Problem Set 7 **R08244002 Shin-Rong, Tsai**

- ## Solving Ordinary Differential Equation

1. Result

Run the code *solve_ODE.c*.

```
cindytsai@TURQUOISEA /cygdrive/d/GitHub/Computational_Physics/Assignment/ProblemSet_7
$ gcc -o a.out solve_ODE.c

cindytsai@TURQUOISEA /cygdrive/d/GitHub/Computational_Physics/Assignment/ProblemSet_7
$ ./a.out
Step    x       Exact_sol     Euler       ModifiedEuler    ImprovedEuler    RungeKutta3rd
-------------------------------------------------------------------------------------------
1       0.01    0.01000       0.01000     0.01000          0.01000          0.01000
2       0.02    0.02000       0.02000     0.02000          0.02000          0.02000
3       0.03    0.03001       0.03001     0.03001          0.03001          0.03001
4       0.04    0.04002       0.04001     0.04002          0.04002          0.04002
5       0.05    0.05004       0.05003     0.05004          0.05004          0.05004
6       0.06    0.06007       0.06006     0.06007          0.06007          0.06007
7       0.07    0.07011       0.07009     0.07011          0.07012          0.07011
8       0.08    0.08017       0.08014     0.08017          0.08017          0.08017
9       0.09    0.09024       0.09020     0.09024          0.09025          0.09024
10      0.10    0.10033       0.10029     0.10033          0.10034          0.10033
11      0.11    0.11045       0.11039     0.11044          0.11045          0.11045
12      0.12    0.12058       0.12051     0.12058          0.12058          0.12058
13      0.13    0.13074       0.13065     0.13074          0.13074          0.13074
```

2. Discussion

Equation

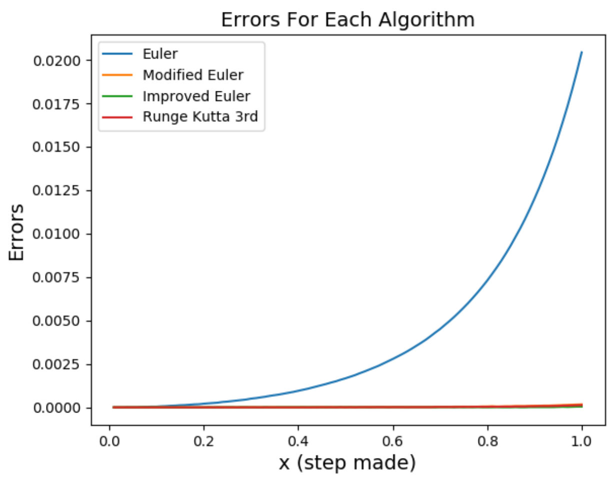
$$\frac{dy}{dx} = y^2 + 1 \;\; and \;\; y(0) = 0$$

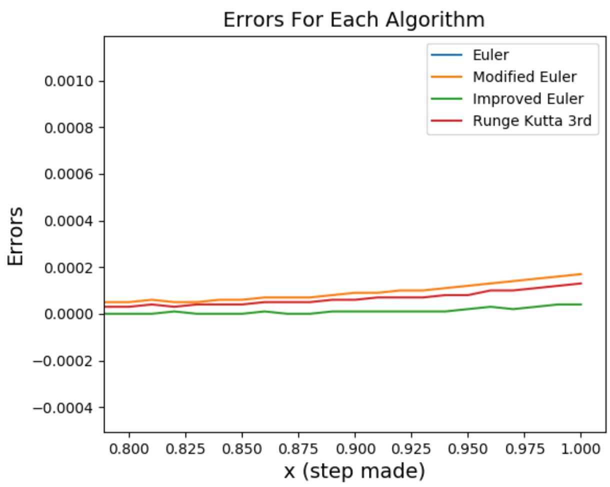
Gives

$$\int \frac{1}{y^2+1} dy = \int dx = x + const. = \tan^{-1} y \Rightarrow y = \tan x$$

And compare each algorithm's solution to the exact solution, where $error = |exact \;\; solution - algorithm|$, we have:

(In file *solve_ODE_errorplot.py*)





We can see that as each step is approximate by algorithms, the error accumulates. So errors will grow in every algorithm. We would want to avoid using Euler method for solving ODE, since the error boost up. It is interesting that the error size is

$$Modified\ \ Euler > Runge\ \ Kutta\ \ 3rd > Improved\ \ Euler$$

Improved Euler method is the same as Runge Kutta 2nd order method. I guess that different shape of the differential equations has different best approximation method.

- **Prove the 3rd order and 4th order Runge-Kutta Formulas**

1. Prove of 3rd order Runge-Kutta Formula

From

$$\frac{dy}{dx} = f(x,y) \Rightarrow y(x_0+h) = y(x_0) + \int_{x_0}^{x_0+h} f(x,y(x))$$

We have

$$y(x_0+h) \cong y(x_0) + h[\alpha \cdot f(x_0,y_0) + \beta \cdot f(x_0+\gamma \cdot h, y_0 + \delta \cdot hf(x_0,y_0))]$$

And let

$$y_0 = y(x_0)\ \ and\ \ f(x_0,y_0) = f_0$$

Do the Taylor Expansion of $f(x_0+\cdots, y_0+\cdots)$ and $y(x_0+h)$ to the 3rd order,

$$LHS = y_0 + h\left\{\alpha f_0 + \beta\left[f_0 + \gamma h\frac{\partial f}{\partial x} + \delta h f_0\frac{\partial f}{\partial y} + \gamma^2h^2\frac{\partial^2 f}{\partial x^2} + 2\gamma\delta h^2 f_0\frac{\partial^2 f}{\partial x\partial y} + \delta^2h^2{f_0}^2\frac{\partial^2 f}{\partial y^2}\right]_{x_0,y_0}\right\}$$

$$\Rightarrow LHS \cong y_0 + h(\alpha+\beta)f_0 + \beta\gamma h^2\frac{\partial f}{\partial x} + \beta\delta h^2 f_0\frac{\partial f}{\partial y} + \frac{1}{2}\beta\gamma^2h^3\frac{\partial^2 f}{\partial x^2} + \frac{1}{2}$$

$$\cdot 2\beta\gamma\delta h^3 f_0\frac{\partial^2 f}{\partial x\partial y} + \frac{1}{2}\beta\delta^2h^3{f_0}^2\frac{\partial^2 f}{\partial y^2}\ \text{ at }\ (x_0,y_0)$$

$$RHS \cong y_0 + h\left.\frac{dy}{dx}\right|_{x_0} + \frac{h^2}{2!}\left.\frac{d^2y}{dx^2}\right|_{x_0} + \frac{h^3}{3!}\left.\frac{d^3y}{dx^3}\right|_{x_0}$$

$$\Rightarrow RHS = y_0 + hf_0 + \frac{h^2}{2!}\left[\frac{\partial f}{\partial x} + f_0\frac{\partial f}{\partial y}\right] + \frac{h^3}{3!}\left[\frac{\partial^2 f}{\partial x^2} + 2f_0\frac{\partial^2 f}{\partial x\partial y} + {f_0}^2\frac{\partial^2 f}{\partial y^2}\right]$$

where $\frac{d^2y}{dx^2} = \frac{df(x,y)}{dx} = \frac{\partial f}{\partial x} + f\frac{\partial f}{\partial y}$ and $\frac{d^3y}{dx^3} = \frac{d}{dx}\left[\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial y}\right] = \frac{\partial^2 f}{\partial x^2} + 2f\frac{\partial^2 f}{\partial x\partial y} + f^2\frac{\partial^2 f}{\partial y^2} + \frac{df}{dx}\frac{\partial f}{\partial y}$. We neglect the last term $\frac{df}{dx}\frac{\partial f}{\partial y}$, since it's a higher order.

Then compare LHS and RHS term by term, we have

$\alpha + \beta = 1$

$\beta\gamma = \frac{1}{2}\ \ and\ \ \beta\delta = \frac{1}{2}$

$\beta\gamma^2 = \frac{1}{3}\ \ and\ \ \beta\gamma\delta = \frac{1}{3}\ \ and\ \ \beta\delta^2 = \frac{1}{3}$

We can solve it and find out that:

$$\alpha = \frac{1}{4} \quad \beta = \frac{3}{4} \quad \gamma = \frac{2}{3} \quad \delta = \frac{2}{3}$$

Which is 3rd order Runge-Kutta Formula

$$y(x_0 + h) \cong y(x_0) + h\left[\frac{1}{4} \cdot f(x_0, y_0) + \frac{3}{4} \cdot f(x_0 + \frac{2}{3} \cdot h, y_0 + \frac{2}{3} \cdot hf(x_0, y_0))\right]$$

2. Prove of 4th order Runge-Kutta Formula

We know that the 4th order Runge-Kutta Formula looks just like Simpson's Rule when doing integration and take $f(x, y) = f(x)$, so we can assume that the equation is

$$y(x_0 + h) \cong y(x_0) + \frac{h}{6}[f_0 + 2f_1 + 2f_2 + f_3]$$

And

$$f_0 = f(x_0, y_0)$$

$$f_1 = f(x_0 + \frac{h}{2}, y_0 + \alpha h f_0)$$

$$f_2 = f(x_0 + \frac{h}{2}, y_0 + \beta h f_1)$$

$$f_3 = f(x_0 + h, y_0 + \gamma h f_2)$$

We assumed the formula to be like this, in the way let they will shrink to Simpson's Rule when we take $f(x, y) = f(x)$.

And again same as 3rd order Runge-Kutta Formula, we do the Taylor Expansion at (x_0, y_0) to find the coefficients.

$$f_1 \cong f_0 + \frac{h}{2}\frac{\partial f}{\partial x} + \alpha h f_0 \frac{\partial f}{\partial y} + \frac{1}{2!}(\frac{h^2}{4}\frac{\partial^2 f}{\partial x^2} + \frac{h}{2}\alpha h f_0 \frac{\partial f}{\partial x \partial y} + \alpha^2 h^2 f_0^2 \frac{\partial^2 f}{\partial y^2})$$

$$f_2 \cong f_0 + \frac{h}{2}\frac{\partial f}{\partial x} + \beta h f_1 \frac{\partial f}{\partial y} + \frac{1}{2!}(\frac{h^2}{4}\frac{\partial^2 f}{\partial x^2} + \frac{h}{2}\beta h f_1 \frac{\partial f}{\partial x \partial y} + \beta^2 h^2 f_1^2 \frac{\partial^2 f}{\partial y^2})$$

$$f_3 \cong f_0 + h\frac{\partial f}{\partial x} + \gamma h f_2 \frac{\partial f}{\partial y} + \frac{1}{2!}(h^2\frac{\partial^2 f}{\partial x^2} + h\gamma h f_2 \frac{\partial f}{\partial x \partial y} + \gamma^2 h^2 f_2^2 \frac{\partial^2 f}{\partial y^2})$$

Since for f_0 , f_x and f_y $\cdots$ are all higher order terms, we neglect them in approximating f_2 and f_3. Thus f_1 and f_2 inside f_2 and f_3 respectively are f_0.

Bring them back to $y(x_0 + h)$ and compare to the Taylor Expansion of itself,

$$y(x_0 + h) \cong y(x_0) + \frac{h}{6}[f_0 + 2f_1 + 2f_2 + f_3]$$

$$y(x_0 + h) \cong y(x_0) + hf_0 + \frac{h^2}{2!}\left[\frac{\partial f}{\partial x} + f_0\frac{\partial f}{\partial y}\right] + \frac{h^3}{3!}\left[\frac{\partial^2 f}{\partial x^2} + 2f_0\frac{\partial^2 f}{\partial x \partial y} + {f_0}^2\frac{\partial^2 f}{\partial y^2}\right]$$

Compare the term $\frac{\partial f}{\partial y}$, we have $2\alpha + 2\beta + \gamma = 3$.

Compare the term $\frac{\partial f}{\partial x \partial y}$, we have $\alpha + \beta + \gamma = 2$.

Compare the term $\frac{\partial^2 f}{\partial y^2}$, we have $\alpha^2 + \beta^2 + \frac{\gamma^2}{2} = 1$

Solve these equation, finally we get

$$\alpha = \frac{1}{2} \quad \beta = \frac{1}{2} \quad \gamma = 1$$

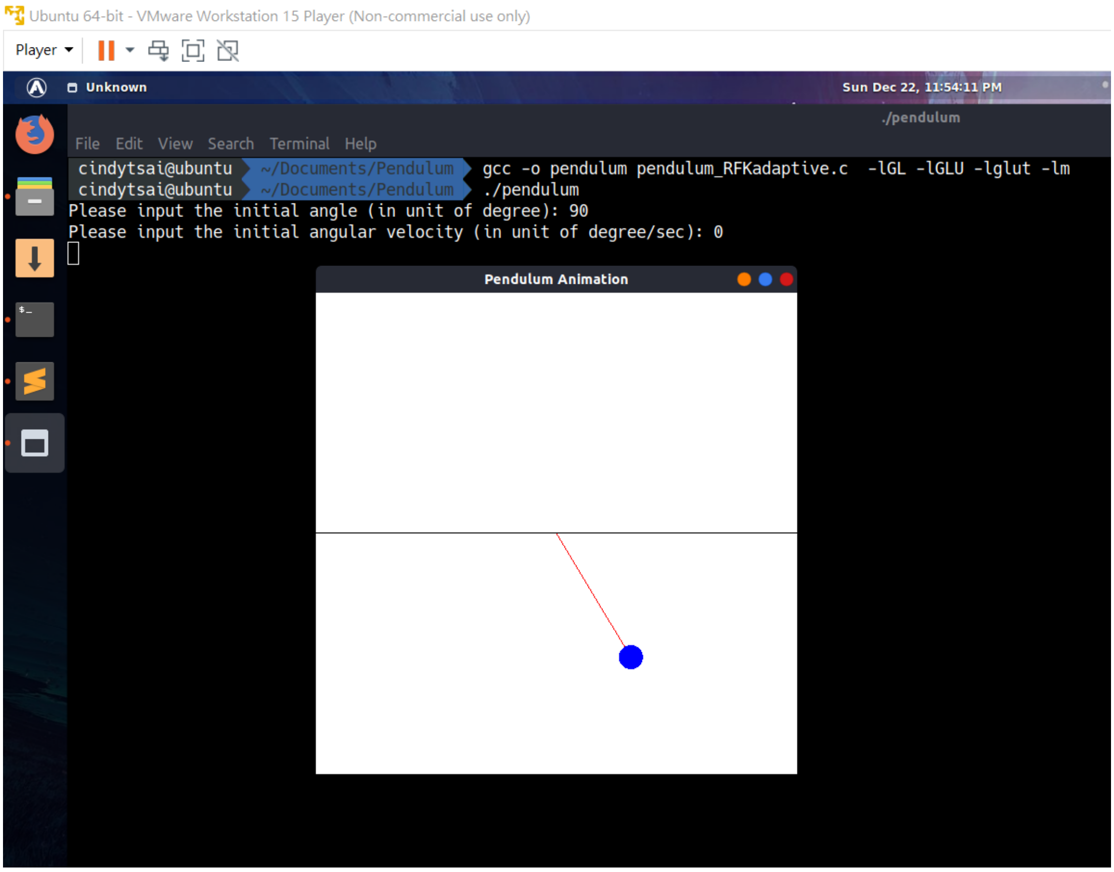
Which give us 4th order Runge-Kutta Formula.

- **Physical Pendulum**

1. Result

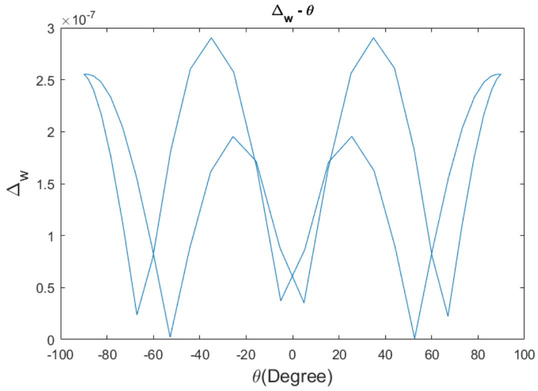
My settings on Cygwin for OpenGL doesn't seem to work, so I use VMware Virtual Machine with Ubuntu Linux instead.

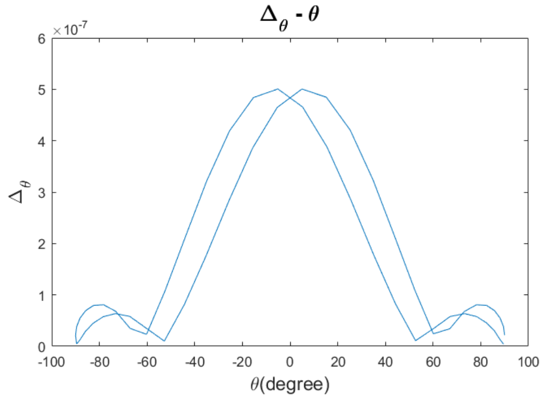
Compile and run the code *pendulum_RFKadaptive.c*.



2. Discussion

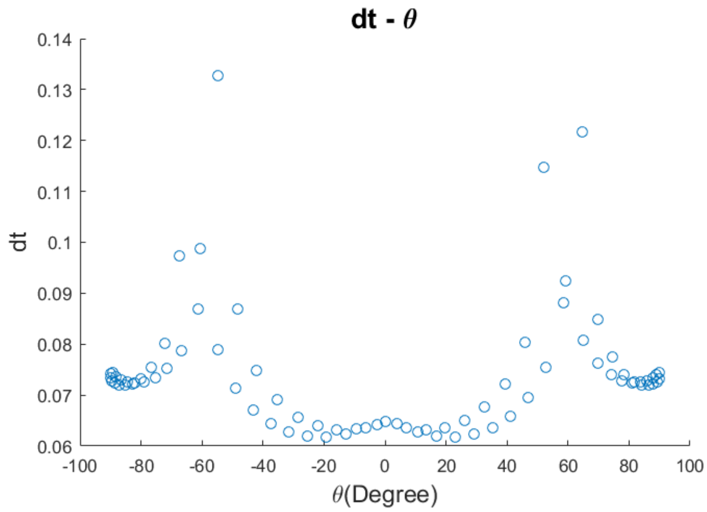
We can see that w and θ are complementary, due to energy conservation.





We can see that we need adaptive step size, if we want them to meet an accuracy under $\Delta_0 \approx 10^{-7}$. The above diagram gives how much they vary under h and $\frac{h}{2}$.

If we define the truncation error as $\Delta \equiv \left|w_{\frac{h}{2}} - w_h\right| + \left|\theta_{\frac{h}{2}} + \theta_h\right|$, so that we consider both affect from w and θ, the two only physical parameter, we would have the adaptive step size like this:



We can see that if we add Δ_w and Δ_θ together with respect to θ, we would see that they are complement to the $\mathrm{dt} - \theta$ diagram, just as what we want the program should do.