In [1]:
```
import pylab
import h5py
import math
import array
from numpy import *
import numpy as np
from pycbc.types import TimeSeries, FrequencySeries
from pycbc.waveform import get_td_waveform, get_fd_waveform
from pycbc.waveform.waveform_modes import get_td_waveform_modes
from pycbc import types, fft, waveform
import lal
from scipy import interpolate
from scipy.interpolate import interp1d
from lal import MSUN_SI, MTSUN_SI, G_SI, PC_SI, C_SI, PI
from pycbc.filter import match
from pycbc.psd import aLIGOZeroDetHighPower
from tqdm import tqdm#

#from matplotlib import rcParams
#rcParams.update({'figure.autolayout': True})
import matplotlib as mpl
#mpl.rcParams['figure.dpi'] = 200
#plt.rcParams["font.family"] = "monospace"

from matplotlib import gridspec
from matplotlib import ticker

import matplotlib.pyplot as plt
#plt.style.reload_library()
#plt.style.use(['science','notebook'])



#pylab.rc('xtick', labelsize=18)
#pylab.rc('ytick', labelsize=18)
#pylab.rc('axes', labelsize=16)
#pylab.rc('legend', fontsize=15)
```

Eccentricity $e(\xi_\phi)$

In [2]:
```
# Eq. (4.17a, 4.17b), Pg. 18, Moore et al (2016)

def epsilon(xi, eta):
    return(( 1 + ( ( -2833/2016 + 197/72 * eta ) * ( xi )**( 2/3 ) +
                  ( -377/144 * np.pi * xi + ( ( 77006005/24385536 + ( -11
         43807/10368 * ( eta )**( 2 ) ) ) * ( xi )**( 4/3 ) + ( np.pi * (
         -202589/362880 * eta ) * ( xi )**( 5/3 ) + ( xi )**( 2 ) * ( -33
         ( 3317/252 * EulerGamma + ( 180721/41472 * ( np.pi )**( 2 ) + (
         3977/2304 * ( np.pi )**( 2 ) ) * eta + ( -359037739/20901888 * (
      ( 10647791/2239488 * ( eta )**( 3 ) + ( -87419/3780 * np.log( 2 ) +
         ( 26001/1120 * np.log( 3 ) + 3317/504 * np.log( 16 * ( xi )**( 2
```

Fit Files

In [3]:

```
# "Hinder+ modified all 20 simulations SEOBNRv4 model, full frequency ran
g=open('tshift_H+modified_20hyb_Feb16.txt',"r")
lines=g.readlines()
A=[]
for x in lines:
    A.append(float(x.split()[1]))
g.close()

def tshift_Hinsp(q,e,l):
    return A[0] + A[1]*q + A[2]*q**2 + A[3]*e + A[4]*e**2 + A[5]*e**3 + A

g=open('tamp_H+modified_20hyb_Feb16.txt',"r")
lines=g.readlines()
B=[]
for x in lines:
    B.append(float(x.split()[1]))
g.close()

def tamp_Hinsp(eta,e,l):
    return B[0] + B[1]*eta + B[2]*eta**2 + B[3]*e + B[4]*e**2 + B[5]*e**3

g=open('tfreq_H+modified_20hyb_Feb16.txt',"r")
lines=g.readlines()
C=[]
for x in lines:
    C.append(float(x.split()[1]))
g.close()

def tfreq_Hinsp(eta,e,l):
    return C[0] + C[1]*eta + C[2]*eta**2 + C[3]*e + C[4]*e**2 + C[5]*e**3
```

Spherical Harmonics

In [4]:
```
def sph_harmonics(inc,ell):
    L=ell
    #inc = 10
    theta = inc
    for l in range(L,L+1):

        for m in range(-l,l+1):
            dlm = 0;
            k1 = max([0, m-2]);
            k2 = min([l+m, l-2]);

            #if(m==l or m==l-1):
            for k in range(k1,k2+1):
                A = []; B = []; cosTerm = []; sinTerm = []; dlmTmp = [];

                A = (-1)**k*math.sqrt(math.factorial(l+m)*math.factorial(
                B = math.factorial(k)*math.factorial(k-m+2)*math.factoria

                cosTerm = pow(math.cos(theta/2), 2*l+m-2*k-2);
                sinTerm = pow(math.sin(theta/2), 2*k-m+2);

                dlmTmp = (A/B)*cosTerm*sinTerm;
                dlm = dlm+dlmTmp

            Ylm = math.sqrt((2*l+1)/(4*math.pi))*dlm
            #print('l:',l,'m:',m,'\t Y_lm:',Ylm)
            if m==ell:
                #globals()['sph' + str(l) + str(m)] = Ylm
                #print('l:',l,'m:',m,'\t Y_lm:',Ylm)
                sphlm = Ylm
            elif m==-ell:
                #globals()['sph' + str(l) + '_' + str(abs(m))] = Ylm
                #print('l:',l,'m:',m,'\t Y_lm:',Ylm)
                sphl_m = Ylm
            else:
                continue
    return sphlm, sphl_m
```

In [5]:
```
def xi(x):
    return x**(3/2)

def xconv(f,M):
    return (PI*M*MTSUN_SI*f)**(2/3)  #22 mode conversion

def fconv(x,M):
    return x**(3/2)/(PI*M*MTSUN_SI)  #22 mode conversion
```

In [6]:
```
def eccmodel(Mass,q0,e0,l0,fmin,inclination=0,d=1,delta_t=1./4096,modes=[

    #delta_t=0.000152089115201025l8
    #delta_t = 1/2**20
    ell = []
    numrows = len(modes)
    numcols = len(modes[0])
    for i in range(0,numrows):
        l = modes[i][0]
        m = modes[i][1]
        ell.append(l)
    angle = inclination
    waveform = {}
    count = 0
    el = 2
    if el in ell:
        mode_data = {}
        mode_data['hp'], mode_data['hc'], mode_data['t'] = MODELECC22(Mas
        waveform['l2_m2'] = mode_data
        count = count + 1

    el = 3
    if el in ell:
        mode_data = {}
        mode_data['hp'], mode_data['hc'], mode_data['t'] = MODEL33(Mass,q
        waveform['l3_m3'] = mode_data
        count = count + 1

    el = 4
    if el in ell:
        mode_data = {}
        mode_data['hp'], mode_data['hc'], mode_data['t'] = MODEL44(Mass,q
        waveform['l4_m4'] = mode_data
        count = count + 1

    el = 5
    if el in ell:
        mode_data = {}
        mode_data['hp'], mode_data['hc'], mode_data['t'] = MODEL55(Mass,q
        waveform['l5_m5'] = mode_data
        count = count + 1

    len_max_mode = '0'
    len_max = 0
    for mode in waveform.keys():
        if len(waveform[mode]['t'])>len_max:
            len_max_mode = mode
            len_max = len(waveform[mode]['t'])

    for mode in waveform.keys():
        if mode != len_max_mode:
            waveform[mode]['hp'].resize(len_max)
            waveform[mode]['hc'].resize(len_max)

    hp=0
    hc=0
    time=waveform[len_max_mode]['t']
    for mode in waveform.keys():
        hp = hp + waveform[mode]['hp']
        hc = hc + waveform[mode]['hc']

    hplus = TimeSeries(hp,delta_t,epoch=time[0])
    hcross = TimeSeries(hc,delta_t,epoch=time[0])
```

```
    hcross = TimeSeries(hc,delta_t,epoch=time[0])

    return hplus, hcross
```

MODEL $(l, m) = (2, 2)$ (Using EccentricTD)

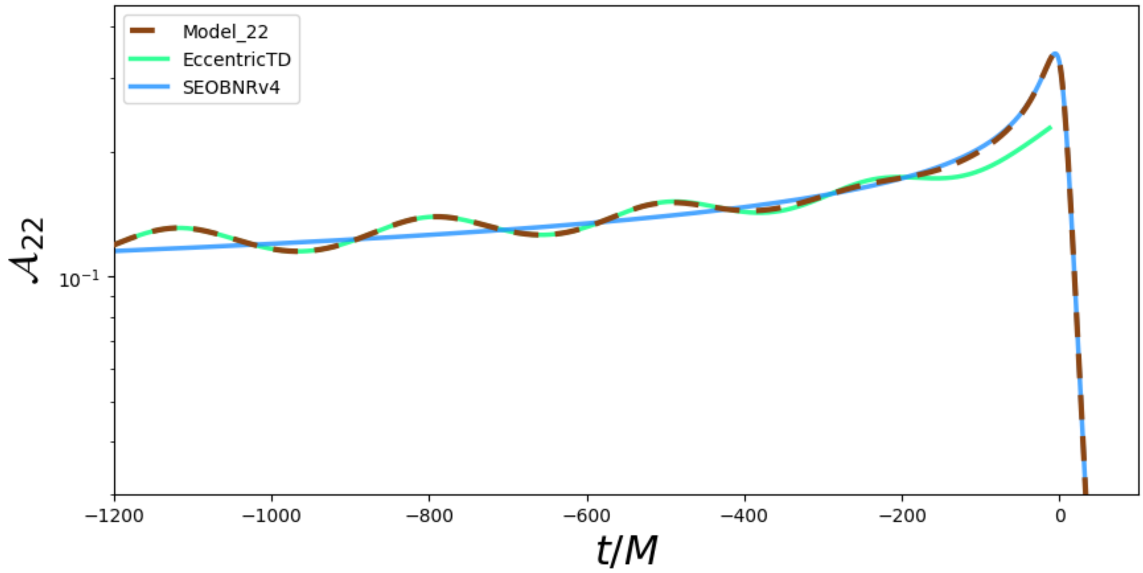
In [7]:
```
def MODELECC22(m,q0,e0,l0,fmin,angle,d,delta_t):

    #m=Total mass, q0= mass ratio, e0=initial orbital ecc, l0=mean anomal
    #angle= inclination, d= distance in Mpc, delta_t=sampling rate



    M=m
    M1=q0*M/(1+q0)
    M2=M/(1+q0)
    eta=q0/(1+q0)**2
    M_SI=M*MSUN_SI
    D_SI=(10**(6))*PC_SI
    mode2polfac=4*(5/(64*np.pi))**(1/2)

    hp, hc = get_td_waveform(approximant='EccentricTD', mass1=M1, mass2=M
    sp, sc = get_td_waveform(approximant='SEOBNRv4', mass1=M1, mass2=M2,

    tshift = -tshift_Hinsp(q0,e0,l0)*M*MTSUN_SI
    tmin = max(hp.sample_times[0]-tshift,sp.sample_times[0])

    # CIRCULAR IMR INTERP1D
    sp_intrp = interp1d(sp.sample_times,sp, kind='cubic',fill_value='extr
    sc_intrp = interp1d(sc.sample_times,sc, kind='cubic',fill_value='extr
    tImr_intrp=np.arange(tmin, sp.sample_times[-1], delta_t)
    sp_intrp=sp_intrp(tImr_intrp)
    sc_intrp=sc_intrp(tImr_intrp)
    tImr = tImr_intrp
    hpImr = sp_intrp
    hcImr = sc_intrp
    h22Imr=hpImr+1j*hcImr

    # ECCTD INTERP1D
    hp_intrp=interp1d(hp.sample_times-tshift, hp, kind='cubic',fill_value
    hc_intrp=interp1d(hc.sample_times-tshift, hc, kind='cubic',fill_value

    tEcc_intrp=np.arange(tmin,hp.sample_times[-1]-tshift, delta_t) #Need
    hp_intrp=hp_intrp(tEcc_intrp)
    hc_intrp=hc_intrp(tEcc_intrp)
    tEcc=tEcc_intrp
    hpEcc=hp_intrp
    hcEcc=hc_intrp
    h22Ecc=hpEcc+1j*hcEcc

    #Matching initial phase
    phaseEcc=np.unwrap(np.angle(h22Ecc)*2)/2
    phaseImr = np.unwrap(np.angle(h22Imr)*2)/2
    dphase = phaseEcc[0]-phaseImr[0]
    hp_new=real(h22Ecc*exp(-1j*dphase))
    hc_new=imag(h22Ecc*exp(-1j*dphase))

    phase_new=np.unwrap(np.angle(hp_new+1j*hc_new)*2)/2
    phaseEcc=phase_new
    h22Ecc_new=hp_new+1j*hc_new

    arg=np.argmin(abs(tEcc-tamp_Hinsp(eta,e0,l0)*M*MTSUN_SI))
    Idxjoin=arg

    t_amp=tEcc[Idxjoin] - 500*M*MTSUN_SI
    idxstr=np.argmin(abs(tEcc-t_amp))

    #Amplitude model
    amp=[]
    count=0
```

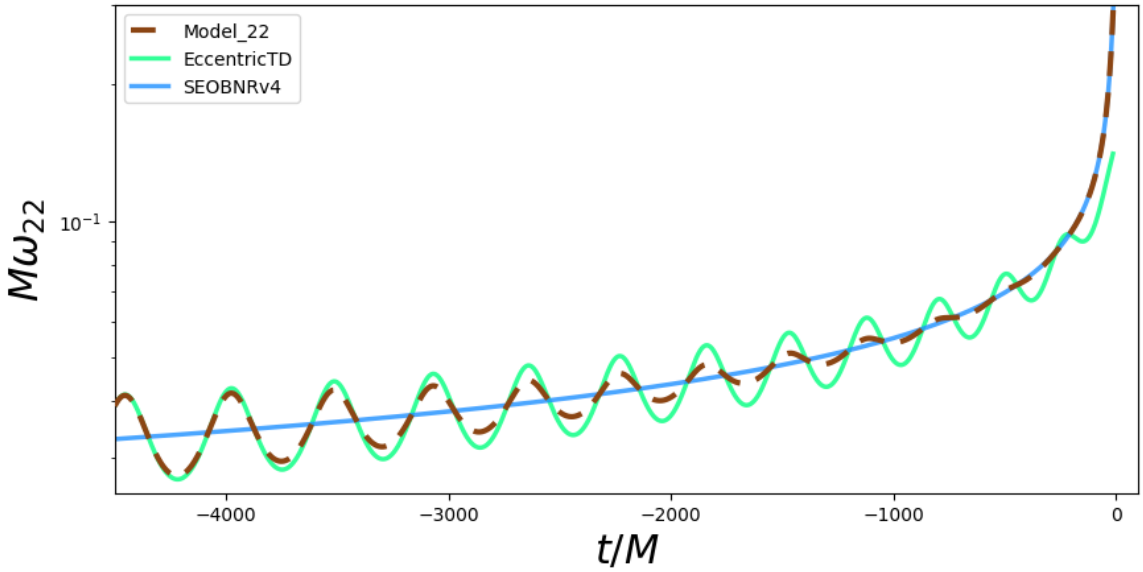
```
count=0
length=Idxjoin-idxstr

for i in range(idxstr,Idxjoin):
    amp.append(((length-count)*abs(h22Ecc_new[i])+count*abs(h22Imr[i]
    count=count+1

t_model=np.concatenate((tEcc[0:Idxjoin],tImr[Idxjoin:len(tImr)]))
h22amp=np.concatenate((abs(h22Ecc_new[0:idxstr]),amp)) #EDIT
h22amp_model=np.concatenate((h22amp,abs(h22Imr[Idxjoin:len(h22Imr)]))

omegaEcc=(M*MTSUN_SI/delta_t)*(np.gradient(phaseEcc))
omegaImr=(M*MTSUN_SI/delta_t)*(np.gradient(phaseImr))

tjoin0=tfreq_Hinsp(eta,e0,l0)
tjoin=tjoin0*M*MTSUN_SI
fjoin=np.argmin(abs(tEcc-tjoin))

#Frequency model
tstop = min(tEcc[-1],-30*M*MTSUN_SI)
lst = np.argmin(abs(tEcc-tstop))

indx = lst - fjoin
a0 = []
n = indx - 1
k = 0
for i in range(fjoin,fjoin+indx):
    a0.append(((n-k)*omegaEcc[i]+k*omegaImr[i])/n)
    k = k+1

f1 = np.concatenate((omegaEcc[0:fjoin],a0))
frequency_model = np.concatenate((f1,omegaImr[fjoin+indx:len(omegaImr
phase_f_model = np.cumsum(frequency_model)/(M*MTSUN_SI/delta_t) + PI/


hp_f_model = h22amp_model * np.cos(phase_f_model)
hc_f_model = h22amp_model * np.sin(phase_f_model)

ht = (mode2polfac/4)*(((1+math.cos(angle))**2*(hp_f_model-1j*hc_f_mod
hplus = np.real(ht)
hcross = np.imag(ht)

#return np.array(hplus), np.array(hcross), np.array(t_model)

print('Inclination = ',angle,' degrees')

#Plot
plt.figure(figsize=(10,4.8))
plt.plot(t_model/(M*MTSUN_SI), h22amp_model/(G_SI*M_SI/D_SI/C_SI/C_SI
plt.plot(tEcc/(M*MTSUN_SI),abs(h22Ecc_new)/(G_SI*M_SI/D_SI/C_SI/C_SI
plt.plot(tImr/(M*MTSUN_SI),abs(h22Imr)/(G_SI*M_SI/D_SI/C_SI/C_SI * mo
plt.xlim(xmin=-1200)
plt.xlim(xmax=100)
plt.ylim(ymax=4.5e-1)
plt.ylim(ymin=3e-2)
plt.ylabel(r'$\mathcal{A}_{22}$',fontsize=22,labelpad=5)
plt.xlabel(r'$t/M$',fontsize=22)
plt.yscale('log')
plt.legend()
#
plt.figure(figsize=(10,4.8))
plt.plot(t_model/(M*MTSUN_SI), frequency_model, color='saddlebrown',a
plt.plot(tEcc/(M*MTSUN_SI),omegaEcc,linestyle='-',linewidth=2.5,color
plt.plot(tImr/(M*MTSUN_SI),omegaImr,linestyle='-',linewidth=2.5,color
```

```
    plt.xlim(xmin=-4500)
    plt.xlim(xmax=100)
    plt.ylim(ymin=2.5e-2)
    plt.ylim(ymax=3e-1)
    plt.ylabel(r'$M\omega_{22}$',fontsize=22,labelpad=5)
    plt.xlabel(r'$t/M$',fontsize=22)
    plt.yscale('log')
    plt.legend()
```

In [8]:
```
MODELECC22(40,2,0.12,-0.181,20,0,1,1./4096)
```

```
Inclination =  0  degrees
```





EccentricTD Parameters

In [9]:
```
def PNparams(M,q,d,f_low,e0,delta_t):
    M2=M/(1+q)
    M1=M2*q
    hpVec_PN, hcVec_PN = get_td_waveform(approximant='EccentricTD', mass1
                                 delta_t=delta_t,
                                 f_lower=f_low,
                                 eccentricity=e0,
                                 distance=d)

    modetopolfac=4*(5/(64*np.pi))**(1/2)  #conversion factor between mode
                                          # check 0704.3764 equation 7 fo
    M_SI = M * MSUN_SI
    D_SI = 10**6 * PC_SI * d
    phase_EccTD = waveform.utils.phase_from_polarizations(hpVec_PN/(modet
    tVec_PN=hpVec_PN.sample_times/(M*MTSUN_SI)

    return phase_EccTD, tVec_PN
```

MODEL $(l, m) = (2, 2)$ (Using Ebersold Amplitudes)

In [10]:
```
def INSP_Eber22(M0,q,e0,l0,flow,inc,d0,delta_t):
    eta=neu=nu=q/(1+q)**2
    G=c=M=d=1
    M2=M/(1+q)
    M1=M2*q
    Delta=math.sqrt(1-(4*neu))
    eta=nu=neu
    gamma=EulerGamma=0.577215664901
    mode2polfac=4*(5/(64*np.pi))**(1/2)

    conv=M*MTSUN_SI
    M_SI=M * MSUN_SI
    D_SI=(10**(6)) * PC_SI * d

    xlow = ((M0*MTSUN_SI*math.pi*flow)**(2/3))
    f_low = (xlow**(3/2)/(M*MTSUN_SI*math.pi))

    %run GW_functions.ipynb

    x=xlow
    v=math.sqrt(x)

    xie=v**3

    if delta_t>=1/2**14:
        del_t = 1/2**14
    elif delta_t<1/2**14 and delta_t>=1/2**16:
        del_t = 1/2**16
    elif delta_t<1/2**16 and delta_t>=1/2**18:
        del_t = 1/2**18
    else:
        del_t = 1/2**20

    phase_EccTD, tVec_PN = PNparams(M,q,d,f_low,e0,del_t)

    tC_NR = 0

    x0=xlow
    xi0=x0**(3/2)
    v0=xi0**(1/3)

    theta=((5*M/(eta))**(1/8))*(tC_NR-tVec_PN)**(-1/8)
    theta0=((5*M/(eta))**(1/8))*(tC_NR-tVec_PN[0])**(-1/8)
    fVec=x_from_t(theta, theta0, e0, M, eta)

    plotIdx2=np.nonzero(fVec>=0)
    fVec=fVec[plotIdx2]
    xiVec=(np.pi*M*fVec)
    xVec=xiVec**(2/3)
    vVec=xiVec**(1/3)
    xband=np.where(xVec<=1/6)
    xVec = xVec[xband]
    maxPNidx = len(xVec)
    tVec_PN=tVec_PN[:maxPNidx]


    lp=2
    mp=2

    j=0
    h22=[]
    h2_2=[]
    for i in xVec:     #tqdm(xVec) for status bar
```

```
    for i in xvec:    #tqdm(xvec) for status bar
        v=math.sqrt(i)
        v0=math.sqrt(x0)
        xie=v**3
        xi0=v0**3
        l=mean_anomaly(xie, xi0, l0, eta, e0)
        e=e0*(xi0/xie)**(19/18)*epsilon(xie, eta)/epsilon(xi0, eta)
        psi=phase_EccTD[j]
        j=j+1
        xi=l   #use xi for amplitude (xie is being used for v**3)
        x=i
        h=amplitude_22(xi,x,nu,Delta,e) #### 22 mode requires additional

        hlm=8*math.sqrt(math.pi/5)*M*neu*i*h*((np.e)**(complex(0,-1)*mp*p
        hl_m=8*math.sqrt(math.pi/5)*M*neu*i*h*((np.e)**(complex(0,+1)*mp*

        h22.append(hlm)
        h2_2.append(hl_m)

    conv_t = M0*MTSUN_SI
    conv_h = G_SI*M0*MSUN_SI/(10**6 * PC_SI * d0)/C_SI/C_SI

    sph22, sph2_2 = sph_harmonics(inc,lp)

    h = np.multiply(h22,sph22)+np.multiply(h2_2,sph2_2)
    #h = np.array(h22)*((mode2polfac/4)*(1+math.cos(inc))**2)+np.array(h2
    hp=(np.real(h))#/(0.5*(1+(math.cos(inc))**2))
    hc=(np.imag(h))#/(math.cos(inc))
    time = tVec_PN - tVec_PN[-1]

    mode2polfac=4*(5/(64*np.pi))**(1/2)

    hp = np.array(hp) * conv_h
    hc = np.array(hc) * conv_h
    time = tVec_PN * conv_t

    hp_intrp = interp1d(time, hp, kind='cubic',fill_value='extrapolate')
    hc_intrp = interp1d(time, hc, kind='cubic',fill_value='extrapolate')
    t_intrp = np.arange(time[0], time[-1], delta_t)
    hp_intrp = hp_intrp(t_intrp)
    hc_intrp = hc_intrp(t_intrp)

    return np.array(hp_intrp), np.array(hc_intrp), np.array(t_intrp)
```

In [11]:

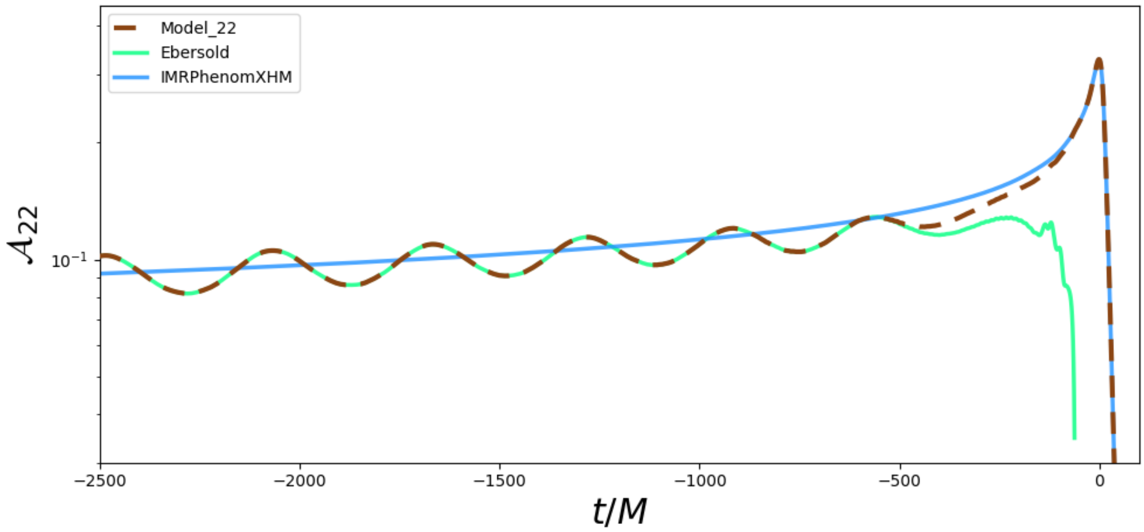
```
def MODEL22(m,q0,e0,l0,fmin,angle,d,delta_t):



    M=m
    M1=q0*M/(1+q0)
    M2=M/(1+q0)
    eta=q0/(1+q0)**2
    M_SI=M*MSUN_SI
    D_SI=(10**(6))*PC_SI
    mode2polfac=4*(5/(64*np.pi))**(1/2)

    hp, hc, tinsp = INSP_Eber22(M,q0,e0,l0,fmin,(np.pi/180)*angle,d,delta
    sp,sc = get_td_waveform(approximant='IMRPhenomXHM', mass1=M1, mass2=M

    tshift = -tshift_Hinsp(q0,e0,l0)*M*MTSUN_SI
    tmin = max(tinsp[0]-tshift, sp.sample_times[0])

    sp=sp#/(0.5*(1+(math.cos(angle))**2)) #inc check
    sc=sc#/(math.cos(angle))
```

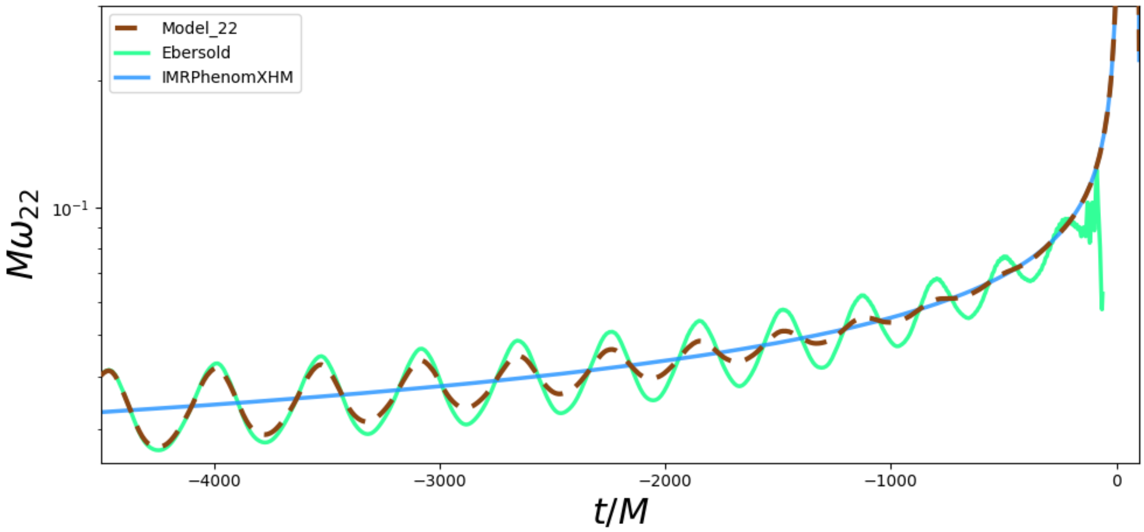
```
#Circular IMR
sp_intrp = interp1d(sp.sample_times, sp, kind='cubic', fill_value='ex
sc_intrp = interp1d(sc.sample_times, sc, kind='cubic', fill_value='ex
tImr_intrp = np.arange(tmin, sp.sample_times[-1], delta_t)
sp_intrp = sp_intrp(tImr_intrp)
sc_intrp = sc_intrp(tImr_intrp)
tImr = tImr_intrp
hpImr = sp_intrp
hcImr = sc_intrp
h22Imr = hpImr + 1j*hcImr


tshift = -tshift_Hinsp(q0,e0,l0)*M*MTSUN_SI


#Interpolation Ebersold
hp_intrp = interp1d(tinsp-tshift, hp, kind='cubic',fill_value='extrap
hc_intrp = interp1d(tinsp-tshift, hc, kind='cubic',fill_value='extrap
tEcc_intrp = np.arange(tmin, tinsp[-1]-tshift, delta_t)
hp_intrp = hp_intrp(tEcc_intrp)
hc_intrp = hc_intrp(tEcc_intrp)
tEcc = tEcc_intrp
hpEcc = hp_intrp
hcEcc = hc_intrp
h22Ecc = hpEcc + 1j*hcEcc

phaseEcc = np.unwrap(np.angle(h22Ecc)*2)/2
phaseImr = -np.unwrap(np.angle(h22Imr)*2)/2
dphase = phaseEcc[0] - phaseImr[0]
hp_new = real(h22Ecc * exp(-1j * dphase))
hc_new = imag(h22Ecc * exp(-1j * dphase))

phase_new = np.unwrap(np.angle(hp_new+1j*hc_new)*2)/2

#phaseEcc = phase_new
phaseEcc = abs(phase_new) #edit
phaseImr = abs(phaseImr) #edit
h22Ecc_new = (hp_new+1j*hc_new)

arg = np.argmin(abs(tEcc-tamp_Hinsp(eta,e0,l0)*M*MTSUN_SI))
Idxjoin = arg

t_amp = tEcc[Idxjoin] - 500*M*MTSUN_SI
idxstr = np.argmin(abs(tEcc-t_amp))

#Amplitude Model
amp=[]
count=0
length=Idxjoin-idxstr

for i in range(idxstr,Idxjoin):
    amp.append(((length-count)*abs(h22Ecc_new[i])+count*abs(h22Imr[i]
    count=count+1

t_model=np.concatenate((tEcc[0:Idxjoin],tImr[Idxjoin:len(tImr)]))
h22amp=np.concatenate((abs(h22Ecc_new[0:idxstr]),amp))
h22amp_model=np.concatenate((h22amp,abs(h22Imr[Idxjoin:len(h22Imr)]))

omegaEcc = (M*MTSUN_SI/delta_t)*(np.gradient(phaseEcc))
omegaImr = (M*MTSUN_SI/delta_t)*(np.gradient(phaseImr))

tjoin0 = tfreq_Hinsp(eta,e0,l0)
tjoin = tjoin0 * M * MTSUN_SI
```

```
    tjoin = tjoin0 * M * MTSUN_SI
    fjoin = np.argmin(abs(tEcc-tjoin))

    #frequency model
    tstop = min(tEcc[-1],-30*M*MTSUN_SI)
    lst=np.argmin(abs(tEcc-tstop))

    indx=lst - fjoin
    a0 = []
    n = indx-1
    k = 0
    for i in range(fjoin,fjoin+indx):
        a0.append(((n-k)*omegaEcc[i]+k*omegaImr[i])/n)
        k=k+1

    f1 = np.concatenate((omegaEcc[0:fjoin],a0))
    frequency_model = np.concatenate((f1,omegaImr[fjoin+indx:len(omegaImr
```

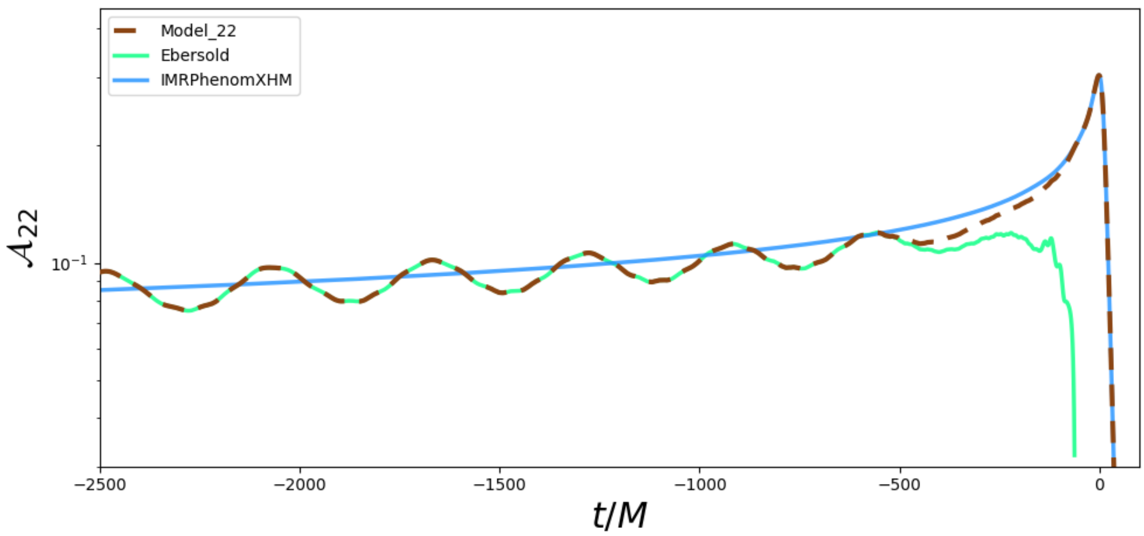
In [12]:

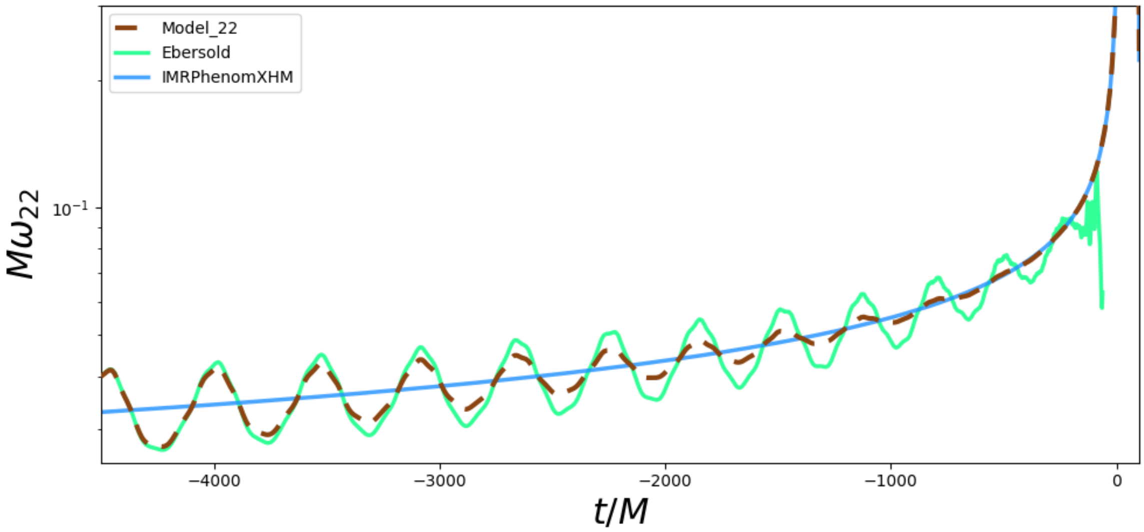
```
MODEL22(40,2,0.12,-0.181,20,10,1,1./4096)
MODEL22(40,2,0.12,-0.181,20,20,1,1./4096)
MODEL22(40,2,0.12,-0.181,20,30,1,1./4096)
    hc_f_model = h22amp_model * np.sin(phase_f_model)

    #return np.array(hp_f_model), np.array(hc_f_model), np.array(t_model)

    print('Mode = (2,2) , Inclination = ',angle,' degrees')
    #Plot
    plt.figure(figsize=(10,4.8))
    plt.plot(t_model/(M*MTSUN_SI),h22amp_model/(G_SI*M_SI/D_SI/C_SI/C_SI
    plt.plot(tEcc/(M*MTSUN_SI),abs(h22Ecc_new)/(G_SI*M_SI/D_SI/C_SI/C_SI
    plt.plot(tImr/(M*MTSUN_SI),abs(h22Imr)/(G_SI*M_SI/D_SI/C_SI/C_SI * mo
    #print(abs(h22Imr)/(G_SI*M_SI/D_SI/C_SI/C_SI * mode2polfac))
    plt.xlim(xmin=-2500)
    plt.xlim(xmax=100)
    plt.ylim(ymax=4.5e-1)
    plt.ylim(ymin=3e-2)
    plt.ylabel(r'$\mathcal{A}_{22}$',fontsize=22,labelpad=5)
    plt.xlabel(r'$t/M$',fontsize=22)
    plt.yscale('log')
    plt.tight_layout()
    plt.legend()
    plt.savefig('sph.pdf')

    #print((mode2polfac/4)*(((1-math.cos(angle))**2)))
    #print((mode2polfac/4)*(((1+math.cos(angle))**2)))
    plt.figure(figsize=(10,4.8))
    plt.plot(t_model/(M*MTSUN_SI), frequency_model, color='saddlebrown',a
    plt.plot(tEcc/(M*MTSUN_SI),omegaEcc,linestyle='-',linewidth=2.5,color
    plt.plot(tImr/(M*MTSUN_SI),omegaImr,linestyle='-',linewidth=2.5,color
    plt.xlim(xmin=-4500)
    plt.xlim(xmax=100)
    plt.ylim(ymin=2.5e-2)
    plt.ylim(ymax=3e-1)
    plt.ylabel(r'$M\omega_{22}$',fontsize=22,labelpad=5)
    plt.xlabel(r'$t/M$',fontsize=22)
    plt.yscale('log')
    plt.tight_layout()
    plt.legend()
```






```
Mode = (2,2) , Inclination =  30  degrees
```




Model_22
Ebersold
IMRPhenomXHM
$M\omega_{22}$
10^{-1}
−4000
−3000
−2000
−1000
0
t/M

MODEL $(l, m) = (3, 3)$

In [13]:
```
def INSP_Eber33(M0,q,e0,l0,flow,inc,d0,delta_t):
    eta=neu=nu=q/(1+q)**2
    G=c=M=d=1
    M2=M/(1+q)
    M1=M2*q
    Delta=math.sqrt(1-(4*neu))
    eta=nu=neu
    gamma=EulerGamma=0.577215664901

    conv=M*MTSUN_SI
    M_SI=M * MSUN_SI
    D_SI=(10**(6)) * PC_SI * d

    xlow = ((M0*MTSUN_SI*math.pi*flow)**(2/3))
    f_low = (xlow**(3/2)/(M*MTSUN_SI*math.pi))

    %run GW_functions.ipynb

    x=xlow
    v=math.sqrt(x)

    xie=v**3

    if delta_t>=1/2**14:
        del_t = 1/2**14
    elif delta_t<1/2**14 and delta_t>=1/2**16:
        del_t = 1/2**16
    elif delta_t<1/2**16 and delta_t>=1/2**18:
        del_t = 1/2**18
    else:
        del_t = 1/2**20

    phase_EccTD, tVec_PN = PNparams(M,q,d,f_low,e0,del_t)

    tC_NR = 0

    x0=xlow
    xi0=x0**(3/2)
    v0=xi0**(1/3)

    theta=((5*M/(eta))**(1/8))*(tC_NR-tVec_PN)**(-1/8)
    theta0=((5*M/(eta))**(1/8))*(tC_NR-tVec_PN[0])**(-1/8)
    fVec=x_from_t(theta, theta0, e0, M, eta)

    plotIdx2=np.nonzero(fVec>=0)
    fVec=fVec[plotIdx2]
    xiVec=(np.pi*M*fVec)
    xVec=xiVec**(2/3)
    vVec=xiVec**(1/3)
    xband=np.where(xVec<=1/6)
    xVec = xVec[xband]
    maxPNidx = len(xVec)
    tVec_PN=tVec_PN[:maxPNidx]


    lp=3
    mp=3

    j=0
    h33=[]
    h3_3=[]
    for i in xVec:    #tqdm(xVec) for status bar
        v=math.sqrt(i)
```

```
        v=math.sqrt(x)
        v0=math.sqrt(x0)
        xie=v**3
        xi0=v0**3
        l=mean_anomaly(xie, xi0, l0, eta, e0)
        e=e0*(xi0/xie)**(19/18)*epsilon(xie, eta)/epsilon(xi0, eta)
        psi=phase_EccTD[j]
        j=j+1
        xi=l   #use xi for amplitude (xie is being used for v**3)
        x=i
        h=amplitude_33(xi,x,nu,Delta,e) #### 22 mode requires additional

        hlm=8*math.sqrt(math.pi/5)*M*neu*i*h*((np.e)**(complex(0,-1)*mp*p
        hl_m=8*math.sqrt(math.pi/5)*M*neu*i*h*((np.e)**(complex(0,+1)*mp*

        h33.append(hlm)
        h3_3.append(hl_m)

    conv_t = M0*MTSUN_SI
    conv_h = G_SI*M0*MSUN_SI/(10**6 * PC_SI * d0)/C_SI/C_SI

    sph33, sph3_3 = sph_harmonics(inc,lp)

    h = np.multiply(h33,sph33)+np.multiply(h3_3,sph3_3)
    hp=np.real(h)#/(math.sin(inc)*(1+(math.cos(inc))**2))   #inc check
    hc=np.imag(h)#/(2*(math.sin(inc)*math.cos(inc)))
    time = tVec_PN - tVec_PN[-1]

    hp = np.array(hp) * conv_h
    hc = np.array(hc) * conv_h
    time = tVec_PN * conv_t

    hp_intrp = interp1d(time, hp, kind='cubic',fill_value='extrapolate')
    hc_intrp = interp1d(time, hc, kind='cubic',fill_value='extrapolate')
    t_intrp = np.arange(time[0], time[-1], delta_t)
    hp_intrp = hp_intrp(t_intrp)
    hc_intrp = hc_intrp(t_intrp)

    return np.array(hp_intrp), np.array(hc_intrp), np.array(t_intrp)
```

In [14]:
```
#INSP_Eber(3,3)
```

In [ ]:

In [15]:
```
def MODEL33(m,q0,e0,l0,fmin,angle,d,delta_t):


    M=m
    M1=q0*M/(1+q0)
    M2=M/(1+q0)
    eta=q0/(1+q0)**2
    M_SI=M*MSUN_SI
    D_SI=(10**(6))*PC_SI
    mode2polfac=4*(5/(64*np.pi))**(1/2)

    hp, hc, tinsp = INSP_Eber33(M,q0,e0,l0,fmin,(np.pi/180)*angle,d,delta
    sp,sc = get_td_waveform(approximant='IMRPhenomXHM', mass1=M1, mass2=M

    tshift = -tshift_Hinsp(q0,e0,l0)*M*MTSUN_SI
    tmin = max(tinsp[0]-tshift, sp.sample_times[0])

    #sp=sp/((math.sin(angle))*(1+(math.cos(angle))**2)) #inc check
    #sc=sc/(2*(math.sin(angle))*(math.cos(angle)))

    #Circular IMR
    sp_intrp = interp1d(sp.sample_times, sp, kind='cubic', fill_value='ex
    sc_intrp = interp1d(sc.sample_times, sc, kind='cubic', fill_value='ex
    tImr_intrp = np.arange(tmin, sp.sample_times[-1], delta_t)
    sp_intrp = sp_intrp(tImr_intrp)
    sc_intrp = sc_intrp(tImr_intrp)
    tImr = tImr_intrp
    hpImr = sp_intrp
    hcImr = sc_intrp
    h22Imr = hpImr + 1j*hcImr


    tshift = -tshift_Hinsp(q0,e0,l0)*M*MTSUN_SI




    #Interpolation Ebersold
    hp_intrp = interp1d(tinsp-tshift, hp, kind='cubic',fill_value='extrap
    hc_intrp = interp1d(tinsp-tshift, hc, kind='cubic',fill_value='extrap
    tEcc_intrp = np.arange(tmin, tinsp[-1]-tshift, delta_t)
    hp_intrp = hp_intrp(tEcc_intrp)
    hc_intrp = hc_intrp(tEcc_intrp)
    tEcc = tEcc_intrp
    hpEcc = hp_intrp
    hcEcc = hc_intrp
    h22Ecc = hpEcc + 1j*hcEcc

    phaseEcc = np.unwrap(np.angle(h22Ecc)*2)/2
    phaseImr = -np.unwrap(np.angle(h22Imr)*2)/2
    dphase = phaseEcc[0] - phaseImr[0]
    hp_new = real(h22Ecc * exp(-1j * dphase))
    hc_new = imag(h22Ecc * exp(-1j * dphase))

    phase_new = np.unwrap(np.angle(hp_new+1j*hc_new)*2)/2

    #phaseEcc = phase_new
    phaseEcc = abs(phase_new) #edit
    phaseImr = abs(phaseImr) #edit
    h22Ecc_new = (hp_new+1j*hc_new)

    arg = np.argmin(abs(tEcc-tamp_Hinsp(eta,e0,l0)*M*MTSUN_SI))
    Idxjoin = arg
```

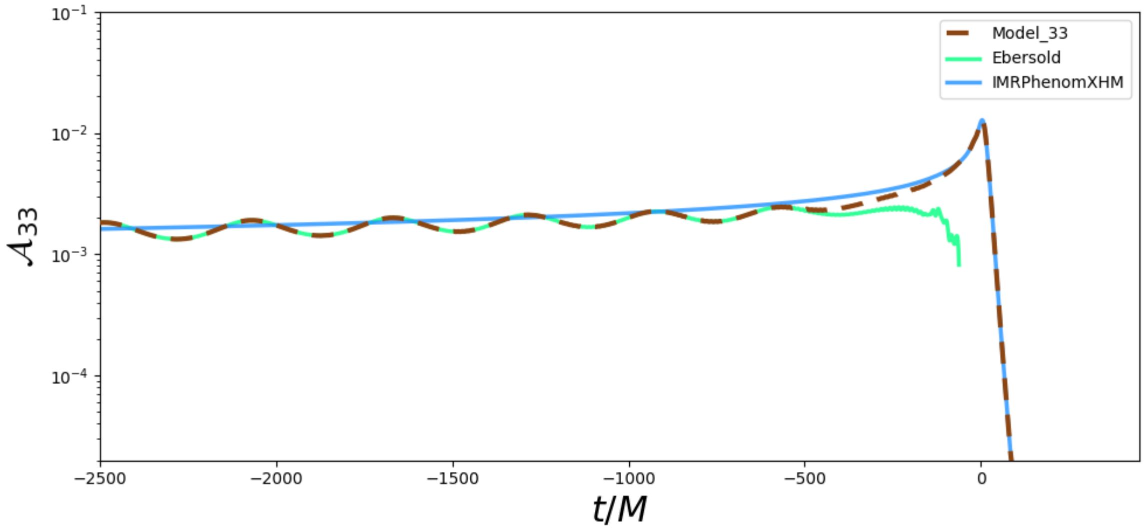
```
Idxjoin = arg

t_amp = tEcc[Idxjoin] - 500*M*MTSUN_SI
idxstr = np.argmin(abs(tEcc-t_amp))

#Amplitude Model
amp=[]
count=0
length=Idxjoin-idxstr

for i in range(idxstr,Idxjoin):
    amp.append(((length-count)*abs(h22Ecc_new[i])+count*abs(h22Imr[i]
    count=count+1

t_model=np.concatenate((tEcc[0:Idxjoin],tImr[Idxjoin:len(tImr)]))
h22amp=np.concatenate((abs(h22Ecc_new[0:idxstr]),amp))
h22amp_model=np.concatenate((h22amp,abs(h22Imr[Idxjoin:len(h22Imr)]))

omegaEcc = (M*MTSUN_SI/delta_t)*(np.gradient(phaseEcc))
omegaImr = (M*MTSUN_SI/delta_t)*(np.gradient(phaseImr))

tjoin0 = tfreq_Hinsp(eta,e0,l0)
tjoin = tjoin0 * M * MTSUN_SI
fjoin = np.argmin(abs(tEcc-tjoin))

#frequency model
tstop = min(tEcc[-1],-30*M*MTSUN_SI)
lst=np.argmin(abs(tEcc-tstop))

indx=lst - fjoin
a0 = []
n = indx-1
k = 0
for i in range(fjoin,fjoin+indx):
    a0.append(((n-k)*omegaEcc[i]+k*omegaImr[i])/n)
    k=k+1

f1 = np.concatenate((omegaEcc[0:fjoin],a0))
frequency_model = np.concatenate((f1,omegaImr[fjoin+indx:len(omegaImr
phase_f_model = np.cumsum(frequency_model)/(M*MTSUN_SI/delta_t)

hp_f_model = h22amp_model * np.cos(phase_f_model)
hc_f_model = h22amp_model * np.sin(phase_f_model)

#return np.array(hp_f_model), np.array(hc_f_model), np.array(t_model)

print('Mode = (3,3) , Inclination = ',angle,' degrees')
#Plot
plt.figure(figsize=(10,4.8))
plt.plot(t_model/(M*MTSUN_SI),h22amp_model/(G_SI*M_SI/D_SI/C_SI/C_SI
plt.plot(tEcc/(M*MTSUN_SI),abs(h22Ecc_new)/(G_SI*M_SI/D_SI/C_SI/C_SI
plt.plot(tImr/(M*MTSUN_SI),abs(h22Imr)/(G_SI*M_SI/D_SI/C_SI/C_SI * mo
#print(abs(h22Imr)/(G_SI*M_SI/D_SI/C_SI/C_SI * mode2polfac))
plt.ylim(ymax=1e-1)
plt.ylim(ymin=2e-5)
plt.xlim(xmin=-2500)
plt.xlim(xmax=450)
plt.ylabel(r'$\mathcal{A}_{33}$',fontsize=22,labelpad=5)
plt.xlabel(r'$t/M$',fontsize=22)
plt.yscale('log')
plt.tight_layout()
plt.legend()

plt.figure(figsize=(10,4.8))
```

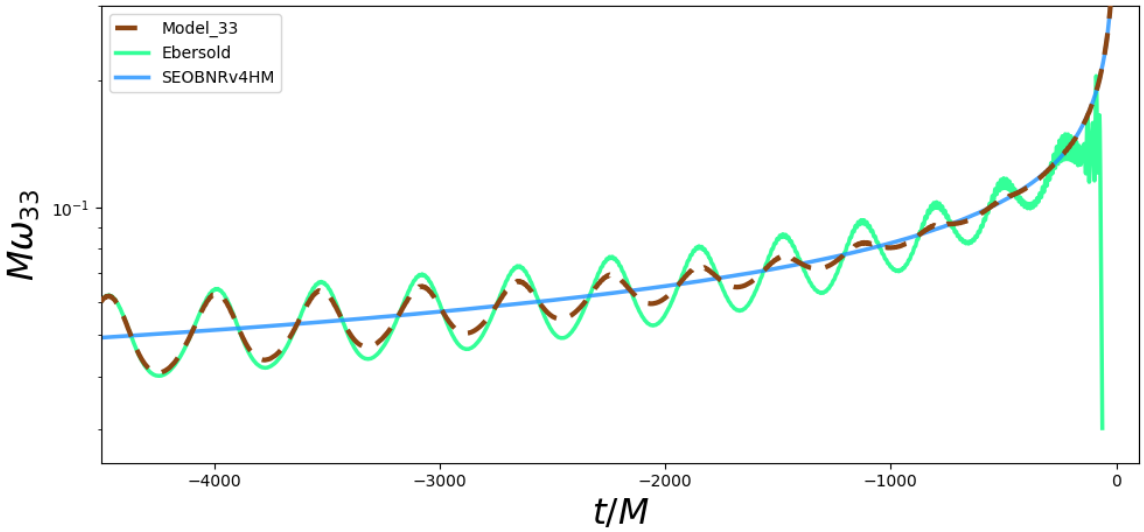
```
    plt.plot(t_model/(M*MTSUN_SI), frequency_model, color='saddlebrown',a
    plt.plot(tEcc/(M*MTSUN_SI),omegaEcc,linestyle='-',linewidth=2.5,color
    plt.plot(tImr/(M*MTSUN_SI),omegaImr,linestyle='-',linewidth=2.5,color
    plt.xlim(xmin=-4500)
    plt.xlim(xmax=100)
    plt.ylim(ymin=2.5e-2)
    plt.ylim(ymax=3e-1)
    plt.ylabel(r'$M\omega_{33}$',fontsize=22,labelpad=5)
    plt.xlabel(r'$t/M$',fontsize=22)
    plt.yscale('log')
    plt.tight_layout()
    plt.legend()
```

In [16]:

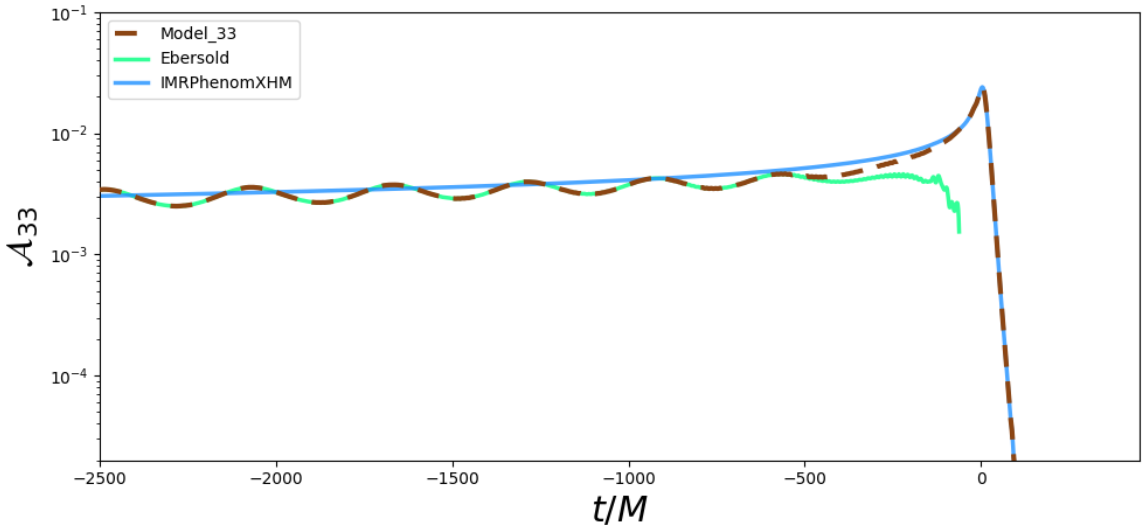
```
MODEL33(40,2,0.12,-0.181,20,10,1,1./4096)
MODEL33(40,2,0.12,-0.181,20,20,1,1./4096)
MODEL33(40,2,0.12,-0.181,20,30,1,1./4096)
```

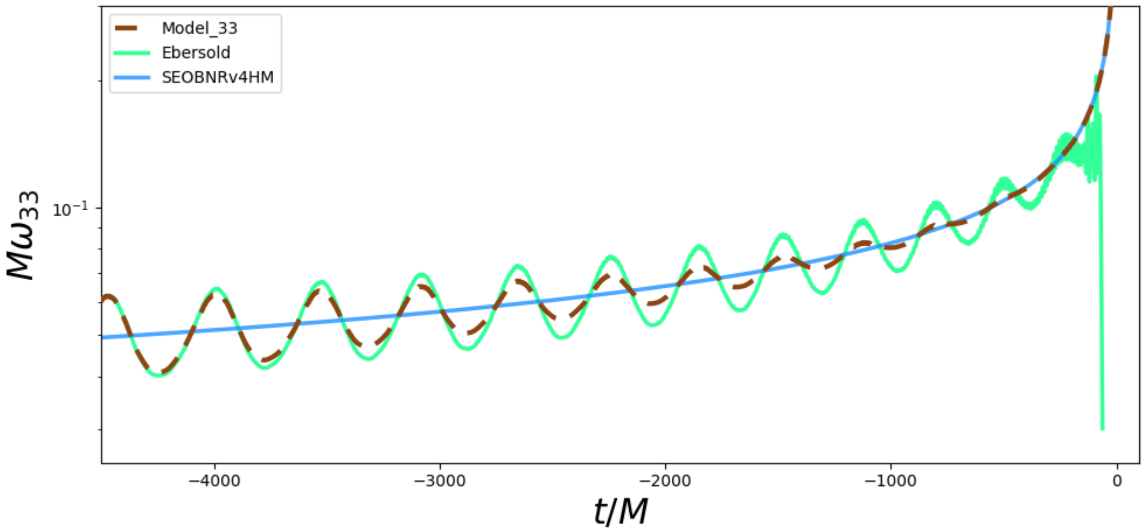
```
Mode = (3,3) , Inclination =  10  degrees
```



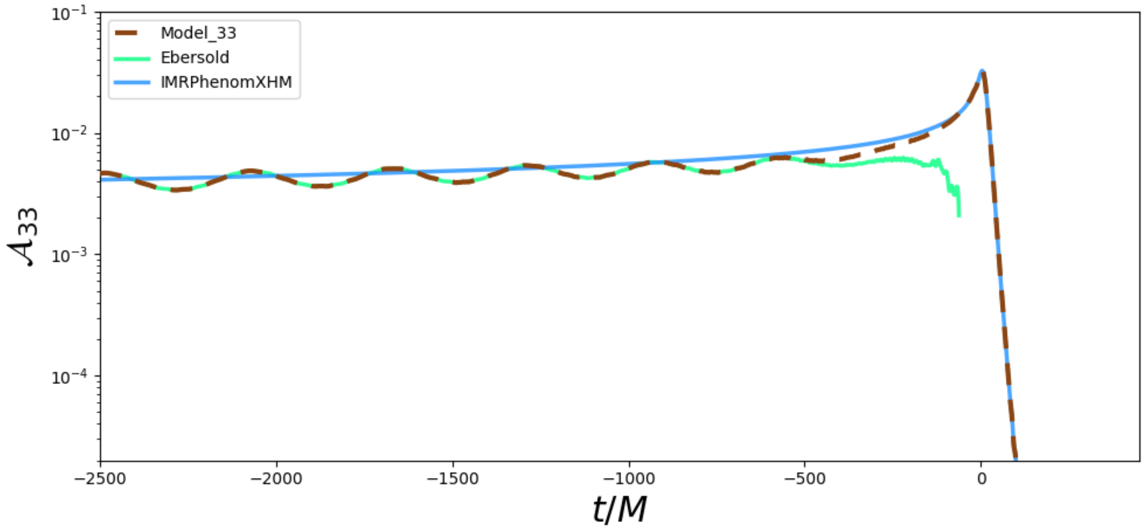


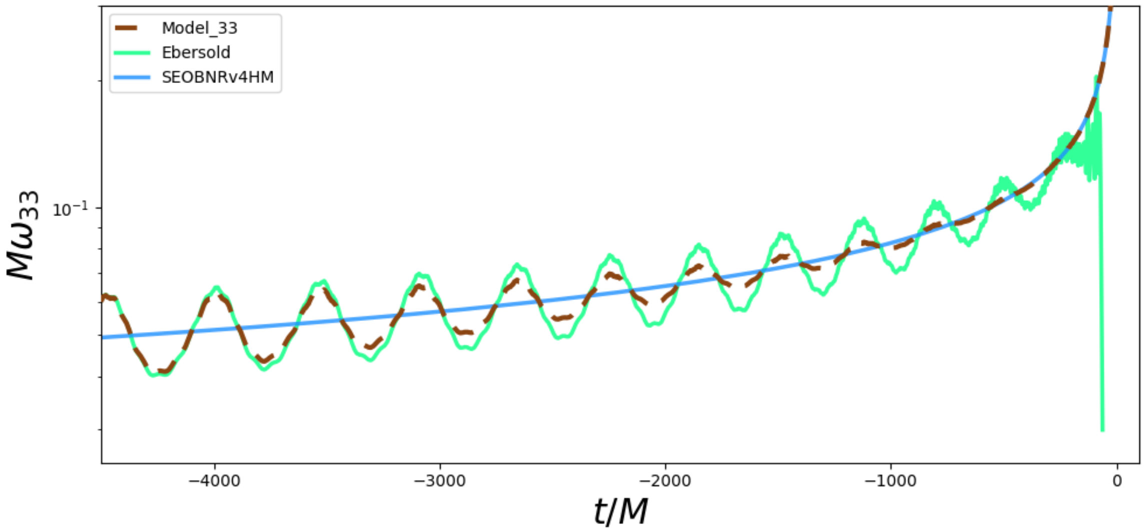

```
Mode = (3,3) , Inclination =  20  degrees
```






```
Mode = (3,3) , Inclination =  30  degrees
```




Model_33
Ebersold
SEOBNRv4HM
$M\omega_{33}$
10^{-1}
−4000
−3000
−2000
−1000
0
t/M

MODEL $(l, m) = (4, 4)$

In [17]:
```
def INSP_Eber44(M0,q,e0,l0,flow,inc,d0,delta_t):
    eta=neu=nu=q/(1+q)**2
    G=c=M=d=1
    M2=M/(1+q)
    M1=M2*q
    Delta=math.sqrt(1-(4*neu))
    eta=nu=neu
    gamma=EulerGamma=0.577215664901

    conv=M*MTSUN_SI
    M_SI=M * MSUN_SI
    D_SI=(10**(6)) * PC_SI * d

    xlow = ((M0*MTSUN_SI*math.pi*flow)**(2/3))
    f_low = (xlow**(3/2)/(M*MTSUN_SI*math.pi))

    %run GW_functions.ipynb

    x=xlow
    v=math.sqrt(x)

    xie=v**3

    if delta_t>=1/2**14:
        del_t = 1/2**14
    elif delta_t<1/2**14 and delta_t>=1/2**16:
        del_t = 1/2**16
    elif delta_t<1/2**16 and delta_t>=1/2**18:
        del_t = 1/2**18
    else:
        del_t = 1/2**20

    phase_EccTD, tVec_PN = PNparams(M,q,d,f_low,e0,del_t)

    tC_NR = 0

    x0=xlow
    xi0=x0**(3/2)
    v0=xi0**(1/3)

    theta=((5*M/(eta))**(1/8))*(tC_NR-tVec_PN)**(-1/8)
    theta0=((5*M/(eta))**(1/8))*(tC_NR-tVec_PN[0])**(-1/8)
    fVec=x_from_t(theta, theta0, e0, M, eta)

    plotIdx2=np.nonzero(fVec>=0)
    fVec=fVec[plotIdx2]
    xiVec=(np.pi*M*fVec)
    xVec=xiVec**(2/3)
    vVec=xiVec**(1/3)
    xband=np.where(xVec<=1/6)
    xVec = xVec[xband]
    maxPNidx = len(xVec)
    tVec_PN=tVec_PN[:maxPNidx]


    lp=4
    mp=4

    j=0
    h44=[]
    h4_4=[]
    for i in xVec:                #tqdm(xVec) for status bar
        v=math.sqrt(i)
```

```
        v=math.sqrt(x)
        v0=math.sqrt(x0)
        xie=v**3
        xi0=v0**3
        l=mean_anomaly(xie, xi0, l0, eta, e0)
        e=e0*(xi0/xie)**(19/18)*epsilon(xie, eta)/epsilon(xi0, eta)
        psi=phase_EccTD[j]
        j=j+1
        xi=l   #use xi for amplitude (xie is being used for v**3)
        x=i
        h=amplitude_44(xi,x,nu,Delta,e) #### 22 mode requires additional

        hlm=8*math.sqrt(math.pi/5)*M*neu*i*h*((np.e)**(complex(0,-1)*mp*p
        hl_m=8*math.sqrt(math.pi/5)*M*neu*i*h*((np.e)**(complex(0,+1)*mp*

        h44.append(hlm)
        h4_4.append(hl_m)

    conv_t = M0*MTSUN_SI
    conv_h = G_SI*M0*MSUN_SI/(10**6 * PC_SI * d0)/C_SI/C_SI

    sph44, sph4_4 = sph_harmonics(inc,lp)

    h = np.multiply(h44,sph44)+np.multiply(h4_4,sph4_4)
    hp=np.real(h)#/(((math.sin(inc))**2)*(1+(math.cos(inc))**2)) #inc che
    hc=np.imag(h)#/(2*((math.sin(inc))**2)*(math.cos(inc)))
    time = tVec_PN - tVec_PN[-1]

    hp = np.array(hp) * conv_h
    hc = np.array(hc) * conv_h
    time = tVec_PN * conv_t

    hp_intrp = interp1d(time, hp, kind='cubic',fill_value='extrapolate')
    hc_intrp = interp1d(time, hc, kind='cubic',fill_value='extrapolate')
    t_intrp = np.arange(time[0], time[-1], delta_t)
    hp_intrp = hp_intrp(t_intrp)
    hc_intrp = hc_intrp(t_intrp)

    return np.array(hp_intrp), np.array(hc_intrp), np.array(t_intrp)
```

In [18]:

```
def MODEL44(m,q0,e0,l0,fmin,angle,d,delta_t):


    M=m
    M1=q0*M/(1+q0)
    M2=M/(1+q0)
    eta=q0/(1+q0)**2
    M_SI=M*MSUN_SI
    D_SI=(10**(6))*PC_SI
    mode2polfac=4*(5/(64*np.pi))**(1/2)

    hp, hc, tinsp = INSP_Eber44(M,q0,e0,l0,fmin,(np.pi/180)*angle,d,delta
    sp,sc = get_td_waveform(approximant='IMRPhenomXHM', mass1=M1, mass2=M

    tshift = -tshift_Hinsp(q0,e0,l0)*M*MTSUN_SI
    tmin = max(tinsp[0]-tshift, sp.sample_times[0])


    #sp=sp/(((math.sin(angle))**2)*(1+(math.cos(angle))**2)) #inc check
    #sc=sc/(2*((math.sin(angle))**2)*(math.cos(angle)))
    #Circular IMR
    sp_intrp = interp1d(sp.sample_times, sp, kind='cubic', fill_value='ex
    sc_intrp = interp1d(sc.sample_times, sc, kind='cubic', fill_value='ex
```

```
tImr_intrp = np.arange(tmin, sp.sample_times[-1], delta_t)
sp_intrp = sp_intrp(tImr_intrp)
sc_intrp = sc_intrp(tImr_intrp)
tImr = tImr_intrp
hpImr = sp_intrp
hcImr = sc_intrp
h22Imr = hpImr + 1j*hcImr


tshift = -tshift_Hinsp(q0,e0,l0)*M*MTSUN_SI

#Interpolation Ebersold
hp_intrp = interp1d(tinsp-tshift, hp, kind='cubic',fill_value='extrap
hc_intrp = interp1d(tinsp-tshift, hc, kind='cubic',fill_value='extrap
tEcc_intrp = np.arange(tmin, tinsp[-1]-tshift, delta_t)
hp_intrp = hp_intrp(tEcc_intrp)
hc_intrp = hc_intrp(tEcc_intrp)
tEcc = tEcc_intrp
hpEcc = hp_intrp
hcEcc = hc_intrp
h22Ecc = hpEcc + 1j*hcEcc

phaseEcc = np.unwrap(np.angle(h22Ecc)*2)/2
phaseImr = -np.unwrap(np.angle(h22Imr)*2)/2
dphase = phaseEcc[0] - phaseImr[0]
hp_new = real(h22Ecc * exp(-1j * dphase))
hc_new = imag(h22Ecc * exp(-1j * dphase))

phase_new = np.unwrap(np.angle(hp_new+1j*hc_new)*2)/2

#phaseEcc = phase_new
phaseEcc = abs(phase_new) #edit
phaseImr = abs(phaseImr) #edit
h22Ecc_new = (hp_new+1j*hc_new)

arg = np.argmin(abs(tEcc-tamp_Hinsp(eta,e0,l0)*M*MTSUN_SI))
Idxjoin = arg

t_amp = tEcc[Idxjoin] - 500*M*MTSUN_SI
idxstr = np.argmin(abs(tEcc-t_amp))

#Amplitude Model
amp=[]
count=0
length=Idxjoin-idxstr

for i in range(idxstr,Idxjoin):
    amp.append(((length-count)*abs(h22Ecc_new[i])+count*abs(h22Imr[i]
    count=count+1

t_model=np.concatenate((tEcc[0:Idxjoin],tImr[Idxjoin:len(tImr)]))
h22amp=np.concatenate((abs(h22Ecc_new[0:idxstr]),amp))
h22amp_model=np.concatenate((h22amp,abs(h22Imr[Idxjoin:len(h22Imr)]))

omegaEcc = (M*MTSUN_SI/delta_t)*(np.gradient(phaseEcc))
omegaImr = (M*MTSUN_SI/delta_t)*(np.gradient(phaseImr))

tjoin0 = tfreq_Hinsp(eta,e0,l0)
tjoin = tjoin0 * M * MTSUN_SI
fjoin = np.argmin(abs(tEcc-tjoin))

#frequency model
tstop = min(tEcc[-1],-30*M*MTSUN_SI)
lst=np.argmin(abs(tEcc-tstop))
```

```
    lst=np.argmin(abs(tEcc-tstop))

    indx=lst - fjoin
    a0 = []
    n = indx-1
    k = 0
    for i in range(fjoin,fjoin+indx):
        a0.append(((n-k)*omegaEcc[i]+k*omegaImr[i])/n)
        k=k+1

    f1 = np.concatenate((omegaEcc[0:fjoin],a0))
    frequency_model = np.concatenate((f1,omegaImr[fjoin+indx:len(omegaImr
    phase_f_model = np.cumsum(frequency_model)/(M*MTSUN_SI/delta_t)
```

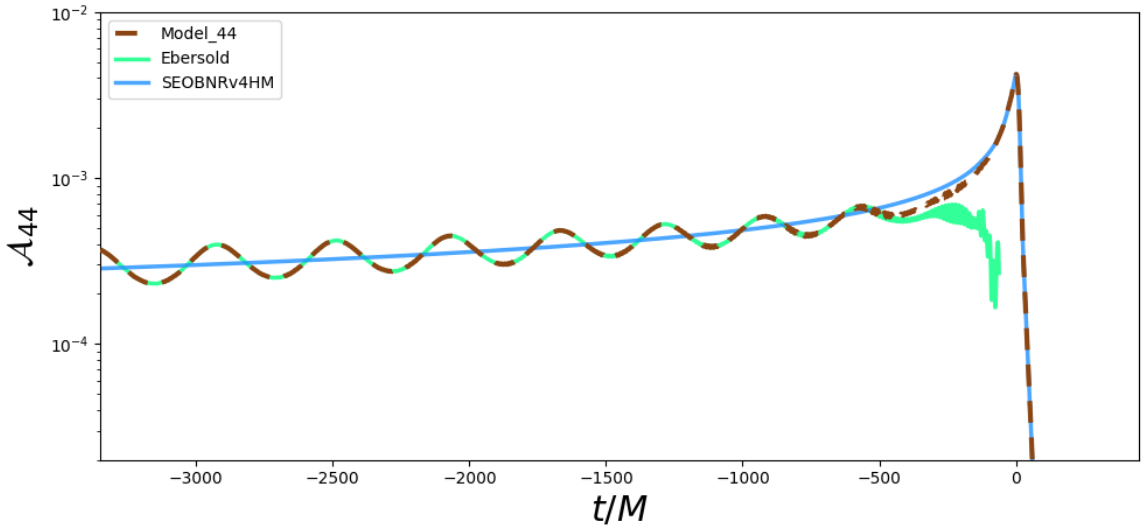
In [19]:

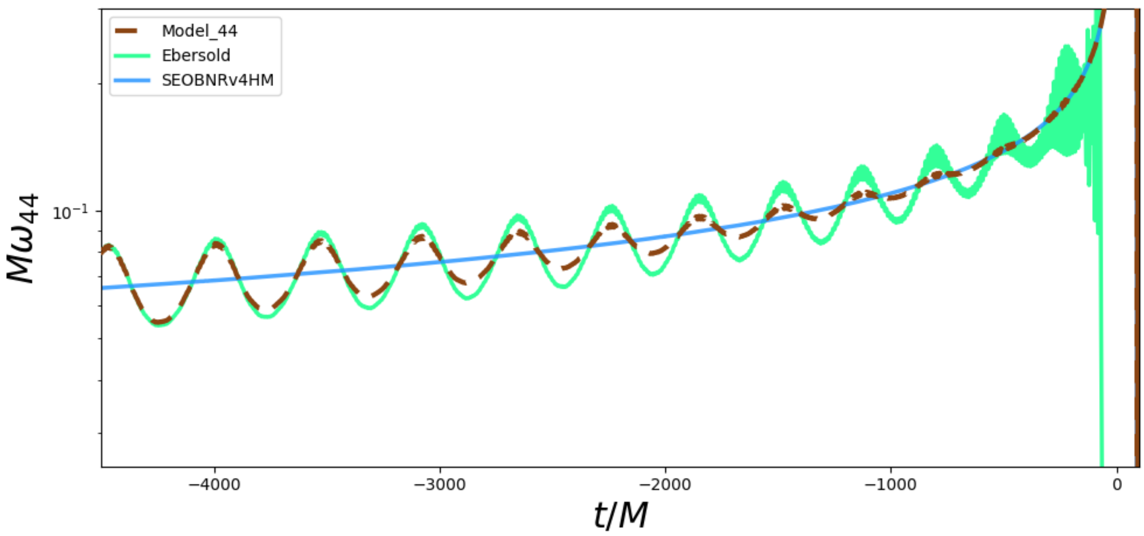
```
MODEL44(40,2,0.12,-0.181,20,10,1,1./4096)
MODEL44(40,2,0.12,-0.181,20,20,1,1./4096)
MODEL44(40,2,0.12,-0.181,20,30,1,1./4096)
```

```
    #return np.array(hp_f_model), np.array(hc_f_model), np.array(t_model)

    print('Mode = (4,4) , Inclination = ',angle,' degrees')
    #Plot
    plt.figure(figsize=(10,4.8))
    plt.plot(t_model/(M*MTSUN_SI),h22amp_model/(G_SI*M_SI/D_SI/C_SI/C_SI
    plt.plot(tEcc/(M*MTSUN_SI),abs(h22Ecc_new)/(G_SI*M_SI/D_SI/C_SI/C_SI
    plt.plot(tImr/(M*MTSUN_SI),abs(h22Imr)/(G_SI*M_SI/D_SI/C_SI/C_SI * mo

    plt.ylim(ymax=1e-2)
    plt.ylim(ymin=2e-5)
    plt.xlim(xmin=-3350)
    plt.xlim(xmax=450)
    plt.ylabel(r'$\mathcal{A}_{44}$',fontsize=22,labelpad=5)
    plt.xlabel(r'$t/M$',fontsize=22)
    plt.yscale('log')
    plt.tight_layout()
    plt.legend()

    plt.figure(figsize=(10,4.8))
    plt.plot(t_model/(M*MTSUN_SI), frequency_model, color='saddlebrown',a
    plt.plot(tEcc/(M*MTSUN_SI),omegaEcc,linestyle='-',linewidth=2.5,color
    plt.plot(tImr/(M*MTSUN_SI),omegaImr,linestyle='-',linewidth=2.5,color
    plt.xlim(xmin=-4500)
    plt.xlim(xmax=100)
    plt.ylim(ymin=2.5e-2)
    plt.ylim(ymax=3e-1)
    plt.ylabel(r'$M\omega_{44}$',fontsize=22,labelpad=5)
    plt.xlabel(r'$t/M$',fontsize=22)
    plt.yscale('log')
    plt.tight_layout()
    plt.legend()
```

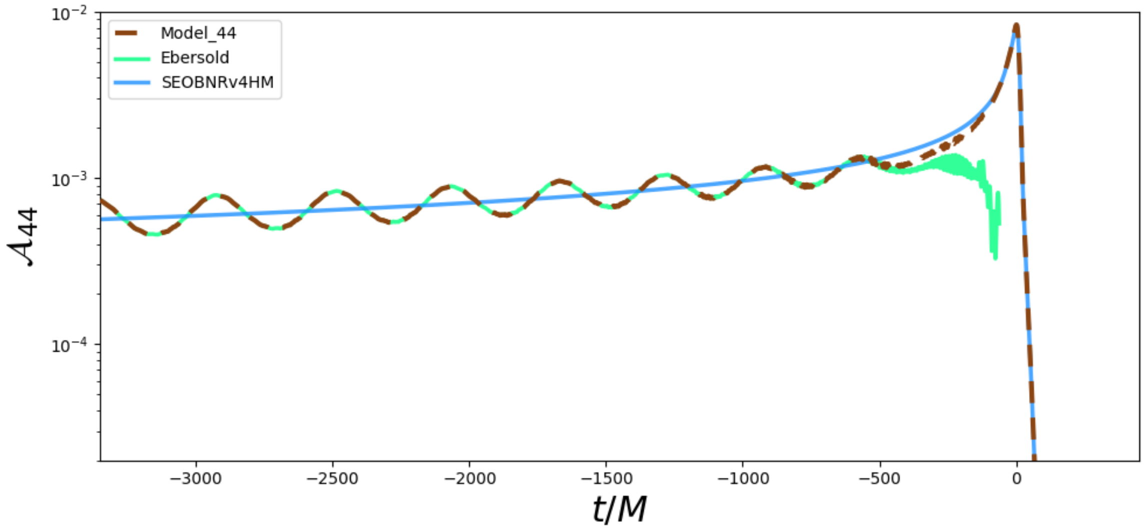
t/M

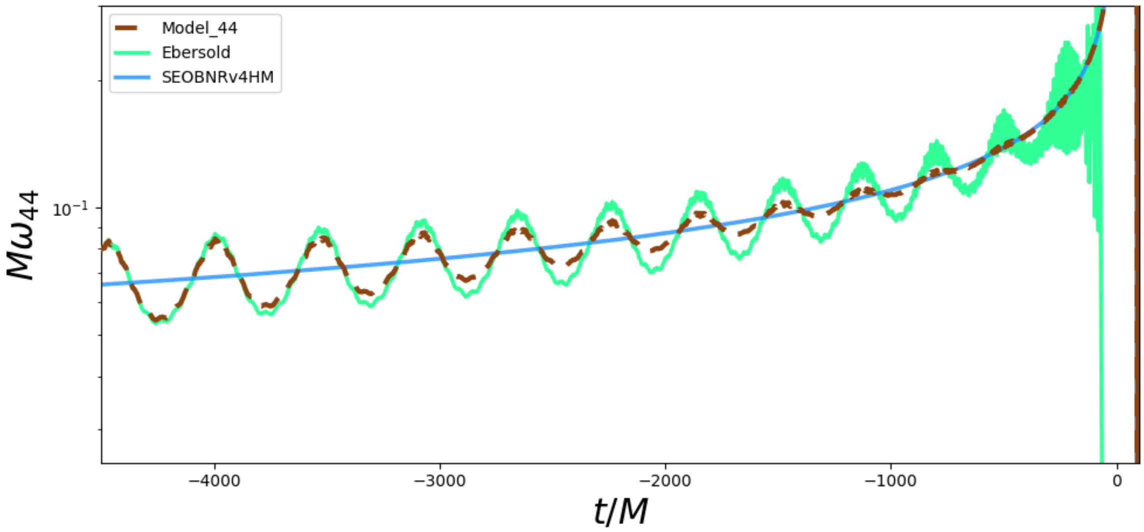
```
Mode = (4,4) , Inclination =  20  degrees
```






```
Mode = (4,4) , Inclination =  30  degrees
```




Model_44
Ebersold
SEOBNRv4HM
10^{-1}
$M\omega_{44}$
−4000
−3000
−2000
−1000
0
t/M

MODEL $(l, m) = (5, 5)$

In [20]:
```
def INSP_Eber55(M0,q,e0,l0,flow,inc,d0,delta_t):
    eta=neu=nu=q/(1+q)**2
    G=c=M=d=1
    M2=M/(1+q)
    M1=M2*q
    Delta=math.sqrt(1-(4*neu))
    eta=nu=neu
    gamma=EulerGamma=0.577215664901

    conv=M*MTSUN_SI
    M_SI=M * MSUN_SI
    D_SI=(10**(6)) * PC_SI * d

    xlow = ((M0*MTSUN_SI*math.pi*flow)**(2/3))
    f_low = (xlow**(3/2)/(M*MTSUN_SI*math.pi))

    %run GW_functions.ipynb

    x=xlow
    v=math.sqrt(x)

    xie=v**3

    if delta_t>=1/2**14:
        del_t = 1/2**14
    elif delta_t<1/2**14 and delta_t>=1/2**16:
        del_t = 1/2**16
    elif delta_t<1/2**16 and delta_t>=1/2**18:
        del_t = 1/2**18
    else:
        del_t = 1/2**20

    phase_EccTD, tVec_PN = PNparams(M,q,d,f_low,e0,del_t)

    tC_NR = 0

    x0=xlow
    xi0=x0**(3/2)
    v0=xi0**(1/3)

    theta=((5*M/(eta))**(1/8))*(tC_NR-tVec_PN)**(-1/8)
    theta0=((5*M/(eta))**(1/8))*(tC_NR-tVec_PN[0])**(-1/8)
    fVec=x_from_t(theta, theta0, e0, M, eta)

    plotIdx2=np.nonzero(fVec>=0)
    fVec=fVec[plotIdx2]
    xiVec=(np.pi*M*fVec)
    xVec=xiVec**(2/3)
    vVec=xiVec**(1/3)
    xband=np.where(xVec<=1/6)
    xVec = xVec[xband]
    maxPNidx = len(xVec)
    tVec_PN=tVec_PN[:maxPNidx]


    lp=5
    mp=5

    j=0
    h55=[]
    h5_5=[]
    for i in xVec:                          #tqdm(xVec) for status bar
```

```
        v=math.sqrt(x)
        v0=math.sqrt(x0)
        xie=v**3
        xi0=v0**3
        l=mean_anomaly(xie, xi0, l0, eta, e0)
        e=e0*(xi0/xie)**(19/18)*epsilon(xie, eta)/epsilon(xi0, eta)
        psi=phase_EccTD[j]
        j=j+1
        xi=l   #use xi for amplitude (xie is being used for v**3)
        x=i
        h=amplitude_55(xi,x,nu,Delta,e) #### 22 mode requires additional

        hlm=8*math.sqrt(math.pi/5)*M*neu*i*h*((np.e)**(complex(0,-1)*mp*p
        hl_m=8*math.sqrt(math.pi/5)*M*neu*i*h*((np.e)**(complex(0,+1)*mp*

        h55.append(hlm)
        h5_5.append(hl_m)

    conv_t = M0*MTSUN_SI
    conv_h = G_SI*M0*MSUN_SI/(10**6 * PC_SI * d0)/C_SI/C_SI

    sph55, sph5_5 = sph_harmonics(inc,lp)

    h = np.multiply(h55,sph55)+np.multiply(h5_5,sph5_5)
    hp=np.real(h)
    hc=np.imag(h)
    time = tVec_PN - tVec_PN[-1]

    hp = np.array(hp) * conv_h
    hc = np.array(hc) * conv_h
    time = tVec_PN * conv_t

    hp_intrp = interp1d(time, hp, kind='cubic',fill_value='extrapolate')
    hc_intrp = interp1d(time, hc, kind='cubic',fill_value='extrapolate')
    t_intrp = np.arange(time[0], time[-1], delta_t)
    hp_intrp = hp_intrp(t_intrp)
    hc_intrp = hc_intrp(t_intrp)

    return np.array(hp_intrp), np.array(hc_intrp), np.array(t_intrp)
```

In [21]:
```
def MODEL55(m,q0,e0,l0,fmin,angle,d,delta_t):


    M=m
    M1=q0*M/(1+q0)
    M2=M/(1+q0)
    eta=q0/(1+q0)**2
    M_SI=M*MSUN_SI
    D_SI=(10**(6))*PC_SI
    mode2polfac=4*(5/(64*np.pi))**(1/2)

    hp, hc, tinsp = INSP_Eber55(M,q0,e0,l0,fmin,angle,d,delta_t) #paused
    sp,sc = get_td_waveform(approximant='SEOBNRv4HM', mass1=M1, mass2=M2,

    tshift = -tshift_Hinsp(q0,e0,l0)*M*MTSUN_SI
    tmin = max(tinsp[0]-tshift, sp.sample_times[0])

    #Circular IMR
    sp_intrp = interp1d(sp.sample_times, sp, kind='cubic', fill_value='ex
    sc_intrp = interp1d(sc.sample_times, sc, kind='cubic', fill_value='ex
    tImr_intrp = np.arange(tmin, sp.sample_times[-1], delta_t)
    sp_intrp = sp_intrp(tImr_intrp)
    sc_intrp = sc_intrp(tImr_intrp)
```

```
tImr = tImr_intrp
hpImr = sp_intrp
hcImr = sc_intrp
h22Imr = hpImr + 1j*hcImr


tshift = -tshift_Hinsp(q0,e0,l0)*M*MTSUN_SI

#Interpolation Ebersold
hp_intrp = interp1d(tinsp-tshift, hp, kind='cubic',fill_value='extrap
hc_intrp = interp1d(tinsp-tshift, hc, kind='cubic',fill_value='extrap
tEcc_intrp = np.arange(tmin, tinsp[-1]-tshift, delta_t)
hp_intrp = hp_intrp(tEcc_intrp)
hc_intrp = hc_intrp(tEcc_intrp)
tEcc = tEcc_intrp
hpEcc = hp_intrp
hcEcc = hc_intrp
h22Ecc = hpEcc + 1j*hcEcc

phaseEcc = np.unwrap(np.angle(h22Ecc)*2)/2
phaseImr = -np.unwrap(np.angle(h22Imr)*2)/2
dphase = phaseEcc[0] - phaseImr[0]
hp_new = real(h22Ecc * exp(-1j * dphase))
hc_new = imag(h22Ecc * exp(-1j * dphase))

phase_new = np.unwrap(np.angle(hp_new+1j*hc_new)*2)/2

#phaseEcc = phase_new
phaseEcc = abs(phase_new) #edit
phaseImr = abs(phaseImr) #edit
h22Ecc_new = (hp_new+1j*hc_new)

arg = np.argmin(abs(tEcc-tamp_Hinsp(eta,e0,l0)*M*MTSUN_SI))
Idxjoin = arg

t_amp = tEcc[Idxjoin] - 500*M*MTSUN_SI
idxstr = np.argmin(abs(tEcc-t_amp))

#Amplitude Model
amp=[]
count=0
length=Idxjoin-idxstr

for i in range(idxstr,Idxjoin):
    amp.append(((length-count)*abs(h22Ecc_new[i])+count*abs(h22Imr[i]
    count=count+1

t_model=np.concatenate((tEcc[0:Idxjoin],tImr[Idxjoin:len(tImr)]))
h22amp=np.concatenate((abs(h22Ecc_new[0:idxstr]),amp))
h22amp_model=np.concatenate((h22amp,abs(h22Imr[Idxjoin:len(h22Imr)]))

omegaEcc = (M*MTSUN_SI/delta_t)*(np.gradient(phaseEcc))
omegaImr = (M*MTSUN_SI/delta_t)*(np.gradient(phaseImr))

tjoin0 = tfreq_Hinsp(eta,e0,l0)
tjoin = tjoin0 * M * MTSUN_SI
fjoin = np.argmin(abs(tEcc-tjoin))

#frequency model
tstop = min(tEcc[-1],-30*M*MTSUN_SI)
lst=np.argmin(abs(tEcc-tstop))

indx=lst - fjoin
a0 = []
```

```
    a0 = []
    n = indx-1
    k = 0
    for i in range(fjoin,fjoin+indx):
        a0.append(((n-k)*omegaEcc[i]+k*omegaImr[i])/n)
        k=k+1

    f1 = np.concatenate((omegaEcc[0:fjoin],a0))
```

In [22]:
```
#MODEL55(40,2,0.12,-0.181,20,10,1,1./4096)
#MODEL55(40,2,0.12,-0.181,20,20,1,1./4096)
#MODEL55(40,2,0.12,-0.181,20,30,1,1./4096)
    hp_f_model = h22amp_model * np.cos(phase_f_model)
    hc_f_model = h22amp_model * np.sin(phase_f_model)
```

In [23]:
```
import pycbc
pycbc.__version__
    return np.array(hp_f_model), np.array(hc_f_model), np.array(t_model)
```

Out[23]:
```
    #print('Mode = (5,5) , Inclination = ',angle,' degrees')
    #Plot
```

In [24]:
```
math.cos(60)
    #plt.plot(t_model/(M*MTSUN_SI),h22amp_model/(G_SI*M_SI/D_SI/C_SI/C_SI
```

Out[24]:
```
    #plt.plot(tEcc/(M*MTSUN_SI),abs(h22Ecc_new)/(G_SI*M_SI/D_SI/C_SI/C_SI
    #plt.plot(tImr/(M*MTSUN_SI),abs(h22Imr)/(G_SI*M_SI/D_SI/C_SI/C_SI * m
```

In [25]:
```
math.cos(np.pi*60/180)
    #plt.ylim(ymax=1e0)
```

Out[25]:
```
    #plt.ylim(ymin=2e-5)
    #plt.xlim(xmin=-3350)
    #plt.xlim(xmax=450)
```

In [26]:
```
np.cos(60)
```

Out[26]:
```
    #plt.xlabel(r'$t/M$',fontsize=22)
    #plt.yscale('log')
    #plt.legend()
```

In [ ]:
```
    #plt.figure(figsize=(10,4.8))
    #plt.plot(t_model/(M*MTSUN_SI), frequency_model, color='saddlebrown',
    #plt.plot(tEcc/(M*MTSUN_SI),omegaEcc,linestyle='-',linewidth=2.5,colo
    #plt.plot(tImr/(M*MTSUN_SI),omegaImr,linestyle='-',linewidth=2.5,colo
    #plt.xlim(xmin=-4500)
    #plt.xlim(xmax=100)
    #plt.ylim(ymin=2.5e-2)
    #plt.ylim(ymax=3e-1)
    #plt.ylabel(r'$M\omega_{55}$',fontsize=22,labelpad=5)
    #plt.xlabel(r'$t/M$',fontsize=22)
    #plt.yscale('log')
    #plt.legend()
```