```
import numpy as np
import matplotlib.tri as mtri
fis = open("Rdaten.txt", "r")
dat1 = fis.readline()
dat2 = fis.readline()
dat3 = fis.readline()
dat4 = fis.readline()
fis.close()
dat1 = dat1[0:len(dat1)-1]
dat2 = dat2[0:len(dat2)-1]
dat3 = dat3[0:len(dat3)-1]
dat4 = dat4[0:len(dat4)-1]
data1 = tuple([float(x) for x in dat1.split(",")])
data2 = tuple([float(x) for x in dat2.split(",")])
data3 = tuple([float(x) for x in dat3.split(",")])
data4 = tuple([float(x) for x in dat4.split(",")])
lm,my,vo = data1
me,re,h1,h2 = data2
pa1,pa2,pa3 = data3
pp,fc1,fc2,fc3 = data4
pa = np.array([pa1,pa2,pa3])
fc = np.array([fc1,fc2,fc3])
de = re/me
me = int(me)
pa = pa/de
def kr(a,b):
    if a == b :
        r = 1
    else :
        r = 0
    return r
def ri(x) :
    if x != 0 :
        r = 1/x
    else :
        r = 0
    return r
def ind(m,k,l,p):
    r = 10*m + (2-k)*(3-k)*(4-k)/6 + (2-k-l)*(3-k-l)/2 + 2-k-l-p
    r = int(round(r))
    return r
class elem :
    def __init__(self, v = None):
        self.v = v
        self.proz()
    def proz(self):
        b = np.vstack((self.v[1]-self.v[0],self.v[2]-self.v[0],self.v[3]-
self.v[0]))
        self.mat = np.linalg.inv(b)
        self.det = np.linalg.det(b)
        ce = np.zeros((10,10))
        for p in range(10) :
            for q in range(10) :
                m,k,l,s = c[p]
                m,xi,et,ze = c[q]
                xi = xi/2
                et = et/2
                ze = ze/2
                ce[p,q] = (xi**k)*(et**l)*(ze**s)
        self.cf = np.linalg.inv(ce)
    def itr(self, x):
        r = np.matmul(self.mat,x-self.v[0])
        return r
    def fc(self, n, x):
        xi = self.itr(x)
        ci = np.zeros(10)
        for p in range(10) :
            m,k,l,s = c[p]
            ci[p] = (xi[0]**k)*(xi[1]**l)*(xi[2]**s)
        r = np.sum(self.cf[n]*ci)
        return r
def ing(a):
    if np.min(a) < 0 :
        r = 0
    elif np.sum(a) == 0 :
        r = 1/6
    elif np.max(a) == 2 :
        r = 1/60
    elif np.sum(a) == 2 and np.max(a) == 1 :
        r = 1/120
    elif np.sum(a) == 1 :
        r = 1/24
    else :
        r = 0
    return r
def nelem():
    global te
    global be
    global se
    x = np.linspace(1, 2*me-1, me)
    y = np.linspace(1, 2*me-1, me)
    z = np.linspace(1, 2*me-1, me)
    x,y,z = np.meshgrid(x,y,z,indexing = "ij")
    x = np.ravel(x)
    y = np.ravel(y)
    z = np.ravel(z)
    n, = x.shape
    te = []
    se = []
    for i in range(n) :
        an = np.array([x[i],y[i],z[i]])
        for j in range(3) :
            d = np.array([kr(0,j),kr(1,j),kr(2,j)])
            for l in range(2) :
                cu = an+(2*l-1)*d
                for k in range(2) :
                    v = np.empty((4,3))
                    v[0] = an
                    v[1] = 1+cu-d
                    v[2] = -1+cu+d
                    q = 0
                    for m in range(3) :
                        if m != j :
                            if q != 0 :
                                v[3,m] = cu[m]+pow(-1,k)
                            else :
                                v[3,m] = cu[m]-pow(-1,k)
                                q = 1
                        else :
                            v[3,m] = cu[m]
                    if max([np.linalg.norm(v[u]-me) for u in range(4)]) <= me :
                        te.append(v)
                        for p in range(4) :
                            h = np.empty((0,3))
                            for q in range(4) :
                                if q != p :
                                    h = np.vstack((h,v[q]))
                            se.append(h)
    be = len(te)
    print("be", be)
    return be, te, se
def lst():
    global c
    global be
    c = [0 for i in range(10*be)]
    for m in range(be):
        for k in range(3) :
            for l in range(3) :
                for p in range(3) :
                    if k + l + p <= 2 :
                        c[ind(m,k,l,p)] = (m,k,l,p)
def lpc(n):
    m,k,l,s = c[int(n)]
    v = te[m]
    xi = 0.5*np.array([k,l,s])
    x = v[0] + np.matmul(np.vstack((v[1]-v[0],v[2]-v[0],v[3]-v[0])),xi)
    return x
def ta(n):
    global te
    global be
    r = []
    for u in range(10*be) :
        m,k,l,s = c[u]
        v = te[m]
        xi = 0.5*np.array([k,l,s])
        x = v[0] + np.matmul(np.vstack((v[1]-v[0],v[2]-v[0],v[3]-v[0])),xi)
        if np.all((np.round_(x,decimals = 1) == np.round_(lpc(n),decimals = 1)))
:
            r.append(m)
    return r
def coeftri(vf):
    f = np.zeros(3)
    x = np.sum(vf, axis = 0)/3
    f[2] = -pp*x[2]/me
    f[1] = pp*(x[0]/np.sqrt(x[1]**2 + x[2]**2) - x[1]/me)
    f[0] = -pp*(x[1]/np.sqrt(x[1]**2 + x[2]**2) + x[0]/me)
    r = np.linalg.norm(np.cross(vf[2]-vf[0],vf[1]-vf[0]))*(de**2)*f
    return r
def lsf():
    global te
    global be
    global psf
    fac = []
    for m in range(be) :
        v = te[m]
        for j in range(4) :
            u = []
            for i in range(4) :
                if i != j :
                    u.append(v[i])
            a = 0
            ac = 0
            for n in range(be) :
                for l in range(3) :
                    for s in range(4) :
                        if np.all((np.round_(u[l], decimals = 1) ==
np.round_(te[n][s], decimals = 1 ))) :
                            ac = ac + 1
                if ac == 3 :
                    a = a +1
                ac = 0
            if a == 1 :
                fac.extend(u)
                fac.extend([0.5*(u[0]+u[1]),0.5*(u[0]+u[2]),0.5*(u[1]+u[2])])
    psf = []
    for vr in fac :
        for nu in range(10*be) :
            if np.any(np.round_(lpc(nu), decimals = 1) != np.round_(vr, decimals
= 1)):
                psf.append(nu)
                st = 1
    psf = list(set(psf))
def lag():
    global ka
    global psf
    global be
    ka = np.zeros((30*be,30*be))
    lk = []
    for n in range(10*be) :
        if len(ta(n)) >= 2 :
            nm = []
            for i in ta(n) :
                for p in range(10) :
                    u,k,l,s = c[p]
                    if np.all((np.round_(lpc(ind(i,k,l,s)), decimals = 1) ==
np.round_(lpc(n), decimals = 1))) :
                        nm.append(ind(i,k,l,s))
            if nm[0] not in lk :
                lk.append(nm[0])
                u = 3*(len(nm)-1)
                a,b = ka.shape
                ka = np.hstack((ka,np.zeros((a,u))))
                ka = np.vstack((ka,np.zeros((u,u+a))))
                for i in range(len(nm)-1) :
                    for j in range(3) :
                        ka[3*nm[0]+j,a+3*i +j] = 1
                        ka[3*nm[i+1]+j,a +3*i +j] = -1
                        ka[a+3*i+j, 3*nm[0]+j] = 1
                        ka[a+3*i+j, 3*nm[i+1]+j] = -1
    for n in psf :
        if lpc(n)[2] <= me :
            a,b = ka.shape
            ka = np.hstack((ka,np.zeros((a,3))))
            ka = np.vstack((ka,np.zeros((3,a+3))))
            for j in range(3) :
                ka[3*n+j,a+j] = 1
                ka[a+j,3*n+j] = 1
def col():
    global psf
    global te
    global be
    a,b = ka.shape
    li = [np.linalg.norm(pa - lpc(n)) for n in psf]
    ni = [n for n in psf if np.linalg.norm(pa - lpc(n)) == min(li)]
    br = np.zeros(a)
    for j in range(3) :
        br[3*ni[0]+j] = fc[j]
    rps = [n for n in psf if lpc(n)[2] >= h1/de and lpc(n)[2] <= h2/de]
    xs = np.array([lpc(n)[0] for n in rps])
    ys = np.array([lpc(n)[1] for n in rps])
    zs = np.array([lpc(n)[2] for n in rps])
    tng = mtri.Triangulation(xs,ys)
    trig = tng.triangles
    a,b = trig.shape
    for k in range(a) :
        vf = np.zeros((3,3))
        for i in range(3) :
            vf[i] = lpc(rps[trig[k,i]])
        for i in range(3) :
            n = rps[trig[k,i]]
            for j in range(3) :
                br[3*n+j] = br[3*n+j] + coeftri(vf)[j]/3
    for n in range(10*be) :
        m,k,l,s = c[n]
        ni = ind(0,k,l,s)
        v = te[m]
        el = elem(v)
        ba = 0
        for p in range(10) :
            m,k,l,s = c[p]
            a = np.array([k,l,s])
            ba = ba + el.cf[ni,ind(0,k,l,s)]*el.det*ing(a)
        br[3*n +2] = -ba*vo*(de**3)
    return xs, ys, zs, br
def cemat(m,p,q,h,k):
    global te
    el = elem(te[m])
    ca = 0
    for i in range(10) :
        n,g,l,s = c[i]
        f = np.array([g,l,s])
        for j in range(10) :
            n,g,l,s = c[j]
            w = np.array([g,l,s])
            for a in range(3) :
                for b in range(3) :
                    kal = np.array([kr(0,a),kr(1,a),kr(2,a)])
                    kbl = np.array([kr(0,b),kr(1,b),kr(2,b)])
                    ca = ca +
el.cf[p,i]*el.cf[q,j]*f[a]*w[b]*el.mat[a,h]*el.mat[b,k]*ing(f+w-kal-kbl)
    ca = ca*el.det
    return ca
def kat():
    global ka
    global be
    for m in range(be) :
        for p in range(10) :
            for q in range(10) :
                su = 0
                for j in range(3) :
                    su = su + cemat(m,p,q,j,j)
                for h in range(3) :
                    for k in range(3) :
                        ka[3*(10*m+p)+h,3*(10*m+q)+k] = (2*lm*cemat(m,p,q,h,k) +
4*kr(h,k)*my*su)*(de**2)
    return ka
def che(x):
    global te
    global be
    m = 0
    b = 0
    while m < be and b == 0 :
        v = te[m]
        j = 0
        a = 0
        while j < 4 and a == 0 :
            u = []
            for i in range(4) :
                if i != j :
                    u.append(v[j])
            d1 = np.vstack((u[1]-u[0],u[2]-u[0],v[j]-u[0]))
            d2 = np.vstack((u[1]-u[0],u[2]-u[0],x-u[0]))
            d1 = np.linalg.det(d1)
            d2 = np.linalg.det(d2)
            if np.sign(d1)*np.sign(d2) < 0 :
                a = 1
            else :
                j = j+1
        if j == 4 :
            b = 1
        else :
            m = m+1
    r = m
    return r
```