Kinematics Equation Solver Documentation

### Introduction

This C++ program is designed to solve basic kinematic equations based on user inputs. It allows users to perform one of the following calculations:

1. \*\*Final Velocity\*\* given initial velocity, acceleration, and time.

2. \*\*Displacement\*\* given initial velocity, acceleration, and time.

3. \*\*Final Velocity\*\* given initial velocity, acceleration, and displacement.

The program uses a class called `Kinematics` that contains functions implementing the kinematic formulas. The user is prompted to select a calculation type and then provide the necessary inputs.

### Purpose of the Program

The purpose of this program is to help users calculate important kinematic values, specifically:

- Final velocity after a certain time with constant acceleration.

- Displacement after a given time with constant acceleration.

- Final velocity based on displacement and acceleration.

### Code Structure

#### 1. \*\*Header Files:\*\*

```cpp

#include <iostream>

#include <cmath>

```

- \*\*`<iostream>`\*\*: This allows for input/output operations, such as reading user inputs and displaying outputs.

- \*\*`<cmath>`\*\*: This is required to perform mathematical operations, such as calculating the square root using `sqrt()`.

#### 2. \*\*Kinematics Class:\*\*

```cpp

Class Kinematics {

Public:

// Function to calculate final velocity: v\_f = v\_i + a \* t

Static double calculateFinalVelocity(double initialVelocity, double acceleration, double time);

// Function to calculate displacement: s = v\_i \* t + 0.5 \* a \* t^2

Static double calculateDisplacement(double initialVelocity, double acceleration, double time);

// Function to calculate final velocity given displacement: v\_f^2 = v\_i^2 + 2 \* a \* s

Static double calculateFinalVelocityWithDisplacement(double initialVelocity, double acceleration, double displacement);

};

```

- The \*\*`Kinematics`\*\* class is created to encapsulate the kinematic formulas.

- All the functions in the class are static, meaning they can be called without creating an instance of the class.

#### 3. \*\*Functions:\*\*

- \*\*`calculateFinalVelocity(double initialVelocity, double acceleration, double time)`\*\*:

- Formula used:

\[

V\_f = v\_i + a \cdot t

\]

* This function calculates the final velocity after a given time with constant acceleration.

- \*\*`calculateDisplacement(double initialVelocity, double acceleration, double time)`\*\*:

- Formula used:

\[

S = v\_i \cdot t + \frac{1}{2} a \cdot t^2

\]

* This function calculates the displacement of an object after a given time with constant acceleration.

- \*\*`calculateFinalVelocityWithDisplacement(double initialVelocity, double acceleration, double displacement)`\*\*:

- Formula used:

\[

V\_f^2 = v\_i^2 + 2 \cdot a \cdot s

\]

* This function calculates the final velocity when displacement is given, using the kinematic formula.

#### 4. \*\*Main Program:\*\*

```cpp

Int main() {

Double initialVelocity, acceleration, time, displacement;

Int choice;

Cout << “Kinematics Equation Solver\n”;

Cout << “Choose a calculation:\n”;

Cout << “1. Final velocity (given initial velocity, acceleration, and time)\n”;

Cout << “2. Displacement (given initial velocity, acceleration, and time)\n”;

Cout << “3. Final velocity (given initial velocity, acceleration, and displacement)\n”;

Cout << “Enter your choice (1/2/3): “;

Cin >> choice;

```

The user is prompted to choose one of the following options:

- Option 1: Calculate final velocity using time.

- Option 2: Calculate displacement using time.

- Option 3: Calculate final velocity using displacement.

Based on the choice, the corresponding case in the `switch` block is executed.

For each choice, the program asks for necessary input parameters such as initial velocity, acceleration, time, or displacement. It then calls the respective function from the `Kinematics` class to compute the result.

### Sample Input/Output

\*\*Example 1: Calculating Final Velocity (Option 1)\*\*

- User input:

- Initial velocity: 10 m/s

- Acceleration: 2 m/s²

- Time: 5 seconds

- Output:

```

Final velocity: 20 m/s

```

\*\*Example 2: Calculating Displacement (Option 2)\*\*

- User input:

- Initial velocity: 5 m/s

- Acceleration: 1 m/s²

- Time: 10 seconds

- Output:

```

Displacement: 100 m

```

### Problems Faced During Development

1. \*\*Lack of C++ Experience\*\*:

- I had limited experience programming in C++, making it difficult to initially understand certain syntax and structures such as classes, static methods, and object-oriented concepts in C++.

- While I have programming experience in C, C++ introduces concepts like classes, which required additional learning to understand and implement.

2. \*\*Tight Time Constraints\*\*:

- The task was assigned with limited time for completion, so balancing time for learning C++ syntax and developing a working solution was a challenge.

- Time pressure increased the difficulty of ensuring all aspects of the program worked correctly and efficiently.

3. \*\*Difficulty of Task\*\*:

- The difficulty level of this task was relatively high for someone new to C++. Implementing kinematic equations required understanding both the mathematical formulas and translating them into code accurately.

- Debugging and testing the program to ensure that it handled a variety of inputs correctly also added complexity.

4. \*\*External Help\*\*:

- To complete the task, I utilized external help from \*\*ChatGPT\*\* and \*\*Google\*\*. ChatGPT provided guidance on structuring the code and using classes in C++, while Google was useful for understanding kinematic formulas and C++ syntax.

- This combination of resources helped accelerate my learning and enabled me to solve the problems efficiently.

### Conclusion

This C++ program demonstrates the application of kinematic formulas in solving real-world physics problems related to motion. The program allows the user to calculate final velocity or displacement based on different sets of known parameters. Despite the challenges faced during development, the task was a great opportunity to learn C++ and apply mathematical concepts in programming.

### Future Improvements

- \*\*Error Handling\*\*: Add input validation to handle invalid or incorrect inputs more gracefully.

- \*\*Modular Design\*\*: Extend the program to handle more complex kinematic problems, such as 2D or 3D motion.

- \*\*Additional Features\*\*: Implement more kinematic calculations, such as calculating time given displacement and velocity.