                w = w + h * 0.5 * (f(t, w) + f(a + i * h, w + h * f(t, w)));
                t = a + i * h;
                printf("(%f, %f)\t ERROR: %f\t\n", t, w, fabs(exact(t) - w));
        }
}

void runge_kutta(double a, double b, int steps, double
```

```
    init)
{
        double h = (b-a)/(double)steps;
        double t = a;
        double w = init;
        printf("(%f, %f)\t ERROR: %f\t\n", t, w, fabs(exact(t) - w));
        double k1, k2, k3, k4;
        for (int i = 1; i <= steps; ++i)
        {
                k1 = h * f(t, w);
                k2 = h * f(t + h / 2, w + k1 / 2);
                k3 = h * f(t + h / 2, w + k2 / 2);
                k4 = h * f(t + h, w + k3);
                w = w + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
                t = a + i * h;
                printf("(%f, %f)\t ERROR: %f\t\n", t, w, fabs(exact(t) - w));
        }
}

int main()
{
        euler(0.0, 1.0, 20, 1.0);
        modified_euler(0.0, 1.0, 20, 1.0);
        runge_kutta(0.0, 1.0, 20, 1.0);
        return 0;
}
```