A Computational Approach to Obtaining Exact Solutions of an Nth-Order Nonlinear Differential Equation in Forms of $f^{(n)}(x) = H(f^{(n-1)}(x), f^{(n-2)}(x), \dots, f^{1}(x), f(x), g(x))$

Author : Yuchung (Troy) Wu
Date : January 2021

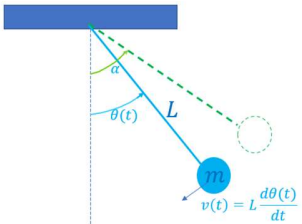
Motivation and Research

Motivation

Skydiving is beautiful, heart-racing, and exhilarating on film. I particularly admire freefall cameramen's skill to control the diving speed under gravitational acceleration and still shoot good movies. It is intriguing to find IB examiners hope students to learn more about the speed change of skydiving by putting it as an examination question. (ref.1)

To understand it in depth, I found a good article in College Physics (ref. 2) well demystifying how skydivers control their terminal speed by changing their cross sections facing ground. However, the nonlinear differential equation, $\frac{d^2S(t)}{dt^2} = -g + \frac{1}{2m}C\rho A\left(\frac{dS(t)}{dt}\right)^2$ (where S: height g: gravitational acceleration C: drag coefficient A: cross section facing ground ρ air density), is really intimating. I looked up college calculus but failed to find the resolution.

As I tried to build a pendulum of true isochronism for my Physics IA, I derived, based on energy conservation, the differential equation describing the time function of swing angle, $\theta(t)$, for a pendulum. It is $\frac{d\theta(t)}{dt} = \pm\sqrt{\frac{2g}{L}[cos\theta(t) - cos\alpha)}$.



By energy conservation:

$K + U = constant$:

$\Delta K = -\Delta U$

$\frac{1}{2}mv^2 = -mg\Delta h$

$\frac{1}{2}m(L\frac{d\theta(t)}{dt})^2 = mg(\ L\cos\theta(t) - L\cos\alpha)$

$(\frac{d\theta(t)}{dt})^2 = \frac{2g}{L}\ [\cos\theta(t) - \cos\alpha]$

$\frac{d\theta(t)}{dt} = \pm\sqrt{\frac{2g}{L}\ [\cos\theta(t) - \cos\alpha]}$

It looks like a neat 1st order differential equation, much simpler than the 2nd order one of skydiving dynamics. However, it seems none of resolutions I have learned from Math HL is applicable to get this resolved. A paper from University of Connecticut (ref.3) suggests it be resolved by "Elliptic Integral of the First Kind".

To get rid of the trauma that even ordinary engineering questions can be only handled by mathematicians, I need to work a way to depict what the solutions of those differential equations look like.

Research

(i) Solution for 1st differential equation

I have learned from IB math high level several skills and procedures to find the solutions for some 1-order differential equations that can be written in the forms below. (ref. 3)

Type 1. $\frac{dy}{dx} = g(x)$ (e.g. $\frac{dy}{dx} = e^{2x}$)

Type 2. $\frac{dy}{dx} = \frac{g(x)}{h(y)}$ (called separable) (e.g. $\frac{dy}{dx} = \frac{x}{y^2}$)

Type 3. $\frac{dy}{dx} = g(\frac{y}{x})$ (called homogeneous) (e.g. $\frac{dy}{dx} = \frac{x}{y} - 1$)
Type 4. $\frac{dy}{dx} + P(x)y = Q(x)$ (e.g. $\frac{dy}{dx} + 3x^2y = 6x^2$)

It is easy to get lost in those type-orientated approaches even though the solutions are only for a portion of 1st order differential equations. Besides, falling in one of the 4 types does not guarantee a solution could be got in a closed form. The pendulum equation $\frac{d\theta(t)}{dt} = \pm\sqrt{\frac{2g}{L}[cos\theta(t) - cos\alpha)}$ falls in Type 1 but to find the function, $\int \sqrt{\frac{2g}{L}[cos\theta(t) - cos\alpha)dt}$,is extremely difficult.

(ii) Solution for 2nd differential equation

I turn to Thomas' Calculus (ref. 4) for resolution of 2nd order differential equations. It's instructed how to solve 2nd order differential equation that can be written in the following forms:

1. Linear nonhomogeneous differential equation with constant coefficient of

$$\frac{d^2y(x)}{dx^2}, \frac{dy(x)}{dx}, y(x)$$

$$a\frac{d^2y(x)}{dx^2} + b\frac{dy(x)}{dx} + cy(x) = G(x) \; where \; a, b, c \; are \; constants$$

2. If the coefficients of $\frac{d^2y(x)}{dx^2}, \frac{dy(x)}{dx}, y(x)$ are not constant, then, to be resolvable, it's necessary that they can put in the following format with G(x) = 0. It's called Euler equation.

$$ax^2\frac{d^2y(x)}{dx^2} + bx\frac{dy(x)}{dx} + cy(x) = 0, \qquad where \; a, b, c \; are \; constants$$

Obviously, the mechanical equation of skydiving is not among those resolvable ones. It seems hard for us to figure out the complete behavior even for an ordinary mechanism of well-defined differential equation.

(iii) Power Series for linear differential equation

Thomas demonstrated how to get solution in Power Series. Here is an example
Solve the equation $y^{(2)}(x) + y(x) = 0$ by the power-series method.
Solution We assume the series solution takes the form of

$$y(x) = \sum_{n=0}^{\infty} C_n x^n$$

and calculate the derivatives
$y'(x) = \sum_{n=1}^{\infty} nC_n x^{n-1}$ and $y(x) = \sum_{n=2}^{\infty} n(n-1)C_n x^{n-2}$

Substitution of these forms into the second-order equation gives us
$\sum_{n=2}^{\infty} n(n-1)C_n x^{n-2} + \sum_{n=0}^{\infty} C_n x^n = 0$

We learn $C_n = -\frac{1}{n(n-1)} C_{n-2}$, $C_2 = -\frac{1}{2\cdot 1} C_0$, $C_3 = -\frac{1}{3\cdot 2} C_1$

Finally, it yields

$y(x) = C_0 \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!} x^{2k} + C_1 \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!} x^{2k+1}$

Thomas reminds that they happen to be Maclaurin Series for cosine and sine function.

$y(x) = C_0 cosx + C_1 sinx$

It seemed "OK" before I went to the example next.

Assume the solution of $y''(x) + xy'(x) + y(x) = 0$ to be

$$y(x) = \sum_{n=0}^{\infty} C_n x^n$$

$$\sum_{n=2}^{\infty} n(n-1) C_n x^{n-2} + x \sum_{n=1}^{\infty} n C_n x^{n-1} + \sum_{n=0}^{\infty} x^n = 0$$

We learn $C_{n+2} = -\frac{1}{n+2} C_n$, $C_2 = -\frac{1}{2} C_0$, $C_3 = -\frac{1}{3} C_1$

Finally, it yields

$y(x) = C_0 \sum_{k=0}^{\infty} \frac{(-1)^k}{(2)(4)\cdots(2k)} x^{2k} + C_1 \sum_{k=0}^{\infty} \frac{(-1)^k}{(3)(5)\cdots(2k+1)} x^{2k+1}$

This time Thomas just left as it is because there is no close form function could be mapped.

Frankly I do not know how I can make use of infinite series even told it is the solution.

Trying to follow the procedure, I learned the powers-series method does not fit nonlineart differential equation.

Let us find Power Series Solution for skydiving equation $y''(x) = -g + a[y'(x)]^2$

$$y(x) = \sum_{n=0}^{\infty} C_n x^n$$

$$\sum_{n=2}^{\infty} n(n-1) C_n x^{n-2} = -g + (\sum_{n=0}^{\infty} C_n x^n)^2$$

It seems hopeless for me to resolve C_n due to the multiplication of two infinite series, $(\sum_{n=0}^{\infty} C_n x^n)^2$. I decide to revisit this later.

After trying so many methods in vain, I consider scrutinizing the fundamental definition might be inspiring.

$$\frac{dy}{dx} = \lim_{h \to 0} \frac{y(x+h) - y(x)}{(x+h) - x}$$

A derive is to tell a function how rapidly it changes around some point. And a differential equation is to tell us how a point is coupled, through derivative of a function y(x), to the other points in its neighborhood. All we are reminded of is not to neglect the neighbors. We are probably able to follow the bread scum left by a differential equation to find the function values of the other points in the neighborhood.

Let us check out the 1st order equation, type 4,

$$\frac{dy}{dx} + P(x)y = Q(x)$$

$$\rightarrow \frac{y(x_0 + h) - y(x_0)}{h} + P(x_0)y(x_0) = Q(x_0), \quad where\ h \rightarrow 0$$

$$\rightarrow y(x_0 + h) = h \cdot [\frac{1}{h}y(x_0) + Q(x_0) - P(x_0)y(x_0)] \cdot, \ \ where\ h \rightarrow 0$$

It shows that we can easily get $y(x_0 + h)$ once given $y(x_0)$,since $Q(x_0)$ and $P(x_0)$ are known function values. (Please be reminded that Q(x) and P(x) are given function. We can get any Q(x_i) and P(x_i) by letting $x = x_i$in the functions)). Likewise, we can easily get $y(x_0 + 2h)$ once knowing $y(x_0 + h)$ since $Q(x_0 + h)$ and $P(x_0 + h)$ are known values. Iteratively, we can find the function values for all x in domain, starting from one known point, x_0.

How about 2nd differential equation?

$$a\frac{d^2y(x)}{dx^2} + b\frac{dy(x)}{dx} + cy(x) = G(x)$$

$$\rightarrow a\frac{y'(x_0 + h) - y'(x_0)}{h} + b\frac{y(x_0 + h) - y(x_0)}{h} + cy(x_0) = G(x_0)$$

$$\rightarrow a\frac{\frac{y(x_0 + 2h) - y(x_0 + h)}{h} - b\frac{y(x_0 + h) - y(x_0)}{h}}{h} + b\frac{y(x_0 + h) - y(x_0)}{h} + cy(x_0) = G(x_0)$$

$$\rightarrow \frac{a}{h^2} \cdot y(x_0 + 2h) = G(x_0) + \left(\frac{a}{h^2} - \frac{b}{h}\right) \cdot y(x_0 + h) + \left(-\frac{a}{h^2} + \frac{b}{h} - c\right) \cdot y(x_0)$$

$$y(x_0 + h) = \frac{h^2}{a}\left[G(x_0) + \left(\frac{a}{h^2} - \frac{b}{h}\right) \cdot y(x_0 + h) + \left(-\frac{a}{h^2} + \frac{b}{h} - c\right) \cdot y(x_0)\right]$$

It shows that we can easily get $y(x_0 + nh)$ if given $y(x_0)$ and $y(x_0 + h)$, since $G(x_0)$ is a known value. (Please be reminded that G(x) is given function.) Likewise, we can easily get $y(x_0 + 3h)$ once knowing $y(x_0 + h)$ and $y(x_0 + 2h)$ since $G(x_0 + h)$ is a known value. Iteratively, we can find the function values for all x in domain, starting from one known value close points, $x_0\ and\ x_0 + h$.

The process above to find $y(x_0 + nh)$ meeting an n-order differential equation based on known $y(x_0), y(x_0 + h), \ldots y(x_0 + h), \ldots, y(x_0 + (n-1)h)$ looks promising. However, the coefficient derivation is tedious and will become more grueling in handling high-order differential equations of more comprehensive forms where the coefficients are not just constants but functions. It is difficult to directly apply the method for the previous two equations.

In this internal assessment, I will explore and examine this idea, develop it into a simple algorithm and make it applicable for various high-order differential equations, to hopefully resolve the skydiving and pendulum equations in the long run.

Proof Of Concepts

Given a general differential equation
$y^{(n)}(x) = F(y(x), y^{(1)}(x), y^{(2)}(x), \cdots, y^{(n-1)}(x))$
together with the initial values
$y(x_0), y(x_0 + h), y(x_0 + 2h), \cdots, y(x_0 + (n-1)h)$, where $h \to 0$

could we acquire $y(x_0 + nh)$ accordingly?

As we can see from research, direct substitution is a promising way to prove that we can get $y(x_0 + h)$ from $y(x_0)$ for "some" 1st differential equations and $y(x_0 + 2h)$ from $y(x_0)$ and y$(x_0 + h)$ for "some" 2nd differential ones. However, we have observed the coefficient acquisition is getting harder as differential order increases. Besides, the types of those differential equation are likely extremely limited once the differential coefficients are not constants. We are thus prompted to figure out a way other than direct substitution to prove this primitive idea.

I will separate the proof into two section.

Section 1: I will prove y$(x_0), y^{(1)}(x_0), y^{(2)}(x_0), \cdots, y^{(n-1)}(x_0)$ can be acquired from $y(x_0), y(x_0 + h), y(x_0 + 2h), \cdots, y(x_0 + (n-1)h)$,

Section2: I will prove $y(x_0 + nh), y^{(1)}(x_0 + nh), y^{(2)}(x_0 + nh), \cdots, y^{(n-1)}(x_0 + nh)$ can be acquired from $y(x_0), y^{(1)}(x_0), y^{(2)}(x_0), \cdots, y^{(n-1)}(x_0)$.

Once finishing Section1 and Section2, it will complete the proof that $y(x_0 + nh)$ can be acquired from known $y(x_0), y(x_0 + h), y(x_0 + 2h), \cdots, y(x_0 + (n-1)h)$

Section 1:
Prove $y(x_0), y^{(1)}(x_0), y^{(2)}(x_0), \cdots, y^{(n-1)}(x_0)$ could be acquired from $y(x_0), y(x_0 + h), y(x_0 + 2h), \cdots, y(x_0 + (n-1)h)$,

Proof:
By definition $y^{(n)}(x_0) = \frac{y^{(n-1)}(x_0+h) - y^{(n-1)}(x_0)}{h}$
We can learn
$y^{(1)}(x_0)$ by $\frac{y^{(0)}(x_0+h) - y^{(0)}(x_0)}{h}$,
$y^{(1)}(x_0 + h)$ by $\frac{y^{(0)}(x_0 + 2h) - y^{(0)}(x_0 + h)}{h}$
……..,
$y^{(1)}(x_0 + (n-2)h)$ by $\frac{y^{(0)}(x_0+(n-1)h) - y^{(0)}(x_0+(n-2)h)}{h}$,

With $y^{(1)}(x_0), y^{(1)}(x_0 + h), \cdots, y^{(1)}(x_0 + (n-2)h)$ learned, we can proceed to get
$y^{(2)}(x_0)$ by $\frac{y(x_0+h) - y^{(1)}(x_0)}{h}$,

$y^{(2)}(x_0+h)$ by $\dfrac{y^{(1)}(x_0+2h)-y^{(1)}(x_0+h)}{h}$

……..,

$y^{(2)}(x_0+(n-3)h)$ by $\dfrac{y^{(1)}(x_0+(n-2)h)-y^{(1)}(x_0+(n-3)h)}{h}$,

…….

Repeat the process

With $y^{(n-3)}(x_0), y^{(n-3)}(x_0+h), y^{(n-3)}(x_0+2h)$ learned, we can proceed to get

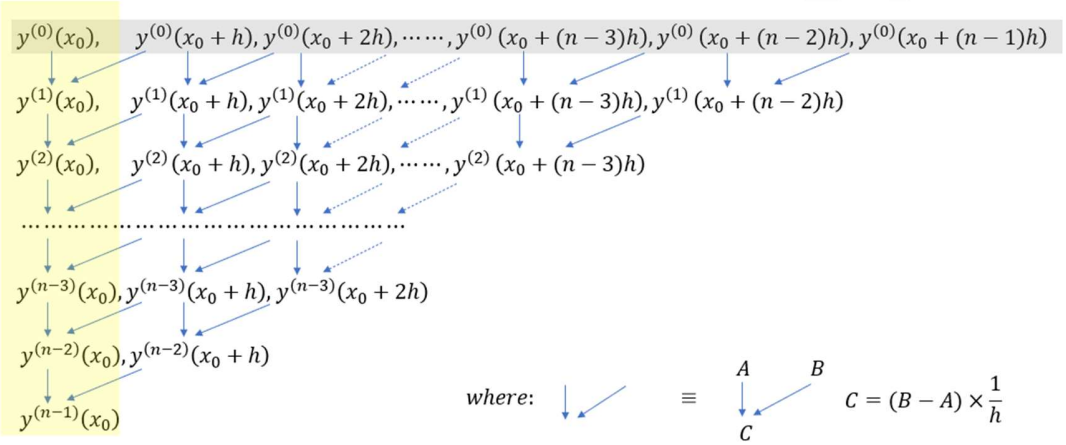
$y^{(n-2)}(x_0)$ by $\dfrac{y^{(n-3)}(x_0+h)-y^{(n-3)}(x_0)}{h}$,

$y^{(n-2)}(x_0+h)$ by $\dfrac{y^{(n-3)}(x_0+2h)-y^{(n-3)}(x_0+h)}{h}$

With $y^{(n-2)}(x_0), y^{(n-2)}(x_0+h)$ learned, we can proceed to get

$y^{(n-1)}(x_0)$ by $\dfrac{y^{(n-2)}(x_0+h)-y^{(n-2)}(x_0)}{h}$,

A figure probably makes it easier to understand.



It is proved that $y(x_0), y^{(1)}(x_0), y^{(2)}(x_0), \cdots, y^{(n-1)}(x_0)$, the first column shaded in yellow, are acquired from known $y(x_0), y(x_0+h), y(x_0+2h), \cdots, y(x_0+(n-1)h),$ which is the first row shaded in grey.

Section2:

Prove that $y(x_0+nh), y^{(1)}(x_0+nh), y^{(2)}(x_0+nh), \cdots, y^{(n-1)}(x_0+nh)$ can be acquired from $y(x_0), y^{(1)}(x_0), y^{(2)}(x_0), \cdots, y^{(n-1)}(x_0)$.

Proof:

It's given that $y^{(n)}(x)=F(y(x), y^{(1)}(x), y^{(2)}(x), \cdots, y^{(n-1)}(x))$ for all x in domain

Since we have learned $y(x_0), y^{(1)}(x_0), y^{(2)}(x_0), \cdots, y^{(n-1)}(x_0)$, we could directly get $y^{(n)}(x_0)$ by

$$y^{(n)}(x_0)=F(y(x_0), y^{(1)}(x_0), y^{(2)}(x_0), \cdots, y^{(n-1)}(x_0))$$

By definition $y^{(k)}(x_0)=\dfrac{y^{(k-1)}(x_0+h)-y^{(k-1)}(x_0)}{h}$

$$\rightarrow y^{(k-1)}(x_0+h)=y^{(k-1)}(x_0)+y^{(k)}(x_0)\times h$$

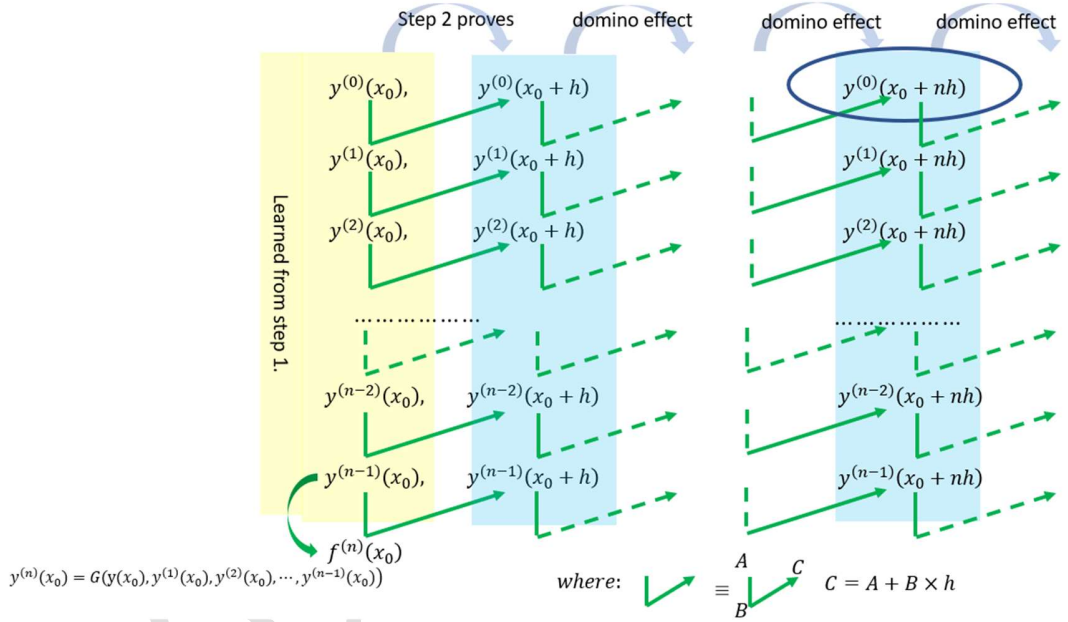
So, we can get

$k = 1: y^{(0)}(x_0 + h) = y^{(0)}(x_0) + y^{(1)}(x_0) \times h$
$k = 2: y^{(1)}(x_0 + h) = y^{(1)}(x_0) + y^{(2)}(x_0) \times h$
$k = 3: y^{(2)}(x_0 + h) = y^{(2)}(x_0) + y^{(3)}(x_0) \times h$

$k = n - 1: y^{(n-2)}(x_0 + h) = y^{(n-2)}(x_0) + y^{(n-1)}(x_0) \times h$
$k = n \quad : y^{(n-1)}(x_0 + h) = y^{(n-1)}(x_0) + y^{(n)}(x_0) \times h$

➔It shows $y(x_0 + h), y^{(1)}(x_0 + h), y^{(2)}(x_0 + h), \cdots, y^{(n-1)}(x_0 + h)$, can be acquired learned from $y(x_0), y^{(1)}(x_0), y^{(2)}(x_0), \cdots, y^{(n-1)}(x_0)$. Once we have learned $y(x_0 + h), y^{(1)}(x_0 + h), y^{(2)}(x_0 + h), \cdots, y^{(n-1)}(x_0 + h)$, we can proceed to get $y(x_0 + 2h), y^{(1)}(x_0 + 2h), y^{(2)}(x_0 + 2h), \cdots, y^{(n-1)}(x_0 + 2h)$. With the domino effect which mathematical induction is based on, it is inferred that we can iterate this process to get $y(x_0 + nh), y^{(1)}(x_0 + nh), y^{(2)}(x_0 + nh), \cdots, y^{(n-1)}(x_0 + nh)$ in the long run and even go beyond.

Again, a figure probably makes it easier to understand.



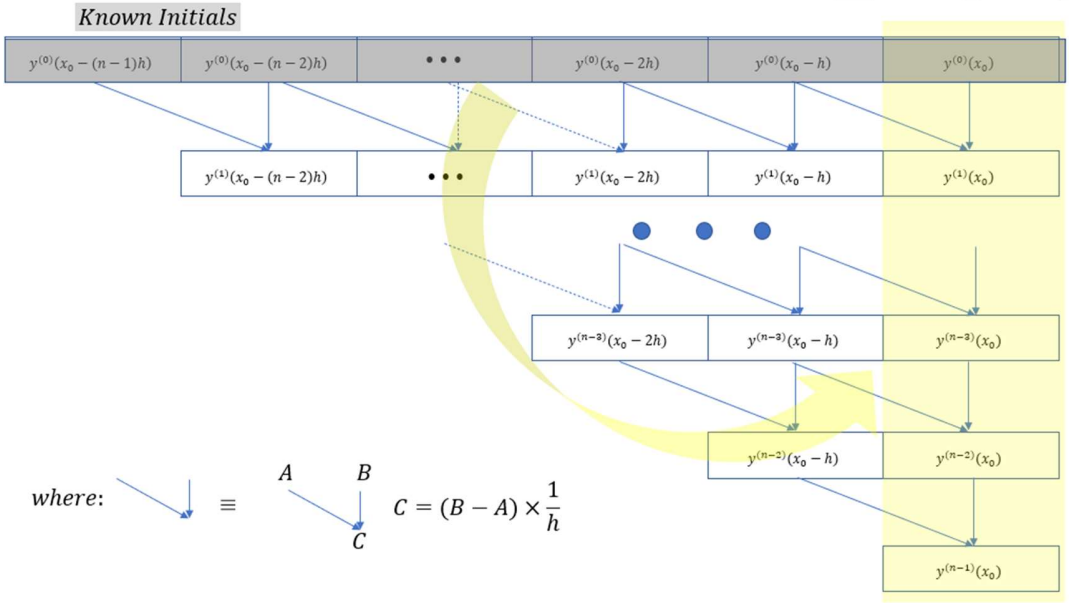
With section 1 and 2,
We have proved that we can always get the value next, $y(x_0 + nh)$, with last n values $y(x_0), y(x_0 + h), y(x_0 + 2h), \cdots, y(x_0 + (n-1)h), where\ h \to 0$. It also means we can get the function values for all x because of domino effect and infinitely small h.

Algorithm Development

While proving the concept, we have developed a systematic way to get $y(x_0 + nh)$, with last n values $y(x_0), y(x_0 + h), y(x_0 + 2h), \cdots, y(x_0 + (n-1)h)$
Or equivalently, to get $y(x_0 + h)$, with last n values $y(x_0), y(x_0 - h), y(x_0 - 2h)), \cdots, y(x_0 - (n-1)h)$

This systematics method goes as:

Step#1 Convert $y(x_0), y(x_0 - h), y(x_0 - 2h), \cdots, y(x_0(n-1))h)$
to $y(x_0), y^{(1)}(x_0), y^{(2)}(x_0), \cdots, y^{(n-1)}(x_0)$ based on the following operation shown below



This is horizontally rotated to the figure we plot in proof section 1. Why?
Recalling

$$\left.\frac{dy}{dx}\right|_{x=x_0} = \lim_{h\to 0}\frac{y(x_0+h)-y(x_0)}{(x_0+h)-x_0} = \lim_{h\to 0}\frac{y(x_0)-y(x_0-h)}{x_0-(x_0-h)}$$

Applying this for 1st and 2nd order derivatives, we will have

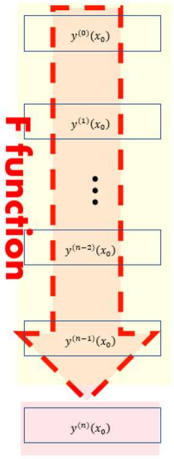
$$y^{(1)}(x_0-(n-2)h) = \frac{y^{(0)}(x_0-(n-2)h)-y^{(0)}(x_0-(n-1)h)}{h}, \cdots,$$
$$y^{(1)}(x_0-h) = \frac{y^{(0)}(x_0-h)-y^{(0)}(x_0-2h)}{h}, y^{(1)}(x_0) = \frac{y^{(0)}(x_0)-y^{(0)}(x_0-h)}{h}$$

$$y^{(2)}(x_0-(n-3)h) = \frac{y^{(1)}(x_0-(n-3)h)-y^{(1)}(x_0-(n-2)h)}{h}, \cdots,$$
$$y^{(2)}(x_0-h) = \frac{y^{(1)}(x_0-h)-y^{(1)}(x_0-2h)}{h}, y^{(2)}(x_0) = \frac{y^{(1)}(x_0)-y^{(1)}(x_0-h)}{h}$$

……

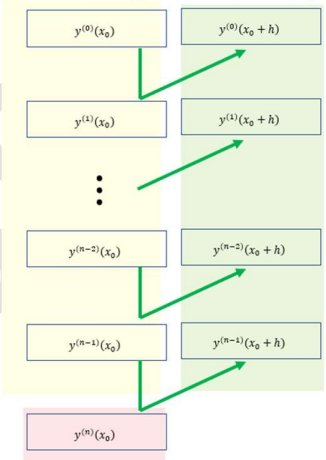
Without finishing the rest higher-order derivatives, we can tell the operation goes in accordance of operation shown in figure above.

Step#2 Get $y^{(n)}(x_0) = G(y(x_0), y^{(1)}(x_0), y^{(2)}(x_0), \cdots, y^{(n-1)}(x_0))$

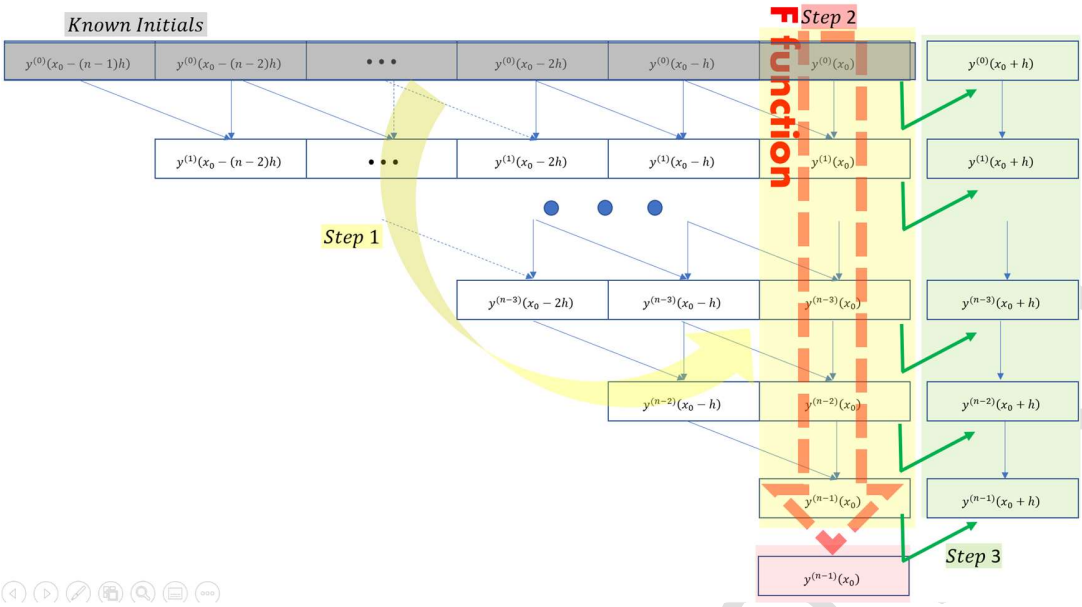


Step#3 Get

$$y^{(0)}(x_0 + h) = y^{(0)}(x_0) + y^{(1)}(x_0) \times h$$
$$y^{(1)}(x_0 + h) = y^{(1)}(x_0) + y^{(2)}(x_0) \times h$$
$$y^{(2)}(x_0 + h) = y^{(2)}(x_0) + y^{(3)}(x_0) \times h$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$y^{(n-2)}(x_0 + h) = y^{(n-2)}(x_0) + y^{(n-1)}(x_0) \times h$$
$$y^{(n-1)}(x_0 + h) = y^{(n-1)}(x_0) + y^{(n)}(x_0) \times h$$



Step4: Until we are satisfied with the range swept by x_0 , assign x_0 with new value $(x_0 + h)$. Go to Step#2



It's often that $y(x_0), y^{(1)}(x_0), y^{(2)}(x_0), \cdots, y^{(n-1)}(x_0)$ are giver as initial conditions, we can save step1 and start from step.2 directly.

Example 1: (2-order linear differential equation)

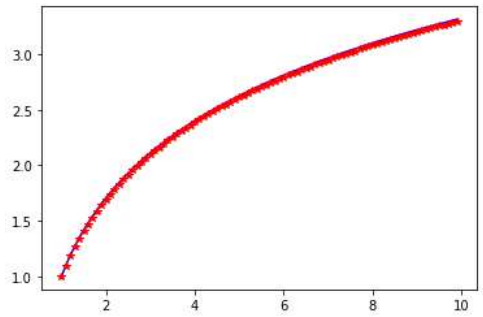
$f''(x)+2f'(x)=-\frac{1}{x^2}+\frac{2}{x}$ with initial values $f(1)=0, f'(1)=1, f''(1)=-1$

(Note: $f(x)=\ln x+1$ is validated to be the actual solution of this equation.)

How about the simulation result of the algorithm?

python code

```
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(1,10,0.1)  # h=0.1
lnValue=np.log(x)+1  # the known actual code directly calling for numpy built-in log functions
# value arrays for f(x), f'(x), f''(x)
f_0_of_x=x*0
f_1_of_x=x*0
f_2_of_x=x*0

# initial value of f(x),f'(x),f''(x) at x=1 (corresponding to n=0)
f_0_of_x[0]=1
f_1_of_x[0]=1
f_2_of_x[0]=-1
h=x[1]-x[0]

#body of iterative algorithm
n=0
while n<len(x)-1:
    f_0_of_x[n+1]=f_1_of_x[n]*h+f_0_of_x[n]
    f_1_of_x[n+1]=f_2_of_x[n]*h+f_1_of_x[n]
    f_2_of_x[n+1]=-2*f_1_of_x[n+1]-1/(x[n+1]*x[n+1])+2/x[n+1]
    n=n+1

# compare the plotting of known actual solution and constructed function by iterative algorith,
plt.plot(x,f_0_of_x,'b')
plt.plot(x,lnValue,'*r')
plt.show()
```



A computational way to get the coefficients of Maclaurin Series.

My computational way to get waveform plot is simple for machines. Though we have been prompted to make good use of calculator, a method without calculate so many point just for one target value is probably desired. Because we know it is not practical to expect close-foirm solution, I decide to revisit Maclaurin Series solution.

The Power Series Solution Professor Thomas suggests is intriscially Maclaurin Series. Since his approach to derive coefficients for nonlinear differential equation fail me, I decide directly work on Maclaurin series directly.

$$Maclaurin\ Series:\ y(x) = y(0) + \frac{y^{(1)}(0)}{1!}x + \frac{y^{(2)}(0)}{2!}x^2 + \frac{y^{(3)}(0)}{3!}x^3 + \cdots + \frac{y^{(n)}(0)}{n!}x^n + \cdots$$

If I could find all the $y^{(k)}(0), \forall k \in N$, problem will be solved! Could I?

Example: $y^{(2)}(x) = -g + a\left[y^{(1)}(x)\right]^2$, given $y(0) = 5000, y'(0) = 0'$

$$y^{(2)}(x) = -g + a\left[y^{(1)}(x)\right]^2 \cdots\cdots (Eq.20)$$

By substituting, $y^{(2)}(0) = -g + a\left[y^{(1)}(0)\right]^2$, we can get
$y^{(2)}(0) = -g$,
How about $y^{(3)}(x)$?

Differentiating Eq(20)

$$y^{(3)}(x) = 2ay^{(1)}(x)y^{(2)}(x) \cdots\cdots (Eq.21)$$

By substituting,

$$y^{(3)}(0) = 2ay^{(1)}(0)y^{(2)}(0) = 0$$

Differentiating Eq(21)

$$y^{(4)}(x) = 2ay^{(2)}(x)y^{(2)}(x) + 2ay^{(1)}(x)y^{(3)}(x) = 2a[y^{(2)}(x)]^2 \cdots\cdots (Eq.22)$$

By substituting,

$$y^{(4)}(0) = 2a[y^{(2)}(0)]^2 = 2ag^2$$

Differentiating Eq(22)

$$y^{(5)}(x) = 4ay^{(2)}(x)y^{(3)}(x) \cdots\cdots (Eq.23)$$

By substituting,

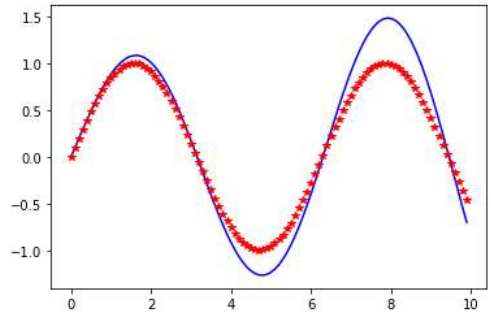
$$y^{(5)}(0) = 0$$

Example 2: (3-order linear differential equation)
$f^{(3)}(x)+f'(x)=0$ with initial values $f(0)=0, f'(0)=1, f''(0)=0, f^{(3)}(0)=-1$

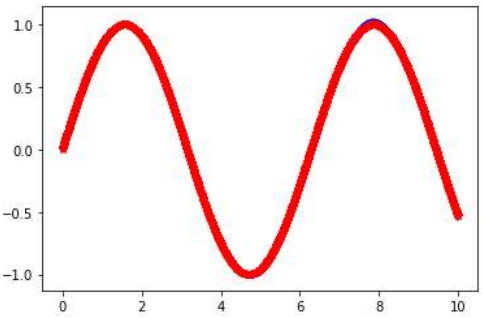
(Note: $f(x)=\sin x$ is validated to be the actual solution of this equation.)

How about the simulation result of the algorithm?

```
***python code***
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(0,10,0.1)  # h=0.1
actualSolutionValue=np.sin(x) # the known actual code directly calling for numpy built-in log functions
# value arrays for f(x), f'(x), f''(x)
f_0_of_x=x*0
f_1_of_x=x*0
f_2_of_x=x*0
f_3_of_x=x*0

# initial value of f(x),f'(x),f''(x) at x=0 (coincidently, corresponding to n=0)
f_0_of_x[0]=0
f_1_of_x[0]=1
f_2_of_x[0]=0
f_3_of_x[0]=-1


h=x[1]-x[0]

#body of iterative algorithm
n=0
while n<len(x)-1:

    f_0_of_x[n+1]=f_1_of_x[n]*h+f_0_of_x[n]
    f_1_of_x[n+1]=f_2_of_x[n]*h+f_1_of_x[n]
    f_2_of_x[n+1]=f_3_of_x[n]*h+f_2_of_x[n]
    f_3_of_x[n+1]=-f_1_of_x[n+1]

    n=n+1

# compare the plotting of known actual solution and constructed function by iterative algorith,
plt.plot(x,f_0_of_x,'b')
plt.plot(x,actualSolutionValue,'*r')
plt.show()
```



After reducing h to 1/10*h



Example 3: (2-order nonlinear differential equation)

$\frac{1}{f''(x)} + \cos\left(f'(x)\right) + e^{f(x)-2x} = -x^2 + cos\frac{1}{x} + \frac{x}{e^{2x}}$ with initial values

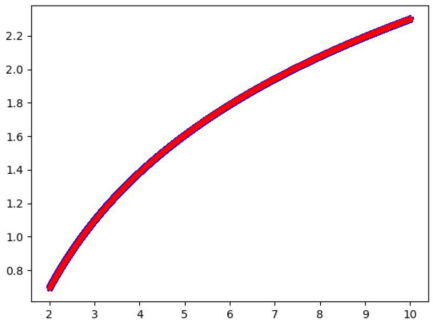
$f(2) = \ln 2, f'(2) = \frac{1}{2}, f''(2) = -\frac{1}{4}$

(Note: $f(x) = \ln x$ is validated to be the actual solution of this equation.)

How about the simulation result of the algorithm?

python code

```
import numpy as np
import matplotlib.pyplot as plt

x=np.arange(2,10,0.0001)
fxValue=np.log(x)

#1/f''(x)+cos(f'(x))+exp(f(x)-2x)=-x^2+cos(1/x)+x/e^2x

h=x[1]-x[0]
f_0_of_x=x*0
f_1_of_x=x*0
f_2_of_x=x*0
f_3_of_x=x*0

f_0_of_x[0]=np.log(2)
f_1_of_x[0]=1/2
f_2_of_x[0]=-1/4

n=0
while n<len(x)-1:
    f_0_of_x[n+1]=f_1_of_x[n]*h+f_0_of_x[n]
    f_1_of_x[n+1]=f_2_of_x[n]*h+f_1_of_x[n]
    f_2_of_x[n+1]=1/(-np.cos(f_1_of_x[n+1])-np.exp(f_0_of_x[n+1]-2*x[n+1])-
(x[n+1])*(x[n+1])+np.cos(1/x[n+1])+x[n+1]/np.exp(2*x[n+1]))

    n=n+1
plt.plot(x,f_0_of_x,'*b')
plt.plot(x,fxValue,'r')
plt.show()
```



Skydiving

Mechanical Case (from University Physics Volume 1):
https://courses.lumenlearning.com/suny-osuniversityphysics/chapter/6-4-drag-force-and-terminal-speed/)

Consider a skydiver falling through air under the influence of gravity.

The two forces acting on him are the force of gravity and the drag force (ignoring the small buoyant force).Drag force FD is proportional to the square of the speed of the object. Mathematically,

$$F_D = \frac{1}{2} C \rho A v^2$$

where *C* is the drag coefficient, *A* is the area of the object facing the fluid, and ρρ is the density of the fluid.

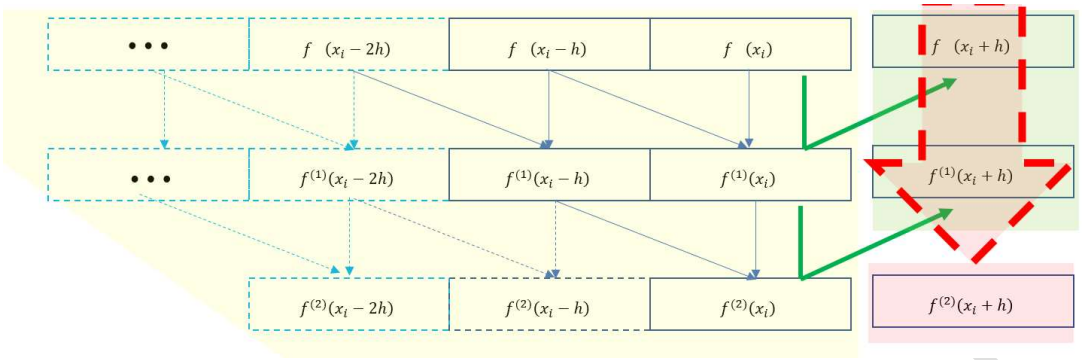
Assume the density of air is ρ=1.21kg/m3 A 75-kg skydiver descending head first has a cross-sectional area of approximately A=0.18m2 and a drag coefficient of approximately C=0.70

$$ma = F_{net} = -mg + \frac{1}{2} C \rho A v^2$$

$$a(t) = -g + \frac{1}{2m} C \rho A v^2(t)$$

$$\frac{d^2 S(t)}{dt^2} = -g + \frac{1}{2m} C \rho A (\frac{dS(t)}{dt})^2$$

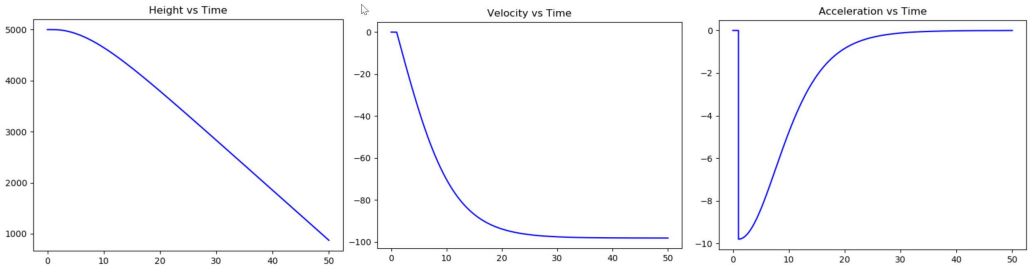
How can we find the time function of height, velocity and acceleration?



Answer:

(by python code)

```
import numpy as np  # To handle mathematics and numerical array, numpy will be come in handy and have better memory-efficiency
import matplotlib.pyplot as plt # To handle plot, matplotlib provide good I/O interfacce

#parameters from the  https://courses.lumenlearning.com/suny-osuniversityphysics/chapter/6-4-drag-force-and-terminal-speed/
C=0.7
rho=1.21
A=0.18
m=75
g=9.8

timeResolution=1e-3
t=np.arange(0,50,timeResolution)  # time array (0s~50S)

S_0_of_t=t*0 # displacement (height) array
S_1_of_t=t*0 # velocity: 1st derivative of displacement
S_2_of_t=t*0 #accelration: 2nd derivative of dispalcement

timeTojump=1 # the timestamp for the sky diver to jump out of airplane
for n in range(0,int(timeTojump/timeResolution),1):# the initial values of displacemnet. We only need the lastest 3 values for 2-order differential equation
    S_0_of_t[n]=5000
for n in range(1,int(timeTojump/timeResolution),1):  # get the 1-order derivatives. We only need the lastest 2 of them,in fact.
    S_1_of_t[n]=(S_0_of_t[n]-S_0_of_t[n-1])/timeResolution
for n in range(2,int(timeTojump/timeResolution),1):   # get the 2-order derivatives. We only need the lastest 1 of them,in fact.
    S_2_of_t[n]=(S_1_of_t[n]-S_1_of_t[n-1])/timeResolution



#body of iterative algorithm
n=int(timeTojump/timeResolution)-1 # this timestamp index is the very one when sky diver jumps to make the differential equation begin to apply
while n<len(t)-1:

    S_0_of_t[n+1]=S_1_of_t[n]*timeResolution+S_0_of_t[n]
    S_1_of_t[n+1]=S_2_of_t[n]*timeResolution+S_1_of_t[n]
    S_2_of_t[n+1]=-g+(1/2)/m*C*rho*A*S_1_of_t[n+1]*S_1_of_t[n+1] # Nonlinear differential equation to define displacement

    n=n+1

# result plotting
plt.plot(t,S_0_of_t,'b')    #plot displacement(height)
plt.title("Height vs Time")
plt.show()
plt.plot(t,S_1_of_t,'b')    #plot velocity
plt.title("Velocity vs Time")
plt.show()
plt.plot(t,S_2_of_t,'b')    #plot acceleration
plt.title("Acceleration vs Time")
plt.show()
```



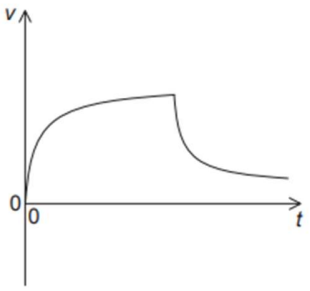
The terminal velocity 98m/s matches the terminal velocity disclosed in the textbook.

Reference:

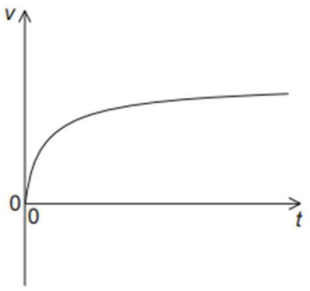
1. IB Diploma Programme Physics Higher level Paper 1 Friday 6 May 2016 (morning) M16/4/PHYSI/HPM/ENG/TZ0/XX, Question 2
   http://ibpastpapers.com/ib-past-papers-12/ or https://mega.nz/fm/HAoDwAJK

2. An aircraft is moving horizontally. A parachutist leaves the aircraft and a few seconds later opens her parachute. Which graph shows the variation of the vertical speed v with time t for the parachutist from the time she leaves the aircraft until just before landing?

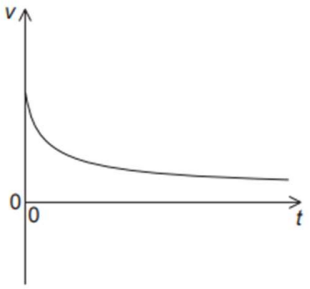
A.



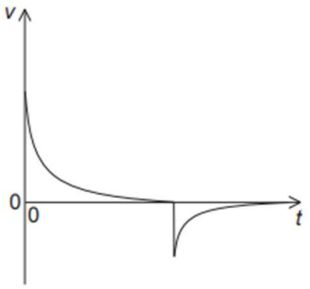
B.



C.



D.



(2) Drag force and terminal speed

https://courses.lumenlearning.com/suny-osuniversityphysics/chapter/6-4-drag-force-and-terminal-speed/

Terminal Velocity

Some interesting situations connected to Newton's second law occur when considering the effects of drag forces upon a moving object. For instance, consider a skydiver falling through air under the influence of gravity. The two forces acting on him are the force of gravity and the drag force (ignoring the small buoyant force). The downward force of gravity remains constant regardless of the velocity at which the person is moving. However, as the person's velocity increases, the magnitude of the drag force increases until the magnitude of the drag force is equal to the gravitational force, thus producing a net force of zero. A zero net force means that there is no acceleration, as shown by Newton's second law. At this point, the person's velocity remains constant and we say that the person has reached his **terminal velocity** (v_{T}). Since F_{D} is proportional to the speed squared, a heavier skydiver must go faster for F_{D} to equal his weight. Let's see how this works out more quantitatively.

At the terminal velocity,

$$F_{\mathrm{net}} = mg - F_{\mathrm{D}} = ma = 0.$$

Thus,

$$mg = F_{\mathrm{D}}.$$

Using the equation for drag force, we have

$$mg = \tfrac{1}{2}C\rho A v_{\mathrm{T}}^2.$$

Solving for the velocity, we obtain

$$v_{\mathrm{T}} = \sqrt{\tfrac{2mg}{\rho CA}}.$$

Assume the density of air is $\rho = 1.21\ \mathrm{kg/m^3}$. A 75-kg skydiver descending head first has a cross-sectional area of approximately $A = 0.18\ \mathrm{m^2}$ and a drag coefficient of approximately $C = 0.70$. We find that

$$v_{\mathrm{T}} = \sqrt{\frac{2(75\ \mathrm{kg})(9.80\ \mathrm{m/s^2})}{(1.21\ \mathrm{kg/m^3})(0.70)(0.18\ \mathrm{m^2})}} = 98\ \mathrm{m/s} = 350\ \mathrm{km/h}.$$

(3)
Solutions of 1st order differential equation from Haeses

(Chapter 25, Diffrential Equations
from Mathematics – Analysis and Approaches HL 2 – Haese 2019 . ,),
startimg fropm page 685

(4)
Solutions of 2nd order differential equation from Thomas' Calculus

(Chapter 17, SECOND – ORDER DIFFERENTIAL EQUATIONS
from Thomas' Calculus by George B. Thomas, Jr.),
starting from page 1
http://www.math.wisc.edu/~passman/Thomas12e_WebChap17.pdf