In [1]:
```
import numpy as np
```

In [17]:
```
def function1(X):
    v = np.zeros(X.shape)
    for j in range(X.shape[0]):
        y = np.zeros(X.shape[1])
        for i in range(j):
            r = np.inner(X[j], v[i])
            y += r * v[i]
        v[j] = (X[j] - y)/np.linalg.norm(X[j] - y)
    return v
```

In [18]:
```
def function2(X):
    v = np.zeros(X.shape)
    for j in range(X.shape[0]):
        w = X[j]
        for i in range(j):
            r = np.inner(w, v[i])
            w = w - r * v[i]
        v[j] = (w)/np.linalg.norm(w)
    return v
```

In [19]:
```
def GenerateHilbert(n):
    H = np.zeros((n,n))
    for i in range(1, n+1):
        h = [1/(i + j) for j in range(n)]
        H[i-1] = h
    return H
```

In [ ]:
```

```

In [63]:
```
n = 1000
```

In [64]:
```
H = GenerateHilbert(n)
```

In [65]:
```
R1 = function1(H)
```

In [66]:
```
R2 = function2(H)
```

In [67]:
```
R3 = function1(R1)
```

In [68]:
```
R4 = function1(R2)
```

Since $R_1^T R_1$ and $R_2^T R_2$ are supposed to be equal to the identity matrix, the sum of all off diagonal entry should be zero, therefore $G_k := R_k^T R_k - \mathbb{I}$, $\Sigma_k = \sum_{i=1}^{n} \sum_{j=1}^{n} |g_{ij}^k|$ hast to be zero. Due to numerical / rounding errors, this does not hold for the computed matrices above. Large Σ_k indicates, that the according matrix R_k is not orthogonal.

In [69]:
```
np.sum(abs(np.matmul(R1.T, R1) - np.identity(n)))
```

Out[69]:
```
319615.3517070741
```

In [70]:
```
np.sum(abs(np.matmul(R2.T, R2) - np.identity(n)))
```
Out[70]:
```
4228.080270900577
```
In [71]:
```
np.sum(abs(np.matmul(R3.T, R3) - np.identity(n)))
```
Out[71]:
```
555770.9540641983
```
In [72]:
```
np.sum(abs(np.matmul(R4.T, R4) - np.identity(n)))
```
Out[72]:
```
1.3740774860795175e-06
```

Function2 yields better results, since the orthogonalization inside the inner loop is not directly based on the according vector x_j, the numerical errors are taken into account during the process. This results in a more orthogonal matrix

In [ ]:

In [ ]:

In [ ]: