巴西果效應

B07901021 潘世軒
B07901036 陳俊廷
B07901016 朱哲廣

你有沒有想過，為什麼
每次打開綜合堅果，最
上面都是大顆的堅果？

- 巴西果效應：在經過晃動後，顆粒大的物體會漸漸向上浮出。
  - 打開綜合堅果時，由於在運送過程中已經過輸送帶和貨車等運輸工具，自然會受到晃動，因此消費者打開時，上方多半是個頭比較大的堅果。

目前對造成巴西果效應的機制的解釋主要是:

幾何填空：

顆粒之間會產生空隙，其中小顆粒較容易從空隙中向下，大顆粒則較不容易向下，久而久之，大顆粒自然會越來越往上移動。

CODE實作：一開始的隨機堆積

- Random球的位置(大球在偏下方生成)
- 落下(在main)

```
for i in range(N):
    if i >= 1 and i <= n: # random big ball position (red)
        r = K * size
        r_a[i] = [r, 0, 0]
        check = 0
        while (check == 0):
            p_a[i] = [2 * L * random() - L, -(L - 6 * size), 2 * L * random() - L]
            for j in range(i):
                p = p_a[j] - p_a[i]
                p_mag = np.sqrt(np.sum(np.square(p)))
                if (p_mag <= r_a[i][0] + r_a[j][0]): break
                if (abs(p_a[i][0]) > L - r_a[i][0]): break
                if (p_a[i][1] < -(L - r_a[i][0])): break
                if (abs(p_a[i][2]) > L - r_a[i][0]): break
                if (j == i - 1): check = 1
        m_a[i] = M1
        nut_new = sphere(pos = vector(p_a[i, 0], p_a[i, 1], p_a[i, 2]), radius = r,m = M1,color = color.red)
    else:
        r = size
        r_a[i] = [r, 0, 0]
        check = 0
        if(i != 0):
            while (check == 0): # random small ball position (orange)
                p_a[i] = [2 * L * random() - L, i * size, 2 * L * random() - L]
                for j in range(i):
                    p = p_a[j] - p_a[i]
                    p_mag = np.sqrt(np.sum(np.square(p)))
                    if (p_mag <= r_a[i][0] + r_a[j][0]): break
                    if (abs(p_a[i][0]) > L - r_a[i][0]): break
                    if (p_a[i][1] < -( L - r_a[i][0])): break
                    if (abs(p_a[i][2]) > L - r_a[i][0]): break
                    if (j == i - 1):
                        check = 1
        m_a[i] = M2
        nut_new = sphere(pos = vector(p_a[i, 0], p_a[i, 1], p_a[i, 2]), radius = r, m = M2,color = color.orange)
    v_a[i] = [0, 0, 0] # particle initially same speed but random direction
    g_a[i] = [0, -g, 0]
    nuts.append(nut_new)
```

```python
# particle position array and particle velocity array, N particles and 3 for x, y, z
p_a, v_a = np.zeros((N, 3)), np.zeros((N, 3))
m_a = np.zeros((N, 3)) # use numpy to set the m of each balls
g_a = np.zeros((N, 3))
r_a = np.zeros((N, 3)) # radius of each balls
```

```python
p_a += v_a * dt
for i in range(N):
    nuts[i].pos = vector(p_a[i][0], p_a[i][1], p_a[i][2])
```

```python
average_pos = 0
T += 1
for i in range(1, n+1):# print the average position of all big balls
    average_pos += p_a[i][1]
pos.plot(pos = (t, average_pos/3))
if stage == 1 and T % 20 == 0: # every 20 dt shake one time
    shake()
```

```python
r_array = p_a - p_a[:, np.newaxis] # array for vector from one atom to another atom for all pairs of atoms
collision_array = r_a + r_a[:, np.newaxis]
rmag = np.sqrt(np.sum(np.square(r_array), -1)) # distance array between atoms for all pairs of atoms
collision_mag = np.sqrt(np.sum(np.square(collision_array), -1))
# if smaller than 2*size meaning these two atoms might hit each other
hit = np.less_equal(rmag,collision_mag) - np.identity(N)
hitlist = np.sort(np.nonzero(hit.flat)[0]).tolist() # change hit to a list
for ij in hitlist: # i,j encoded as i*Natoms+j
    i, j = divmod(ij, N) # atom pair, i-th and j-th atoms, hit each other
    hitlist.remove(j * N + i) # remove j,i pair from list to avoid handling the collision twice
    # only handling collision if two atoms are approaching each other
    if sum((p_a[i] - p_a[j]) * (v_a[i] - v_a[j])) < 0 :
        v_a[i], v_a[j] = vcollision(p_a[i], p_a[j], v_a[i], v_a[j],m_a[i], m_a[j])# handle collision
```

CODE實作：位置更新

- 用陣列來儲存位置，速度，加速度等物理量(直接對陣列進行加減速度很快)
- 欲在圖上呈現只要更新pos就好了
- 生成球之間會產生碰撞的距離陣列，和球之間的距離陣列，創造一個hitlist

```python
for i in range(N):# balls collides with walls(using strings)
    if abs(p_a[i][0]) >= L - nuts[i].radius and p_a[i][0] * v_a[i][0] > 0 :
        v_a[i][0] = -v_a[i][0] + abs(L - nuts[i].radius - abs(p_a[i][0])) * k * (-p_a[i][0])
    if abs(p_a[i][1]) >= L - nuts[i].radius and p_a[i][1] * v_a[i][1] > 0 :
        v_a[i][1] = -v_a[i][1] + abs(L - nuts[i].radius - abs(p_a[i][0])) * k * (-p_a[i][1])
    if abs(p_a[i][2]) >= L - nuts[i].radius and p_a[i][2] * v_a[i][2] > 0 :
        v_a[i][2] = -v_a[i][2] + abs(L - nuts[i].radius - abs(p_a[i][0])) * k * (-p_a[i][2])
for i in range(1, n+1):
    if p_a[i][0] <= -L + 0.99 * nuts[i].radius:
        p_a[i][0] = -L + nuts[i].radius
    if p_a[i][0] >= L - 0.99 * nuts[i].radius:
        p_a[i][0] = L - nuts[i].radius

    if p_a[i][1] <= -L + 0.99 * nuts[i].radius:
        p_a[i][1] = -L + nuts[i].radius
    if p_a[i][1] >= L - 0.99 * nuts[i].radius:
        p_a[i][1] = L - nuts[i].radius

    if p_a[i][2] <= -L + 0.99 * nuts[i].radius:
        p_a[i][2] = -L + nuts[i].radius
    if p_a[i][2] >= L - 0.99 * nuts[i].radius:
        p_a[i][2] = L - nuts[i].radius
```

CODE實作：球和牆的碰撞

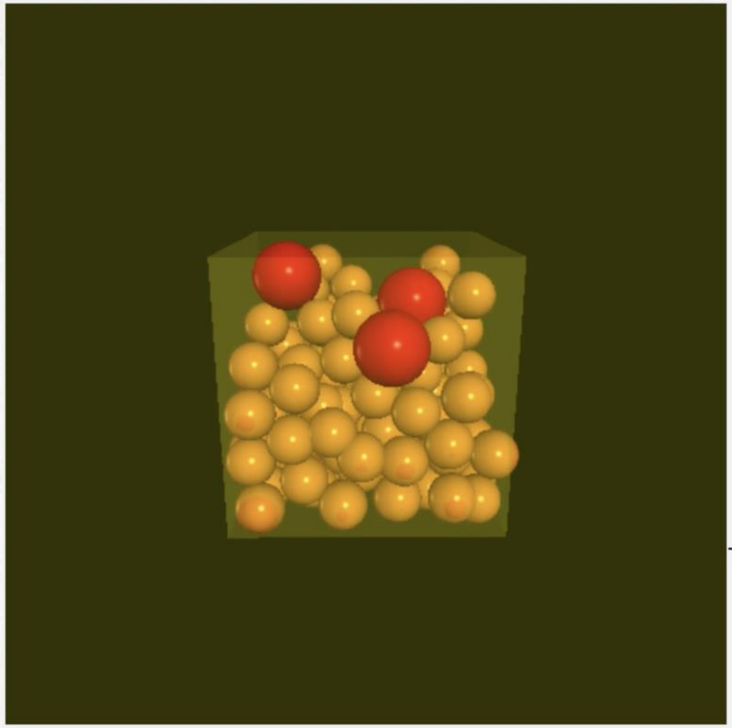
- 彈性碰撞+彈簧
- 註：這裡我們再加入了彈簧的效應(若是缺乏彈簧的效應，等於說容器底部沒有給予球一個正向力，球也會被上面的球擠壓而陷下去)

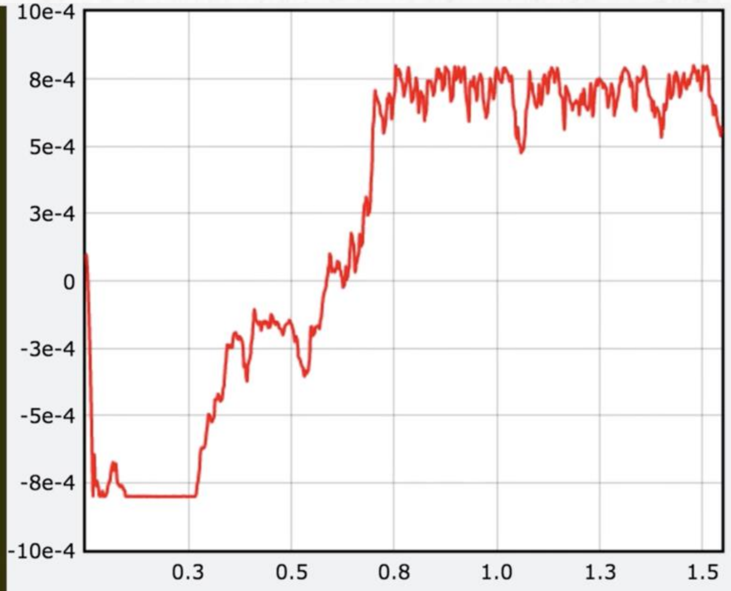
```python
def shake(): #shake the container
    for i in range(N):
        v_a[i] += [np.random.uniform(-1, 1) / 10, np.random.uniform(-1, 1) / 10, np.random.uniform(-1, 1) / 10]
def keyinput(evt):#press n -> start shaking or stop shaking
    plus = {'n':1}
    s = evt.key
    global stage
    if s in plus:
        if stage == 0:
            stage = 1
        elif stage == 1:
            stage = 0
```

CODE實作：搖動

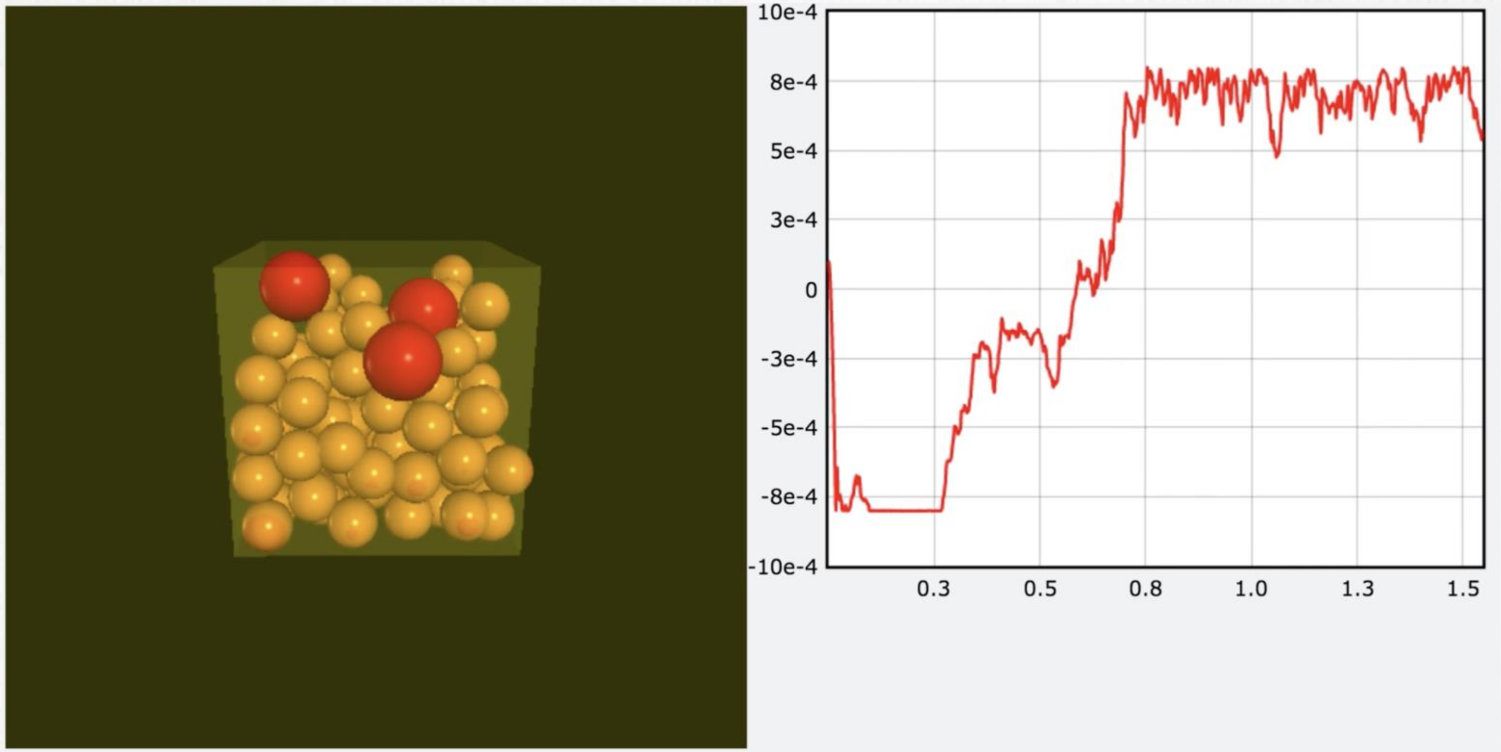
- 給予球一個速度，模擬對盒子進行搖動的狀況
- 其實就是觀察者在盒子中心的搖動

結果：所有大球的平均位置



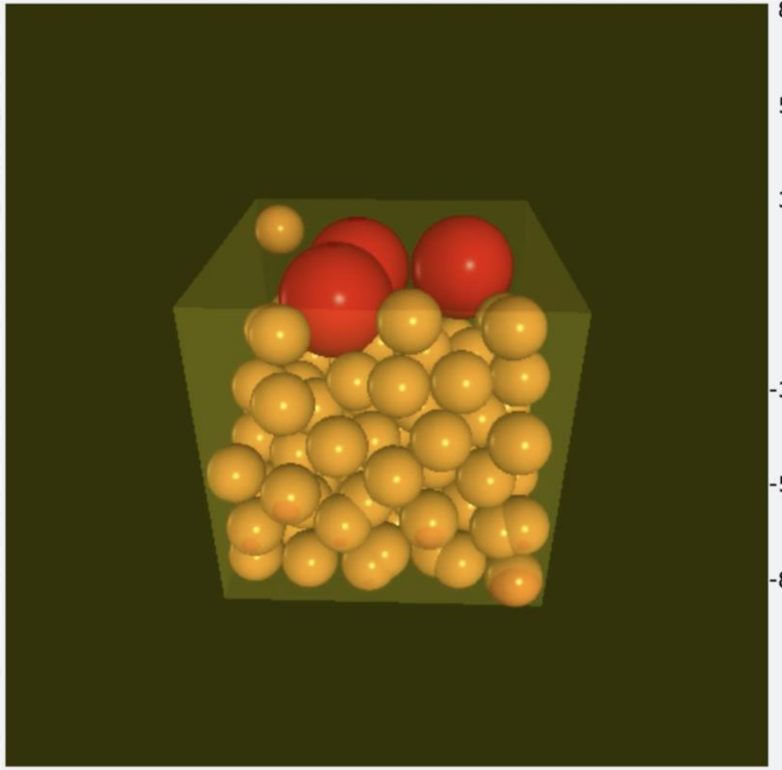


變因：改變大球的大小(和小球的半徑比 = 1.5)



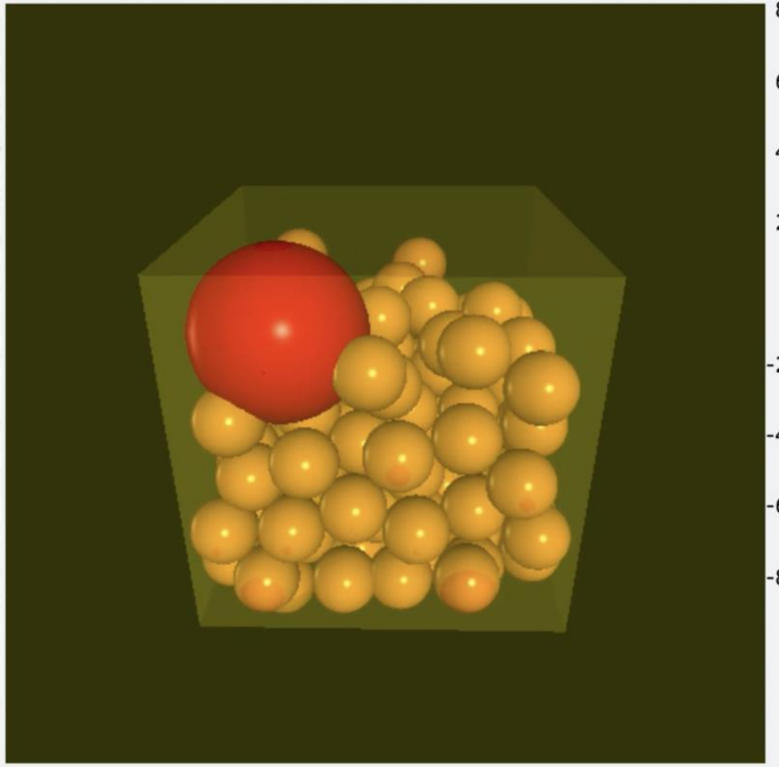
https://youtu.be/h37QpTgJMlY

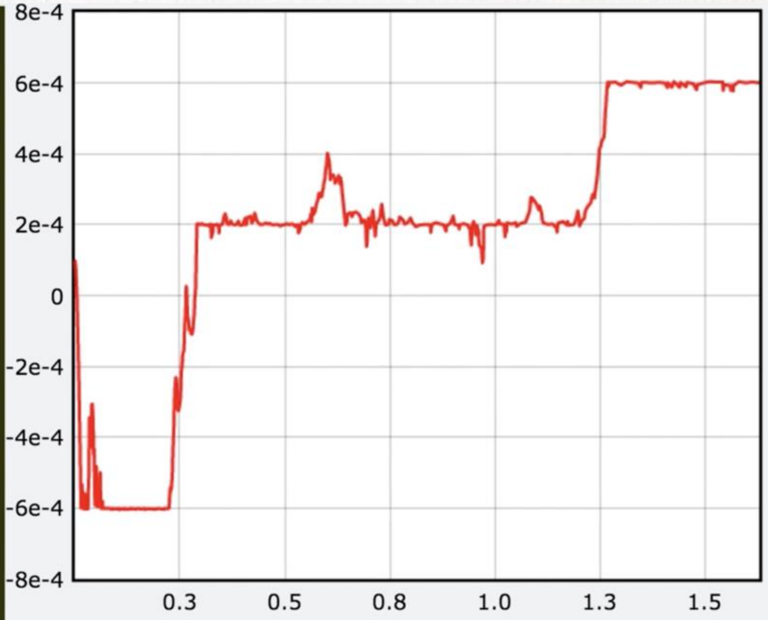
變因：改變大球的大小(和小球的半徑比 = 2)



https://youtu.be/OE1MtOUgtEA

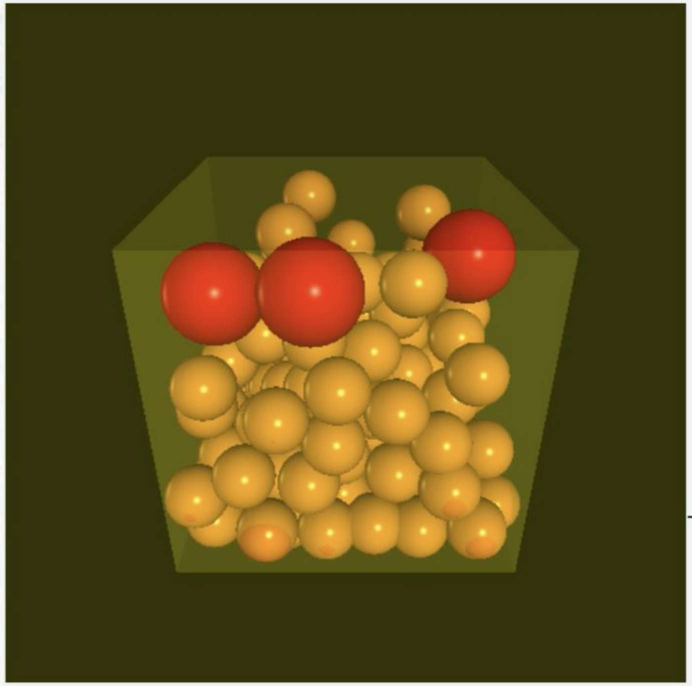
變因：改變大球的大小(和小球的半徑比 = 2.5)

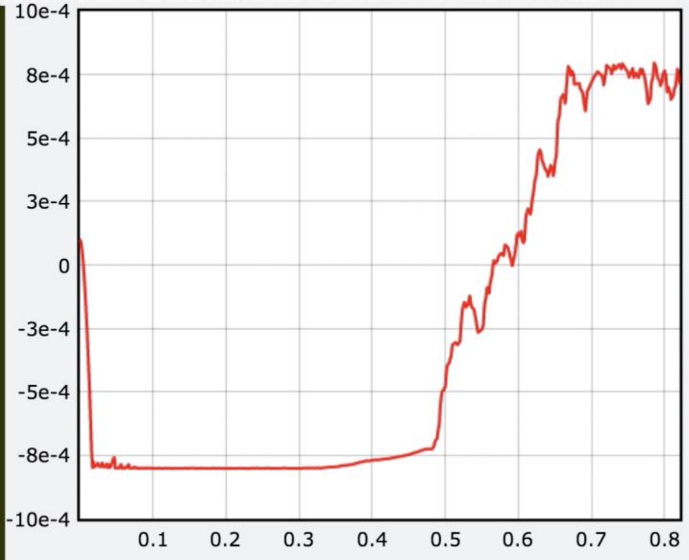




https://youtu.be/cCU3rd_-A-o

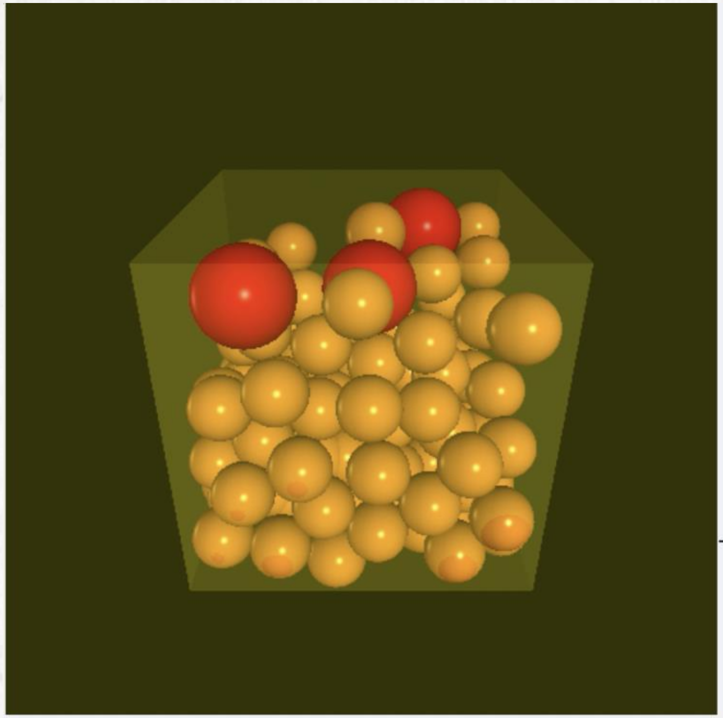
變因：改變大球的密度(和小球之間的密度比 = 1.5)

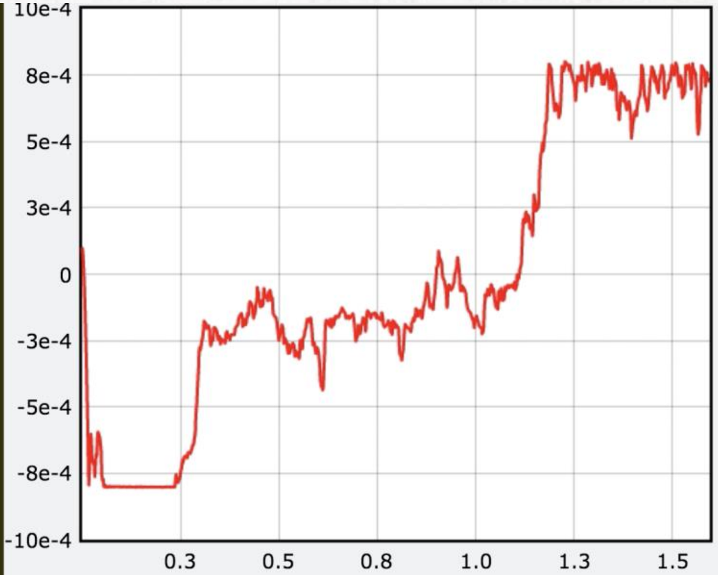




https://youtu.be/v524P_gkQYU

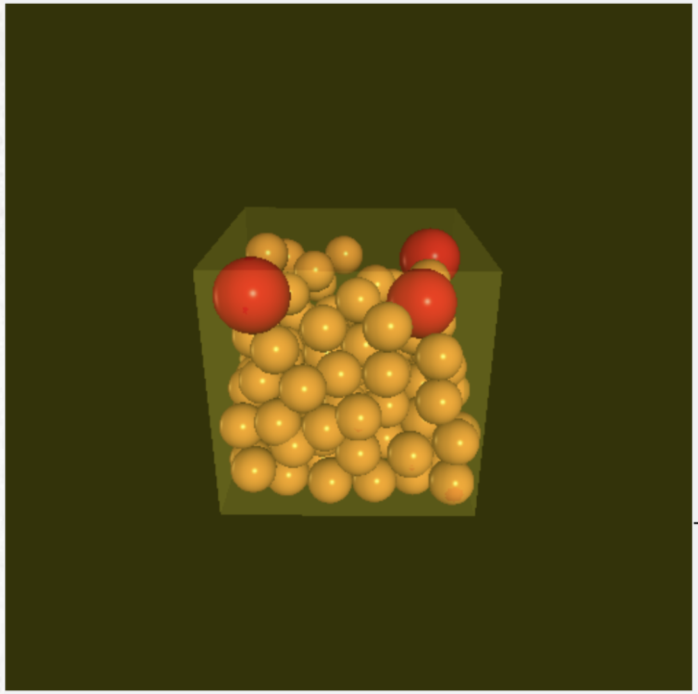
變因：改變大球的密度(和小球之間的密度比 = 2)

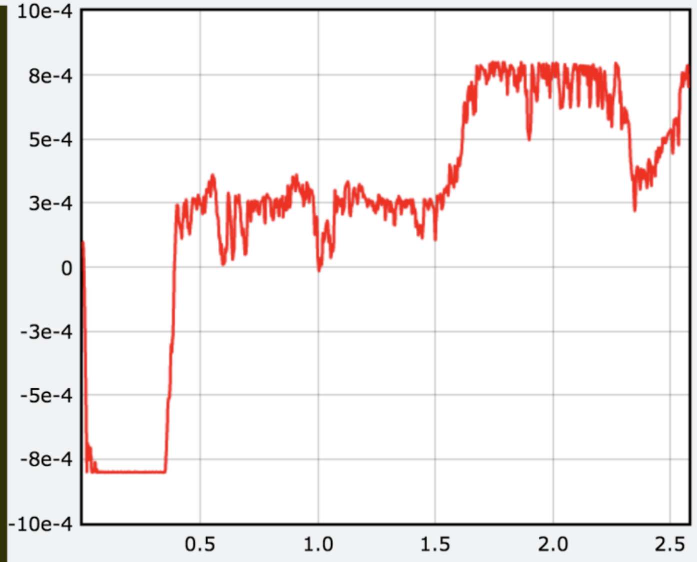




https://youtu.be/T-31rgfCJQk

變因：改變大球的密度(和小球之間的密度比)





https://youtu.be/Td0hboQdQb4

實際上還要考慮：

- **摩擦力**

很難去估計摩擦力和球作用的時間，因此難以將摩擦力完美地考慮進程式模擬之中。

- **氣壓**

芝加哥大學研究發現，氣壓會影響大球上升的平均速度，推測是因為氣體的黏滯力的影響。但在程式模擬中，若是增加氣壓的討論，還會牽涉到流體力學之類的分析，變得非常複雜。

- **碰撞**

碰撞不是完全彈性碰撞，也不是完全非彈性碰撞。

KNOW MORE ?

- 巴西果效應和重力有關，若是沒有重力，不會產生巴西果效應，巴西果效應與顆粒 "流體" 在振動條件下的對流有關，而重力是對流的必要條件。
- 在某些特定的振動條件下，會產生反巴西果效應。
- 當大小顆粒的密度比超過一臨界值，也會產生反巴西果效應。
- 註：下方兩點這些都會牽涉到空氣和顆粒的作用(流體力學)，較難於程式中模擬出來

參考資料

- https://www.zhihu.com/question/31688918/answer/53230327
- http://wulixb.iphy.ac.cn/fileup/PDF/2012-13-134501.pdf
- https://www.youtube.com/watch?v=kRkwGyBpUm8