Problem 1

Terminal velocity is when the acceleration vanishes. That is, $v_x=0$ and $v_z=-v_f$ with (from the condition $a_z=0$) $g=\alpha v_f^2$, or $\alpha=g/v_f^2$.

Completed code:

```python
# Homework 3, PHZ 5156, September 13, 2005

from scipy import *

def f_air(yn,tn):
    #RHS of the first-order coupled ODE's
    #Input tn is a particular time.
    #Input yn = array( (x,z,vx,vz) ) at that time.
    #Output: array( (vx,vz,ax,az) ) at that time.
    x,z,vx,vz = yn
    v = sqrt(vx**2 + vz**2)
    ax = -alpha*v*vx
    az = -alpha*v*vz - g
    f = array( (vx,vz,ax,az) )
    return f

def rk4(f,y0,t):
    #Implements 4th order Runge Kutta, using an external f(y).
    #Usage: y=rk4(f,y0,t)
    #Input f is the name of the "RHS" function.
    #Input y0 is an array that contains the initial conditions.
    #Notice that y0 can have arbitrary length.
    #Input t is an array of times.
    #Output array y has shape len(t) x len(y0) where
    #y[n, :] contains y(tn) for all computed times.

    y = zeros( (len(t),len(y0)), Float )
    y[0, :] = y0
    for n in arange(0,len(t)-1):
        tn,yn = t[n],y[n, :]
        h1 = tau*f(yn,tn)
        h2 = tau*f(yn+h1/2.,tn+tau/2.)
        h3 = tau*f(yn+h2/2.,tn+tau/2.)
        h4 = tau*f(yn+h3,tn+tau)
        y[n+1, :] = yn + h1/6.+h2/3.+h3/3.+h4/6.
    return y
```
# main routine

# Problem 2: Debug f_air
# The following gives v=5 and so ax=-15, az=-30

g = 10.
alpha = 1.
print "Problem 2 test: f_air = ",f_air( array((0.,0.,3.,4.)),5.)

# Set up constants

g = 9.8  # [m/s^2]
alpha = 0.  # [m/s^2]
Vfinal = 120.*1000./3600.
alpha = g / Vfinal**2  # drag coefficient [1/m] for Prob 5

# Set up initial conditions

v0 = 30000. / 3600.  # m/s
vx0 = v0 * cos(30.*pi/180.)  # m/s
vz0 = v0 * sin(30.*pi/180.)  # m/s
y0= array( (0.,0.,vx0,vz0) )  # initial conditions

#Times

tau = 1.  # time step [s]
tfinal = 20.  # [s]
t = arange(0.,tfinal+tau,tau)  # array of times

#Calculate y at all times, and then unpack into physical variables
y = rk4(f_air,y0,t)
x = y[:,0]
z = y[:,1]
vx = y[:,2]
vz = y[:,3]
print "tau=",tau,"gives final values at t=",t[-1],"of x,z,vx,vz=\n",y[-1,:

#Analytic solution
xa = vx0*t
za = vz0*t - .5*g*t**2
vxa = vx0*ones(len(t))
vza = vz0 - g*t

sys.exit()  # uncomment to interrupt code without plotting

#Plotting (written for MacPython)
import Gnuplot,Gnuplot.funcutils
g = Gnuplot.Gnuplot(debug=0)
g(‘set terminal aqua’)
g.title(‘Trajectory’)
g.xlabel('x [m]
')
g.ylabel('z [m]
')
d = Gnuplot.Data(x,z,title='numeric',with='linespoints 3 3')
da = Gnuplot.Data(xa,za,title='analytic')
g.plot(d,da)
raw_input('Please press return to continue...
')

g.reset()
g.title('x as a function of time')
g.xlabel('t [s]')
g.ylabel('x [m]')
g('set data style lines')
d = Gnuplot.Data(t,x,title='numeric',with='linespoints 3 3')
da = Gnuplot.Data(t,xa,title='analytic')
g.plot(d,da)
raw_input('Please press return to continue...
')

g.reset()
g.title('z as a function of time')
g.xlabel('t [s]')
g.ylabel('z [m]')
g('set data style lines')
d = Gnuplot.Data(t,z,title='numeric',with='linespoints 3 3')
da = Gnuplot.Data(t,za,title='analytic')
g.plot(d,da)
raw_input('Please press return to continue...
')

g.reset()
g.title('vx as a function of time')
g.xlabel('t [s]')
g.ylabel('vx [m/s]')
g('set data style lines')
d = Gnuplot.Data(t,vx,title='numeric',with='linespoints 3 3')
da = Gnuplot.Data(t,vxa,title='analytic')
g.plot(d,da)
raw_input('Please press return to continue...
')

g.reset()
g.title('vz as a function of time')
g.xlabel('t [s]')
g.ylabel('vz [m/s]')
g('set data style lines')
d = Gnuplot.Data(t,vz,title='numeric',with='linespoints 3 3')
da = Gnuplot.Data(t,vza,title='analytic')
g.plot(d,da)
### Problem 2

Executing the code with the test piece

```python
# The following gives v=5 and so ax=-15, az=-10-20=-30
g = 10.
alpha = 1.
print "Problem 2 test: f_air = ",f_air( array((0.,0.,3.,4.)),5.)
```

produces the output

```
Problem 2 test: f_air = [  3.   4. -15. -30.]
```

which is clearly correct.

### Problem 4

Running with alpha=0 produces the following. The analytic and numerical results agree exactly, a very powerful code check.
**Problem 5**

Turning on drag and trying different step sizes $\tau$ produces the following final positions and velocities:

- For $\tau=10.0$ gives final values at $t=20.0$ of $x,z,v_x,v_z = [7.60699850e+14, -1.39498536e+16, -5.62403013e+28, 1.03134498e+30]$

- For $\tau=5.0$ gives final values at $t=20.0$ of $x,z,v_x,v_z = [3.75213065e+01, -5.51641698e+02, 2.12144192e-02, -1.96946592e+01]$

- For $\tau=1.0$ gives final values at $t=20.0$ of $x,z,v_x,v_z = [3.88401913e+01, -5.71867634e+02, 4.23716006e-02, -3.33325041e+01]$

- For $\tau=0.1$ gives final values at $t=20.01$ of $x,z,v_x,v_z = [3.88366216e+01, -5.72203294e+02, 4.22298889e-02, -3.33325197e+01]$

The larger two step sizes $\tau=10,5$ show indications that rk4 may be unstable for this problem if $\tau$ is too large. But $\tau=1$ already gives results very close to what smaller $\tau$ gives. For quick running and nice plots the remaining plots here are based on $\tau=1$ s.

Following are the final plots in the presence of drag.
Problem 6

Running for 60 seconds produces a final velocity $[3.31271192 \times 10^{-7} \ -3.33333333 \times 10^1] \text{ m/s}$. This is nearly exactly the correct terminal velocity of $v_x=0$ and $v_z=-120 \text{ km/hr}$ (it is correct to 7 decimal places). This is a very powerful code check.