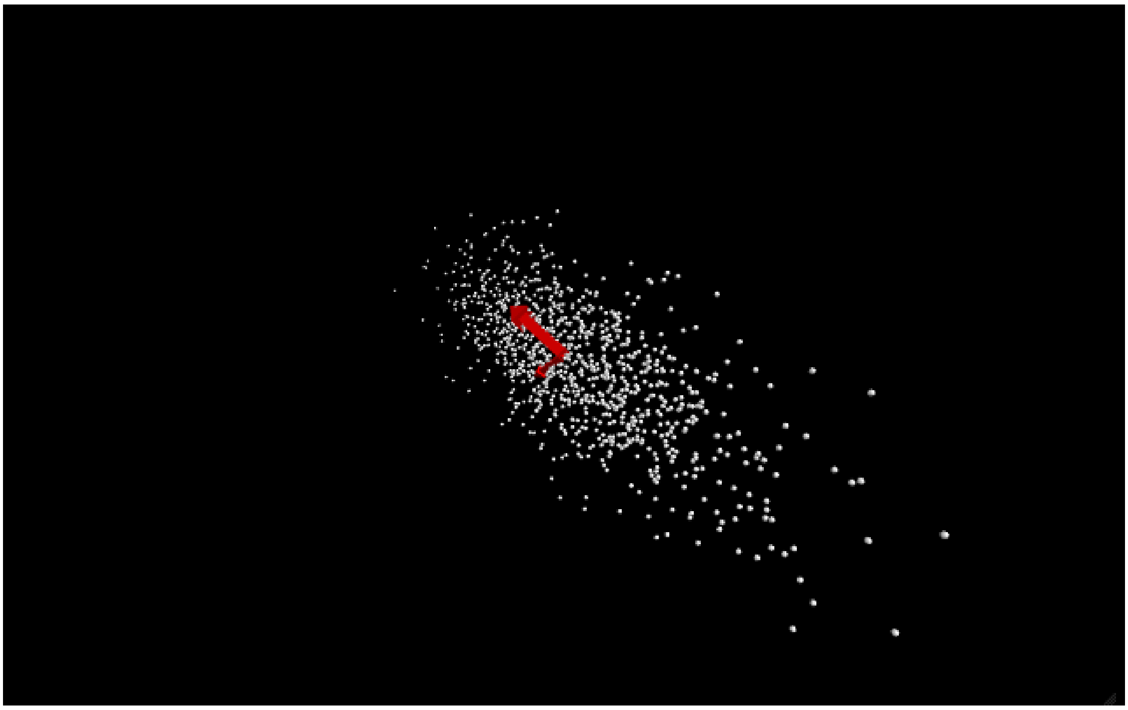
Chapter 3. Basis and Dimension

3.1. Finite dimensional linear space

Program: lincombi.py

In [1]:
```
from vpython import vec, arrow, color, points
from numpy.random import normal

x = vec(*normal(0, 1, 3))
arrow(pos=vec(0, 0, 0), axis=x, color=color.red)
y = vec(*normal(0, 1, 3))
arrow(pos=vec(0, 0, 0), axis=y, color=color.red)
W = [a * x + b * y for (a, b) in normal(0, 1, (1000, 2))]
points(pos=W, radius=2)
```



Program: eqn1.py / eqn1.ipynb

In [1]:
```
from sympy import solve
from sympy.abc import a, b, x, y

ans = solve([a + 2*b - x, 2*a + 3*b - y], [a, b])
print(ans)
```

```
{a: -3*x + 2*y, b: 2*x - y}
```

In [2]:
```
ans[a]
```

Out[2]: $-3x + 2y$

In [3]:
```
ans[b]
```

Out[3]: $2x - y$

Program: eqn2.py / eqn2.ipynb

In [1]:
```
from sympy import solve
from sympy.abc import a, b, x, y, z

ans = solve([a + 2*b - x, 2*a + 3*b - y, 3*a + 4*b - z], [a, b])
print(ans)
```

```
[]
```

In [2]:
```
ans = solve([a + 2*b - x, 2*a + 3*b - y, 3*a + 4*b - z], [a, b, x])
```

In [3]:
```
ans
```

Out[3]:
```
{a: -4*y + 3*z, b: 3*y - 2*z, x: 2*y - z}
```

Program: eqn3.py / eqn3.ipynb

In [1]:
```
from sympy import solve
from sympy.abc import a, b, c, x, y, z

ans = solve([a + 2*b + 3*c - x, 2*a + 3*b + c - y,
             3*a + 4*b + 2*c - z], [a, b, c])

print(ans)
```

```
{a: -2*x/3 - 8*y/3 + 7*z/3, b: x/3 + 7*y/3 - 5*z/3, c: x/3 - 2*y/3 + z/3}
```

In [2]:
```
N = [ans[k].subs([[x, 2], [y, 3], [z, 5]]) for k in [a, b, c]]; N
```

Out[2]:
```
[7/3, -2/3, 1/3]
```

In [3]:
```
[n.evalf(2) for n in N]
```

Out[3]:
```
[2.3, -0.67, 0.33]
```

3.2. Linear dependence and linear independence

Program: eqn4.py / eqn4.ipynb

In [1]:
```
from sympy import solve
from sympy.abc import x, y

ans = solve([x + 2*y, 2*x + 3*y], [x, y])
print(ans)
```

```
{x: 0, y: 0}
```

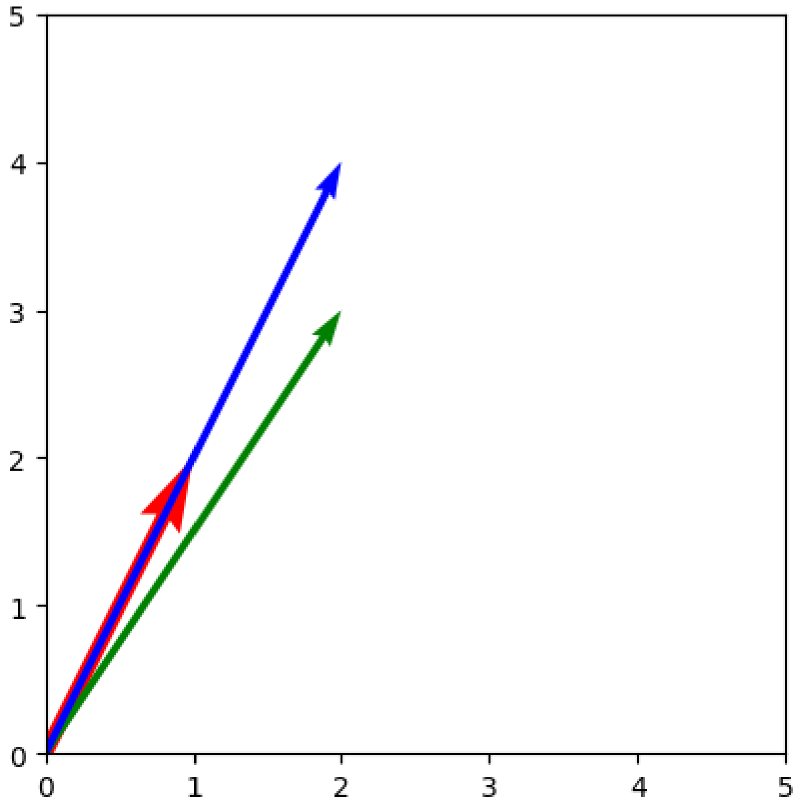
Program: eqn5.py / eqn5.ipynb

In [1]:
```
from sympy import solve
from sympy.abc import x, y

ans = solve([x + 2*y, 2*x + 4*y], [x, y])
print(ans)
```

```
{x: -2*y}
```

Program: arrow2d.py / arrow2d.ipynb

In [1]:
```
import matplotlib.pyplot as plt

o, a, b, c = (0, 0), (1, 2), (2, 3), (2, 4)
arrows = [[o, a, 'r', 0.1], [o, b, 'g', 0.05], [o, c, 'b', 0.05]]
plt.axis('scaled'), plt.xlim(0, 5), plt.ylim(0, 5)
for p, v, c, w in arrows:
    plt.quiver(p[0], p[1], v[0], v[1],
               units='xy', scale=1, color=c, width=w)
```



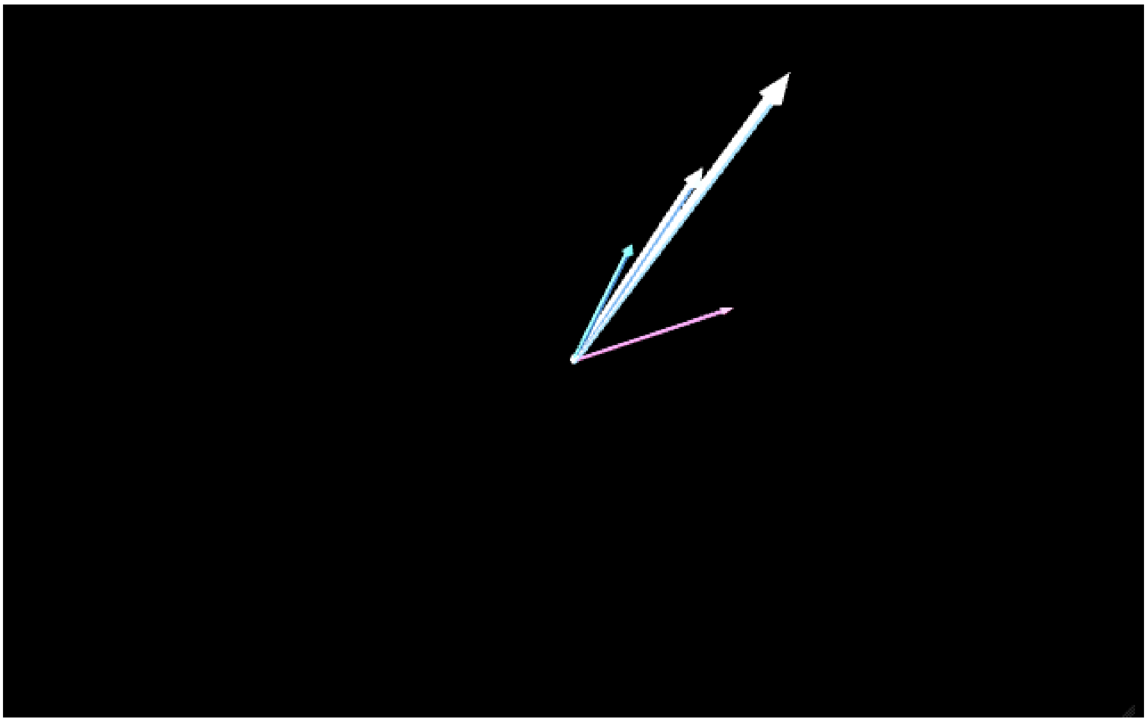
Program: eqn6.py / eqn6.ipynb

In [1]:
```
from sympy import solve
from sympy.abc import x, y, z

ans1 = solve([x + 2*y + 3*z, 2*x + 3*y + 4*z, 3*x + 4*y + 5*z],
             [x, y, z])
print(ans1)

ans2 = solve([x + 2*y + 3*z, 2*x + 3*y + z, 3*x + y + 2*z],
             [x, y, z])
print(ans2)
```

```
{x: z, y: -2*z}
{x: 0, y: 0, z: 0}
```

Program: arrow3d.py / arrow3d.ipynb

In [1]:
```
from vpython import vec, arrow, mag

o = vec(0, 0, 0)
for p in [(1, 2, 3), (2, 3, 4), (3, 4, 5), (3, 1, 2)]:
    v = vec(*p)
    arrow(pos=o, axis=v, color=v, shaftwidth=mag(v) * 0.02)
```

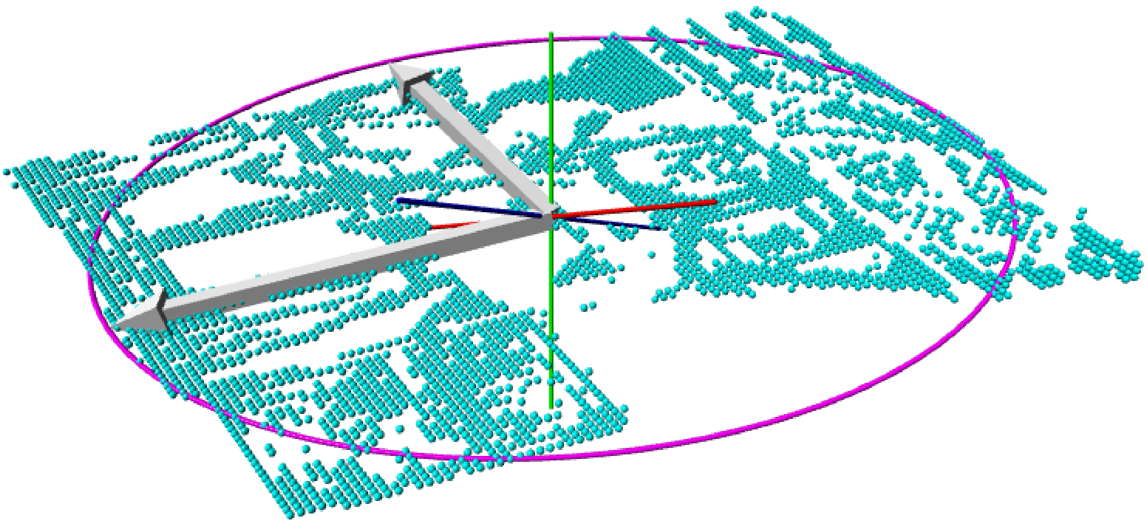


3.3. Basis and representation

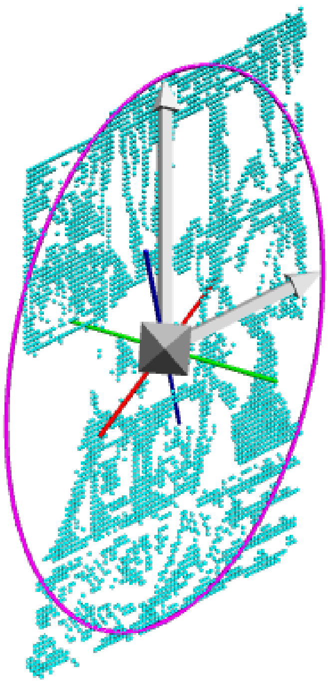
Program: mypict4.py

In [1]:
```
from vpython import canvas, vec, curve, arrow, color, points
from numpy import array, linspace, sin, cos, pi, random

canvas(background=color.white, foreground=color.black)
for v in [vec(1, 0, 0), vec(0, 1, 0), vec(0, 0, 1)]:
    curve(pos=[-v, v], color=v)

with open('mypict1.txt', 'r') as fd:
    XY = eval(fd.read())

random.seed(123)
a = vec(*random.normal(0, 1, 3))
arrow(pos=vec(0, 0, 0), axis=a, shaftwidth=0.1)
b = vec(*random.normal(0, 1, 3))
arrow(pos=vec(0, 0, 0), axis=b, shaftwidth=0.1)
P = [x * a + y * b for (x, y) in XY]
Q = [cos(t) * a + sin(t) * b for t in linspace(0, 2 * pi, 101)]
```

```
points(pos=P, radius=2, color=color.cyan)
curve(pos=Q, color=color.magenta)
```



Below is a view from a direction orthogonal to the projection plane.



mypict40.py

3.4. Dimension and Rank

Program: rank.py

In [1]:
```
from numpy.linalg import matrix_rank

def f(*x): return matrix_rank(x)

a, b, c = (1, 2), (2, 3), (2, 4)
print(f(a, b), f(b, c), f(a, c), f(a, b, c))
a, b, c, d = (1, 2, 3), (2, 3, 4), (3, 4, 5), (3, 4, 4)
print(f(a, b), f(a, b, c), f(a, b, d), f(a, b, c, d))
```

```
2 2 1 2
2 2 3 3
```

3.5. Direct sum

Untitled.ipynb

In [1]:
```
[1, 2] + [3, 4, 5]
```

Out[1]:
```
[1, 2, 3, 4, 5]
```

In [2]:
```
from numpy import array, concatenate
concatenate([array([1, 2]), array([3, 4, 5])])
```

Out[2]:
```
array([1, 2, 3, 4, 5])
```

In [3]:
```
from sympy import Matrix
Matrix([1, 2]).col_join(Matrix([3, 4, 5]))
```

Out[3]:
$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

In [4]:
```
Matrix([[1, 2]]).row_join(Matrix([[3, 4, 5]]))
```

Out[4]:
$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

3.6. Remark on dimension

Program: dim.py

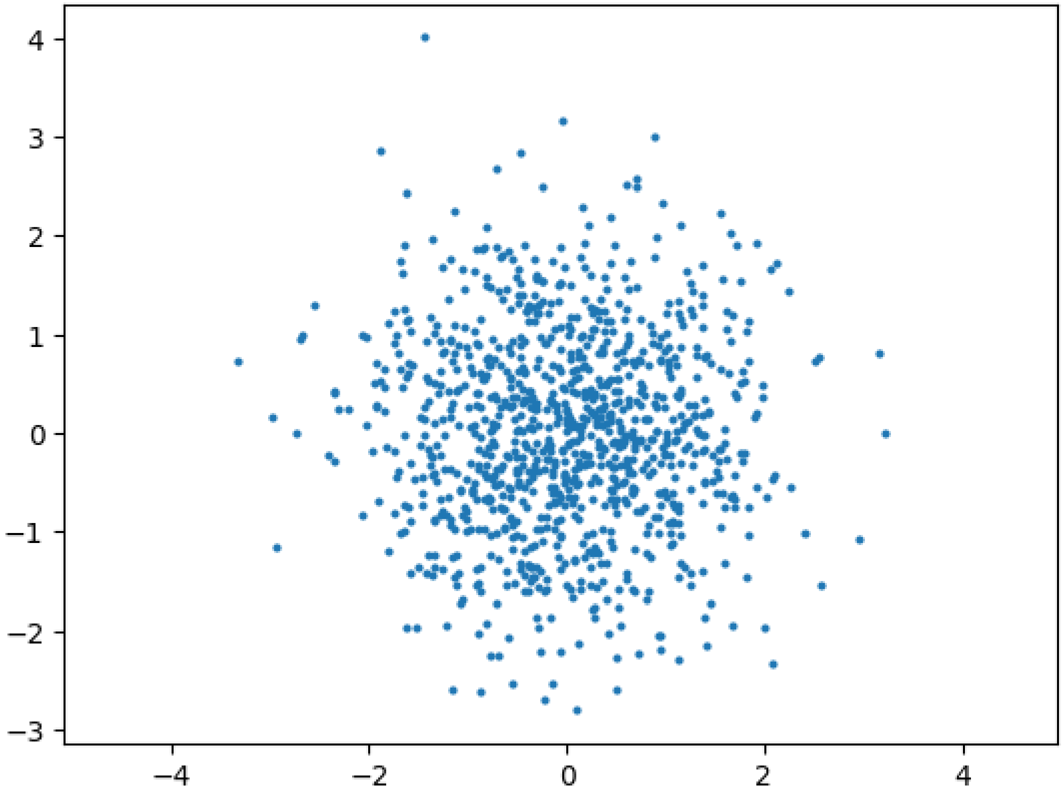
In [1]:
```
from numpy import array

A = array([1, 2, 3])
B = array([[1, 2, 3], [4, 5, 6]])
C = array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print(f'A={A}')
print(f'B={B}')
print(f'C={C}')
```

```
A=[1 2 3]
B=[[1 2 3]
 [4 5 6]]
C=[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

Program: random2d.py

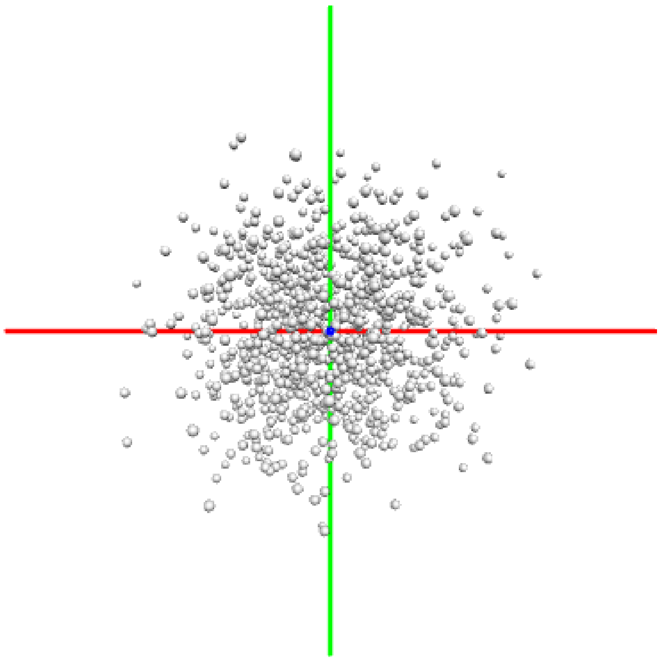
In [1]:
```
import matplotlib.pyplot as plt
from numpy.random import normal

P = normal(0, 1, (1000, 2))
plt.scatter(P[:, 0], P[:, 1], s=4)
plt.axis('equal')
```

Out[1]:
```
(-3.660189033869834,
 3.5410746858760254,
 -3.1456133818414185,
 4.345107626666322)
```



Program: random3d.py

In [1]:
```
from vpython import canvas, color, vec, curve, points
from numpy.random import normal

canvas(background=color.white, foreground=color.black)
for v in [vec(5, 0, 0), vec(0, 5, 0), vec(0, 0, 5)]:
    curve(pos=[-v, v], color=v)
P = normal(0, 1, (1000, 3))
points(pos=[vec(*p) for p in P], radius=4)
```



```
In [ ]:
```