Computational methods-Lecture 12

Householder's method and QR method for eigenvalues

1 First approach for QR decomposition using Householder transform

Using the Householder transforms, one can obtain the following result:

Theorem 1 (QR decomposition)**.** *If A is nonsingular, then there exists an orthogonal matrix Q such that*

$$A = QR,$$

where R is upper triangular. If one requires the diagonal elements of R to be positive, then the decomposition is unique.

Moreover, if A is of size $m \times n$ $(m > n)$ and with rank n, then $QA = \begin{pmatrix} R \\ 0 \end{pmatrix}$.

The proof is straightforward: we apply Householder's transform repeatedly.

We first pick $Q_1 = H_1$ that maps the the first column of A to $-\sigma_1 e_1$. Let

$$A_1 = Q_1 A.$$

Next, one is going to construct matrix of the form

$$Q_2 = \begin{pmatrix} 1 & 0 \\ 0 & H_2 \end{pmatrix}$$

where H_2 will map the first column of the $(2:n, 2:n)$ submatrix of A_1 into $-\sigma_2 e_1$. Then,

$$A_2 = Q_2 A_1.$$

Repeating this process, the theorem can be proved.

Another approach that is useful for the QR decomposition, and in fact used in practice for tridiagonal matrices is based on Givens transform. This is not required in this course.

2 Householder's method

We have seen that using Householder transforms one can have the QR decomposition

$$A = QR.$$

However. this is not a similar transform. A question is: can we use Householder transform to obtain a similar transform to obtain a band matrix with small bandwidth?

Below, we try to argue that using Householder transform, we can find similar transformations to change

- a real square matrix into Upper Hessenberg matrix (i.e. $h_{ij} = 0$ for $i \geq j + 2$)
- a real symmetric matrix into a tridiagonal matrix.

Let us make a first attempt: take $Q_1 = H_1$ that maps the first column of A into $-\sigma_1 e_1$. Then, consider the similar transform:

$$H_1 A H_1^T = [-\sigma_1 e_1 \ v_2 \ \cdots \ v_n] H_1$$

Unfortunately, after we multiply H_1 on the right, the matrix will generally be full again. Hence, we cannot desire an orthogonal similar transform that changes A into upper triangular or so.

An important observation is that though we cannot change the first column into a vector with only one nonzero component, we can change it to a vector with two nonzero components!

Let the first column of A be

$$u_1 = \begin{pmatrix} a_{11} \\ c_1 \end{pmatrix}$$

We take H_1 such that $H_1 c_1 = -\sigma_1 e_1$. Consider the transform

$$Q_1 = \begin{pmatrix} 1 & 0 \\ 0 & H_1 \end{pmatrix}$$

Then, we have

$$Q_1 A Q_1^T = \begin{pmatrix} a_{11} & A_{12}^{(1)} \\ H_1 c_1 & H_1 A_{22}^{(1)} H_1 \end{pmatrix}.$$

One can see that Hc_1 has only one nonzero entry.

Note that Q_1 is also a **reflection matrix**.

Next, one considers

$$Q_2 = \begin{pmatrix} I_2 & 0 \\ 0 & H_2 \end{pmatrix}$$

where H_2 is of size $(n-2)\times(n-2)$. We use this matrix to change the vector below the first entry in the first column of $H_1 A_{22}^{(1)} H_1$ into $-\sigma_2 e_1$.

Repeat this process, we have

Theorem 2. *Let A be a real square matrix. Then, there exists reflection matrices $Q_1, \cdots, Q_{n-2}$ such that*

$$Q_{n-2} \cdots Q_1 A Q_1 \cdots Q_{n-1}$$

is upper Hessenberg. If A is symmetric, this matrix is tridiagonal.

3 The QR Algorithm for finding all eigenvalues

The QR algorithm can be used for find all eigenvalues of matrices provided some conditions. The QR algorithm is as following.

QR algorithm

1. Let $A_1 \leftarrow A$
2. For $k = 1, \cdots,$

   $A_k = Q_k R_k$

   $A_{k+1} = R_k Q_k$.

Below is a simple observation:

Lemma 1. *A_k is similar to A_{k+1}.*

In fact, $A_{k+1} = R_k Q_k = (Q_k^{-1} A_k) Q_k = Q_k^T A_k Q_k$.

That means the sequence $\{A_k\}$ will preserve eigenvalues!

The following fact then guarantees the convergence (for general case, read the reference books):

Theorem 3. *Suppose that A has eigenvalues $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$, and $A = XDX^{-1}$ such that X^{-1} has LU decomposition. Then, the diagonal elements of A_k converge to the eigenvalues.*

Though we have a way for finding QR decomposition using Householder transform, it is not simple. In the case that the matrix is **upper Hessenberg or tridiagonal**, one can use Givens transform to find the QR decomposition efficiently.

Hence, the for general matrices, the approach is as following:

1. Use the Householder's method to transform a matrix into upper Hessenberg or tridiagonal.

2. Apply the QR algorithm (using Givens transform) to find all eigenvalues.

4 Initial value problems for ODEs

The initial value problem (also called Cauchy problem) of an ODE is Consider the ODE

$$u'(t) = f(t, u(t)), \quad u(t_0) = u_0,$$

where u could be a vector valued function. Any ODE can be reduced to a first order system, so this is general enough.

For studying ODEs, we often assume

Assumption 1. *The function f is Lipschitz continuous in u with a uniform Lipschitz constant. In other words, there exists $L > 0$ such that*

$$|f(t, x) - f(t, y)| \leq L|x - y|$$

This condition guarantees the existence and uniqueness of the solution to the ODE for a given initial value.

Suppose we care the solution at T. For discretization, we take the time step:

$$h = \frac{T}{N}.$$

Define

$$t_n := nh.$$

The ODE solvers are all approximations to

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(s, u(s))ds.$$

$\int_{t_n}^{t_{n+1}} f(s, u(s))$ will be approximated by data $u_0, u_1, \ldots, u_{n+1}$.

Below we consider some simple approximations.

- If we approximate $f(s, u(s)) \approx f(t_n, u_n)$, then we have the forward Euler:

$$u_{n+1} = u_n + hf(t_n, u_n)$$

- $f(s, u(s)) \approx f(t_{n+1}, u_{n+1})$, we have the backward Euler:

$$u_{n+1} = u_n + hf(t_{n+1}, u_{n+1})$$

- $f(s, u(s)) \approx \frac{1}{2}(f(t_n, u^n) + f(t_{n+1}, u_{n+1}))$, then we have the trapezoidal method:

$$u_{n+1} = u_n + \frac{h}{2}(f(t_n, u_n) + f(t_{n+1}, u_{n+1}))$$

- (*) The higher order Taylor methods (see below, not required)

The forward Euler and backward Euler methods are different in the sense that the foward Euler is an **explicit** scheme while the backward Euler is an implicit scheme. By **"implicit"**, we mean the right hand side contains the desired solution u_{n+1}, so u_{n+1} cannot be evaluated directly. Instead, one must solve the algebraic equation to obtain u_{n+1}.

Example For the equation $y' = \lambda y$. Apply forward Euler scheme and backward Euler scheme. Take $h \to 0$, does it give the true solution?

The forward Euler has

$$y_n = y_0(1 + \lambda h)^n,$$

while the backward Euler gives

$$y_n = \frac{y_0}{(1 - \lambda h)^n}.$$

For any λ, we have

$$y_0(1 + \lambda h)^n \leq y_0 e^{\lambda n h} \leq \frac{y_0}{(1 - \lambda h)^n}.$$

This means that the explicit method often gives a solution that is below the true solution while an implicit method often gives a solution above the true solution.

Taking $h \to 0$, we have

$$y_{t/h} \to e^{\lambda t}.$$