Monte Carlo Simulation of a Globular Cluster

Michael Sell

UPC

December 15, 2024

Computational Astrophysics

Santiago Torres

Contents

Abstract

The goal of this project is to simulate a Globular Cluster population using Monte Carlo methods. This is done to visualize the properties of main-sequence and white dwarf stars. Using this method, the mass distribution, star temperature, and luminosity relationships can be observed from the simulation for different cluster ages. From these values, the percentage of white dwarfs, main-sequence, neutron stars, and black holes can be obtained, comparing them to their respective ages. It can also be determined the MSTO, or main-sequence turn-off, for each age. From this, it can then be analyzed, how sensitive the main sequence track is to the change of variables, in particular, the minimum star mass. This is what we hope to observe.

1 Introduction

Based on the age and mass of stars, a wide range of properties can be observed. With this in mind, we can perform a simulation using the Initial Mass Function (IMF) to generate a population of stars in order to estimate these aforementioned properties, such as luminosity, effective temperature (T_{eff}), and cooling times.

2 Methodology

Using a population of $N = 1000$ stars using a random sampling method based on the IMF. The conditions, inputs, and guidelines for the simulation can be seen in the following.

1. Minimum and maximum masses: 0.1 $M_\odot$ to 100 $M_\odot$.
2. Cluster age, $t_{cluster}$: 8,10, and 12 Gyr.
3. Born age, t_{born}, according to a constant SFR with a burst of $\Delta t = 1.0$ Gyr.
4. Luminosity, radius, and T_{eff} computed using standard stellar models.
5. Main sequence mass, M_{ms}, following the IMF of Salpeter (1995) with standard slope $\alpha = -2.35$ and a range of masses $0.1 < M/M_\odot \leq 100$ with a relationship of:
$$\phi(M) = (M/M_\odot)^\alpha$$
6. For main sequence luminosities, from Salaris & Cassisi (2005):
$$\frac{L}{L_\odot} = \begin{cases} 0.23M^{2.3} & \text{if } M \leq 0.43M_\odot \\ M^4 & \text{if } 0.43 < M \leq 2M_\odot \\ 1.4M^{3.5} & \text{if } 2 < M \leq 5M_\odot \\ 32000M & \text{if } M > 5M_\odot \end{cases}$$
7. For the main sequence life-time:, t_{MS}, from Iben & Laughlin (1989):
$$t_{\text{MS}} = 10\left(\frac{M_{\text{MS}}}{M_\odot}\right)^{-3.5} \text{ Gyr}$$
8. Progenitors, or stars with $M < 10M_\odot$ evolve into white dwarfs. Stars with $M > 10M_\odot$ form neutron stars or black holes.
9. White dwarf mass, m_{WD}, from Iben & Laughlin (1989):
$$M_{\text{WD}} = 0.49\exp(0.095M_{\text{MS}})$$
10. White dwarf cooling age:
$$t_{\text{cool}} = t_{\text{cluster}} - t_{\text{born}} - t_{\text{MS}}$$
11. White dwarf cooling model follows the Mestel law: $t_{\text{cool}} = CL^{-5/7}$, which gives
$$-\log\left(\frac{L}{L_\odot}\right) = \frac{7}{5}\log(t_{\text{cool}}) + 3$$
12. Effective temperature from the Stefan–Boltzmann law:
$$L = 4\pi R^2\sigma T_{\text{eff}}$$
13. Main-sequence star radii:
$$R_{\text{MS}}/R_\odot = \begin{cases} 100.66\log(M/M_\odot) + 0.05 & \text{if } M \geq 1.12M_\odot \\ M/M_\odot & \text{if } M \leq 1.12M_\odot \end{cases}$$
14. White dwarf radii:
$$R_{\text{WD}}/R_\odot = C\left(\frac{M_{\text{WD}}}{M_\odot}\right)^{1/3}, C = 0.0101$$
15. $L = 3.827 \times 10^{26}\,\text{W}$
16. $R = 6.969 \times 10^{8}\,\text{m}$.

The simulation is implemented in Python, and the core code is appended in appendix A, below.

3 Results

3.1 Mass Distribution

The stellar mass distribution can be seen below for 10 Gyrs. A similar graph could be generated for other values of age; however, they all follow the same distribution. It can be observed that most stars have low masses, consistent with the IMF.

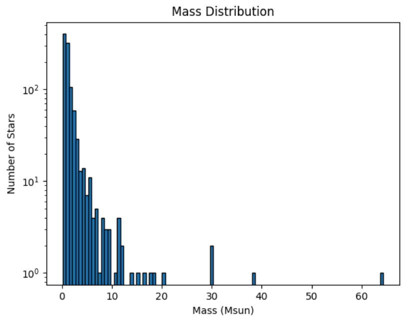

Figure 1: Mass distribution of simulated stars for 10 Gyrs (logarithmic scale).

3.2 Star Formation Ages (t_{born})

The t_{born} distribution for 10 Gyrs. can be seen below. Again, this can also be generated for different ages. However, for the sake of brevity, an age of 10 Gyrs is shown below, while other ages produce a similar result. It can be seen that random birth ages are uniformly distributed over the lifetime of the cluster.

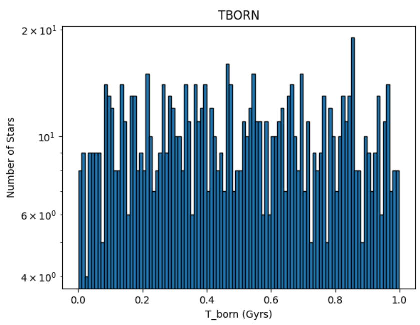

Figure 2: Distribution of star formation ages for 10 Gyrs. (t_{born}).

3.3 Temperature vs Luminosity

The relationship between T_{eff} and luminosity for main-sequence and white dwarf stars is shown in the following figures for 8, 10, and 12 Gyrs., respectively.

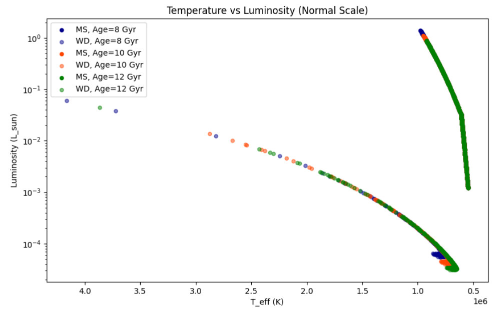

Figure 3: Temperature (T_{eff}) vs Luminosity for main sequence (blue) and white dwarf (orange) stars for 8 Gyrs.

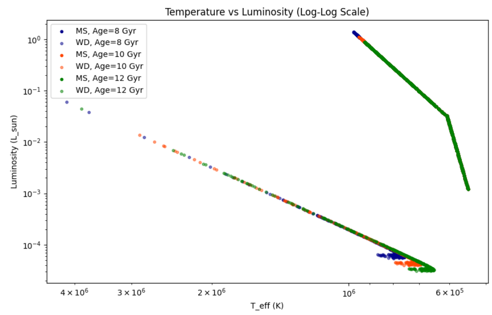

Figure 4: Log-log plot of T_{eff} vs Luminosity for 12 Gyrs.

3.4 Temperature and Luminosity with Noise

The following figures incorporate a 10% Gaussian noise, providing a visualization of the scatter in observed stellar populations. The following figures are again generated for 8, 10, and 12 Gyrs.

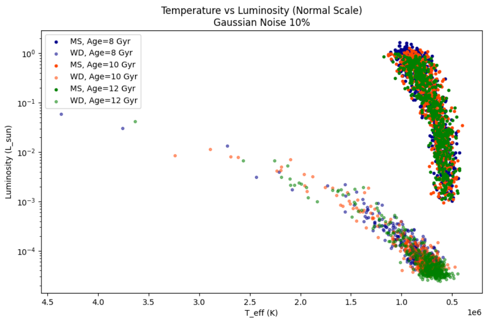

Figure 5: Noisy temperature (T_{eff}) vs Luminosity for main sequence (blue) and white dwarf (orange) stars for 8 Gyrs.

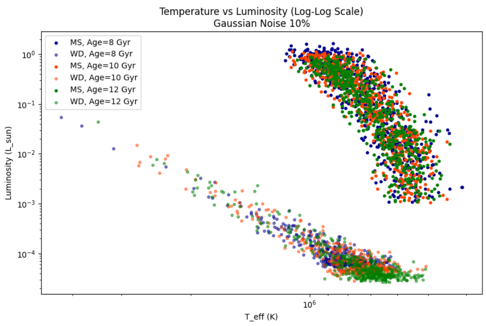

Figure 6: Temperature (T_{eff}) vs Luminosity for main sequence (blue) and white dwarf (orange) stars with 10% Gaussian noise using log-log axis for 8 Gyrs.

3.5 Percentage of WD, MS, NS, and Black Holes per age

Cluster Age (Gyr)	8 Gyr	10 Gyr	12 Gyr
Percentage of Black Holes/Neutron Stars	1.7%	1.7%	1.7%
Percentage of White Dwarfs	36.4%	40.3%	43.9%

Table 1: Percentage of Black Holes/Neutron Stars and White Dwarfs for Different Cluster Ages

3.6 Main-sequence Turn-off and Sensitivity (MSTO)

The main-sequence turn-off refers to the point at which MS stars begin to leave the MS, normally due to fuel exhaustion. We can modify the code slightly in order to find this point, while simultaneously running the simulation for multiple values of minimum star masses to analyze how sensitive the main sequence track is to a change in variables.

Table 2: MSTO Properties for min_mass = $0.1\,M_\odot$:

Age (Gyr)	**MSTO Mass ($M_\odot$)**	T_{eff} **(K)**	**Luminosity ($L_\odot$)**
8	1.03	5843.82	1.11
10	0.96	5636.55	0.83
12	0.43	3803.18	0.04

Table 3: MSTO Properties for min_mass = $0.3\,M_\odot$:

Age (Gyr)	**MSTO Mass ($M_\odot$)**	T_{eff} **(K)**	**Luminosity ($L_\odot$)**
8	0.91	5496.08	0.68
10	0.78	5098.98	0.37
12	0.81	5192.38	0.43

Table 4: MSTO Properties for min_mass = $0.5\,M_\odot$:

Age (Gyr)	**MSTO Mass ($M_\odot$)**	T_{eff} **(K)**	**Luminosity ($L_\odot$)**
8	0.72	4902.14	0.27
10	0.58	4405.37	0.12
12	0.53	4183.75	0.08

The tables above give numerical values for the relationship the minimum mass has with parameters such as MSTO, effective temperature, and luminosity. We see that there is an inverse relationship between the minimum star mass and the main sequence track in general. We can also plot these values to obtain visual representations of what is happening here:

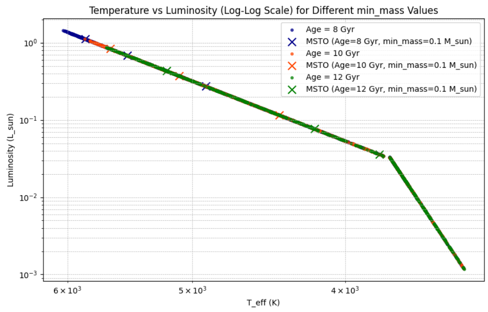

Figure 7: Temperature (T_{eff}) vs Luminosity and MSTOs for different values of minimum mass

We can see above the difference in the MSTOs as the age increases, as well as the difference in T_eff and Luminosity. It should be noted that because of the scale, the difference between the effective temperature and luminosity is not as apparent as when the graphs are separated. I have not included the separate graphs as we have numerical data and for the sake of brevity.

4 Conclusions

Throughout the various simulations, we were able to see the various mass distributions of simulated stellar systems, the relationships between effective temperature and luminosity over certain values of years, what these systems would look like with a 10% Gaussian noise, and finally were able analyze the effects that the minimum solar mass has on the main sequence and Main-sequence Turn-off values.

While most of the values and simulations above seem to be accurately modeling the expected results of these systems, there are some values that should be addressed in this report. First of all, the values in section 3.6 are mostly correct, following an inverse trend with time. However, in table 3, there seems to be an anomaly where the values for MSTO, T_eff, and Luminosity drop unexpectedly. From what we expect, this should not be the case. We should see a consecutive decrease between the three.

While we could be attribute this to many things, this could be due to inconsistencies in the initial mass function (IMF), or perhaps something to do with a large distribution of low-mass stars. Because MSTO is largely impacted by stars with greater mass, this could potentially impact our results. In fact, we can see a more desirable result when we plot a population of just $N = 10$:

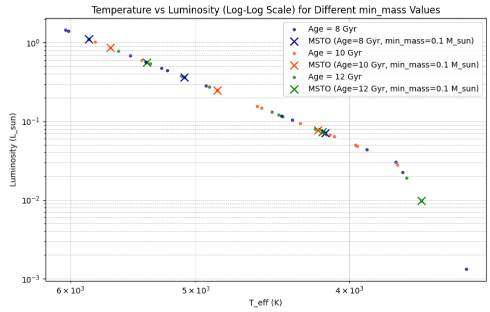

Figure 8: Temperature (T_{eff}) vs Luminosity and MSTOs for $N = 10$

This could signify that there might be an over-population of smaller-mass stars skewing the MSTO points.

Besides this being said, we have managed to produce the desired results for our stellar simulation. From here, future steps could be taken to expand on the basis we have formed. Some examples could be adding realistic error to the luminosity and temperature, computing the luminosity function, building a spatial distribution of the objects in the cluster, or even adding a binary population. Along with all of these things come many more possibilities, made possible by the flexibility of our software.

However, we can simply add a small amount of code to model a 2d HR-Diagram to visualize a Density Map using the Seaborn library. This gives us a beautiful looking graph:

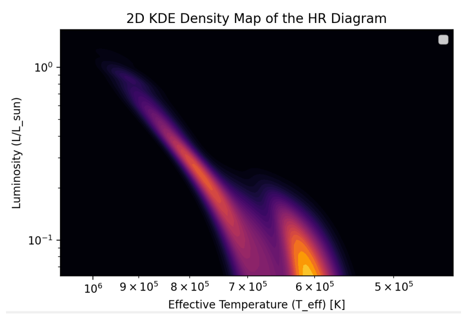

Figure 9: Temperature (2D HR-Diagram

A Appendix: Python Code

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


np.random.seed(9)

# Create variables
N = 1000            # Number of objects
# t_cluster = 10      # Cluster age
t_born = []           # Born age
t_ms = []             # Main Sequence Temperature
t_cool = []           # T_cool temperatures
dt = 1                # Timestep
alpha = 2.35          # IMF constant
min_mass = 0.1        # Minimum Star Mass
max_mass = 100        # Maximum Star Mass
mass_ms = []          # Main Sequence Mass Array
mass_wd = []          # White Dwarf Mass Array
num_wd = 0            # White Dwarf Count
num_bh = 0            # Black Holes and Neutron Star count
lum_ms = []           # Main Sequence Luminosity Array
lum_wd = []           # White Dwarf Luminosity Array
t_eff = []            # Main Sequence Effective Temperature
t_eff_wd = []         # White Dwarf Effective Temperature
R_ms = []             # Main Sequence Radius
R_wd = []             # White Dwarf Radius
sigma = 5.67e-8       # W/m^2 K^4
C = 0.0101            # Constant
R_solar = 6.969e8     # Solar Radius
L_solar = 3.827e26    # Solar Luminosity



# Define cluster ages
cluster_ages = [8, 10, 12]

# Store results for different cluster ages
all_lum_ms = []       # Main sequence luminosities
all_t_eff = []        # Main sequence effective temperatures
all_lum_wd = []       # White dwarf luminosities
all_t_eff_wd = []     # White dwarf effective temperatures

# Simulate for each cluster age
for t_cluster in cluster_ages:

    np.random.seed(9)  # Ensure reproducibility for each age

    # Initialize lists for combined graphs
    mass_ms, t_ms, t_born, t_cool = [], [], [], []
    lum_ms, lum_wd, t_eff, t_eff_wd = [], [], [], []
    num_wd, num_bh = 0, 0

    # Monte Carlo Simulation
    while len(mass_ms) < N:

        # Generate random number for comparison between 1 and 0
        rand_y = np.random.uniform(0,1)

        # Generate IMF with random mass vector between min-max mass
        rand_mass = np.random.uniform(min_mass, max_mass)
```

```python
        mass_i = (rand_mass)**(-alpha) # Use mass_i as the probability function of acceptance


        # Accept Reject Method
        if rand_y < mass_i:
            # ACCEPT
            mass_ms.append(rand_mass) # Accept the random mass in between min and max, not mass_i
        else:
            # REJECT
            continue


    # Create T_ms and T_i arrays
    for i in range(N):
        # Calculate t_ms
        t_msi = 10 * mass_ms[i] ** -3.5
        t_ms.append(t_msi)

        # Calculate t_born
        t_i = dt * np.random.uniform(0, 1)
        t_born.append(t_i)

        # Calculate t_cool
        t_cool_i = t_cluster - t_born[i] - t_ms[i]
        t_cool.append(t_cool_i)

    # Finding luminosity and temperature, etc. for MS and WD
    for i in range(N):

        mass = mass_ms[i]

        # MAIN SEQUENCE CONDITIONS
        if t_cool[i] <= 0:
            # Calculate Luminosity based on mass ranges
            if mass > 55:
                lum_i = 32000 * mass
            elif 2 < mass <= 55:
                lum_i = 1.4 * mass ** 3.5
            elif 0.43 < mass <= 2:
                lum_i = mass ** 4
            elif mass <= 0.43:
                lum_i = 0.23 * mass ** 2.3
            lum_ms.append(lum_i)

            # Main Sequence Radius
            if mass >= 1.12:
                R_i = 10 ** (0.66) * np.log(mass) + 0.5
            else:
                R_i = mass
            R_ms.append(R_i)

            # Main Sequence Effective Temperature
            t_eff_i = (lum_i * L_solar/ (4 * np.pi * R_i**2 * R_solar * sigma)) ** (1 / 4)
            t_eff.append(t_eff_i)

            # print(f"mass[{i}] = {mass}, t_cool[{i}] = {t_cool[i]}, lum[{i}] = {lum_i}, R[{i}] = {R_i}, T_eff[{i}] = {t_eff_i}")


        # WHITE DWARF CONDITIONS
```

```python
        elif t_cool_i > 0:
            if mass_ms[i] < 10:
                num_wd += 1 # White Dwarf

                # White Dwarf Mass
                mass_wd_i = 0.49 * np.exp(0.095 * mass_ms[i])
                mass_wd.append(mass_wd_i)

                # White Dwarf Radius
                R_wd_i = C / mass_wd_i**(1/3)
                R_wd.append(R_wd_i)

                # White Dwarf Luminosity
                lum_wd_i = 10**(-3) / t_cool[i]**(7/5)
                lum_wd.append(lum_wd_i)

                # White Dwarf T_eff
                t_eff_wd_i = (lum_wd_i * L_solar / (4 * np.pi * R_wd_i**2 *
    R_solar * sigma)) ** (1 / 4)
                t_eff_wd.append(t_eff_wd_i)

            # Black Hole or Neutron Star
            elif mass_ms[i] > 10:
                num_bh += 1 # Black Hole or Neutron Star

    # Store results for this cluster age
    all_lum_ms.append(lum_ms)
    all_t_eff.append(t_eff)
    all_lum_wd.append(lum_wd)
    all_t_eff_wd.append(t_eff_wd)

    print(f'Number of Black Holes/Neutron Stars for Age: {t_cluster} = {num_bh}
    ')
    print(f'Number of Black White Dwarfs for Age: {t_cluster} = {num_wd}')



    #_________________________________________________________________
    # Individual Plots:
    # Un-comment for individual plots
    #_________________________________________________________________

    # # Create Gaussian Noise Main Sequence
    # lum_noisy = lum_ms * (1 + np.random.normal(0, 0.1, size=len(lum_ms)))
    # temp_noisy = t_eff * (1 + np.random.normal(0, 0.1, size=len(t_eff)))

    # # Create Gaussian Noise White Dwarf
    # lum_wd_noisy = lum_wd * (1 + np.random.normal(0, 0.1, size=len(lum_wd)))
    # temp_wd_noisy = t_eff_wd * (1 + np.random.normal(0, 0.1, size=len(
    t_eff_wd)))

    # # Debug, Masses from largest to smallest
    # # sorted_arr = sorted(mass_ms, reverse=True)
    # # print("Sorted array from largest to smallest:", sorted_arr)

    # # Print Number of Black Holes and Neutron Stars, as well as White Dwarfs
    # print(f"Number of Black Holes or Neutron Stars: {num_bh}")
    # print(f"Number of White Dwarfs: {num_wd}")

    # # Plot Original Mass
    # plt.hist(mass_ms, bins=100, log=True, edgecolor='black')
    # plt.xlabel('Mass (Msun)')
    # plt.ylabel('Number of Stars')
```

```
    # plt.title('Mass Distribution')
    # plt.show()

    # # Plot t_born
    # plt.hist(t_born, bins=100, log=True, edgecolor='black')
    # plt.xlabel('T_born (Gyrs)')
    # plt.ylabel('Number of Stars')
    # plt.title('TBORN')
    # plt.show()

    # # Plot Main Sequence Luminosity vs T_eff
    # plt.scatter(t_eff, lum_ms, color='#00008B')

    # # Plot White Dwarf Luminosity vs T_eff_wd
    # plt.scatter(t_eff_wd, lum_wd, color='#FFA500')
    # plt.yscale('log')
    # plt.xlabel('T_eff')
    # plt.ylabel('Luminosity')
    # plt.title('Plot of Temperature vs Luminosity')
    # plt.gca().invert_xaxis()
    # plt.show()

    # # Plot White Dwarf / Main Sequence Luminosity vs T_eff_wd LOG-LOG
    # plt.scatter(t_eff, lum_ms, color='#00008B')
    # plt.scatter(t_eff_wd, lum_wd, color='#FFA500')
    # plt.yscale('log')
    # plt.xscale('log')
    # plt.xlabel('T_eff')
    # plt.ylabel('Luminosity')
    # plt.title('Plot of Temperature vs Luminosity')
    # plt.gca().invert_xaxis()
    # plt.show()

    # # Plot with 10% Gaussian Noise
    # plt.scatter(temp_noisy, lum_noisy, s=2, color='#00008B')
    # plt.scatter(temp_wd_noisy, lum_wd_noisy, s=2, color='#FFA500')
    # plt.yscale('log')
    # plt.xlabel('T_eff')
    # plt.ylabel('Luminosity')
    # plt.title('Plot of Temperature vs Luminosity')
    # plt.gca().invert_xaxis()
    # plt.show()

    # # Plot with 10% Gaussian Noise LOG-LOG
    # plt.scatter(temp_noisy, lum_noisy, s= 2, color='#00008B')
    # plt.scatter(temp_wd_noisy, lum_wd_noisy, s = 2, color='#FFA500')
    # plt.yscale('log')
    # plt.xscale('log')
    # plt.xlabel('T_eff')
    # plt.ylabel('Luminosity')
    # plt.title('Plot of Temperature vs Luminosity')
    # plt.gca().invert_xaxis()
    # plt.show()

#-----------------------------------------------------------
# Combined Plots
#-----------------------------------------------------------

# Plot Main Sequence and White Dwarfs for all ages (normal scale)
plt.figure(figsize=(10, 6))
colors = ['#00008B', '#FF4500', '#008000']  # Colors for ages
for i, age in enumerate(cluster_ages):
    plt.scatter(all_t_eff[i], all_lum_ms[i], s=20, color=colors[i], label=f'MS,
```

```
     Age={age} Gyr')
    plt.scatter(all_t_eff_wd[i], all_lum_wd[i], s=20, color=colors[i], alpha
    =0.5, label=f'WD, Age={age} Gyr')

plt.yscale('log')
plt.xlabel('T_eff (K)')
plt.ylabel('Luminosity (L_sun)')
plt.title('Temperature vs Luminosity (Normal Scale)')
plt.gca().invert_xaxis()
plt.legend()
plt.show()

# Plot Main Sequence and White Dwarfs for all ages (log-log scale)
plt.figure(figsize=(10, 6))
for i, age in enumerate(cluster_ages):
    plt.scatter(all_t_eff[i], all_lum_ms[i], s=10, color=colors[i], label=f'MS,
     Age={age} Gyr')
    plt.scatter(all_t_eff_wd[i], all_lum_wd[i], s=10, color=colors[i], alpha
    =0.5, label=f'WD, Age={age} Gyr')
plt.xscale('log')
plt.yscale('log')
plt.xlabel('T_eff (K)')
plt.ylabel('Luminosity (L_sun)')
plt.title('Temperature vs Luminosity (Log-Log Scale)')
plt.gca().invert_xaxis()
plt.legend()
plt.show()

# Plot Gaussian Noise for all ages (normal scale)
plt.figure(figsize=(10, 6))
for i, age in enumerate(cluster_ages):
    # Create Gaussian Noise Main Sequence
    lum_noisy = all_lum_ms[i] * (1 + np.random.normal(0, 0.1, size=len(
    all_lum_ms[i])))
    temp_noisy = all_t_eff[i] * (1 + np.random.normal(0, 0.1, size=len(
    all_t_eff[i])))

    # Create Gaussian Noise White Dwarf
    lum_wd_noisy = all_lum_wd[i] * (1 + np.random.normal(0, 0.1, size=len(
    all_lum_wd[i])))
    temp_wd_noisy = all_t_eff_wd[i] * (1 + np.random.normal(0, 0.1, size=len(
    all_t_eff_wd[i])))

    # Plot Main Sequence and White Dwarfs with noisy data
    plt.scatter(temp_noisy, lum_noisy, s=10, color=colors[i], label=f'MS, Age={
    age} Gyr')
    plt.scatter(temp_wd_noisy, lum_wd_noisy, s=10, color=colors[i], alpha=0.5,
    label=f'WD, Age={age} Gyr')

plt.yscale('log')
plt.xlabel('T_eff (K)')
plt.ylabel('Luminosity (L_sun)')
plt.title('Temperature vs Luminosity (Normal Scale)\nGaussian Noise 10%')
plt.gca().invert_xaxis()
plt.legend()
plt.show()

# Plot Gaussian Noise for all ages (log-log scale)
plt.figure(figsize=(10, 6))
for i, age in enumerate(cluster_ages):
    # Create Gaussian Noise for Main Sequence
    lum_noisy = all_lum_ms[i] * (1 + np.random.normal(0, 0.1, size=len(
    all_lum_ms[i])))
```

```python
    temp_noisy = all_t_eff[i] * (1 + np.random.normal(0, 0.1, size=len(
    all_t_eff[i])))

    # Create Gaussian Noise for White Dwarf
    lum_wd_noisy = all_lum_wd[i] * (1 + np.random.normal(0, 0.1, size=len(
    all_lum_wd[i])))
    temp_wd_noisy = all_t_eff_wd[i] * (1 + np.random.normal(0, 0.1, size=len(
    all_t_eff_wd[i])))

    # Plot Main Sequence and White Dwarfs with gaussian noise
    plt.scatter(temp_noisy, lum_noisy, s=10, color=colors[i], label=f'MS, Age={
    age} Gyr')
    plt.scatter(temp_wd_noisy, lum_wd_noisy, s=10, color=colors[i], alpha=0.5,
    label=f'WD, Age={age} Gyr')

plt.xscale('log')
plt.yscale('log')
plt.xlabel('T_eff (K)')
plt.ylabel('Luminosity (L_sun)')
plt.title('Temperature vs Luminosity (Log-Log Scale)\nGaussian Noise 10%')
plt.gca().invert_xaxis()
plt.legend()
plt.show()


#---------------------------------------------
# Adding 2d HR Diagram
#---------------------------------------------


# Combine Main Sequence and White Dwarf Data
t_eff_combined = np.concatenate(all_t_eff)
lum_combined = np.concatenate(all_lum_ms)
t_eff_combined_wd = np.concatenate(all_t_eff_wd)
lum_combined_wd = np.concatenate(all_lum_wd)

plt.figure(figsize=(10, 6))

# Create a KDE plot (Main Sequence + White Dwarfs)
sns.kdeplot(x=t_eff_combined, y=lum_combined, cmap='inferno', fill=True, thresh
    =0, levels=30, label='Main Sequence')
sns.kdeplot(x=t_eff_combined_wd, y=lum_combined_wd, cmap='coolwarm', fill=True,
     thresh=0, levels=30, label='White Dwarfs')

plt.xlabel('Effective Temperature (T_eff) [K]')
plt.ylabel('Luminosity (L/L_sun)')
plt.title('2D KDE Density Map of the HR Diagram')
plt.xscale('log')
plt.yscale('log')
plt.gca().invert_xaxis()
plt.legend()
plt.show()
```

Listing 1: Python Code for Stellar Population Simulation

```python
import numpy as np
import matplotlib.pyplot as plt

# Constants
np.random.seed(9)

N = 1000            # Number of stars
dt = 1              # Timestep
alpha = 2.35        # IMF constant
```

```
max_mass = 100      # Maximum star mass
sigma = 5.67e-8     # Stefan-Boltzmann constant (W/m^2 K^4)
C = 0.0101          # White Dwarf constant
R_solar = 6.969e8   # Solar radius (m)
L_solar = 3.827e26 # Solar luminosity (W)

# Cluster ages and minimum mass values
cluster_ages = [8, 10, 12]
min_mass_values = [0.1, 0.3, 0.5] # Chosen as solar masses

# Main code but split into definitions to simplify
def generate_masses(min_mass, max_mass, alpha, N):
    masses = []
    while len(masses) < N:
        rand_y = np.random.uniform(0, 1)
        rand_mass = np.random.uniform(min_mass, max_mass)
        mass_i = (rand_mass) ** (-alpha)
        if rand_y < mass_i:
            masses.append(rand_mass)
    return masses

def main_sequence_luminosity(mass):
    if mass > 55:
        return 32000 * mass
    elif 2 < mass <= 55:
        return 1.4 * mass ** 3.5
    elif 0.43 < mass <= 2:
        return mass ** 4
    elif mass <= 0.43:
        return 0.23 * mass ** 2.3

def main_sequence_radius(mass):
    if mass >= 1.12:
        return 10 ** 0.66 * np.log(mass) + 0.5
    else:
        return mass

def effective_temperature(lum, radius):
    return (lum * L_solar / (4 * np.pi * radius ** 2 * R_solar ** 2 * sigma))
    ** (1 / 4)

def calculate_msto(cluster_age, masses, t_ms):
    # MSTO is the biggest star that is still on main sequence
    for i, age in enumerate(t_ms):
        if cluster_age - age <= 0:
            return i, masses[i]
    return len(masses) - 1, masses[-1]  # Default to least massive star if no
    MSTO

# Create figure for combined plot
plt.figure(figsize=(10, 6))

# Loop through all min_mass values and ages, and plot the data (COMBINED)
colors = ['#00008B', '#FF4500', '#008000']
labels = ['min_mass = 0.1 M_sun', 'min_mass = 0.3 M_sun', 'min_mass = 0.5 M_sun
    ']

for idx, min_mass in enumerate(min_mass_values):
    print(f"Analyzing for min_mass = {min_mass} M_sun")

    all_lum_ms, all_t_eff, all_msto = [], [], []

    for age_idx, t_cluster in enumerate(cluster_ages):
```

```
        # Generate initial star parameters
        masses = generate_masses(min_mass, max_mass, alpha, N)
        t_ms = [10 * mass ** -3.5 for mass in masses]
        t_born = [dt * np.random.uniform(0, 1) for _ in range(N)]
        t_cool = [t_cluster - t_born[i] - t_ms[i] for i in range(N)]

        # Separate stars into MS and WD
        lum_ms, t_eff_ms = [], []
        for i, mass in enumerate(masses):
            if t_cool[i] <= 0:
                lum = main_sequence_luminosity(mass)
                radius = main_sequence_radius(mass)
                t_eff = effective_temperature(lum, radius)
                lum_ms.append(lum)
                t_eff_ms.append(t_eff)

        # Calculate MSTO
        msto_index, msto_mass = calculate_msto(t_cluster, masses, t_ms)
        msto_lum = main_sequence_luminosity(msto_mass)
        msto_radius = main_sequence_radius(msto_mass)
        msto_t_eff = effective_temperature(msto_lum, msto_radius)
        all_msto.append((msto_mass, msto_t_eff, msto_lum))

        # Append data
        all_lum_ms.append(lum_ms)
        all_t_eff.append(t_eff_ms)

        # Print MSTO details
        print(f"Age: {t_cluster} Gyr, MSTO Mass: {msto_mass:.2f} M_sun, "
              f"T_eff: {msto_t_eff:.2f} K, Luminosity: {msto_lum:.2f} L_sun")

    # Plot all MS stars and MSTO stars for this min_mass
    for i, age in enumerate(cluster_ages):
        plt.scatter(all_t_eff[i], all_lum_ms[i], s=10, color=colors[i], alpha=0.7, label=f'Age = {age} Gyr' if idx == 0 else "")
        msto_mass, msto_t_eff, msto_lum = all_msto[i]
        plt.scatter(msto_t_eff, msto_lum, s=100, edgecolor='black', color=colors[i], marker='x', label=f'MSTO (Age={age} Gyr, min_mass={min_mass} M_sun)' if idx == 0 else "")

# Plotsss
plt.xscale('log')
plt.yscale('log')
plt.xlabel('T_eff (K)')
plt.ylabel('Luminosity (L_sun)')
plt.title('Temperature vs Luminosity (Log-Log Scale) for Different min_mass Values')
plt.grid(True, which="both", linestyle="--", linewidth=0.5)
plt.gca().invert_xaxis()
plt.legend()
plt.show()
```

Listing 2: Python Code for Combined MSTO Analysis