

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 $v_c = 4 - at \text{ ms}^{-1}$ and $v_m = v_{max} - ks = 3 - 0.02s \text{ ms}^{-1}$, respectively, with $0 < a$. The velocity of the change in the gap between Tom and Jerry, s , is given by $\frac{ds}{dt} = v_m - v_c = -1 - 0.02s + at \text{ ms}^{-1}$.

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:

$$\begin{aligned} \int \frac{d}{dt}s \cdot e^{0.02t} dt &= a \cdot \int t \cdot e^{0.02t} dt - \int e^{0.02t} dt \\ s \cdot e^{0.02t} &= 50at \cdot e^{0.02t} = 2500a \cdot e^{0.02t} - 50e^{0.02t} + c \end{aligned}$$

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

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

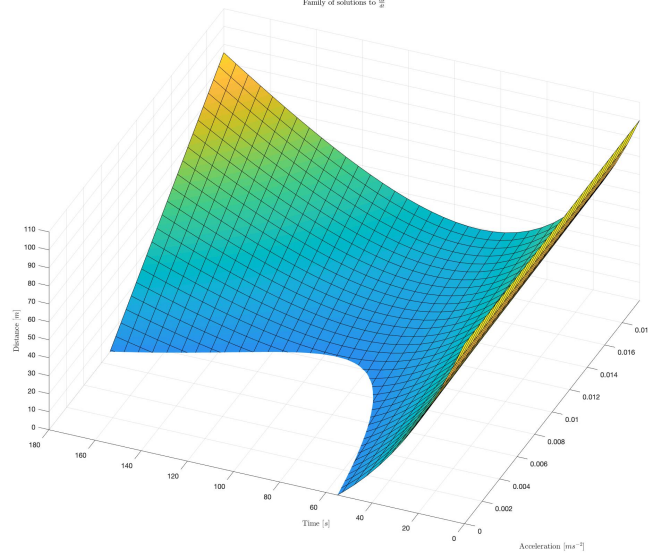
Solving for c at our initial value of $s(0) = 100 \text{ m}$ will yield an equation we can use software to plot. Since $t = 0$, we have:

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

$$c = 2500a + 150$$

So, our final equation we want to plot is:

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

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 $a = 0.01 \text{ ms}^{-2}$, Tom will catch Jerry at roughly 82 seconds.

2. PROBLEM

For $a = 0.01 \text{ ms}^{-2}$, 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

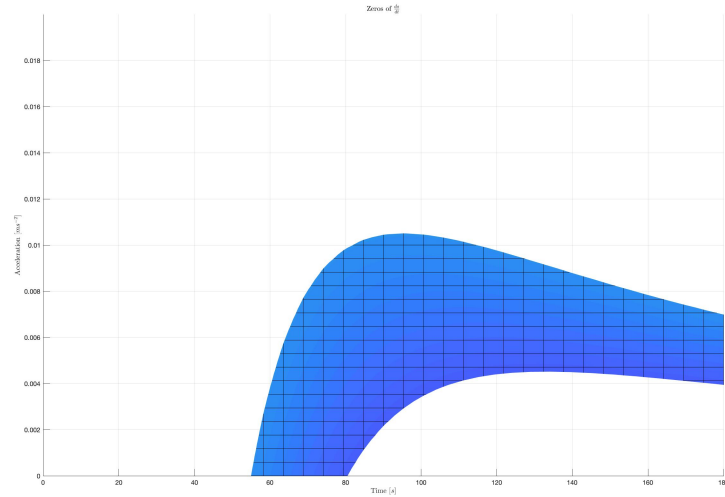


FIGURE 2. Time vs Acceleration Zeros

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 Appendix 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 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. 3D PLOT 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. 2D PLOT 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. COMPLETE TABLES

PROGRAM OF MATHEMATICS AND STATISTICS, LOUISIANA TECH UNIVERSITY
Email address: `jfe004@latech.edu`