In [1]:
```
from sympy import *
from sympy.plotting import (plot, plot_parametric)
```

EXPANDING POLYNOMIAL EQUATIONS IN PYTHON

With linear factors, it can be easy to find the unknown coefficients in Partial Fraction Decomposition (if x-a is a linear factor of the denominator, let x=a after clearing the fractions). With irreducible quadratics, finding the unknown coefficients isn't so easy. The easiest strategy is to multiply everything out, then "match powers" (clearly, if two polynomials are supposed to be the same, the matching powers of x should have the same coefficients: x^2 - 3x + 5 and x^2 - 3x + (2+3) are the same, but x^2 - 3x + 5 and x^2 - 3x + (2+2) are not).

In Python, we incorporate the same strategy (after clearing the fractions by hand).

For example, if we want to integrate f(x)=x^3/(x^4+x^2+1), it can be shown that we have to do the following:

Example: Find A, B, C, and D if (Ax+B)(x^2+x+1) + (Cx+D)(x^2-x+1) = x^3

In [2]:
```
x=symbols('x')
A,B,C,D=symbols('A B C D')
LHS=(A*x+B)*(x**2+x+1)+(C*x+D)*(x**2-x+1)
RHS=x**3
# Step 1: multiply everything out on the left using the expand command
LHSexp=expand(LHS)
print(LHSexp)
```

```
A*x**3 + A*x**2 + A*x + B*x**2 + B*x + B + C*x**3 - C*x**2 + C*x + D*x**2 - D*x + D
```

Obviously, this is a mess. It would be nice if Python would organize the powers for us so we can more easily match them. Enter the collect command

In [3]:
```
collect(LHSexp,x)
```

Out[3]:
```
B + D + x**3*(A + C) + x**2*(A + B - C + D) + x*(A + B + C - D)
```

Since this polynomial has to be equal to the RHS (x^3) for all values of x, the coefficients of x^3, x^2, and x and the constants much "match up" (i.e., be equal). This gives us 4 equations to solve for the 4 variables:

In [5]:
```
eq1=A+C-1 # coefficients of x^3 must total 1-moved all to one side
eq2=A+B-C+D #coefficients of x^2 must total 0
eq3=A+B+C-D #coefficients of x must total 0
eq4=B+D #constants must total 0

# Now solve for the 4 variables. Can use matrices or the solve command (as done here)
coeffs=solve([eq1,eq2,eq3,eq4],[A,B,C,D])
print('Solution is',coeffs)
```

```
Solution is {B: -1/2, C: 1/2, D: 1/2, A: 1/2}
```

NOTICE that the solution is of type "dictionary". This means the expressions could be automatically substituted into our partial fraction expansion form (compare the variable "LHS" to what you get when you clear the fractions in the variable "pfracexpand" below)

In [6]:
```
pfracexpand=(A*x+B)/(x**2-x+1)+(C*x+D)/(x**2+x+1)
fintegrate=pfracexpand.subs(coeffs)
print('The partial fraction expansion is',fintegrate)
```

```
The partial fraction expansion is (x/2 - 1/2)/(x**2 - x + 1) + (x/2 + 1/2)/(x**2 + x + 1)
```

In [7]:
```
#Check answer using the built-in function apart for f(x)=x^3/((x^2-x+1)(x^2+x+1)
f=x**3/((x**2-x+1)*(x**2+x+1))
apart(f,x)
```

Out[7]:
```
(x - 1)/(2*(x**2 - x + 1)) + (x + 1)/(2*(x**2 + x + 1))
```

The only difference is that all of the "1/2"s have been moved into the denominator.

In [ ]: