User Guide for Code to solve systems of ODEs using Runge-Kutta method On systems running Microsoft Windows 10

December 16, 2016

Abstract

This is only the guide to use our code for the purpose said in the title, for more theoretical details, please see [1] and [2]. For reference to the `Fortran` programming language, you may want to visit [3].

Contents

1 Preparation

To use our code, which was written in `Fortran`, we need the following things on a system running Windows 10:

1. A text editor: You can use any text editor, but in this guide, we will use `gedit`, you can download it at `https://wiki.gnome.org/Apps/Gedit#Download`.
2. `gfortran`: You can download an unofficial build of GCC 5 source at `https://gcc.gnu.org/wiki/GFortranBinaries#Windows`, or more explicitly, `http://users.humboldt.edu/finneyb/gfortran-windows-20140629.exe`
3. `gnuplot`: You can download it at `gnuplot` primary download site on SourceForge: `https://sourceforge.net/projects/gnuplot/files/gnuplot/`

• **Important note:** When installing `gnuplot`, you must tick the option "Add application directory to your PATH environment variable".

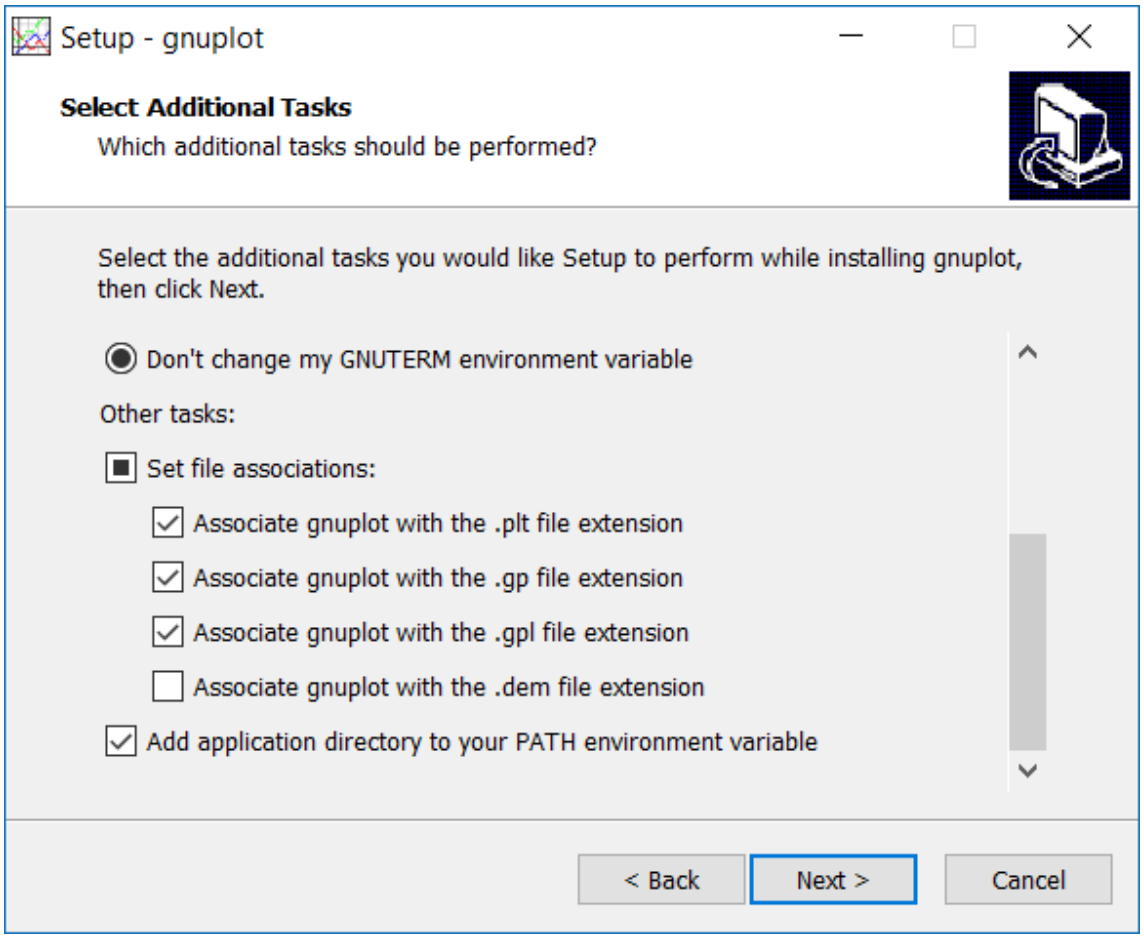


Figure 1: Tick the option "Add application directory to your PATH environment variable"

• **However, if you miss that option**, you can add the `gnuplot` directory to the PATH environment variable manually by the following steps. Otherwise, you may skip to section 2.

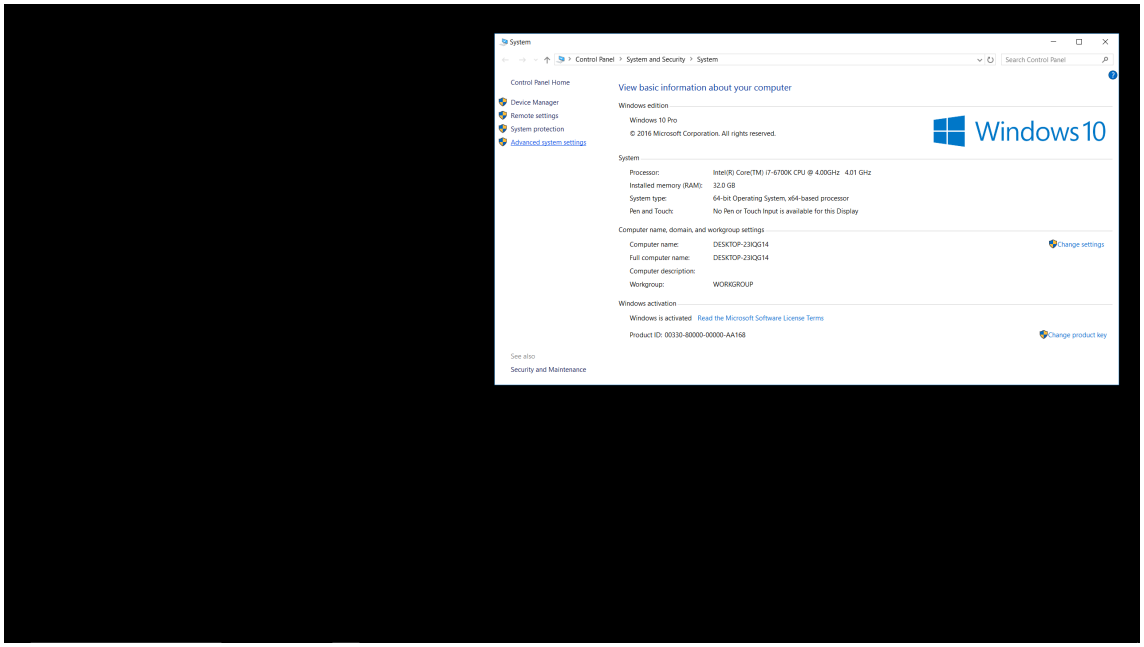


Figure 2: Press **Windows key + Pa/Br** to open system properties window

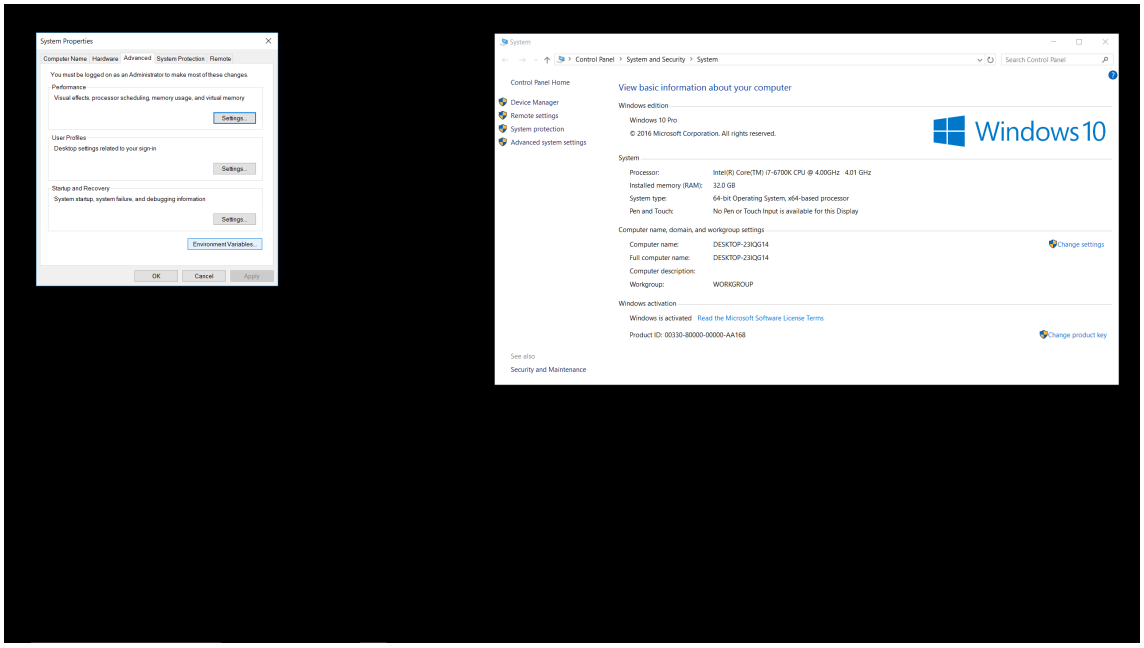


Figure 3: Click on **Advanced system settings** on the left side to open "Advanced" tab in "System properties"

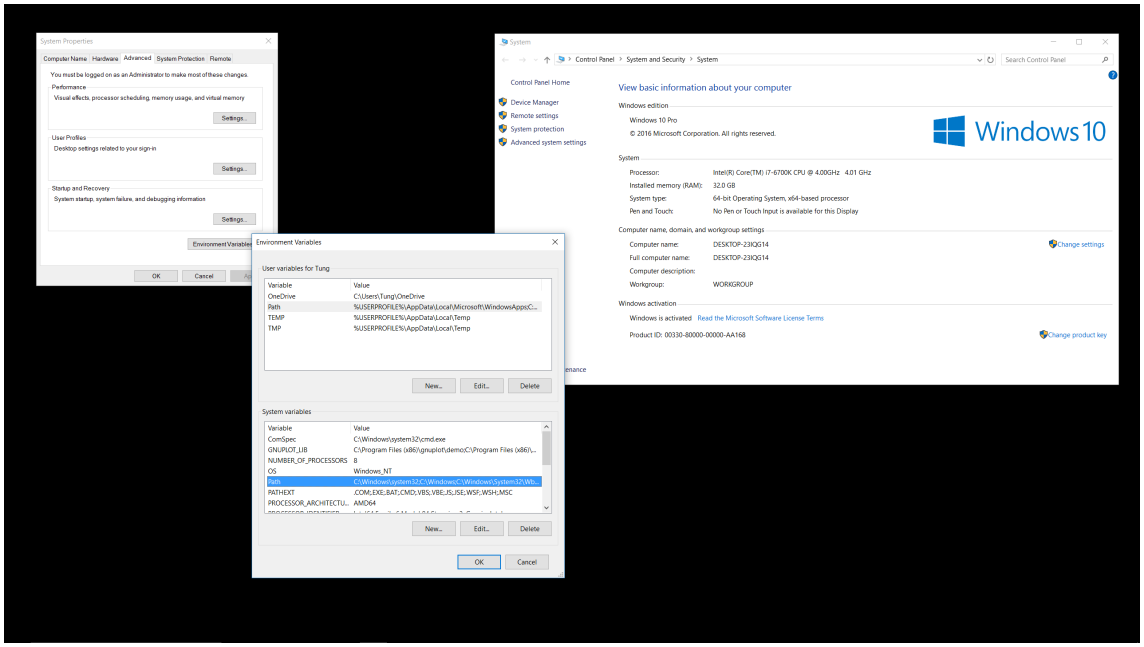


Figure 4: Click on **Environment Variables** to open "Environment Variables" window and highlight the **Path** variable in the "System variable" section

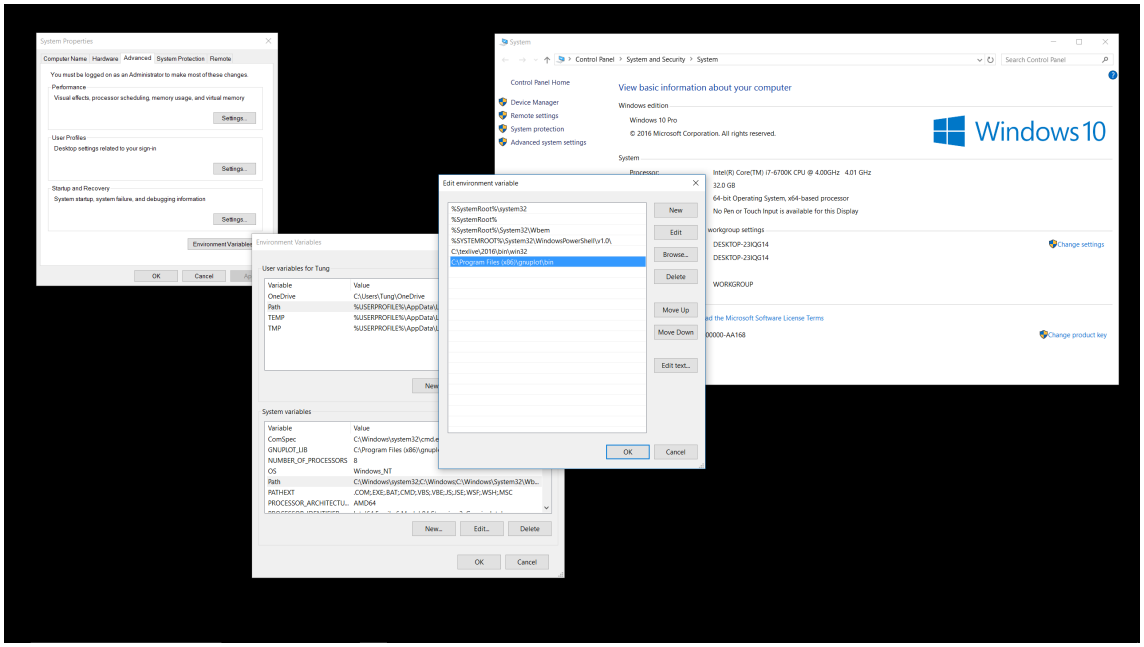


Figure 5: Click the **Edit** button to open "Edit environment variable" window and add the `gnuplot` directory there

2 The source files

We wrote multiple source files with different purposes:

- `rkfts.f90`: `Fortran` source code for Runge-Kutta method using fixed time step
- `rkats.f90`: `Fortran` source code for Runge-Kutta method using adaptive time step

- `data_plot.plt` and `data_plot_dependency.plt` :`gnuplot` commands to plot the results
- `customf.f90`: source code for module containing the specific equation you want to solve, how to modify it will be discussed later.
- The other `*.f90`: `Fortran` source code for modules containing the example equations to solve, we included modules for: Curtiss-Hirschfelder equation, Brusselator equation, Van der Pol equation,...

3 Specifying equation by modifying the source files

As mentioned before, there are module files containing the example equations, you may want to take a look at those and [3] for reference. This section will explain about what you need to specify in the module file. For example, a module file look like this

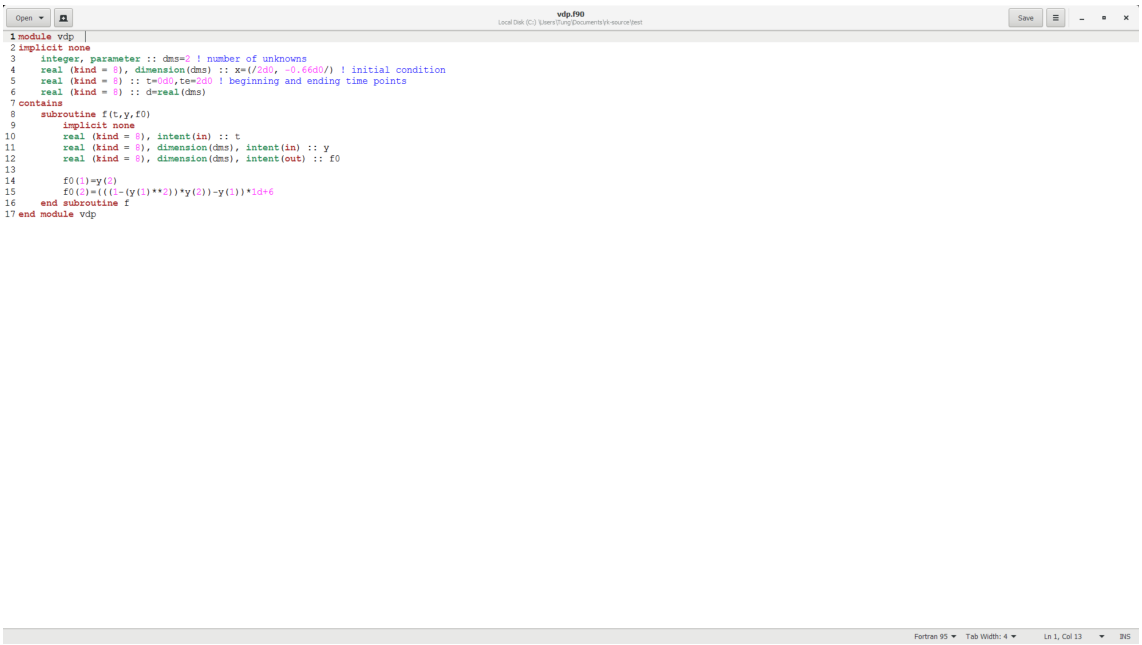
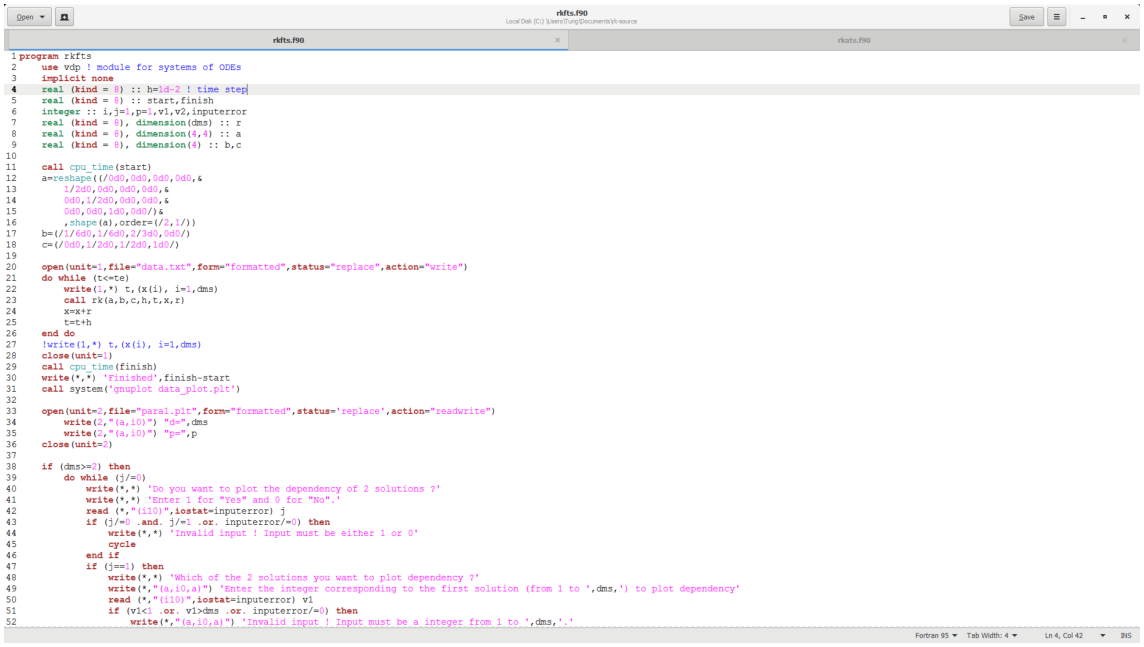


Figure 6: Module file of the Van der Pol equation

As commented in the file, line 3, 4, 4 contain the information about "number of unknowns", "initial conditions", "beginning and ending time points". The information in Figure 6 means that we are considering a equation with 2 unknown variables, the beginning and ending time points are $t_0 = 0$ and $t_e = 2$ respectively, the initial condition is $f(0, y) = (2, -0.66)$.
Where $y = y_1, y_2$ is the vector of unknown variables and

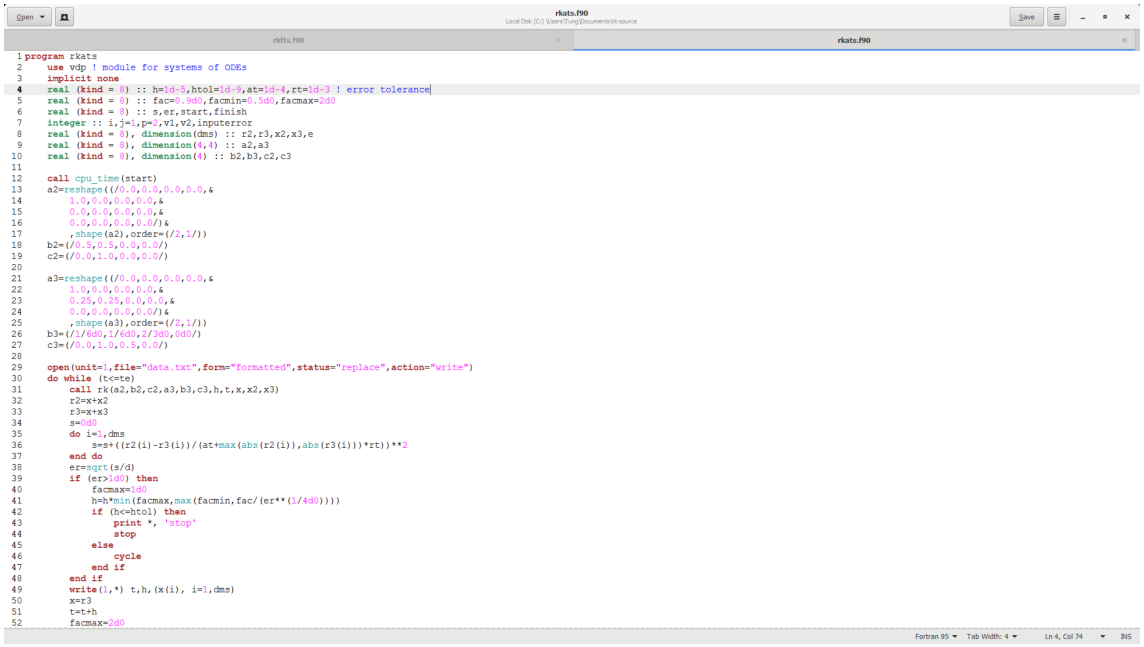
$$f(t, y) = \left[y_2(t), \frac{(1 - y_1(t))^2 y_2(t) - y_1(t)}{10^{-6}} \right]$$

More ever, we may also need to modify the time step in `rkfts.f90` or the error tolerance in `rkats.f90`, for more details about this, refer to [1] and [2].



```
program rkfts
  use vdp ! module for systems of ODEs
  implicit none
  real (kind = 8) :: h=1d-2 ! time step
  real (kind = 8) :: start,finish
  integer :: i,j=1,p=1,v1,v2,inputerror
  real (kind = 8), dimension(dms) :: r
  real (kind = 8), dimension(4,4) :: a
  real (kind = 8), dimension(4) :: b,c

  call cpu_time(start)
  a=reshape((/0d0,0d0,0d0,0d0,&
      1/2d0,0d0,0d0,0d0,&
      0d0,1/2d0,0d0,0d0,&
      0d0,0d0,1d0,0e0/)&
      ,shape(a),order=(/2,1/))
  b=(/1/6d0,1/6d0,2/3d0,0d0/)
  c=(/0d0,1/2d0,1/2d0,1d0/)

  open(unit=1,file="data.txt",form="formatted",status="replace",action="write")
  do while (t<=te)
      write(1,*) t,(x(i), i=1,dms)
      call rk(a,b,c,h,t,x,r)
      x=x+r
      t=t+h
  end do
  !write(1,*) t,(x(i), i=1,dms)
  close(unit=1)
  call cpu_time(finish)
  write(*,*) 'Finished',finish-start
  call system('gnuplot data_plot.plt')

  open(unit=2,file="para1.plt",form="formatted",status='replace',action="readwrite")
      write(2,"(a,i0)") "d=",dms
      write(2,"(a,i0)") "p=",p
  close(unit=2)

  if (dms>=2) then
      do while (j/=0)
          write(*,*) 'Do you want to plot the dependency of 2 solutions ?'
          write(*,*) 'Enter 1 for "Yes" and 0 for "No".'
          read (*,"(i10)",iostat=inputerror) j
          if (j/=0 .and. j/=1 .or. inputerror/=0) then
              write(*,*) 'Invalid input ! Input must be either 1 or 0'
              cycle
          end if
          if (j==1) then
              write(*,*) 'Which of the 2 solutions you want to plot dependency ?'
              write(*,"(a,i0,a)") 'Enter the integer corresponding to the first solution (from 1 to ',dms,') to plot dependency'
              read (*,"(i10)",iostat=inputerror) v1
              if (v1<1 .or. v1>dms .or. inputerror/=0) then
                  write(*,"(a,i0,a)") 'Invalid input ! Input must be a integer from 1 to ',dms,'.'
```

Figure 7: rkfts.f90



```
program rkats
  use vdp ! module for systems of ODEs
  implicit none
  real (kind = 8) :: h=1d-5,htol=1d-9,at=1d-4,rt=1d-3 ! error tolerance
  real (kind = 8) :: fac=0.9d0,facmin=0.5d0,facmax=2d0
  real (kind = 8) :: s,er,start,finish
  integer :: i,j=1,p=1,v1,v2,inputerror
  real (kind = 8), dimension(dms) :: r2,r3,x2,x3,e
  real (kind = 8), dimension(4,4) :: a2,a3
  real (kind = 8), dimension(4) :: b2,b3,c2,c3

  call cpu_time(start)
  a2=reshape((/0.0,0.0,0.0,0.0,&
      1.0,0.0,0.0,0.0,&
      0.0,0.0,0.0,0.0,&
      0.0,0.0,0.0,0.0/)&
      ,shape(a2),order=(/2,1/))
  b2=(/0.5,0.5,0.0,0.0/)
  c2=(/0.0,1.0,0.0,0.0/)

  a3=reshape((/0.0,0.0,0.0,0.0,&
      1.0,0.0,0.0,0.0,&
      0.25,0.25,0.0,0.0,&
      0.0,0.0,0.0,0.0/)&
      ,shape(a3),order=(/2,1/))
  b3=(/1/6d0,1/6d0,2/3d0,0d0/)
  c3=(/0.0,1.0,0.5,0.0/)

  open(unit=1,file="data.txt",form="formatted",status="replace",action="write")
  do while (t<=te)
      call rk(a2,b2,c2,a3,b3,c3,h,t,x,x2,x3)
      r2=x+x2
      r3=x+x3
      s=0d0
      do i=1,dms
          s=s+((r2(i)-r3(i))/(at+max(abs(r2(i)),abs(r3(i)))*rt))**2
      end do
      er=sqrt(s/d)
      if (er>1d0) then
          facmax=1d0
          h=h*min(facmax,max(facmin,fac/(er**(1/4d0))))
          if (h<=htol) then
              print *, 'stop'
              stop
          else
              cycle
          end if
      end if
      write(1,*) t,h,(x(i), i=1,dms)
      x=r3
      t=t+h
      facmax=2d0
```

Figure 8: rkats.f90

4 Using the code

• **Step 1:** Decide which equation you want to solve and whether you want to solve it using Runge-Kutta method with fixed time step or adaptive time step. Put the file containing the module specifies that equation and the file corresponding to the method of your choice together with `data_plot.plt` and `data_plot_dependency.plt` in a same folder.

For example: You want to solve the Van der Pol equation specified in `vfp.f90` using Runge-Kutta method with adaptive time step, you need to put `rkats.f90`, `vdp.f90`, `data_plot.plt` and

`data_plot_dependency.plt` in the same folder. From now on, we will use this example for sake of convenience.

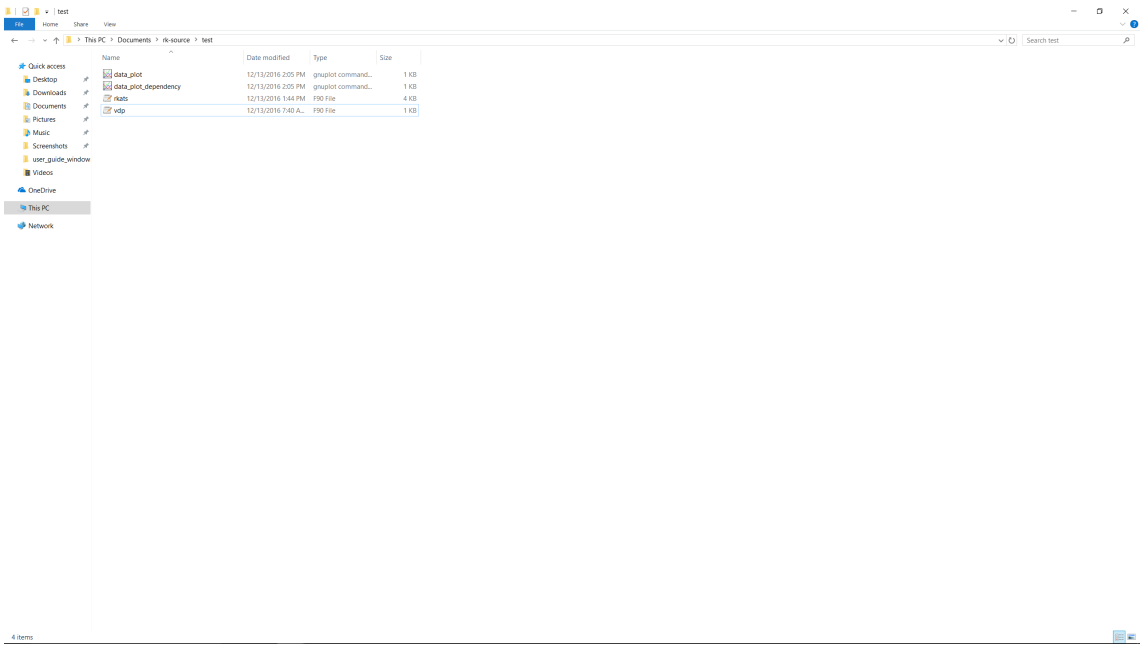


Figure 9: Put the needed files in the same folder

• **Step 2:** Open the folder in **Command Prompt**

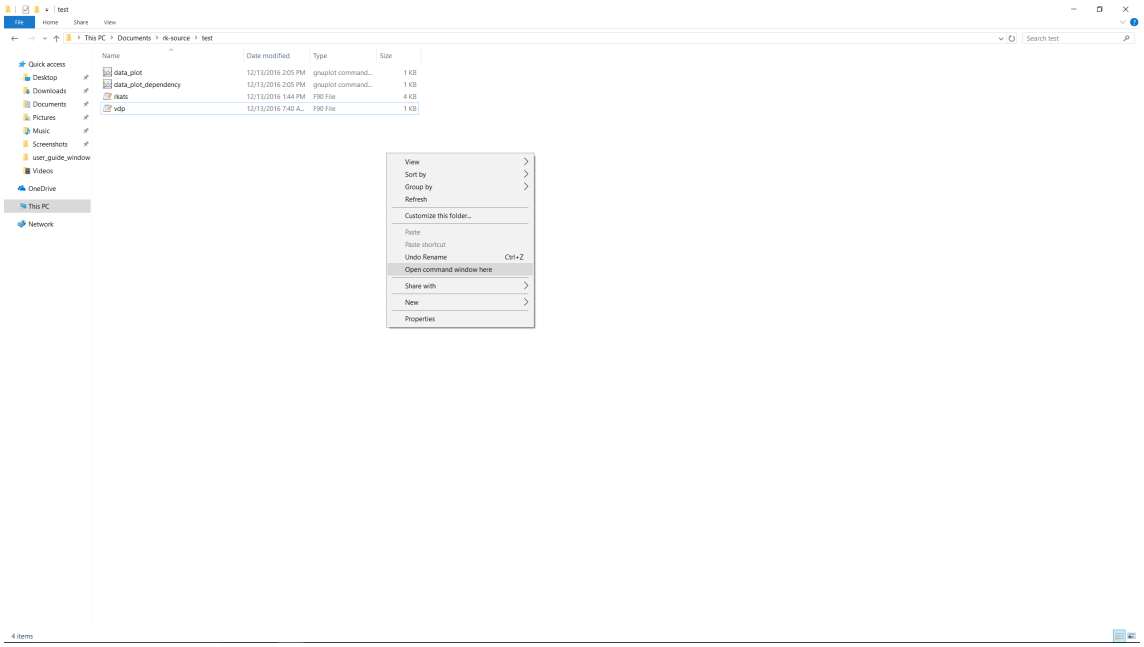


Figure 10: Press **Shift + Right click** at the empty space and select `Open command window here`

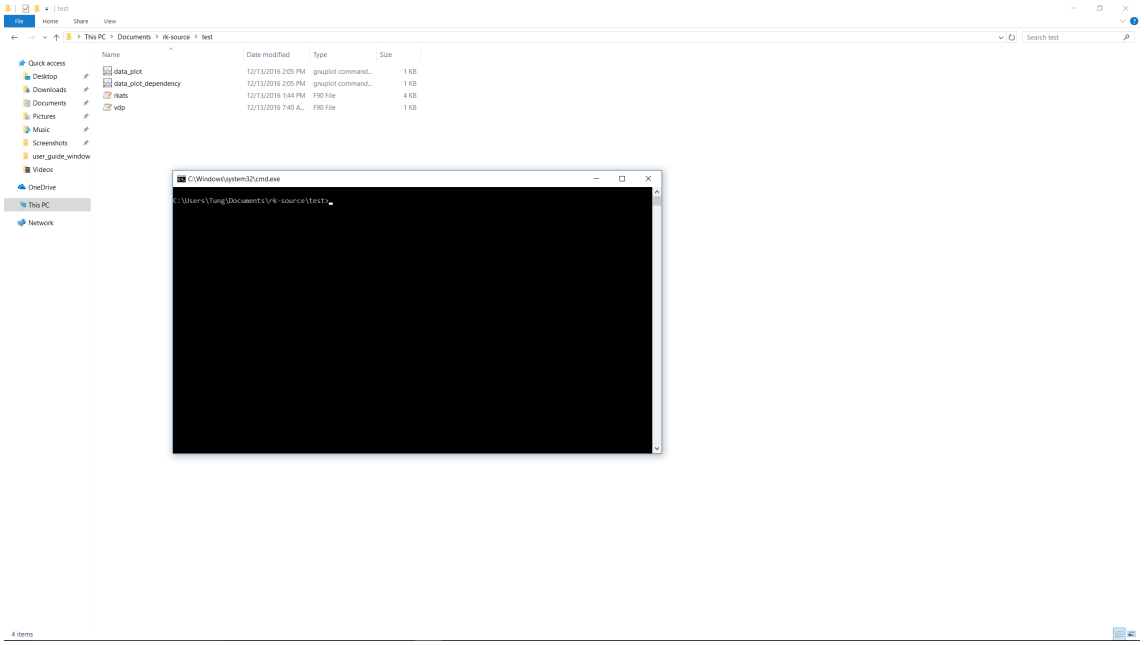
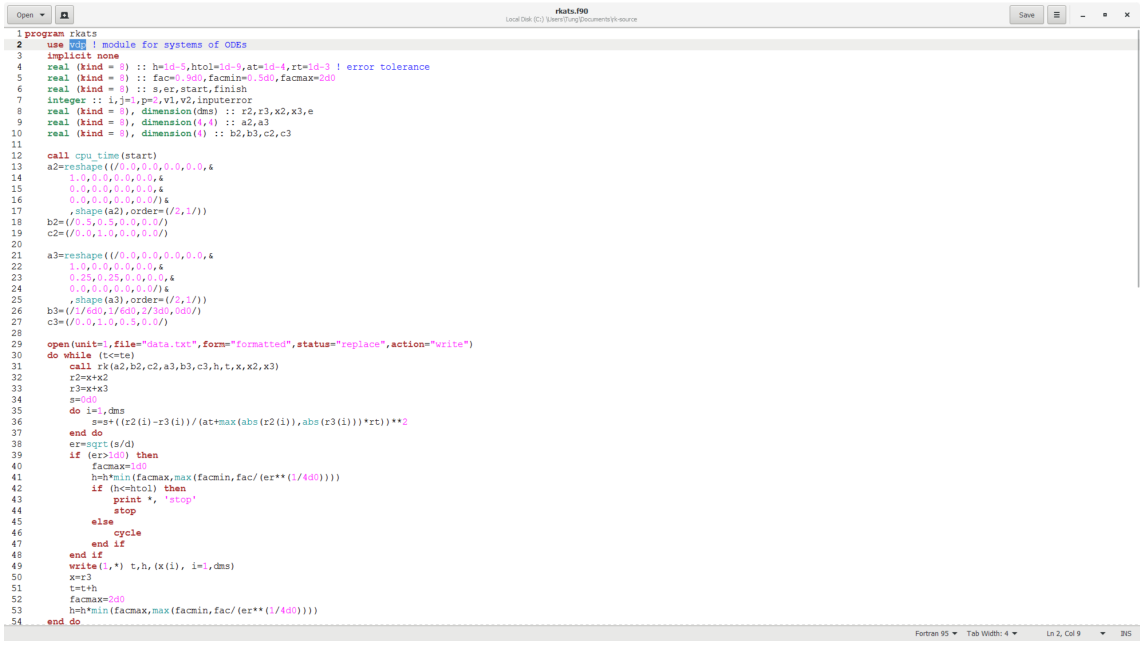


Figure 11: The folder is now opened in Command Prompt

• **Step 3:** Open `rkats.f90` in a text editor and make sure that it use the module for your equation of choice (`vdp.f90` for the Van der Pol equation in this example) by modifying the second line: from `use [something]` to `use vdp`



```
program rkats
    use vdp ! module for systems of ODEs
    implicit none
    real (kind = 8) :: h=1d-5,htol=1d-9,at=1d-4,rt=1d-3 ! error tolerance
    real (kind = 8) :: fac=0.9d0,facmin=0.5d0,facmax=2d0
    real (kind = 8) :: s,er,start,finish
    integer :: i,j=1,p=2,v1,v2,inputerror
    real (kind = 8), dimension(dms) :: r2,r3,x2,x3,e
    real (kind = 8), dimension(4,4) :: a2,a3
    real (kind = 8), dimension(4) :: b2,b3,c2,c3

    call cpu_time(start)
    a2=reshape((/0.0,0.0,0.0,0.0,&
        1.0,0.0,0.0,0.0,&
        0.0,0.0,0.0,0.0,&
        0.0,0.0,0.0,0.0/)&
        ,shape(a2),order=(/2,1/))
    b2=(/0.5,0.5,0.0,0.0/)
    c2=(/0.0,1.0,0.0,0.0/)

    a3=reshape((/0.0,0.0,0.0,0.0,&
        1.0,0.0,0.0,0.0,&
        0.25,0.25,0.0,0.0,&
        0.0,0.0,0.0,0.0/)&
        ,shape(a3),order=(/2,1/))
    b3=(/1/6d0,1/6d0,2/3d0,0d0/)
    c3=(/0.0,1.0,0.5,0.0/)

    open(unit=1,file="data.txt",form="formatted",status="replace",action="write")
    do while (t<=te)
        call rk(a2,b2,c2,a3,b3,c3,h,t,x,x2,x3)
        r2=x+x2
        r3=x+x3
        s=0d0
        do i=1,dms
            s=s+((r2(i)-r3(i))/(at+max(abs(r2(i)),abs(r3(i)))*rt))**2
        end do
        er=sqrt(s/d)
        if (er>1d0) then
            facmax=1d0
            h=h*min(facmax,max(facmin,fac/(er**(1/4d0))))
            if (h<=htol) then
                print *, 'stop'
                stop
            else
                cycle
            end if
        end if
        write(1,*) t,h,(x(i), i=1,dms)
        x=r3
        t=t+h
        facmax=2d0
        h=h*min(facmax,max(facmin,fac/(er**(1/4d0))))
    end do
```

Figure 12: The result look like this

• **Step 4:** Type `gfortran vdp.f90 rkats.f90 -o main.out` to make the output file named `main.out`, of course you can choose other names as well.

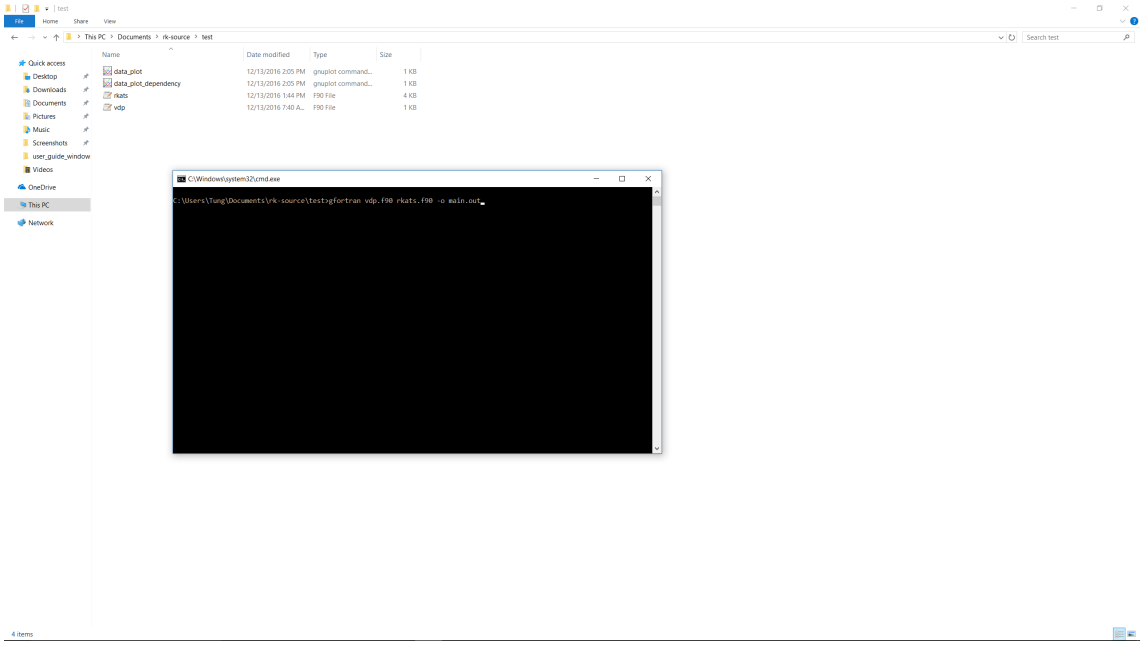


Figure 13: Typing the command in the Command Prompt

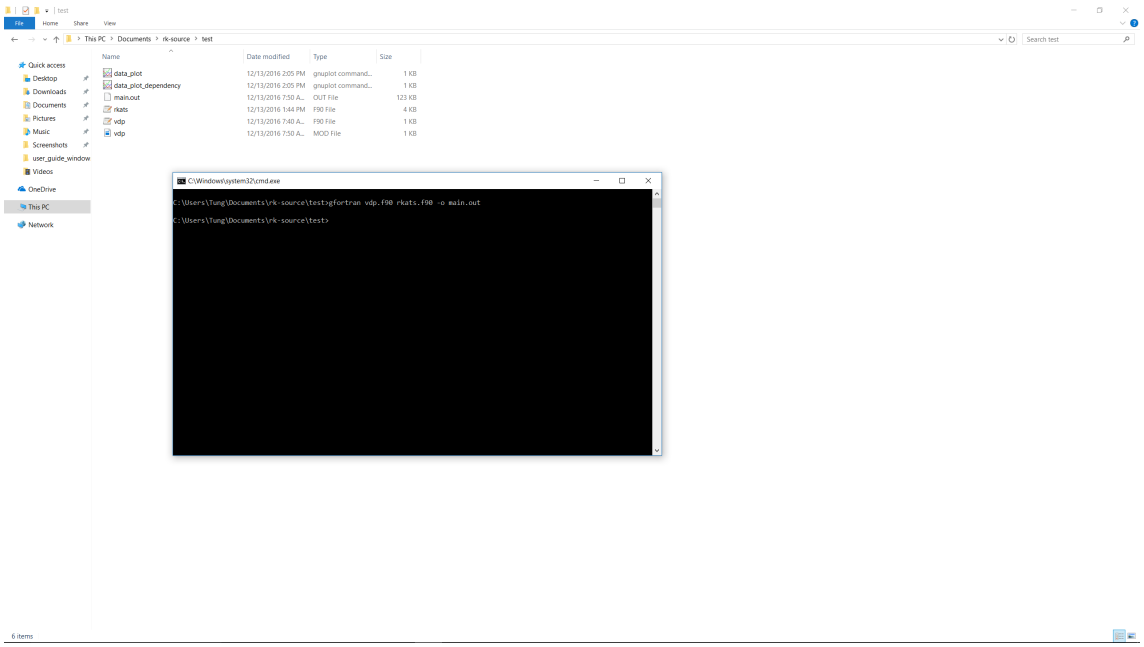


Figure 14: Completing the command will result in some new files

- **Step 5:** Type `.\main.out` to run the file `main.out`

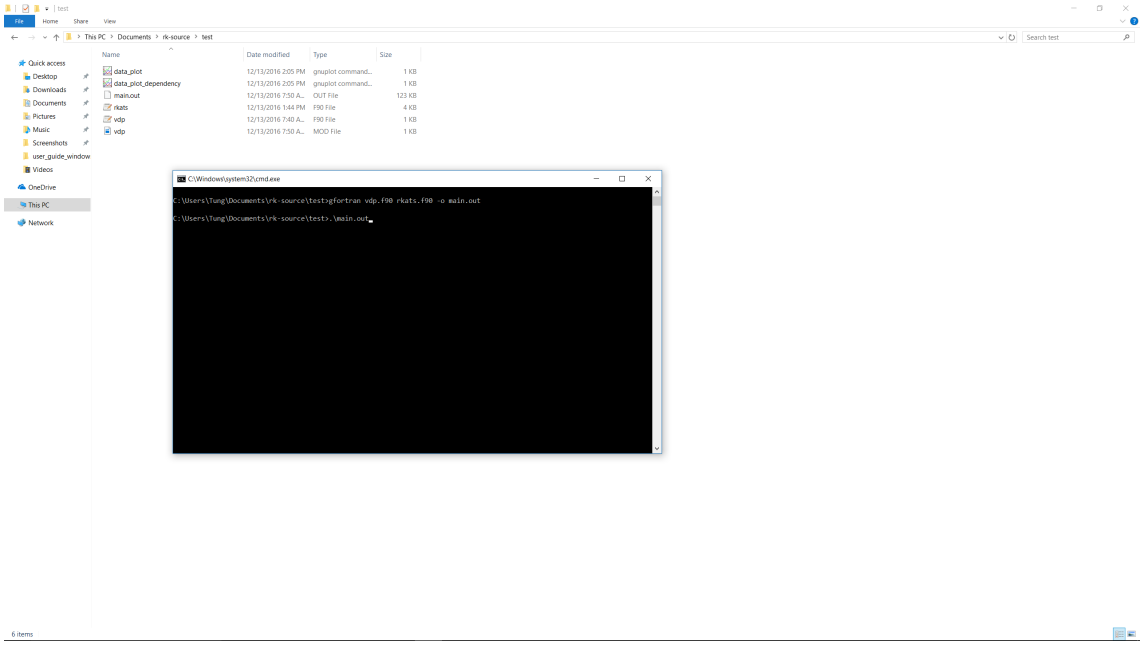


Figure 15: Typing the command in the Command Prompt

The command will run the program to solve the equation. The numerical result will be save into the file `data.txt` and the plots of the size of the step h at time t (only appear if you use adaptive time step method), the **solutions of the unknown variables of the equation** will appear, those plots would be saved in the same folder with the other file in the form of `*.pdf` and `*.tex` files. Additionally, several other files created by `gnuplot` are needed to compile the newly created `*.tex` files.

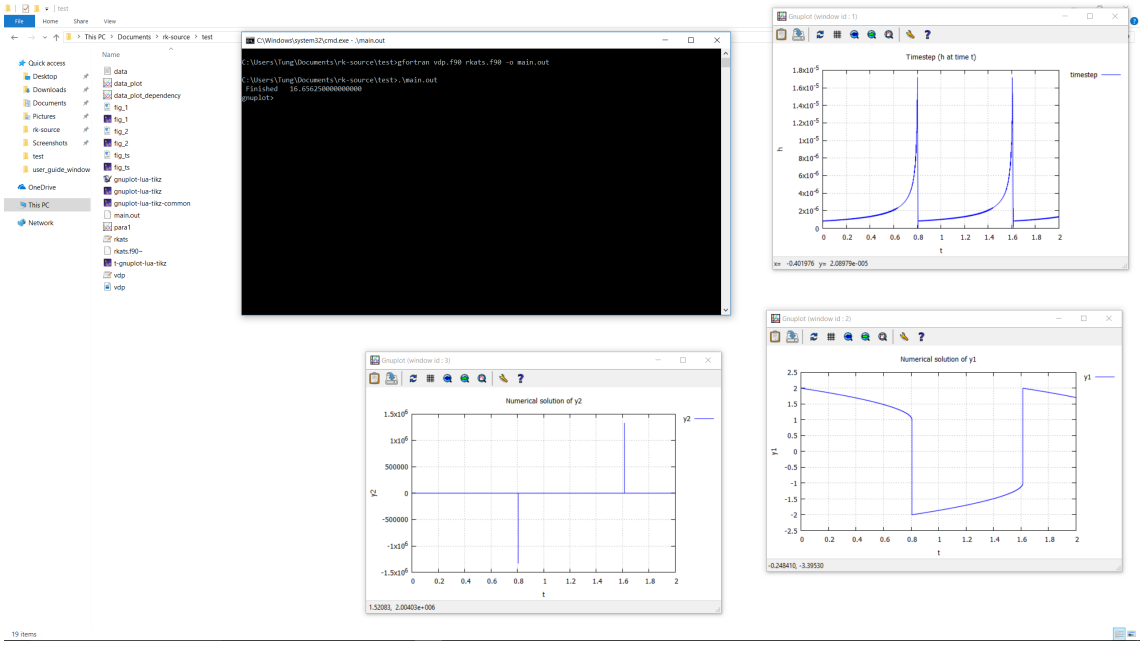


Figure 16: The result look like this

To continue, type `quit` in the Command Prompt to quit the current `gnuplot` windows

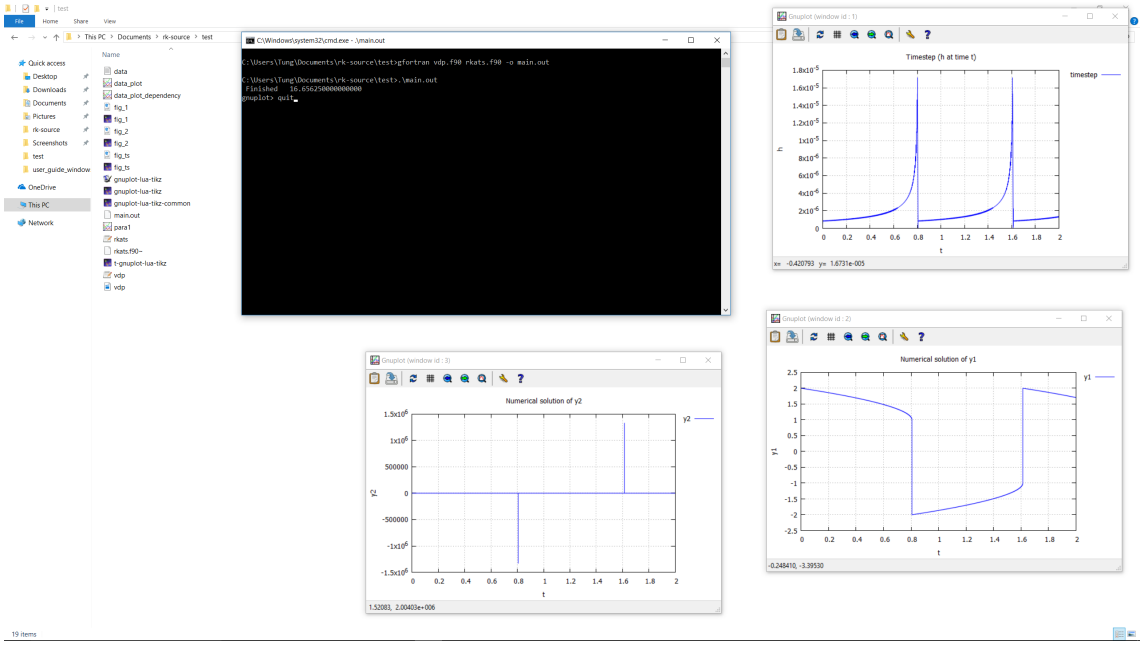


Figure 17: Typing the command in the Command Prompt

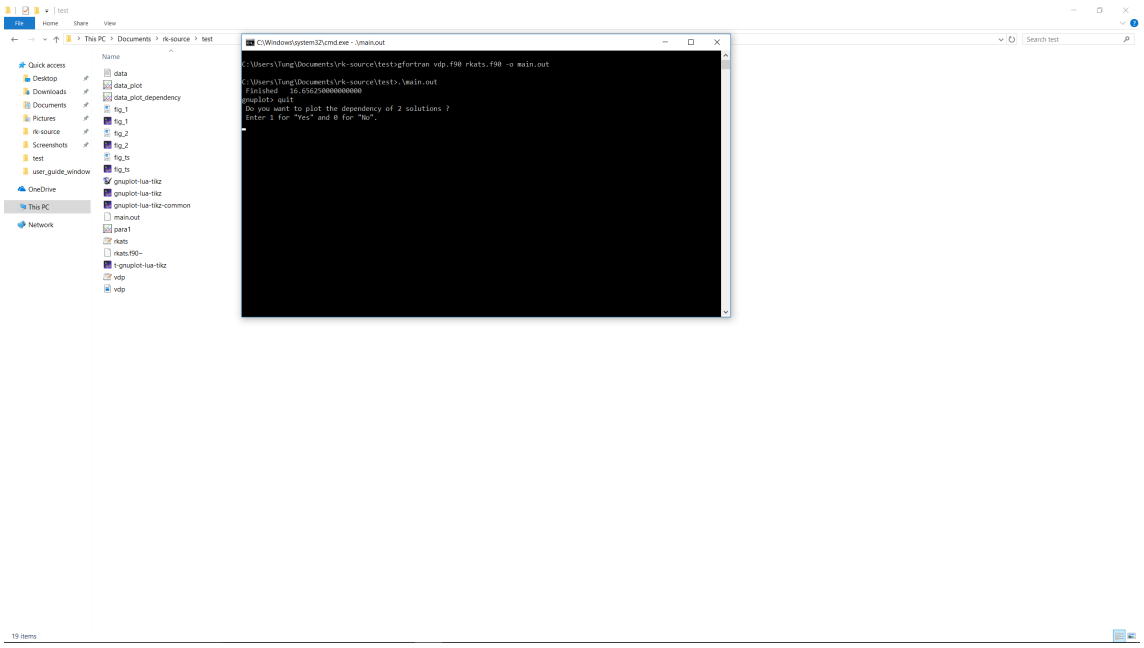


Figure 18: The result look like this

If the equation has more than one unknown variable (like in this example), the program will also ask whether if you want to plot the dependency between the solutions of those variables. This will be discussed in the next step.

• **Step 6*:** Plot the dependency between the solutions of variables. To do this, you just need to follow the instructions in the **Command Prompt**. Suppose you want to plot the dependency of y_1 and y_2, type 1 in the **Command Prompt** to confirm your intention, then type in 1 and 2 respectively for the next 2 line.

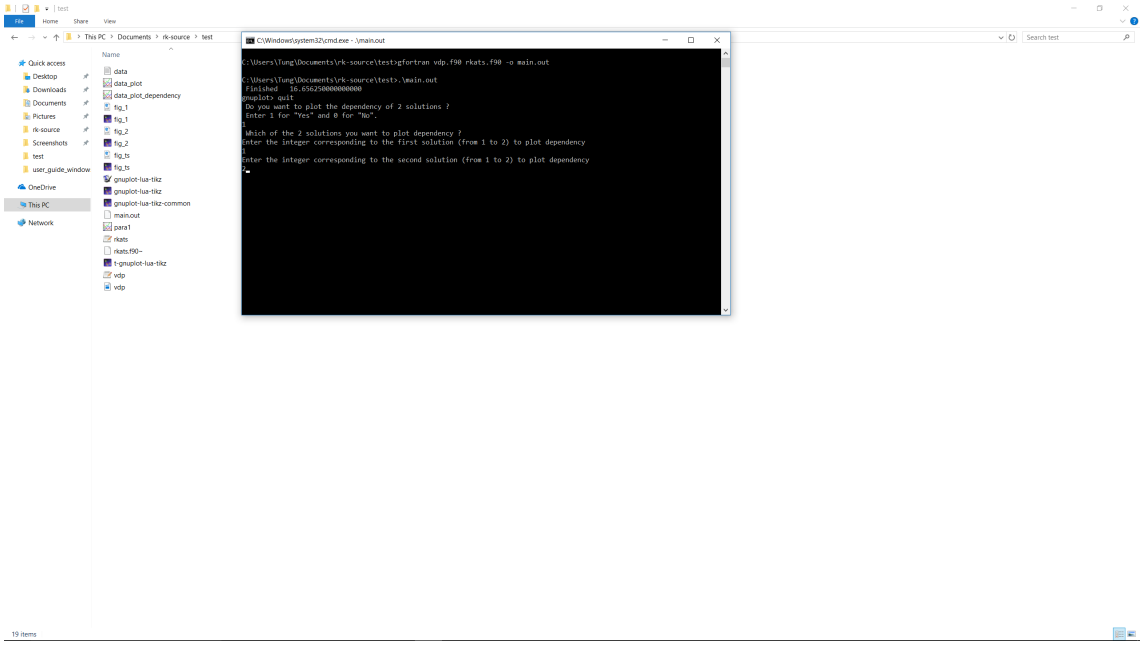


Figure 19: Typing the integers correspond to your choices in the Command Prompt

The result will be a new plot of the dependency, like before, additional files of the plot will appear.

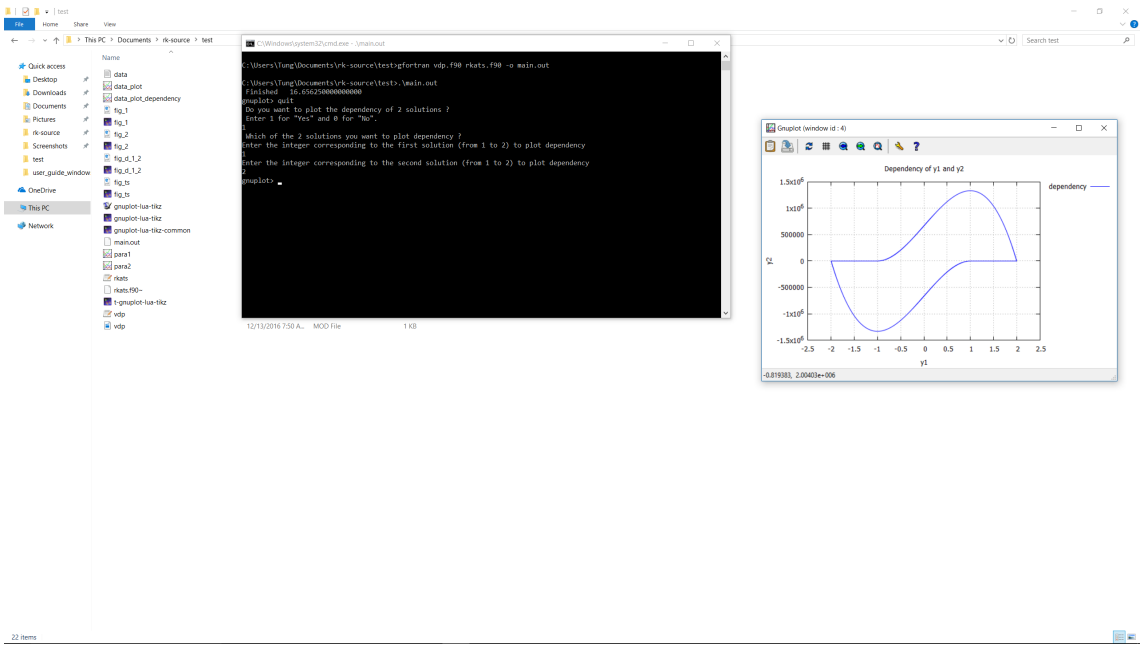


Figure 20: The result look like this

Similarly to above, to continue, type `quit` in the Command Prompt to quit the current `gnuplot` windows.

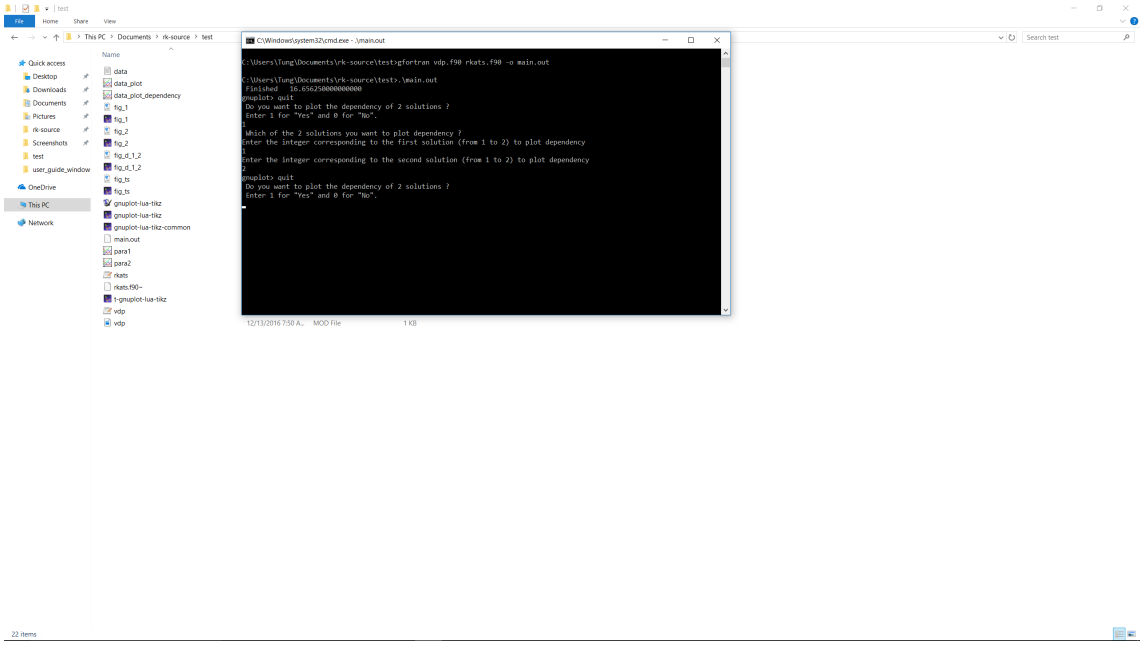


Figure 21: The result look like this

Now, if you want to continue plotting dependency (maybe if the equation you want to solve has more than 2 variables), type 1, or if you want to stop, type 0.

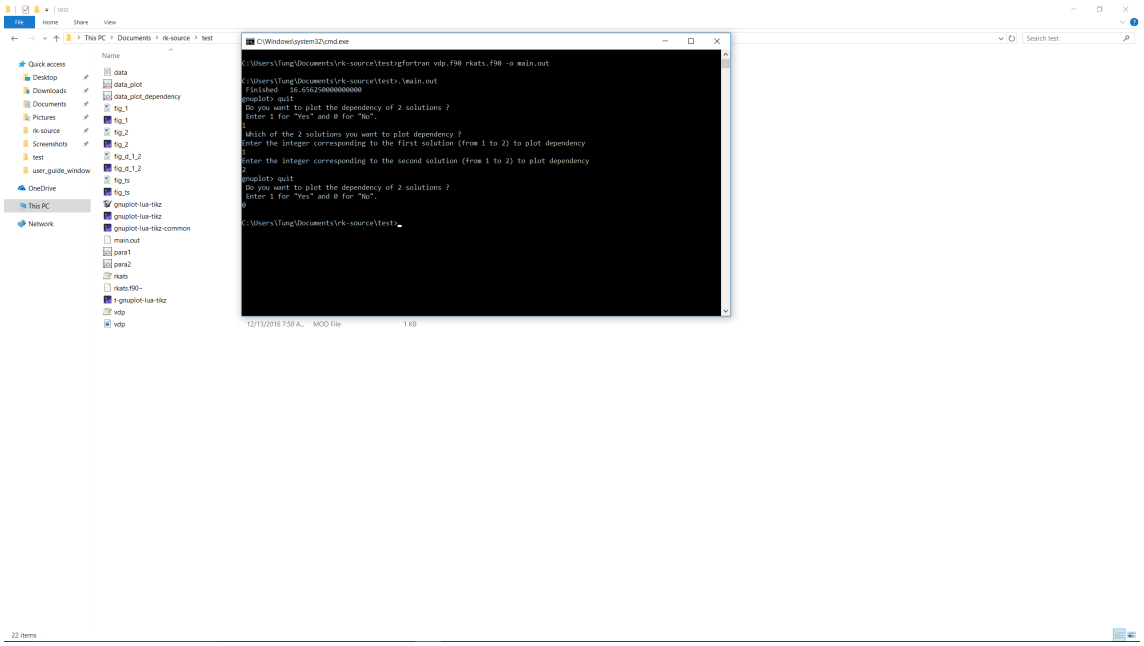


Figure 22: The program is stopped

References

[1] Tan Trung Nguyen, Frederique Laurent, Rodney Fox, Marc Massot. *Solution of population balance equations in applications with fine particles: mathematical modeling and numerical schemes*. Journal of Computational Physics, Elsevier, 2016, 325

[2] Nguyen Quan Ba Hong, Doan Tran Nguyen Tung, Nguyen An Thinh, *Runge Kutta Methods for Ordinary Differential Equations*, 2016.

[3] `http://fortranwiki.org/fortran/show/HomePage`