In [1]:
```
%matplotlib inline
import sys, os
import numpy as np
import matplotlib.pyplot as plt
sys.path.append(os.environ['rapp'])
sys.path.append(os.environ['raco'])
from rayleigh_diagnostics import Shell_Spectra, Shell_Slices, GridInfo
from reference_tools import equation_coefficients
import common
sys.path
```

Out[1]:
```
['/pleiades/home1/lmatilsk/sf_test',
 '/home1/lmatilsk/miniconda3/envs/custom/lib/python37.zip',
 '/home1/lmatilsk/miniconda3/envs/custom/lib/python3.7',
 '/home1/lmatilsk/miniconda3/envs/custom/lib/python3.7/lib-dynload',
 '',
 '/home1/lmatilsk/miniconda3/envs/custom/lib/python3.7/site-packages',
 '/home1/lmatilsk/miniconda3/envs/custom/lib/python3.7/site-packages/IPyt
hon/extensions',
 '/lou/la2/lmatilsk/.ipython',
 '/home1/lmatilsk/rayleigh/code/post_processing',
 '/home1/lmatilsk/rayleigh/utils/compute',
 '/home1/lmatilsk/rayleigh/code/post_processing']
```

In [2]:
```
#dirname = '/pleiades/nobackupp17/lmatilsk/case_M-Prm4.0_linear/00_test_e
#dirtag = 'caseM_orig'

#dirname = '/pleiades/nobackupp17/lmatilsk/case_M-Prm4.0_linear/00_test_e
#dirtag = 'caseM_dealias2'

#dirname = '/pleiades/nobackupp17/lmatilsk/case_M-Prm4.0_linear_consteta/
#dirtag = 'caseM_consteta'

dirname = '/pleiades/nobackupp17/lmatilsk/benchmarks/j2011_mhd_linear/00_
dirtag = 'j2011_mhd'

print (dirname)
```

```
/pleiades/nobackupp17/lmatilsk/benchmarks/j2011_mhd_linear/00_test_equati
ons_sf2/
```

In [3]:
```
radatadir = dirname + 'Shell_Spectra/'
the_file = radatadir + os.listdir(radatadir)[0]
spec = Shell_Spectra(the_file,'')
print(the_file)
```

```
/pleiades/nobackupp17/lmatilsk/benchmarks/j2011_mhd_linear/00_test_equati
ons_sf2/Shell_Spectra/15200050
```

In [4]:
```
# break up data
vals = spec.vals

times =  spec.time
dt = times[2:] - times[1:-1]
dt = dt.reshape((1, 1, 1, len(dt)))
dt_old = times[1:-1] - times[:-2]
dt_old = dt_old.reshape((1, 1, 1, len(dt_old)))

lut = spec.lut
irvals = spec.inds

print ('irvals = ', irvals)
print ('r/rsun = ', spec.radius/common.rsun)
```

```
irvals =  [ 18  29  42  53  63  74  85  98 109]
r/rsun =  [0.09730686 0.09241647 0.08438459 0.07625216 0.06831853 0.05958
225
 0.05144982 0.04341794 0.03852755]
```

In [5]:
```
print (np.shape(vals))
```

```
(128, 128, 9, 19, 50)
```

In [6]:
```
# get rotation period (for time axis)
eq = equation_coefficients()
eq.read(dirname + 'equation_coefficients')
Om0 = eq.constants[0]/2
prot = 2*np.pi/Om0
t = times/prot
```

In [7]:
```
# check if timestep changes (usually doesn't over <1000 iters)
print(np.mean(np.diff(times)))
print(np.std(np.diff(times)))
print (len(times))
```

```
200.0
0.0
50
```

In [8]:
```
# get theta weights
gi = GridInfo(dirname + 'grid_info', '')
tw = gi.tweights.reshape((gi.ntheta, 1, 1))
```

In [9]:

```
# function to integrate power (|quantity|^2) over spherical surface
# normalized to return the SQUARE ROOT (rms) of the power AVERAGED over t
# should return same quantity for Shell_Spectra or Shell_Slices
def my_abs(arr, sslice=False, subl0=False):
    arr_orig = np.copy(arr)
    if subl0:
        if sslice:
            arrl0 = np.mean(arr_orig, axis=0)
            arrl0 = np.sum(arrl0*tw, axis=0)
            theshape = np.array(np.shape(arr_orig))
            theshape[0] = theshape[1] = 1
            arrl0 = arrl0.reshape((theshape))
            arr_orig -= arrl0
        else:
            arr_orig[0,0,...] = 0.0
    out = np.abs(arr_orig)**2
    if sslice:
        out = np.mean(out, axis=0)
        out = np.sum(out*tw, axis=0)
    else:
        out = np.sum(np.sum(out, axis=0), axis=0)/(4*np.pi)
    return np.sqrt(out)
# on a complete slice or spectra, will return positive-definite array of
```

In [10]:

```
# A var equation
a_var = vals[..., lut[2901], :]
a_dr = vals[..., lut[2902], :]
a_dr2 = vals[..., lut[2903], :]
a_hlap = vals[..., lut[2904], :]
a_diff = vals[..., lut[2905], :]

# C var equation
c_var = vals[..., lut[2906], :]
c_dr = vals[..., lut[2907], :]
c_dr2 = vals[..., lut[2908], :]
c_hlap = vals[..., lut[2909], :]
c_diff = vals[..., lut[2910], :]
```

In [11]:

```
# test shape of my_abs -- should equal (n_r, n_iter)
print(np.shape(my_abs(a_var)))
```

```
(9, 50)
```

In [12]:

```
# get L^2 = l (l + 1)
np.shape(a_var)
```

Out[12]:

```
(128, 128, 9, 50)
```

In [13]:

```
lvals = np.arange(spec.nell, dtype='float')
L2 = (lvals*(lvals+1)).reshape((spec.nell, 1, 1, 1))
```

In [14]:

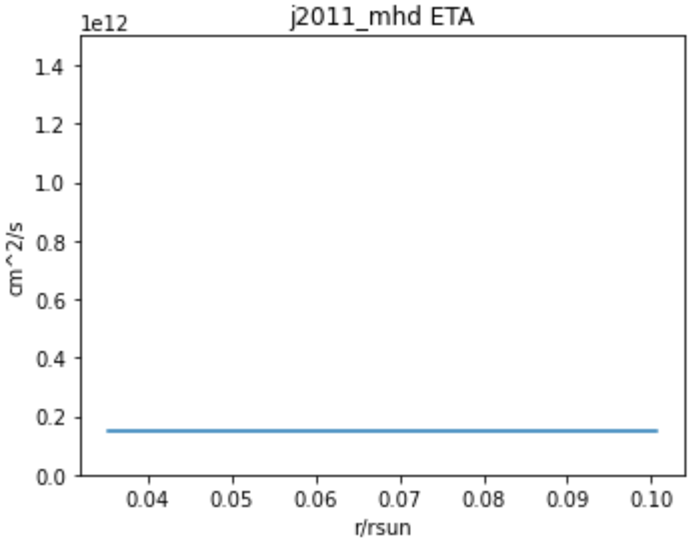
```
# get r vals
rvals = spec.radius.reshape((1, 1, spec.nr, 1))
```

In [15]:

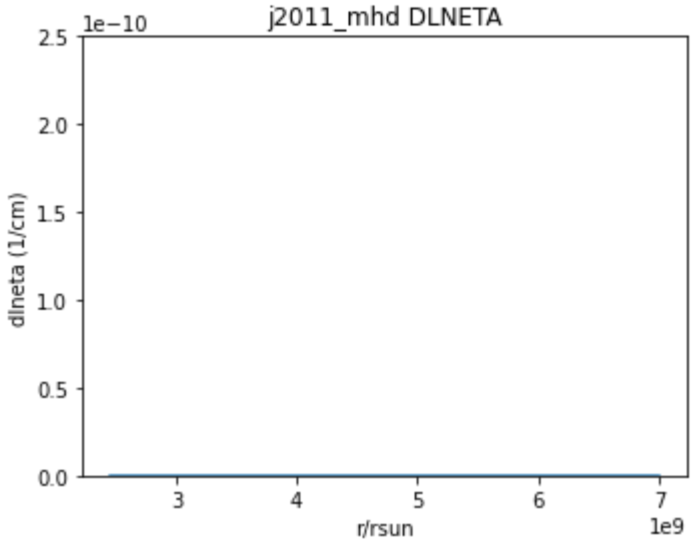
```
# eta
eta_full = eq.constants[6]*eq.functions[6]
dlneta_full = eq.functions[12]
eta = eta_full[spec.inds].reshape((1, 1, spec.nr, 1))
dlneta = dlneta_full[spec.inds].reshape((1, 1, spec.nr, 1))
```

In [16]:

```
plt.figure(figsize=(5, 4))
plt.plot(eq.radius/common.rsun,eta_full)
plt.xlabel('r/rsun')
plt.ylabel('cm^2/s')
plt.title(dirtag + ' ETA')
plt.ylim(0, 1.5e12)
plt.tight_layout()
plt.savefig(dirtag + '_eta.pdf')
```



In [17]:
```
plt.figure(figsize=(5, 4))
plt.plot(eq.radius,dlneta_full)
plt.xlabel('r/rsun')
plt.ylabel('dlneta (1/cm)')
plt.title(dirtag + ' DLNETA')
plt.ylim(0, 2.5e-10)
plt.tight_layout()
plt.savefig(dirtag + '_dlneta.pdf')
```



In [18]:
```
# make a definition of "error" (relative difference between two quantitie
# for this, will average over time axis (axis = 1)
# returns "errors" at each radius sampled (array of shape (nr,))
def get_err(quant1, quant2, sslice=False, subl0=False):
    numer = np.mean(my_abs(quant1 - quant2, sslice=sslice, subl0=subl0),a
    denom = np.mean(my_abs(quant1 + quant2, sslice=sslice, subl0=subl0),a
    return numer/denom
```

In [19]:
```
# check all the quantities are self consistent
print ('a hlap err = ', get_err(a_hlap, - L2/rvals**2*a_var))
print ('a diff err = ', get_err(a_diff, eta*(a_dr2 - L2/rvals**2*a_var +
print ('c hlap err = ', get_err(c_hlap, -L2/rvals**2*c_var))
print ('c diff err = ', get_err(c_diff, eta*(c_dr2 - L2/rvals**2*c_var)))
```

```
a hlap err =  [1.29544331e-09 1.29567683e-09 1.29642620e-09 1.29753492e-0
9
 1.29886676e-09 1.30045690e-09 1.30189790e-09 1.30308114e-09
 1.30352822e-09]
a diff err =  [2.21650126e-10 2.45775883e-10 2.94971329e-10 3.61576944e-1
0
 4.50928673e-10 5.93639386e-10 7.97082195e-10 1.12033883e-09
 1.42331380e-09]
c hlap err =  [3.09924945e-09 3.10053455e-09 3.10103808e-09 3.10001030e-0
9
 3.09771516e-09 3.09417712e-09 3.09020706e-09 3.08594551e-09
 3.08326993e-09]
c diff err =  [6.68117224e-10 7.40233243e-10 8.84873032e-10 1.07749492e-0
9
 1.33269792e-09 1.73698102e-09 2.31116123e-09 3.22422087e-09
 4.08336110e-09]
```

In [20]:

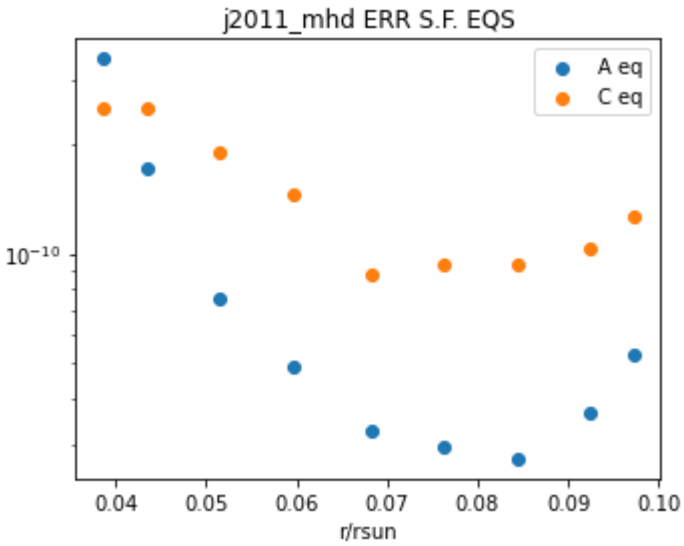
```
# get lhs = d var / dt
def get_ddt(variable):
    dvar = variable[..., 2:] - variable[..., 1:-1]
    return dvar/dt

a_dt = get_ddt(a_var)
c_dt = get_ddt(c_var)
```

In [21]:

```
# get weighted rhs (for consistency include fexp in the function,
# even though it is always zero for these tests

def get_weighted_rhs(fimp, fexp):
    return 0.5*(fimp[..., 2:] + fimp[..., 1:-1]) + fexp[..., 1:-1] +\
        0.5*(dt/dt_old)*(fexp[..., 1:-1] - fexp[..., :-2])

a_rhs = get_weighted_rhs(a_diff, np.zeros_like(a_diff))
c_rhs = get_weighted_rhs(c_diff, np.zeros_like(c_diff))
```

In [22]:

```
# will use 'r axis a lot'
rr = spec.radius/common.rsun

# plot errors in stream function equations (ignore l = 0 bit)
plt.figure(figsize=(5, 4))
plt.scatter(rr, get_err(a_dt, a_rhs, subl0=True), label='A eq')
plt.scatter(rr, get_err(c_dt, c_rhs, subl0=True), label='C eq')
plt.yscale('log')
plt.xlabel('r/rsun')
plt.title(dirtag + ' ERR S.F. EQS')
plt.legend()
plt.tight_layout()
plt.savefig(dirtag + '_err_sf.pdf')
```



In [23]:

```
# print these errors
print ('A eq err = ', get_err(a_dt, a_rhs, subl0=True))
print ('C eq err = ', get_err(c_dt, c_rhs, subl0=True))
```

```
A eq err =  [5.24836019e-11 3.65866661e-11 2.73681131e-11 2.95147376e-11
 3.26446427e-11 4.88345805e-11 7.53443577e-11 1.71208287e-10
 3.42962877e-10]
C eq err =  [1.26629685e-10 1.03768536e-10 9.34115947e-11 9.38283541e-11
```

```
8.77191662e-11 1.45783569e-10 1.89151925e-10 2.51971059e-10
2.52591729e-10]
```

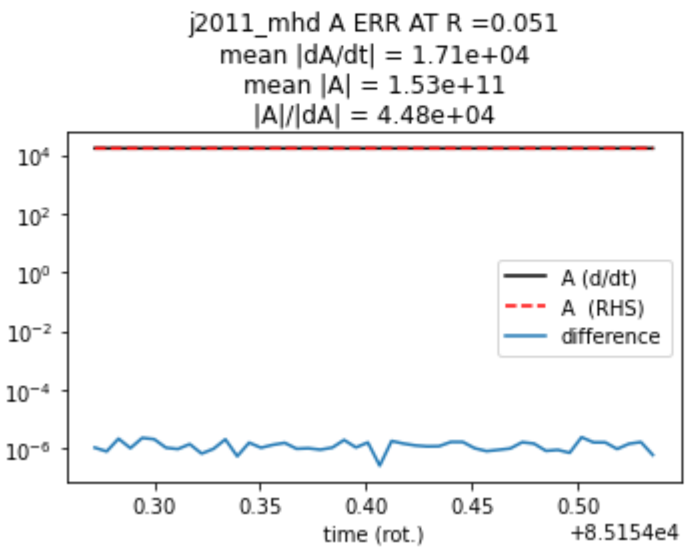
In [24]:

```
# pick a radial location to plot (in RZ for case M)
ir0 = 6
print ('r0/rsun =', rr[ir0])
```

```
r0/rsun = 0.05144981907048454
```

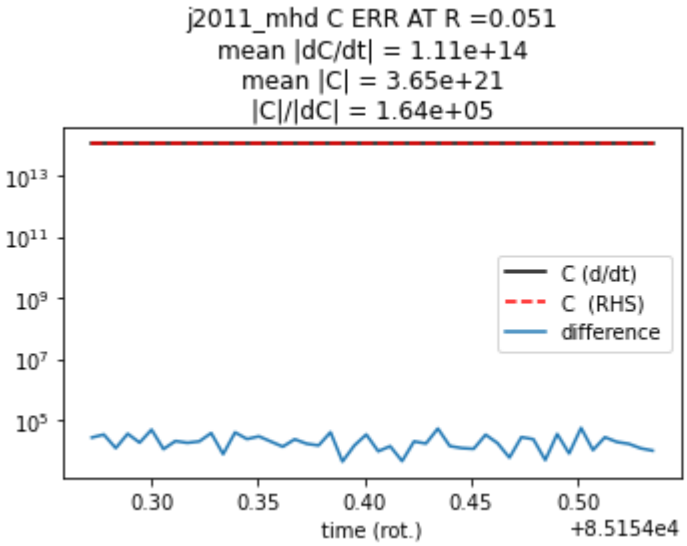
In [25]:

```
# plot A eq error vs time at ir0
plt.figure(figsize=(5, 4))
plt.plot(t[2:], my_abs(a_dt[..., ir0, :], subl0=True), 'k', label='A (d/d
plt.plot(t[2:], my_abs(a_rhs[..., ir0, :], subl0=True), 'r--', label='A
plt.plot(t[2:], my_abs((a_dt-a_rhs)[..., ir0, :], subl0=True), label='dif
plt.yscale('log')
plt.legend()

# compute how long it would take source term to do anything (in iteration
# ...i.e., if it would do anything on the timescale of the simulation (mi
# we care about getting it right
title = dirtag + ' A ERR AT R =%.3f' %rr[ir0]
title += '\n' + 'mean |dA/dt| = %1.2e' %np.mean(my_abs(a_dt[..., ir0, :],
title += '\n' + 'mean |A| = %1.2e' %np.mean(my_abs(a_var[..., ir0, :], su
title += '\n' + '|A|/|dA| = %1.2e' %(np.mean(my_abs(a_var[..., ir0, :], s
        (np.mean(dt)*np.mean(my_abs(a_dt[..., ir0, :], subl0=True))))
plt.title(title)
plt.xlabel('time (rot.)')
plt.tight_layout()
plt.savefig(dirtag + '_aerr_vs_time.pdf')
```



In [26]:

```
# same for C eq
plt.figure(figsize=(5, 4))
plt.plot(t[2:], my_abs(c_dt[..., ir0, :], subl0=True), 'k', label='C (d/d
plt.plot(t[2:], my_abs(c_rhs[..., ir0, :], subl0=True), 'r--', label='C
plt.plot(t[2:], my_abs((c_dt-c_rhs)[..., ir0, :], subl0=True), label='dif
plt.yscale('log')
plt.legend()

# compute how long it would take source term to do anything (in iteration
# ...i.e., if it would do anything on the timescale of the simulation (mi
# we care about getting it right
title = dirtag + ' C ERR AT R =%.3f' %rr[ir0]
title += '\n' + 'mean |dC/dt| = %1.2e' %np.mean(my_abs(c_dt[..., ir0, :],
title += '\n' + 'mean |C| = %1.2e' %np.mean(my_abs(c_var[..., ir0, :], su
title += '\n' + '|C|/|dC| = %1.2e' %(np.mean(my_abs(c_var[..., ir0, :], s
        (np.mean(dt)*np.mean(my_abs(c_dt[..., ir0, :], subl0=True))))
plt.title(title)
plt.xlabel('time (rot.)')
plt.tight_layout()
plt.savefig(dirtag + '_cerr_vs_time.pdf')
```



In [27]:

```
# get radial br equation
b = vals[..., lut[801], :]
fimp = diff = vals[..., lut[1605], :]
dbdt = get_ddt(b)
fweighted = get_weighted_rhs(fimp, np.zeros_like(fimp))
```

In [28]:

```
# get err in "does br = -l(l+1)C/r^2"
print ('br = C? err spec = ', get_err(b, -c_hlap))
```

```
br = C? err spec =  [0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

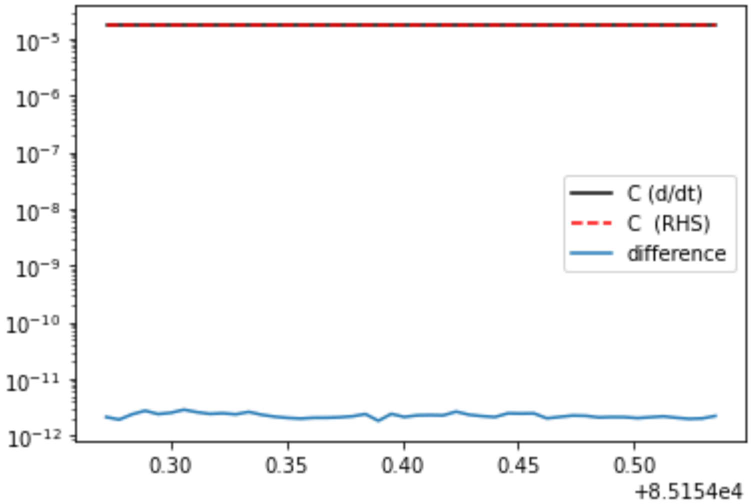
In [29]:

```
# get err in "does diffusion = what its supposed to"
print ('diff. err spec = ', get_err(diff, L2/rvals**2*c_diff))
```

```
diff. err spec =  [0.00107099 0.00120072 0.00144247 0.00173967 0.00210762
0.00265748
 0.00340367 0.0045519  0.00560959]
```

In [30]:
```
# I think that's my error!

# let's multiply C eq. by L2/r^2 (- hor lapl)

plt.plot(t[2:], my_abs(((L2/rvals**2)*c_dt)[..., ir0, :], subl0=True), 'k
plt.plot(t[2:], my_abs(((L2/rvals**2)*c_rhs)[..., ir0, :], subl0=True), '
plt.plot(t[2:], my_abs(((L2/rvals**2)*(c_dt-c_rhs))[..., ir0, :], subl0=T
plt.yscale('log')
plt.legend()
```

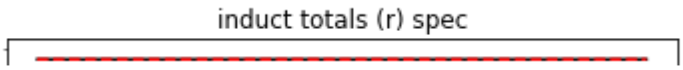
Out[30]: <matplotlib.legend.Legend at 0x2aaaaed8f150>



In [31]:
```
# all good...
```

In [32]:
```
# plot the B_r equation
plt.plot(t[2:], my_abs(dbdt - fweighted, subl0=True)[ir0, :], 'b', label=
plt.plot(t[2:], my_abs(dbdt, subl0=True)[ir0, :], 'k', label='dB_r /dt (n
plt.plot(t[2:], my_abs(fweighted, subl0=True)[ir0, :], 'r--', label='weig
plt.yscale('log')

# label
plt.legend()
plt.xlabel('time (prot)')
plt.ylabel('induction (G/s)')
plt.title('induct totals (r) spec')
```

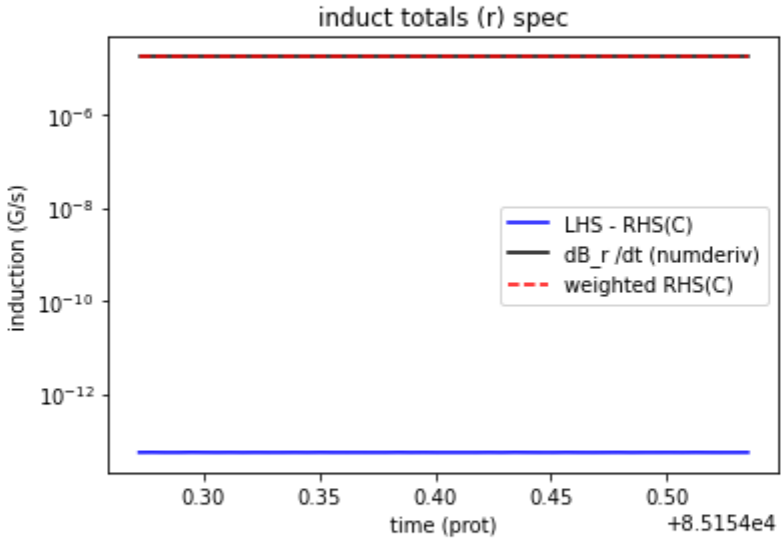
Out[32]: Text(0.5, 1.0, 'induct totals (r) spec')



In [33]:

```
# plot the B_r equation, now with l(l+1)/r^2 RHS C eq
plt.plot(t[2:], my_abs(dbdt - L2/rvals**2*c_rhs, subl0=True)[ir0, :], 'b'
plt.plot(t[2:], my_abs(dbdt, subl0=True)[ir0, :], 'k', label='dB_r /dt (n
plt.plot(t[2:], my_abs(L2/rvals**2*c_rhs, subl0=True)[ir0, :], 'r--', lab
plt.yscale('log')

# label
plt.legend()
plt.xlabel('time (prot)')
plt.ylabel('induction (G/s)')
plt.title('induct totals (r) spec')
```

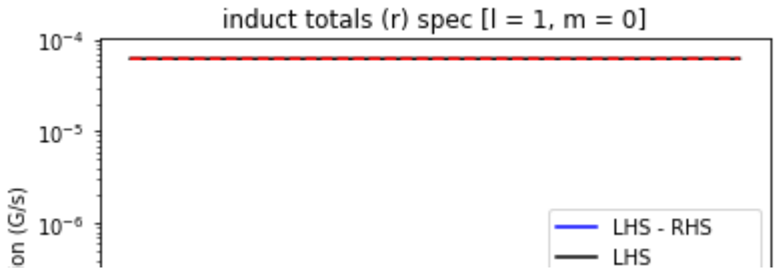
Out[33]: Text(0.5, 1.0, 'induct totals (r) spec')



In [34]:

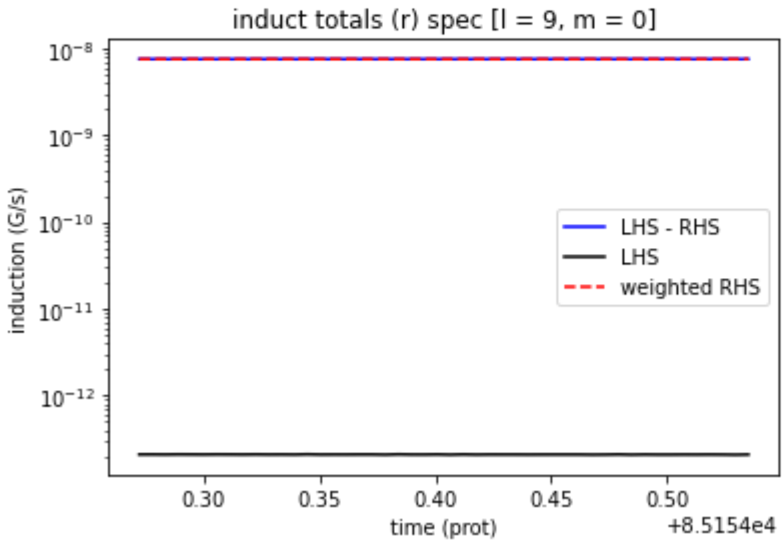
```
# pick particular l, m value (m = 1 has most power in case M)
l0 = 1
m0 = 0

# plot the B_r equation
plt.plot(t[2:], np.abs((dbdt - fweighted)[l0, m0, ir0, :]), 'b', label='L
plt.plot(t[2:], np.abs(dbdt[l0, m0, ir0, :]), 'k', label='LHS')
plt.plot(t[2:], np.abs(fweighted[l0, m0, ir0, :]), 'r--', label='weighted
plt.yscale('log')

# label
plt.legend()
plt.xlabel('time (prot)')
plt.ylabel('induction (G/s)')
plt.title('induct totals (r) spec [l = %i, m = %i]' %(l0, m0))
```

Out[34]: Text(0.5, 1.0, 'induct totals (r) spec [l = 1, m = 0]')



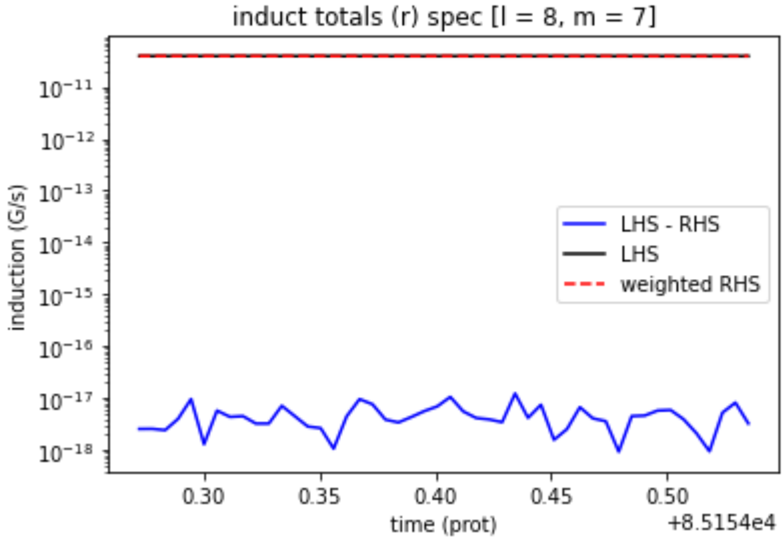
In [35]:
```
# pick particular l, m value --- higher l
l0 = 9
m0 = 0

# plot the B_r equation
plt.plot(t[2:], np.abs((dbdt - fweighted)[l0, m0, ir0, :]), 'b', label='L
plt.plot(t[2:], np.abs(dbdt[l0, m0, ir0, :]), 'k', label='LHS')
plt.plot(t[2:], np.abs(fweighted[l0, m0, ir0, :]), 'r--', label='weighted
plt.yscale('log')

# label
plt.legend()
plt.xlabel('time (prot)')
plt.ylabel('induction (G/s)')
plt.title('induct totals (r) spec [l = %i, m = %i]' %(l0, m0))
```

Out[35]: Text(0.5, 1.0, 'induct totals (r) spec [l = 9, m = 0]')



In [36]:
```
# pick particular l, m value -- higher m
l0 = 8
m0 = 7

# plot the B_r equation
plt.plot(t[2:], np.abs((dbdt - fweighted)[l0, m0, ir0, :]), 'b', label='L
plt.plot(t[2:], np.abs(dbdt[l0, m0, ir0, :]), 'k', label='LHS')
plt.plot(t[2:], np.abs(fweighted[l0, m0, ir0, :]), 'r--', label='weighted
plt.yscale('log')

# label
plt.legend()
plt.xlabel('time (prot)')
plt.ylabel('induction (G/s)')
plt.title('induct totals (r) spec [l = %i, m = %i]' %(l0, m0))
```

Out[36]: Text(0.5, 1.0, 'induct totals (r) spec [l = 8, m = 7]')



In [37]:

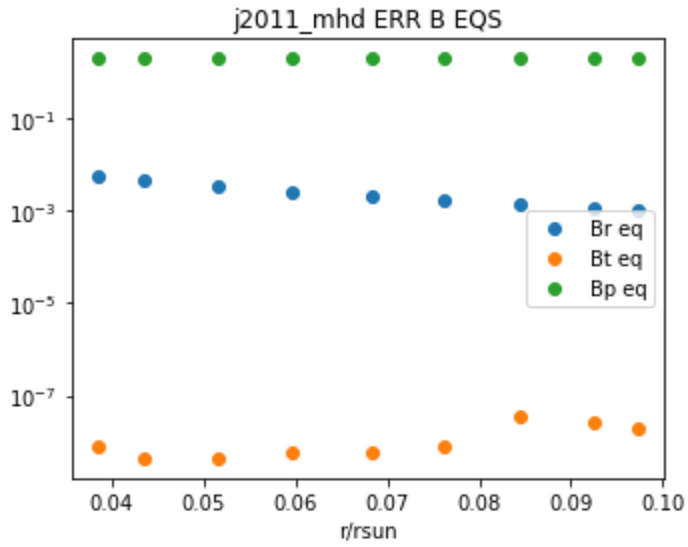
```
# now check the b_theta and b_phi equations
br = vals[..., lut[801], :]
bt = vals[..., lut[802], :]
bp = vals[..., lut[803], :]

diffr = vals[..., lut[1605], :]
difft = vals[..., lut[1610], :]
diffp = vals[..., lut[1615], :]

diffr_sf = vals[..., lut[2911], :]
difft_sf = vals[..., lut[2912], :]
diffp_sf = vals[..., lut[2913], :]
```

In [38]:

```
dbdt_r = get_ddt(br)
dbdt_t = get_ddt(bt)
dbdt_p = get_ddt(bp)

fweightedr = get_weighted_rhs(diffr, np.zeros_like(diffr))
fweightedr_sf = get_weighted_rhs(diffr_sf, np.zeros_like(diffr_sf))

fweightedt = get_weighted_rhs(difft, np.zeros_like(difft))
fweightedt_sf = get_weighted_rhs(difft_sf, np.zeros_like(difft_sf))

fweightedp = get_weighted_rhs(diffp, np.zeros_like(diffp))
fweightedp_sf = get_weighted_rhs(diffp_sf, np.zeros_like(diffp_sf))
```

In [39]:

```
# plot errors in stream function equations (ignore l = 0 bit)
plt.figure(figsize=(5, 4))
plt.scatter(rr, get_err(dbdt_r, fweightedr, subl0=True), label='Br eq')
plt.scatter(rr, get_err(dbdt_t, fweightedt), label='Bt eq')
plt.scatter(rr, get_err(dbdt_t, fweightedp), label='Bp eq')
plt.yscale('log')
plt.xlabel('r/rsun')
plt.title(dirtag + ' ERR B EQS')
plt.legend()
plt.tight_layout()
plt.savefig(dirtag + '_err_ind.pdf')
```



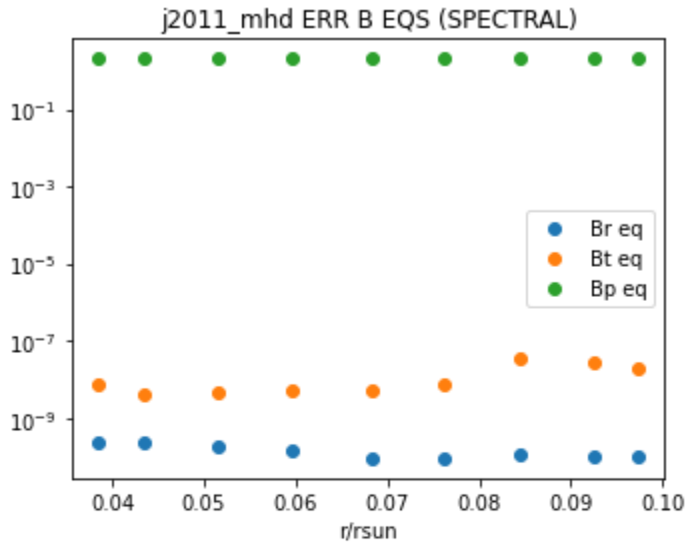
In [40]:

```
# print these errors
print ('Br eq err = ', get_err(dbdt_r, fweightedr, subl0=True))
print ('Bt eq err = ', get_err(dbdt_t, fweightedt))
print ('Bp eq err = ', get_err(dbdt_p, fweightedp))
```

```
Br eq err =  [0.00107099 0.00120071 0.00144247 0.00173967 0.00210762 0.00
265747
 0.00340367 0.00455189 0.00560958]
Bt eq err =  [2.00080894e-08 2.69100055e-08 3.61500183e-08 7.55036444e-09
 5.57869956e-09 5.53878020e-09 4.48197766e-09 4.37093762e-09
 7.45488677e-09]
Bp eq err =  [5.51864842e-11 4.14575472e-11 3.04726504e-11 3.45598756e-11
 3.77961556e-11 5.16337332e-11 7.94175197e-11 1.72819058e-10
 3.44077930e-10]
```

In [41]:

```
# plot errors B equation with new "spectral" stream function outputs for
plt.figure(figsize=(5, 4))
plt.scatter(rr, get_err(dbdt_r, fweightedr_sf, subl0=True), label='Br eq'
plt.scatter(rr, get_err(dbdt_t, fweightedt_sf), label='Bt eq')
plt.scatter(rr, get_err(dbdt_t, fweightedp_sf), label='Bp eq')
plt.yscale('log')
plt.xlabel('r/rsun')
plt.title(dirtag + ' ERR B EQS (SPECTRAL)')
plt.legend()
plt.tight_layout()
plt.savefig(dirtag + '_err_ind_spectral.pdf')
```



In [42]:

```
# print these errors
print ('Br eq err (spectral) = ', get_err(dbdt_r, fweightedr_sf, subl0=Tr
print ('Bt eq err (spectral) = ', get_err(dbdt_t, fweightedt_sf))
print ('Bp eq err (spectral) = ', get_err(dbdt_p, fweightedp_sf))
```

```
Br eq err (spectral) =  [1.09535797e-10 1.00852804e-10 1.13495179e-10 9.4
6023016e-11
 9.76442059e-11 1.45823048e-10 1.87945052e-10 2.53216387e-10
 2.49659454e-10]
Bt eq err (spectral) =  [2.00270492e-08 2.69905903e-08 3.61133642e-08 7.5
7983171e-09
 5.57474049e-09 5.51493856e-09 4.47535997e-09 4.36987547e-09
 7.45083856e-09]
Bp eq err (spectral) =  [5.50774686e-11 4.12998233e-11 3.01566154e-11 3.4
1882933e-11
 3.72835758e-11 5.09549969e-11 7.86212262e-11 1.72048609e-10
 3.43297790e-10]
```

In [ ]: