Computational Physics Case Study 3: Star Formation

Augustin Ionescu

December 15, 2017

Abstract

The aim of this study is to look into the formation of stars in an interstellar gas cloud with the 1-dimensional spherically symmetric Bondi model. By using a Monte Carlo simulation done in Python, we study Bondi accretion and generate an Initial Mass Function that is then compared to the observed IMF from the Salpeter law.

1 Introduction

Stars form most likely from interstellar gas clouds that are initially in relative equilibrium, with some forces such as the thermodynamic pressure force pushing them apart, while others providing resistance to compression (e.g. magnetic forces). But if the density ρ of the interstellar gas becomes high enough locally, the pull of its own gravity becomes the dominant force and the region may contract to form a star. The different forces involved are generally well balanced on all spatial scales, therefore star formation processes happen at both the galactic and interstellar scale, ranging from large regions inside the galactic arms all the way down to dense clumps and clump cores within relatively small gas clouds that usually generate clusters. A typical cloud of molecular gas and dust ($\mu m - mm$ sized grains of, e.g., silicates, carbon and ices) of this size measures a few light years across and usually emits infrared light due to its small temperature (around 20 K).

In the regions where star formation is ongoing, the gas accumulates in different stages. When a core has formed, it often keeps accreting gas through an accretion disc. The latter does not only channel gas down to stars, but may also form self-gravitating condensations (e.g. planets), which can keep orbiting the stars long after the accretion disc has dispersed. In our own solar system, the coplanar orbits of the planets, as well as the presence of a tenuous dust disc are the direct results of the accretion disc our Sun had about 4.5 billion years ago.

2 Bondi accretion

2.1 Background

This project presents a study done on star formation with the 1-dimensional spherically symmetric Bondi model. Because it is spherically symmetric, it does not allow for accretion

discs or planet formation. We can, however address the growth and mass distribution of stars. The Bondi model is a solution of the hydrodynamics equations, which describe gas flows in the presence of a (proto-) star's gravity and pressure.

The theory of hydrodynamics comprises three equation corresponding to the conservation of mass, momentum and energy. The continuity equation describes the conservation of mass and reads in 1D spherical coordinates:

$$\frac{\partial}{\partial t}\rho + \frac{1}{r^2}\frac{\partial}{\partial t}(r^2\rho v) = 0 \tag{1}$$

Here, t denotes the time, r the spherical radius, and v the velocity (positive outwards). Next, with $\Phi = GM/r$ the potential for a point mass M, the so-called Euler equation, which describes the conservation of momentum, can be written, again for spherical symmetry, as:

$$\frac{\partial}{\partial t}\rho v + \frac{1}{r^2}\frac{\partial}{\partial t}(r^2\rho v) = -\rho\frac{GM}{r^2} - \frac{\partial}{\partial r}p \tag{2}$$

The energy equation, which is in general also a partial differential equation, can be expressed as:

$$\frac{p}{p_0} = \left(\frac{\rho}{\rho_0}\right)^{\gamma} \tag{3}$$

with the adiabatic index γ and some reference values for pressure p_0 and density ρ_0. For most astrophysical situations of interest, 1 ¡ γ ¡ 5/3. The exact value of γ encodes the importance of radiative processes with $\gamma = 5/3$ meaning no energy loss by radiation, and $\gamma = 1$ for a plasma that is kept isothermal by external conditions.

The equations can be further simplified by assuming stationarity, i.e. all time derivatives vanish. This is physically reasonable, because a system will keep evolving until it finds a stationary solution.

We can now integrate Eq. 1 to find:

$$r^2\rho v = \text{const} \equiv \frac{\dot{M}}{4\pi} \tag{4}$$

where we have introduced the constant accretion rate $\dot{M}$. [1] Next, we introduce the sound speed $c_s^2 = \partial p/\partial\rho$, which is a function of r, and replace the pressure gradient in Eq. 2 via

$$-\frac{\partial}{\partial r}p = -\frac{\partial p}{\partial\rho}\frac{\partial}{\partial r}\rho = -c_s^2\frac{\partial}{\partial r}\rho \tag{5}$$

We can then take the derivative of Eq. 4:

$$0 = \frac{\partial}{\partial r}(r^2\rho v) = r^2 v\frac{\partial\rho}{\partial r} + r^2\rho\frac{\partial v}{\partial r} + 2r\rho v \tag{6}$$

so that

$$-\frac{\partial}{\partial r}p = -c_s^2\left(-\frac{\rho}{v}\frac{\partial v}{\partial r} - \frac{2\rho}{r}\right) \tag{7}$$

Using this and stationarity in Eq. 2, we get:

$$\frac{1}{r^2}\frac{\partial}{\partial r}(r^2\rho v^2) = -\rho\frac{GM}{r^2} + c_s^2\left(\frac{\rho}{v}\frac{\partial v}{\partial r} + \frac{2\rho}{r}\right) \tag{8}$$

The LHS of Eq. 8 can be simplified with the help of Eq. 6:

$$\tfrac{1}{r^2}\frac{\partial}{\partial r}(r^2\rho v^2) = \tfrac{2\rho v^2}{r} + v^2\frac{\partial \rho}{\partial r} + \rho\frac{\partial}{\partial r}(v^2) = \tfrac{2\rho v^2}{r} + v^2\left(-\tfrac{\rho}{v}\frac{\partial v}{\partial r}\right) + \rho\frac{\partial}{\partial r}(v^2) = \tfrac{1}{2}\rho\frac{\partial}{\partial r}(v^2)$$

Applying this to Eq. 8 and rearranging, we get:

$$\frac{1}{2}\left(1 - \frac{c_s^2}{v^2}\right)\frac{\partial}{\partial r}(v^2) = -\frac{GM}{r^2}\left(1 - \frac{2c_s^2 r}{GM}\right) \tag{9}$$

and by introducing the Mach number $\mathcal{M} = v/c_s$ and the sonic radius $r_s = GM/2c_s^2$ we finally arrive at:

$$\frac{1}{2}\left(1 - \frac{1}{\mathcal{M}^2}\right)\frac{\partial}{\partial r}(v^2) = -\frac{GM}{r^2}\left(1 - \frac{r}{r_s}\right) \tag{10}$$

Even without integration, it becomes apparent that the equation has a sonic point where both sides vanish. For this to happen on the left hand side, the Mach number has to be unity, hence the name, and the radius has to be the sonic radius. The exact value of the sonic radius follows from the full solution, since c_s is a function of radius itself. For an accretion problem where the flow accelerates inwards, no signal travelling at sound speed can reach the outside world once that part of the flow has passed the sonic radius.

2.2 Analysis

A solution to Eq. 10 can be found analytically in the case of the strongly supersonic region far inside the sonic radius. Indeed, in this situation, $r << r_s$ so $r/r_s \to 0$ and $v >> c_s$ so $\mathcal{M} \to \infty$. Then, Eq. 10 can be re-written as:

$$\frac{1}{2}\frac{\partial}{\partial r}(v^2) \approx -\frac{GM}{r^2} \tag{11}$$

and by integration, the solution becomes approximately:

$$v^2 = \frac{2GM}{r} \tag{12}$$

By plugging this into Eq. 4 and re-writing for ρ we get:

$$\rho = \frac{\dot{M}}{4\pi\sqrt{2GMr^3}} \tag{13}$$

The infall at $r << r_s$ is supersonic, and the infalling gas is in free fall, but the velocity will not increase indefinitely. We can immediately observe that in this situation the solution presented in Eq. 12 is actually the escape (parabolic) velocity. This means that eventually the gas will slam into the surface of the compact accreting core, and its velocity just before it hits the compact object will be v_{esc}. Also, from Eq. 13 we can see that the accretion rate is proportional to the gas density.

3 Accretion rate

3.1 Background

According to Eq. 4, the accretion rate $\dot{M}$ is independent of time, so it can be evaluated at any radius with the same result. It is interesting to write the accretion rate as a function of the thermodynamic properties at large radius. However, as the accretion velocity gets small in that case, in practice it will be dominated by other gas motions, unrelated to the accretion. We can therefore calculate the accretion rate at the sonic radius, where $-v = c_s(r_s)$, and relate $c_s(r_s)$ and ρ to the properties far away from the central object. The accretion rate at r_s can be expressed as:

$$\dot{M} = 4\pi r_s^2 \rho(r_s) c_s(r_s) \tag{14}$$

and by using the expression for the sonic radius from Section 2.1, we get:

$$\dot{M} = \frac{\pi G^2 M^2 \rho(r_s)}{c_s^3(r_s)} \tag{15}$$

Outside the sonic radius the flow is subsonic, which implies that density and sound speed do not change significantly. The full integral of the Bondi problem relates $\rho(r_s)$ to $\rho(\infty)$ via a factor of order unity that depends on γ, similar for the sound speed. The Bondi accretion rate hence becomes:

$$\dot{M} = \frac{\pi G^2 M^2 \rho(\infty)}{c_s^3(\infty)} \tag{16}$$

where

$$f = \left(\frac{2}{5-3\gamma}\right)^{\frac{5-3\gamma}{2(\gamma-1)}} \tag{17}$$

3.2 Analysis

Cases of particular interest for astrophysics are $\gamma = 1$ (isothermal temperature established by external condition, e.g., radiation field) and $\gamma = 5/3$ (adiabatic, monoatomic gas). By using the *sympy* package in Python (see Appendix 1), the limiting values of γ for these cases were calculated and are presented below.

$$\begin{cases} \lim_{\gamma\to 1} f(\gamma) = e^{3/2} \\ \lim_{\gamma\to 5/3} f(\gamma) = 1 \end{cases}$$

Next, by using the *astropy* constants G and *M-sun* and the *astropy* units module in Python (see Appendix 2), we can show that for $\gamma = 1.4$, the Bondi accretion rate may be written as:

$$\dot{M} = 1.4 \times 10^{11} \left(\frac{M}{M_\odot}\right)^2 \left(\frac{\rho(\infty)}{10^{-24}\ \mathrm{g\ cm^{-3}}}\right) \left(\frac{c_s(\infty)}{10\ \mathrm{km\ s^{-1}}}\right)^{-3} \mathrm{g\ s^{-1}} \tag{18}$$

Eq. 18 relates the properties of gas in interstellar space (the interstellar medium) to the rate massive objects can grow via Bondi accretion. It is quite obvious that this form of the equation is much more useful as the properties of the interstellar medium vary by many orders of magnitude.

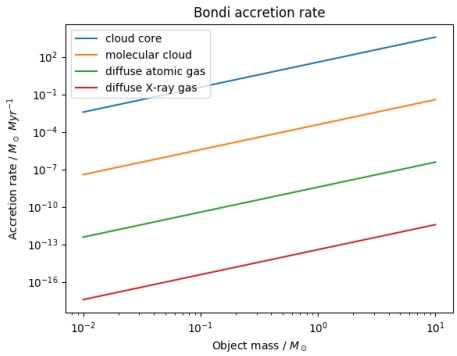

Figure 1: Bondi accretion rate for four typical gas phases

In this project we also considered four typical gas phases in the interstellar medium and studied the accretion rate. The plot in Fig. 1 shows the Bondi accretion rate in solar masses per Myr as a function of the mass of the central object for these gas phases listed in Table 1 below. It was done using the *plot* function of the Python module *matplotlib* (see Appendix 3) and it shows that while the collapse initially starts in a cold molecular cloud, the infalling material then accreting onto the protostar increases its kinetic energy and temperature, transforming it into an IR source. This hot molecular cloud core is the environment where the most significant growth of the protostar takes place.

Table 1: Gas phases parameters

	cloud core	molecular cloud	diffuse atomic gas	diffuse X-ray gas
density [m_p/cm^3]	10^4	10^2	1	10^{-1}
sound speed [km/s]	10^{-1}	1	10	10^2

4 Initial mass function

4.1 Background

One deduces from observations that after having been embedded in dense gas for a few Myr the star formation regions (clusters or associations of stars) become exposed, i.e. objects whose age exceeds this time are found to be gas-free. Gas accretion has then terminated, and the masses of the newly formed ensemble of stars are set. The stars will in the course of their evolution lose again mass due to, e.g., stellar winds. When talking about the mass distribution of an ensemble of stars, it is therefore important to specify the time for which the statement is made. The distribution function of the stellar masses at the time when a star-forming region becomes exposed is called the initial mass function. Observations show that for stars more massive than 0.5 $M_\odot$ the initial mass function follows the Salpeter law, i.e.:

$$\frac{dn}{dN} \propto M^{-2.35} \tag{19}$$

Here, dN denotes the number of stars in a given mass bin at mass M with width dM. After a sufficiently long accretion time, Bondi accretion will transform any core mass function into:

$$\frac{dn}{dN} \propto M^{-2} \tag{20}$$

This is often regarded as quite close to the observed IMF and taken as an indication that Bondi accretion is plausibly playing a role in star formation.

4.2 Analysis

In this study we modeled the initial mass function with a Monte Carlo experiment. This is a broad class of computational algorithms based on repeated random sampling. First, the Python module *numpy.random* was used to generate a uniform random function for the masses of an ensemble of n protostellar cores between 0 and 0.01 $M_\odot$ (see Appendix 4). Fig. 2 presents the histogram of core masses with both logarithmic mass binning and linear mass binning for an initial ensemble of n=10,000 cores.

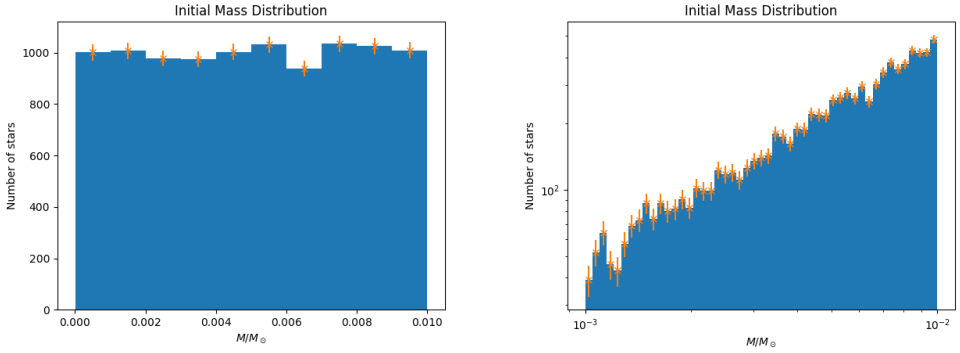

Figure 2: Initial mass distribution for 10,000 protostellar cores generated by a uniform random function with (a) linear binning and (b) logarithmic binning

The appearance of the two plots is as expected. While the linear plot shows a rather "constant" number of stars in each bin as a result of the uniform random generation function, the bins in the logarithmic plot have larger and larger actual sizes as they approach 10^{-2} due to the logarithmic nature of the axis scale. So as a result, each bin in the logarithmic plot will contain more stars than the previous bin because it will have a larger linear size.

Next multiple simulations were performed using a Python code that integrates numerically the accretion of mass on to the cores over time using the Bondi formula in Eq. 18 (see the code in Appendix 5). Initially, we set a total accretion time of 3 Myr, a timestep of 10^5 years, a density of 1.6×10^{-20} g/cm^3 and a sound speed of 100 m/s, i.e. typical parameters for molecular cloud cores and star formation in general and we imposed a mass limit of 150 $M_\odot$ (the observed upper mass where accretion usually ends). The results are presented in Fig. 3, along with a function proportional to 1/M for comparison, for decreasing timesteps in the range of 10^5 to 10^1 years.

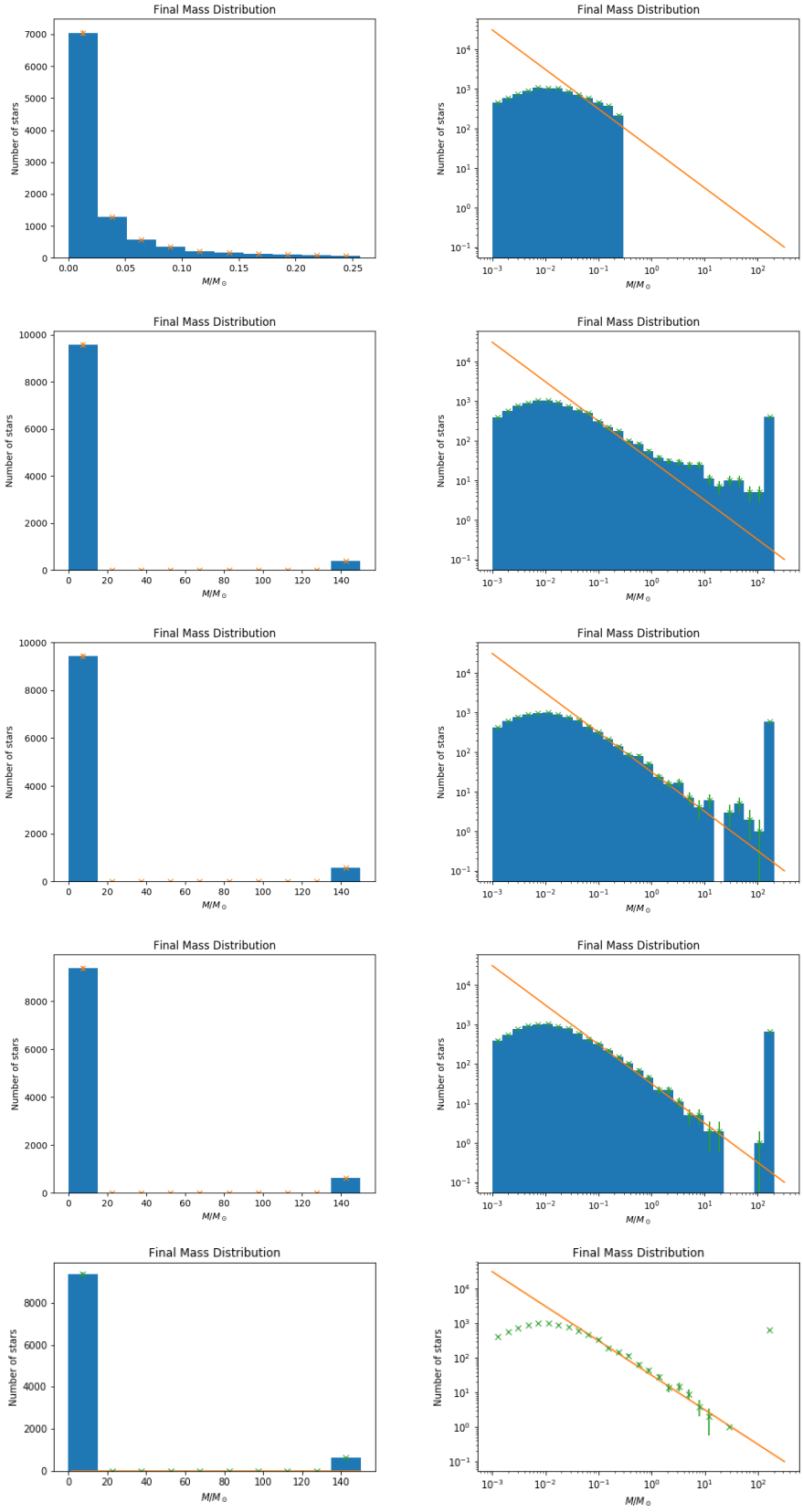

Figure 3: Final mass distribution following Bondi accretion. Timestep of integration decreases from top to bottom in steps of a factor of 10

This means that the final mass distribution of the cluster, generated by the uniform random function we used and Bondi accretion in Eq. 18 is very well fitted by a function $f(M) = \alpha \frac{1}{M}$, with the proportionality constant $\alpha \approx 31.6$. From Fig. 3 it is obvious that as the timestep of the simulation is decreased, a more accurate fit is produced and the IMF converges to the above mentioned function.

The next step in the simulation was to repeat the experiment after increasing and then decreasing the gas density by a factor of 3. The results are presented in Fig. 4. The observed initial mass function does not show an excess at 150 $M_\odot$.

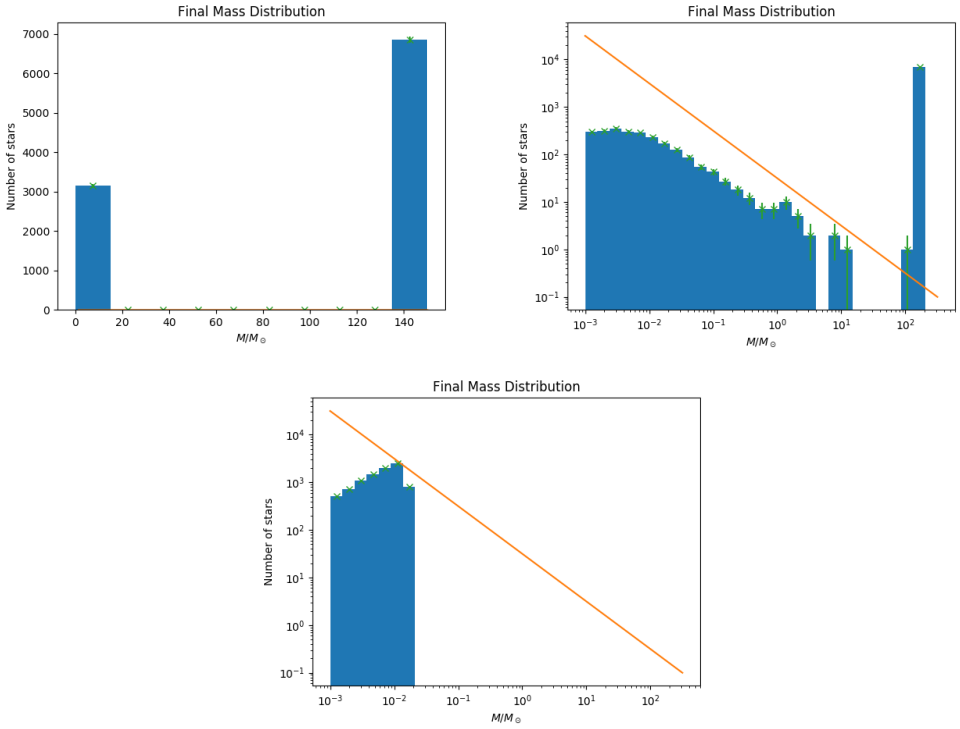

Figure 4: Final mass distribution following Bondi accretion for (a) $3 \times \rho$ and (b) $\frac{1}{3} \times \rho$, where ρ is the initial density

The observed initial mass function does not show an excess at 150 $M_\odot$. However, it is apparent from Fig. 4 that for the larger density case there is a significant number of stars with the maximum total mass that is allowed by the simulation. We can see from Eq. 13 that an increase in the gas density will increase the accretion rate. As we know, in the absence of any pre-existing stars that would generate accretion, the initial gas usually collapses and forms a proto-star containing all the gas mass in the region, in a period of time approximately equal to one free-fall time (dynamical timescale). [2] As a result, as the density increases towards infinity on the dynamical timescale, the accretion rate also increases on this timescale apparently until all the gas is used up, or has formed another star in the centre of the cluster, meaning that generally, this would be a viable model of the process.

The last part of the project was to assume that instead of having the same density for all stars, the gas densities in the immediate environment of the protostars are rather randomly distributed with a probability distribution that follows a lognormal function as expected from turbulence theory. This was done in Python through a *numpy.random* function (see Appendix 5), and the result was compared to the observed IMF generated by the code in Appendix 6. The plots are presented in Fig. 5 below, along with the observed IMF in Eq. 20.

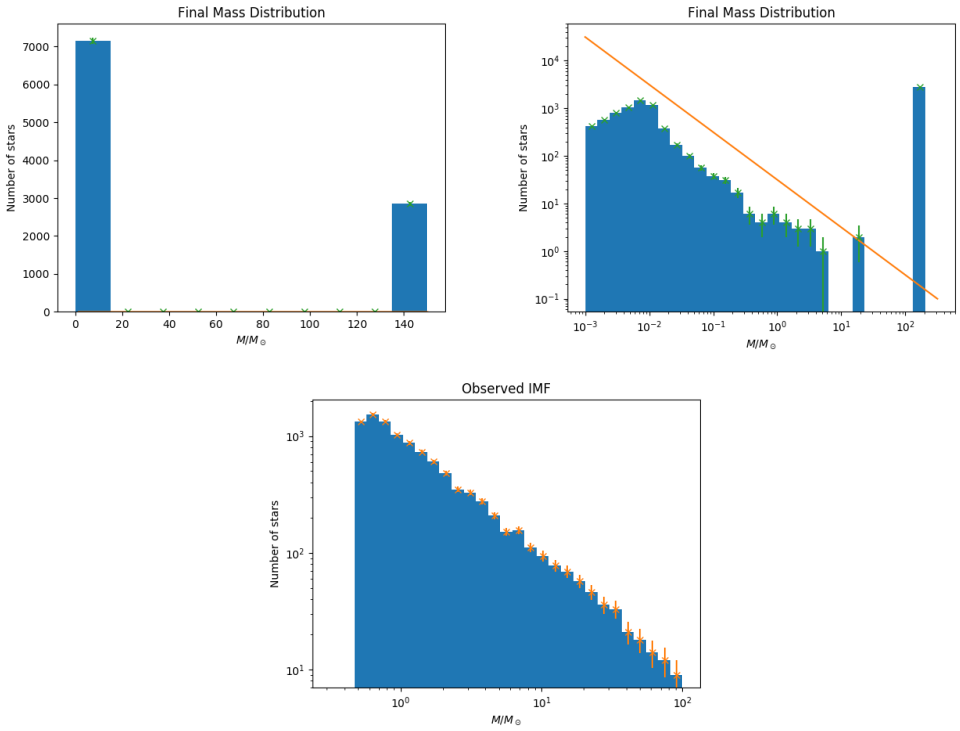

Figure 5: (a) Final mass distribution following Bondi accretion for a random-generated density compared to (b) the observed IMF

It is apparent that the two situations have a lot of similarities, except for the fact that there seems to be quite a large number of stars very close to the upper mass limit. Despite this, we can conclude that there is a rather strong correlation between the two cases and that the model is quite accurate in describing the accretion process that takes place in such clouds of gas.

5 Summary

We can conclude that the random function used in this simulation, along with a uniform random distribution of density, generate a final mass distribution that quite accurately describes stellar formation in the cloud core through Bondi accretion, by comparison with a Salpeter model observed IMF.

References

[1] Bondi (1952) MNRAS *On Spherically Symmetric Accretion* Vol. 112, p. 196.

[2] I. A. Bonnell, M. R. Bate, C. J. Clarke and J. E. Pringle, *Competitive accretion in embedded stellar clusters*, p. 3

Appendix 1

```
# -*- coding: utf-8 -*-
"""
Created on Thu Dec 14 22:08:32 2017

@author: ai15aax
"""

import sympy as sp
from sympy.abc import x

f=(2/(5-3*x))**((5-3*x)/(2*x-2))
gamma1=1
gamma2=5/3
print('\n For gamma=1, f=',sp.limit(f,x,gamma1))
print('\n For gamma=5/3, f=',sp.limit(f,x,gamma2))
```

Appendix 2

```
# -*- coding: utf-8 -*-
"""
Created on Thu Dec 14 23:06:19 2017

@author: ai15aax
"""

from astropy.constants import G,M_sun
import astropy.units as u
import numpy as np

Msun=1*M_sun
rhoinf=10**(-24)*(u.g)*(u.cm)**(-3)
cs=10*(u.km)*(u.s)**(-1)

x=1.4 # gamma
f=(2/(5-3*x))**((5-3*x)/(2*x-2))

Mdot=f*np.pi*(G**2)*(Msun**2)*rhoinf/(cs**3)

print('\n Mdot =', Mdot.cgs)
```

Appendix 3

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

import numpy as np
import matplotlib.pyplot as plt
from astropy.constants import M_sun


M=np.logspace(-2,1,100)
density=[10**4,10**2,1,10**(-2)]
soundspeed=[10**(-1),1,10,10**2]

Msun=1*M_sun
print(Msun)
case_names=['cloud core','molecular cloud','diffuse atomic gas','diffuse X-ray gas']

for i in range(4):
    rho=density[i]
    cs=soundspeed[i]
    Mdot=1.4*(10**11)*(M**2)*(rho*1.6726219)*((cs/10)**(-3))/(Msun.cgs/(365.25*24*3600*10**6))
    plt.plot(M,Mdot,label=case_names[i])

#plt.axis([10**(-2),10,10**(-16),10**2])
plt.legend()
plt.xscale('log')
plt.yscale('log')
plt.title('Bondi accretion rate')
plt.xlabel('Object mass / $M_\odot$')
plt.ylabel('Accretion rate / $M_\odot$ $Myr^{-1}$')
filename='bondi_accretion'
plt.savefig(filename)
plt.show()
plt.close()
```

Appendix 4

```
# -*- coding: utf-8 -*-
"""
Created on Tue Dec 12 13:49:59 2017

@author: ai15aax
"""
#Exercise 5 A
import numpy.random as rnd
import numpy as np
import matplotlib.pyplot as plt

x=0.01*rnd.uniform(size=10000)
print(x)

a=plt.figure(1) # creating the log-log histogram:
counts,bin_edges=np.histogram(x,bins=np.logspace(-3,-2,70))
plt.hist(x,range=(-3.5,-1.5),bins=np.logspace(-3,-2,70),log=True)
plt.yscale('log')
plt.xscale('log')
bin_centres=(bin_edges[:-1]+bin_edges[1:])/2. # this calculates the x position of the error bars
err=np.sqrt(counts) # error from Poisson distribution
plt.errorbar(bin_centres,counts,yerr=err,fmt='x') # plotting the error bars
plt.xlabel("$M/M_\odot$")
plt.ylabel("Number of stars")
plt.title("Initial Mass Distribution")
plt.savefig("log_histogram") # saves with extension .png

b=plt.figure(2)
counts2,bin_edges2=np.histogram(x)
plt.hist(x)
bin_centres2=(bin_edges2[:-1]+bin_edges2[1:])/2. # this calculates the x position of the error bars
err2=np.sqrt(counts2) # error from Poisson distribution
plt.errorbar(bin_centres2,counts2,yerr=err2,fmt='x') # plotting the error bars
plt.xlabel("$M/M_\odot$")
plt.ylabel("Number of stars")
plt.title("Initial Mass Distribution")
plt.savefig("linear_histogram") # saves with extension .png

plt.show()

plt.close()
```

Appendix 5

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Dec 12 14:30:35 2017

@author: ai15aax
"""
#Exercise 5 B, C, D

import numpy.random as rnd
import numpy as np
import matplotlib.pyplot as plt

in_string1=input('Number of stars= ')
n=int(in_string1)
in_string2=input('Timestep in years= ')
dt=float(in_string2)

M=0.01*rnd.uniform(size=n) # random generation function for core masses
print('\n Initial mass distribution=',M)

T=3*10**6 # total accretion time (years)
cs=0.1 # sound speed (km/s)
#rho=1.6*10**(-20) # initial density (constant) (g/cm^3)

# implementation of random density generation:
rho_max=10**(-20)
rho=10.**rnd.normal(size=n)*rho_max

N=int(T/dt)

Sum=[]
limit=150. # maximum accretion limit in solar masses
print('\n Generating stellar cluster...')

for i in range(n):
    mass=M[i]
    density=rho[i] # this is uncommented when using the random density generation
    for j in range(N):
        # rate of accretion calculation in solar masses/year (mass dependent):
        Mdot=1.4*(10**11)*(mass**2)*(density/(10**(-24)))*((cs/10)**(-3))/((1.989*10**33)/(365.25*24*3600))
        # accretion:
        dM=Mdot*dt
        mass=mass+dM
        if mass>limit:
           mass=150.
    Sum.append(mass)

a=plt.figure(1) # creating the log-log histogram:
ax  = a.add_subplot(111) # adding a subplot for plotting 1/M on the same graph
# now plot
counts,bin_edges=np.histogram(Sum,bins=np.logspace(-3,2.5,30))
ax.hist(Sum,range=(-3.5,2.5),bins=np.logspace(-3,2.5,30),log=True)
x = np.logspace(-3,2.5,30)
alpha=31.6 # proportionality constant
f=alpha*(1/x)
h = ax.plot(x, f) # plotting 1/M

plt.yscale('log')
plt.xscale('log')
bin_centres=(bin_edges[:-1]+bin_edges[1:])/2. # this calculates the x position of the error bars
```

```
err=np.sqrt(counts) # error from Poisson distribution
plt.errorbar(bin_centres,counts,yerr=err,fmt='x') # plotting the error bars
plt.xlabel("$M/M_\odot$")
plt.ylabel("Number of stars")
plt.title("Final Mass Distribution")
plt.savefig("log_histogram_final") # saves with extension .png

b=plt.figure(2)
axx  = b.add_subplot(111)
counts2,bin_edges2=np.histogram(Sum)
axx.hist(Sum)
y=np.setdiff1d(np.linspace(0,150,100),[0]) #to remove the zero
ff=1/y
hh = axx.plot(y, ff)

bin_centres2=(bin_edges2[:-1]+bin_edges2[1:])/2. # this calculates the x position of the error bars
err2=np.sqrt(counts2) # error from Poisson distribution
plt.errorbar(bin_centres2,counts2,yerr=err2,fmt='x') # plotting the error bars
plt.xlabel("$M/M_\odot$")
plt.ylabel("Number of stars")
plt.title("Final Mass Distribution")
plt.savefig("linear_histogram_final") # saves with extension .png

plt.show()

plt.close()

print('\n Stellar cluster done.')
```

Appendix 6

```
# -*- coding: utf-8 -*-
"""
Created on Fri Dec 15 18:11:30 2017

@author: ai15aax
"""

import numpy as np
import matplotlib.pyplot as plt
in_string1=input('Number of stars= ')
n=float(in_string1)
print('Generating stellar cluster...')

def pdf(x,zeta0):
    """
    Probability distribution function used in Salpeter's power law IMF model:
    ξ(M) = ξ0 * M^(-2.35)
    Input:
    x = mass of the star (M)
    zeta0 = constant which sets the local stellar density (ξ0)
    Output:
    Initial Mass Function ξ(M)  (IMF) i.e. the number of stars in the cluster
    with masses between M and M + ΔM
    """
    return zeta0*x**(-2)

def randomvariate(pdf,n,xmin,xmax):
 """
 Rejection method for random number generation
 ==============================================
 Uses the rejection method for generating random numbers derived from an arbitrary
 probability distribution.
 Usage:
 >>> randomvariate(pdf,n,xmin,xmax)
  where
  pdf : probability distribution function from which you want to generate random numbers
  n : desired number of random values
  xmin,xmax : range of random numbers desired
 Returns:
  the sequence (ran,ntrials) where
   ran : array of shape N with the random variates that follow the input P
   ntrials : number of trials the code needed to achieve n
 The algorithm:
 - generate x' in the desired range
 - generate y' between Pmin and Pmax (Pmax is the maximal value of your pdf)
 - if y'<pdf(x') accept x', otherwise reject
 - repeat until desired number is achieved
  """
  # Calculates the minimal and maximum values of the PDF in the desired
  # interval. The rejection method needs these values in order to work
  # properly.
  x=np.linspace(xmin,xmax,n)
  zeta0=(n*1.35)/((xmin)**(-1.35)-(xmax)**(-1.35)) # the zeta0 factor taken from literature
  y=pdf(x,zeta0)
  pmin=0.
  pmax=y.max()
  # Counters
  naccept=0
  ntrial=0
  # Keeps generating numbers until we achieve the desired n
```

```
  ran=[] # output list of random numbers
  while naccept<n:
      x=np.random.uniform(xmin,xmax) # x'
      y=np.random.uniform(pmin,pmax) # y'
      if y<pdf(x,zeta0):
          ran.append(x)
          naccept=naccept+1
      ntrial=ntrial+1
  ran=np.asarray(ran)

  return ran,ntrial

result=randomvariate(pdf,n,0.5,150)
cluster=result[0]
print('\n cluster=',cluster)

a=plt.figure(1) # creating the log-log histogram:
counts,bin_edges=np.histogram(cluster,bins=np.logspace(-0.5,2,30))
plt.hist(cluster,range=(-0.5,2.5),bins=np.logspace(-0.5,2,30),log=True)
plt.yscale('log')
plt.xscale('log')
bin_centres=(bin_edges[:-1]+bin_edges[1:])/2. # this calculates the x position of the error bars
err=np.sqrt(counts) # error from Poisson distribution
plt.errorbar(bin_centres,counts,yerr=err,fmt='x') # plotting the error bars

plt.xlabel("$M/M_\odot$")
plt.ylabel("Number of stars")
plt.title("Observed IMF")
plt.savefig("histogram") # saves with extension .png

plt.show()

plt.close()
```