MEK4420 - Obligatorisk oppgave 2

Ana Costa Conrado
Universitetet i Oslo
(Dated: May 7, 2018)

I. MEK4420 - THE FORCES AND RESPONSE OF A HEAVING SECTION IN 2D

A. The Boundary value problem (BVP) of the heave problem

- Formulate the boundary value problem (BVP) for the heaving potential ϕ_2 due to a geometry-section floating in the free surface, in two dimensions, where the radiation potential is given by

$$\Phi_R(x,y,t) = Re\left(i\omega\xi_2\phi_2(x,y)e^{i\omega t}\right) \tag{1}$$

  where $\frac{\omega^2}{g} = K$ the given complex heave amplitude and ϕ_2 the complex potential in the heave mode of motion.

ϕ_2 tilfredsstiller følgende sett av randverdiproblemer:

$$\nabla^2\Phi \equiv \frac{\partial^2\phi_2}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = 0 \qquad \text{i fluidet}$$

$$\frac{\partial\phi_2}{\partial n} = i\omega n_2 \qquad \text{på } S_B$$

$$-\omega^2\phi_2 + g\frac{\partial\phi_2}{\partial y} = 0 \qquad \text{på } y = 0$$

Strålingsbetingelse: $\phi_2 \propto e^{\mp iKx}, x \to \pm\infty$

B. The BVP for the Green function

Formulate the boundary value problem for the Green function G, where

$$g(x,y;\bar{x},\bar{y},t) = Re\left(G(x,y;\bar{x},\bar{y})e^{i\omega t}\right) \tag{2}$$

$$G(x,y;\bar{x},\bar{y}) = \log r + H(x,y;\bar{x},\bar{y}), \tag{3}$$

and $r = \sqrt{(x-\bar{x})^2 + (y-\bar{y})^2}$. In the formulation of this BVP, formulate the field equation, the boundary conditions at the free surface, at $x \to \pm\infty$, and at $y \to -\infty$.

$$\nabla^2\Phi = 0 \qquad \text{i fluidet, } x \neq \bar{x}, y \neq \bar{y}$$

$$-\nu G + \frac{\partial G}{\partial y} = 0, \text{ på } y = 0$$

$$\frac{\partial G}{\partial(\pm x)} = -i\nu G, x \to \pm\infty$$

$$|\nabla G| \to 0, \qquad y \to -\infty$$

1. *Integral equation*

- Use Green's theorem to derive an integral equation for the heave problem with a free surface.

$$\nabla^2 \phi = 0 \tag{4}$$

$$0 = \iiint\limits_V \nabla \cdot (\phi \nabla \varphi - \varphi \nabla \phi)\, dV = \iint\limits_{\bar{S}} \left(\phi \frac{\partial \varphi}{\partial n} - \frac{\partial \phi}{\partial n} \varphi \right) dS = \iiint\limits_V \phi \nabla^2 \varphi + \nabla \phi \cdot \nabla \varphi - \varphi \nabla^2 \phi - \nabla \varphi \nabla \phi \, dV \tag{5}$$

hvor $\bar{S} = S_F + S_B + S_{-\infty} + S_\infty + S_{bunn} + S_\epsilon$ og $S = \bar{S} - S_\epsilon$.

$$-\iint\limits_{S_\epsilon} \phi \frac{\partial \varphi}{\partial n} - \varphi \frac{\partial \phi}{\partial n} \, dS = \iint\limits_S \phi \frac{\partial \varphi}{\partial n} - \varphi \frac{\partial \phi}{\partial n} \, dS \tag{6}$$

$$\pi \phi(\bar{x}, \bar{y}) = \iint\limits_S \phi \frac{\partial \varphi}{\partial n} - \varphi \frac{\partial \phi}{\partial n} \, dS \tag{7}$$

Vi lar $\varphi = G$.

$$G(x, y; \bar{x}, \bar{y}) = \log r + H(x, y; \bar{x}, \bar{y}) \tag{8}$$

$$r = \sqrt{(x - \bar{x})^2 + (y - \bar{y})^2} \tag{9}$$

$$\pi\, \phi(\bar{x}, \bar{y}) = \iint\limits_S \phi \frac{\partial G}{\partial n} - G \frac{\partial \phi}{\partial n} \, dS = \iint\limits_{S_B} \phi \frac{\partial G}{\partial n} - G \frac{\partial \phi}{\partial n} \, dS$$

Vi betrakter bare hiv:

$$\frac{\partial \phi_2}{\partial n} = n_2 \qquad \text{på } S_B \tag{10}$$

$$\pi\, \phi_2(\bar{x}, \bar{y}) = \iint\limits_{S_B} \phi_2 \frac{\partial G}{\partial n} - G \frac{\partial \phi_2}{\partial n} \, dS = \iint\limits_{S_B} \phi_2 \frac{\partial G}{\partial n} - G\, n_2 \, dS \tag{11}$$

$$-\pi \phi_2(\bar{x}, \bar{y}) + \iint\limits_{S_B} \phi_2 \frac{\partial G}{\partial n} \, dS = \iint\limits_{S_B} G\, n_2 \, dS$$

for $(\bar{x}, \bar{y})$ på legemesrandet S_B.

2. *Integral equation (2)*

The integral equation in section 6.3 becomes

$$-\pi \phi_2 \left(\bar{x}, \bar{y}\right) + \iint\limits_{S_B} \phi_2 \frac{\partial G}{\partial n} \, dS = \iint\limits_{S_B} G\, n_2 \, dS$$

for $(\bar{x}, \bar{y})$ på legemesrand S_B. In the case when $(\bar{x}, \bar{y})$ is in the fluid (not on S_B) the equation becomes (it is then not an integral equation)

$$2\pi \phi_2 \left(\bar{x}, \bar{y}\right) = \iint\limits_{S_B} \left(\phi_2 \frac{\partial G}{\partial n} - G\, n_2 \right) dS$$

3. Numerical solution of the integral equation

a. Dircretisation of the wetted surface S_B

- Make a discretisation of the wetted part of a rectangular geometry of width L and draught D where D is chosen as unit length in the problem. Use N_2 as resolution along the vertical sides and N_1 as resolution along the bottom of the rectangle, where $\Delta y = \frac{D}{N_2}$ and $\Delta x = \frac{L}{N_1}$.
- Make discretisations for three geometries with \$L/D $= 2, 1$ \$ and 0.1. It suffices to use $N_2 = 10$. Use a similar resolution along the bottom and $N_1 = 5$ for the thin rectangle. The equation to be solved is

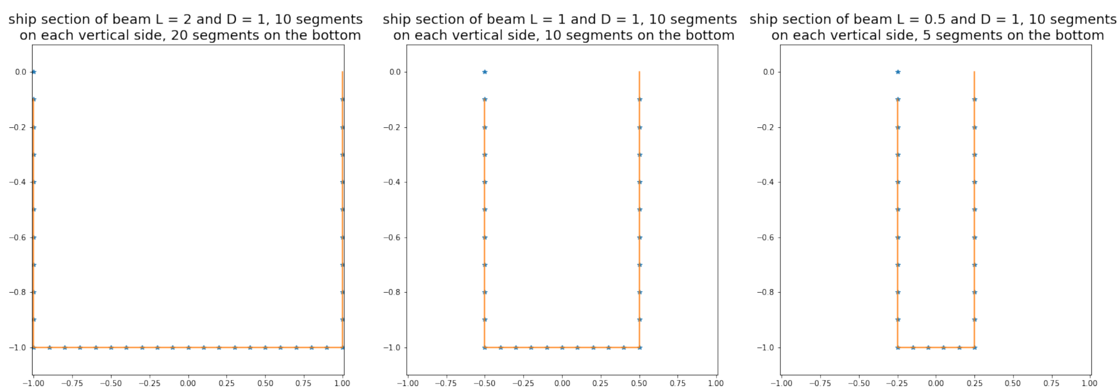
$$-\pi\,\phi_2 + \iint\limits_{S_B} \phi_2 \frac{\partial G}{\partial n}\,dS = \iint\limits_{S_B} G\,n_2\,dS$$

In [10]:
```
from __future__ import division
import numpy
import matplotlib
%matplotlib inline
import math
import mpmath
import scipy.linalg
```

In [11]:
```
def func1(D, L, Nside, Nbott):
    N= Nside + Nbott + Nside
    dy = D/Nside
    dx = L/Nbott

    xp = [-L/2 for _ in range(Nside)]
    xm = [-L/2 for _ in range(Nside)]
    yp = [-dy*(i+1) for i in range(Nside)]
    ym = [-dy*(i) for i in range(Nside)]

    xp.extend([-L/2 + dx*(i+1) for i in range(Nbott)])
    xm.extend([-L/2 + dx*(i) for i in range(Nbott)])
    yp.extend([-D for _ in range(Nbott)])
    ym.extend([-D for _ in range(Nbott)])

    xp.extend([L/2 for _ in range(Nside)])
    xm.extend([L/2 for _ in range(Nside)])
    yp.extend([-D+dy*(i+1) for i in range(Nside)])
    ym.extend([-D+dy*i for i in range(Nside)])

    coord= numpy.stack((xp,xm,yp,ym), axis=1)
    return xp, xm, yp, ym
```

In [36]:
```
fig = matplotlib.pyplot.figure(figsize=[25,8])
D = 1
L = 2
Nside = 10
Nbott = 20
xp1, xm1, yp1, ym1 = func1(D , L , Nside , Nbott )
ax1 = fig.add_subplot(1,3,1)
ax1.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax1.set_title(xlabel, fontsize=18)
ax1.plot(xm1, ym1, '*', xp1, yp1, '-')
```

```
D = 1
L = 1
Nside = 10
Nbott = 10
xp2, xm2, yp2, ym2 = func1(D , L , Nside , Nbott )
ax3= fig.add_subplot(1,3,2)
ax3.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+' segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax3.set_title(xlabel, fontsize=18)
ax3.plot(xm2, ym2, '*', xp2, yp2, '-')

D = 1
L = 0.5
Nside = 10
Nbott = 5
xp3, xm3, yp3, ym3 = func1(D , L , Nside, Nbott)
ax5 = fig.add_subplot(1,3,3)
ax5.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+' segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax5.set_title(xlabel, fontsize=18)
ax5.plot(xm3, ym3, '*', xp3, yp3, '-');
```



4. Solution of a relation where all terms are known

A variant of the equation (122) is

$$\pi\,\varphi_0 + \iint_{S_B} \varphi_0 \frac{\partial G}{\partial n}\, dS = \iint_{S_B} G \frac{\partial \varphi_0}{\partial n} dS \tag{12}$$

where $\varphi_0 = e^{K\,y - i\,K\,x}$ and $K = \frac{\omega^2}{g}$, where all contributions in the terms in (123) are known. * Formulate a discrete version of (123) using the procedures described in the Lecture Notes. * Make a numerical implementation of (123) using analytical integration of the normal derivatives of the log-terms (see the Lecture Notes), two-point Gauss integration of the $\log r$ contribution and mid-point rule for the remaining terms. * Test the numerical implementation for one of the rectangular geometries for a given $K\,D = 1.2$ or 0.9.

Illustrate by plots that

$$Re(L.H.S) = Re(R.H.S), \tag{13}$$
$$Im(L.H.S) = Im(R.H.S), \tag{14}$$

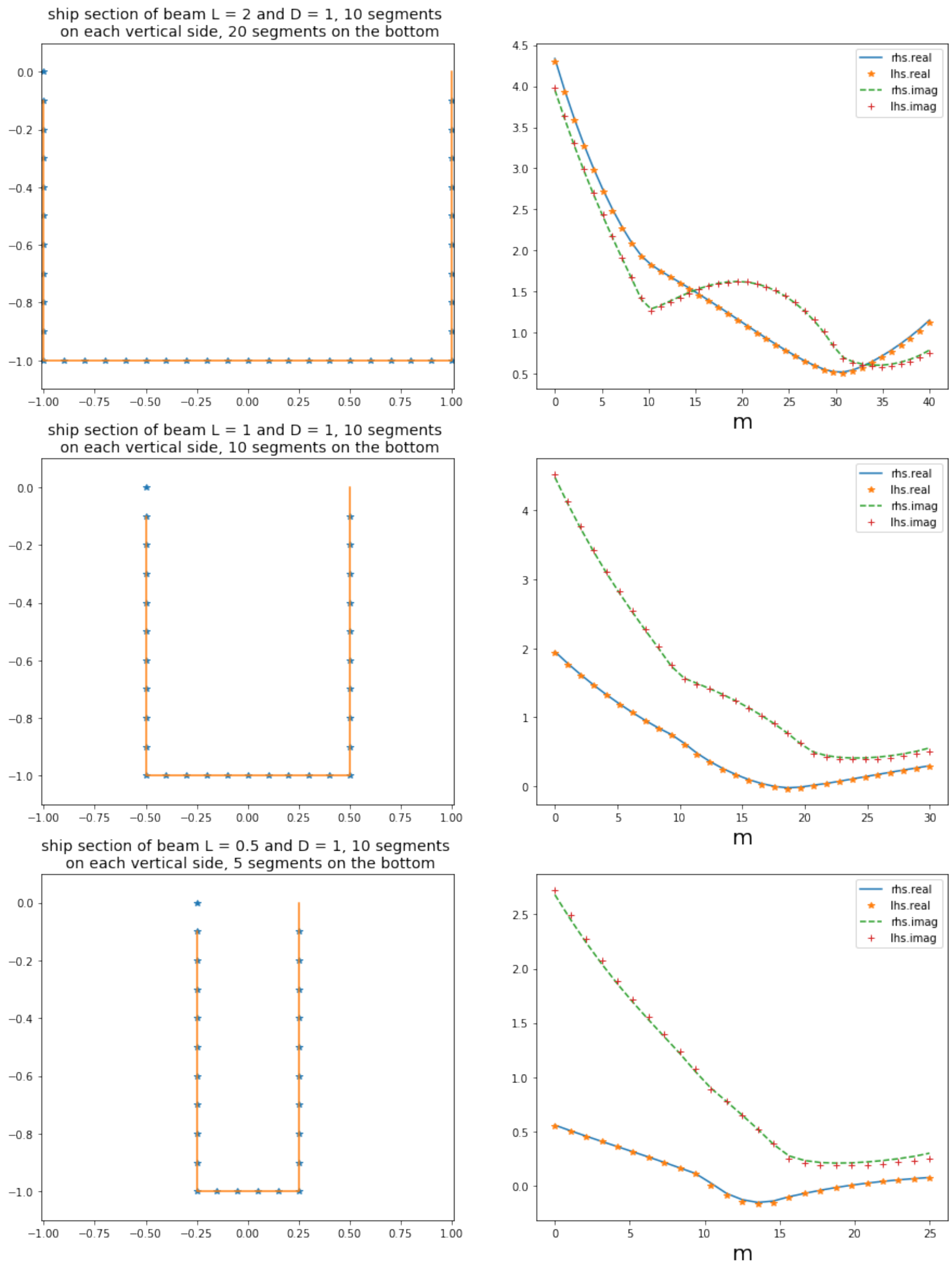
along the wetted body surface S_B.

```
In [13]: def func2 (xp, xm, yp, ym, nu):
             xp = numpy.array(xp)
             xm = numpy.array(xm)
             yp = numpy.array(yp)
             ym = numpy.array(ym)
             NN = xp.shape[0]  # = N= Nside + Nbott + Nside
             iinn = numpy.linspace(0, NN, num = NN, endpoint=True)
             dx = xp - xm
             dy = yp - ym
             ds = numpy.sqrt(dx**2 + dy**2)
             bx = 0.5*(xm + xp)
             by = 0.5*(ym + yp)
             n1 = - (yp - ym)/ds
             n2 = (xp - xm)/ds
             #points for Gauss integration on each segment
             xg1 = - 0.5*dx/math.sqrt(3) + bx
             xg2 = 0.5*dx/math.sqrt(3) + bx
             yg1 = - 0.5*dy/math.sqrt(3) + by
             yg2 = 0.5*dy/math.sqrt(3) + by
             #incoming wave potential
             phi0 = numpy.exp(nu*(by- complex(0,1)*bx))
             phi0n = nu*(n2-complex(0,1)*n1)*phi0
             gg = numpy.zeros((NN,NN), dtype=complex)
             ss = numpy.zeros((NN,NN), dtype=complex)
             #contributions to integral equation, rhs stores rhs, lhs stores lhs
             for i in range(0,NN):
                 for j in range(0,NN):
                     #rhs, log(r) term with 2pts Gauss quadrature
                     xa1 = xg1[j] - bx[i]
                     xa2 = xg2[j] - bx[i]
                     ya1 = yg1[j] - by[i]
                     ya2 = yg2[j] - by[i]
                     ra1 = math.sqrt(xa1*xa1 + ya1*ya1)
                     ra2 = math.sqrt(xa2*xa2 + ya2*ya2)
                     g0 = (math.log(ra1) + math.log(ra2))*0.5
                     #all other terms with midpoint rule
                     xa = bx[j] - bx[i]
                     yb = by[j] + by[i]
                     rb = math.sqrt(xa*xa + yb*yb)
                     g1 = -numpy.log(rb)
                     zz = nu* (yb - complex(0,1)*xa)
                     f1 = -2*numpy.exp(zz)*(mpmath.e1(zz) + numpy.log(zz) - numpy.log(-zz))
                     f2 = 2*math.pi*numpy.exp(zz)
                     g2 = f1.real + complex(0,1)*f2.real
                     gg[i][j]= (g0 + g1 + g2)*ds[j]
                     # lhs
                     if j-i ==0:
                         arg0 = math.pi
                     else:
                         arg0 =(numpy.log((xm[j] -bx[i] +complex(0,1)*(ym[j] -by[i]))/(xp[j]
```

```
-bx[i]+complex(0,1)*(yp[j] -by[i])))).imag

            arg1= (numpy.log((xm[j] - bx[i]+complex(0,1)*(ym[j] + by[i]))/(xp[j] -bx[i]
+ complex(0,1)*(yp[j] + by[i])))).imag
            help1= (n1[j]*(f1.imag+complex(0,1)*f2.imag) + n2[j]*(f1.real +
complex(0,1)*f2.real))*nu*ds[j]
            ss[i][j] = arg0 + arg1 + help1
    rhs = numpy.matmul(gg, phi0n)  #gg*phi0n
    lhs = numpy.matmul(ss, phi0)   #ss*phi0
    return rhs, lhs, iinn
```

In [41]:
```
nu = 0.9
fig = matplotlib.pyplot.figure(figsize=[15,20])
D = 1
L = 2
Nside = 10
Nbott = 20
xp1, xm1, yp1, ym1 = func1(D , L , Nside , Nbott )
ax1 = fig.add_subplot(3,2,1)
ax1.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax1.set_title(xlabel, fontsize=14)
ax1.plot(xm1, ym1, '*', xp1, yp1, '-')
rhs1, lhs1, iinn1 = func2(xp1, xm1, yp1, ym1, nu)
ax2 = fig.add_subplot(3,2, 2)
matplotlib.pyplot.xlabel('m', fontsize=20)
rhsreal1, = ax2.plot( iinn1, rhs1.real, '-', label='rhs.real')
lhsreal1, = ax2.plot( iinn1, lhs1.real, '*', label='lhs.real')
rhsimag1, = ax2.plot( iinn1, rhs1.imag, '--', label='rhs.imag')
lhsimag1, = ax2.plot( iinn1, lhs1.imag, '+', label='lhs.imag')
matplotlib.pyplot.legend(handles= [rhsreal1, lhsreal1, rhsimag1, lhsimag1])

D = 1
L = 1
Nside = 10
Nbott = 10
xp2, xm2, yp2, ym2 = func1(D , L , Nside , Nbott )
ax3= fig.add_subplot(3,2,3)
ax3.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax3.set_title(xlabel, fontsize=14)
ax3.plot(xm2, ym2, '*', xp2, yp2, '-')
rhs2, lhs2, iinn2 = func2(xp2, xm2, yp2, ym2, nu)
ax4 = fig.add_subplot(3,2, 4)
matplotlib.pyplot.xlabel('m', fontsize=20)
rhsreal2, = ax4.plot( iinn2, rhs2.real, '-', label='rhs.real')
lhsreal2, = ax4.plot( iinn2, lhs2.real, '*', label='lhs.real')
rhsimag2, = ax4.plot( iinn2, rhs2.imag, '--', label='rhs.imag')
lhsimag2, = ax4.plot( iinn2, lhs2.imag, '+', label='lhs.imag')
matplotlib.pyplot.legend(handles= [rhsreal2, lhsreal2, rhsimag2, lhsimag2])

D = 1
L = 0.5
Nside = 10
Nbott = 5
```

```
xp3, xm3, yp3, ym3 = func1(D , L , Nside, Nbott)
ax5 = fig.add_subplot(3,2,5)
ax5.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax5.set_title(xlabel, fontsize=14)
ax5.plot(xm3, ym3, '*', xp3, yp3, '-')
rhs3, lhs3, iinn3 = func2(xp3, xm3, yp3, ym3, nu)
ax6 = fig.add_subplot(3,2, 6)
matplotlib.pyplot.xlabel('m', fontsize=20)
rhsreal3, = ax6.plot( iinn3, rhs3.real, '-', label= 'rhs.real')
lhsreal3, = ax6.plot( iinn3, lhs3.real, '*', label= 'lhs.real')
rhsimag3, = ax6.plot( iinn3, rhs3.imag, '--', label= 'rhs.imag')
lhsimag3, = ax6.plot( iinn3, lhs3.imag, '+', label= 'lhs.imag')
matplotlib.pyplot.legend(handles= [rhsreal3, lhsreal3, rhsimag3, lhsimag3]);
```



5. Solution of the heave problem

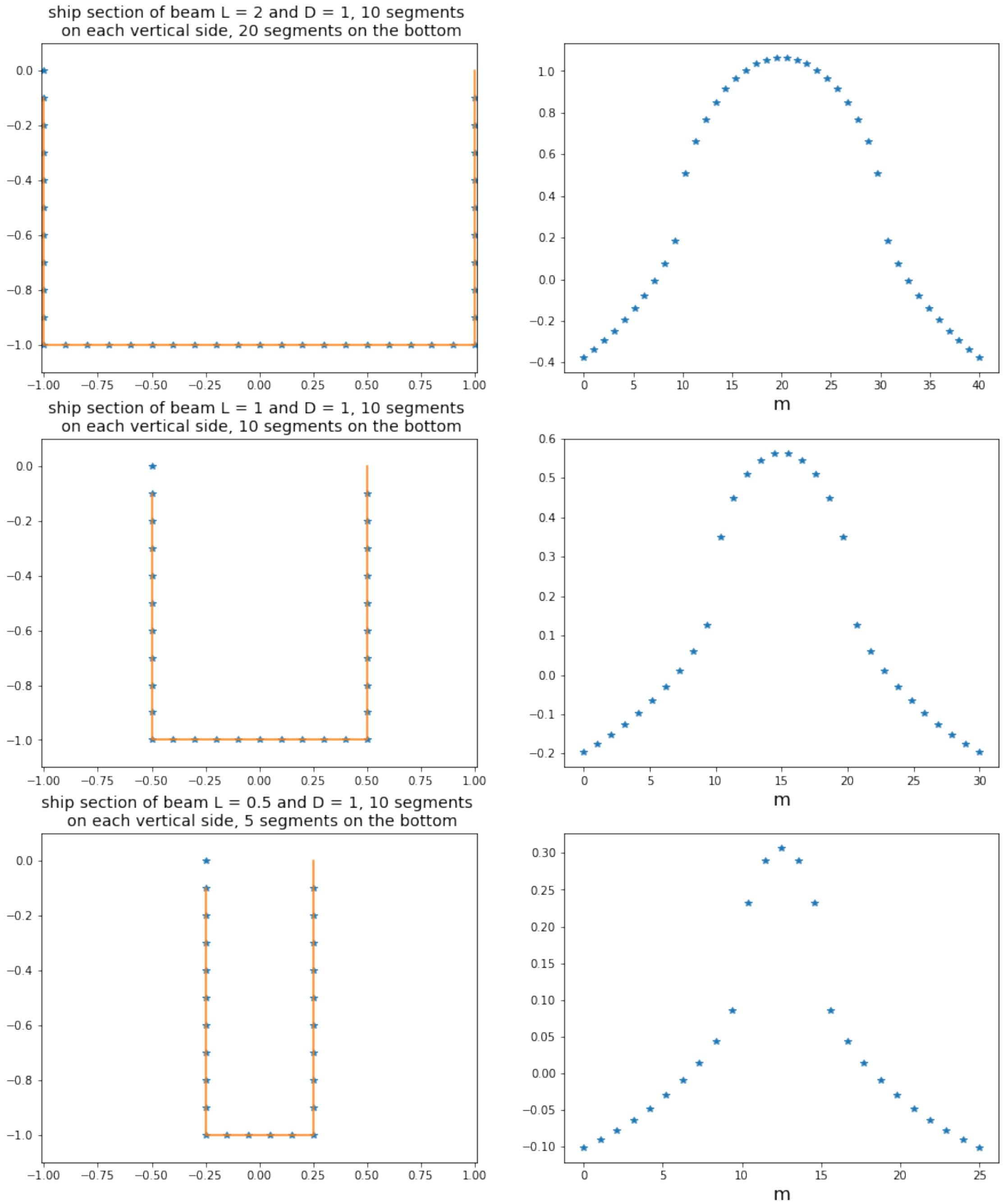
- Then solve (122) numerically for the three different rectangular geometries.

```
In [15]: def func22 (xp, xm, yp, ym, nu):
             xp = numpy.array(xp)
             xm = numpy.array(xm)
             yp = numpy.array(yp)
             ym = numpy.array(ym)
             NN = xp.shape[0]   # = N= Nside + Nbott + Nside
             iinn = numpy.linspace(0, NN, num = NN, endpoint=True)
             dx = xp - xm
             dy = yp - ym
             ds = numpy.sqrt(dx**2 + dy**2)
             bx = 0.5*(xm + xp)
             by = 0.5*(ym + yp)
             n1 = - (yp - ym)/ds
             n2 = (xp - xm)/ds
             #points for Gauss integration on each segment
             xg1 = - 0.5*dx/math.sqrt(3) + bx
             xg2 = 0.5*dx/math.sqrt(3) + bx
             yg1 = - 0.5*dy/math.sqrt(3) + by
             yg2 = 0.5*dy/math.sqrt(3) + by
             #incoming wave potential
             phi0 = numpy.exp(nu*(by- complex(0,1)*bx))
             phi0n = nu*(n2-complex(0,1)*n1)*phi0
             gg = numpy.zeros((NN,NN), dtype=complex)
             ss = numpy.zeros((NN,NN), dtype=complex)
             #contributions to integral equation, rhs stores rhs, lhs stores lhs
             for i in range(0,NN):
                 for j in range(0,NN):
                     #rhs, log(r) term with 2pts Gauss quadrature
                     xa1 = xg1[j] - bx[i]
                     xa2 = xg2[j] - bx[i]
                     ya1 = yg1[j] - by[i]
                     ya2 = yg2[j] - by[i]
                     ra1 = math.sqrt(xa1*xa1 + ya1*ya1)
                     ra2 = math.sqrt(xa2*xa2 + ya2*ya2)
                     g0 = (math.log(ra1) + math.log(ra2))*0.5
                     #all other terms with midpoint rule
                     xa = bx[j] - bx[i]
                     yb = by[j] + by[i]
                     rb = math.sqrt(xa*xa + yb*yb)
                     g1 = -numpy.log(rb)
                     zz = nu* (yb - complex(0,1)*xa)
                     f1 = -2*numpy.exp(zz)*(mpmath.e1(zz) + numpy.log(zz) - numpy.log(-zz))
                     f2 = 2*math.pi*numpy.exp(zz)
                     g2 = f1.real + complex(0,1)*f2.real
                     gg[i][j]= (g0 + g1 + g2)*ds[j]
                     # lhs
                     if j-i ==0:
                         arg0 = - math.pi
                     else:
                         arg0 =(numpy.log((xm[j] -bx[i] +complex(0,1)*(ym[j] -by[i]))/(xp[j]
         -bx[i]+complex(0,1)*(yp[j] -by[i])))).imag

                     arg1= (numpy.log((xm[j] - bx[i]+complex(0,1)*(ym[j] + by[i]))/(xp[j] -bx[i]
         + complex(0,1)*(yp[j] + by[i])))).imag
                     help1= (n1[j]*(f1.imag+complex(0,1)*f2.imag) + n2[j]*(f1.real +
         complex(0,1)*f2.real))*nu*ds[j]
```

```
                ss[i][j] = arg0 + arg1 + help1

        rhs = numpy.matmul(gg, n2)       # matrix multiplication gg*n2
        phi2 = scipy.linalg.solve(ss, rhs)    # in matlab: ss\rhs
        ff22 = phi2* n2 * ds
        sff22 = sum(ff22)

        AM2 = complex(0,1)*(phi2*(nu*n2 - nu*complex(0,1)*n1) - n2)*phi0*ds
        AP2 = complex(0,1)*(phi2*(nu*n2 + nu*complex(0,1)*n1) - n2)*numpy.conj(phi0)*ds
        sAM2 = sum(AM2)
        sAP2 = sum(AP2)

        dampingb22 = 0.5*(sAM2**2 + sAP2**2)

        v1H = sum(-complex(0,1)*(phi0 * n2 - phi2* phi0n)*ds)

        return phi2, iinn, sff22, sAM2, sAP2, v1H
```

In [43]:
```
nu = 0.9
fig = matplotlib.pyplot.figure(figsize=[15,18])
D = 1
L = 2
Nside = 10
Nbott = 20
ax1 = fig.add_subplot(3,2,1)
ax1.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax1.set_title(xlabel, fontsize=14)
ax1.plot(xm1, ym1, '*', xp1, yp1, '-')
phi2_1, inn1, sff22_1, sAMP2_1, sAP2_1, _ = func22(xp1, xm1, yp1, ym1, nu)
ax2 = fig.add_subplot(3,2, 2)
matplotlib.pyplot.xlabel('m', fontsize=16)
ax2.plot( iinn1, phi2_1.real, '*')

D = 1
L = 1
Nside = 10
Nbott = 10
ax3= fig.add_subplot(3,2,3)
ax3.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax3.set_title(xlabel, fontsize=14)
ax3.plot(xm2, ym2, '*', xp2, yp2, '-')
phi2_2, inn2, sff22_2, sAMP2_2, sAP2_2, _ = func22(xp2, xm2, yp2, ym2, nu)
ax4 = fig.add_subplot(3,2, 4)
matplotlib.pyplot.xlabel('m', fontsize=16)
ax4.plot( iinn2, phi2_2.real, '*')

D = 1
L = 0.5
Nside = 10
Nbott = 5
ax5 = fig.add_subplot(3,2,5)
ax5.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
```

```
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax5.set_title(xlabel, fontsize=14)
ax5.plot(xm3, ym3, '*', xp3, yp3, '-')
phi2_3, inn3, sff22_3, sAMP2_3, sAP2_3, _ = func22(xp3, xm3, yp3, ym3, nu)
ax6 = fig.add_subplot(3,2, 6)
matplotlib.pyplot.xlabel('m', fontsize=16)
ax6.plot( iinn3, phi2_3.real, '*');
```



6. Far field behavior of ϕ_2

The potential ϕ_2 has far field form

$$\phi_2(\bar{x},\bar{y}) \to A_2^{-\infty} e^{K\,\bar{y}+i\,K\,\bar{x}}, \quad \bar{x}\to-\infty, \tag{15}$$
$$\phi_2(\bar{x},\bar{y}) \to A_2^{\infty} e^{K\,\bar{y}-i\,K\,\bar{x}}, \quad \bar{x}\to\infty, \tag{16}$$

where $K=\frac{\omega^2}{g}$. * Use (121) to derive expressions for $A_2^{-\infty}$ and A_2^{∞}.
(121):

$$2\pi\,\phi_2(\bar{x},\bar{y}) = \iint\limits_{S_B} \left(\phi_2\frac{\partial G}{\partial n} - G\,n_2\right) dS \tag{17}$$

$$2\pi\,A_2^{-\infty} e^{K\bar{y}+iK\bar{x}} = \iint\limits_{S_B} \left(\phi_2\frac{\partial G}{\partial n} - G\,n_2\right) dS, \quad \bar{x}\to-\infty \tag{18}$$
$$2\pi\,A_2^{\infty} e^{K\bar{y}-iK\bar{x}} = \iint\limits_{S_B} \left(\phi_2\frac{\partial G}{\partial n} - G\,n_2\right) dS, \quad \bar{x}\to\infty \tag{19}$$

$$A_2^{-\infty} = \frac{e^{-K\bar{y}-iK\bar{x}}}{2\pi}\iint\limits_{S_B} \left(\phi_2\frac{\partial G}{\partial n} - G\,n_2\right) dS, \quad \bar{x}\to-\infty \tag{20}$$
$$A_2^{\infty} = \frac{e^{-K\bar{y}+iK\bar{x}}}{2\pi}\iint\limits_{S_B} \left(\phi_2\frac{\partial G}{\partial n} - G\,n_2\right) dS, \quad \bar{x}\to\infty \tag{21}$$

C. The outgoing wave amplitude

The wave elevation of the outgoing waves has (complex) amplitudes

$$amp_2^{-\infty} = \xi_2 A_2^{-\infty}\frac{\omega^2}{g} \tag{22}$$
$$amp_2^{\infty} = \xi_2 A_2^{\infty}\frac{\omega^2}{g} \tag{23}$$

The mean energy flux of the outgoing waves is given by

$$En.Flux = \bar{E}^{\infty}c_g + \bar{E}^{-\infty}c_g \tag{24}$$

where $\bar{E}^{\infty}c_g = \frac{1}{2}\rho g\left|amp^{\pm\infty}\right|^2$ is the mean energy density of the outgoing waves and $c_g = \frac{\partial\omega}{\partial K}$ the group velocity.

D. Added mass and damping

- Use the numerical solution of ϕ_2 along the wetted body surface S_B to calculate the added mass a_{22} and the damping b_{22} for the wavenumber range $0 < K\,D = \frac{\omega^2 D}{g} < 2$ for the three geometries.
- Calculate also b_{22} from the energy balance

$$\frac{1}{2}\left|\xi_2\right|^2\omega^2 b_{22} = \bar{E}^{\infty}c_g + \bar{E}^{-\infty}c_g \tag{25}$$

$$b_{22} = \frac{2}{|\xi_2|^2 \omega^2} \left(\bar{E}^{\infty} c_g + \bar{E}^{-\infty} c_g \right) \tag{26}$$

$$\bar{E}^{\pm\infty} = \frac{\rho g}{2} \left| amp_2^{\pm\infty} \right|^2 = \frac{\rho g}{2} \left| \xi_2 A_2^{\pm\infty} \frac{\omega^2}{g} \right|^2 \tag{27}$$

$$b_{22} = \frac{2}{|\xi_2|^2 \omega^2} c_g \left(\frac{\rho g}{2} \left| \xi_2 A_2^{\infty} \frac{\omega^2}{g} \right|^2 + \frac{\rho g}{2} \left| \xi_2 A_2^{-\infty} \frac{\omega^2}{g} \right|^2 \right) = \frac{c_g \rho \omega^2}{g} \left(\left| A_2^{-\infty} \right|^2 + |A_2^{\infty}|^2 \right) \tag{28}$$

I dypt vann,

$$c_g = \frac{\partial \omega}{\partial K} = \frac{\partial \sqrt{gK}}{\partial K} = \frac{1}{2} \sqrt{\frac{g}{K}} = \frac{1}{2} \sqrt{\frac{g^2}{\omega^2}} = \frac{1}{2} \frac{g}{\omega} \tag{29}$$

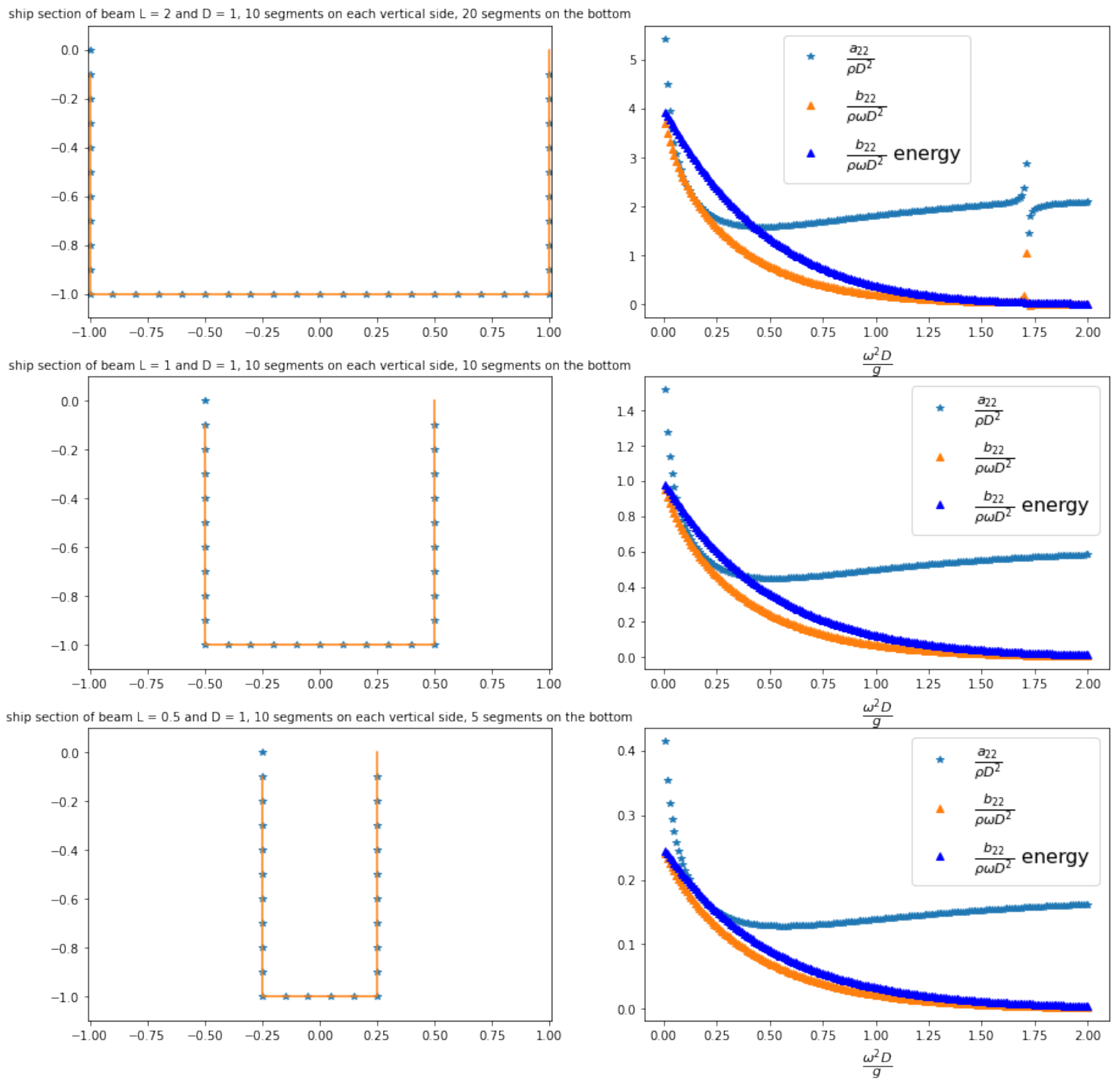
$$b_{22} = \rho \omega \left(\left| A_2^{-\infty} \right|^2 + |A_2^{\infty}|^2 \right) \tag{30}$$

```
In [52]: def funcD(xp, xm, yp, ym, nu):
             xp = numpy.array(xp)
             xm = numpy.array(xm)
             yp = numpy.array(yp)
             ym = numpy.array(ym)
             NN = xp.shape[0]   # = N= Nside + Nbott + Nside
             iinn = numpy.linspace(0, NN, num = NN, endpoint=True)
             dx = xp - xm
             dy = yp - ym
             ds = numpy.sqrt(dx**2 + dy**2)
             bx = 0.5*(xm + xp)
             by = 0.5*(ym + yp)
             n1 = - (yp - ym)/ds
             n2 = (xp - xm)/ds
             #incoming wave potential
             phi0 = numpy.exp(nu*(by- complex(0,1)*bx))
             ss = numpy.zeros((NN,NN), dtype=complex)
             #contributions to integral equation, rhs stores rhs, lhs stores lhs
             for i in range(0,NN):
                 for j in range(0,NN):
                     xa = bx[j] - bx[i]
                     yb = by[j] + by[i]
                     zz = nu* (yb - complex(0,1)*xa)
                     f1 = -2*numpy.exp(zz)*(mpmath.e1(zz) + numpy.log(zz) - numpy.log(-zz))
                     f2 = 2*numpy.pi*numpy.exp(zz)
                     g2 = f1.real + complex(0,1)*f2.real
                     # lhs
                     if j-i ==0:
                         arg0 = - math.pi
                     else:
                         arg0 =(numpy.log((xm[j]-bx[i]+complex(0,1)*(ym[j]-by[i]))/(xp[j]-bx[i]+c
         omplex(0,1)*(yp[j]-by[i])))).imag

                     arg1= (numpy.log((xm[j] - bx[i]+complex(0,1)*(ym[j] + by[i]))/(xp[j] -bx[i]
         + complex(0,1)*(yp[j] + by[i])))).imag
                     help1= (n1[j]*(f1.imag+complex(0,1)*f2.imag) + n2[j]*(f1.real +
         complex(0,1)*f2.real))*nu*ds[j]
```

```
                ss[i,j] = arg0 + arg1 + help1
        rhsD = -2*numpy.pi*phi0
        phiD = scipy.linalg.solve(ss, rhsD)    # in matlab: ss\rhs

        XX2 = phiD* n2* ds
        sXX2 = sum(XX2)
        X2 = - complex(0,1) * sXX2

        return phiD, iinn, X2
```

In [31]:
```
nu = numpy.linspace(0.0, 2.0, num = 201, endpoint=True)[1:]
fig = matplotlib.pyplot.figure(figsize=[15,15])
D = 1
L = 2
Nside = 10
Nbott = 20
ax1 = fig.add_subplot(3,2,1)
ax1.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax1.set_title(xlabel, fontsize=10)
ax1.plot(xm1, ym1, '*', xp1, yp1, '-')
sff2_1 = []
X2_1 = []
for nui in nu:
    _,_,sff2_, _, _, _ = func22(xp1, xm1, yp1, ym1, nui)
    sff2_1.append(sff2_)
    _,_,X2_ = funcD(xp1, xm1, yp1, ym1, nui)
    X2_1.append(X2_)
ax2 = fig.add_subplot(3,2, 2)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=16)
a221, = ax2.plot( nu, numpy.real(sff2_1), '*', label = r'$\frac{a_{22}}{\rho D^2}$')
b221, = ax2.plot( nu, - numpy.imag(sff2_1), '^', label = r'$\frac{b_{22}}{\rho \omega
D^2}$')
b221e, = ax2.plot( nu, 4*numpy.square(numpy.exp(-nu*D)*numpy.sin(nu*L/2)/nu), 'b^',
label = r'$\frac{b_{22}}{\rho \omega D^2}$'+' energy')
#ax2.plot( nu, numpy.square(numpy.absolute(X2_1)), 'b^', label = r'$\frac{b_{22}}{\rho
\omega D^2}$'+' energy')
matplotlib.pyplot.legend(handles= [a221, b221, b221e], fontsize= 16);

D = 1
L = 1
Nside = 10
Nbott = 10
ax3= fig.add_subplot(3,2,3)
ax3.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax3.set_title(xlabel, fontsize=10)
ax3.plot(xm2, ym2, '*', xp2, yp2, '-')
sff2_2 = []
X2_2 = []
for nui in nu:
    _,_,sff2_, _, _, _ = func22(xp2, xm2, yp2, ym2, nui)
    sff2_2.append(sff2_)
    _,_,X2_ = funcD(xp2, xm2, yp2, ym2, nui)
    X2_2.append(X2_)
```

```
ax4 = fig.add_subplot(3,2, 4)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=16)
a222, = ax4.plot( nu, numpy.real(sff2_2), '*', label = r'$\frac{a_{22}}{\rho D^2}$')
b222, = ax4.plot( nu, - numpy.imag(sff2_2), '^', label = r'$\frac{b_{22}}{\rho \omega
D^2}$')
b222e, = ax4.plot( nu, 4*numpy.square(numpy.exp(-nu*D)*numpy.sin(nu*L/2)/nu), 'b^',
label = r'$\frac{b_{22}}{\rho \omega D^2}$'+' energy')
#ax4.plot( nu, numpy.square(numpy.absolute(X2_2)), 'b^', label = r'$\frac{b_{22}}{\rho
\omega D^2}$'+' energy')
matplotlib.pyplot.legend(handles= [a222, b222, b222e], fontsize= 16);

D = 1
L = 0.5
Nside = 10
Nbott = 5
ax5 = fig.add_subplot(3,2,5)
ax5.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax5.set_title(xlabel, fontsize=10)
ax5.plot(xm3, ym3, '*', xp3, yp3, '-')
sff2_3 = []
X2_3 = []
for nui in nu:
    _,_,sff2_, _, _,_ = func22(xp3, xm3, yp3, ym3, nui)
    sff2_3.append(sff2_)
    _,_,X2_ = funcD(xp3, xm3, yp3, ym3, nui)
    X2_3.append(X2_)
ax6 = fig.add_subplot(3,2, 6)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=16)
a223, = ax6.plot( nu, numpy.real(sff2_3), '*', label = r'$\frac{a_{22}}{\rho D^2}$')
b223, = ax6.plot( nu, - numpy.imag(sff2_3), '^', label = r'$\frac{b_{22}}{\rho \omega
D^2}$')
b223e, = ax6.plot( nu, 4*numpy.square(numpy.exp(-nu*D)*numpy.sin(nu*L/2)/nu), 'b^',
label = r'$\frac{b_{22}}{\rho \omega D^2}$'+' energy')
#ax6.plot( nu, numpy.square(numpy.absolute(X2_3)), 'b^', label = r'$\frac{b_{22}}{\rho
\omega D^2}$'+' energy')
matplotlib.pyplot.legend(handles= [a223, b223, b223e], fontsize= 16);
```



E. Aproximate solution

- Solve problem 6.17 in Newman [1]

Bevegelsesligningen for det flytende legemet er

$$Re\left[e^{i\omega t}\left(i\omega\right)^2\xi_2 M_{22}\right] = Re\left[F_2 e^{i\omega t}\right] \tag{31}$$

$$F_2 = -c_{22}\xi_2 - \xi_2 \left[(i\omega)^2 a_{22} + i\omega b_{22} \right] + AX_2 \tag{32}$$

$$c_{22} = \rho g S, \qquad \text{S svømme flaten} \tag{33}$$

$$S = \pi \frac{d^2}{4} \tag{34}$$

$$M_{22} = m = \rho \forall \qquad \text{Archimedes} \tag{35}$$

$$\forall = ST = \pi \frac{d^2}{4} T \qquad \text{fortrengt volum} \tag{36}$$

hvor $\xi_2 c_{22}$ er hydrostatiske krefter, a_{22} er addert masse, b_{22} er dempingskoefficient og AX_2 er eksitasjonskrefter.

$$Re \left[e^{i\omega t} \left[(i\omega)^2 \xi_2 M_{22} + \xi_2 c_{22} + \xi_2 \left[(i\omega)^2 a_{22} + i\omega b_{22} \right] - AX_2 \right] \right] = 0 \tag{37}$$

$$\left[-\omega^2 \left(M_{22} + a_{22} \right) + i\omega b_{22} + c_{22} \right] \xi_2 = AX_2 \tag{38}$$

Resonans skjer når $c_{22} - \omega^2 \left(M_{22} + a_{22} \right) = 0$.

$$\omega_n^2 = \frac{c_{22}}{a_{22} + M_{22}} = \frac{\rho g S}{a_{22} + \rho \forall} = \frac{\rho g \pi \frac{d^2}{4}}{a_{22} + \rho \pi \frac{d^2}{4} T} \tag{39}$$

Hvis $a_{22} \ll m$,

$$\omega_n^2 = \frac{c_{22}}{a_{22} + M_{22}} = \frac{\rho g S}{a_{22} + \rho \forall} = \frac{\rho g \pi \frac{d^2}{4}}{a_{22} + \rho \pi \frac{d^2}{4} T} \approx \frac{\rho g \pi \frac{d^2}{4}}{\rho \pi \frac{d^2}{4} T} = \frac{g}{T} \tag{40}$$

Resonansfrekvensen er:

$$\omega_n \approx \sqrt{\frac{g}{T}} \tag{41}$$

På (2D) resonans

$$\frac{\xi_2}{A} = \frac{X_2}{i\omega b_{22}} \tag{42}$$

$$\frac{b_{22}}{\rho \omega} = \left| \frac{X_2}{\rho g} \right|^2, \qquad \text{Haskind} \tag{43}$$

Froude-Krylov kraften er definert som

$$X_j^{FK} = -i\omega\rho \iint_{S_B} \phi_0 n_j \, dS \tag{44}$$

For hiv, $j = 2$. $\phi_0 = e^{Ky - iKx}$

$$X_2^{FK} = -i\omega\rho \iint_{S_B} \phi_0 n_2 \, dS = \rho g \int_{-d/2}^{d/2} e^{-KT - iKx} dx = \rho g d e^{-KT} \frac{\sin(Kd/2)}{Kd/2} \tag{45}$$

$$b_{22} = \rho \omega \left| \frac{X_2}{\rho g} \right|^2 = \frac{\omega}{g^2 \rho} \left| \frac{\rho g d e^{-KT} \sin(Kd/2)}{Kd/2} \right|^2 = \frac{4\omega\rho}{g} \left| \frac{e^{-KT} \sin(Kd/2)}{K} \right|^2 \tag{46}$$

Hiv responsen:

$$\frac{\xi_2}{A} = \frac{X_2}{i\omega b_{22} + c_{22} - \omega^2 (m + a_{22})} = \frac{2\rho g d e^{-KT} \frac{\sin(Kd/2)}{Kd}}{\rho g S - \omega^2 (m + a_{22}) + i4 \frac{\omega^2 \rho}{g} \left| \frac{e^{-KT} \sin(Kd/2)}{K} \right|^2} \tag{47}$$

F. The diffraction problem

In the diffraction problem the geometry is fixed. The fluid motion is given by the velocity potential

$$\Phi_D(x, y, t) = Re\left(A\phi_D(x, y)e^{i\omega t}\right), \tag{48}$$

where A is the amplitude of the incoming waves, $\phi_D(x, y) = \phi_0(x, y) + \phi_7(x, y)$, $\phi_0 = \frac{ig}{\omega}e^{Ky-iKx}$ denotesthe potential of the incoming waves (given) and ϕ_7 is the scattering potential (unknown). Further, $K = \frac{\omega^2}{g}$. The integral equation to determine the sum $\phi_D = \phi_0 + \phi_7$ for a point $(\bar{x}, \bar{y})$ on S_B is

$$-\pi\phi_D(\bar{x}, \bar{y}) + \iint_{S_B} \phi_D \frac{\partial G}{\partial n} dS = -2\pi\phi_0(\bar{x}, \bar{y}). \tag{49}$$

* Solve (132) numerically for the three rectangular sections.

In [ ]:
```
nu = 0.9
fig = matplotlib.pyplot.figure(figsize=[15,15])
D = 1
L = 2
Nside = 10
Nbott = 20
ax1 = fig.add_subplot(3,2,1)
ax1.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax1.set_title(xlabel, fontsize=10)
ax1.plot(xm1, ym1, '*', xp1, yp1, '-')
phiD_1, inn1, X2_1 = funcD(xp1, xm1, yp1, ym1, nu)
ax2 = fig.add_subplot(3,2, 2)
matplotlib.pyplot.xlabel('m', fontsize=16)
phiDreal1, = ax2.plot( iinn1, phiD_1.real, '*', label = r'$Re(\phi_D)$')
phiDimag1, = ax2.plot( iinn1, phiD_1.imag, '^', label = r'$Im(\phi_D)$')
matplotlib.pyplot.legend(handles= [phiDreal1, phiDimag1], fontsize = 14)


D = 1
L = 1
Nside = 10
Nbott = 10
ax3= fig.add_subplot(3,2,3)
ax3.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax3.set_title(xlabel, fontsize=10)
ax3.plot(xm2, ym2, '*', xp2, yp2, '-')
phiD_2, inn2, X2_2 = funcD(xp2, xm2, yp2, ym2, nu)
ax4 = fig.add_subplot(3,2, 4)
matplotlib.pyplot.xlabel('m', fontsize=16)
phiDreal2, = ax4.plot( iinn2, phiD_2.real, '*', label = r'$Re(\phi_D)$')
phiDimag2, = ax4.plot( iinn2, phiD_2.imag, '^', label = r'$Im(\phi_D)$')
matplotlib.pyplot.legend(handles= [phiDreal2, phiDimag2], fontsize = 14)

D = 1
L = 0.5
Nside = 10
Nbott = 5
```

```
ax5 = fig.add_subplot(3,2,5)
ax5.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax5.set_title(xlabel, fontsize=10)
ax5.plot(xm3, ym3, '*', xp3, yp3, '-')
phiD_3, inn3, X2_3 = funcD(xp3, xm3, yp3, ym3, nu)
ax6 = fig.add_subplot(3,2, 6)
matplotlib.pyplot.xlabel('m', fontsize=16)
phiDreal3, = ax6.plot( iinn3, phiD_3.real, '*', label = r'$Re(\phi_D)$')
phiDimag3, = ax6.plot( iinn3, phiD_3.imag, '^', label = r'$Im(\phi_D)$')
matplotlib.pyplot.legend(handles= [phiDreal3, phiDimag3], fontsize=14);
```

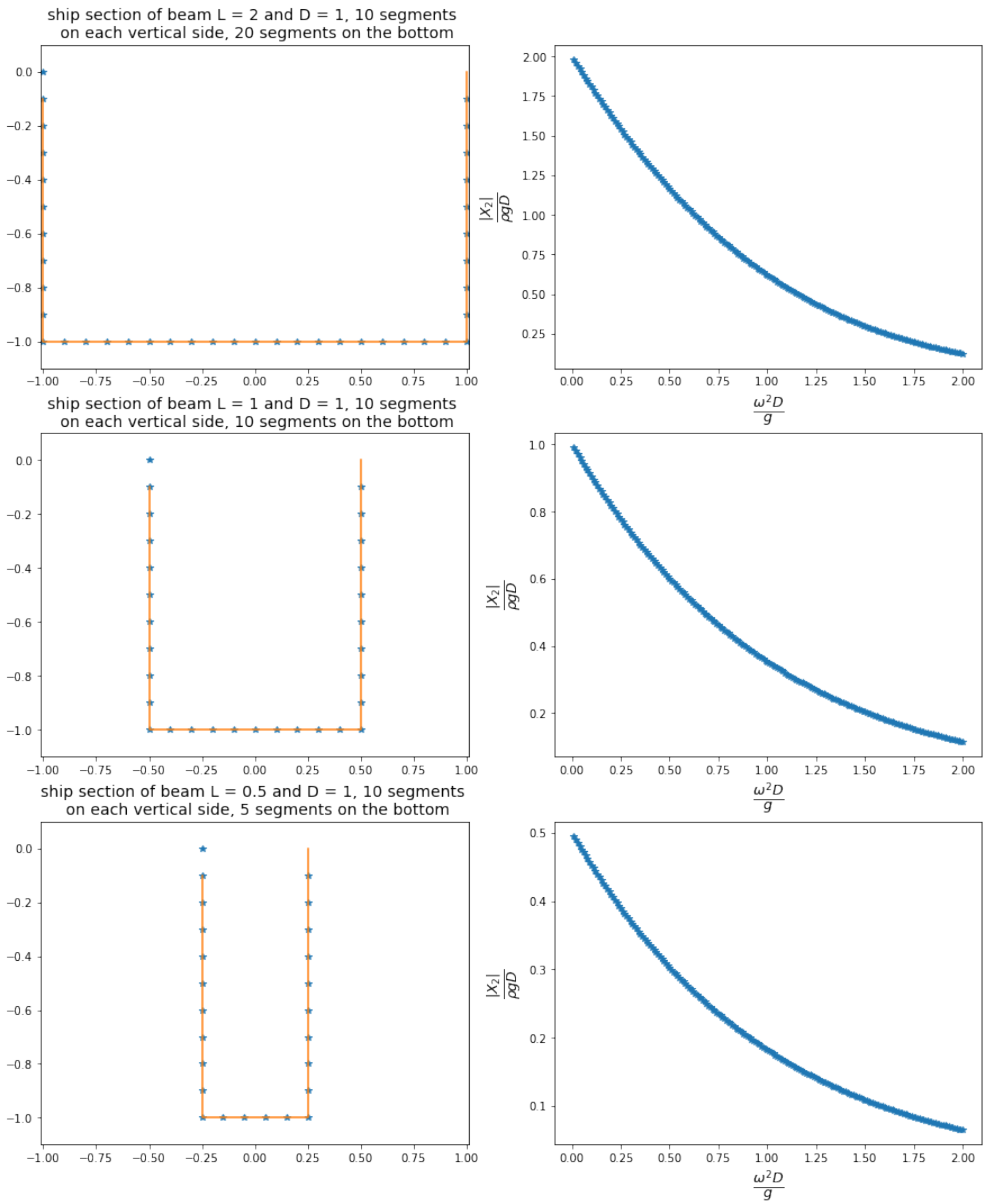
1. The exciting force

- Obtain numerically the exciting force

$$\frac{X_2}{\rho g} = -\frac{i\omega}{g}\iint\limits_{S_B} \phi_D n_2\, dS \tag{50}$$

```
In [53]: nu = numpy.linspace(0.0, 2.0, num = 201, endpoint=True)[1:]
         fig = matplotlib.pyplot.figure(figsize=[15,18])
         D = 1
         L = 2
         Nside = 10
         Nbott = 20
         ax1 = fig.add_subplot(3,2,1)
         ax1.axis([-1.01, 1.01, -1.1, 0.1])
         xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
         segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
         ax1.set_title(xlabel, fontsize=14)
         ax1.plot(xm1, ym1, '*', xp1, yp1, '-')
         X2_1 = []
         for nui in nu:
             _,_,X2_ = funcD(xp1, xm1, yp1, ym1, nui)
             X2_1.append(X2_)
         ax2 = fig.add_subplot(3,2, 2)
         matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
         matplotlib.pyplot.ylabel(r'$\frac{\left|X_{2}\right|}{\rho g D}$', fontsize=20)
         ax2.plot( nu, 2*numpy.exp(-nu*D)*numpy.sin(nu*L/2)/nu, '*', label =
         r'$\frac{\left|X_{2}\right|}{\rho g D}$');
                  #numpy.absolute(X2_1), '*', label = r'$\frac{\left|X_{2}\right|}{\rho g D}$')

         D = 1
         L = 1
         Nside = 10
         Nbott = 10
         ax3= fig.add_subplot(3,2,3)
         ax3.axis([-1.01, 1.01, -1.1, 0.1])
         xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
         segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
         ax3.set_title(xlabel, fontsize=14)
         ax3.plot(xm2, ym2, '*', xp2, yp2, '-')
         X2_2 = []
         for nui in nu:
```

```
    _,_,X2_ = funcD(xp2, xm2, yp2, ym2, nui)
    X2_2.append(X2_)
ax4 = fig.add_subplot(3,2, 4)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
matplotlib.pyplot.ylabel(r'$\frac{\left|X_{2}\right|}{\rho g D}$', fontsize=20)
ax4.plot( nu, 2*numpy.exp(-nu*D)*numpy.sin(nu*L/2)/nu, '*', label =
r'$\frac{\left|X_{2}\right|}{\rho g D}$');
         #numpy.absolute(X2_2), '*', label = r'$\frac{\left|X_{2}\right|}{\rho g D}$')

D = 1
L = 0.5
Nside = 10
Nbott = 5
ax5 = fig.add_subplot(3,2,5)
ax5.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax5.set_title(xlabel, fontsize=14)
ax5.plot(xm3, ym3, '*', xp3, yp3, '-')
X2_3 = []
for nui in nu:
    _,_,X2_ = funcD(xp3, xm3, yp3, ym3, nui)
    X2_3.append(X2_)
ax6 = fig.add_subplot(3,2, 6)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
matplotlib.pyplot.ylabel(r'$\frac{\left|X_{2}\right|}{\rho g D}$', fontsize=20)
ax6.plot( nu, 2*numpy.exp(-nu*D)*numpy.sin(nu*L/2)/nu, '*', label =
r'$\frac{\left|X_{2}\right|}{\rho g D}$');
         #numpy.absolute(X2_3), '*', label = r'$\frac{\left|X_{2}\right|}{\rho g D}$');
```



2. Haskind relations

- Show that

$$\frac{X_2^{\mathrm{Haskind},2}}{\rho g} = iA_2^{-\infty} \tag{51}$$

$$\frac{X_2^{\text{Haskind},2}}{i\omega\rho} = \iint\limits_{S_B} \left(\phi_0 \frac{\partial \phi_2}{\partial n} - \phi_2 \frac{\partial \phi_0}{\partial n} \right) dS \tag{52}$$

$$\phi_2 = A_2^{-\infty} e^{Ky+iKx}, \qquad x \to -\infty \tag{53}$$

$$\phi_0 = \frac{ig}{\omega} e^{Ky-iKx} \tag{54}$$

$$\frac{X_2^{\text{Haskind},2}}{\rho} = i\omega \iint\limits_{S_B} \left(\phi_0 \frac{\partial \phi_2}{\partial n} - \phi_2 \frac{\partial \phi_0}{\partial n} \right) dS = i\omega \iint\limits_{S_B} \left(\frac{ig}{\omega} e^{Ky-iKx} \frac{\partial \phi_2}{\partial n} - \phi_2 \frac{\partial \left(\frac{ig}{\omega} e^{Ky-iKx} \right)}{\partial n} \right) dS \tag{55}$$

$$= -g \iint\limits_{S_B} \left(e^{Ky-iKx} \frac{\partial \phi_2}{\partial n} - \phi_2 \frac{\partial \left(e^{Ky-iKx} \right)}{\partial n} \right) dS \tag{56}$$

$$\frac{X_2^{\text{Haskind},2}}{\rho g} = -\iint\limits_{S_B} \left(e^{Ky-iKx} \frac{\partial \phi_2}{\partial n} - \phi_2 \frac{\partial \left(e^{Ky-iKx} \right)}{\partial n} \right) dS = -\iint\limits_{S_B} \left(e^{Ky-iKx} n_2 - \phi_2 \left(n_1 \frac{\partial}{\partial x} + n_2 \frac{\partial}{\partial y} \right) e^{Ky-iKx} \right) dS \tag{57}$$

$$= -\iint\limits_{S_B} \left[n_2 - \phi_2 \left(n_1 \frac{\partial}{\partial x} + n_2 \frac{\partial}{\partial y} \right) \right] e^{Ky-iKx} dS = iA_2^{-\infty} \tag{58}$$

- Calculate the exciting force $\frac{X_2}{\rho g}$ for the three sections using the direct pressure integration (133), the version 1 of the Haskind relation (136) and the version 2 of the Haskind relation (140).

(136):

$$\frac{X_2^{\text{Haskind},1}}{i\omega\rho} = -\iint\limits_{S_B} \left(\phi_0 \frac{\partial \phi_2}{\partial n} - \phi_2 \frac{\partial \phi_0}{\partial n} \right) dS \tag{59}$$

(140):

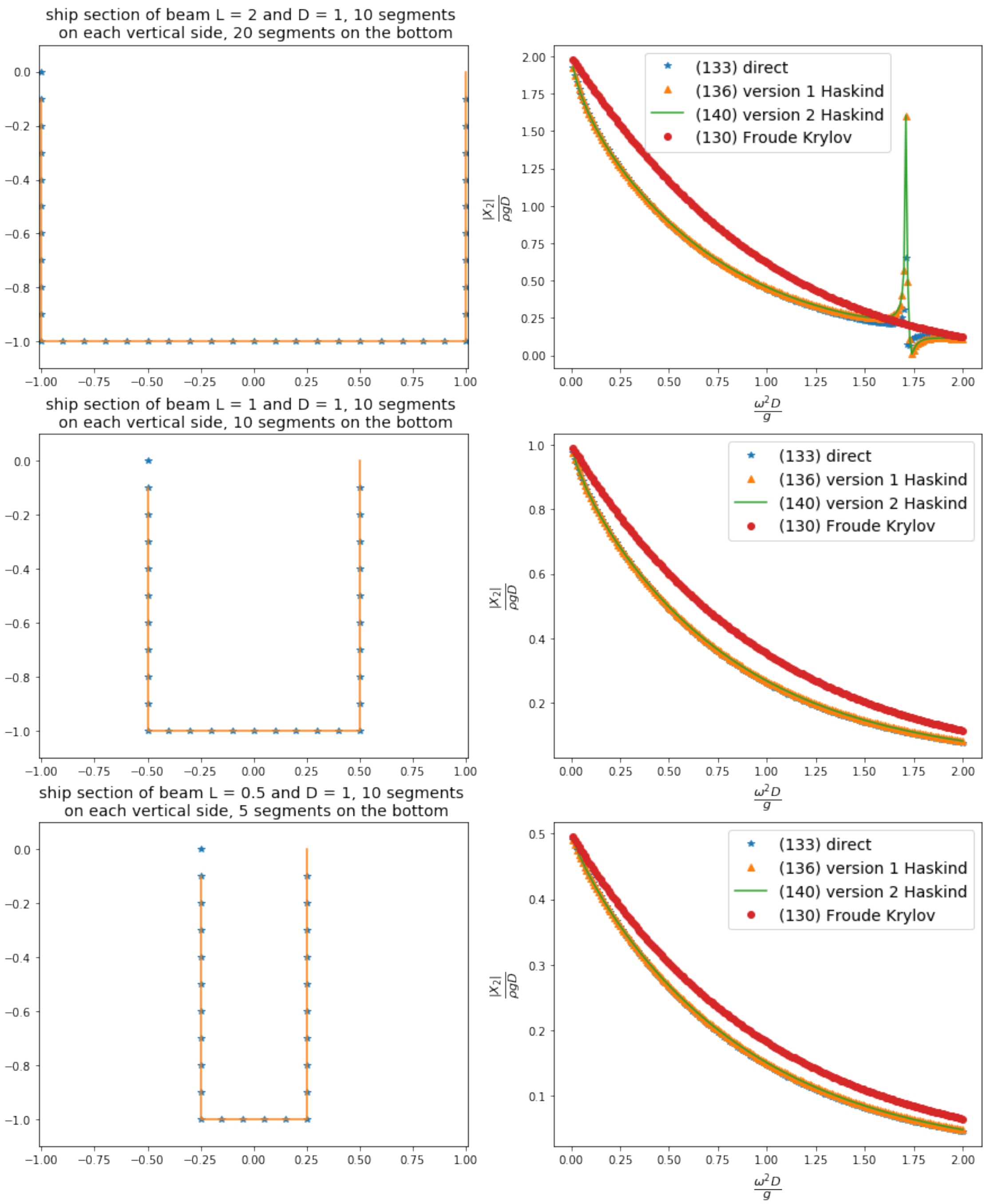
$$\frac{X_2^{\text{Haskind},2}}{\rho g} = iA_2^{-\infty} \tag{60}$$

- Compare also the Froude Krylov approximation in (130). Perform calculations for the wavenumber range $0 < K\,D < 2$.

In [54]:
```
nu = numpy.linspace(0.0, 2.0, num = 201, endpoint=True)[1:]
fig = matplotlib.pyplot.figure(figsize=[18,18])
D = 1
L = 2
Nside = 10
Nbott = 20
ax1 = fig.add_subplot(3,2,1)
ax1.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax1.set_title(xlabel, fontsize=14)
ax1.plot(xm1, ym1, '*', xp1, yp1, '-')
#X2_1 = []
X2136_1 = []
```

```
X2140_1 = []
for nui in nu:
    _, _, _, sAM2_, _, v1H_ = func22(xp1, xm1, yp1, ym1, nui)
    X2140_1.append(sAM2_)
    X2136_1.append(v1H_)
X130_1 = 2*numpy.exp(- nu*D)*numpy.sin(nu*L/2)/nu
ax2 = fig.add_subplot(3,2, 2)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=18)
matplotlib.pyplot.ylabel(r'$\frac{\left|X_{2}\right|}{\rho g D}$', fontsize=18)
X133, = ax2.plot( nu, numpy.absolute(X2_1), '*', label = '(133) direct')
X136, = ax2.plot( nu, numpy.absolute(X2136_1), '^', label = '(136) version 1 Haskind')
X140, = ax2.plot( nu, numpy.absolute(X2140_1), '-', label = '(140) version 2 Haskind')
X130, = ax2.plot( nu, numpy.absolute(X130_1), 'o', label = '(130) Froude Krylov' )
matplotlib.pyplot.legend(handles= [X133, X136, X140, X130], fontsize = 14)

D = 1
L = 1
Nside = 10
Nbott = 10
ax3= fig.add_subplot(3,2,3)
ax3.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax3.set_title(xlabel, fontsize=14)
ax3.plot(xm2, ym2, '*', xp2, yp2, '-')
#X2_2 = []
X2136_2 = []
X2140_2 = []
for nui in nu:
    _, _, _, sAM2_, _, v1H_ = func22(xp2, xm2, yp2, ym2, nui)
    X2140_2.append(sAM2_)
    X2136_2.append(v1H_)
X130_2 = 2*numpy.exp(- nu*D)*numpy.sin(nu*L/2)/nu
ax4 = fig.add_subplot(3,2, 4)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=18)
matplotlib.pyplot.ylabel(r'$\frac{\left|X_{2}\right|}{\rho g D}$', fontsize=18)
X133, = ax4.plot( nu, numpy.absolute(X2_2), '*', label = '(133) direct')
X136, = ax4.plot( nu, numpy.absolute(X2136_2), '^', label = '(136) version 1 Haskind')
X140, = ax4.plot( nu, numpy.absolute(X2140_2), '-', label = '(140) version 2 Haskind')
X130, = ax4.plot( nu, numpy.absolute(X130_2), 'o', label = '(130) Froude Krylov' )
matplotlib.pyplot.legend(handles= [X133, X136, X140, X130], fontsize = 14)

D = 1
L = 0.5
Nside = 10
Nbott = 5
ax5 = fig.add_subplot(3,2,5)
ax5.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax5.set_title(xlabel, fontsize=14)
ax5.plot(xm3, ym3, '*', xp3, yp3, '-')
#X2_3 = []
X2136_3 = []
X2140_3 = []
for nui in nu:
```

```
    _, _, _, sAM2_, _, v1H_ = func22(xp3, xm3, yp3, ym3, nui)
    X2140_3.append(sAM2_)
    X2136_3.append(v1H_)
X130_3 = 2*numpy.exp(- nu*D)*numpy.sin(nu*L/2)/nu
ax6 = fig.add_subplot(3,2, 6)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=18)
matplotlib.pyplot.ylabel(r'$\frac{\left|X_{2}\right|}{\rho g D}$', fontsize=18)
X133, = ax6.plot( nu, numpy.absolute(X2_3), '*', label = '(133) direct')
X136, = ax6.plot( nu, numpy.absolute(X2136_3), '^', label = '(136) version 1 Haskind')
X140, = ax6.plot( nu, numpy.absolute(X2140_3), '-', label = '(140) version 2 Haskind')
X130, = ax6.plot( nu, numpy.absolute(X130_3), 'o', label = '(130) Froude Krylov' )
matplotlib.pyplot.legend(handles= [X133, X136, X140, X130], fontsize = 14);
```



3. Body response in heave

- Formulate the equation of motion in the heave mode of motion (assuming no motion in the other modes). Obtain an expression for the response $|\xi_2|/A$.

$$\xi_2 \left[-\omega^2 (m + a_{22}) + i\omega b_{22} + c_{22}\right] = A\, X_2 \tag{61}$$

$c_{i2} = 0,\ i \neq 2$

$$\frac{\xi_2}{A} = \frac{X_2}{-\omega^2\left(m + a_{22}\right) + i\omega b_{22} + c_{22}} \tag{62}$$

4. Resonance frequency

- Determine the resonance frequency of the three rectangular sections with $L/D = 2, 1$ and 0.1.

$$\omega_n = \sqrt{\frac{c_{22}}{a_{22} + m}} \tag{63}$$

$$c_{22} = \rho g S \tag{64}$$

$$m = \rho S D \tag{65}$$

$$\omega_n = \sqrt{\frac{g}{D}\frac{1}{\frac{a_{22}}{\rho D L} + 1}} \tag{66}$$

In [82]:
```
nu = numpy.linspace(0.0, 2.0, num = 201, endpoint=True)[1:]
fig = matplotlib.pyplot.figure(figsize=[18,24])
D = 1
L = 2
Nside = 10
Nbott = 20
ax1 = fig.add_subplot(3,2,1)
ax1.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax1.set_title(xlabel, fontsize=14)
ax1.plot(xm1, ym1, '*', xp1, yp1, '-')
omegan1 = numpy.sqrt(1/(D*(1+numpy.real(sff2_1)*D/L)))
ax2 = fig.add_subplot(3,2, 2)
ax2.set_title('resonance frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
matplotlib.pyplot.ylabel(r'$\frac{\omega_n \sqrt{D}}{\sqrt{g}}  $', fontsize=20)
ax2.plot( nu, omegan1 , '*')

D = 1
L = 1
Nside = 10
Nbott = 10
ax3= fig.add_subplot(3,2,3)
ax3.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax3.set_title(xlabel, fontsize=14)
ax3.plot(xm2, ym2, '*', xp2, yp2, '-')
omegan2 = numpy.sqrt(1/(D*(1+numpy.real(sff2_2)*D/L)))
ax4 = fig.add_subplot(3,2, 4)
ax4.set_title('resonance frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
matplotlib.pyplot.ylabel(r'$\frac{\omega_n \sqrt{D}}{\sqrt{g}}  $', fontsize=20)
ax4.plot( nu, omegan2, '*')

D = 1
L = 0.5
```

```
Nside = 10
Nbott = 5
ax5 = fig.add_subplot(3,2,5)
ax5.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax5.set_title(xlabel, fontsize=14)
ax5.plot(xm3, ym3, '*', xp3, yp3, '-')
omegan3 = numpy.sqrt(1/(D*(1+numpy.real(sff2_3)*D/L)))
ax6 = fig.add_subplot(3,2, 6)
ax6.set_title('resonance frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
matplotlib.pyplot.ylabel(r'$\frac{\omega_n \sqrt{D}}{\sqrt{g}}  $', fontsize=20)
ax6.plot( nu, omegan3, '*');
```

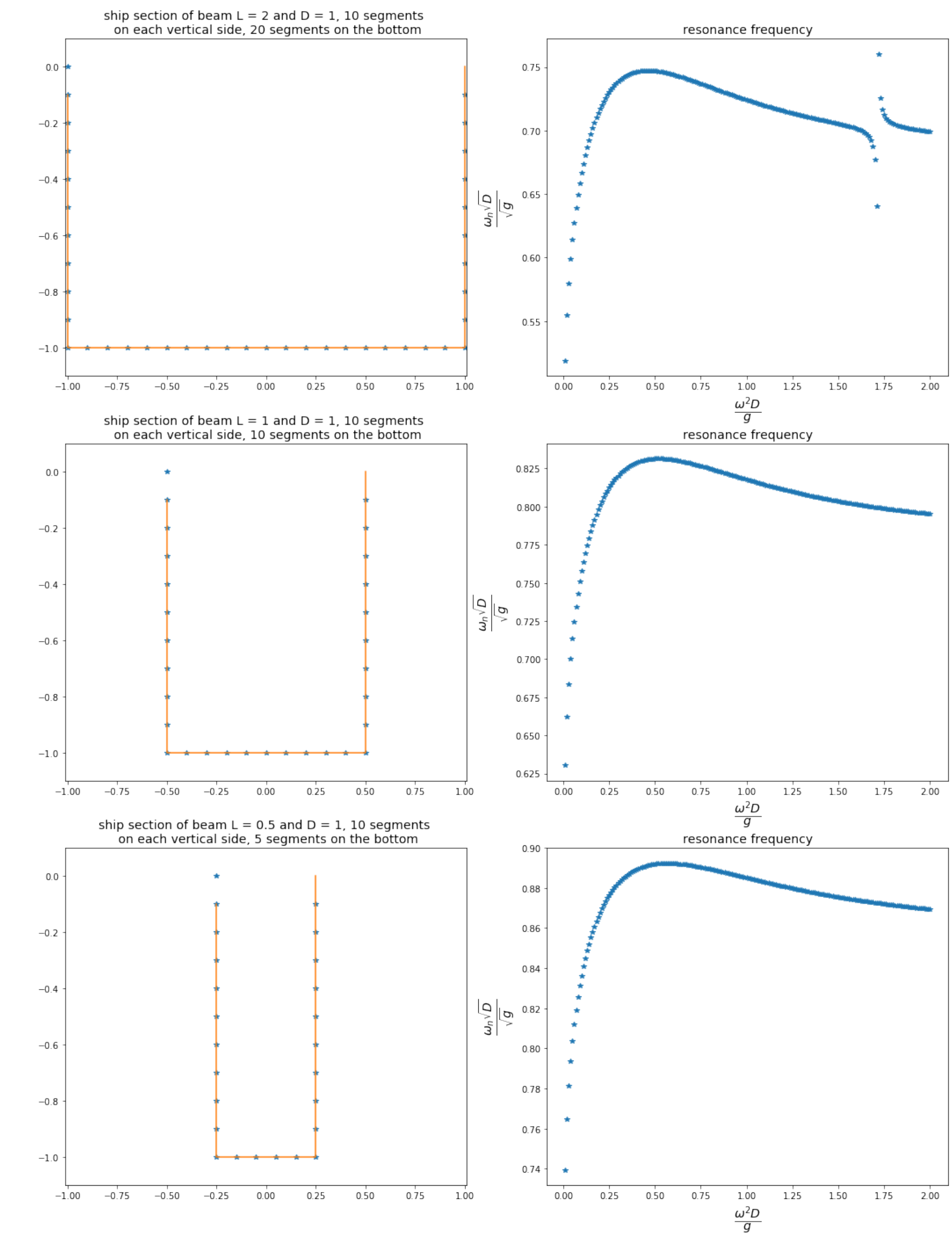


Table 1: Resonance frequency.

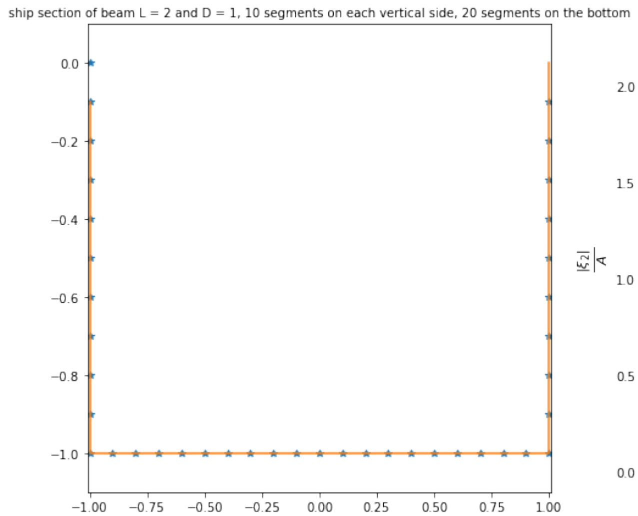
Geometri (L/D)	ω_n (Hz)
2	0.72
1	0.81
0.5	0.88

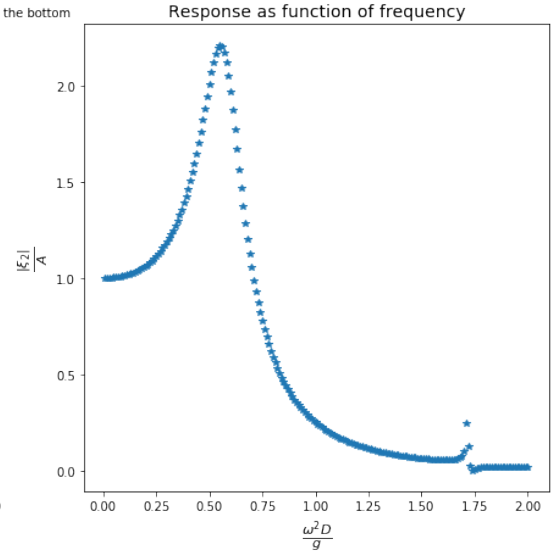
5. Response as a function of the frequency

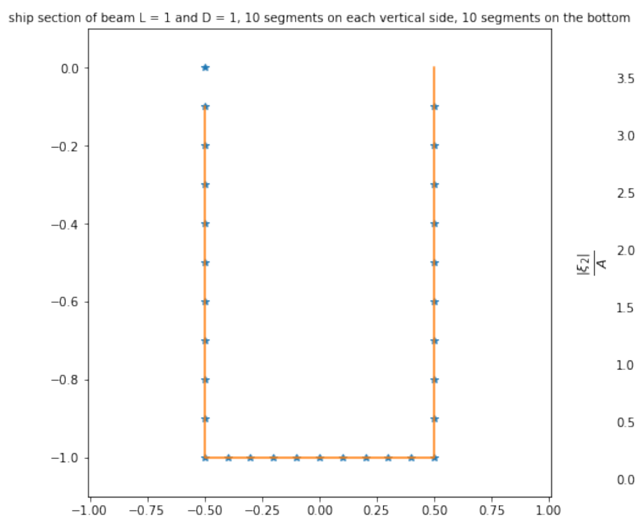
- Plot $\left|\frac{\xi_2}{A}\right|$ as function of the wavenumber for each of the three geometries for $0 < K\,D < 2$.

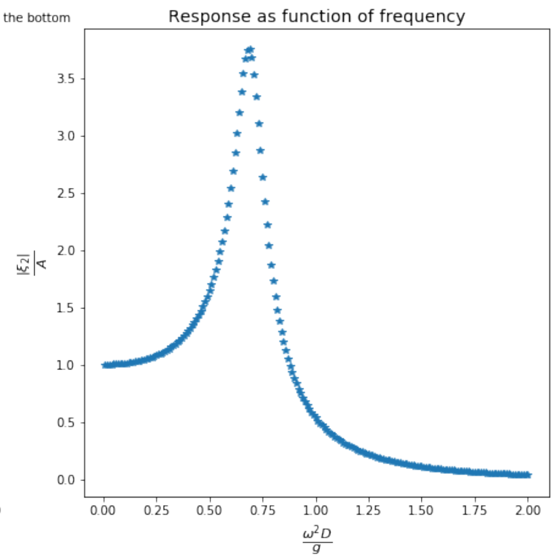
In [84]:
```
nu = numpy.linspace(0.0, 2.0, num = 201, endpoint=True)[1:]
fig = matplotlib.pyplot.figure(figsize=[15,24])
D = 1
L = 2
Nside = 10
Nbott = 20
ax1 = fig.add_subplot(3,2,1)
ax1.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax1.set_title(xlabel, fontsize=10)
ax1.plot(xm1, ym1, '*', xp1, yp1, '-')
xiA1 = numpy.absolute(X2140_1/(L - nu* (L + numpy.real(sff2_1) * D) + complex(0,1)* nu*
D* numpy.imag(sff2_1)))
ax2 = fig.add_subplot(3,2, 2)
ax2.set_title('Response as function of frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=16)
matplotlib.pyplot.ylabel(r'$\frac{\left|\xi_2 \right|}{A}  $', fontsize=16)
ax2.plot( nu, xiA1 , '*')

D = 1
L = 1
Nside = 10
Nbott = 10
ax3= fig.add_subplot(3,2,3)
ax3.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax3.set_title(xlabel, fontsize=10)
ax3.plot(xm2, ym2, '*', xp2, yp2, '-')
xiA2 = numpy.absolute(X2140_2/(L - nu *(L + numpy.real(sff2_2) * D) + complex(0,1)* nu*
D* numpy.imag(sff2_2)))
ax4 = fig.add_subplot(3,2, 4)
ax4.set_title('Response as function of frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=16)
matplotlib.pyplot.ylabel(r'$\frac{\left|\xi_2 \right|}{A}  $', fontsize=16)
ax4.plot( nu, xiA2, '*')

D = 1
L = 0.5
Nside = 10
Nbott = 5
ax5 = fig.add_subplot(3,2,5)
ax5.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
```

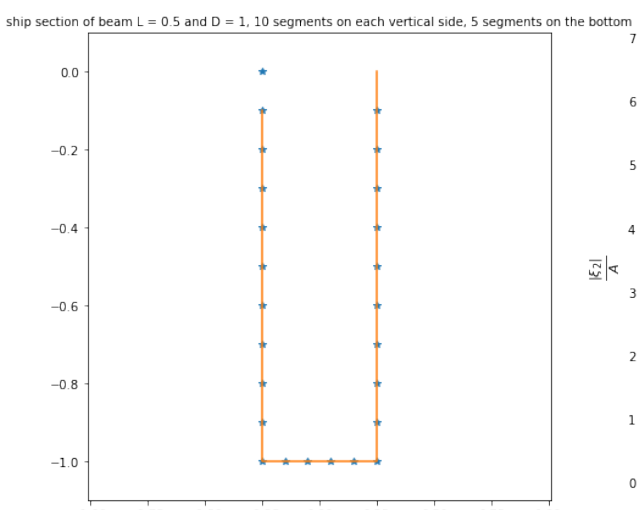
```
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax5.set_title(xlabel, fontsize=10)
ax5.plot(xm3, ym3, '*', xp3, yp3, '-')
xiA3 = numpy.absolute(X2140_3/(L - nu * (L + numpy.real(sff2_3) * D) + complex(0,1)* nu*
D* numpy.imag(sff2_3)))
ax6 = fig.add_subplot(3,2, 6)
ax6.set_title('Response as function of frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=16)
matplotlib.pyplot.ylabel(r'$\frac{\left|\xi_2 \right|}{A}  $', fontsize=16)
ax6.plot( nu, xiA3, '*');
```

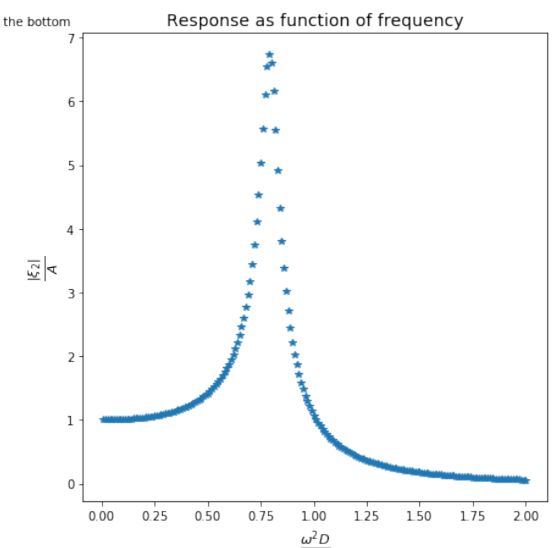

ship section of beam L = 2 and D = 1, 10 segments on each vertical side, 20 segments on the bottom


Response as function of frequency
$\frac{|\xi_2|}{A}$
$\frac{\omega^2 D}{g}$


ship section of beam L = 1 and D = 1, 10 segments on each vertical side, 10 segments on the bottom


Response as function of frequency
$\frac{|\xi_2|}{A}$
$\frac{\omega^2 D}{g}$


ship section of beam L = 0.5 and D = 1, 10 segments on each vertical side, 5 segments on the bottom

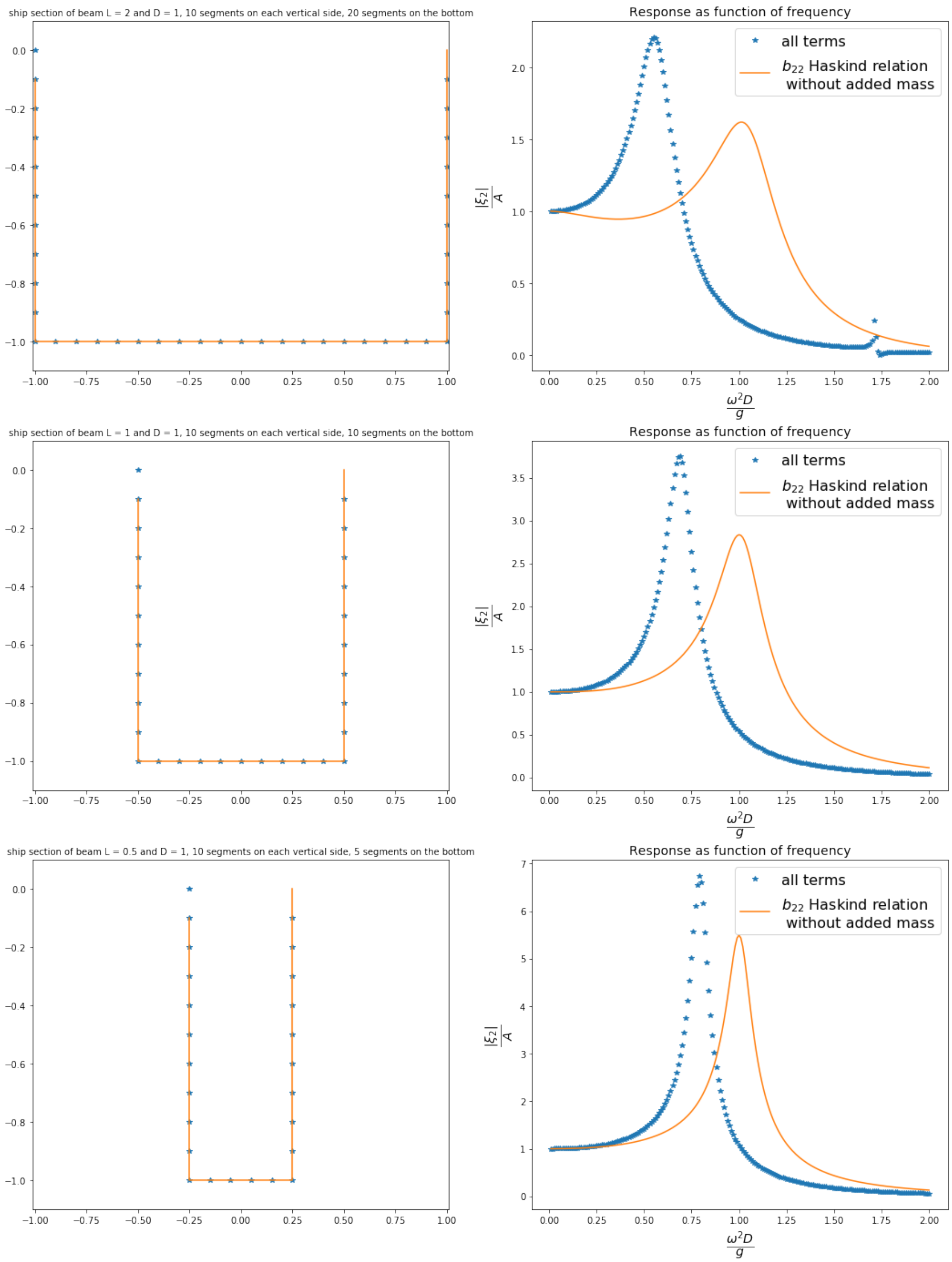

Response as function of frequency
$\frac{|\xi_2|}{A}$
$\frac{\omega^2 D}{g}$

6. Response as function of the frequency (2)

- Include also in the plots the response calculated by the Froude Krylov approximation X_2^{FK}, with b_{22} obtained by the Haskind relation based on X_2^{FK} and where the effect of the added mass a_{22} is neglected.

In [83]:
```
nu = numpy.linspace(0.0, 2.0, num = 201, endpoint=True)[1:]
fig = matplotlib.pyplot.figure(figsize=[18,24])
D = 1
L = 2
Nside = 10
Nbott = 20
ax1 = fig.add_subplot(3,2,1)
ax1.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax1.set_title(xlabel, fontsize=10)
ax1.plot(xm1, ym1, '*', xp1, yp1, '-')
xiA1 = numpy.absolute(X2140_1/(L - nu* (L + numpy.real(sff2_1) * D) + complex(0,1)* nu*
D* numpy.imag(sff2_1)))
xiA1appwtout = numpy.absolute(X130_1/(L - nu * (L ) + complex(0,1)* nu* D*
X130_1**2/D**2))
ax2 = fig.add_subplot(3,2, 2)
ax2.set_title('Response as function of frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
matplotlib.pyplot.ylabel(r'$\frac{\left|\xi_2 \right|}{A}  $', fontsize=20)
allterms1, = ax2.plot( nu, xiA1, '*', label= 'all terms')
addedmassnegl1, = ax2.plot( nu, xiA1appwtout, '-', label= r'$b_{22}$ Haskind relation
'+'\n without added mass')
matplotlib.pyplot.legend(handles= [allterms1, addedmassnegl1 ], fontsize= 16);

D = 1
L = 1
Nside = 10
Nbott = 10
ax3= fig.add_subplot(3,2,3)
ax3.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax3.set_title(xlabel, fontsize=10)
ax3.plot(xm2, ym2, '*', xp2, yp2, '-')
xiA2 = numpy.absolute(X2140_2/(L - nu *(L + numpy.real(sff2_2) * D) + complex(0,1)* nu*
D* numpy.imag(sff2_2)))
xiA2appwtout = numpy.absolute(X130_2/(L - nu * (L ) + complex(0,1)* nu* D* X130_2**2))
ax4 = fig.add_subplot(3,2, 4)
ax4.set_title('Response as function of frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
matplotlib.pyplot.ylabel(r'$\frac{\left|\xi_2 \right|}{A}  $', fontsize=20)
allterms2, = ax4.plot( nu, xiA2, '*', label= 'all terms')
addedmassnegl2, = ax4.plot( nu, xiA2appwtout, '-', label= r'$b_{22}$ Haskind relation
'+'\n without added mass')
matplotlib.pyplot.legend(handles= [allterms2, addedmassnegl2 ], fontsize= 16);
```

```
D = 1
L = 0.5
Nside = 10
Nbott = 5
ax5 = fig.add_subplot(3,2,5)
ax5.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments on each vertical side, '+str(Nbott)+' segments on the bottom'
ax5.set_title(xlabel, fontsize=10)
ax5.plot(xm3, ym3, '*', xp3, yp3, '-')
xiA3 = numpy.absolute(X2140_3/(L - nu * (L + numpy.real(sff2_3) * D) + complex(0,1)* nu*
D* numpy.imag(sff2_3)))
xiA3appwtout = numpy.absolute(X130_3/(L - nu * (L ) + complex(0,1)* nu* D* X130_3**2))
ax6 = fig.add_subplot(3,2, 6)
ax6.set_title('Response as function of frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
matplotlib.pyplot.ylabel(r'$\frac{\left|\xi_2 \right|}{A}  $', fontsize=20)
allterms3, = ax6.plot( nu, xiA3, '*', label= 'all terms')
addedmassnegl3, = ax6.plot( nu, xiA3appwtout, '-', label= r'$b_{22}$ Haskind relation
'+'\n without added mass')
matplotlib.pyplot.legend(handles= [allterms3, addedmassnegl3 ], fontsize= 16);
```

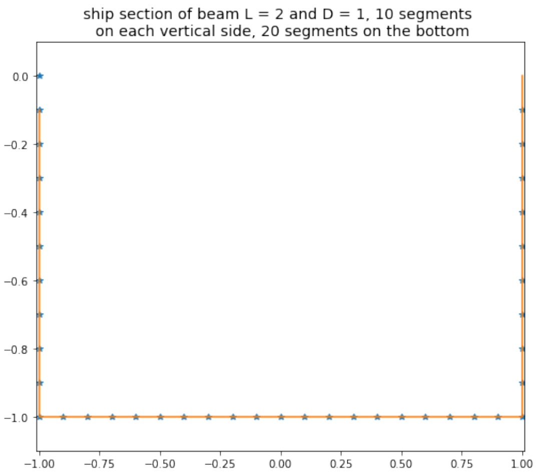


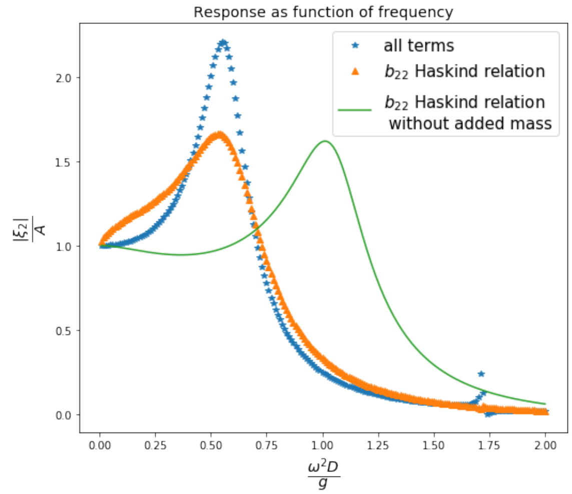
7. Response as function of the frequency (3)

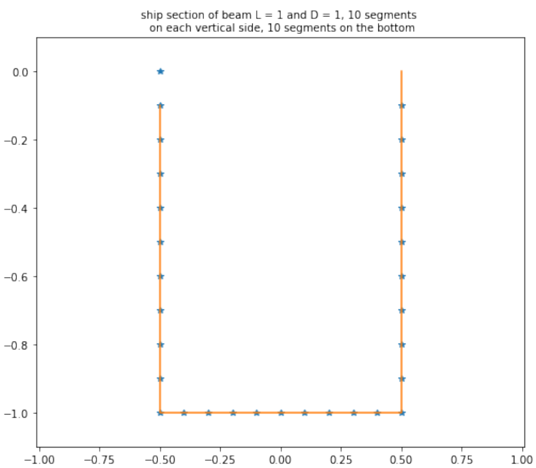
- Use the approximate method with a_{22} included in the calculation.

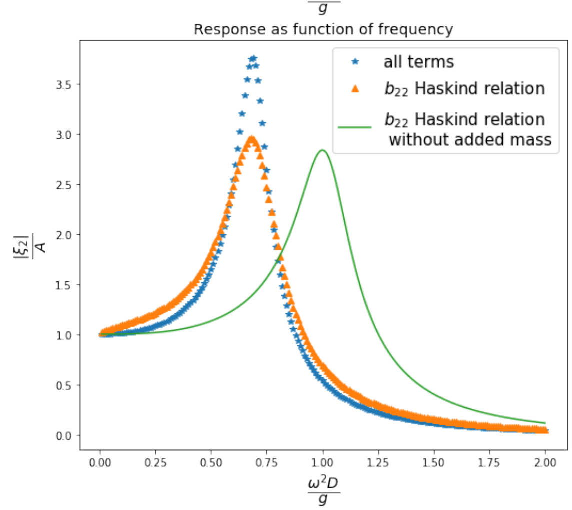
In [73]:
```
nu = numpy.linspace(0.0, 2.0, num = 201, endpoint=True)[1:]
fig = matplotlib.pyplot.figure(figsize=[18,24])
D = 1
L = 2
Nside = 10
Nbott = 20
ax1 = fig.add_subplot(3,2,1)
ax1.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax1.set_title(xlabel, fontsize=14)
ax1.plot(xm1, ym1, '*', xp1, yp1, '-')
xiA1 = numpy.absolute(X2140_1/(L - nu* (L + numpy.real(sff2_1) * D) + complex(0,1)* nu*
D* numpy.imag(sff2_1)))
xiA1appwta22 = numpy.absolute(X130_1/(L - nu * (L + numpy.real(sff2_1) * D) +
complex(0,1)* nu* D* (X130_1**2)/D**2))
xiA1appwtout = numpy.absolute(X130_1/(L - nu * (L ) + complex(0,1)* nu* D*
X130_1**2/D**2))

ax2 = fig.add_subplot(3,2, 2)
ax2.set_title('Response as function of frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
matplotlib.pyplot.ylabel(r'$\frac{\left|\xi_2 \right|}{A}  $', fontsize=20)
allterms1, = ax2.plot( nu, xiA1, '*', label= 'all terms')
approx1, = ax2.plot( nu, xiA1appwta22, '^', label= r'$b_{22}$ Haskind relation')
addedmassnegl1, = ax2.plot( nu, xiA1appwtout, '-', label= r'$b_{22}$ Haskind
relation'+'\n without added mass')
matplotlib.pyplot.legend(handles= [allterms1, approx1, addedmassnegl1 ], fontsize= 15);

D = 1
L = 1
Nside = 10
Nbott = 10
ax3= fig.add_subplot(3,2,3)
ax3.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax3.set_title(xlabel, fontsize=10)
ax3.plot(xm2, ym2, '*', xp2, yp2, '-')
xiA2 = numpy.absolute(X2140_2/(L - nu *(L + numpy.real(sff2_2) * D) + complex(0,1)* nu*
D* numpy.imag(sff2_2)))
xiA2appwta22 = numpy.absolute(X130_2/(L - nu * (L + numpy.real(sff2_2) * D) +
complex(0,1)* nu* D* X130_2**2/D**2))
xiA2appwtout = numpy.absolute(X130_2/(L - nu * (L ) + complex(0,1)* nu* D*
X130_2**2/D**2))
ax4 = fig.add_subplot(3,2, 4)
ax4.set_title('Response as function of frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
matplotlib.pyplot.ylabel(r'$\frac{\left|\xi_2 \right|}{A}  $', fontsize=20)
allterms2, = ax4.plot( nu, xiA2, '*', label= 'all terms')
approx2, = ax4.plot( nu, xiA2appwta22, '^', label= r'$b_{22}$ Haskind relation')
addedmassnegl2, = ax4.plot( nu, xiA2appwtout, '-', label= r'$b_{22}$ Haskind
relation'+'\n without added mass')
matplotlib.pyplot.legend(handles= [allterms2, approx2, addedmassnegl2 ], fontsize= 15);

D = 1
```

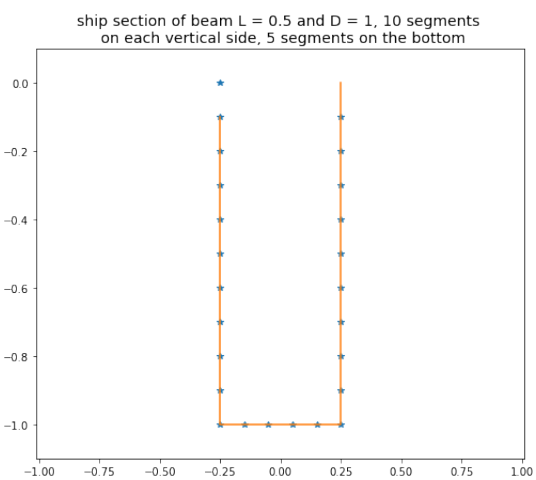
```
L = 0.5
Nside = 10
Nbott = 5
ax5 = fig.add_subplot(3,2,5)
ax5.axis([-1.01, 1.01, -1.1, 0.1])
xlabel = 'ship section of beam L = '+str(L)+' and D = '+str(D)+', '+str(Nside)+'
segments \n on each vertical side, '+str(Nbott)+' segments on the bottom'
ax5.set_title(xlabel, fontsize=14)
ax5.plot(xm3, ym3, '*', xp3, yp3, '-')
xiA3 = numpy.absolute(X2140_3/(L - nu * (L + numpy.real(sff2_3) * D) + complex(0,1)* nu*
D* numpy.imag(sff2_3)))
xiA3appwta22 = numpy.absolute(X130_3/(L - nu * (L + numpy.real(sff2_3) * D) +
complex(0,1)* nu* D* X130_3**2))
xiA3appwtout = numpy.absolute(X130_3/(L - nu * (L ) + complex(0,1)* nu* D* X130_3**2))
ax6 = fig.add_subplot(3,2, 6)
ax6.set_title('Response as function of frequency', fontsize=14)
matplotlib.pyplot.xlabel(r'$\frac{\omega^2 D}{g}$', fontsize=20)
matplotlib.pyplot.ylabel(r'$\frac{\left|\xi_2 \right|}{A}  $', fontsize=20)
allterms3, = ax6.plot( nu, xiA3, '*', label= 'all terms')
approx3, = ax6.plot( nu, xiA3appwta22, '^', label= r'$b_{22}$ Haskind relation')
addedmassnegl3, = ax6.plot( nu, xiA3appwtout, '-', label= r'$b_{22}$ Haskind
relation'+'\n without added mass')
matplotlib.pyplot.legend(handles= [allterms3, approx3, addedmassnegl3 ], fontsize= 15);
```

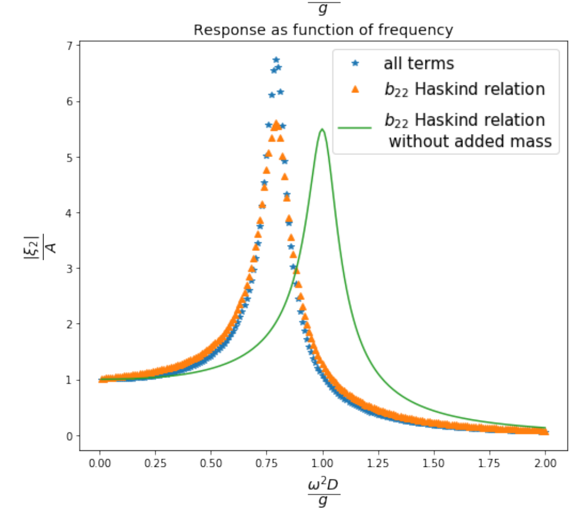

ship section of beam L = 2 and D = 1, 10 segments
on each vertical side, 20 segments on the bottom


Response as function of frequency
all terms
b_{22} Haskind relation
b_{22} Haskind relation without added mass
$\frac{|\xi_2|}{A}$
$\frac{\omega^2 D}{g}$


ship section of beam L = 1 and D = 1, 10 segments
on each vertical side, 10 segments on the bottom


Response as function of frequency
all terms
b_{22} Haskind relation
b_{22} Haskind relation without added mass
$\frac{|\xi_2|}{A}$
$\frac{\omega^2 D}{g}$


ship section of beam L = 0.5 and D = 1, 10 segments
on each vertical side, 5 segments on the bottom


Response as function of frequency
all terms
b_{22} Haskind relation
b_{22} Haskind relation without added mass
$\frac{|\xi_2|}{A}$
$\frac{\omega^2 D}{g}$

G. Conclusion

- Addert masse kan ikke generelt bli forkastet. Det spørs om geometrien. I noen tillfeller var tilnærmingen uten addert masse ikke god nok i forhold til regningen med alle termene (Det skjer i geometrien med $L = 2, D = 1.$). Hvis

$$\frac{a_{22}}{m} \ll 1, \tag{67}$$

addert masse kan bli forkastet.
- Regningen fra Haskind relasjonene og uten addert masse gir høyere resonans frekvensen.

- Geometrien med $L/D = 0.1$ får høyest resonansfrekvens. $L/D = 2$ får minst.
- Geometrien med $L/D = 2$ får høyest dempingskoeficient $\frac{b_{22}}{\rho\omega D^2}$ og addert masse $\frac{a_{22}}{\rho D^2}$.
- For geometriene med $L/D = 0.1$ og $L/D = 1$ var tilnærmingen med Haskind relasjonen og med addert masse bedre enn for geometrien med $L/D = 2$.
- Geometrien med $L/D = 0.1$ får høyest respons $\frac{|\xi_2|}{A}$.

[1] J. N. Newman, *Marine Hydrodynamics* (The MIT Press, 2017).