In [47]:

```
# #! /usr/bin/python2
# from itertools import chain
# import numpy as np
# from copy import copy
# from numpy.linalg import det

# def get_hermit(dimensions=5, max_minor=2):
#     diag = [1]*(dimensions)
#     diag[-1] = max_minor
#     pre_last = np.random.randint(max_minor, size=dimensions - 1)
#     hermit = np.append(np.diag(diag), np.random.randint(-max_minor, max_m
#     hermit[dimensions-1][:-1] = pre_last
#     return hermit


# def minor_mat(arr, i=0):
# #     ith row, jth column removed
#     return arr[np.array(range(i)+range(i+1, arr.shape[0]))[:, np.newaxis],
#                 np.array(range(arr.shape[1]))]


# # def minor_mat(arr, i=0, j=0):
#     # ith row, jth column removed
# #     return arr[np.array(range(i)+range(i+1, arr.shape[0]))[:, np.newaxis
# #                 np.array(range(j)+range(j+1, arr.shape[1]))]

# def exclude_row(arr, i=0):
#     # ith row, jth column removed
#     return arr[np.array(chain(range(i),range(i+1, arr.shape[0])))[:, np.ne
#                 np.array(range(arr.shape[1]))]


# def get_det_of_minor(hermite):
#     # return [abs(d) for d in hermite: ]
#     for minor_counter in range(hermite.shape[1]):
#         yield abs(det(minor_mat(hermite, i=0, j=minor_counter)))
#         # prime = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,

# h = get_hermit()
# # print(h)
# # print('\n')
# # print(h[chain(range(1,2), range(3,5))])
# # print(exclude_row(h, 0))

# for i in range(h.shape[0]):
#     print(det(minor_mat(h,i)))
# #     m = copy(h[:-1])
# #     print(m)
# #     m[i] = h[-1]
# #     print(m)
#     #h[list(range(i)).extend(list(range(i+2, h.shape[0])))]
# #             np.array(range(h.shape[1]),dtype='int')])
# #     print(m)
# # det(h)


# A = get_hermit()
# print(A)
# print('\n')
# # print(is_valid(A))
# # if is_valid(A):
# for b in generate_b():
#     print(b)
# # print(np.linalg.solve(A, b))
```

In [18]:

```
import numpy as np
from numpy.linalg import det
from sympy import Matrix, symbols, EmptySet
from sympy.solvers.solveset import linsolve


def get_hermit(dimensions=6, max_minor=5):
    for i in xrange(100):
        diag = [1]*(dimensions)
        diag[-1] = max_minor
        pre_last = np.random.randint(max_minor, size=dimensions - 1)
        hermit = np.append(np.diag(diag), np.random.randint(-max_minor, max_
        hermit[dimensions-1][:-1] = pre_last
        if is_valid(hermit):
            return hermit
        continue


def minor(arr, i=0):
#     ith row, jth column removed
    return arr[np.array(range(i)+range(i+1, arr.shape[0]))[:, np.newaxis],
               np.array(range(arr.shape[1]))]


def is_valid(A):
    for i in xrange(A.shape[0]):
        return abs(det(minor(A,i))) != 0


# prime = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 6


def generate_b(dimensions=6, max_minor=5):
    """finish it"""
    b = np.zeros(shape=(dimensions+1,1))
    for i in xrange(-max_minor,max_minor+1):
        b[-2] = i
        for j in xrange(-10,10+1):
            b[-1] = j
            yield b


def is_valid_solution(solution):
    return not isinstance(s, EmptySet) and all(x for x in solution)
```

In [19]:

```
A = get_hermit()
for b in generate_b():
    m = Matrix(np.concatenate((A,b),axis=1))
    s = linsolve(m, symbols('x1 x2 x3 x4 x5 x6'))
    if is_valid_solution(s):
        print(A)
        print(b)
        print(s)
        print('                        \n')
```

```
[[ 1  0  0  0  0  0]
 [ 0  1  0  0  0  0]
 [ 0  0  1  0  0  0]
 [ 0  0  0  1  0  0]
 [ 0  0  0  0  1  0]
 [ 4  2  1  4  0  5]
 [ 2 -5  0  3  3 -5]]
[[ 0.]
 [ 0.]
 [ 0.]
 [ 0.]
 [ 0.]
 [-5.]
 [ 5.]]
{(0, 0, 0, 0, 0, -1.0)}
_____________________

[[ 1  0  0  0  0  0]
 [ 0  1  0  0  0  0]
 [ 0  0  1  0  0  0]
 [ 0  0  0  1  0  0]
 [ 0  0  0  0  1  0]
 [ 4  2  1  4  0  5]
 [ 2 -5  0  3  3 -5]]
[[ 0.]
 [ 0.]
 [ 0.]
 [ 0.]
 [ 0.]
 [-4.]
 [ 4.]]
{(0, 0, 0, 0, 0, -0.8)}
_____________________

[[ 1  0  0  0  0  0]
 [ 0  1  0  0  0  0]
 [ 0  0  1  0  0  0]
 [ 0  0  0  1  0  0]
 [ 0  0  0  0  1  0]
 [ 4  2  1  4  0  5]
 [ 2 -5  0  3  3 -5]]
[[ 0.]
 [ 0.]
 [ 0.]
 [ 0.]
 [ 0.]
 [-3.]
 [ 3.]]
{(0, 0, 0, 0, 0, -0.6)}
_____________________

[[ 1  0  0  0  0  0]
 [ 0  1  0  0  0  0]
 [ 0  0  1  0  0  0]
 [ 0  0  0  1  0  0]
 [ 0  0  0  0  1  0]
 [ 4  2  1  4  0  5]
 [ 2 -5  0  3  3 -5]]
[[ 0.]
```

```
In [ ]:
```