Método QR

O método de QR possui como objetivo obter uma matriz $\bar{A}$ que seja diagonal. Se $\bar{A}$ for diagonal, os autovalores de $\bar{A}$ são os próprios elementos da sua diagonal e os autovetores v_i associados são as colunas da matriz identidade I.

Assim, teremos automaticamente os autovalores de A (que são os mesmos de $\bar{A}$, e pela equação abaixo, também temos os autovetores de A.

$$X = PI = P$$

Ou seja, os autovetores de A são as próprias colunas da matriz P.

Lembrando que, a matriz $\bar{A}$ é uma matriz tal que existe um P onde $\bar{A} = P^T AP$.

Desenvolvimento

O método QR faz uma sequência de transformações de similaridade até que a matriz final tenha uma estrutura simples chamada matriz diagonal (lembrando que isso só ocorre pois estamos lidando com matrizes simétricas). As seguintes operações são realizadas em sequência:

$$\bar{A} = Q_k^T \left(Q_i^T \dots \left(Q_2^T \left(Q_1^T A Q_1\right) Q_2\right) \dots Q_i\right) Q_k$$

onde em cada passo nos aproximaremos mais de uma matriz diagonal. Vamos explorar um pouco mais como o método se dá, ainda ignorando quem é a nossa matriz Q.

Método QR

O algoritmo receber uma matriz A, a qual desejamos achar os autovalores e os autovetores, e uma tolerância ϵ que serve para verificar se os elementos fora da diagonal já estão suficientemente próximos de zero. Todas as matrizes envolvidas são acumuladas para serem usadas na recuperação dos autovetores da matriz original.

Algoritmo

```
metodoQR(Matriz A, int n, float epsilon) {
    Matriz P, Q, R, A_nova, A_velha, A_barra;
    Vetor Lamb; // Armazenará os autovalores de A
    float val = INFINITO; // Valor arbitrário, só precisa ser maior que
epsilon
```

```
    P <- I; // Aqui teremos o produto das matrizes Q (acumulada) para
retornar os autovetores.
    A_velha <- A;

    Enquanto(val > epsilon) {
        (Q, R) <- decomposiçãoQR(A_velha, n); // Explicaremos depois
        A_nova <- RQ; // O mesmo que A_nova <- Q_transposta A_velha Q
        A_velha <- A_nova;
        P <- P Q // Matriz acumulada

        val = somaDosQuadradosDosTermosAbaixoDaDiagonal(A_nova, n) // Também
explicaremos
    }
    Lamb(1 : n) <- (A_nova)i,i (1:n); // Copia os elementos da diagonal da
matriz
    return(P, Lamb)
}
```

Podemos ver que é um algoritmo bem direto! Mas ainda falta mostrar como a decomposição QR é feita.

Decomposição QR

A decomposição QR é assim chamada pois conseguimos representar uma matriz A da seguinte maneira:

$$A = QR$$

Onde Q é uma matriz ortogonal e R é uma matriz triangular superior. A técnica que usaremos para decompor a matriz será utilizando uma série de **Givens rotation** em sequência. Cada rotação será responsável por zerar um elemento na subdiagonal da matriz, formando a matriz R. A concatenação de todas as matrizes de rotação formarão a matriz ortogonal Q.

Algoritmo

```
decomposiçãoQR(Matriz A, int n) {
    A_old <- A;
    Q <- I;

    Para j = 1 : n - 1 {
        Para i = (j + 1) : n {
            q <- I;
```

```
            x <- A_old(j,j);
            y <- A_old(i,j);
            l <- sqrt(x² + y²);
            c <- x / l; // cos
            s <- - y / l; // sen
            q(i, i) <- c;
            q(j, j) <- c;
            q(i, j) <- s;
            q(j, i) <- -s; // Constroi a matriz de rotação
            A_nova <- q A_old; // Aplica a rotação
            Q <- Q q^T; // Acumula as rotações
            A_old = A_nova;
        }
    }
    return (Q, A_nova);
}
```

Vamos esclarecer algumas coisas:

- A é uma matrix $m \times n$, logo, a matriz de rotação q precisa ser inicializada como $m \times m$ para que a multiplicação seja possível.
- No nosso código, não precisaremos passar n como parâmetro, e extrairemos o valor da matriz.
- c e s representam o $\cos$ e o $\sin$, respectivamente.

Método QR + Householder

Aproveitando o que já vimos anteriormente, podemos tentar adaptar o método QR para utilizar da matriz de Householder, aproveitando da estrutura tridiagonal existente. Vejamos um algoritmo que usa da matriz H

Algoritmo

```
métodoQR(T, n H, epislon) {
    A_old <- T;
    X <- H;
    e <- INFINITO;
    Enquanto ( e > epislon ) {
        (Q, R) <- decomposiçãoQR(A_old, n);
        A_I <- RQ // Equivalente a Q^T A_old Q
        A_old <- A_i
```

```
        X <- X Q
        e <- testMatDiag(A_old, n) // Mesmo que no método QR normal,
verificar convergência
    }
    Return (A_i, X);
}
```

Novamente, escalerecendo algumas coisas:

- A matriz T é a matriz resultante "A" do método de Householder
- H é a matriz acumulada, também resultante do método de Householder
- A_i possui como elementos da diagonal principal como autovalores da matriz A, enquanto X acumula as matrizez ortogonais geradas. Ao final, X será a matriz cujas colunas são os autovetores da matriz original T. Como T foi fruto da decomposição da matriz de Householder, precisaremos fazer HX para encontrar os autovetores da matriz original.

Testando a Convergência

Por último, vamos mostrar o algoritmo que ficou, pendente, o que verifica o valor que usamos para determinar a convergência da matriz:

```
testMatDiag(A, n) {
    s <- 0
    Para (j = 1 : n - 1) {
        Para (i = j + 1, n) {
            S <- S + A.(i).(j)^2
        }

    }
    L = sqrt(S);
    return L;
}
```