# Numerical Linear Algebra for Computational Sciences and Information Engineering
# CITHN2006

Least-Squares Problems

by Nicolas Venkovic

Chair of Computational Mathematics
School of Computation, Information and Technology (CIT)
Technical University of Munich, Germany

**1.** *Implement the Golub-Kahan bidiagonalization procedure. Then, compare* $U_{k+1}^T A V_k$ *to* $\underline{B}_k$*, and check the loss of orthogonality of* $U_{k+1}$ *and* $V_k$*. What do you observe?*

[1]:
```
using LinearAlgebra, Printf, BenchmarkTools, Random, SparseArrays
```

[2]:
```
function bidiagonalization(A, r0, k)
  m, n = size(A)
  U = zeros(Float64, m, k + 1)
  V = zeros(Float64, n, k + 1)
  beta = zeros(Float64, k + 1)
  alpha = zeros(Float64, k + 1)
  U[:, 1] = r0
  beta[1] = norm(U[:,1])
  U[:, 1] = U[:, 1] ./ beta[1]
  V[:, 1] = A'U[:, 1]
  alpha[1] = norm(V[:, 1])
  V[:, 1] = V[:, 1] ./ alpha[1]
  for i = 1:k
    U[:, i + 1] = A * V[:, i] - alpha[i] * U[:, i]
    beta[i + 1] = norm(U[:, i+ 1])
    U[:, i + 1] = U[:, i + 1] / beta[i + 1]
    V[:, i + 1] = A'U[:, i + 1] - beta[i + 1] * V[:, i]
    alpha[i + 1] = norm(V[:, i + 1])
    V[:, i + 1] = V[:, i + 1] / alpha[i + 1]
  end
  return U, V[:, 1:k], alpha[1:k], beta
end;
```

[3]:
```
Random.seed!(3)
m = 100_000; n = 200;
A = sprand(Float64, m, n, .4);
b = rand(m);
```

[4]:
```
k = 5;
U, V, alpha, beta = bidiagonalization(A, rand(m), k);
B = spdiagm(k + 1, k, 0=>alpha, -1=>beta[2:k+1]);
norm(U'A*V - B)
```

[4]: 6.415087381394998

[5]:
```
U'A*V
```

[5]:
```
6×5 Matrix{Float64}:
 773.996         8.41039e-11   1.57414e-7   0.000298512   0.589457
 457.727         4.65152       9.30947e-8   0.00017654    0.348604
   9.09368e-10  96.5391        4.41971      9.91893e-8    0.000195861
   1.70404e-6    4.54171e-9   96.5037       4.36742       1.04521e-7
   0.00322698    8.59641e-6    4.58986e-9  96.6373        4.1883
   6.37841       0.0169915     9.07075e-6   4.78334e-9   96.5428
```

[6]:
```
B
```

[6]:
```
6×5 SparseMatrixCSC{Float64, Int64} with 10 stored entries:
 773.996
 457.727   4.65152
          96.5391    4.41971
                    96.5037    4.36742
                              96.6373    4.1883
                                        96.5428
```

[7]:
```
norm(I - U'U)
```

[7]: 0.01003147391463673

[8]:
```
norm(I - V'V)
```

[8]: 0.0010770355772659224

**2.** *Implement LSQR using the following stopping criterion:*

$$\frac{\|A^T r_i\|_2}{\|A\|_F \|r_i\|_2} < \varepsilon_{tol}.$$

*Apply your algorithm to large sparse least-squares problems.*

[9]:
```
function LSQR(A, b, x0, k, tol)
  A_F = sqrt(sum(A.nzval.^2))
  m, n = size(A)
```

```
  U = zeros(Float64, m, k + 1)
  V = zeros(Float64, n, k)
  B = spdiagm(k + 1, k, 0=>zeros(Float64, k), -1=>zeros(Float64, k))
  U[:, 1] = b - A * x0
  beta1 = norm(U[:,1])
  U[:, 1] = U[:, 1] ./ beta1
  V[:, 1] = A'U[:, 1]
  B[1, 1] = norm(V[:, 1])
  V[:, 1] = V[:, 1] / B[1, 1]
  beta1_x_e1 = zeros(Float64, k + 1); beta1_x_e1[1] = beta1
  i = 0
  err = 1.
  yi = 1.
  while (err > tol && i < k)
    i += 1
    U[:, i + 1] = A * V[:, i] - B[i, i] * U[:, i]
    B[i + 1, i] = norm(U[:, i + 1])
    U[:, i + 1] = U[:, i + 1] / B[i + 1, i]
    V[:, i + 1] = A'U[:, i + 1] - B[i + 1, i] * V[:, i]
    B[i + 1, i + 1] = norm(V[:, i + 1])
    V[:, i + 1] = V[:, i + 1] / B[i + 1, i + 1]
    yi = B[1:i+1, 1:i] \ beta1_x_e1[1:i+1]
    ti = beta1_x_e1[1:i+1] - B[1:i+1, 1:i] * yi  # ||ti|| =  ||ri||
    si = B[1:i+1, 1:i+1]'ti                      # ||si|| = ||A'ri||
    err = norm(si) / (A_F * norm(ti))
    @printf "i = %i, ||A'ri||/(||A||.||ri||) = %.2E\n" i err
  end
  return x0 + V[:, 1:i] * yi
end;
```

[10]:
```
x = LSQR(A, b, rand(n), 100, 1e-5);
norm(b - A * x)
```

```
i = 1, ||A'ri||/(||A||.||ri||) = 5.75E-02
i = 2, ||A'ri||/(||A||.||ri||) = 1.07E-02
i = 3, ||A'ri||/(||A||.||ri||) = 4.53E-04
i = 4, ||A'ri||/(||A||.||ri||) = 2.17E-05
i = 5, ||A'ri||/(||A||.||ri||) = 1.07E-06
```

[10]: 92.84273159426289

[11]:
```
x = A \ b;
norm(A * x - b)
```

[11]: 92.84273158157164