Falling body movement

First, the computer simulations: Freefall problem

Q: Want to know where the shells hit

Physical quantities: the trajectory of x with time (t), y (t)

The laws of physics (mathematical models): $\mathbf{f} = m\mathbf{a}$, in this **f** purely from gravity (**boldface** representative vector)

Particles in the vicinity of the gravity acting on the surface is g ($g = GmM / R$), $g = 9.8$ N / m

$\mathbf{f} = m\mathbf{a} = m(a_x \mathbf{e}_x + a_y \mathbf{e}_y)$, where $a_x = dv_x / dt$, $a_y = dv_y / dt$

Only gravity, $\mathbf{f} = f_y \mathbf{e}_y = gm\mathbf{e}_y$, that is $f_y = gm$

Algorithm:

Use Euler algorithm :

Position $x_{N+1} = x_N + v_{x,N} t$; $y_{N+1} = y_N + v_{y,N} t$

Velocity $v_{x,N+1} = v_{x,N} + a_{x,N} t$; $v_{y,N+1} = v_{y,N} + a_{y,N} t$

Advanced added: stability and accuracy of the calculation, there are better options

Euler-Richarson algorithm, first find

$a_{x,N} = f_x(r_N , v_N , t) / m$

$v_{x,mid} = v_{x,N} + (1/2) a_{x,N} t$

$x_{mid} = x_N + (1/2) v_{x,N} t$

$a_{x,mid} = f_x(x_{mid} , v_{x,mid} , t + (1/2) t) / m$

Then do

$v_{x,N+1} = v_{x,N} + a_{x,mid} t$

$x_{N+1} = x_N + v_{x,mid} t$

Note: Why Euler-Richarson than Euler's algorithm, and can refer to derive Gould textbook: p1 , p2

Program flow:

(0), declaring variables

(1) ask the quality, the initial velocity (including the initial rate and angle)

(2) establish the position of x and y, velocity v_x and v_y of the initial value

(3) with a small algorithm seeking the next period Δt after the position and speed

(4) Draw the particle's position and checks whether the height has reached ground, below the ground then back to (3)

(5) Any further shot once asked, to the back (1)

(6) End

You can not wait for it? First steal play with the teacher has to do executable file cannon.x

You usually do not come? First peek at the teacher had written programs range cannon.f cannon.f.txt

(Do not look at the last resort)

Fixed brains, moving hands, and then transform themselves (or home made):

1. Draw a cannon barrel angle to show it

2. Fill the number of gunpowder packed performance shells kinetic energy is converted into initial rate

3. Objects bombing fragments pop (with the random number generator ran3.f , using the method)

4. try to write a fireworks program will multistage flowering discoloration

Analysis and discussion:

Want to predict where the shells hit, one that is mainly based on the laws of physics?

What kind of precedent (initial) conditions determine the trajectory of the projectile (and placement)?

The same number of gunpowder filled bag, how to play the shells only fly the farthest?

Second, the computer simulations: by the air resistance of a falling body

Question: In the case of air resistance, the projectile trajectory becomes how?

Physical quantities: as is the change over time in the trajectory x (t), y (t)

Physical laws (mathematical model): **f** = m **a** , the force **f** includes both gravity and air resistance

Air resistance is very complex (and therefore the resistance of high-speed trains or planes designed to rely on wind tunnel experiments and computer simulations carried out), basically it is related with the rate, the greater the resistance, the higher the rate, stationary objects is no air resistance. It is a visible amount of rate-related, common simplified formula, there is a square with a rate directly proportional, but also with the two parties was directly proportional to, the direction is the speed of the reverse (negative number)

$F_d = - k_1 v$

$F_d '= - k_2 v^2$

Because the two components x and y, the angle between the horizontal velocity is assumed to θ, i.e., theta = Tan^{-1} (Vy / vx) [Fortran program written `Theta = ATAN (Vy / vx)`],

$F_{d, x} = F_d \cos$

$F_{d, y} = F_d \sin$

(Alternatively, the direct use of $v_x / \sqrt{(v_x^2 + v_y^2)}$ and $v_y / \sqrt{(v_x^2 + v_y^2)}$ than first seek theta = ATAN (v_x / v_y) before taking the angle to seek cos (θ) and sin (θ) better)

Therefore, the relevant portion of the power equation,

$f_x = F_d \cos$

$f_y = - mg + F_d \sin$

Other parts, the same as the previous issue

New issues: how to obtain room air resistance coefficient k?

Use terminal velocity, which plummeted during the balance of gravity and drag up, so that is no longer accelerating force is zero, but the speed was constant during exercise.

Force = $0 = - mg + F_d = - mg + k_1 v_y^2$

I.e., $k_1 = v_y^2 / (mg)$, this terminal speed can be obtained by experiment amount

A chalk-sized pebbles its terminal velocity of about thirty meters per second. In addition, a heavy 0.254 g, 2.54 cm radius Poly Dragon Ball following the measured data (cited Physics Teacher **24** , 153 (1986))

Time (sec)	Position (m)

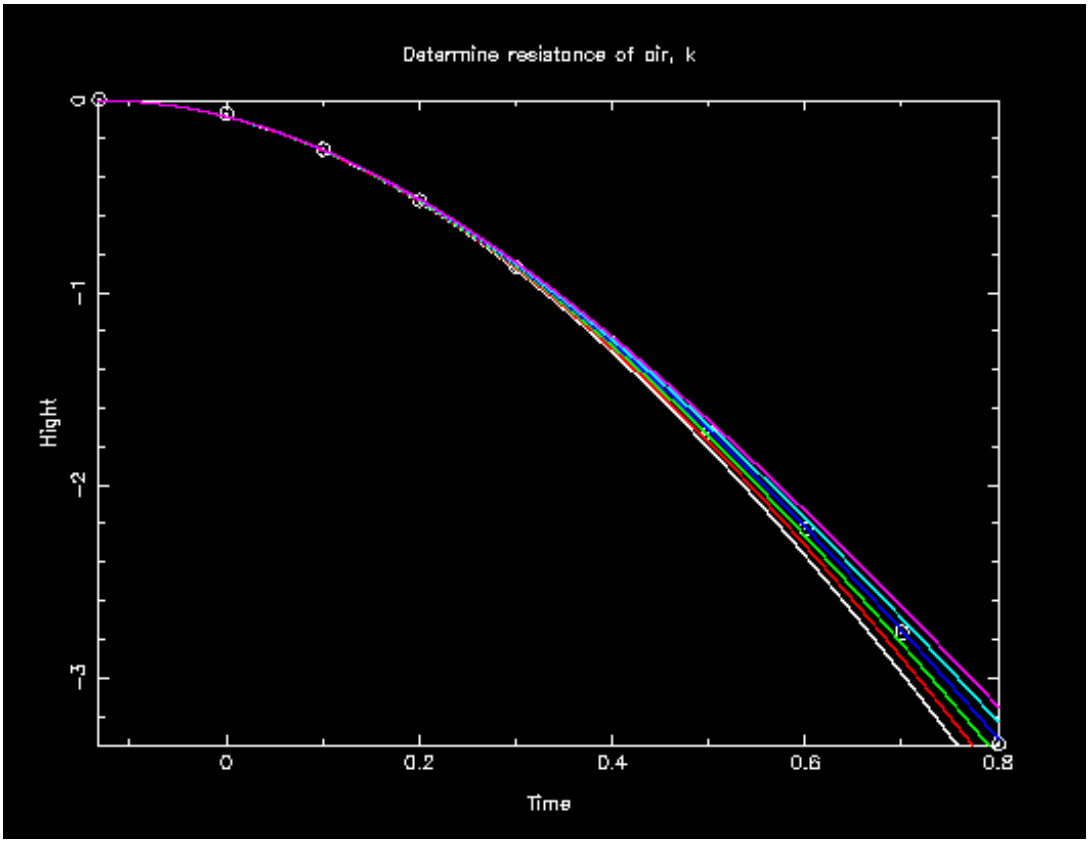
-0.132	0.0
0.0	0.075
0.1	0.260
0.2	0.525
0.3	0.870
0.4	1.27
0.5	1.73
0.6	2.23
0.7	2.77
0.8	3.35

Think about it, how we want to air resistance coefficient k obtained from the above data?

The sample program teacher wrote: styrofoam_k.f.txt styrofoam_k.f styrofoam_k.x

Above tabular data profile: styrofoam.txt

(as they want, no longer reference)



Algorithm:

Also using Euler or Euler-Richardson algorithm

Programming:

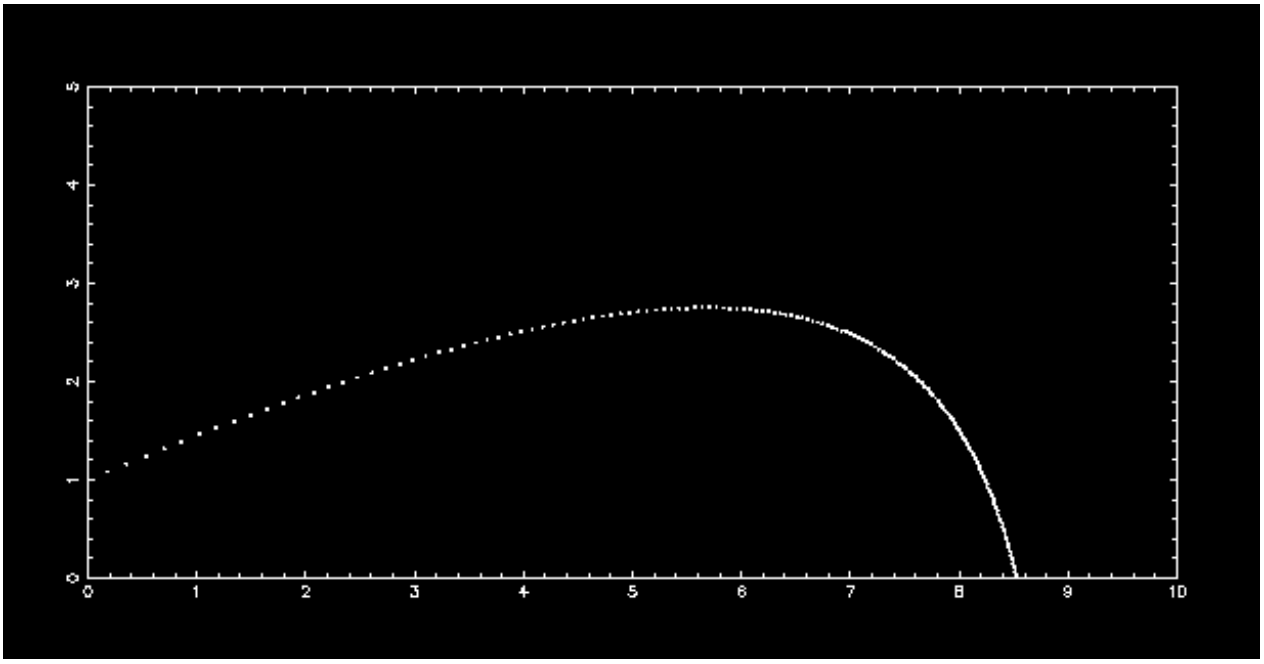
The same as the previous issue

Reference sample program

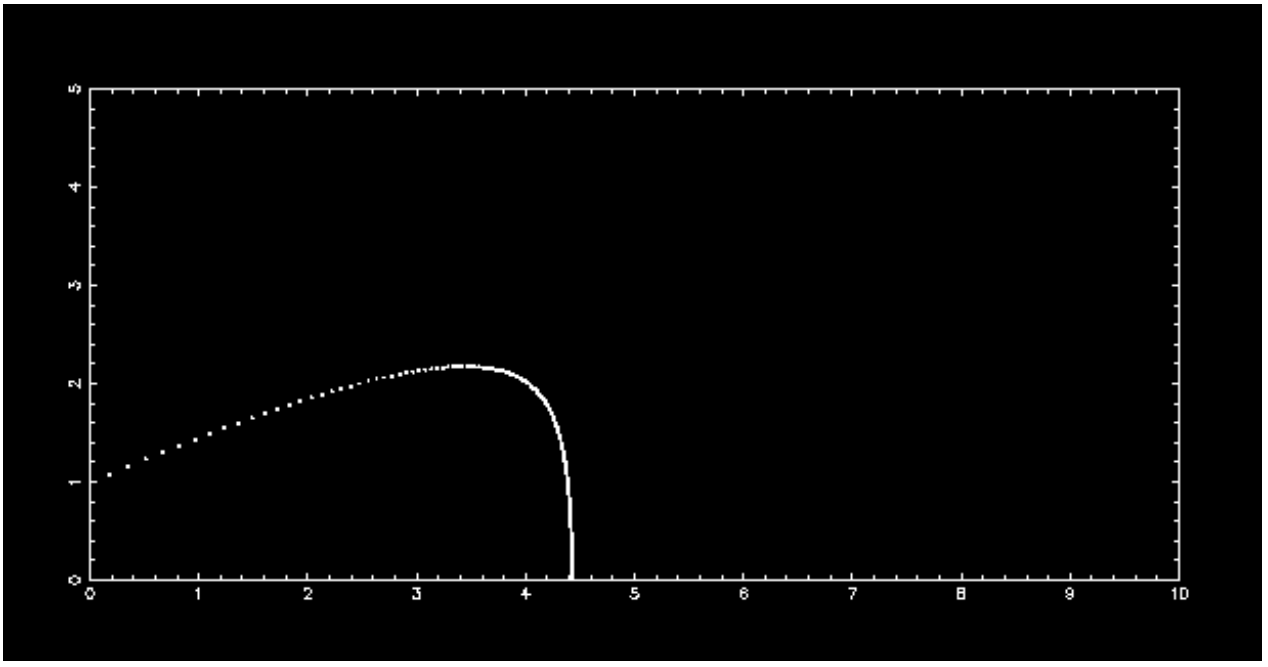
cannon_drift.f cannon_drift.f.txt cannon_drift.x

Affect the quality of the air resistance coefficient of the projectile trajectory

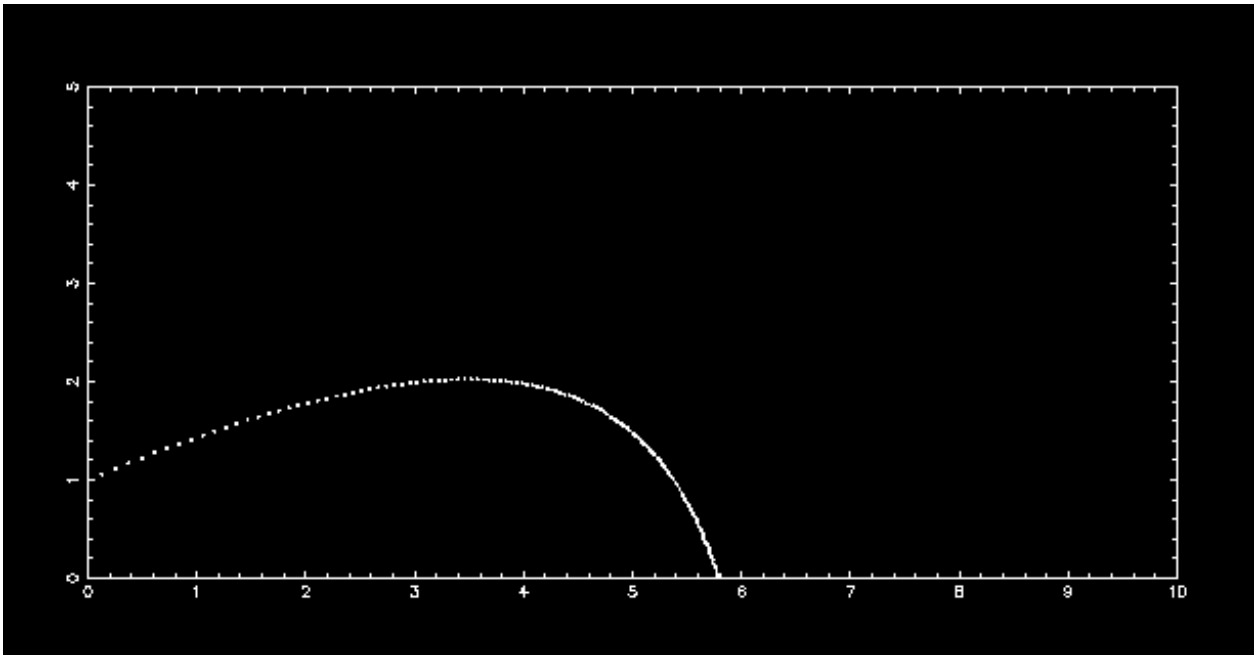
mass = 1.0, k = 2.0



mass = 0.5, k = 1.0



mass = 2.0, k = 2.0



Analysis and discussion:

In the same force up a pebble thrown in there is no room air resistance and air resistance both cases comparison, what kind would stay in the air longer?

That kind of trajectory like badminton? That kind of like the shot put?

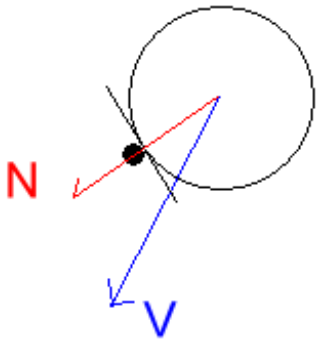
In the case of air resistance, the angle of the projectile to how to set it will be the farthest distance?

Challenge:

Try to make your own Angry Birds game (technical details are welcome to discuss with Professor).

Job: Pinball program

Marbles and smooth nails interaction mode



$\mathbf{N} = N_x \mathbf{e}_x + N_y \mathbf{e}_y$

$|\mathbf{N}| = 1$

$\mathbf{V} \text{ dot } \mathbf{N} = V_x N *_x + V_y * N_y$

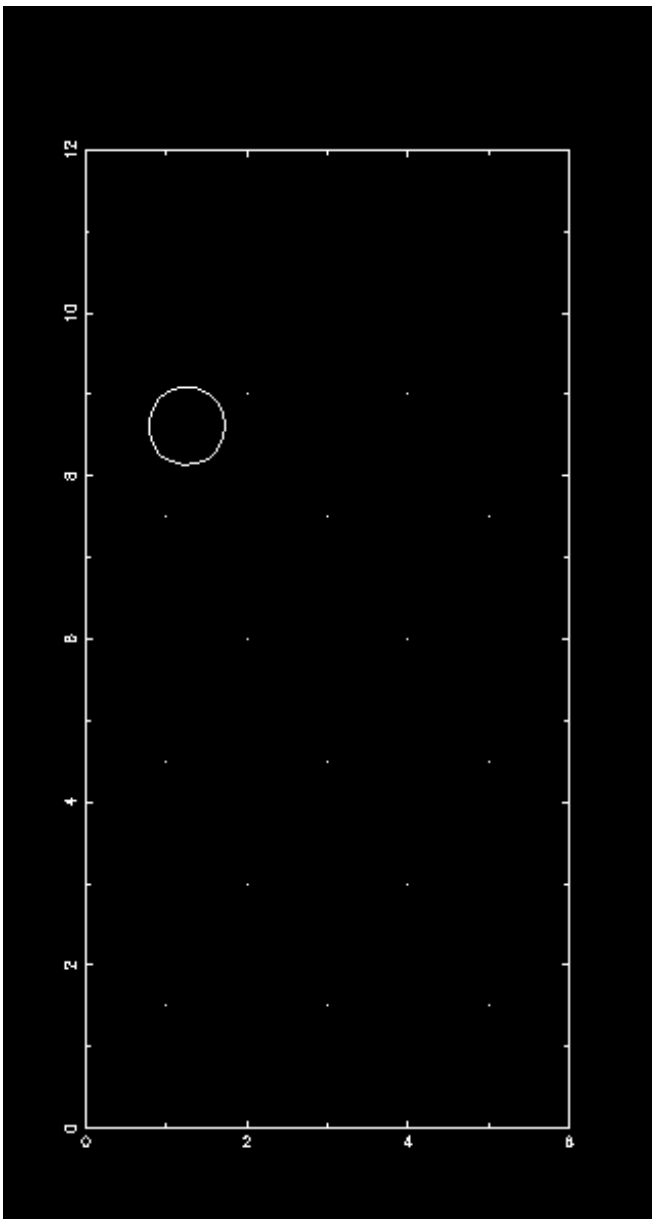
After the collision V parallel to the N component due to the rebound in the opposite direction, with the N component perpendicular to continue to maintain, so after collision

$\mathbf{V}_\perp = (-1) * (\mathbf{V} \text{ dot } \mathbf{N})\, \mathbf{N}$

$\mathbf{V}_{//} = \mathbf{V} - (\mathbf{V} \text{ dot } \mathbf{N})\, \mathbf{N} = \mathbf{V} + \mathbf{V}_\perp$

The new rate was $\mathbf{V}_{//} + \mathbf{V}_\perp$

Example program teachers provided `pinball.f pinball.f.txt pinball.x` , input sample

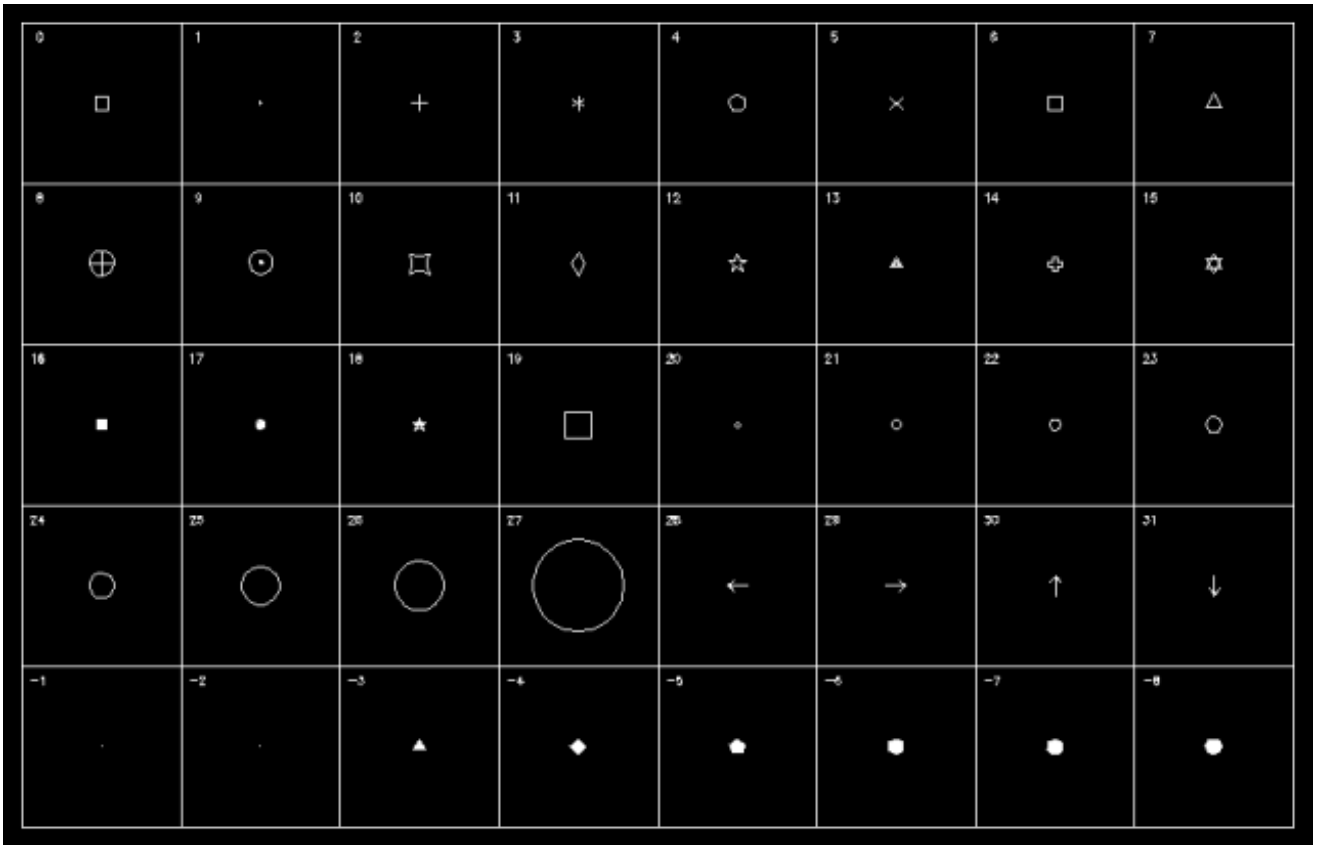


Hands-on and discussion of the transformation:

The program will be changed once you release two, three marbles.

If there is friction on the marbles nails, how to model the interaction with the marbles?

PGPLOT: PGPT (N, x, y, SYMBOL)



Random number generator usage

In ran3.f embedded ran3 (iseed) function, for example:

```
iSeed = 7
x = RAN3 (iSeed)
y = RAN3 (iSeed)
```

Ready to be placed in any one of the integer value of an integer variable iseed, depending on the method described above each use ran3 (), then the x and y values will get between a 0.0 and 1.0, although the value is not the same every time, But the chances of these values appear between 0-1 is equal. In addition, directly to the integer fill ran3 () but does not support, such as `xx = ran3 (7), will lead to "section error (Sgementation Fault)"`.

Try to write a throw of the dice program. Example Fan: `dice.f` , remember to compile with ran3.f, as `gfortran diec.f ran3.f <Enter>` .

In addition to support built-in subroutine gfortran `random_number ( real number of variables )` , you do not have another function with external random number, and can refer to another programming paradigm dice `dice_gfortran.f` . Reset random number seed subroutine is `random_seed` details see: http://www.nsc.liu.se/~boein/f77to90/a5.html#section21c

Further reading:

Gould and Tobochnik, An Introduction to Computer Simulation Methods - Applications to Physical Systems, Addison Wesley (1996) Chapter 2, Chapter 3