# qr_shift

March 24, 2021

[57]:
```
using LinearAlgebra
```

[99]:
```
# alg. 9.6, with slight modifications
function qrws(A::SymTridiagonal{Float64,Array{Float64,1}}, tol=1e-5, maxit=1000)
    a = A.dv;
    d = copy(a);
    b = A.ev;
    n = length(diag);
    k = 1;
    shift = 0;
    all_lams = zeros(Float64, 0);
    while k <= maxit
        # step 3
        if abs(b[n]) <= tol
            all_lams = append!(all_lams, a[n] + shift);
            n = n - 1;
        end
        # step 4
        if abs(b[2]) <= tol
            all_lams = append!(all_lams, a[1] + shift);
            n = n - 1;
            a[1] = a[2];
            for j in collect(2:1:n)
                a[j] = a[j+1];
                b[j] = b[j+1];
            end
        end
        # step 5
        if n == 0
           break
        end
        # step 6
        if n == 1
            lambda = a[1] + shift;
            all_lams = append!(all_lams, lambda);
        end
        # step 7
        for j in collect(3:1:(n-1))
            if abs(b[j]) <= tol
                # split, no output, this solution is for simplicity
                break
            end
        end
        # step 8
        _b = -(a[n-1] + a[n]);
        _c = a[n]*a[n-1] - (b[n])^2;
        _d = sqrt((_b^2 - 4*_c));
        if _b > 0
            m1 = -2*_c/(_b+_d);
            m2 = -(_b+_d)/2;
        else
            m1 = (_d-_b)/2;
            m2 = 2*_c/(_d-_b);
        end
        # step 10
        if n == 2
            lambda1 = m1 + shift;
            lambda2 = m2 + shift;
            all_lams = append!(all_lams, [lambda1, lambda2]);
            break
        end
        #  step 11 choosing
        if abs(m1 - a[n]) <= abs(m2 - a[n])
            sig = m1;
        else
            sig = m2;
        end
        # step 12
        shift = shift + sig;

        # step 13
        for j in collect(1:n)
            d[j] = a[j] - sig;
        end
        # step 14 and 15
        x = zeros(Float64, n); y = zeros(Float64, n);
        c = zeros(Float64, n); sigmas = zeros(Float64, n);
        z = zeros(Float64, n); q = zeros(Float64, n);
        r = zeros(Float64, n);
        x[1] = d[1];
        y[1] = b[1]; # A[2,1]
        for j in collect(2:n)
            z[j-1] = sqrt((x[j-1])^2 + (b[j])^2);
            c[j] = x[j-1]/z[j-1];
            # println(z)
            sigmas[j] = b[j]/z[j-1];
            # there was a typo in B&F !
            q[j-1] = c[j]*y[j-1] + sigmas[j]*d[j];
            x[j] = -sigmas[j]*y[j-1] + c[j]*d[j];
            if j != n
                r[j-1] = sigmas[j]*b[j+1];
                y[j] = c[j]*b[j+1];
            end
        end
            # steps 16-18
        z[n] = x[n];
        a[1] = sigmas[2]*q[1] + c[2]*z[1];
        b[2] = sigmas[2]*z[2];

        for j in collect(2:1:(n-1))
            a[j] = sigmas[j+1]*q[j] + c[j]*c[j+1]*z[j];
            b[j+1] = sigmas[j+1]*z[j+1];
        end
        # step 18
        a[n] = c[n]*z[n];
        k = k + 1;
    end
end
```

```
[99]: qrws (generic function with 9 methods)
```