```python
In [ ]: import matplotlib
        from helper_functions import datamaker
        from scipy.integrate import quad
        from scipy.interpolate import griddata
        import numpy as np
        import matplotlib.pyplot as plt
        from sympy import *
        import pickle
        import os
        from matplotlib.ticker import FormatStrFormatter

        # Defining the Observables
        q = Symbol('q')
        omega = Symbol('\Omega')
        sigma = Symbol('\Sigma')
        sigmatot = Symbol('Sigma_tot')
        sigmasfr = Symbol('Sigma_SFR')
        T = Symbol('T')

        # Defining the Constants
        gamma = Symbol('gamma')
        boltz = Symbol('k_B')
        mu = Symbol('mu')
        mh = Symbol('m_H')

        # Defining the general parameters
        u = Symbol('u')
        tau = Symbol('tau')
        l = Symbol('l')
        h = Symbol('h')

        # conversion factors
        pc_kpc = 1e3   # number of pc in one kpc
        cm_km = 1e5   # number of cm in one km
        cm_kpc = 3.086e+21   # number of centimeters in one parsec
        s_Myr = 1e+6*(365*24*60*60)   # megayears to seconds
        deg_rad = 180e0/np.pi
        arcmin_deg = 60e0
        arcsec_deg = 3600e0


        #################################################################################
        #################################################################################
        current_directory=os.path.abspath(os.curdir)
        #reading the parameters
        params = {}
        with open(current_directory+ '\parameter_file.in', 'r') as FH:
            for file in FH.readlines():
                line = file.strip()
                try:
                    par_name, value = line.split('=')
                except ValueError:
                    print("Record: ", line)
                    raise Exception(
                        "Failed while unpacking. Not enough arguments to supply.")
```

```python
        try:
            params[par_name] = np.float64(value)
        except ValueError:
            num, denom = value.split('/')
            params[par_name] = np.float64(num) / np.float64(denom)


#########################################################################################
os.chdir(current_directory)

# def choose_galaxy(name):
#     if name=="m33":
#         from data.data_magfield_observables_m33 import G_dat_Bord,G_dat_Breg,G_da

#         os.chdir(current_directory)
#         from data.data_conv_M33 import nandeleted_data,data_v_disp
#         corrected_radius,dat_sigmatot,dat_sigmaHI,dat_sigmaH2, dat_q, dat_omega,
#         dat_u=data_v_disp

#         with open(current_directory+r'\mag_observables_m33.pickle', 'rb') as f:
#             kpc_r, h_f, l_f, u_f, cs_f, alphak_f, tau_f, biso_f, bani_f, Bbar_f,
#     return G_dat_Bord,G_dat_Breg,G_dat_Btot,rmdat_tanpo,rm_errdat_tanpo,M_dat_pb,

# choose_galaxy('m33')
#########################################################################################

# from data.data_magfield_observables_6946 import G_dat_Bord,G_dat_Breg,G_dat_Btot,
# from data.data_magfield_observables_m51 import G_dat_Bord,G_dat_Breg,G_dat_Btot,r
from data.data_magfield_observables_m33 import G_dat_Bord,G_dat_Breg,G_dat_Btot,rmd
# from data.data_magfield_observables_m31 import G_dat_Bord,G_dat_Breg,G_dat_Btot,R

#########################################################################################
from get_magnetic_observables import omt, kah, taue_f, taur_f

#change file name here
with open(current_directory+r'\mag_observables_m33.pickle', 'rb') as f:
    kpc_r, h_f, l_f, u_f, cs_f, alphak_f, tau_f, biso_f, bani_f, Bbar_f, tanpB_f, t

os.chdir(current_directory)
with open(current_directory+r'\errors_quan.pickle', 'rb') as f:
    h_err, l_err, u_err, cs_err, alphak_err, tau_err, biso_err, bani_err, Bbar_err

os.chdir(current_directory+'\data')
with open('zip_data.pickle', 'rb') as f:
    kpc_r, data_pass = pickle.load(f)
#########################################################################################

os.chdir(current_directory+'\expressions')
from expressions.magnetic_expressions import Dk, Dc

dkdc_f = datamaker((Dk/Dc), data_pass, h_f, tau_f, alphak_f)
alpham_f = alphak_f*((1/dkdc_f)-1)
alphasat_f = alphak_f + alpham_f
#########################################################################################
#########################################################################################

# M31
```

```python
# os.chdir(current_directory)
# # from data.data_conv_M31 import kpc_r, dat_u,dat_u_warp
# from data.data_M31 import kpc_r, dat_u,dat_u_warp
# corrected_radius=kpc_r
###############################################################################

# M51
# os.chdir(current_directory)
# from data.data_m51 import nandeleted_data,vel_disp
# corrected_radius,dat_sigmatot,dat_sigmaHI,dat_sigmaH2, dat_q, dat_omega, dat_sigm
# dat_u=vel_disp #this has been interpolated in data.m51
###############################################################################
# M33
os.chdir(current_directory)
from data.data_conv_M33 import nandeleted_data,data_v_disp
corrected_radius,dat_sigmatot,dat_sigmaHI,dat_sigmaH2, dat_q, dat_omega, dat_sigmas
dat_u=data_v_disp
###############################################################################
# 6946
# os.chdir(current_directory)
# from data.data_6946 import nandeleted_data,vel_disp
# corrected_radius,dat_sigmatot,dat_sigmaHI,dat_sigmaH2, dat_q, dat_omega, dat_sigm
# dat_u=vel_disp
###############################################################################

os.chdir(current_directory)

pB = np.arctan(-tanpB_f)
pB_err = -tanpB_err/(1+tanpB_f**2)
pbb = np.arctan(tanpb_f)
pbb_err = tanpb_err/(1+tanpb_f**2)
pbo = (1/2)*((1+(2*Bbar_f*bani_f*np.cos(pbb-pB))/
            (bani_f**2+Bbar_f**2))*np.arctan((Bbar_f*np.sin(pB) + bani_f*np.sin(p
                                        ((Bbar_f*np.cos(pB)) + bani_f*np.cos
                                        + (1-(2*Bbar_f*bani_f*np.co
                                        (bani_f**2+Bbar_f**2))*n


def pogen(b, B, pb, pB, s):
    return (np.exp(-b**2/(2*s**2)))/
            (np.sqrt(2*(np.pi))*s))*(1+(2*B*b*np.cos(pb-pB))/
                                    (b**2 + B**2))*np.arctan((B*np.sin(pB) + b*np.

brms = np.sqrt(np.average(bani_f**2))

h = 1e-8 #here h is the tolerance
def dpodbani(b, B, pb, pB, s):
    return (pogen(b, B, pb, pB, s+h)-pogen(b, B, pb, pB, s-h))/(2*h)
def dpodBbar(b, B, pb, pB, s):
    return (pogen(b, B+h, pb, pB, s)-pogen(b, B-h, pb, pB, s))/(2*h)
h = 0.01 #here h is the tolerance
def dpodpB(b, B, pb, pB, s):
    return (pogen(b, B, pb, pB+h, s)-pogen(b, B, pb, pB-h, s))/(2*h)
def dpodpb(b, B, pb, pB, s):
    return (pogen(b, B, pb+h, pB, s)-pogen(b, B, pb-h, pB, s))/(2*h)
```

```python
def integrator(fn, interval = 1e-3):
    return np.array([quad(fn, 1e-11, 1e-9, args=(Bbar_f[i], pbb[i], pB[i], bani_f[i
                points=[-interval*brms, interval*brms])[0] for i in range(len(kpc_r)
pog = integrator(pogen)
inte = 1e-10
pog_err = np.array([quad(pogen, -inte, inte, args=(Bbar_f[i], pbb[i], pB[i], bani_f
                points=[-inte*brms, inte*brms])[1] for i in range(len(kpc_r))])
pog_err += np.sqrt((integrator(dpodbani,inte)*bani_err)**2 +(integrator(dpodBbar,in
                +(integrator(dpodpB,inte)*pB_err)**2+(integrator(dpodpb,inte)*pbb

G_scal_Bbartot = np.sqrt(biso_f**2 + bani_f**2 + Bbar_f**2)
G_scal_Bbarreg = Bbar_f
G_scal_Bbarord = np.sqrt(bani_f**2 + Bbar_f**2)

G_scal_Bbartot_err = np.sqrt((biso_err*biso_f )**2+ (bani_err*bani_f)**2 + (Bbar_er
G_scal_Bbarreg_err = Bbar_err
G_scal_Bbarord_err = np.sqrt((bani_err*bani_f)**2 + (Bbar_err*Bbar_f)**2)/G_scal_Bb

os.chdir(current_directory)
```

```
Root found succesfully
Solved the magnetic expressions
```

In [ ]:
```python
# len(corrected_radius)
# corrected_radius
```

In [ ]:
```python
m = 2
dm = 2.5
fs = 15
lfs = 10
leg_textsize = 10
axis_textsize = 10
rc = {"font.family" : "serif",
      "mathtext.fontset" : "stix"}
plt.rcParams.update(rc)
plt.rcParams["font.serif"] = ["Times New Roman"] + plt.rcParams["font.serif"]
matplotlib.rc('xtick', labelsize=fs)
matplotlib.rc('ytick', labelsize=fs)
matplotlib.ticker.AutoMinorLocator(n=None)
plt.rcParams["xtick.minor.visible"] =  True
plt.rcParams["ytick.minor.visible"] =  True
plt.rcParams["legend.loc"] = 'upper right'
plt.rcParams["errorbar.capsize"] = 2
```

In [ ]:
```python
#change axis params here
def axis_pars(ax):
    ax.xaxis.set_ticks(np.arange(1,15, 1))
    # ax.xaxis.set_ticks(np.arange(6,15, 1)) # for m31 only
    ax.xaxis.set_major_formatter(FormatStrFormatter('%g'))
    ax.tick_params(axis='both', which='minor',
                   labelsize=axis_textsize, colors='k', length=3, width=1)
    ax.tick_params(axis='both', which='major',
                   labelsize=axis_textsize, colors='k', length=5, width=1.25)
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.spines['bottom'].set_linewidth(2)
```

```python
        ax.spines['left'].set_linewidth(2)
        ax.legend(fontsize=lfs, frameon=False, handlelength=4, ncol=1, prop={
                'size': leg_textsize, 'family': 'Times New Roman'}, fancybox=True, fram

    def fill_error(ax, quan_f, quan_err, color = 'red', alpha = 0.2):
        ax.fill_between(kpc_r, (quan_f+quan_err), (quan_f-quan_err)
                        , alpha=alpha, edgecolor='k', facecolor=color, where = None, in
```

## plotting h

```python
In [ ]:  fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)
         ax.plot(kpc_r, h_f*pc_kpc/cm_kpc, c='r', linestyle='-', mfc='k',
                    mec='k', markersize=m, marker='o', label=r' $h$(pc)')
         ax.plot(kpc_dat_r, pc_dat_h, c='b', linestyle='dotted',
                    marker='*',mfc='y',mec='b',mew=1, markersize = 7, label=r'Fiducial va
         ax.plot(kpc_r, l_f*pc_kpc/cm_kpc, c='g',
                    linestyle='-', mfc='k', mec='k', markersize=m, marker='o', label=r'Co
         # ax.plot(kpc_r, datamaker(lsn , data_pass, h_f, tau_f)*pc_kpc/cm_kpc,c = 'y',lines
         ax.axhline(y=100, color='black', linestyle='--', alpha = 0.2)
         # ax.set_yticks(list(plt.yticks()[0])+[100])
         axis_pars(ax)
         fill_error(ax, h_f*pc_kpc/cm_kpc, h_err*pc_kpc/cm_kpc)
         ax.set_xlabel(r'Radius (kpc)', fontsize=fs)
         ax.set_ylabel(r'Scale height (pc)', fontsize=fs)
         ax.axhline(y=0, color='black', linestyle='--', alpha = 0.2)

         hf=h_f*(pc_kpc/cm_kpc)
         lf=l_f*(pc_kpc/cm_kpc)
         index_sup = np.where(kpc_r== 8.75)
         index_sub = np.where(kpc_r== 14.25)
         print('h sup',hf[index_sup],kpc_r[index_sup])
         print(hf[index_sub],kpc_r[index_sub])
         print('l sup',lf[index_sup],kpc_r[index_sup])
         print(lf[index_sub],kpc_r[index_sub])

         print(hf)
         print(kpc_r)
         # plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M
```
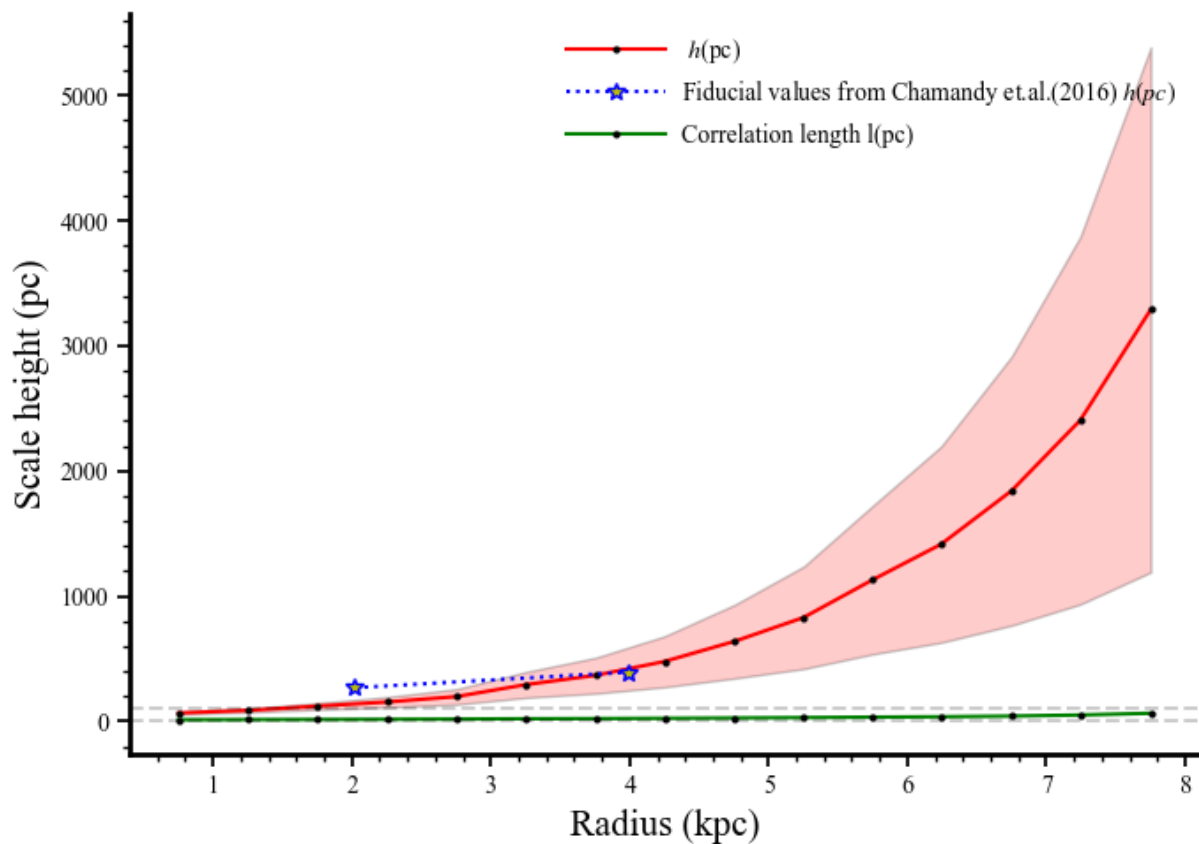
```
h sup [] []
[] []
l sup [] []
[] []
[  66.48304549   87.92889571  121.11101626  154.56103714  197.55765238
   292.37293407  367.18251153  477.98796917  639.82662858  829.60753569
  1129.58784164 1414.70207731 1841.74882399 2405.41613531 3288.90965758]
[0.7504 1.257  1.752  2.259  2.754  3.249  3.756  4.251  4.758  5.253
 5.748  6.243  6.75   7.245  7.752 ]
```
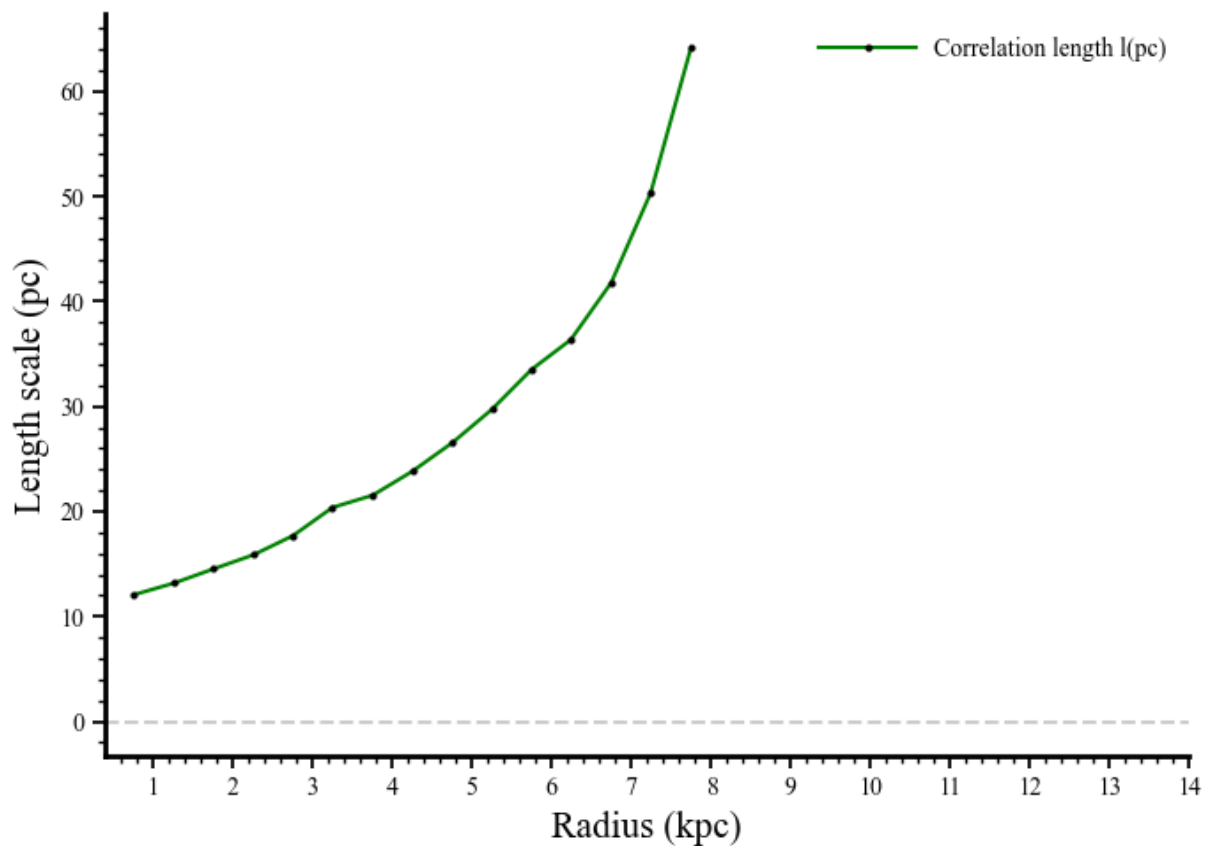
```
In [ ]:  fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)

         ax.plot(kpc_r, l_f*pc_kpc/cm_kpc, c='g',
                     linestyle='-', mfc='k', mec='k', markersize=m, marker='o', label=r'Co
         # ax.plot(kpc_r, datamaker(lsn , data_pass, h_f, tau_f)*pc_kpc/cm_kpc,c = 'y',lines
         # ax.axhline(y=100, color='black', linestyle='--', alpha = 0.2)
         # ax.set_yticks(list(plt.yticks()[0])+[100])
         axis_pars(ax)
         # fill_error(ax, h_f*pc_kpc/cm_kpc, h_err*pc_kpc/cm_kpc)
         ax.set_xlabel(r'Radius (kpc)', fontsize=fs)
         ax.set_ylabel(r'Length scale (pc)', fontsize=fs)
         ax.axhline(y=0, color='black', linestyle='--', alpha = 0.2)

         print(l_f*pc_kpc/cm_kpc)
         # plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M
```

```
[12.05609414 13.17015035 14.51697052 15.87397173 17.6956794  20.37664876
 21.54849053 23.81956155 26.5689569  29.73336505 33.51357813 36.36620393
 41.79375887 50.34642486 64.20158927]
```

## plotting u

```
In [ ]:  fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)
         ax.plot(kpc_r, u_f/cm_km, color='tab:orange', marker='o', mfc='k',
                     mec='k', markersize=m, label=r'$u$')
         fill_error(ax, u_f/cm_km,u_err/cm_km, 'tab:orange', 0.5)

         ax.plot(kpc_r, alphak_f/cm_km, color='b', marker='o',
                     mfc='k', mec='k', markersize=m, label=r'$\alpha_k$')
         fill_error(ax, alphak_f/cm_km,alphak_err/cm_km, 'blue')

         ax.plot(kpc_r, alpham_f/cm_km, color='r', marker='o',
                     mfc='k', mec='k', markersize=m, label=r'$\alpha_m$')
         ax.plot(kpc_r, alphasat_f/cm_km, color='m', marker='o',
                     mfc='k', mec='k', markersize=m, label=r'$\alpha_{sat}$')

         sig = np.sqrt(u_f**2 + (cs_f)**2)
         ax.plot(kpc_r, sig /
                     cm_km, color='r', marker='o', mfc='k', mec='k', markersize=m, label=r
         fill_error(ax, sig /cm_km,np.sqrt((u_f*u_err)**2 + (cs_f*cs_err)**2)/(sig*cm_km), '

         ax.plot(kpc_r, cs_f /
                     cm_km, color='g', linestyle='--', label=r'$c_s$', alpha = 0.5)
         fill_error(ax, cs_f/cm_km,cs_err/cm_km, 'green')

         ax.plot(corrected_radius, dat_u,
                     c='y', linestyle='--', label='Without warp',alpha = 1,marker='*',mfc=
```

```python
ax.axhline(y=0, color='black', linestyle='--', alpha = 0.2)


#next line is for warped u data for M31
# ax.plot(kpc_r, dat_u_warp,
#                 c='tab:cyan',  linestyle='dashdot', label='With warp', alpha = 0.3,
axis_pars(ax)



uf=u_f/cm_km
csf=cs_f/cm_km
sigf=sig /cm_km
# index_sup = np.where(kpc_r== 8.75)
# index_sub = np.where(kpc_r== 14.25)
# print('sup u',uf[index_sup],kpc_r[index_sup])
# print(uf[index_sub],kpc_r[index_sub])
# print('sup cs',csf[index_sup],kpc_r[index_sup])
# print(csf[index_sub],kpc_r[index_sub])
# print('sup sig',sigf[index_sup],kpc_r[index_sup])
# print(sigf[index_sub],kpc_r[index_sub])
# print('**************************************************************************
akf=alphak_f/cm_km
amf=alpham_f/cm_km
asf=alphasat_f/cm_km
# index_sup = np.where(kpc_r== 8.75)
# index_sub = np.where(kpc_r== 14.25)
# print('sup ak',akf[index_sup],kpc_r[index_sup])
# print(akf[index_sub],kpc_r[index_sub])
# print('sup am',amf[index_sup],kpc_r[index_sup])
# print(amf[index_sub],kpc_r[index_sub])
# print('sup as',asf[index_sup],kpc_r[index_sup])
# print(asf[index_sub],kpc_r[index_sub])
# print('**************************************************************************

print('uf',uf)
print('cs',csf)
print('sig',sigf)
print('**************************************************************************
print('alpha_k',akf)
print('alpha_m',amf)
print('alpha_s',asf)


#ax.set_ylim(0)
ax.set_xlabel(r'Radius ($kpc$)', fontsize=fs)
ax.set_ylabel(r'Speed (km/s)',  fontsize=fs)

# plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M

# print(sigf)
```

```
uf [5.17201194 5.02399899 5.18900902 4.79453905 4.44012556 5.12680227
 4.59711938 4.36088845 4.27720292 3.99019149 4.10172605 3.48600095
 3.10267534 2.78868172 2.89456643]
cs [6.28401545 6.37906034 6.47058056 6.56299656 6.6519866  6.73980174
 6.82857515 6.91414785 7.00071078 7.0842045  7.16672556 7.24830719
 7.33092546 7.4106997  7.49152729]
sig [8.13870737 8.11991236 8.29422856 8.12776282 7.99772722 8.46811839
 8.23182514 8.17452069 8.20392688 8.13065689 8.25748823 8.04301932
 7.96046873 7.91803105 8.03128233]
*********************************************************************************
***********************
alpha_k [0.34399273 0.27107423 0.1826586  0.21427354 0.20759698 0.12233038
 0.12939264 0.11058642 0.0942951  0.0838828  0.07297151 0.06340953
 0.05995034 0.06241546 0.06997556]
alpha_m [-0.12395924 -0.13729891 -0.10369931 -0.1523929  -0.1718699  -0.10735501
 -0.1190702  -0.10467016 -0.09123278 -0.08224179 -0.07208809 -0.06303043
 -0.05975104 -0.06230262 -0.06988737]
alpha_s [2.20033487e-01 1.33775321e-01 7.89592938e-02 6.18806347e-02
 3.57270892e-02 1.49753713e-02 1.03224396e-02 5.91625919e-03
 3.06232770e-03 1.64101219e-03 8.83422913e-04 3.79102613e-04
 1.99296999e-04 1.12839701e-04 8.81892255e-05]
```

Out[ ]:   Text(0, 0.5, 'Speed (km/s)')
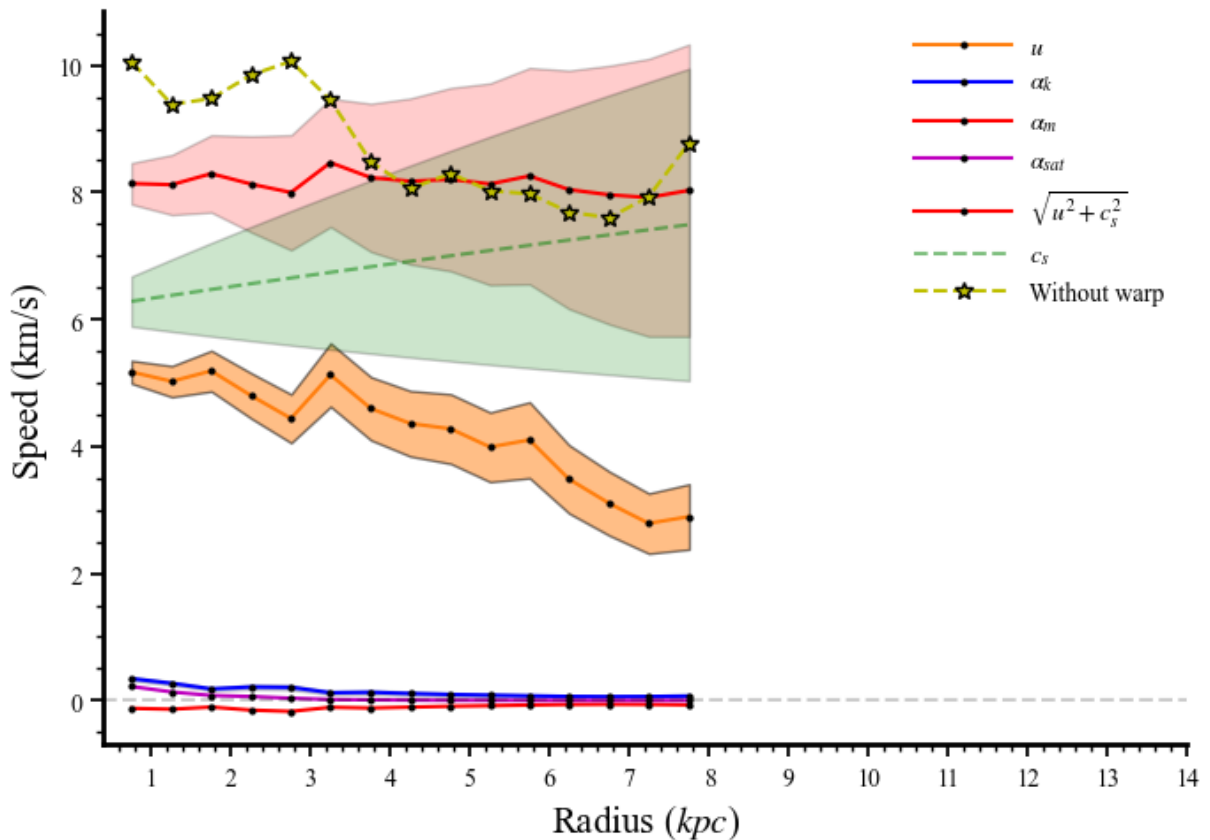


```
In [ ]:  fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)
         ax.plot(kpc_r, (u_f/cm_km)/(cs_f /cm_km), color='tab:green', marker='o', mfc='k',
                     mec='k', markersize=m, label=r'model u')
         ax.plot(kpc_r, (dat_u)/(cs_f /cm_km), color='tab:orange', marker='o', mfc='k',
                     mec='k', markersize=m, label=r'observed u')

         ax.axhline(y=1, color='black', linestyle='--', alpha = 0.2)
```
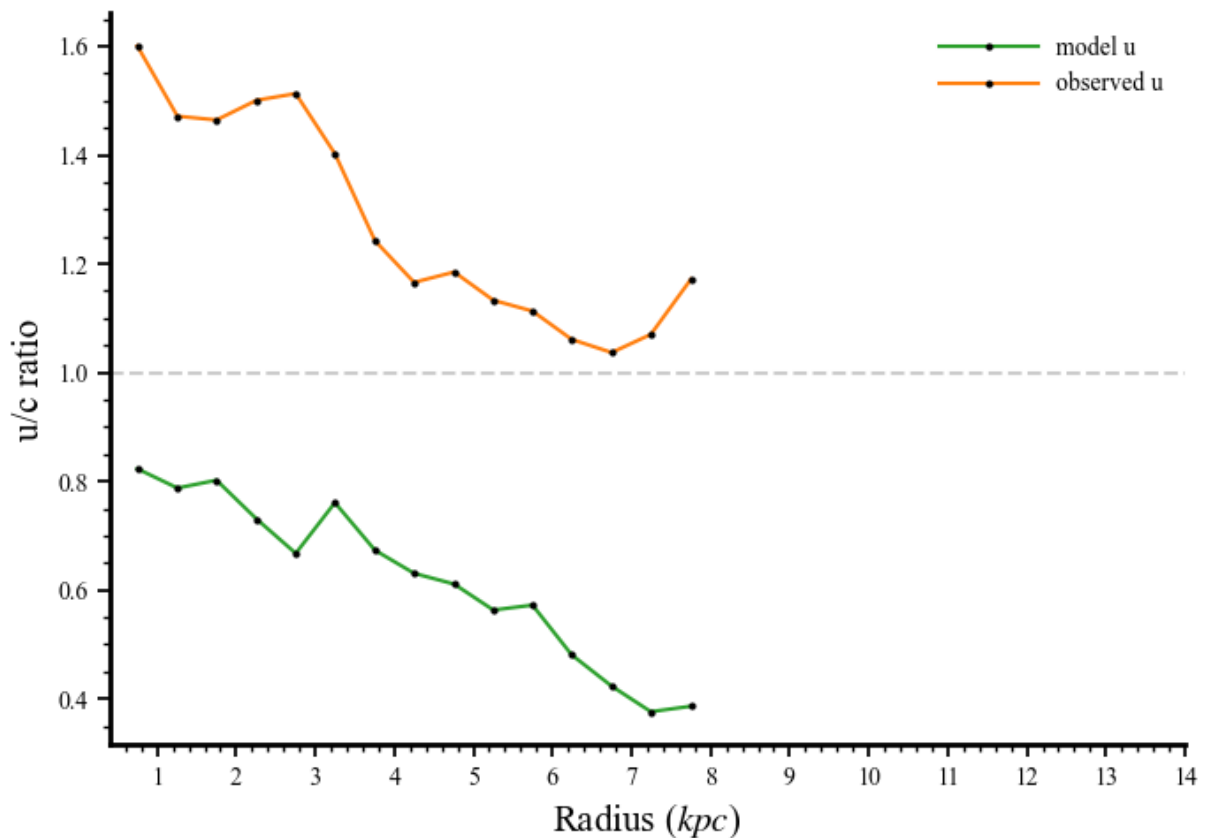
```
axis_pars(ax)

#ax.set_ylim(0)
ax.set_xlabel(r'Radius ($kpc$)', fontsize=fs)
ax.set_ylabel(r'u/c ratio',  fontsize=fs)

# plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M
```

Out[ ]:  Text(0, 0.5, 'u/c ratio')



```
In [ ]: fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)

ax.plot(kpc_r, alphak_f/cm_km, color='b', marker='o',
            mfc='k', mec='k', markersize=m, label=r'$\alpha_k$')
fill_error(ax, alphak_f/cm_km,alphak_err/cm_km, 'blue')

ax.plot(kpc_r, alpham_f/cm_km, color='r', marker='o',
            mfc='k', mec='k', markersize=m, label=r'$\alpha_m$')
ax.plot(kpc_r, alphasat_f/cm_km, color='m', marker='o',
            mfc='k', mec='k', markersize=m, label=r'$\alpha_{sat}$')

axis_pars(ax)
ax.axhline(y=0, color='black', linestyle='--', alpha = 0.2)

# ax.set_ylim(0)
ax.set_xlabel(r'Radius ($kpc$)', fontsize=fs)
ax.set_ylabel(r'$\alpha$ (km/s)',  fontsize=fs)

# plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M
```
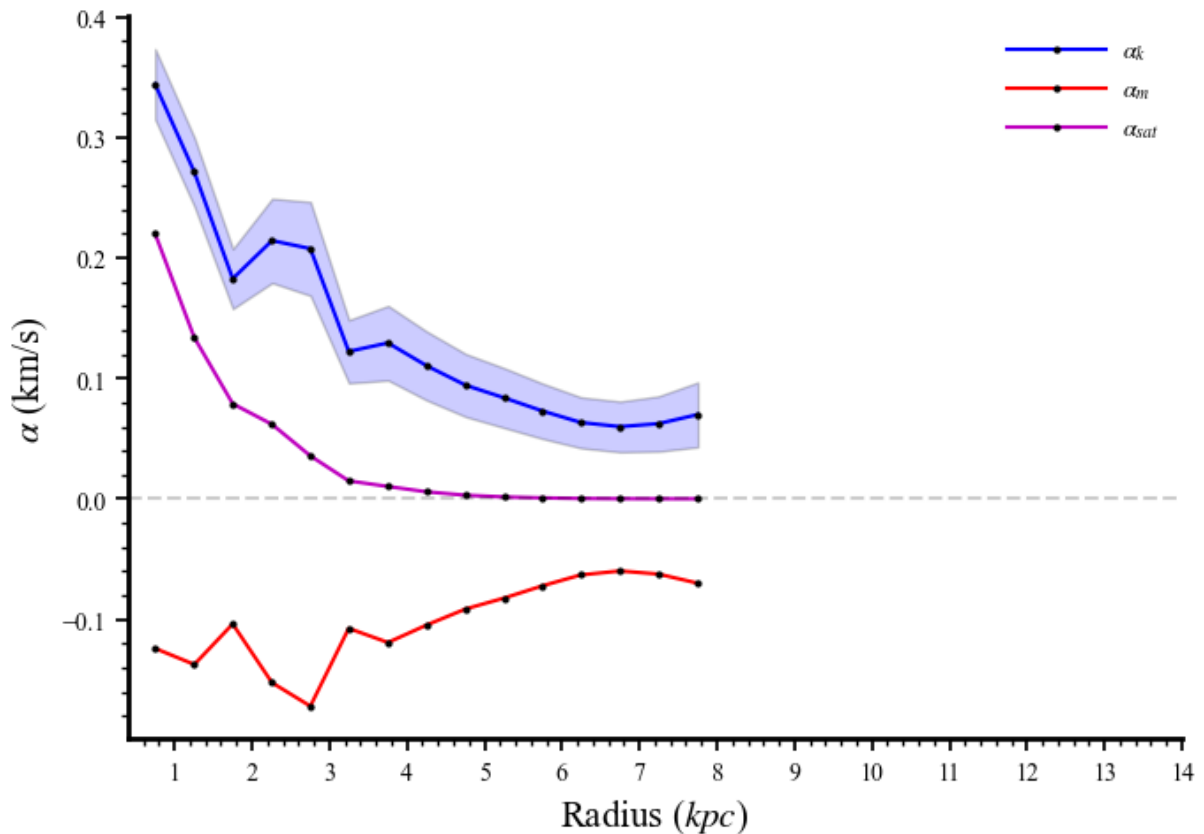
Out[ ]: Text(0, 0.5, '$\\alpha$ (km/s)')



## plotting magnetic fields

```
In [ ]:  # without observational data

         # fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)
         # ax.plot(kpc_r, G_scal_Bbartot*1e+6, c='b', linestyle='-', marker='o', mfc='k', me
         #              markersize=m, label=r' $B_{tot}=\sqrt{\bar{B}^2+b_{iso}^2+b_{ani}^2
         # fill_error(ax, G_scal_Bbartot*1e+6,G_scal_Bbartot_err*1e+6, 'b')

         # ax.plot(kpc_r, G_scal_Bbarreg*1e+6, c='r', linestyle='-', marker='o',
         #              mfc='k', mec='k', markersize=m, label=r' $B_{reg} = \bar{B}$')
         # fill_error(ax, G_scal_Bbarreg*1e+6,G_scal_Bbarreg_err*1e+6, 'r', 0.2)


         # ax.plot(kpc_r, G_scal_Bbarord*1e+6, c='green', linestyle='-', marker='o', mfc='k'
         #              mec='k', markersize=m, label=r' $B_{ord} = \sqrt{\bar{B}^2+b_{ani}^
         # fill_error(ax, G_scal_Bbarord*1e+6,G_scal_Bbarord_err*1e+6, 'g', 0.2)

         # ax.axhline(y=0, color='black', linestyle='--', alpha = 0.2)

         # ax.set_xlabel(r'Radius ($kpc$)', fontsize=fs)
         # ax.xaxis.set_ticks(np.arange(4, 9, 1))
         # ax.xaxis.set_major_formatter(FormatStrFormatter('%g'))
         # ax.set_ylabel('Magnetic field strength ($\mu G$)', fontsize=fs)
         # axis_pars(ax)
```

```python
fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)
ax.plot(mrange, G_dat_Btot, c='b', linestyle='--', marker='*',mfc='yellow',mec='tab
ax.plot(kpc_r, G_scal_Bbartot*1e+6, c='b', linestyle='-', marker='o', mfc='k', mec=
            markersize=m, label=r' $B_{tot}=\sqrt{\bar{B}^2+b_{iso}^2+b_{ani}^2}$
fill_error(ax, G_scal_Bbartot*1e+6,G_scal_Bbartot_err*1e+6, 'b')

ax.plot(mrange, G_dat_Breg, c='r', linestyle='--', marker='*',mfc='yellow',mec='tab
ax.plot(kpc_r, G_scal_Bbarreg*1e+6, c='r', linestyle='-', marker='o',
            mfc='k', mec='k', markersize=m, label=r' $B_{reg} = \bar{B}$')
fill_error(ax, G_scal_Bbarreg*1e+6,G_scal_Bbarreg_err*1e+6, 'r', 0.2)


ax.plot(mrange, G_dat_Bord, c='g', linestyle='dotted', marker='*',mfc='yellow',mec=
ax.plot(kpc_r, G_scal_Bbarord*1e+6, c='green', linestyle='-', marker='o', mfc='k',
            mec='k', markersize=m, label=r' $B_{ord} = \sqrt{\bar{B}^2+b_{ani}^2}
fill_error(ax, G_scal_Bbarord*1e+6,G_scal_Bbarord_err*1e+6, 'g', 0.2)

ax.axhline(y=0, color='black', linestyle='--', alpha = 0.2)

ax.set_xlabel(r'Radius ($kpc$)', fontsize=fs)
ax.xaxis.set_ticks(np.arange(6, 9, 1))
ax.xaxis.set_major_formatter(FormatStrFormatter('%g'))
ax.set_ylabel('Magnetic field strength ($\mu G$)', fontsize=fs)
axis_pars(ax)

G_scal_Bbartot=G_scal_Bbartot*1e+6
G_scal_Bbarreg=G_scal_Bbarreg*1e+6
G_scal_Bbarord=G_scal_Bbarord*1e+6
index_sup = np.where(kpc_r== 8.75)
index_sub = np.where(kpc_r== 14.25)
print('G_scal_Bbartot',G_scal_Bbartot[index_sup],kpc_r[index_sup])
print(G_scal_Bbartot[index_sub],kpc_r[index_sub])
print('G_scal_Bbarreg',G_scal_Bbarreg[index_sup],kpc_r[index_sup])
print(G_scal_Bbarreg[index_sub],kpc_r[index_sub])
print('G_scal_Bbarord',G_scal_Bbarord[index_sup],kpc_r[index_sup])
print(G_scal_Bbarord[index_sub],kpc_r[index_sub])

print('G_scal_Bbartot',G_scal_Bbartot)
print('G_scal_Bbarreg',G_scal_Bbarreg)
print('G_scal_Bbarord',G_scal_Bbarord)

# plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M
```

```
G_scal_Bbartot [] []
[] []
G_scal_Bbarreg [] []
[] []
G_scal_Bbarord [] []
[] []
G_scal_Bbartot [26.66227223 22.94063034 20.55381884 16.84800963 13.52229332 12.77269
65
 10.62673104  8.77127214  7.41828367  5.95319762  5.20635698  4.01195217
  2.98913313  2.13491851  1.61070901]
G_scal_Bbarreg [1.76276031 1.68408904 1.37295082 1.31185988 1.2722729  1.14789508
 1.01399194 0.87934483 0.80096481 0.71541112 0.65812245 0.6167878
 0.53719876 0.46846067 0.39015687]
G_scal_Bbarord [5.57689509 5.20241425 4.19427821 3.91783366 3.58923138 3.1266438
 2.83483318 2.36763226 2.10184832 1.79412526 1.6332492  1.43598567
 1.16584924 0.93849774 0.74255335]
```
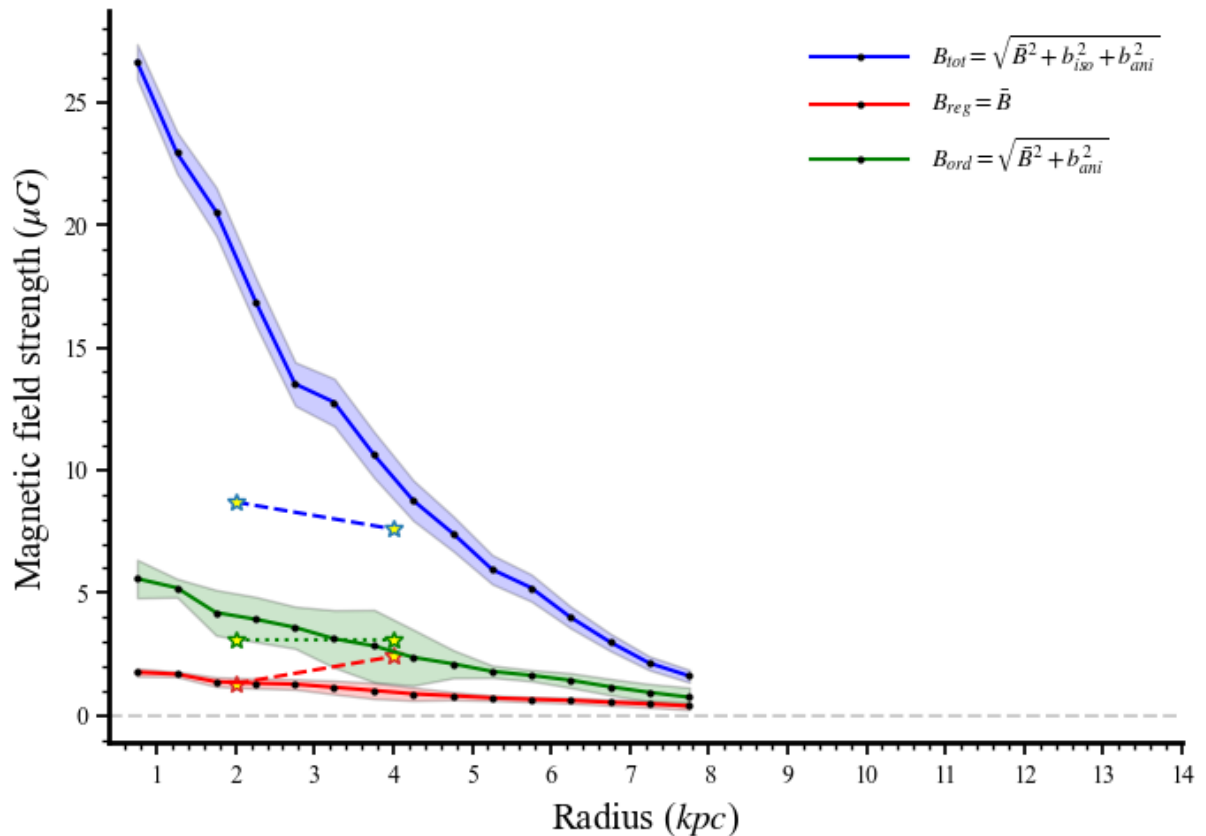


```
In [ ]:  fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)

         ###############################     M31     ###############################
         # ax.errorbar(mrange, 180*M_dat_pb/np.pi,elinewidth=1, yerr=180*err_M_dat_pb/np.pi,
         #                   c='r', linestyle='--', mfc='r', mec='k',label=r' $p_{B}$ data (
         # ax.errorbar(rmrange, 180*RM_dat_pb/np.pi,elinewidth=1, yerr=180*err_RM_dat_pb/np.
         #                   c='r', linestyle='--', mfc='r', mec='k',label=r' $p_{B}$ data (
         # ax.errorbar(rmrange, 180*RM_dat_po/np.pi,elinewidth=1, yerr=180*err_RM_dat_po/np.
         #                   c='g', linestyle='--', marker='*', mfc='g', mec='k',ecolor='k')

         # ax.plot(kpc_r, 180*pB/np.pi, c='r', linestyle='-', marker='o',
         #              markersize=m, mfc='k', mec='k', label=r' $p_{B}$ (mean field)')
         # fill_error(ax, 180*pB/np.pi,180*pB_err/np.pi, 'r')
```

```
# ax.plot(kpc_r, 180*pbb/np.pi,c = 'r',linestyle='--', marker='o',label = r' $p_{b}
# ax.plot(kpc_r, 180*pbo/np.pi, c='g', linestyle='-', mfc='k', markersize=m,mec='k'
# fill_error(ax, 180*pog/np.pi,180*pog_err/np.pi, 'g')
###############################         M31     ###############################

###############################         M33     ###############################
ax.errorbar(mrange, 180*M_dat_pb/np.pi,elinewidth=1, yerr=180*err_M_dat_pb/np.pi,ec
                c='r', linestyle='--', mfc='r', mec='k',label=r' $p_{B}$ data',ba
ax.errorbar(po_mrange, 180*RM_dat_po/np.pi,elinewidth=1, yerr=180*err_RM_dat_po/np.
                c='g', linestyle='--', marker='*', mfc='g', mec='k',ecolor='k')#,

ax.plot(kpc_r, 180*pB/np.pi, c='r', linestyle='-', marker='o',
            markersize=m, mfc='k', mec='k', label=r' $p_{B}$ (mean field)')
fill_error(ax, 180*pB/np.pi,180*pB_err/np.pi, 'r')
ax.plot(kpc_r, 180*pbb/np.pi,c = 'r',linestyle='--', marker='o',label = r' $p_{b}$
ax.plot(kpc_r, 180*pbo/np.pi, c='g', linestyle='-', mfc='k', markersize=m,mec='k',
fill_error(ax, 180*pog/np.pi,180*pog_err/np.pi, 'g')
###############################         M33     ###############################

###############################         M51     ###############################
# ax.errorbar(mrange, 180*M_dat_pb/np.pi,elinewidth=1, yerr=180*err_M_dat_pb/np.pi,
#                   c='r', linestyle='--', mfc='r', mec='k',label=r' $p_{B}$ data (
# # ax.errorbar(mrange, 180*M_dat_pb/np.pi,elinewidth=1, yerr=180*err_M_dat_pb/np.p
# #                   c='r', linestyle='--', mfc='r', mec='k',barsabove=True,marker
# ax.errorbar(range_po, 180*RM_dat_po/np.pi,elinewidth=1, yerr=180*err_RM_dat_po/np
#                   c='g', linestyle='--', marker='*', mfc='g', mec='k',ecolor='k')
# # ax.errorbar(range_po, 180*RM_dat_po/np.pi,elinewidth=1, yerr=180*err_RM_dat_po/
# #                   c='g', linestyle='--', marker='*', mfc='g',label=r' $p_{o}$ d
# # ax.errorbar(range_po, 180*RM_dat_po/np.pi,elinewidth=1, yerr=180*err_RM_dat_po/
# #                   c='g', linestyle='--', marker='*', mfc='g',label=r' $p_{o}$

# ax.plot(kpc_r, 180*pB/np.pi, c='r', linestyle='-', marker='o',
#               markersize=m, mfc='k', mec='k', label=r' $p_{B}$ (mean field)')
# fill_error(ax, 180*pB/np.pi,180*pB_err/np.pi, 'r')
# ax.plot(kpc_r, 180*pbb/np.pi,c = 'r',linestyle='--', marker='o',label = r' $p_{b}
# ax.plot(kpc_r, 180*pbo/np.pi, c='g', linestyle='-', mfc='k', markersize=m,mec='k'
# fill_error(ax, 180*pog/np.pi,180*pog_err/np.pi, 'g')
###############################         M51     ###############################

###############################      NGC 6946      ###############################
# # ax.errorbar(mrange, 180*M_dat_pb/np.pi,elinewidth=1, yerr=180*err_M_dat_pb/np.p
# #                   c='r', linestyle='--', mfc='r', mec='k',label=r' $p_{B}$ data
# # ax.errorbar(mrange, 180*M_dat_pb/np.pi,elinewidth=1, yerr=180*err_M_dat_pb/np.p
# #                   c='r', linestyle='--', mfc='r', mec='k',barsabove=True,marker

# print('no pB data for 6946 currently')
# ax.errorbar(range1, 180*RM_dat_po_range1/np.pi,elinewidth=1, yerr=180*err_RM_dat_
#                   c='g', linestyle=' ', marker='*', mfc='g',label=r' $p_{o}$ dat
# ax.errorbar(range2, 180*RM_dat_po_range2/np.pi,elinewidth=1, yerr=180*err_RM_dat_
#                   c='g', linestyle=' ', marker='*', mfc='g',label=r' $p_{o}$ dat

# ax.plot(kpc_r, 180*pB/np.pi, c='r', linestyle='-', marker='o',
#               markersize=m, mfc='k', mec='k', label=r' $p_{B}$ (mean field)')
# fill_error(ax, 180*pB/np.pi,180*pB_err/np.pi, 'r')
# ax.plot(kpc_r, 180*pbb/np.pi,c = 'r',linestyle='--', marker='o',label = r' $p_{b}
# ax.plot(kpc_r, 180*pbo/np.pi, c='g', linestyle='-', mfc='k', markersize=m,mec='k'
```

```python
# fill_error(ax, 180*pog/np.pi,180*pog_err/np.pi, 'g')
#################################    NGC 6946    #################################


pB=180*pB/np.pi
pbb=180*pbb/np.pi
po=180*pbo/np.pi
index_sup = np.where(kpc_r== 8.75)
index_sub = np.where(kpc_r== 14.25)
print('pB',pB[index_sup],kpc_r[index_sup])
print(pB[index_sub],kpc_r[index_sub])
print('pbb',pbb[index_sup],kpc_r[index_sup])
print(pbb[index_sub],kpc_r[index_sub])
print('po',po[index_sup],kpc_r[index_sup])
print(po[index_sub],kpc_r[index_sub])

axis_pars(ax)
ax.set_xlabel(r'Radius ($kpc$)', fontsize=fs)
ax.set_ylabel(r'Pitch angle (deg)', fontsize=fs)

# plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M
print('pB',pB)
print('pbb',pbb)
print('po',po)
```

```
pB [] []
[] []
pbb [] []
[] []
po [] []
[] []
pB [13.81553576  9.68307249  7.2000738   5.9249191   3.97754227  2.1490833
  1.69636396  1.20718937  0.76460668  0.50888486  0.32194442  0.18761994
  0.12553561  0.08573704  0.06921851]
pbb [43.33235313 43.05440774 43.42377648 42.97151203 42.40619827 42.80901229
 42.38682544 42.35353705 42.11169801 41.78707972 41.5365651  40.61823448
 39.96338055 38.86678241 38.45010913]
po [40.925335   40.36547114 40.58315978 39.9818643  39.0376261  39.16819259
 38.92036329 38.62680583 38.19901879 37.53137347 37.19845369 35.7538602
 34.45848978 32.57054427 31.6072775 ]
```
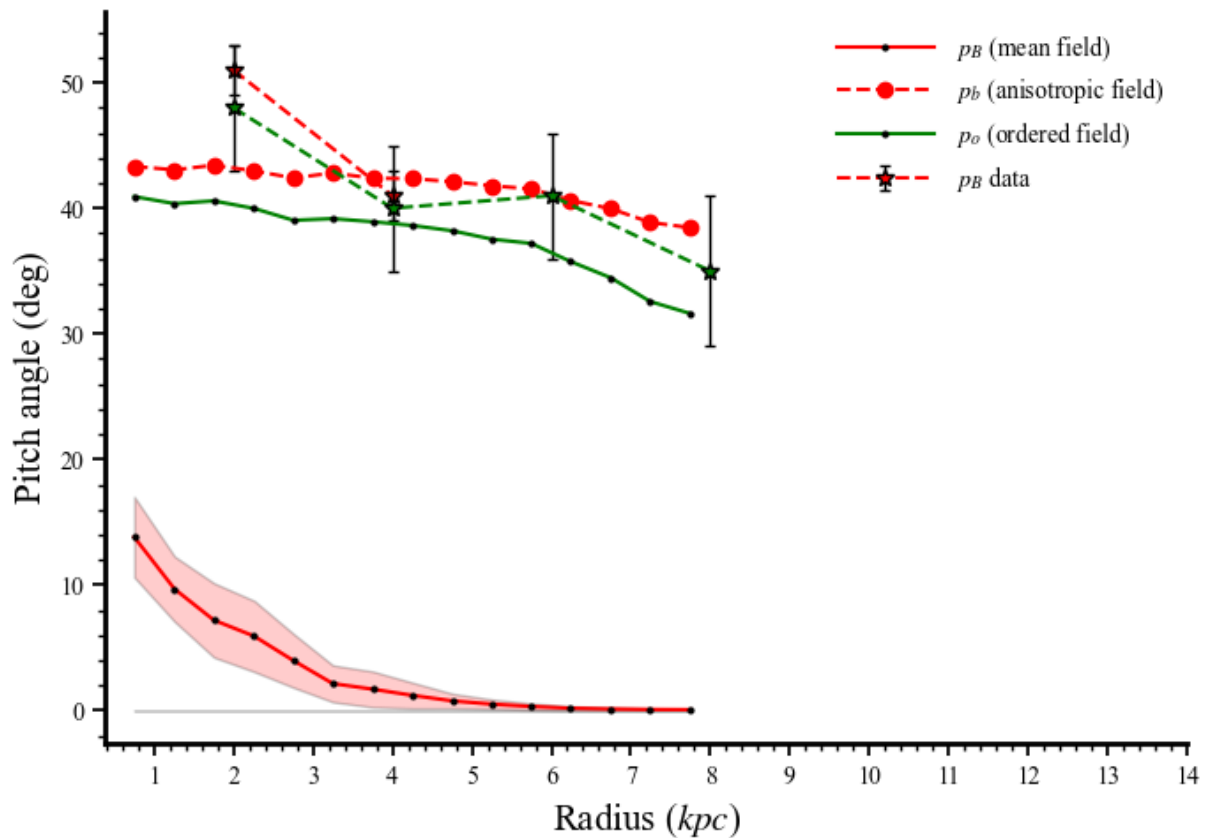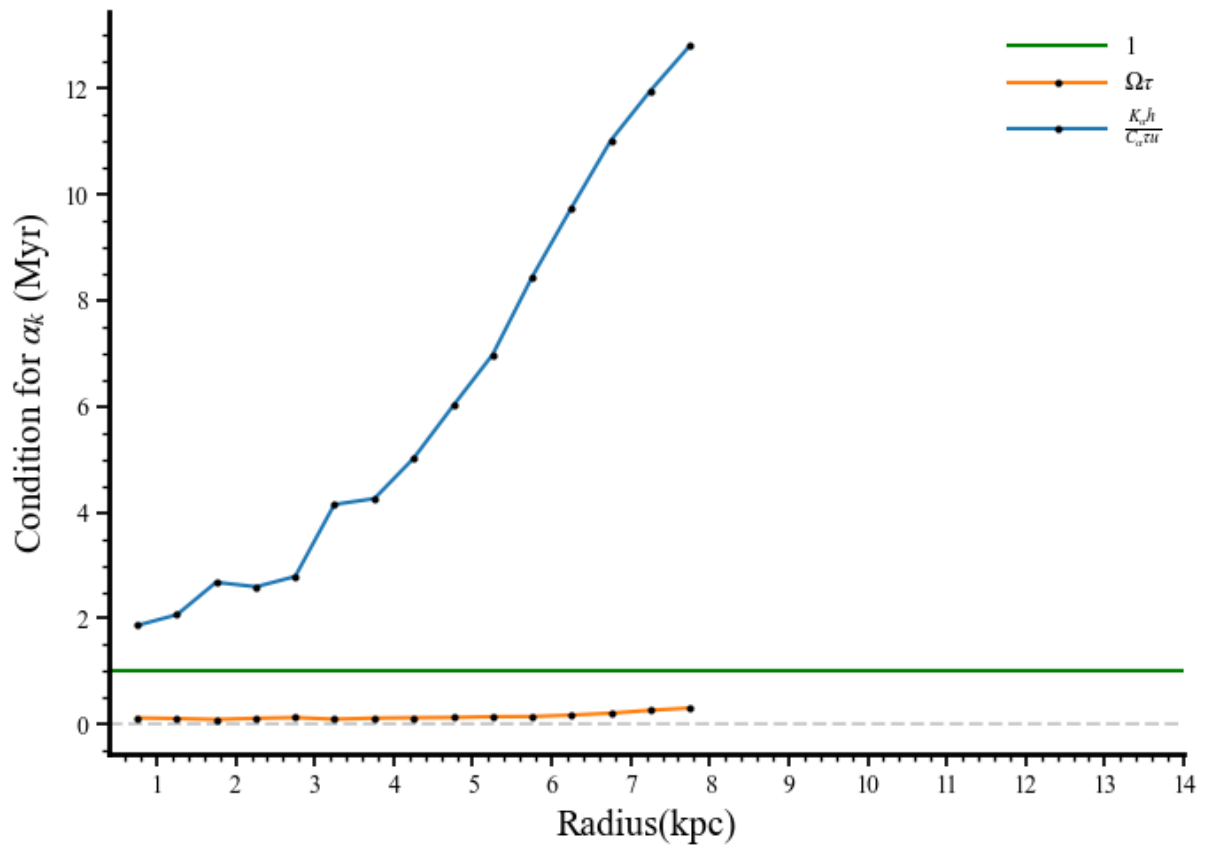
```
In [ ]:  fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)
         ax.axhline(y=1, color='g', linestyle='-', label=r'1')
         ax.plot(kpc_r, omt, marker='o', markersize=m,
                     c='tab:orange', mfc='k', mec='k', label=r'$\Omega\tau$')
         ax.plot(kpc_r, kah, marker='o',
                     markersize=m, c='tab:blue', mfc='k', mec='k', label=r'$\frac{K_\alpha
         ax.axhline(y=0, color='black', linestyle='--', alpha = 0.2)

         axis_pars(ax)
         ax.set_xlabel('Radius(kpc)', fontsize=fs)
         ax.set_ylabel(r'Condition for $\alpha_k$ (Myr)', fontsize=fs)




         index_sup = np.where(kpc_r== 8.75)
         index_sub = np.where(kpc_r== 14.25)
         print('omt',omt[index_sup],kpc_r[index_sup])
         print(omt[index_sub],kpc_r[index_sub])
         print('kah',kah[index_sup],kpc_r[index_sup])
         print(kah[index_sub],kpc_r[index_sub])
         # plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M

         omt [] []
         [] []
         kah [] []
         [] []
```

```
In [ ]:  fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)
         ax.plot(kpc_r, taue_f/s_Myr, c='b', markersize=m,
                     linestyle='-', marker='o', mfc='k', mec='k', label=r'$\tau^e$')
         ax.plot(kpc_r, taur_f/s_Myr, c='g',
                     markersize=m, linestyle='-', marker='o', mfc='k', mec='k', label=r'$\
         ax.plot(kpc_r, h_f/(u_f*s_Myr), c='y', markersize=m,
                     linestyle='-', marker='o', mfc='k', mec='k', label=r'$h/u$')
         ax.set_xlabel('Radius(kpc)', fontsize=fs)
         ax.set_ylabel(r'Correlation Time $\tau$ (Myr)', fontsize=fs)
         axis_pars(ax)


         tauef=taue_f/s_Myr
         taurf=taur_f/s_Myr
         t=h_f/(u_f*s_Myr)
         index_sup = np.where(kpc_r== 8.75)
         index_sub = np.where(kpc_r== 14.25)
         print('tauef',tauef[index_sup],kpc_r[index_sup])
         print(tauef[index_sub],kpc_r[index_sub])
         print('taurf',taurf[index_sup],kpc_r[index_sup])
         print(taurf[index_sub],kpc_r[index_sub])
         print('t',t[index_sup],kpc_r[index_sup])
         print(t[index_sub],kpc_r[index_sub])


         # plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M
```
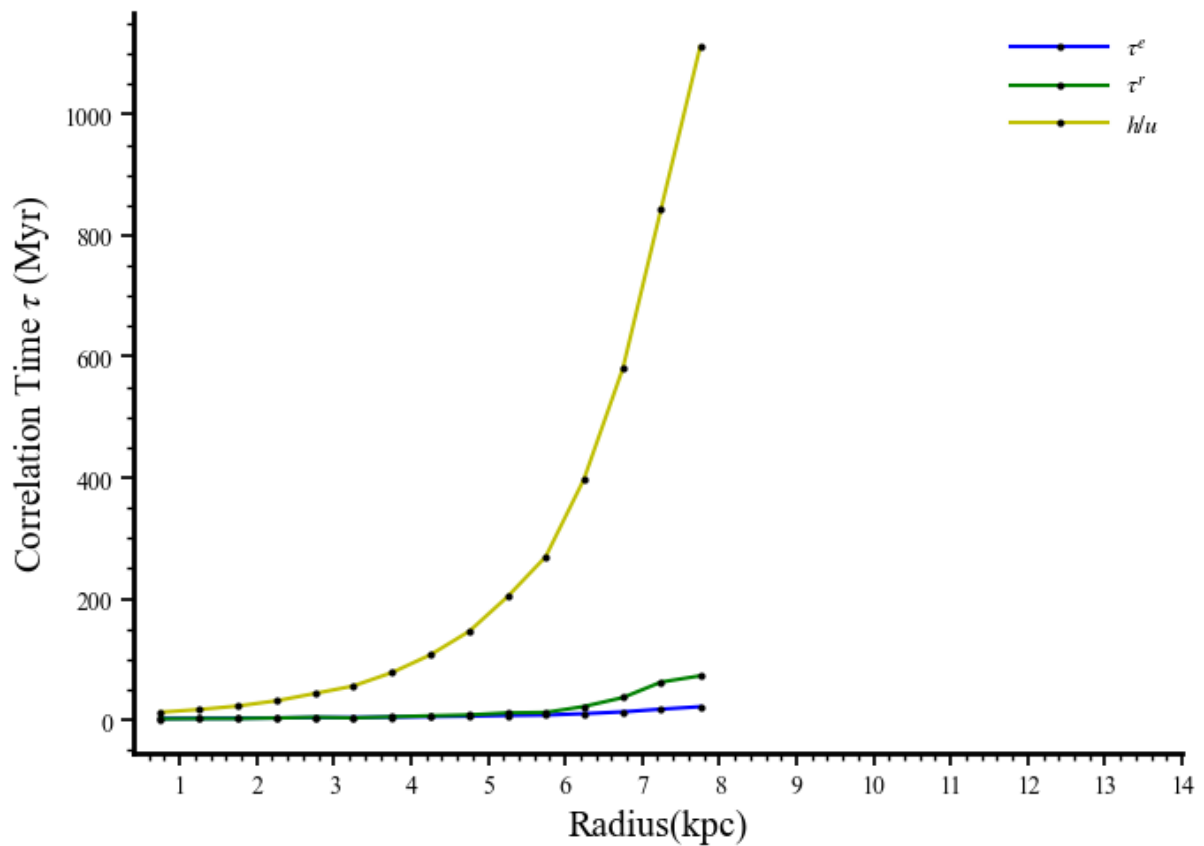
```
tauef [] []
[] []
taurf [] []
[] []
t [] []
[] []
```



```
In [ ]:  fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)

         ax.plot(kpc_r, dkdc_f, markersize=m, linestyle='-',
                       marker='o', mfc='k', mec='k', label=r'$D_k/D_c$')
         ax.plot(kpc_r, 1*np.ones(len(kpc_r)))
         ax.axhline(y=1, color='black', linestyle='--', alpha = 0.2)

         ax.set_xlabel('Radius(kpc)', fontsize=fs)
         ax.set_ylabel(r'$D_k/D_c$',  fontsize=fs)
         axis_pars(ax)

         index_sup = np.where(kpc_r== 8.75)
         index_sub = np.where(kpc_r== 14.25)
         print('dkdc_f',dkdc_f[index_sup],kpc_r[index_sup])
         print(dkdc_f[index_sub],kpc_r[index_sub])
         # plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M
```
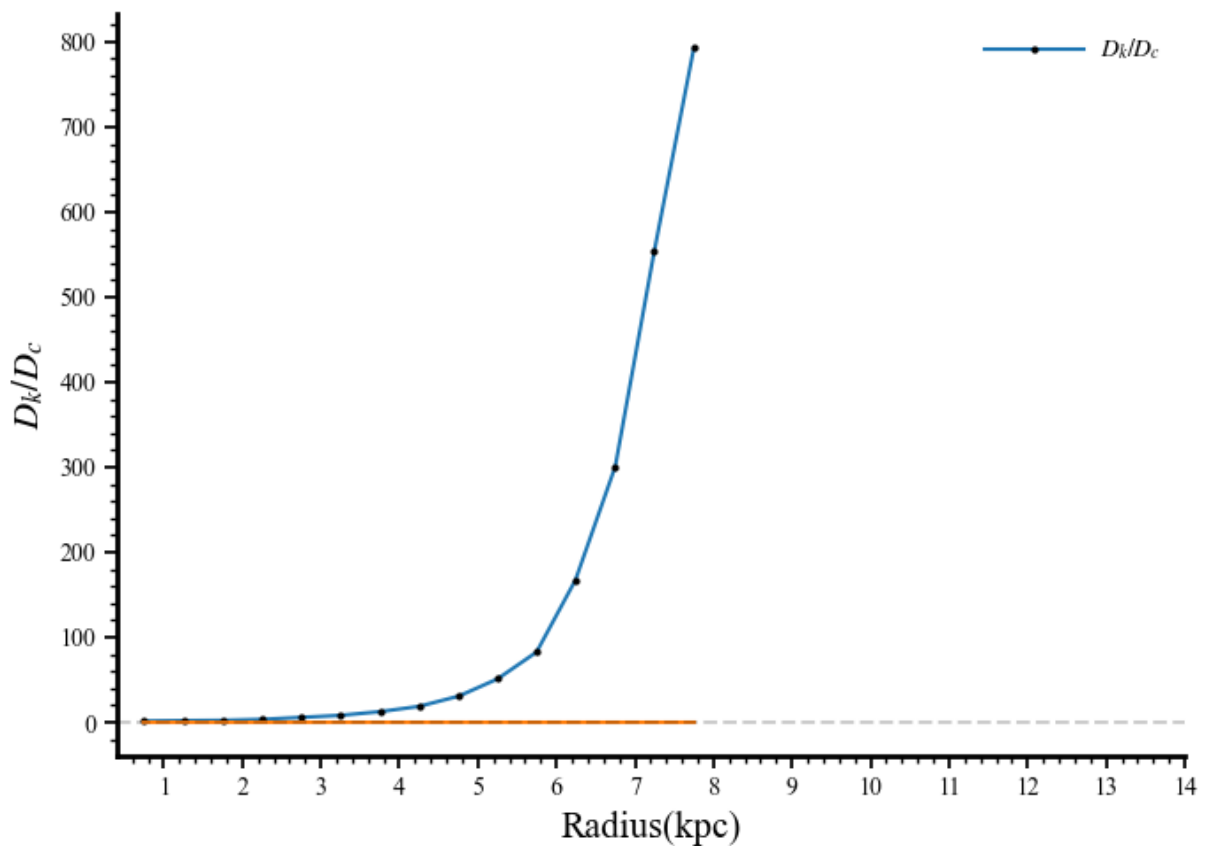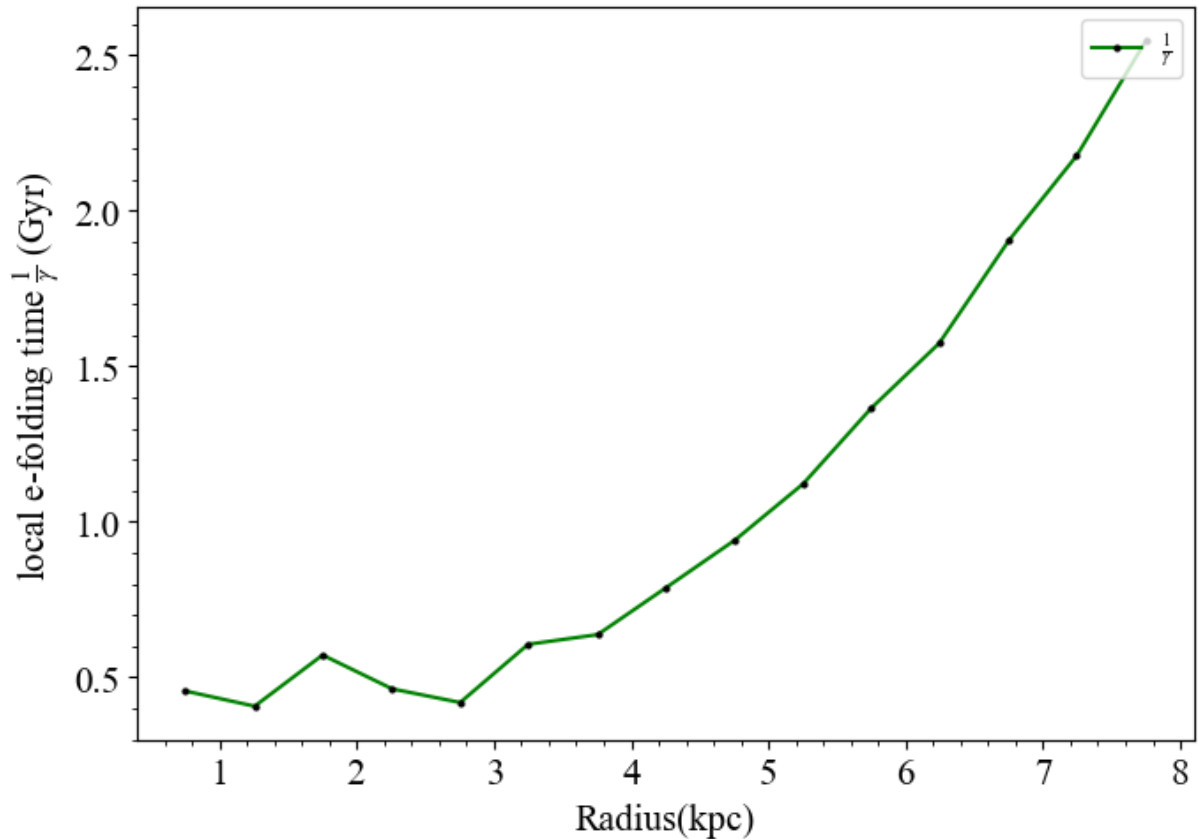
```
dkdc_f [] []
[] []
```

```
In [ ]:  fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)
         ax.plot(kpc_r, (((np.pi**2)*(tau_f*(u_f**2))/3*(np.sqrt(dkdc_f)-1)/(4*h_f**2))**(-1
                      (s_Myr*1e+3), c='g', markersize=m, linestyle='-', marker='o', mfc='k'
         ax.set_xlabel('Radius(kpc)', fontsize=fs)
         ax.set_ylabel(r'local e-folding time $\frac{1}{\gamma}$ (Gyr)', fontsize=fs)
         ax.legend(fontsize = lfs)


         # plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M
```

Out[ ]:  `<matplotlib.legend.Legend at 0x288e0902a40>`

```
In [ ]:  dat_sigmatot, dat_sigma, dat_sigmasfr, dat_q, dat_omega, zet, T_tb, psi, bet, ca, r
            np.array([data_pass[i][j] for i in range(len(kpc_r))]) for j in range(len(data_
         from helper_functions import g_Msun
         ks_const = (dat_sigmasfr/(dat_sigma)**(1.4)).mean()
         dat_sigmasfr2 = ks_const*(dat_sigma)**(1.4)
         dat_sigma3 = ((1/ks_const)*(dat_sigmasfr))**(1/1.4)
         fig,ax = plt.subplots(nrows=1, ncols=1, figsize=(7, 5), tight_layout=True)

         ax.plot(kpc_r, dat_sigmasfr*(cm_kpc**2)*(s_Myr/1e+6)/g_Msun, c='r',
                 linestyle='--', mfc='k', mec='k', markersize=m, marker='o', label=r'orig
         ax.plot(kpc_r, dat_sigmasfr2*(cm_kpc**2)*(s_Myr/1e+6)/g_Msun, c='b', linestyle='--'
                 mfc='k', mec='k', markersize=m, marker='o', label=r'$\Sigma_{SFR} \propt
         # ax.plot(kpc_r, dat_sigma3*(cm_kpc**2)*(s_Myr/1e+6)/g_Msun, c='g', linestyle='--',
         #           mfc='k', mec='k', markersize=m, marker='o', label=r'$\Sigma \propto \S
         ax.set_xlabel('Radius(kpc)',fontsize=fs)
         ax.set_ylabel(r'$\Sigma_{SFR}$ ($\mathrm{M_\odot\,yr^{-1}kpc^{-2}}$)',fontsize=fs)
         axis_pars(ax)

         # plt.savefig(r'D:\Documents\Gayathri_college\MSc project\codes\MSc.-Thesis\plots\M
```
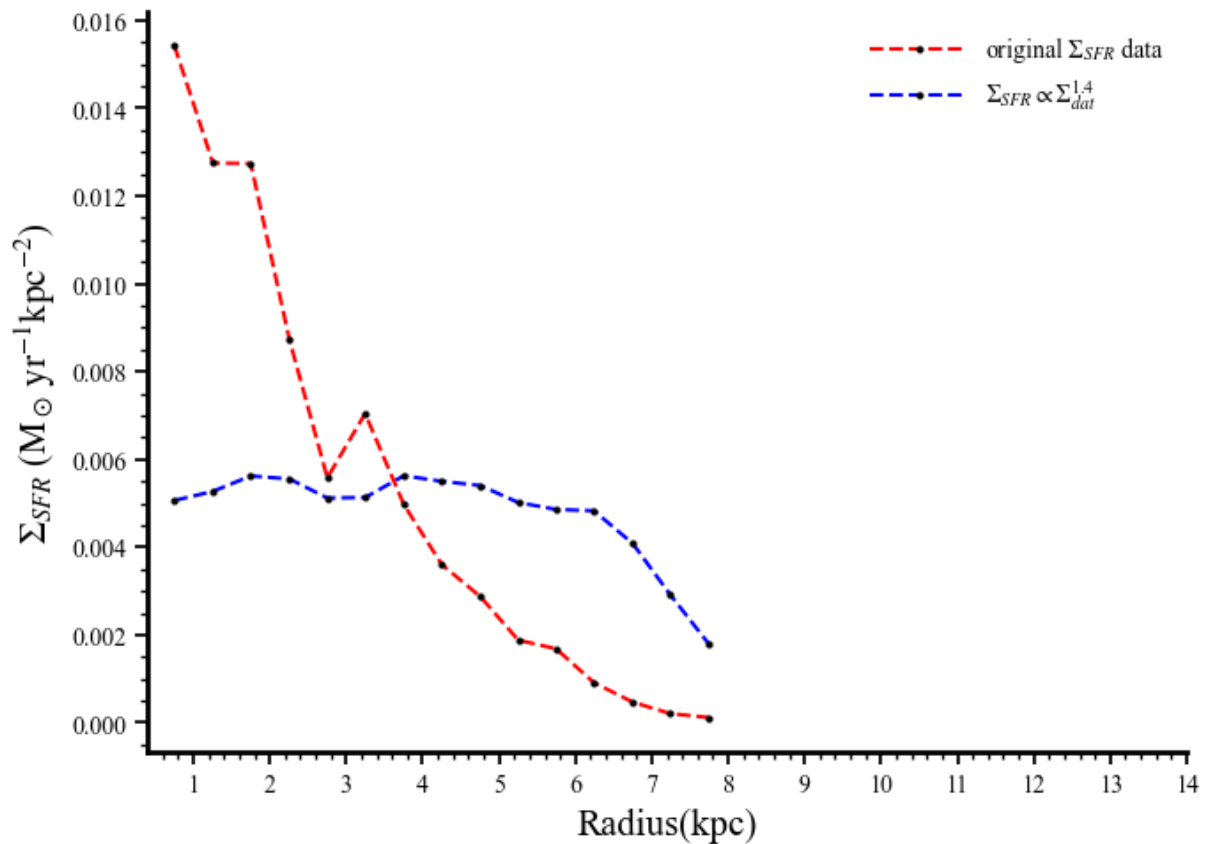
```
In [ ]:  # os.chdir(current_directory+'\plots')

         # #change pdf name here for different galaxies

         # from matplotlib.backends.backend_pdf import PdfPages
         # PDF = PdfPages('m33_model3_ca_'+str(params[r'C_\alpha'])+'rk_'+str(params[r'R_\ka
         #         params[r'\zeta'])+'psi_'+str(params[r'\psi'])+'b_'+str(params[r'\beta'])+'.p
         # fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(10, 10), tight_layout=True)
         # i = 0
         # j = 0
         # ax[i][j].plot(kpc_r, h_f*pc_kpc/cm_kpc, c='r', linestyle='-', mfc='k',
         #                mec='k', markersize=m, marker='o', label=r' $h$(pc)')
         # ax[i][j].plot(kpc_dat_r, pc_dat_h, c='b', linestyle='dotted',
         #                marker='*',mfc='y',mec='b',mew=1, markersize = 7, label=r'Fiducial
         # ax[i][j].plot(kpc_r, l_f*pc_kpc/cm_kpc, c='g',
         #                linestyle='-', mfc='k', mec='k', markersize=m, marker='o', label=r'
         # # ax[i][j].plot(kpc_r, datamaker(lsn , data_pass, h_f, tau_f)*pc_kpc/cm_kpc,c = '
         # ax[i][j].axhline(y=100, color='black', linestyle='--', alpha = 0.2)
         # ax[i][j].set_yticks(list(plt.yticks()[0])+[100])
         # axis_pars(ax[i][j])
         # fill_error(ax[i][j], h_f*pc_kpc/cm_kpc, h_err*pc_kpc/cm_kpc)
         # ax[i][j].set_xlabel(r'Radius (kpc)', fontsize=fs)
         # ax[i][j].set_ylabel(r'Length scale (pc)', fontsize=fs)


         # i = 0
         # j = 1
         # ax[i][j].plot(kpc_r, u_f/cm_km, color='tab:orange', marker='o', mfc='k',
         #                mec='k', markersize=m, label=r'$u$')
```

```python
# fill_error(ax[i][j], u_f/cm_km,u_err/cm_km, 'tab:orange', 0.5)

# ax[i][j].plot(kpc_r, alphak_f/cm_km, color='b', marker='o',
#                mfc='k', mec='k', markersize=m, label=r'$\alpha_k$')
# fill_error(ax[i][j], alphak_f/cm_km,alphak_err/cm_km, 'blue')

# ax[i][j].plot(kpc_r, alpham_f/cm_km, color='r', marker='o',
#                mfc='k', mec='k', markersize=m, label=r'$\alpha_m$')
# ax[i][j].plot(kpc_r, alphasat_f/cm_km, color='m', marker='o',
#                mfc='k', mec='k', markersize=m, label=r'$\alpha_{sat}$')

# sig = np.sqrt(u_f**2 + cs_f**2)
# ax[i][j].plot(kpc_r, sig /
#                cm_km, color='r', marker='o', mfc='k', mec='k', markersize=m, label
# fill_error(ax[i][j], sig /cm_km,np.sqrt((u_f*u_err)**2 + (cs_f*cs_err)**2)/(sig*c

# ax[i][j].plot(kpc_r, cs_f /
#                cm_km, color='g', linestyle='--', label=r'$c_s$', alpha = 0.5)
# fill_error(ax[i][j], cs_f/cm_km,cs_err/cm_km, 'green')

# ax[i][j].plot(kpc_r, dat_u/cm_km,
#                c='y', linestyle='--', label='Without warp',alpha = 1,marker='*',mf
# ,mec='k',mew=1, markersize = 7)
# # ax[i][j].plot(kpc_r, dat_u_warp/cm_km,
# #                c='tab:cyan',  linestyle='dashdot', label='With warp', alpha = 0.
# # ,mec='k',mew=1, markersize = 7)
# axis_pars(ax[i][j])

# ax[i][j].set_ylim(0)
# ax[i][j].set_xlabel(r'Radius ($kpc$)', fontsize=fs)
# ax[i][j].set_ylabel(r'Speed (km/s)',  fontsize=fs)

# i = 1

# ###########################################   M31   ####################
# # ax[i][j].errorbar(rmrange, 180*RM_dat_pb/np.pi,elinewidth=1, yerr=180*err_RM_da
# #                c='r', linestyle='--', mfc='r', mec='k',barsabove=True,marker
# # ax[i][j].errorbar(rmrange, 180*RM_dat_po/np.pi,elinewidth=1, yerr=180*err_RM_da
# #                c='g', linestyle='--', marker='*', mfc='g', mec='k',ecolor='k
# ###########################################   M31   ####################

# ###########################################   M33   ####################
# ax[i][j].errorbar(mrange, 180*M_dat_pb/np.pi,elinewidth=1, yerr=180*err_M_dat_pb/
#                c='r', linestyle='--', mfc='r', mec='k',barsabove=True,marker='
# ax[i][j].errorbar(po_mrange, 180*RM_dat_po/np.pi,elinewidth=1, yerr=180*err_RM_da
#                c='g', linestyle='--', marker='*', mfc='g', mec='k',ecolor='k')
# ###########################################   M33   ####################

# ###########################################   6946   ####################
# # ax[i][j].errorbar(range1, 180*RM_dat_po_range1/np.pi,elinewidth=1, yerr=180*err
# #                c='r', linestyle='--', mfc='r', mec='k',barsabove=True,marker
# # ax[i][j].errorbar(range2, 180*RM_dat_po_range2/np.pi,elinewidth=1, yerr=180*err
# #                c='g', linestyle='--', marker='*', mfc='g', mec='k',ecolor='k
# ###########################################   6946   ####################

# ax[i][j].plot(kpc_r, 180*pB/np.pi, c='r', linestyle='-', marker='o',
```

```python
#                markersize=m, mfc='k', mec='k', label=r' $p_{B}$ (mean field)')
# fill_error(ax[i][j], 180*pB/np.pi,180*pB_err/np.pi, 'r')
# ax[i][j].plot(kpc_r, 180*pbb/np.pi,c = 'r',linestyle='--', marker='o',label = r'
# ax[i][j].plot(kpc_r, 180*pog/np.pi, c='g', linestyle='-', mfc='k', markersize=m,

#                mec='k', marker='o', label=r' $p_{o}$ (ordered field)')
# fill_error(ax[i][j], 180*pog/np.pi,180*pog_err/np.pi, 'g')

# axis_pars(ax[i][j])
# ax[i][j].set_xlabel(r'Radius ($kpc$)', fontsize=fs)
# ax[i][j].set_ylabel(r'Pitch angle (deg)', fontsize=fs)

# j= 0
# ax[i][j].plot(mrange, G_dat_Btot, c='b', linestyle='--', marker='*',mfc='yellow',
# ='tab:blue',mew=1,markersize = 7)#, label='Average Binned data $B_{tot}$ ($\mu G$
# ax[i][j].plot(kpc_r, G_scal_Bbartot*1e+6, c='b', linestyle='-', marker='o', mfc='
#                markersize=m, label=r' $B_{tot}=\sqrt{\bar{B}^2+b_{iso}^2+b_{ani}^2
# fill_error(ax[i][j], G_scal_Bbartot*1e+6,G_scal_Bbartot_err*1e+6, 'b')

# ax[i][j].plot(mrange, G_dat_Breg, c='r', linestyle='--', marker='*',mfc='yellow',
# ='tab:red',mew=1,markersize = 7)#, label='Average Binned data $B_{reg}$ ($\mu G$)
# ax[i][j].plot(kpc_r, G_scal_Bbarreg*1e+6, c='r', linestyle='-', marker='o',
#                mfc='k', mec='k', markersize=m, label=r' $B_{reg} = \bar{B}$')
# fill_error(ax[i][j], G_scal_Bbarreg*1e+6,G_scal_Bbarreg_err*1e+6, 'r', 0.2)


# ax[i][j].plot(mrange, G_dat_Bord, c='g', linestyle='dotted', marker='*',mfc='yell
# ,mec='green',mew=1,markersize = 7)#, label='Average Binned data $B_{ord}$ ($\mu G
# ax[i][j].plot(kpc_r, G_scal_Bbarord*1e+6, c='green', linestyle='-', marker='o', m
#                mec='k', markersize=m, label=r' $B_{ord} = \sqrt{\bar{B}^2+b_{ani}^
# fill_error(ax[i][j], G_scal_Bbarord*1e+6,G_scal_Bbarord_err*1e+6, 'g', 0.2)


# ax[i][j].set_xlabel(r'Radius ($kpc$)', fontsize=fs)
# ax[i][j].xaxis.set_ticks(np.arange(1, 7, 2))
# ax[i][j].xaxis.set_major_formatter(FormatStrFormatter('%g'))
# ax[i][j].set_ylabel('Magnetic field strength ($\mu G$)', fontsize=fs)
# axis_pars(ax[i][j])


# PDF.savefig(fig)
# fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(10, 10), tight_layout=True)

# i = 0

# ax[i][j].axhline(y=1, color='g', linestyle='-', label=r'1')
# ax[i][j].plot(kpc_r, omt, marker='o', markersize=m,
#                c='tab:orange', mfc='k', mec='k', label=r'$\Omega\tau$')
# ax[i][j].plot(kpc_r, kah, marker='o',
#                markersize=m, c='tab:blue', mfc='k', mec='k', label=r'$\frac{K_\alp
# axis_pars(ax[i][j])
# ax[i][j].set_xlabel('Radius(kpc)', fontsize=fs)
# ax[i][j].set_ylabel(r'Condition for $\alpha_k$ (Myr)', fontsize=fs)


# j = 1
```

```
# ax[i][j].plot(kpc_r, taue_f/s_Myr, c='b', markersize=m,
#               linestyle='-', marker='o', mfc='k', mec='k', label=r'$\tau^e$')
# ax[i][j].plot(kpc_r, taur_f/s_Myr, c='g',
#               markersize=m, linestyle='-', marker='o', mfc='k', mec='k', label=r'
# ax[i][j].plot(kpc_r, h_f/(u_f*s_Myr), c='y', markersize=m,
#               linestyle='-', marker='o', mfc='k', mec='k', label=r'$h/u$')
# ax[i][j].set_xlabel('Radius(kpc)', fontsize=fs)
# ax[i][j].set_ylabel(r'Correlation Time $\tau$ (Myr)', fontsize=fs)
# axis_pars(ax[i][j])

# i = 1
# ax[i][j].plot(kpc_r, dkdc_f, markersize=m, linestyle='-',
#               marker='o', mfc='k', mec='k', label=r'$D_k/D_c$')
# ax[i][j].plot(kpc_r, 1*np.ones(len(kpc_r)))
# ax[i][j].set_xlabel('Radius(kpc)', fontsize=fs)
# ax[i][j].set_ylabel(r'$D_k/D_c$',  fontsize=fs)
# axis_pars(ax[i][j])

# j = 0
# ax[i][j].plot(kpc_r, (((np.pi**2)*(tau_f*(u_f**2))/3*(np.sqrt(dkdc_f)-1)/(4*h_f**
#               (s_Myr*1e+3), c='g', markersize=m, linestyle='-', marker='o', mfc='
# ax[i][j].set_xlabel('Radius(kpc)', fontsize=fs)
# ax[i][j].set_ylabel(r'local e-folding time $\gamma$ (Gyr)', fontsize=fs)
# ax[i][j].legend(fontsize = lfs)

# PDF.savefig(fig)
# PDF.close()
# os.chdir(current_directory)
```

In [ ]:  `os.chdir(current_directory)`