Analysis of Numerical Methods in Mechanical and Aerospace Engineering (December 2018)

Rahul Shridhar, Harrisburg University of Science and Technology

***Abstract*—In this paper, I describe the implementation of numerical methods used to calculate the upward velocity of the rocket. I explain the derivation of the formula to calculate this velocity using integrals which gives us the mathematical significance of this problem in the field of aerospace engineering. I detail the performance of the different numerical methods using practical observations and determine the most efficient method to solve our scientific problem in terms of quickness of convergence and accuracy of computation.**

I. Introduction

The thrust of a rocket can be modeled from the generalization of Newton's second law of motion. This law states that the force acting on an object is equal to the mass of the object times its acceleration. We will derive the relation between the upward velocity of the rocket as a function of time. I will follow that up with calculating the time required for the rocket to reach a designated value by running them through a series of numerical methods.

In this paper, I will describe the positives and negative impacts of applying these numerical methods in the context of finding the upward velocity of the rocket. This answers the primary motivation of this paper which is to understand why a particular numerical method performs better in certain situations as compared to the others. Further, I will delve deeper into the analysis of determining the best possible numerical method to solve this particular problem. The remainder of this paper is organized as follows. Section 2 describes the equipment that was employed for performing this research. Section 3 describes in detail the methodology that was used in this problem. Section 4 provides the results based on the outputs of the numerical methods. Before concluding, Section 5 discusses the observations and provides a gateway in determining the best numerical method to solve our problem.

II. equipment

The primary tool used to develop programs for the numerical methods was PyCharm. PyCharm is an integrated development environment (IDE) that is used to write computer programs in Python. It was originally developed by the Czech company JetBrains in 2010. PyCharm has stable releases for the Windows, MacOS and Linux environment. The development for this research paper was done using Python 3. PyCharm has a nice graphical debugger and provides code analysis with an integrated unit testing framework. The primary research for solving this Aerospace Engineering problem was done using the book *Numerical Methods for Engineers: Seventh Edition* co-authored by *Steven C. Chapra* and *Raymond P. Canale*.

III. Problem analysis

We research the scenario of rocket propulsion in this problem. We derive an analysis to calculate the velocity of the rocket that is launched from the earth's surface. Our final goal is to figure out the velocity of a rocket at time t. This would depend on a number of factors like fuel consumption rate and the initial mass of the rocket. We derive the equation that gives us the upward velocity of the rocket considering all the important factors pertinent to this problem.

A. Derivation

We know that the net external force acting on an object can be calculated as the rate of change of momentum. From Newton's second law of motion, we also know the acceleration of an object is produced by a net force that is directly proportional to the magnitude of the net force, in the same direction as the net force, and inversely proportional to the mass of the object.

$$F = ma \quad (1)$$

The acceleration of a body can be defined as the rate of change of velocity per unit of time.

$$a = dv/dt \quad (2)$$

Therefore, we can insert 2) into 1).

$$F = m\frac{dv}{dt} = dp/dt \quad (3)$$

The average force on a constant mass system is equal to the rate of change of momentum. This relationship can be generalized by calculus methods to include instantaneous rates of change as follows:-

$$F = \frac{dp}{dt} = m\frac{dv}{dt} + v\frac{dm}{dt} \quad (4)$$

This formulation of the force relationship allows varying mass that would be necessary for calculating velocity during rocket propulsion. We get to 4) using the product rule for derivatives.

The thrust of a rocket can be modeled from 4) with the inclusion of variable mass. By substituting 1) on the LHS of 4) and dividing by the mass m on both sides, we get the following:

$$a = \frac{dv}{dt} = \left(\frac{v}{m}\right) * \left(\frac{dm}{dt}\right) - g \quad (5)$$

$$dv = \left(\frac{u}{M}\right) * dm - g * dt \quad (6)$$

Integrating from the initiation of thrust to an arbitrary time t:

$\int_{v0}^{v} dv = u * \int_{m0}^{m} dm - g * \int_{0}^{t} dt$

This gives the velocity at time t:

$$v = v0 + u \ln m/m0 - gt \quad (7)$$

This is our final equation to find the upward velocity of a rocket when launched from the earth's surface.

Rocket thrust is a result of the high-speed ejection of material and does not require any medium to push against. Conservation of momentum dictates that if the material is ejected backward, the forward momentum of the rocket must increase in order to not let the system change its net momentum. Developing the expression for rocket thrust involves the application of conservation of momentum in an accelerating reference frame. It can be deduced using approximation using finite differences and algebra. The terms like dm and dv can be neglected while using appropriate limits to get to the correct instantaneous expression.

IV. METHODOLOGY

To solve this research problem, I set up initial values to better understand the problem. As derived in the above section, the upward velocity of the rocket is given by the following formula:

$$v = u \ln\left(\frac{m}{m - qt}\right) - gt$$

where v = upward velocity, u = velocity at which fuel is expelled relative to the rocket, m = initial mass of the rocket at time t = 0, q = fuel consumption rate, and g = downward acceleration of gravity. To figure out the best possible method, I set a target in this problem. The question was to find the time at which the upward velocity reaches a value of 1000 m/s. We all know that the downward acceleration of gravity is 9.81 m/s^2. The relative velocity at which fuel is expelled is set at 2200m/s. The mass of the rocket is set at 160,000 kg. The fuel consumption rate is 2680 kg/s. These were the values set in this equation before applying the numerical methods. I will elaborate on the numerical methods employed in the following section.

The numerical methods provide a way to find the roots of the equation by employing different strategies. There are two broad classification of methods to find the roots of equations. Bracketing methods and Open Methods are these two classifications. This research paper talks about different bracketing and open methods that have been employed to find the value of time t at which the upward velocity reaches a pre-defined value. Bracketing methods is a strategy that deals with methods that start with guesses that contain the root and then systematically reduce the width of the bracket. These methods exploit the fact that the function changes sign in the vicinity of a root. Open methods involve systematic trial and error iterations to reach the solution of the equation. They do not require the use of guesses that bracket the root. We will understand how the open methods are computationally efficient to find roots but are not always successful in finding them.

A. Bracketing Methods

The bracketing methods works on the principle that the function changes sign in the vicinity of the root. These methods are called bracketing methods since they require two initial guesses of the root. These guesses must bracket the root. These methods employ different strategies to systematically reduce the width of the bracket and deduce the correct answer.

Before understanding the bracketing methods, let's discuss graphical methods for depicting functions and its roots. The most rudimentary method to obtain an estimate of the root of the equation f(x)=0 is to plot the function and observe its intersection on the x axis. This intersecting point on the x axis provides a rough approximation of the root. These methods also serve as a good precursor for the starting guesses in the numerical methods discussed in the following sections. When the numerical methods are used in collaboration with the numerical methods, they are extremely useful in solving many roots of equations in the field of Engineering and Mathematics.

1. *False Position*

False position is a bracketing method that is based on a graphical insight. It takes into consideration the closeness of $f(x_l)$ and $f(x_u)$ to zero. This gives an improved estimate of the root than blindly assuming equal halves in the interval between x_l and x_u. This method exploits the graphical insight by joining $f(x_l)$ and $f(x_u)$ by a straight line. The intersection of this line with the x axis represents and improved estimate of the root. The replacement of the curve between these points instead of the curve gives a "false position" of the root thereby giving it the name false position.

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$

This equation provides an equation to find an improved estimate of the root in every iteration thereby improving the chances of finding the solution. This is the false-position formula. The value of x_r is replaced by either x_l or x_u based on which value has the same sign as $f(x_r)$. This process is repeated until the root is estimated adequately.

Since the root estimate is continuously changing in every iteration, the root is known with great accuracy than the prescribed tolerance. However, it is possible for the false position to function poorly in case where the functions have significant curvature. This happens because the bracketing points that are used tend to stay fixed. This leads to poor convergence in determining the root value.

2. Bisection Method

The bisection method is also called as binary chopping or Bolzano's method. It is a type of incremental search method in which the interval between the guesses is divided into half. If the function changes sign over an interval, the function value at the midpoint is calculated. The location of the root is determined as lying at the midpoint of the subinterval within which the sign change occurs. The process is repeated to obtain refined estimates.

The termination criteria can be manipulated in such a way that if there is not much improvement in the value of the root in the present and previous iteration, the iteration could be stopped. In my development, this value was kept at 0.5%. If the approximate percent relative error drops below 0.5%, the iteration was terminated. Although bisection is slower than other methods, the neatness of its error analysis is a positive aspect that makes it usable for certain engineering applications.

B. Open Methods

In bracketing methods, the implied assumption is that root is located within the interval prescribed in the lower and upper bound. The bracketing methods are convergent as they move closer to the truth as the computation progresses. Open methods do not need the initial guess to bracket the root. They move from the true root and diverge in some cases. However, when they converge they do so much more quickly than the bracketing methods.

1. Secant Method

The secant method employs the use of derivatives to find the improved root estimate during its computation. However, its not always easy to find the derivatives of certain functions. In such cases, the derivative can be approximated using a backward finite divided difference.

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

This approach requires two initial estimates of x. However since f(x) is not required to change signs between the estimates, it is not classified as a bracketing method.

The secant method is very similar to the false position method. Both use two initial estimates to compute an approximation of the slope of the function to project to the x-axis for a new estimate of the root. The false-position method estimated the latest root to a value that yielded a function value with the same sign as the function value of the root in that iteration. However, the secant method replaces x_{i+1} with x_i and x_i replaces x_{i-1}. This could lead to divergence in this method. However, when the method converges, it does so much quickly than the false-position method.

2. Modified Secant

The modified secant is very similar to the Secant method. However instead of using two arbitrary values to estimate the derivative, it deploys the use of a fractional perturbation to estimate the derivative of f(x).

$$x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i + \delta x_i) - f(x_i)}$$

The choice of a proper value of the δ is not automatic. If the value is too small, the method can be swamped by round-off error caused by subtractive cancelation in the denominator of the above equation. If the value is too big, the technique can become inefficient and even divergent. However, if chosen correctly, it provides an excellent alternative for cases when evaluating the derivative is difficult and developing two initial guesses is not possible.

3. Newton-Raphson

The Newton-Raphson method is one of the most widely used root-locating formula. If the initial guess at the root is x_i, a tangent can be extended from the point [xi, f(xi)]. The point where this tangent crosses the x axis represents an improved estimate of the root.

Although the Newton-Raphson method is efficient, it can perform poorly in certain situations. Aside from slow convergence due to the nature of the function, there could be other difficulties. The Newton-Raphson technique has the tendency to oscillate between the local maximum or minimum. There is no convergence criterion for this method. Its convergence depends on the nature of the function and the accuracy of the initial guess. The only remedy is to have a solution that is close to the root.

There are some strict specifications that would be required during implementation of the Newton-Raphson method. This includes the use of a plotting routine in the program. At the end of the computation, the final root estimate should be substituted into the original function to compute whether the result is close to zero. This guards against cases where slow or oscillating convergence leads to a small value of the approximate relative error. The program should always include an upper limit on the number of iterations.

V. OBSERVATIONS

A. Graphical Methods

In our problem I employed the graphical approach to estimate the root by plotting the 7) using initial guesses of 10 and 50. This provided me with an initial view of the value of time t at which the velocity of the rocket is 1000m/s.

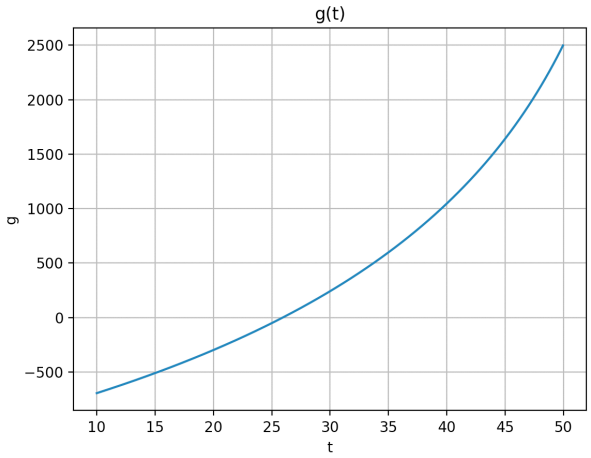

Fig1: Plotting the function with initial guesses of t=10 and t=50

Visual inspection of the graph reveals the root to be between 25 and 30. To narrow it down further, I plotted another graph with the guesses as 25 and 30. This led me to the following graph.

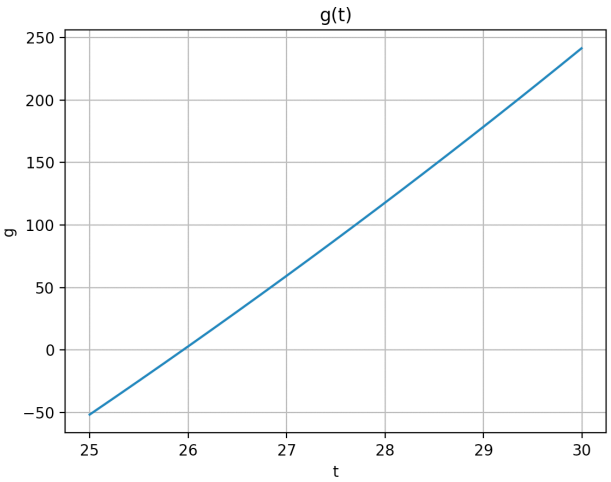

Fig2: Plotting the function with initial guesses of t=25 & t=30

This graph clearly reveals the root between 25 and 26.

B. False Position

I will describe the results obtained using False position below. The false position method obtained the root of the equation in 13 iterations. The output is as follows:

```
false position:
iter      xl        xu        xr        ea
1         10.0      50.0      18.68     46.4664%
2         18.68     50.0      22.59     17.3106%
3         22.59     50.0      24.38     7.3593%
4         24.38     50.0      25.22     3.3008%
5         25.22     50.0      25.61     1.5161%
6         25.61     50.0      25.79     0.7040%
7         25.79     50.0      25.87     0.3286%
8         25.87     50.0      25.91     0.1537%
9         25.91     50.0      25.93     0.0720%
10        25.93     50.0      25.94     0.0337%
11        25.94     50.0      25.94     0.0158%
12        25.94     50.0      25.95     0.0074%
13        25.95     50.0      25.95     0.0035%
R = 25.946283162055707
```

Fig3: False-position iteration for computing root of the equation

The false position method yields a root that is equal to 25.94628 with e_s = 0.005. Because of the efficient scheme of root location in the false position method, the error approximation reduces quickly to below 1%. It takes only 6 iterations for the error approximation to go below 1%.

C. Bisection Method

Next, I solved the function using the Bisection method. The bisection method obtained the root of the equation in 15 iterations which is longer than the false position method for the same error tolerance.

```
using bisection method:
iter      xl        xu        xr        ea
1         10.0      50.0      30.0      66.6667%
2         10.0      30.0      20.0      50.0000%
3         20.0      30.0      25.0      20.0000%
4         25.0      30.0      27.5      9.0909%
5         25.0      27.5      26.25     4.7619%
6         25.0      26.25     25.62     2.4390%
7         25.62     26.25     25.94     1.2048%
8         25.94     26.25     26.09     0.5988%
9         25.94     26.09     26.02     0.3003%
10        25.94     26.02     25.98     0.1504%
11        25.94     25.98     25.96     0.0752%
12        25.94     25.96     25.95     0.0376%
13        25.94     25.95     25.94     0.0188%
14        25.94     25.95     25.94     0.0094%
15        25.94     25.95     25.95     0.0047%
R = 25.946044921875
```

Fig4: Bisection method solution to compute the root of the equation

The root of the function by Bisection method turns out to be R = 25.94604. From the results, it can be concluded that the bisection method performs slightly worse than the false position method in this case. The bisection method obtains this root after 15 iterations with a percent relative error of 4.7 x 10^{-5}. The percent error approximation falls below 1% in the 8th iteration. This is slower than what we observed in the false position for the same error tolerance. In the bisection method the interval between the lower and upper estimate keeps decreasing during the course of a computation. Among the two bracketing methods that have been tested for this problem, I conclude that the false position method is slightly superior to the bisection method.

D. Secant Method

Secant method was the first open method that was tested. I could obtain the root of the equation in 6 iterations.

```
using secant method:
iter=0, xr=18.67984424374218, ea=167.668185%
iter=1, xr=22.590375426456102, ea=17.310607%
iter=2, xr=26.363457373077928, ea=14.311787%
iter=3, xr=25.922859309299405, ea=1.699651%
iter=4, xr=25.946902917422275, ea=0.092665%
iter=5, xr=25.947079001431906, ea=0.000679%
R = 25.947079001431906
```

Fig5: Secant method to compute the root of the equation

This was a markedly improved performance as compared to the bracketing methods of false position and bisection. The error approximation goes below 1% in the 5th iteration. This is quicker than both the bracketing methods that we tried earlier. The secant method calculates the derivative by using a backward finite divided difference. I kept the lower and upper estimate for this function to be at the 10 and 50 to keep it consistent with the other methods and be able to compare between these methods. But this is not necessary for open methods.

The secant method converges twice as quickly as compared to the false-position method. The inferiority of the false-position method is due to one end staying fixed to maintain the bracketing of the root. This does avoid divergence, but it does not solve the problem of converging quickly to the root.

E. Modified Secant

The modified secant method proves to be the most efficient so far in finding the upward velocity of the rocket. It returns the root for the equation in 5 iterations.

```
modified secant method:
iter=0, xr=30.13955218004831, ea=66.821007%
iter=1, xr=26.283023856746606, ea=14.673077%
iter=2, xr=25.950594251420487, ea=1.281010%
iter=3, xr=25.947095059866044, ea=0.013486%
iter=4, xr=25.94707900014371, ea=0.000062%
R = 25.94707900014371
```

Fig6: Modified Secant method to compute root

The percentage error approximation falls below 1% in the 4th iteration. The modified secant method converges to the real root after 5 iterations which is almost thrice as quick as the false-position method.

The modified secant method employs the use of a perturbation factor instead of using two initial estimates as done in the

secant method. This proves to be more effective since the value converges to the real root within 5 iterations. It is a significant improvement over the secant method which takes longer to converge and also requires two initial estimates to calculate the real root.

F. Newton-Raphson method

This method turns out to be as efficient as the Modified secant method. This method provides the root of the equation in 5 iterations.

```
using newton-rhapson method:
iter=0, xr=30.165616336467735, ea=66.849675%
iter=1, xr=26.26417673986349, ea=14.854605%
iter=2, xr=25.948834875126025, ea=1.215245%
iter=3, xr=25.947078980868415, ea=0.006767%
iter=4, xr=25.947078927102172, ea=0.000000%
R = 25.947078927102172

Process finished with exit code 0
```

Fig6: Newton-Raphson computation

The root of the function is equal to R = 25.94707. The percentage error approximation goes below 1% in the 4th iteration. This performance is similar to the modified secant method and this is slightly better than the observation in the secant method.

This method works on the fact that the initial guess is picked correctly. If the initial guess is set to the higher value i.e let's say 100, the routine diverges. It prints out a math domain error which means its reached a computation where it divided by zero or performed an illegal mathematical operation.

This occurs because the function's slope at the initial guess causes the iteration to jump to a negative value. So we can conclude the Newton-Raphson method requires good initial guesses to function efficiently.

VI. Future Work and Recommendations

This research in this paper is derived from the findings for the numerical methods that have been programmed using Python. Fixed-point iteration is another open numerical method that can be used to research and identify the best numerical method. Another important technique that can be used is Brent's method. This method combines the reliability of bracketing with the speed of open methods. It would provide the best of both worlds and help improve the performance further and converge to a solution quicker. Since there are pros and cons for every method, the researcher needs to understand the constraints and find the best possible method for his implementations.

VII. Conclusion

I described the implementation of the numerical methods to find the upward velocity of the rocket. In addition, there was detailed analysis on the results from these numerical methods. The performance of these numerical methods were compared to calculate the upward velocity of the rocket.

The Newton-Raphson and the Modified secant method prove to be the best of all the methods used to calculate the friction factor. They converge to the correct value in 5 iterations which is superior to all the other methods. Both methods employ the use of a single guess but use a perturbation factor in the case of the modified secant method. Whereas the Newton-Raphson uses the derivative of the function to converge to the real roots.

VIII. Acknowledgment

The author thanks Professor Luis Paris and Harrisburg University of Science and Technology for giving this opportunity to research the importance of numerical methods in the fields of Aerospace Engineering and for providing the chance to present this paper to present these findings.

References

[1] Steven C. Chapra & Raymond P. Canale, "Numerical Methods for Engineers", Seventh Edition, New York, NY

[2] J. Peraire, S. Widnall, Fall 2008, Variable Mass Systems, The Rocket Equation

[3] Mark Abate, Eswar Anandapadmanaban, Larry Bao, Siddharth Challani, Jason Gaughan, Andrew Jiang, Abhishek Lingineni, Anish Vora, Christopher Yang, Dustin Zhao, "Correlation Between Simulated, Calculated, And Measured Model Rocket Flight"

[4] Benson T. Brief History of Rockets [Internet]. National Aeronautics and Space Administration ; [2014 Jun 12, cited 2014 Jul 28] . Available from: http://www.grc.nasa.gov/WWW/k-12/TRC/Rockets/history_of_rockets.html

[5] Hartmann A. History of Rocketry [Internet]. Cape Canaveral(FL):Spaceline Inc.; cited 2014 Jul 28]. Available from: http://www.spaceline.org/spaceline.html

[6] Stamp J. 2013. The History of Rocket Science: When was the First-Ever Rocket Built?. Smithsonian Magazine [Internet]. [cited 2014 Jul 28]. Available from: http://www.smithsonianmag.com/innovation/the-history-of-rocket-science-4078981/?noist

[7] Barrowman J. 1970. Stability of a Model Rocket in Flight [Internet]. Phoenix(AZ):Centuri Engineering Company; [cited 2014 Jul 28] Available from: http://www.rockets4schools.org/images/Rocket.Stability.Flight.pdf