NUMERICAL ANALYSIS PROGRAMMING PROJECT DR. SONGMING HOU

JOHN EMORY

0. Introduction

Tom the Cat is chasing Jerry the Mouse, with an initial gap between them of 100m. Tom and Jerry's velocities are given as $v_c = 4 - at$ ms^{-1} and $v_m = v_{max} - ks = 3 - 0.02s$ ms^{-1}, respectively, with $0 < a$. The velocity of the change in the gap between Tom and Jerry, s, is given by $\frac{ds}{dt} = v_m - v_c = -1 - 0.02s + at$ ms^{-1}.

1. Problem

Find the true solution for when Tom will catch Jerry by plotting the gap distance.

First, we need to solve $\frac{ds}{dt}$. Noting that our equation is a linear first-order ODE, we need to put it into standard form:

$$\frac{ds}{dt} + 0.02s = at - 1$$

Next, we find the integration factor. Observing that in the second additive term on the left hand side we are multiplying by t^0, we see the integration factor is $e^{0.02t}$. This gives us the form:

$$\frac{d}{dt}s \cdot e^{0.02t} = (at - 1) \cdot e^{0.02t}$$

Taking the antiderivative of both sides gives:

$$\int \frac{d}{dt}s \cdot e^{0.02t}dt = a \cdot \int t \cdot e^{0.02t}dt - \int e^{0.02t}dt$$

$$s \cdot e^{0.02t} = 50at \cdot e^{0.02t} = 2500a \cdot e^{0.02t} - 50e^{0.02t} + c$$

Then, canceling $e^{0.02t}$ gives:

$$s = 50a(t - 50) - 50 + c \cdot e^{-0.02t}$$

Solving for c at our initial value of $s(0) = 100$ m will yield an equation we can use software to plot. Since $t = 0$, we have:

$$100 = -2500a - 50 + c \cdot e^{-0.02t}$$

$$c = 2500a + 150$$

So, our final equaiton we want to plot is:

$$s(a, t) = 50a(t - 50 + 50 \cdot e^{-0.02t}) + 150 \cdot e^{-0.02t} - 50$$

Date: February 18, 2019.

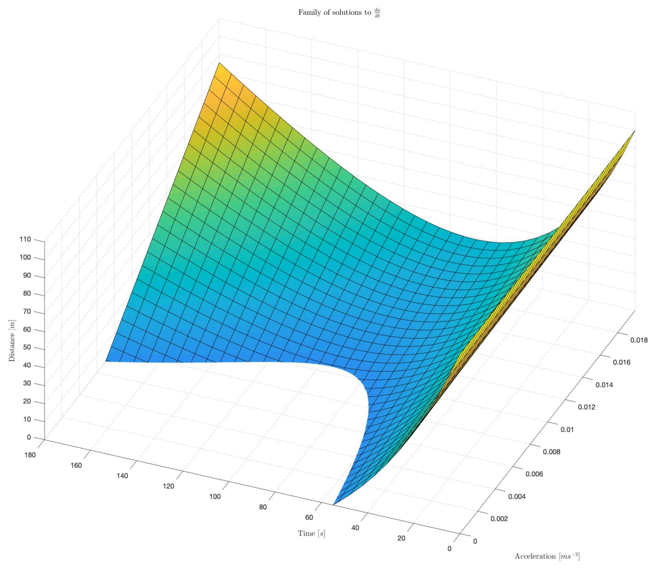

FIGURE 1. Plot of solutions to $\frac{ds}{dt}$

The exact solutions to when Tom catches Jerry are the points on the surface in Figure 1 that intersect with the plane at $s = 0$ with minimal t value. This can be seen in the figure as the curve traced by the surface where it intersects the bottom of the plot. The plot indicates that at $a = 0.01$ ms^{-1}, Tom will catch Jerry at roughly 82 seconds.

2. PROBLEM

For $a = 0.01$ ms^{-2}, use the fourth-order Runge-Kutta method to compute when Tom will catch Jerry. Use an appropriate step size to ensure an accurate result.

The source code for both Runge-Kutta and Adams-Bashforth is attached as Appendix 1, and at https://github.com/jfemory/numericalAnalysisFinal. From Table 1, the condensed output of the fourth-order Runge-Kutta calculations, we see that Tom catches Jerry between 81.5 and 82 seconds, when the sign of the distance changes to negative.

TABLE 1. Runge-Kutta Output

Time (s)	80.0	80.5	81.0	81.5	82.0	82.5	83
Distance (m)	0.3319	0.2303	0.1323	0.0377	-0.0535	-0.1413	-0.2257

3. PROBLEM

Use the Adams-Bashforth forth-order predictor-corrector to compute when Tom will catch Jerry using the results form Runge-Kutta, above, for the initial values of

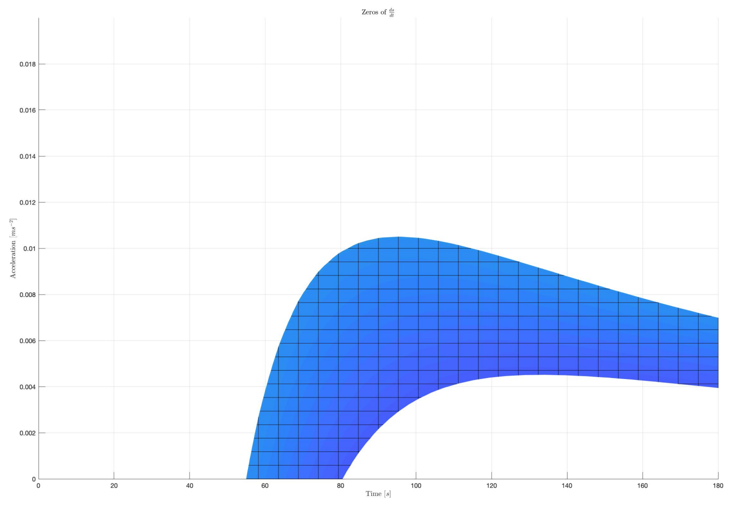

Figure 2. Time vs Acceleration Zeros

Adams-Bashforth.

Similar to the previous question using the Runge-Kutta technique, relevant values near the sign change for the Adams-Bashforth are given in Table 2.

Table 2. Adams-Bashforth Output

Time (s)	80.0	80.5	81.0	81.5	82.0	82.5	83
Distance (m)	0.3319	0.2303	0.1323	0.0377	-0.0535	-0.1413	-0.2257

It can be seen that the Tables 1 and 2 are in agreement up to four decimal places. The results between the two methods do in fact differ at higher precision. Please see Apendix 2 for a complete comparison at precision. The initial four values from Runge-Kutta were used as input for the Adams-Bashforth method.

4. Problem

Suppose Tom's acceleration is unknown. If Tom does not catch Jerry in 120s, is it possible that Tom will catch Jerry?

No. It is not possible. Figure 2 indicates the accelerations and times where the gap is less than zero. The border of the blue area indicates where the sign of the gap changes. So, we can see can see that any constant acceleration path (horizontal lines) that does not reduce the gap to 0m by 120s has no solutions to the right of 120s. In other words, any constant acceleration path that has a zero to the right of 120s also has a zero to the left of 120s. Therefore, no, if tom has not caught Jerry by 120s, Tom will never catch Jerry.

Appendix A. Runge-Kutta and Adams-Bashforth Source Code (Golang)

```
package main

import (
        "fmt"
)

type Entry struct {
        t float64
        w float64
}

//Initial conditions
var acc = 0.01
var a = 0.0
var b = 90.0
var N = 180
var alpha = 100.0
var h = (b - a) / float64(N)

func main() {
        outputRK := rungeKutta(a, alpha)
        fmt.Println(outputRK)
        outputAB := adamsBashforth(outputRK)
        fmt.Println("Runge-Kutta")
        printIt(outputRK)
        fmt.Println("Adams-Bashforth")
        printIt(outputAB)
}

//printIt takes an array of entries, and prints them to stdo
//in a format that can easily be copied out
func printIt(in []Entry) {
        for i := 0; i < len(in); i++ {
                fmt.Println(in[i].t, in[i].w)
        }
}

//adamsBashforth takes an array of t and w values, takes the
//first four values, and returns an array of AB values
func adamsBashforth(in []Entry) []Entry {
        output := make([]Entry, 4)
        for i := 0; i < 4; i++ {
                output[i] = in[i]
        }
        for j := 4; j < N; j++ {
                var z Entry
                w0, w1, w2, w3 := in[j-4], in[j-3], in[j-2], in[j-1]
                //Update t
                z.t = a + float64(j)*h
                //Predict new w value
                z.w = w3.w + ((h * ((55 * f(w3.t, w3.w)) -
                        (59 * f(w2.t, w2.w)) + (37 * f(w1.t, w1.w)) -
                        (9 * f(w0.t, w0.w)))) / 24)
                //Correct new w value
                z.w = w3.w + ((h * ((9 * f(z.t, z.w)) +
                        (19 * f(w3.t, w3.w)) - (5 * f(w2.t, w2.w)) +
                        f(w1.t, w1.w))) / 24)
                output = append(output, z)
        }
        return output
}

//rungeKutta takes an initial t and w and returns a slice based on
//the global step size of t and w
func rungeKutta(t0 float64, w0 float64) []Entry {
        output := make([]Entry, 0)
        t := t0
        w := w0
        output = append(output, Entry{t, w})
        for i := 1; i < N; 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 + float64(i)*h
                output = append(output, Entry{t, w})
        }
        return output
}

//func f is the function being evaluated by the numerical
//methods considered.
func f(t float64, w float64) float64 {
        return (-1 - 0.02*w + (acc * t))
}
```

Appendix B. Figure 1 Source Code (Matlab)

```
syms a t
s = a*(50*t - 2500 + exp(-0.02*t)*2500)+exp(-0.02*t)*150-50;
fs = fsurf(s, [0.0 0.02 0 180]);
zlim([0,110]);
title('Family␣of␣solutions␣to␣$\frac{ds}{dt}$','Interpreter','latex')
xlabel('Acceleration␣$[ms^{-2}]$','Interpreter','latex')
ylabel('Time␣$[s]$','Interpreter','latex')
zlabel('Distance␣$[m]$','Interpreter','latex')
```

Appendix C. Figure 2 Source Code (Matlab)

```
syms a t
s = a*(50*t - 2500 + exp(-0.02*t)*2500)+exp(-0.02*t)*150-50;
fs = fsurf(s, [0.0 0.02 0 180]);
zlim([-20,0]);
%rotate to where you see the projection onto the a,t-plane.
title('Zeros of $\frac{ds}{dt}$','Interpreter','latex')
xlabel('Acceleration $[ms^{-2}]$','Interpreter','latex')
ylabel('Time $[s]$','Interpreter','latex')
zlabel('Distance $[m]$','Interpreter','latex')
```

Appendix D. Extended Tables

Time (s)	Distance (m)
0	100
0.5	98.50872090625
1	97.03476782897047
1.5	95.57796837141707
2	94.13815185222172
...	...
75	1.5477780308970484
75.5	1.409246147796525
76	1.2745802215262776
76.5	1.1437417851731293
77	1.0166927545760849
77.5	0.8933954245178823
78	0.7738124649544401
78.5	0.6579069172818212
79	0.545642190640342
79.5	0.4369820582554564
80	0.3318906538150474
80.5	0.23033246788276646
81	0.13227234434706014
81.5	0.03767547690552964
82	-0.0534925944157284
82.5	-0.14126598670814805
83	-0.22567847759243165
83.5	-0.30676350859634005
84	-0.38455418849887313
84.5	-0.4590832966411755
85	-0.5303832862044977
85.5	-0.5984862874555416
86	-0.6634241109595136
86.5	-0.725228250761208
87	-0.7839298875344373
87.5	-0.8395598917001258
88	-0.8921488265133776
88.5	-0.941726951119827
89	-0.988324223581579
89.5	-1.0319703038730401

(A) Runge-Kutta

Time (s)	Distance (m)
0.0	100
0.5	98.50872090625
1.0	97.03476782897047
1.5	95.57796837141707
2.0	94.13815185160837
...	...
75.0	1.5477780307546034
75.5	1.4092461476554976
76.0	1.2745802213866533
76.5	1.1437417850348943
77.0	1.0166927544392255
77.5	0.8933954243823846
78.0	0.7738124648202905
78.5	0.6579069171490064
79.0	0.5456421905088489
79.5	0.43698205812527163
80.0	0.331890653686158
80.5	0.23033246775515948
81.0	0.13227234422072293
81.5	0.03767547678044947
82.0	-0.05349259453956401
82.5	-0.14126598683075153
83.0	-0.22567847771381516
83.5	-0.30676350871651575
84.0	-0.3845541886178531
84.5	-0.4590832967589716
85.0	-0.5303832863211217
85.5	-0.5984862875710052
86.0	-0.6634241110738284
86.5	-0.7252282508743852
87.0	-0.7839298876464884
87.5	-0.839559891811062
88.0	-0.8921488266232099
88.5	-0.9417269512285665
89.0	-0.9883242236892366
89.5	-1.0319703039796264

(B) Adams-Bashforth

Table 3. Extended Output for Runge-Kutta and Adams-Bashforth

Program of Mathematics and Statistics, Louisiana Tech University
Email address: `jfe004@latech.edu`