FROM SPHERICAL BESSEL FUNCTION TO GRAVITATIONAL WAVES

For code in this document see my Github!

THE GOAL

- Deyan has modified a code originally touched on by;
  - Richard Easther and Will Kinney (https://arxiv.org/abs/astro-ph/0210345)
  - Arthur Kosowsky and Jerod Caliguiri (https://arxiv.org/abs/1409.3195)
- **The primary goal is to run this inflationary code potentially with a modified calculation of the tensor power spectrum.**
  - Previous tests have estimated that the WKB approximation may overestimate the amplitude of GWs, i.e. (Pritchard, Kaminiokowski - https://arxiv.org/abs/astro-ph/0412581)
  - *Should we find a better analytic approach we can enhance the computational power of the code to look for more things. Perhaps we can even consider the duration of reheating and how that impacts our spectrum.*

PHASE 1: TESTING WKB APPROXIMATION

In order to test the WKB analysis, we needed to do it on something we know the answer to which is the spherical Bessel function.

In general the WKB approximation generally is developed to solve second order DE like Bessel functions, Schrodinger, etc and gives a solution valid within certain regions.

These regions are often where the change in the wavelength/phase is small compared to the wave itself, basically the wave does not experience abrupt changes (it is an adiabatic where the wave changes slowly). *You will see that at turning points (x = x0 or w = 0) , the wavelength becomes very large and and to bridge solutions we may need to use something like an Airy function.*

WKB APPROXIMATION COMPONENTS

- Generally speaking, the WKB approximation works to solve:

$$y'' + Q(x)*y = 0$$

$$y'' = -Q(x)*y$$

We define the turning point to be where $Q(x) = 0$, I call this point x0. We can define regions where $Q(x) < 0$ or $Q(x) > 0$.

$Q(x)$ will be useful as well when we define an integral known as S0 = integral from 0 to x of the square root of Q.

SPHERICAL BESSEL WKB APPROXIMATION

The equation we are solving really is:

$$r^2\,y'' + 2\,r\,y' + y\,(r^2 - l(l+1)) = 0$$

(in this case the prime denotes derivative wrt r)

Divide by r^2

$$y'' + 2y'/r + y\,(1 - (l(l+1))/r\text{^}2) = 0$$

Should y(r) = u(r)/r ,

$$u'' + u\,(1 - (l(l+1)/r^2)) = 0$$

Given, ϵ^2 = 1/(l(l+1))

$$u'' = (1/(\epsilon^2 * r^2) - 1)u$$

$$\epsilon^2\ u'' = (1/r^2 - \epsilon^2)u$$

Assuming β^2 takes the place of l (used in Arthur's paper),

$$u'' = (1/(\epsilon^2 * r^2) - \beta^2)u$$

Or equivalently,

$$\epsilon^2\ u'' = (1/r^2 - \beta^2\epsilon^2)u$$

In flat universe, $\beta = sqrt[k^2 + K]\ becomes\ \beta = k$

$$\epsilon^2\ u'' = (1/r^2 - k^2\epsilon^2)u$$

Divide by k^2,

$$\frac{\epsilon^2}{k^2}u'' = (1/k^2r^2 - \epsilon^2)u$$

Now prime denotes derivative wrt to x, where $x = kr$ thus,

$$\epsilon^2\ u'' = \left(\frac{1}{x^2} - \epsilon^2\right)\ u$$

THIS IS NOW IN APPROPRIATE FORM FOR THE WKB APPROXIMATION!

SPHERICAL BESSEL WKB APPROXIMATION CONT

With the form from the previous slide we can expect a solution to the expression to look like so, equation 6 from Kosowksy https://arxiv.org/abs/astro-ph/9805173

$$y(x) \simeq 2\sqrt{\pi}C \left(\frac{3S_0(x)}{2\epsilon}\right)^{1/6} [Q(x)]^{-1/4} \operatorname{Ai}\left[\left(\frac{3S_0(x)}{2\epsilon}\right)^{2/3}\right],$$

$$S_0(x) = \int_0^x \sqrt{Q(t)}\, dt.$$

The expectation is that $j_l(kr) = j_l(x) = y(x)/x$

Remember here: x is a generic array, this may get harder to deal with in the gravitational case

SPHERICAL BESSEL WKB APPROXIMATION CONT

GIVEN $\epsilon^2 \, u'' = \left(\frac{1}{x^2} - \epsilon^2\right) \, u$

We can now distinguish a few things;

$Q(x) = \left(\frac{1}{x^2} - \epsilon^2\right)$ and $S_0(x) = \int_0^x \sqrt{Q(t)}dt$

Our turning point occurs when Q(x0) = 0, which I will refer to as x0.

$x0 = 1/\epsilon$

Our regions of interest are x < x0 and x > x0.

Where x spans for some semi-reasonable range, for us it may go from from 1e-3 to 15+2*l. Keep in mind that depending on how the value relates to the turning point we will see either exponential or oscillatory behavior.

We also have the constant $C = \sqrt{\epsilon/2}$

SPHERICAL BESSEL WKB APPROXIMATION CONT

In order to properly compare to a turning point at zero for an easier to read plot,

We will define a new variable that will effectively **make our turning point be at zero**. In order for this to happen, use the following: $w = x0 - x$ this adjust the next few values such that now,

$Q(w) = \left(\frac{1}{(x0-x)^2} - \epsilon^2\right)$ and $S_0(w) = \int_0^w \sqrt{Q(t)}dt$

Our turning point occurs when Q(w=0) = 0.

Our regions of interest are w < 0 and w > 0. All that has occurred was a minor shift.

We still have the constant $C = \sqrt{\epsilon/2}$, and $x0 = 1/\epsilon$.

PYTHON SPHERICAL BESSEL FUNCTION VIA WKB APPROXIMATION

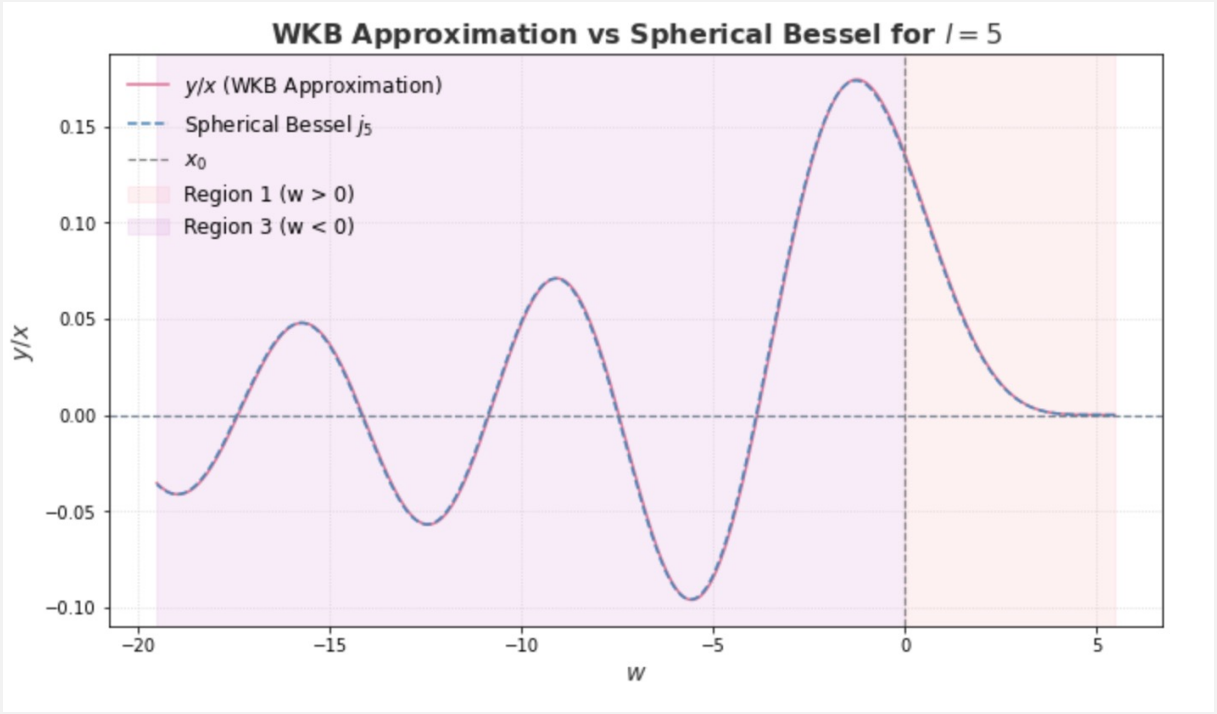
```python
from scipy.integrate import quad
import numpy as np
from scipy.special import airy, spherical_jn
import matplotlib.pyplot as plt

def plot_wkb_approximation(l):

    # Calculate epsilon, x0, and constant based on l
    epsilon = 1 / np.sqrt(l * (l + 1))
    x0 = 1 / epsilon  <- Turning Point
    constant = np.sqrt(epsilon) / 2<- Constant

    # Define x_array and corresponding w_array
    x_array = np.linspace(1e-3, 15 + 2 * l, 1000)  <- Defining x and w
    w_array = x0 - x_array  # w = x0 - x
    #print('w_array:', w_array)
    #print('w_array min:', np.amin(w_array), 'w_array max:', np.amax(w_array))


    # Define Q(w) function using x = x0 - w
    Q_func = lambda w: (1 / (x0 - w)**2) - epsilon**2  <- Defing Q(w)

    # Calculate S0 using adaptive integration
    S0 = np.array([quad(lambda w: np.sqrt(np.abs(Q_func(w))), 0, wi)[0] for wi in w_array])

                                                                <- Numerical
    # Calculate expr and Q_vals, preserving signs               Integration for S0
    expr = (3 * S0) / (2 * epsilon)
    Q_vals = Q_func(w_array)
    Q_factor = np.sign(Q_vals) * (np.abs(Q_vals) ** (-1/4))

    # Calculate the general solution y using signed powers
    y_general = 2 * np.sqrt(np.pi) * constant * (np.sign(expr) * np.abs(expr) ** (1/6)) \
                * Q_factor * airy(np.sign(expr) * np.abs(expr) ** (2/3))[0]  <- General Solution
```

PYTHON SPHERICAL BESSEL FUNCTION VIA WKB APPROXIMATION

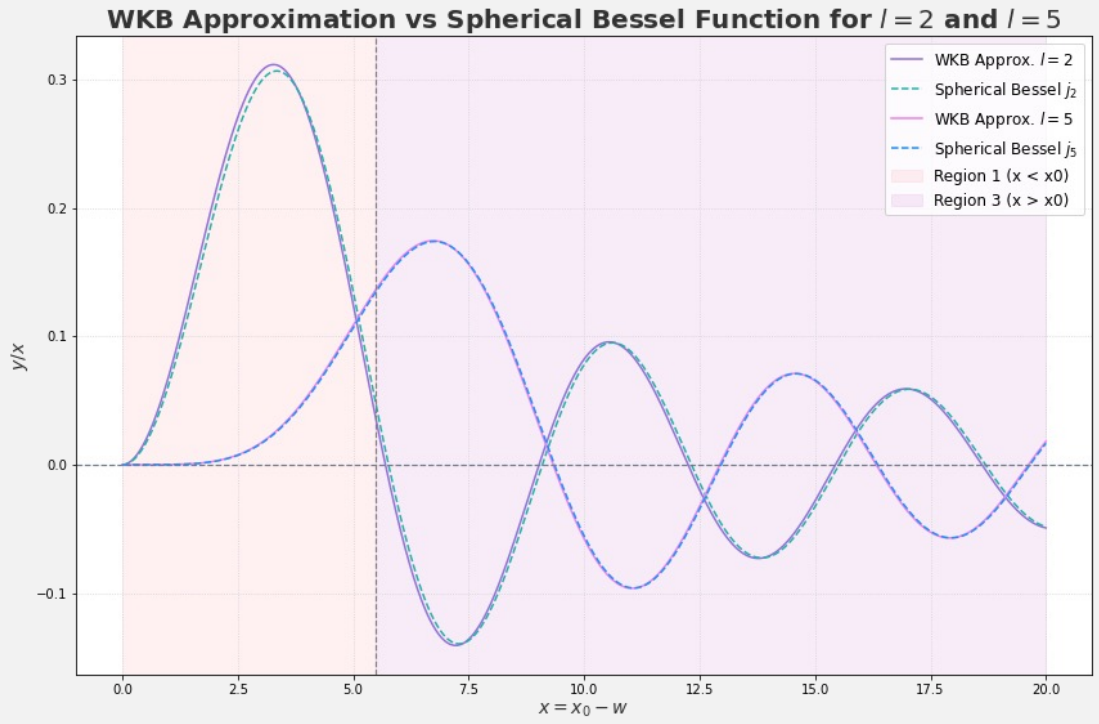
Rest of this is just plotting and calling the function

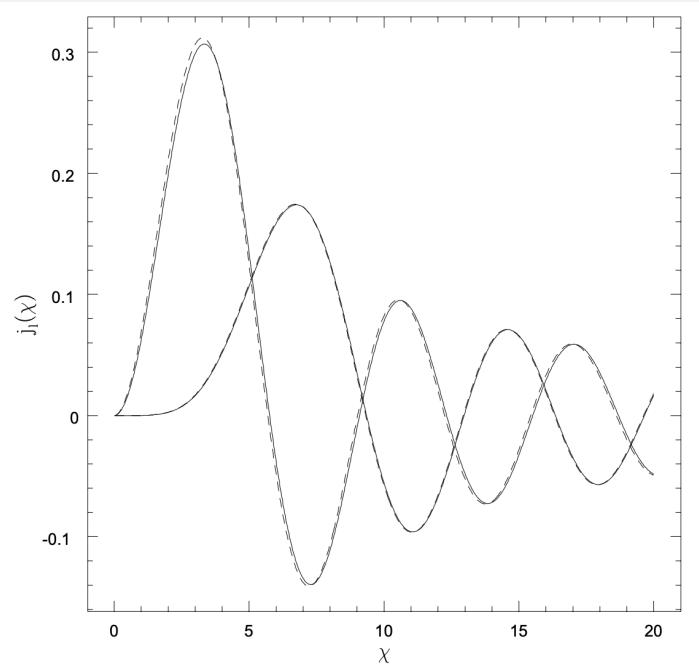
```
    t y/x vs spherical Bessel function for comparison
    spherical_bessel = spherical_jn(l, x_array)
    plt.figure(figsize=(10, 6))

    # Define colors for the WKB and Bessel function
    wkb_color = '#DB7093'  # Vibrant rose for WKB approximation
    bessel_color = '#4682B4'  # Soft teal for the Bessel function
    y_general
    # Background shading for Regions 1 and 3
    plt.axvspan(0, max(w_array), color='#FFB6C1', alpha=0.2, label='Region 1 (w > 0)')
    plt.axvspan(min(w_array), 0, color='#DDA0DD', alpha=0.2, label='Region 3 (w < 0)')

    # Plot the WKB Approximation and Bessel function
    plt.plot(w_array, y_general / x_array, label=r'$y/x$ (WKB Approximation)', color=wkb_color, linewidth=1.5)
    plt.plot(w_array, spherical_bessel, label=f'Spherical Bessel $j_{l}$', color=bessel_color, linewidth=1.5, li

    # Titles and labels with enhanced, stylish colors
    plt.title(f'WKB Approximation vs Spherical Bessel for $l = {l}$', fontsize=16, weight='bold', color='#333333
    plt.xlabel('$w$', fontsize=14, color='#333333')
    plt.ylabel(r'$y/x$', fontsize=14, color='#333333')

    # Adding grid, legend, and axis enhancements
    plt.axvline(x=0, linestyle='--', color='gray', linewidth=1.2, label='$x_0$')
    plt.axhline(y=0, linestyle='--', color='slategray', linewidth=1.2)
    plt.grid(color='lightgrey', linestyle=':', linewidth=0.8)
    plt.legend(fontsize=12, loc='best', frameon=False)

    # Enhance plot aesthetics
    plt.tight_layout(pad=2)
    plt.savefig('wkb_l5_w.png')
    plt.show()

# Example usage for l = 5
plot_wkb_approximation(5)
```

PYTHON SPHERICAL BESSEL FUNCTION VIA WKB APPROXIMATION



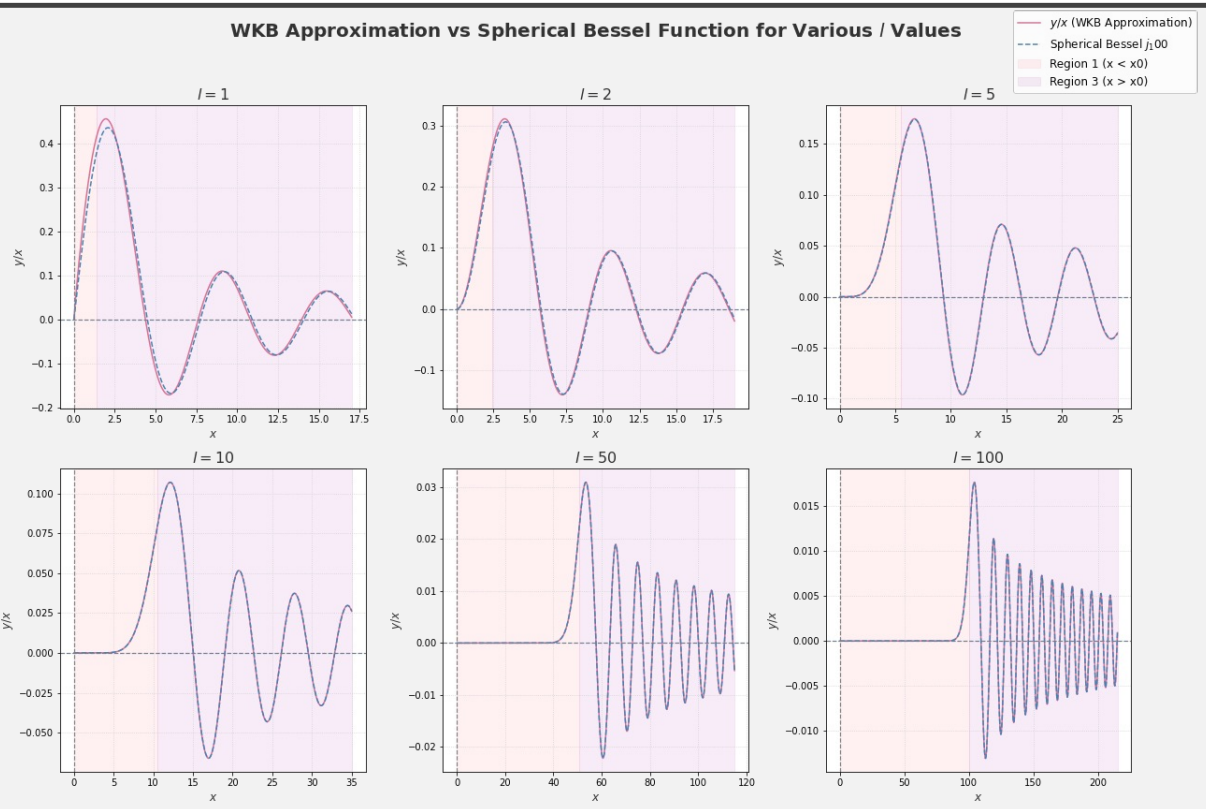
PYTHON SPHERICAL BESSEL FUNCTION VIA WKB APPROXIMATION





PYTHON SPHERICAL BESSEL FUNCTION VIA WKB APPROXIMATION

PYTHON SPHERICAL BESSEL FUNCTION VIA WKB APPROXIMATION



PHASE 2: TESTING GW-WKB APPROXIMATION

In order to test the GW-WKB analysis, we need to possibly compare it to something that has been somewhat done before. Pritchard and Kaminokowski have generic forms for a radiation only and matter only universe using a WKB approximation (https://arxiv.org/abs/astro-ph/0412581). They claim in equations 25 and 26 that without the source term from the anisotropic stress (mostly relevant in a radiation dominated epoch):

$$h_{\rm rad}(\tau) = h(0) j_0(k\tau) = h(0)\frac{\sin k\tau}{k\tau},$$

$$h_{\rm mat}(\tau) = 3h(0)\frac{j_1(k\tau)}{k\tau}.$$

It is a useful idea to try and match this using our WKB analysis method applied to the case of gravitational waves.

GW MATHEMATICAL LANDSCAPE

Let's first start out with the following:

$$u(\tau)'' + u(\tau) * \left(k^2 - \frac{a(\tau)''}{a(\tau)}\right) = 0$$

If we can assume then that $x = k\tau$, such is the case that $dx = k\, d\tau$ and thus $\frac{d}{d\tau}\, k = \frac{d}{dx}$ such that $\frac{d^2u}{d\tau^2} = k^2\frac{d^2u}{dx^2}$ and $\frac{d^2a}{d\tau^2} = k^2\frac{d^2a}{dx^2}$ meaning that our new expression appears as:

$$u(x)'' + u(x) * \left(1 - \frac{a(x)''}{a(x)}\right) = 0$$

Most of our cosmology piece will be embedded into the $\frac{a(x)''}{a(x)}$ term, and we solve for our turning point using $Q(x) = -1 + \frac{a(x)''}{a(x)} = 0$

GW MATHEMATICAL LANDSCAPE CONT

The cosmology term, $\frac{a(x)''}{a(x)} = \frac{\dot{a}^2 a + \ddot{a} a}{k^2}$

Where our cosmology tells us that, $\dot{a}^2 a + \ddot{a} a = \frac{4\pi G\,(\rho - 3P) a^2}{3}$ and our term $(\rho - 3P)$ is dictated by the cosmological epoch.

During radiation domination: $\rho - 3P = 0 \; and \; so \frac{a(x)''}{a(x)} = 0$

This means $Q_{rad}(x) = -1 + \frac{a(x)''}{a(x)}$ OR a0 exists for $1 = \frac{a(x)''}{a(x)} = 0$

During matter domination: $\rho - 3P = \frac{\rho_{m0}}{a^3} \; and \; so \frac{a(x)''}{a(x)} = \frac{H_0^2}{2k^2} \frac{\Omega_m}{a}$

$Q_{matter}(x) = -1 + \frac{a(x)''}{a(x)}$ OR a0 exists for $\frac{H_0^2}{2k^2} \Omega_m = a$

At present day: $\rho - 3P = \rho_{m0} + 4\rho_{\Lambda 0} \; and \; so \frac{a(x)''}{a(x)} = \frac{H_0^2}{2k^2} \left[\frac{\Omega_{m0}}{a} + 4\Omega_{\Lambda 0} a^2\right]$

$Q_{present}(x) = -1 + \frac{a(x)''}{a(x)}$ OR a0 exists for $1 = \frac{a(x)''}{a(x)} = \frac{H_0^2}{2k^2} \left[\frac{\Omega_{m0}}{a} + 4\Omega_{\Lambda 0} a^2\right]$

As you can see each cosmological epoch should have a corresponding turning point (a0) for a given H0/k ratio.

COMPONENTS OF GW-WKB ANALYSIS

- As you can see key components for us in this analysis ask the question of what cosmological epoch we are in AND what is our ratio of H0/k.
  - This ratio tells us the scale of the gravitational mode relative to the Hubble horizon.
    - A small H0/k corresponds to large k or short wavelengths. These modes would have been within the size of the horizon sooner.
    - A large ratio corresponds to short k or a large wavelength. These modes may have come within the horizon size at a later time and have been affected differently by cosmic history.
- Another essential component comes from how we choose to describe our cosmic scale factor a(x) = a(kτ). This becomes important because in order to solve our gravitational wave equation we need to have $h(x = k\tau) = \frac{u(x)}{a(x)}$

Let's define a(x) and get back to the code that makes it possible for us to test out our GW-WKB analysis.

COSMIC SCALE FACTOR

To define our cosmic scale factor computationally there are a few steps that are useful for us to follow:

I. Define $t = \int_{a_{min}}^{a_{max}} \frac{1}{H(a)}\,\mathrm{da}$ where H = Hubble parameter during the radiation/matter/present epoch (includes H_0 in sec)

II. Generate a spline of a(t) evaluated from t_{min} and t_{max}

III. Define $a(\tau) = \int_{t_{min}}^{t_{max}} \frac{1}{a(t)}\,dt$

IV. This allows us to calculate an array of τ(t) which we can use to generate a spline of $a(\tau)$ evaluated from τ_{min} and τ_{max}

V. Finally, we can define a range of kτ values ranging from [kτ_{min} , kτ_{max}]

VI. Which ultimately, allows us to generate a spline of a(x = kτ) evaluated over our kτ range

I include code that shows an example of this calculation for a radiation dominated universe (keep in mind these are used in a larger function).

PYTHON CALCULATING COSMIC SCALE FACTOR

```python
#Define Hubble for the Radiation Only Era

def H_radiation(a):
    """Hubble parameter during the radiation-dominated era."""
    return H0_s * np.sqrt((omega_r / a**4) + (omega_k / a**2))

#DEFINE a(t): a_t_rad

# Define scale factor range
a_min = 1e-2
a_max = 1  # Up to a = 1
a_range_rad = np.logspace(np.log10(a_min), np.log10(a_max), num=500)
a_range_rad= a_range_rad

# Compute t(a) for the radiation-dominated era
t_a_rad = np.array([
    quad(lambda a: 1 / H_radiation(a), a_min, a, limit=1000)[0]
    for a in a_range_rad
])

# Create a spline for a(t)
spline_a_t_rad = UnivariateSpline(t_a_rad, a_range_rad, s=0)

# Define time range for evaluating the spline
t_range_rad = np.linspace(t_a_rad.min(), t_a_rad.max(), len(a_range_rad))
a_t_rad = spline_a_t_rad(t_range_rad)
```

PYTHON CALCULATING COSMIC SCALE FACTOR

```python
#DEFINE a(tau): a_tau

# Define the integral to compute conformal time tau
def compute_tau(spline_a_t, t_min, t_max):
    """Compute conformal time tau for a given spline a(t)."""
    result, _ = quad(lambda t: 1 / spline_a_t(t), t_min, t_max, limit=1000)
    return result

# Compute tau for the full radiation-dominated era
tau_t_rad = compute_tau(spline_a_t_rad, t_a_rad.min(), t_a_rad.max()) #computes one at a time

# Generate sampled tau values for t_range
tau_t_array_rad = np.array([compute_tau(spline_a_t_rad, t_a_rad.min(), t) for t in t_range_rad])
#print('max tau(t) value', np.amax(tau_t_array_rad))
tau_t_array_rad = tau_t_array_rad / 1e18


# Create a spline for a(tau)
spline_a_tau = UnivariateSpline(tau_t_array_rad, a_t_rad, s=0)

# Define a range of tau values for evaluation
tau_range = np.linspace(tau_t_array_rad.min(), tau_t_array_rad.max(), 500)
a_tau = spline_a_tau(tau_range)
```

PYTHON CALCULATING COSMIC SCALE FACTOR

```python
# DEFINE a(k*tau): a_k_tau

# Define the new spline a(ktau)
def compute_extended_a_k_tau(tau_array, spline_a_tau, k, num_points=500):

    # Extend tau range for the specific k
    tau_min_k = tau_array.min() * k
    tau_max_k = tau_array.max() * k
    ktau_range = np.linspace(tau_min_k, tau_max_k, num_points)

    # Create a spline for a(k * tau)
    spline_a_ktau = UnivariateSpline(k * tau_array, spline_a_tau(tau_array), s=0)

    # Evaluate a(k * tau) over the extended range
    a_k_tau = spline_a_ktau(ktau_range)

    return ktau_range, a_k_tau, spline_a_ktau


ktau_range, a_k_tau, spline_a_ktau = compute_extended_a_k_tau(tau_t_array_rad, spline_a_tau, k)
```

COMPARING H

- For comparison purposes to the aforementioned Pritchard and Kaminiokowski paper we can assume k =1 for now, this should be sufficient to compare to their results. Keep in mind their initial plots look at conformal time, τ, and h(τ).
- They do use a slightly different form for their WKB approximation.
- Note their division by $\tau^{1/2}(\tau+2)$ and they include the second Airy function

$$h(\tau) = \frac{\Gamma(k\tau)^{-1/4}}{\tau^{1/2}(\tau+2)} \left(\frac{3}{2}S_0(\tau)\right)^{1/6} \times \left\{2\sqrt{\pi}C_2 \mathrm{Ai}\left[\left(\frac{3}{2}\mathrm{S}_0(\tau)\right)^{2/3}\right] + \sqrt{\pi}\mathrm{C}_1\mathrm{Bi}\left[\left(\frac{3}{2}\mathrm{S}_0(\tau)\right)^{2/3}\right]\right\}, \quad (30)$$

with

$$\Gamma(s) = \frac{1}{4} + \frac{2s}{s+2k} - s^2, \quad (31)$$

and

$$S_0(\tau) = \int_{k\tau}^{k\tau_T} \sqrt{\Gamma(s)}\,\frac{ds}{s}. \quad (32)$$

Here, τ_T is the solution to $\Gamma(k\tau) = 0$, Ai and Bi are Airy functions, and C_1 and C_2 are constant coefficients set by the boundary conditions $h(0) = 1$ and $\dot{h}(0) = 0$. For technical reasons, these boundary conditions must be extrapolated to small τ via asymptotic approximation to Eq. (23) and then applied. Care must be taken in evaluating the above expressions when $\tau > \tau_T$. These details are discussed further in Appendix B. This WKB expression reproduces the phase of h in both radiation and matter dominated regimes, but underestimates the amplitude.

NEXT STEPS

- Fix radiation only and matter only cases
- Apply this WKB analysis to a present day universe
  - This will involve choosing a computational way of solving for a0, possibly Brent's method or the Newton-Raphson method.
- Compare this method for determining h with that which is presented in a set of models given by Deyan's code –Ran some of these models already (2 at a time) on my machine
  - Basically, do I see the same results within some amount of error?
- Maybe later: If that does in fact check out, can I incorporate a phase of reheating that is effectively a function of time and does that have an effect on my GW amplitude?