Parallel & Distributed Computing: Lecture 34

Alberto Paoluzzi

December 18, 2019

1 Linear Algebra Iterative Methods

Linear Algebra Iterative Methods

Systems of linear equations

Section 8.3 of VMLS book

Solving linear equations

Section 11.3 of VMLS book

QR factorization

Any real square matrix A may be decomposed as $A = QR$ where Q is an orthogonal matrix and R is an upper triangular matrix

```
julia> R = randn(4,4) # normally-distributed w mean=0 & dev=1

julia> LinearAlgebra.qr(R)
LinearAlgebra.QRCompactWY{Float64,Array{Float64,2}}
Q factor:
4×4 LinearAlgebra.QRCompactWYQ{Float64,Array{Float64,2}}:
 -0.146012   0.488873  -0.576087   -0.638598
 -0.621825  -0.672311   0.0309913  -0.400462
  0.240313  -0.449997  -0.79812     0.320556
 -0.730933   0.326348  -0.173688    0.573643
R factor:
4×4 Array{Float64,2}:
 -1.67505  -0.157206  -0.517163  0.480058
  0.0      -1.7074     0.511217  0.267463
  0.0       0.0       -1.01251   0.628989
  0.0       0.0        0.0       0.41568
```

Gram–Schmidt algorithm

Gram–Schmidt algorithm Section 5.4, page 97 of VMLS

QR factorization example Section 10.4, page 190, of VMLS companion

Julia example

VMLS Algorithm 11.1

```
using LinearAlgebra
function back_subst(R,b)
    n = length(b)
    x = zeros(n)
    for i=n:-1:1
        x[i] = (b[i] - R[i,i+1:n]' * x[i+1:n]) / R[i,i]
    end
    return x
end;

R = triu(randn(4,4)) # Random 4x4 upper triangular matrix 4×4
b = rand(4);
x = back_subst(R, b);
norm(R * x - b) # norm of error
```