Homework 14 – 5.9

# Section 5.9

## Problem 1

Use the Runge-Kutta method for system to approximate the solution of the following systems of first-order differential equations, and compare the results to the actual solutions.

Using a program:

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std;

// The functions.

double f1(double, double, double);

double f2(double, double, double);

// The correct values.

double U1(double);

double U2(double);

// K values for function 1.

double K11(double, double, double, double);

double K12(double, double, double, double, double, double);

double K13(double, double, double, double, double, double);

double K14(double, double, double, double, double, double);

//K values for function 2.

double K21(double, double, double, double);

double K22(double, double, double, double, double, double);

double K23(double, double, double, double, double, double);

double K24(double, double, double, double, double, double);

double RK4(double, double, double, double, double, double, double); // Runge Kutta Order 4.

int main()

{

// Create the file where the data will be stored.

ofstream outputFile;

outputFile.open("Runge-Kutta 2 System, h = 0.2.txt");

double a = 0, b = 1; // t boundary.

double h = 0.2; // Mesh size.

double w1 = 1, w2 = 1; // initial value.

double k11, k12, k13, k14, k21, k22, k23, k24; // K values.

int P = 10; // Precision.

double s = (b - a) / h; // Number of steps.

double t = a; // initial time.

// Heading for the file.

outputFile << showpoint << setprecision(P) << "Time" << " " << "W1(t)" << " " << "U1(t)" << " " << "W2(t)" << " " << "U2(t)" << endl;

outputFile << showpoint << setprecision(P) << t << " " << w1 << " " << U1(t) << " " << w2 << " " << U2(t) << endl; // Inital points.

for (int i = 0; i < s; i++)

{

// The K values.

k11 = K11(w1, w2, h, t);

k21 = K21(w1, w2, h, t);

k12 = K12(w1, w2, t, h, k11, k21);

k22 = K22(w1, w2, t, h, k11, k21);

k13 = K13(w1, w2, t, h, k12, k22);

k23 = K23(w1, w2, t, h, k12, k22);

k14 = K14(w1, w2, t, h, k13, k23);

k24 = K24(w1, w2, t, h, k13, k23);

// Finding U(t\_(i+1)).

w1 = RK4(w1, t, h, k11, k12, k13, k14);

w2 = RK4(w2, t, h, k21, k22, k23, k24);

t += h; // Increase the time.

outputFile << showpoint << setprecision(P) << t << " " << w1 << " " << U1(t) << " " << w2 << " " << U2(t) << endl; // Points.

}

outputFile.close(); // Close the file.

return 0;

}

// Function 1.

double f1(double u1, double u2, double t)

{

double u1p = 3 \* u1 + 2 \* u2 - (2 \* t \* t + 1) \* exp(2 \* t);

return u1p;

}

// Function 2.

double f2(double u1, double u2, double t)

{

double u2p = 4 \* u1 + u2 + (t \* t + 2 \* t - 4) \* exp(2 \* t);

return u2p;

}

// U1(t).

double U1(double t)

{

double U1 = (1. / 3.) \* exp(5 \* t) - (1. / 3.) \* exp(-1 \* t) + exp(2 \* t);

return U1;

}

// U2(t).

double U2(double t)

{

double U2 = (1. / 3.) \* exp(5 \* t) + (2. / 3.) \* exp(-1 \* t) + t \* t \* exp(2 \* t);

return U2;

}

//K1 values.

double K11(double w1, double w2, double h, double t)

{

double K11 = h \* f1(w1, w2, t);

return K11;

}

double K12(double w1, double w2, double t, double h, double K11, double K21)

{

double K12 = h \* f1(w1 + (K11 / 2), w2 + (K21 / 2), t + (h / 2));

return K12;

}

double K13(double w1, double w2, double t, double h, double K12, double K22)

{

double K13 = h \* f1(w1 + (K12 / 2), w2 + (K22 / 2), t + (h / 2));

return K13;

}

double K14(double w1, double w2, double t, double h, double K13, double K23)

{

double K14 = h \* f1(w1 + K13, w2 + K23, t + h);

return K14;

}

//K2 values.

double K21(double w1, double w2, double h, double t)

{

double K21 = h \* f2(w1, w2, t);

return K21;

}

double K22(double w1, double w2, double t, double h, double K11, double K21)

{

double K22 = h \* f2(w1 + (K11 / 2), w2 + (K21 / 2), t + (h / 2));

return K22;

}

double K23(double w1, double w2, double t, double h, double K12, double K22)

{

double K23 = h \* f2(w1 + (K12 / 2), w2 + (K22 / 2), t + (h / 2));

return K23;

}

double K24(double w1, double w2, double t, double h, double K13, double K23)

{

double K24 = h \* f2(w1 + K13, w2 + K23, t + h);

return K24;

}

// Runge-Kutta Method.

double RK4(double w, double t, double h, double K1, double K2, double K3, double K4)

{

double wi1 = w + (1. / 6.) \* (K1 + 2 \* K2 + 2 \* K3 + K4);

return wi1;

}

We get the following values:

Time W1(t) U1(t) W2(t) U2(t)

0.0000000000 1.000000000 1.000000000 1.000000000 1.000000000

0.2000000000 2.120365828 2.125008389 1.506991852 1.511587433

0.4000000000 4.441227756 4.465119613 3.242240207 3.265985279

0.6000000000 9.739133286 9.832358685 8.163416996 8.256295491

0.8000000000 22.67655977 23.00263945 21.34352778 21.66887674

1.000000000 55.66118088 56.73748265 56.03050296 57.10536209

## Problem 3.

Use the Runge-Kutta for Systems Algorithm to approximate the solution of the following higher-order differential equations, and compare the results to the actual solutions.

Using a Program:

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std;

// The functions.

double f1(double, double);

double f2(double, double, double);

double g(double);

// The correct values.

double Y(double);

// K values for function 1.

double K11(double, double, double);

double K12(double, double, double, double, double);

double K13(double, double, double, double, double);

double K14(double, double, double, double, double);

//K values for function 2.

double K21(double, double, double, double);

double K22(double, double, double, double, double, double);

double K23(double, double, double, double, double, double);

double K24(double, double, double, double, double, double);

double RK4(double, double, double, double, double, double, double); // Runge Kutta Order 4.

int main()

{

// Create the file where the data will be stored.

ofstream outputFile;

outputFile.open("Second Order DE RK4 System, h = 0.1.txt");

double a = 0, b = 1; // t boundary.

double h = 0.1; // Mesh size.

double y = 0, w2 = 0; // initial value.

double k11, k12, k13, k14, k21, k22, k23, k24; // K values.

int P = 8; // Precision.

double s = (b - a) / h; // Number of steps.

double t = a; // initial time.

// Heading for the file.

outputFile << showpoint << setprecision(P) << "Time" << " " << "W(t)" << " " << "Y(t)" << endl;

outputFile << showpoint << setprecision(P) << t << " " << y << " " << Y(t) << endl; // Inital points.

for (int i = 0; i < s; i++)

{

// The K values.

k11 = K11(w2, h, t);

k21 = K21(y, w2, h, t);

k12 = K12(w2, t, h, k11, k21);

k22 = K22(y, w2, t, h, k11, k21);

k13 = K13(w2, t, h, k12, k22);

k23 = K23(y, w2, t, h, k12, k22);

k14 = K14(w2, t, h, k13, k23);

k24 = K24(y, w2, t, h, k13, k23);

// Finding U(t\_(i+1)).

y = RK4(y, t, h, k11, k12, k13, k14);

w2 = RK4(w2, t, h, k21, k22, k23, k24);

t += h; // Increase the time.

outputFile << showpoint << setprecision(P) << t << " " << y << " " << Y(t) << endl; // Points.

}

outputFile.close(); // Close the file.

return 0;

}

// g(t).

double g(double t)

{

double g = t \* exp(t) - t;

return g;

}

// Function 1.

double f1(double u2, double t)

{

double u1p = u2;

return u1p;

}

// Function 2.

double f2(double u1, double u2, double t)

{

double u2p = 2 \* u2 - u1 + g(t);

return u2p;

}

// Y(t).

double Y(double t)

{

double Y = (1. / 6.) \* t \* t \* t \* exp(t) - t \* exp(t) + 2 \* exp(t) - t - 2;

return Y;

}

//K1 values.

double K11(double w2, double h, double t)

{

double K11 = h \* f1(w2, t);

return K11;

}

double K12(double w2, double t, double h, double K11, double K21)

{

double K12 = h \* f1(w2 + (K21 / 2), t + (h / 2));

return K12;

}

double K13(double w2, double t, double h, double K12, double K22)

{

double K13 = h \* f1(w2 + (K22 / 2), t + (h / 2));

return K13;

}

double K14(double w2, double t, double h, double K13, double K23)

{

double K14 = h \* f1(w2 + K23, t + h);

return K14;

}

//K2 values.

double K21(double w1, double w2, double h, double t)

{

double K21 = h \* f2(w1, w2, t);

return K21;

}

double K22(double w1, double w2, double t, double h, double K11, double K21)

{

double K22 = h \* f2(w1 + (K11 / 2), w2 + (K21 / 2), t + (h / 2));

return K22;

}

double K23(double w1, double w2, double t, double h, double K12, double K22)

{

double K23 = h \* f2(w1 + (K12 / 2), w2 + (K22 / 2), t + (h / 2));

return K23;

}

double K24(double w1, double w2, double t, double h, double K13, double K23)

{

double K24 = h \* f2(w1 + K13, w2 + K23, t + h);

return K24;

}

// Runge-Kutta Method.

double RK4(double w, double t, double h, double K1, double K2, double K3, double K4)

{

double wi1 = w + (1. / 6.) \* (K1 + 2 \* K2 + 2 \* K3 + K4);

return wi1;

}

We get that:

Time W(t) Y(t)

0.00000000 0.00000000 0.00000000

0.10000000 8.9724419e-06 8.9394967e-06

0.20000000 0.00015351945 0.00015350170

0.30000000 0.00083426835 0.00083433751

0.40000000 0.0028320548 0.0028323130

0.50000000 0.0074296778 0.0074302659

0.60000000 0.016561490 0.016562597

0.70000000 0.032996171 0.032998049

0.80000000 0.060558963 0.060561940

0.90000000 0.10440070 0.10440520

1.0000000 0.17132224 0.17132880