Second laboratory work

Kondratev Egor 206

March 14, 2021

Task 3 variant 1

Create some constants which will be useful in future for plot 3d graphics.

```
xmin, xmax = -10, 10
ymin, ymax = -10, 10
zmin, zmax = -10, 10
```

Say for sage that x, y, z is variables, and create function f.

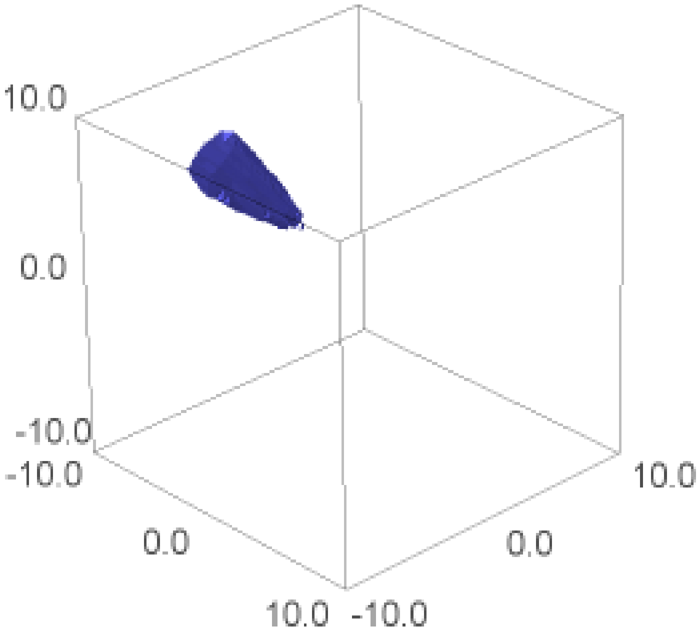
```
var("x y z")
f(x, y, z) = 7*x^2 + 3*y^2 + 3*z^2 +  8*x*y + 8*x*z + 6*y*z + 6*x + y + 7
```

Display the function formula.

$7\,x^2 + 8\,xy + 3\,y^2 + 8\,xz + 6\,yz + 3\,z^2 + 6\,x + y + 7$

Plot graphic of unchanged figure.

```
p = implicit_plot3d(f(x=x, y=y, z=z), (x, xmin, xmax), (y, ymin, ymax), (z, zmin, zmax))
```



Function that, according to a given algorithm, converts the function passed as an argument to the canonical form. Return tuple (lambdas, a, a0).

```
def kanonic_coeffs(fun):
    try:
        var("l l1 l2 l3")
        lvcts = []
        svcts = []
        tmp_fun = fun
        a = vector(RR, 3)
        tmp_vct = vector(RR, 9)
        var_combs_tmp = (x^2, y^2, z^2, x*y, x*z, y*z)
        var_combs_0 = (x^2, x*y, x*z, x*y, y^2, y*z, x*z, y*z, z^2)
        var_combs_1 = (x, y, z)
        for i, var_comb in enumerate(var_combs_0):
            if i == 0 or i == 4 or i == 8:
                tmp_vct[i] = fun.coefficient(var_comb)
            else:
```

```
                tmp_vct[i] = fun.coefficient(var_comb) / 2
        for var_comb in var_combs_tmp:
            tmp_fun -= fun.coefficient(var_comb)*var_comb
        for i, var_comb in enumerate(var_combs_1):
            a[i] = tmp_fun.coefficient(var_comb) / 2
            tmp_fun -= tmp_fun.coefficient(var_comb)*var_comb
        a0 = tmp_fun.n()
        A = matrix(SR, 3, tmp_vct)
        L = matrix(SR, 3, 3, var('l'))
        E = matrix(SR, 3, 3, 1)
        lvct = vector([l1, l2, l3])
        lambdas = solve([(A-L).determinant() == 0], l)
        for i, el in enumerate(lambdas):
            nums = []
            lhs = (A-el.rhs()*E)*lvct
            res = solve([lhs[0] == 0, lhs[1] == 0, lhs[2] == 0], l1, l2, l3)[0]
            for i in range(len(res)):
                if len(res[i].rhs().variables()) == 0:
                    nums.append(res[i].rhs())
                else:
                    nums.append(res[i].rhs()(1))
            lvcts.append(vector(nums))
        for el in lvcts:
            norm_lvct = (el / sqrt((el*el).n())).n()
            svcts.append(norm_lvct)
        ST = matrix(RR, 3)
        for i in range(len(svcts)):
            ST[i] = svcts[i]
        a_ = ST*a
        for i in range(len(lambdas)):
            lambdas[i] = lambdas[i].rhs().n()
        return (lambdas, a_, a0)
    except:
        print("Something gone wrong\n")
        return (None, None, None)
```

Obtain from the function a tuple of the required coefficients.

```
(lambdas, a, a0) = kanonic_coeffs(f)
```

Display the obtained coefficients.

```
print("Lambdas:", tuple(lambdas))
print("a:", a)
print("a0:", a0)
```

Lambdas: $(0.821091654199726, 12.1789083458003, 0.000000000000000)$
a: $(-1.76500958148313, 2.45147734586162, -0.353553390593274)$
a0: 7.00000000000000

Create a function with obtained coefficients.

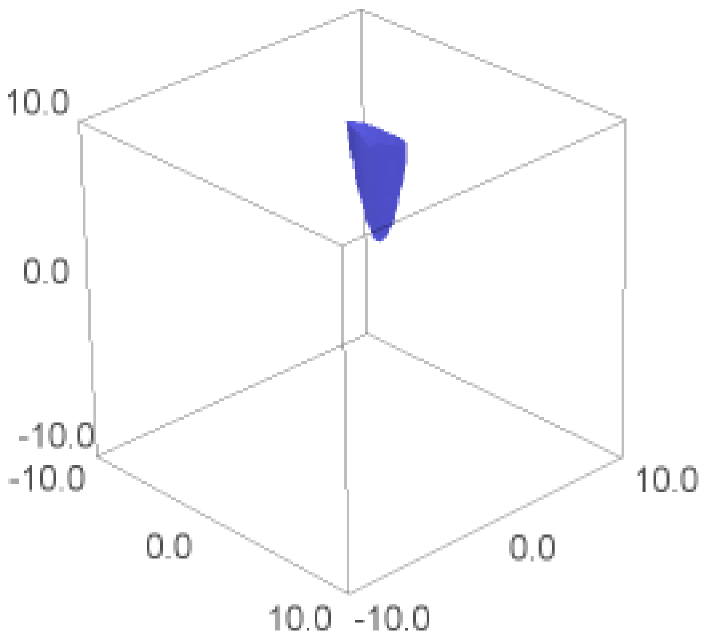
```
var("kx ky kz")
first_part = lambdas[0]*kx^2 + lambdas[1]*ky^2 + lambdas[2]*kz^2
second_part = 2*a[0]*kx + 2*a[1]*ky + 2*a[2]*kz + a0
kanonic_func(kx, ky, kz) = first_part + second_part
```

Display the obtained function formula.

$0.821091654199726\,kx^2+12.1789083458003\,ky^2-3.53001916296626\,kx+4.90295469172324\,ky-0.707106781186547\,kz+7.00000000000000$

Plot graphic of changed figure.

```
p = implicit_plot3d(kanonic_func(kx=kx, ky=ky, kz=kz), (kx, xmin, xmax),\
(ky, ymin, ymax), (kz, zmin, zmax))
```



Initial function.

$$7*x^2+3*y^2+3*z^2+8*x*y+8*x*z+6*y*z+6*x+y+7$$

Canonical function.

$$0.821091654199726*kx^2+12.1789083458003*ky^2-0.353553390593274*kz$$