

# THE PAPER!!!!

*by* Teifion Gibson

---

**Submission date:** 25-Nov-2024 01:44PM (UTC+0000)

**Submission ID:** 243653135

**File name:** 50533\_Teifion\_Gibson\_THE\_PAPER\_\_628791\_174352134.docx (363.17K)

**Word count:** 1339

**Character count:** 7566

# Integration methods and applications in gaming

By Teifion Gibson

Subject : Games Architecture and Mechanics

Student number : 2304294

## **References**

|                      |   |
|----------------------|---|
| Introduction .....   | 2 |
| Section 1 .....      | 2 |
| Integration 1 .....  | 2 |
| Integration 2 .....  | 3 |
| Uses in gaming ..... | 4 |
| Section 2 .....      | 5 |
| Conclusion .....     | 8 |
| References .....     | 9 |

## **Introduction**

Within this paper, we will be discussing some basic integration methods and additionally, we will cover their applications in a gaming context, then in the second part of this paper I will be creating code for a friction calculation.  2

## **Section 1**

Numerical integration methods are algorithms, which according to Wikipedia "numerical integration comprises a broad family of algorithms for calculating the numeric  3 ue of a definite integral" [1].

Numerical integration methods are commonly used for physics calculations in video games[2], numerical integrations are used for estimating the value of a value when it is difficult to find the intended value[3]. In the section below will discuss two integration methods, as well as commonly used integrations in gaming.

## **Integration 1**

The first integration we will be discussing is the Midpoint rule, the midpoint rule defined as :

$$(x_m, y_m) = \left( \frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right)$$

Fig.1 Midpoint rule definition [4]

The midpoint rule finds the halfway point along a line, the advantage to using the midpoint rule is that it gives a good value for a line between two points, however it is ineffective for a curve as it doesn't factor in the change in gradient, as the integration loses accuracy[5].

An application for the usage of the midpoint rule is that it allows us to calculate the halfway point on a line. An example usage of this is to find the center of a shape with a straight line such as a circle [9]. This is helpful as it allows us to find the CenterPoint of a shape, even when its size has been changed. This is especially helpful, as for collision calculations we want to find the CenterPoint of an object, if possible, but when an object is created it will use a default position for the object, which is usually in the corner of the square image. Making usage of this integration allows us to define a Centrepoint for the calculations.

## Integration 2

The second integration we will discuss is the trapezoidal rule. The trapezoidal rule is defined as:

*The formula:*

$$\int_a^b y \, dx \approx \frac{1}{2}h\{(y_0 + y_n) + 2(y_1 + y_2 + \dots + y_{n-1})\}, \text{ where } h = \frac{b-a}{n}$$

Fig.2 Trapezium rule formula [5]

The trapezium rule is used for the calculation of the area under a curve [9]. The rule does this by defining the area under a graph as trapezoids, and then defining the area of these shapes [9]. This rule assists us in the approximation of integrals.

A negative to the usage of the trapezium rule is that for a declining gradient, the rule will always give an underestimate, while with an inclining gradient the integration will always give an overestimate [5].

A common usage of the trapezium rule is for the estimation of volumes such as the calculation of rainfall [10].

### Uses in gaming

Integrals have many different uses in gaming, one usage of numerical integration in gaming is the calculation of realistic physics [6], a method of integration that is commonly used in game development is the Euler calculations which are used to calculate the velocity and speed of a object in a direction after a time[6]. (example of equation below)

$$\text{Velocity}_{n+1} = \text{Velocity}_n + (\text{Acceleration}_n * dt)$$

Fig.3 Calculation of velocity [6]

This example integration can then be used to calculate the position change, such as a object moving on a 2d plane[6]

$$\text{Position}_{n+1} = \text{Position}_n + (\text{Velocity}_n * dt)$$

Fig.4 Change in position [6]

These calculations can be applied to a program to make more realistic physics calculations and improve the games feel, and immersion.

Euler calculations are often used for rigid body physics calculations as the calculation does not consider that the objects change shape/ deform.

Another commonly used numerical integration method is Verlet integration, Verlet integration can be used to simulate physics such as cloth physics [7]. Verlet integration is used frequently as it is more efficient and less resource intensive to implement compared to other popular integration methods [8].

$$x(t + dt) = 2x(t) - x(t - dt) + a(t)dt^2$$

Fig.5 Calculation for Verlet integration [7]

However, Verlet integration has some disadvantages that need to be planned around for usage in games development, one is that extra implementation is needed for velocity to affect the integration [6].

## Section 2

Within this section of the report, we will be creating and discussing implementation of integration and using it to create a calculation for friction the program then should generate outputs based on this calculation, which is stated in the assignment brief as follows:

$$F_d = C_r mg + \frac{1}{2} C_D \rho A V^2$$

Fig. 6 integration from brief

$F_d$  is the forward motion of the car in movement, and additionally this integration factors in the rolling distance, wind resistance, and any breaking force on the car. We then need to ensure the program can calculate the velocity as we are not given this. Additionally, the program needs to calculate the distance travelled by the car and output the distance travelled in meters and the time in seconds. At this data will then need to be calculated and graphed in a table.

Additionally, if the distance travelled is greater than 6 meters, then  $t_f$  needs to be set to 0. And if velocity is less than or = to 0 then  $F_d$  needs to be set to 0. For this calculation as it simulates friction on a straight line, we do not need to calculate the y axis only the x axis

The code is outputted into an external file and then plotted onto a graph, this graph is attached below

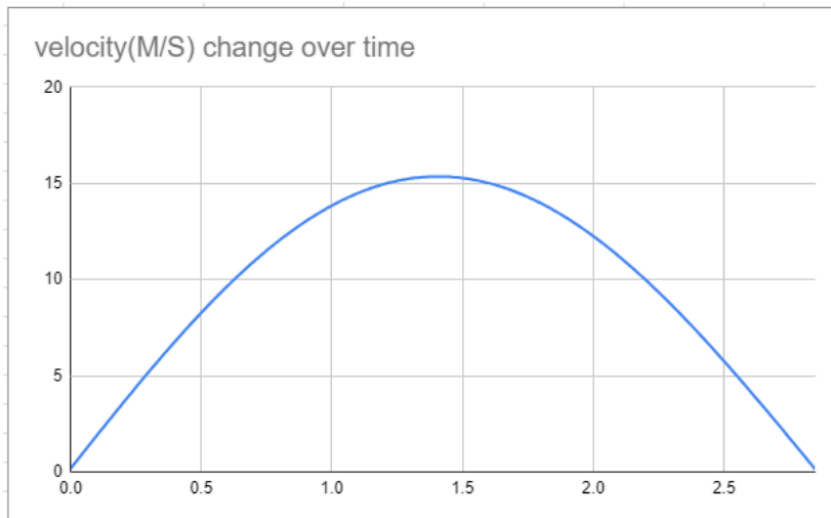


Fig. 7 velocity change over time

Additionally, the data is also stored within a spreadsheet, a picture is shown below

|    | A       | B           | C             |
|----|---------|-------------|---------------|
| 1  | time(s) | Position(M) | Velocity(M/S) |
| 2  | 0       | 0.0017      | 0.1714        |
| 3  | 0.01    | 0.0051      | 0.3428        |
| 4  | 0.02    | 0.0103      | 0.5141        |
| 5  | 0.03    | 0.0171      | 0.6854        |
| 6  | 0.04    | 0.0257      | 0.8566        |
| 7  | 0.05    | 0.036       | 1.0276        |
| 8  | 0.06    | 0.048       | 1.1986        |
| 9  | 0.07    | 0.0617      | 1.3694        |
| 10 | 0.08    | 0.0771      | 1.54          |
| 11 | 0.09    | 0.0942      | 1.7104        |
| 12 | 0.1     | 0.113       | 1.8806        |
| 13 | 0.11    | 0.1335      | 2.0506        |
| 14 | 0.12    | 0.1557      | 2.2203        |
| 15 | 0.13    | 0.1796      | 2.3898        |
| 16 | 0.14    | 0.2052      | 2.5589        |
| 17 | 0.15    | 0.2324      | 2.7277        |
| 18 | 0.16    | 0.2614      | 2.8961        |
| 19 | 0.17    | 0.292       | 3.0642        |
| 20 | 0.18    | 0.3244      | 3.2319        |
| 21 | 0.19    | 0.3584      | 3.3992        |
| 22 | 0.2     | 0.394       | 3.5661        |
| 23 | 0.21    | 0.4313      | 3.7325        |

Fig. 8 table of calculated values

This data is counted until velocity reaches 0

Now it is time to evaluate this model and discuss its issues and advantages. With this integration model it allows for a decently accurate calculation of friction as well as velocity in a 2d plane. However, this calculation does not take into account some factors, one of these is if the car is on a non-level plane, as if the ground is not flat the data will have a different result output.

The code is structured as follows, first the values for the calculation are declared, next the code is programmed to loop until time exceeds 5 seconds or if velocity is equals to or is less than 0. The calculation, first calculated  $F_d$ , then  $F_t$ , then acceleration, this is then used to calculate the velocity. This value is finally used to calculate the position. Additionally, the code on each loop writes the values of position, velocity and time to an external file.

```

#include <iostream>
#include <math.h>
#include "filewriter.h"

int main()
{
    FileWriter fw("file.csv");
    //this is for the calculation of velocity
    const float g = 9.81;
    float velocity = 0.0, position = 0, deltaTime = 0.01;

    float Fd, Ft, acceleration, Cr = 0.21, M = 1 /kg, Cd = 0.34, airDensity = 1.2 /kg/m^3, A = 0.01; //values for calculation of forward distance

    //calculation for velocity and also writes to velocity, this loop can also be used for the full calculation
    for (float time = 0.0; time <= 5.0; time += deltaTime)
    {
        Fd = Cr * M * g + 0.5 * (Cd * airDensity * A * pow(velocity, 2)) / 2; // calc forward distance;
        Ft = 1.2 * (16 - position);
        acceleration = (Ft - Fd) / M; //calc for acceleration;

        velocity = velocity + (acceleration * deltaTime); //calculates velocity
        position = position + (velocity * deltaTime); //calculates position change
        if (velocity <= 0)
        {
            Fd = 0;
            break;
        }
        std::cout << "time: " << time << " Position: " << position << "M Velocity: " << velocity << std::endl;

        if (position == 0)
            break;
        fw.write(time, position, velocity); // writes time, position and velocity to a file.
    }
}

```

Fig. 9 code for the friction calculation

```

#pragma once

#include <string>
#include <iostream>
#include <fstream>
#include <iomanip>

class FileWriter //this writes the results of the code to a output file.
{
private:
    std::string filepath;
    std::ofstream file;

public:
    FileWriter(const std::string& filepath) : filepath(filepath)
    {
        file.open(filepath);
        if (!file.is_open())
        {
            std::cerr << "error: could not open file for writing. \n";
        }
        else
        {
            file << "time(s), Position(M), Velocity(M/S), \n";
        }
    }

    ~FileWriter()
    {
        if (!file.is_open())
        {
            file.close();
        }
        std::cout << "Simulation is completed. Results were saved to: " << filepath;
    }

    void write(float time, float position, float velocity)
    {
        if (file.is_open())
        {
            file << std::fixed << std::setprecision(2) << time << ", "
                << std::fixed << std::setprecision(4) << position << ", "
                << std::fixed << std::setprecision(4) << velocity << "\n";
        }
    }

    void log(float time, float position, float velocity)
    {
        std::cout << "time: " << time << "s position: " << position << "M velocity" << velocity << std::endl;
    }
};

```

Fig. 10 FileWriter code

## **Conclusion**

In conclusion this essay has covered 2 integration methods, these being verlet integration and trapezium rule. Additionally  we have discussed a few integration methods and their applications in a gaming context. Within the second segment of this paper we have discussed the integration from the brief, calculated it using a program and evaluated and analysed the code written.

## References

[1]

A. Al-Naib, "Numerical Integration," *researchgate.net*, Jun. 08, 2019.  
[https://www.researchgate.net/publication/333661710\\_Numerical\\_Integration](https://www.researchgate.net/publication/333661710_Numerical_Integration) (accessed Nov. 21, 2024).

[2]

G. Pezzi, "Fundamental Math for Game Developers," *pikuma.com*, Jan. 31, 2023.  
<https://pikuma.com/blog/math-for-game-developers> (accessed Nov. 21, 2024).

[3]

G. Strang, "2.5: Numerical Integration - Midpoint, Trapezoid, Simpson's rule," *Mathematics LibreTexts*, Sep. 19, 2018.  
[https://math.libretexts.org/Courses/Mount\\_Royal\\_University/MATH\\_2200%3A\\_Calculus\\_for\\_Scientists\\_II/2%3A\\_Techniques\\_of\\_Integration/2.5%3A\\_Numerical\\_Integration\\_-\\_Midpoint%2C\\_Trapezoid%2C\\_Simpson%27s\\_rule](https://math.libretexts.org/Courses/Mount_Royal_University/MATH_2200%3A_Calculus_for_Scientists_II/2%3A_Techniques_of_Integration/2.5%3A_Numerical_Integration_-_Midpoint%2C_Trapezoid%2C_Simpson%27s_rule) (accessed Nov. 21, 2024).

[4]

A. Barta, "Midpoint Rule | Definition, Formula & Examples," *Study.com*, 2022.  
<https://study.com/learn/lesson/midpoint-rule-formula-examples.html> (accessed Nov. 21, 2024).

[5]

theChalkface.net, "Numerical Integration," Aug. 2016. Accessed: Nov. 21, 2024. [Online].  
 Available: [https://www.thechalkface.net/resources/Numerical\\_Integration.pdf](https://www.thechalkface.net/resources/Numerical_Integration.pdf)

[6]

jdickinson91, "Numerical Integration in Games Development," *Understanding Games Development*, Jan. 22, 2015. <https://jdickinsongames.wordpress.com/2015/01/22/numerical-integration-in-games-development-2/> (accessed Nov. 21, 2024).

[7]

M. Pekár, "Verlet Integration and Cloth Physics Simulation," *Pikuma.com*, 2022.  
<https://pikuma.com/blog/verlet-integration-2d-cloth-physics-simulation> (accessed Nov. 21, 2024).

[8]

cuemath, "Midpoint Formula - How to Find Midpoint? Examples, Formulas," *Cuemath*.  
<https://www.cuemath.com/geometry/midpoint/> (accessed Nov. 21, 2024).

[9]

BYJU'S, "Trapezoidal Rule for Integration (Definition, Formula, and Examples)," *BYJU'S*, Jul. 22, 2020. <https://byjus.com/maths/trapezoidal-rule/> (accessed Nov. 21, 2024).

[10]

mathspace, "5.07 The trapezoidal rule | Year 11 Maths | NSW Mathematics Standard 11 - 2020 Edition," *Mathspace*. <https://mathspace.co/textbooks/syllabuses/Syllabus-812/topics/Topic-18337/subtopics/Subtopic-246433/?activeTab=theory> (accessed Nov. 21, 2024).

# THE PAPER!!!!

## GRADEMARK REPORT

FINAL GRADE

53/100

GENERAL COMMENTS

Engagement with Literature: 10/15

Knowledge and Understanding: 10/15

Cognitive and Intellectual Skills: 10/20

Practical Skills: 20/45

Transferable Skills: 3/5

Overall 53/100

Comment:

A good variety of references

Overall layout was a bit rough and could do with some improvements and a bout of proofreading.

Your code has a fatal flaw, you have limited the loop to 5 seconds.

This calculation for  $f_t$  also suffered a flaw of it constantly applying the force of the spring for the entire duration of the simulation, where you were meant to set tractive force to 0 after the first 6 meters covered, to simulate the spring.

This is also why your simulation ended in <3 seconds rather than taking the usual 6~

You also did not discuss/implement a second NI method, such as RK4 as outlined in the brief.

See inline comments also

PAGE 1

PAGE 2



### Comment 1

Does this mean to say table of contents?

PAGE 3



### Comment 2

In academic writing, particularly in scientific and technical papers, the use of the first person (e.g., "I",

"we") is generally discouraged as it can make the writing appear subjective and less formal. The focus of academic work should be on the research, analysis, or findings rather than the researcher. Using an objective tone ensures that the argument or evidence takes precedence over personal opinions or experiences, which enhances the perceived credibility and professionalism of the work. For example, instead of saying, "I conducted the experiment," you could write, "The experiment was conducted," which maintains an impersonal and objective stance. To improve in this area, aim to use passive voice judiciously or rephrase sentences to emphasise the action rather than the actor. Additionally, consider adopting phrases like "the research indicates" or "it was observed" to convey findings without self-referencing. Reviewing published papers in your field can provide valuable examples of how to maintain an objective tone while ensuring clarity and precision.



### Comment 3

Wikipedia, while a valuable starting point for general information and preliminary research, is not considered a reliable academic source. The main issue lies in its open-edit nature, where content can be created or modified by anyone, potentially leading to inaccuracies or biases. Furthermore, the information is often unsourced or lacks the rigorous peer review required for scholarly work. Academic standards require the use of credible, authoritative sources such as peer-reviewed journal articles, books, and reputable industry publications. Instead of citing Wikipedia directly, it is better to use it as a guide to locate primary or secondary sources through its references section. This approach ensures your work is based on verifiable and credible evidence, maintaining the academic integrity and reliability of your arguments.

PAGE 4

---

PAGE 5

---



### Comment 4

typo



### Comment 5

here it would have been more professional to say see fig.3 as you have given it a label, thus a means of navigation

PAGE 6

---

PAGE 7

---

PAGE 8

---

PAGE 9

---



### Comment 6

typo and it was a report!

an essay is subjective, a report is objective. Let me know if you want my notes on the definition/best practices for each.

PAGE 10

---