Power Iteration

- finding eigenvalues is mathematically equivalent to finding zeros of polynomials
  - Abel-Ruffini Theorem: general polynomial equations of degree higher than 4 do not admit solutions by radicals (i.e., solutions via rational numbers using the addition, subtraction, multiplication, division, and kth roots)
  - all eigenvalue solvers must be iterative
- power iteration/method: a method of numerically computing an eigenvalue and an eigenvector of a given matrix
- simple but provides the idea for a bunch of eigenvalue algorithms
- not the best in convergence speed
- suitable for large-scale sparse eigenvalue problems, e.g., PageRank (similar to cases in linear systems and LS, solving eigenvalue problems for large sparse matrices is an important topic)
- a comprehensive coverage of various computational methods for the eigenvalue problem can be found in the textbook **[Golub-Van Loan'13]**

Power Iteration

- assumptions:
  - $\mathbf{A}$ admits an eigendecomposition $\mathbf{A} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1}$
  - λ_i's are ordered such that $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$
  - we have an initial guess $\mathbf{q}^{(0)}$ that satisfies $[\mathbf{V}^{-1}\mathbf{q}^{(0)}]_1 \neq 0$ (random guess should do)

- consider $\mathbf{A}^k\mathbf{x}$. Let $\boldsymbol{\alpha} = \mathbf{V}^{-1}\mathbf{q}^{(0)}$, and observe

$$\mathbf{A}^k\mathbf{q}^{(0)} = \mathbf{V}\boldsymbol{\Lambda}^k\mathbf{V}^{-1}\mathbf{q}^{(0)} = \sum_{i=1}^{n} \alpha_i\lambda_i^k\mathbf{v}_i = \alpha_1\lambda_1^k\left(\mathbf{v}_1 + \underbrace{\sum_{i=2}^{n} \frac{\alpha_i}{\alpha_1}\left(\frac{\lambda_i}{\lambda_1}\right)^k \mathbf{v}_i}_{=\mathbf{r}_k}\right)$$

where $\mathbf{r}_k$ is a residual and has

$$\|\mathbf{r}_k\|_2 \leq \sum_{i=2}^{n} \left|\frac{\alpha_i}{\alpha_1}\right| \left|\frac{\lambda_i}{\lambda_1}\right|^k \|\mathbf{v}_i\|_2 \leq \left|\frac{\lambda_2}{\lambda_1}\right|^k \sum_{i=2}^{n} \left|\frac{\alpha_i}{\alpha_1}\right|$$

- convergence: let $c_k = \frac{|\alpha_1||\lambda_1|^k}{\alpha_1\lambda_1^k}$, i.e., the sign of $\alpha_1\lambda_1^k$ (note $|c_k| = 1$). We have

$$\lim_{k\to\infty} c_k \frac{\mathbf{A}^k\mathbf{q}^{(0)}}{\|\mathbf{A}^k\mathbf{q}^{(0)}\|_2} = \mathbf{v}_1$$

Power Iteration

```
**Algorithm:** Power Iteration
**input:** $\mathbf{A} \in \mathbb{C}^{n \times n}$ and a starting vector $\mathbf{q}^{(0)} \in \mathbb{C}^n$
$k = 0$
repeat
    $\tilde{\mathbf{q}}^{(k+1)} = \mathbf{A}\mathbf{q}^{(k)}$
    $\mathbf{q}^{(k+1)} = \tilde{\mathbf{q}}^{(k+1)} / \|\tilde{\mathbf{q}}^{(k+1)}\|_2$        % normalization
    $\lambda^{(k+1)} = R(\mathbf{q}^{(k+1)}) = (\mathbf{q}^{(k+1)})^H \mathbf{A}\mathbf{q}^{(k+1)}$
    $k := k + 1$
until a stopping rule is satisfied
**output:** $\mathbf{q}^{(k)}$, $\lambda^{(k)}$
```

- by induction on k, it can be verified that $\mathbf{v}^{(k)} = \dfrac{\mathbf{A}^k \mathbf{q}^{(0)}}{\|\mathbf{A}^k \mathbf{q}^{(0)}\|_2}$
- it finds the dominant eigen-pair, i.e., dominant eigenvalue λ_1 (largest eigenvalue in modulus) and dominant eigenvector $\mathbf{v}_1$ only, unless $\alpha_1 = 0$
- complexity per iteration: $\mathcal{O}(n^2)$, or $\mathcal{O}(\mathrm{nnz}(\mathbf{A}))$ for sparse $\mathbf{A}$

Power Iteration

- convergence rate depends on $\left|\frac{\lambda_2}{\lambda_1}\right|$
  - $\|\mathbf{q}^{(k)} - \mathbf{v}_1\|_2 = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$ and $|\lambda^{(k)} - \lambda_1| = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$ ($\mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right)$ for Hermitian $\mathbf{A}$)
  - slower if $|\lambda_2|$ is closer to $|\lambda_1|$, i.e., $\left|\frac{\lambda_2}{\lambda_1}\right|$ is closer to 1
  - reduction per iteration is a constant, i.e., linear convergence

- now what if $|\lambda_1| = |\lambda_2| = \ldots = |\lambda_K| \geq \ldots \geq |\lambda_n|$ for some K?
  - by extending the convergence analysis, it can be shown $\mathbf{q}^{(k)}$ will converge to a vector in the subspace of $\mathrm{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K\}$
  - an important special case: for $\mathbf{A} \in \mathbb{R}^{2\times 2}$, if its complex eigenvalues come in conjugate pairs, then $\mathbf{q}^{(k)}$ will always be in the subspace spanned by the eigenvectors corresponding to the two eigenvalues

Power Iteration With Deflation

- the power method finds the largest eigenvalue (in modulus) and the correponding eigenvector only
- how can we compute all the eigenvalues and eigenvectors?
- there are many ways and let's first consider a simple method called deflation
- consider a Hermitian $\mathbf{A}$ with $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$, and note the outer-product representation

$$\mathbf{A} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^H = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^H.$$

- Hotelling's deflation: use the power iteration to obtain $\mathbf{v}_1, \lambda_1$, do the subtraction

$$\mathbf{A} := \mathbf{A} - \lambda_1 \mathbf{v}_1 \mathbf{v}_1^H = \sum_{i=2}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^H,$$

  and repeat until all the eigenvectors and eigenvalues are found
- there are more deflation techniques which are not just for Hermitian matrices (learn by yourself)

Power Iteration With Shift

- $\mathbf{A} - \mu\mathbf{I}$ is with eigenvalue $(\lambda_i - \mu)$'s and the same eigenvector $\mathbf{v}_i$'s as $\mathbf{A}$
- if λ_1 is known (approximately), convergence can be made faster by applying power iteration for $\mathbf{A} - \mu\mathbf{I}$ s.t.
  - $(\lambda_1 - \mu)$ is the largest eigenvalue in modulus for $\mathbf{A} - \mu\mathbf{I}$
  - $\max_{i=2,\ldots,n} \left|\frac{\lambda_i - \mu}{\lambda_1 - \mu}\right|$ is as smaller as possible than $\left|\frac{\lambda_2}{\lambda_1}\right|$
- obviously, in practice hard to decide μ; the extent of acceleration is limited
- shift technique is commonly used together with inverse iteration and QR iteration to be introduced later

Inverse Iteration

- $(\mathbf{A} - \mu\mathbf{I})^{-1}$ is with eigenvalue $(\lambda_i - \mu)^{-1}$'s and the same eigenvector $\mathbf{v}_i$'s as $\mathbf{A}$
- inverse (power) iteration with shift: apply power iteration on $(\mathbf{A} - \mu\mathbf{I})^{-1}$
- if $\mu \approx \lambda_J$ for some J, $|(\lambda_J - \mu)^{-1}|$ may be far larger than $|(\lambda_i - \mu)^{-1}|$ for $i \neq J$, so power iteration can converge rapidly

```
Algorithm: Inverse Iteration
input: $\mathbf{A} \in \mathbb{C}^{n\times n}$ and a starting vector $\mathbf{q}^{(0)} \in \mathbb{C}^n$
$k = 0$
repeat
    $\tilde{\mathbf{q}}^{(k+1)} = (\mathbf{A} - \mu\mathbf{I})^{-1}\mathbf{q}^{(k)}$    % solve $(\mathbf{A} - \mu\mathbf{I})\tilde{\mathbf{q}}^{(k+1)} = \mathbf{q}^{(k)}$
    $\mathbf{q}^{(k+1)} = \tilde{\mathbf{q}}^{(k+1)}/\|\tilde{\mathbf{q}}^{(k+1)}\|_2$    % normalization
    $\lambda^{(k+1)} = R(\mathbf{q}^{(k+1)}) = (\mathbf{q}^{(k+1)})^H\mathbf{A}\mathbf{q}^{(k+1)}$
    $k := k + 1$
until a stopping rule is satisfied
output: $\mathbf{q}^{(k)}$, $\lambda^{(k)}$
```

- $\mathbf{q}^{(k)}$ converges to eigenvector $\mathbf{v}_J$ if parameter μ is close to λ_J
- complexity per iteration: $\mathcal{O}(n^2)$ (matrix $(\mathbf{A} - \mu\mathbf{I})$ is processed in advance)

Inverse Iteration

- convergence rate with
  - $\|\mathbf{q}^{(k)} - \mathbf{v}_J\|_2 = \mathcal{O}\left(\left(\max_{i=1,\ldots,n}\left|\frac{\lambda_J - \mu}{\lambda_i - \mu}\right|\right)^k\right)$
  - $|\lambda^{(k)} - \lambda_J| = \mathcal{O}\left(\left(\max_{i=1,\ldots,n}\left|\frac{\lambda_J - \mu}{\lambda_i - \mu}\right|\right)^k\right)$

    where λ_J is the closest eigenvalue to μ
  - reduction per iteration is a constant, i.e., linear convergence

- standard method for determining any eigenvector given an eigenvalue

- a linear system needs to be solved; similar to power iteration, can only compute one eigenpair

- inverse iteration without shift: taking $\mu = 0$ the algorithm converges to the eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}$ (in modulus)

Rayleigh Quotient Iteration

- parameter μ is constant in inverse iteration, but convergence is better for μ close to the eigenvalue
- improvement: setting μ as the last computed Rayleigh quotient at each iteration

```
Algorithm: Rayleigh Quotient Iteration
input: $\mathbf{A} \in \mathbb{C}^{n \times n}$ and a starting vector $\mathbf{q}^{(0)} \in \mathbb{C}^n$
$k = 0$
$\mu^{(k)} = R(\mathbf{q}^{(k)})$
repeat
    $\tilde{\mathbf{q}}^{(k+1)} = (\mathbf{A} - \mu^{(k)}\mathbf{I})^{-1}\mathbf{q}^{(k)}$        % solve $(\mathbf{A} - \mu^{(k)}\mathbf{I})\tilde{\mathbf{q}}^{(k+1)} = \mathbf{q}^{(k)}$
    $\mathbf{q}^{(k+1)} = \tilde{\mathbf{q}}^{(k+1)} / \|\tilde{\mathbf{q}}^{(k+1)}\|_2$        % normalization
    $\mu^{(k+1)} = \lambda^{(k+1)} = R(\mathbf{q}^{(k+1)}) = (\mathbf{q}^{(k+1)})^H \mathbf{A} \mathbf{q}^{(k+1)}$
    $k := k + 1$
until a stopping rule is satisfied
output: $\mathbf{q}^{(k)}$, $\lambda^{(k)}$
```

- at least quadratic convergence, but uncertain to which eigenvalue it will converge
- complexity per iteration: $\mathcal{O}(n^3)$ (solving a different linear system each iteration)

Orthogonal Iteration

- for the previous methods, only find one eigenpair each time
  - what if we want more eigenvalues rather than λ_1
  - what if λ_1 and λ_2 are close or equal and we cannot decide shift μ; in this case, we might want to look for an invariant subspace associated with λ_1 and λ_2
- subspace iteration: starting with a set of linearly independent vectors or a subspace $\mathcal{Q}^{(0)} = \mathrm{span}\{\mathbf{q}_1^{(0)},\ \mathbf{q}_2^{(0)},\ \cdots, \mathbf{q}_r^{(0)}\}$, $\mathcal{Q}^{(k)} = \mathbf{A}^k\mathcal{Q}^{(0)}$ will converge (under suitable assumptions) to a subspace spanned by eigenvectors associated with the r largest eigenvalues in magnititude, i.e., the dominant invariant subspace
  - in contrast, the power iteration is sometimes called vector iteration
  - use thin QR to get the bases $\mathbf{Q}^{(k)}$ as $\mathbf{Q}^{(k)}\mathbf{R}^{(k)} = \mathbf{A}^k\big[\mathbf{q}_1^{(0)}\ \mathbf{q}_2^{(0)}\ \cdots\ \mathbf{q}_r^{(0)}\big]$
- the above subspace iteration is an unnormalized simultaneous (power) iteration; since all of $\{\mathbf{A}^k\mathbf{q}_1^{(0)},\ \mathbf{A}^k\mathbf{q}_2^{(0)},\ \cdots, \mathbf{A}^k\mathbf{q}_r^{(0)}\}$ will converge to a multiple of $\mathbf{v}_1$, columns of $\mathbf{Q}^{(k)}$ will form an extremely ill-conditioned basis for $\mathcal{Q}^{(k)}$
- in practice, we use the orthogonal (simultaneous power) iteration also called block power iteration

Orthogonal Iteration

- suppose there is a gap between the r $(1 \leq r \leq n)$ largest eigenvalues and λ_{r+1} in magnititude, i.e, $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_r| > |\lambda_{r+1}|$

```
Algorithm: Orthogonal Iteration
input: $\mathbf{A} \in \mathbb{C}^{n\times n}$ and a starting semi-unitary matrix $\mathbf{Q}^{(0)} \in \mathbb{C}^{n\times r}$
$k = 0$
repeat
    $\tilde{\mathbf{Q}}^{(k+1)} = \mathbf{A}\mathbf{Q}^{(k)}$
    $\mathbf{Q}^{(k+1)}\mathbf{R}^{(k+1)} = \tilde{\mathbf{Q}}^{(k+1)}$   % orthogonalization; perform thin QR
    $\{\lambda_1^{(k+1)}, \lambda_2^{(k+1)}, \dots, \lambda_r^{(k+1)}\} = \sigma\left((\mathbf{Q}^{(k+1)})^H \mathbf{A}\mathbf{Q}^{(k+1)}\right)$
    $k := k + 1$
until a stopping rule is satisfied
output: $\mathbf{Q}^{(k)}$, $\{\lambda_1^{(k+1)}, \lambda_2^{(k+1)}, \dots, \lambda_r^{(k+1)}\}$
```

- it can be verified that $\mathcal{R}(\mathbf{Q}^{(k)}) = \mathcal{R}(\tilde{\mathbf{Q}}^{(k)}) = \mathcal{R}(\mathbf{A}\mathbf{Q}^{(k-1)})$ (recall QR Topic)
- then $\mathcal{R}(\mathbf{Q}^{(k)}) = \mathcal{R}(\mathbf{A}^k\mathbf{Q}^{(0)})$ (verify by yourself)

Orthogonal Iteration

- denote the Schur decomposition of $\mathbf{A}$ by $\mathbf{A} = \mathbf{U}\mathbf{T}\mathbf{U}^H$, s.t. $|t_{11}| \geq |t_{22}| \geq \cdots \geq |t_{rr}| > |t_{r+1,r+1}| \geq \cdots \geq |t_{nn}|$

- $\mathbf{Q}^{(k)}$ converges linearly to an orthonormal basis for the dominant invariant subspace associated with the r largest eigenvalues in magnitude $\mathcal{R}(\mathbf{U}(:, 1:r))$

- $\left[\lambda_1^{(k)}\ \lambda_2^{(k)}\ \cdots\ \lambda_r^{(k)}\right] = \mathrm{diag}\left(\left(\mathbf{Q}^{(k)}\right)^H \mathbf{A}\mathbf{Q}^{(k)}\right) \rightarrow [\lambda_1\ \lambda_2\ \cdots\ \lambda_r]$

- $\left|\lambda_i^{(k)} - \lambda_i\right| = \mathcal{O}\left(\left(\max_{i=1,\ldots,r}\left|\frac{\lambda_{i+1}}{\lambda_i}\right|\right)^k\right),\ i = 1,\ 2,\ \cdots,\ r$

Orthogonal Iteration

- let's take a look at the span of the columns in $\mathbf{Q}^{(k)}$
- given $\mathbf{Q}^{(k+1)}\mathbf{R}^{(k+1)} = \mathbf{A}\mathbf{Q}^{(k)}$, notice that the first p ($p = 1, \ldots, r$) columns $\mathbf{Q}^{(k+1)}$ satisfies the recurrence

$$\mathbf{Q}^{(k+1)}\mathbf{R}^{(k+1)}(:, 1:p) = \mathbf{Q}^{(k+1)}(:, 1:p)\mathbf{R}^{(k+1)}(1:p, 1:p) = \mathbf{A}\mathbf{Q}^{(k)}(:, 1:p)$$

  then $\mathcal{R}\big(\mathbf{Q}^{(k)}(:, 1:p)\big) = \mathcal{R}\big(\mathbf{A}^k\mathbf{Q}^{(0)}(:, 1:p)\big)$

- over iterations, the first p columns of $\mathbf{Q}^{(k)}$ converge to a basis for the dominant p-dimensional invariant subspace
- setting the initial $\mathbf{Q}^{(0)} \in \mathbb{C}^{n\times n}$, directly get a n-dimensional invariant subspaces
  - when $r = n$, orthogonal iteration resembles the QR iteration (or QR algorithm)

LR Iteration

- The QR iteration was preceded by the LR iteration (or LR algorithm), which uses LU (a.k.a. LR) decomposition instead of QR decomposition as building blocks.

```
Algorithm: LR Iteration
input: $\mathbf{A} \in \mathbb{C}^{n \times n}$
$\mathbf{A}^{(0)} = \mathbf{A}$
$k = 0$
repeat
    $\mathbf{L}^{(k+1)}\mathbf{R}^{(k+1)} = \mathbf{A}^{(k)}$        % perform LU decomp. for $\mathbf{A}^{(k)}$
    $\mathbf{A}^{(k+1)} = \mathbf{R}^{(k+1)}\mathbf{L}^{(k+1)}$
    $k := k + 1$
until a stopping rule is satisfied
output: $\mathbf{A}^{(k)}$
```

- $\mathbf{A}^{(k)}$ is similar to $\mathbf{A}^{(k+1)}$ in that $\mathbf{A}^{(k+1)} = \mathbf{L}^{-(k+1)}\mathbf{A}^{(k)}\mathbf{L}^{(k+1)}$
  - and hence to $\mathbf{A}^{(0)}$ since $\mathbf{A}^{(0)} = \left(\mathbf{L}^{(1)} \cdots \mathbf{L}^{(k)}\right)\mathbf{A}^{(k)}\left(\mathbf{L}^{-(k)} \cdots \mathbf{L}^{-(1)}\right)$
- under some mild assumptions, $\mathbf{A}^{(k)}$ converges linearly to an upper-triangular matrix with eigenvalues on its diagonal

QR Iteration

```
Algorithm: QR Iteration
input: $\mathbf{A} \in \mathbb{C}^{n\times n}$
$\mathbf{A}^{(0)} = \mathbf{A}$
$k = 0$
repeat
    $\mathbf{Q}^{(k+1)}\mathbf{R}^{(k+1)} = \mathbf{A}^{(k)}$        % perform QR decomp. for $\mathbf{A}^{(k)}$
    $\mathbf{A}^{(k+1)} = \mathbf{R}^{(k+1)}\mathbf{Q}^{(k+1)}$
    $k := k + 1$
until a stopping rule is satisfied
output: $\mathbf{A}^{(k)}$
```

- $\mathbf{A}^{(k)}$ is unitarily similar to $\mathbf{A}^{(k+1)}$ in that $\mathbf{A}^{(k+1)} = \left(\mathbf{Q}^{(k+1)}\right)^H \mathbf{A}^{(k)}\mathbf{Q}^{(k+1)}$
  - and hence to $\mathbf{A}^{(0)}$ since $\mathbf{A}^{(0)} = \left(\mathbf{Q}^{(1)}\cdots\mathbf{Q}^{(k)}\right)\mathbf{A}^{(k)}\left(\mathbf{Q}^{(1)}\cdots\mathbf{Q}^{(k)}\right)^H$
- denote the Schur decomposition of $\mathbf{A}$ by $\mathbf{A} = \mathbf{UTU}^H$; under some mild assumptions, $\mathbf{A}^{(k)}$ converges linearly to $\mathbf{T}$
  - if our problem is to compute all the eigenvalues of $\mathbf{A}$, picking the diagonal elements of $\mathbf{A}^{(k)}$ for a sufficiently large k would do

QR Iteration

- QR iteration is equivalent to orthogonal iteration with $\mathbf{Q}^{(0)} = \mathbf{I}$ (verify by yourself)
- the most popular method for computing all the eigenvalues of a general $\mathbf{A}$
  - the practical QR algorithm used in modern software is more sophisticated
- how to find the eigenvectors?
  - for λ_i, solve the eigen-equation $(\mathbf{T} - \lambda_i \mathbf{I})\mathbf{v} = \mathbf{0}$, which is an upper-triangular linear system
  - for λ_i, use the inverse iteration
- as a counterpart to power iteration, to accelerate the convergence speed of orthogonal iteration there exist orthogonal simultaneous inverse iteration, orthogonal simultaneous iteration with shift, etc.
  - in the next we investigate QR iteration with shift

QR Iteration With Shift

Example: consider matrix $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \mathbf{A}^{(0)}$

$$\underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}_{\mathbf{Q}^{(0)}} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{R}^{(0)}} = \mathbf{A}^{(0)}$$

$$\mathbf{A}^{(1)} = \mathbf{R}^{(0)}\mathbf{Q}^{(0)} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \mathbf{A}^{(0)}$$

no convergence of $\mathbf{A}^{(k)}$ observed

- shift can also help to make QR iteration converge, i.e., $\mathbf{A}^{(k)}$ converge to a upper triangular matrix

QR Iteration With Shift

```
**Algorithm:** QR Iteration With Shift
**input:** $\mathbf{A} \in \mathbb{C}^{n\times n}$
$\mathbf{A}^{(0)} = \mathbf{A}$
$k = 0$
repeat
    choose a shift $\mu^{(k)}$
    $\mathbf{Q}^{(k+1)}\mathbf{R}^{(k+1)} = \mathbf{A}^{(k)} - \mu^{(k)}\mathbf{I}$        % perform QR for $\mathbf{A}^{(k)} - \mu^{(k)}\mathbf{I}$
    $\mathbf{A}^{(k+1)} = \mathbf{R}^{(k+1)}\mathbf{Q}^{(k+1)} + \mu^{(k)}\mathbf{I}$
    $k := k + 1$
until a stopping rule is satisfied
**output:** $\mathbf{A}^{(k)}$
```

- similar to QR iteration, $\mathbf{A}^{(k)}$ is unitarily similar to $\mathbf{A}^{(k+1)}$
  - $\mathbf{A}^{(k+1)} = \mathbf{R}^{(k+1)}\mathbf{Q}^{(k+1)} + \mu^{(k)}\mathbf{I} = \left(\mathbf{Q}^{(k+1)}\right)^H\left(\mathbf{A}^{(k)} - \mu^{(k)}\mathbf{I}\right)\mathbf{Q}^{(k+1)} + \mu^{(k)}\mathbf{I} = \left(\mathbf{Q}^{(k+1)}\right)^H\mathbf{A}^{(k)}\mathbf{Q}^{(k+1)}$
- shift $\mu^{(k)}$ may differ from iteration to iteration

QR Iteration With Shift

- Rayleigh quotient shift
  - $\mu^{(k)} = \mathbf{A}^{(k)}(n,n)$ which will converge to the smallest eigenvalue in modulus
  - no guarantee on convergence
  - if converged, at least quadratic convergence

- Wilkinson shift
  - denote the lower-rightmost 2×2 matrix of $\mathbf{A}^{(k)}$ by

$$\bar{\mathbf{A}}^{(k)} = \begin{bmatrix} \mathbf{A}^{(k)}(n-1,n-1) & \mathbf{A}^{(k)}(n-1,n) \\ \mathbf{A}^{(k)}(n,n-1) & \mathbf{A}^{(k)}(n,n) \end{bmatrix}$$

  - chose the eigenvalue of $\bar{\mathbf{A}}^{(k)}$ closer to $\mathbf{A}^{(k)}(n,n)$
  - always converge with at least linear convergence **[Wilkinson'68]**

QR Iteration

- for $\mathbf{A} \in \mathbb{C}^{n\times n}$, each iteration requires $\mathcal{O}(n^3)$ to compute the QR decomposition and the matrix multiplication; too computational expensive!

Question: can we directly transform $\mathbf{A}$ into an upper triangular matrix (i.e., introducing zeros below the diagonal) based on unitary similarity transformations?

- a naive try via Householder reflections: let $\mathbf{Q}_1 = \mathbf{I} - \frac{2}{\|\mathbf{v}_1\|_2^2}\mathbf{v}_1\mathbf{v}_1^H$ with $\mathbf{v}_1 = \mathbf{a}_1 + \text{sign}(a_{11})\|\mathbf{a}_1\|_2\mathbf{e}_1$ be the reflection matrix that reflects $\mathbf{a}_1$ to $-\text{sign}(a_{11})\|\mathbf{a}_1\|_2\mathbf{e}_1$

$$\mathbf{A} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix}}_{\mathbf{Q}_1\mathbf{A}=\mathbf{Q}_1^H\mathbf{A}} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}_{\mathbf{Q}_1\mathbf{A}\mathbf{Q}_1=\mathbf{Q}_1^H\mathbf{A}\mathbf{Q}_1}$$

- Mission failed!

QR Iteration

- Power iteration with deflation: For a general $\mathbf{A}$ with $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$, and
$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}.$$

- once the eigenvector $\mathbf{v}_1$ and λ_1 is found via power iteration, deflation can be carried out by constructing a Householder reflection $\mathbf{Q}_1$ so that $\mathbf{Q}_1\mathbf{v}_1 = \mathbf{e}_1$, and then $\mathbf{Q}_1\mathbf{A}\mathbf{Q}_1$ is a matrix with block upper-triangular structure
$$\mathbf{A} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \to \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix}}_{\mathbf{Q}_1\mathbf{A}} \to \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix}}_{\mathbf{Q}_1\mathbf{A}\mathbf{Q}_1}$$

- This decouples the problem of computing the eigenvalues of $\mathbf{A}$ into the (solved) problem of computing λ_1, and computing the remaining eigenvalues by carrying out power iteration on the lower right $(n-1) \times (n-1)$ submatrix.

Property 6. Any $\mathbf{A} \in \mathbb{C}^{n\times n}$ is unitarily similar to an upper Hessenberg matrix $\mathbf{H}$ (i.e., introducing zeros below the first subdiagonal), i.e., $\mathbf{Q}^H\mathbf{A}\mathbf{Q} = \mathbf{H}$.

Hessenberg Reduction

- an upper Hessenberg matrix is given as

$$\mathbf{H} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix}$$

- Hessenberg reduction via Householder reflections: let $\tilde{\mathbf{a}}_1 = \mathbf{A}(2:n,1)$ and $\mathbf{Q}_1$ be the Householder reflection matrix that reflects $\tilde{\mathbf{a}}_1$ to $-\text{sign}(\tilde{a}_{11})\|\tilde{\mathbf{a}}_1\|_2\mathbf{e}_1$

$$\mathbf{A} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \to \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix}}_{\mathbf{Q}_1^H\mathbf{A}} \to \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix}}_{\mathbf{Q}_1^H\mathbf{A}\mathbf{Q}_1}$$

- repeat the above procedure: $\underbrace{(\mathbf{Q}_1 \cdots \mathbf{Q}_{n-2})^H}_{\mathbf{Q}^H} \mathbf{A} \underbrace{\mathbf{Q}_1 \cdots \mathbf{Q}_{n-2}}_{\mathbf{Q}} = \mathbf{H}$

Hessenberg Reduction

- for any $\mathbf{A} \in \mathbb{C}^{n \times n}$, the following algorithm reduces $\mathbf{A}$ to be upper Hessenberg

```
**Algorithm:** Householder Reduction to Upper-Hessenberg Form
**input:** $\mathbf{A} \in \mathbb{C}^{n \times n}$
**for** $k = 1 : n-2$
    $\mathbf{x} = \mathbf{A}(k+1:n, k)$
    $\mathbf{v}_k = \mathbf{x} + \text{sign}(x_1)\|\mathbf{x}\|_2\mathbf{e}_1$
    $\mathbf{v}_k = \mathbf{v}_k / \|\mathbf{v}_k\|_2$
    $\mathbf{A}(k+1:n, k:n) = \mathbf{A}(k+1:n, k:n) - 2\mathbf{v}_k(\mathbf{v}_k^H\mathbf{A}(k+1:n, k:n))$
    $\mathbf{A}(1:n, k+1:n) = \mathbf{A}(1:n, k+1:n) - 2(\mathbf{A}(1:n, k+1:n)\mathbf{v}_k)\mathbf{v}_k^H$
**end**
**output:** $\mathbf{A}$
```

- compexity: $\frac{10}{3}n^3 + \mathcal{O}(n^2)$

Property 7. Any $\mathbf{A} \in \mathbb{H}^n$ is unitarily similar to a tridiagonal matrix.

- for $\mathbf{A} \in \mathbb{H}^n$, the above algorithm reduces $\mathbf{A}$ to be tridiagonal (i.e., Householder Reduction to Tridiagonal Form)

Hessenberg QR Iteration

```
Algorithm: Hessenberg QR Iteration
input: $\mathbf{A} \in \mathbb{C}^{n \times n}$
$\mathbf{H} = \mathbf{Q}^H \mathbf{A} \mathbf{Q}$, $\mathbf{A}^{(0)} = \mathbf{H}$        % Hessenberg reduction for $\mathbf{A}$
$k = 0$
repeat
    $\mathbf{Q}^{(k+1)} \mathbf{R}^{(k+1)} = \mathbf{A}^{(k)}$        % perform QR for $\mathbf{A}^{(k)}$
    $\mathbf{A}^{(k+1)} = \mathbf{R}^{(k+1)} \mathbf{Q}^{(k+1)}$
    $k := k + 1$
until a stopping rule is satisfied
output: $\mathbf{A}^{(k)}$
```

- **Fact:** $\mathbf{A}^{(k)}$ preserves the upper-Hessenberg property over iterations
  - Proof: $\mathbf{A}^{(k)}$ is upper Hessenberg and $\mathbf{R}^{(k+1)}$ is upper triangular, then $\mathbf{Q}^{(k+1)}$ is upper Hessenberg; since $\mathbf{R}^{(k+1)}$ is upper triangular, $\mathbf{A}^{(k+1)} = \mathbf{R}^{(k+1)} \mathbf{Q}^{(k+1)}$ is upper Hessenberg
  - for $\mathbf{A} \in \mathbb{H}^n$, $\mathbf{A}^{(k)}$ preserves the tridiagonal property over iterations

Hessenberg QR Iteration

- for $\mathbf{A} \in \mathbb{C}^{n\times n}$, QR decomposition step for $\mathbf{A}^{(k)}$ and the matrix multiplication step requires $\mathcal{O}(n^2)$
  - using Givens rotations to compute $\mathbf{A}^{(k+1)} = \left(\mathbf{Q}^{(k+1)}\right)^H \mathbf{A}^{(k)}\mathbf{Q}^{(k+1)}$

$$\mathbf{A}^{(k)} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix}}_{\mathbf{G}_1^H\mathbf{A}} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & \times & \times \end{bmatrix}}_{\mathbf{G}_2^H\mathbf{G}_1^H\mathbf{A}} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{bmatrix}}_{\mathbf{G}_3^H\mathbf{G}_2^H\mathbf{G}_1^H\mathbf{A}=\mathbf{R}^{(k+1)}}$$

$$\mathbf{R}^{(k+1)} = \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{bmatrix} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{bmatrix}}_{\mathbf{R}^{(k+1)}\mathbf{G}_1} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & & \times \end{bmatrix}}_{\mathbf{R}^{(k+1)}\mathbf{G}_1\mathbf{G}_2} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix}}_{\mathbf{R}^{(k+1)}\mathbf{G}_1\mathbf{G}_2\mathbf{G}_3=\mathbf{A}^{(k+1)}}$$

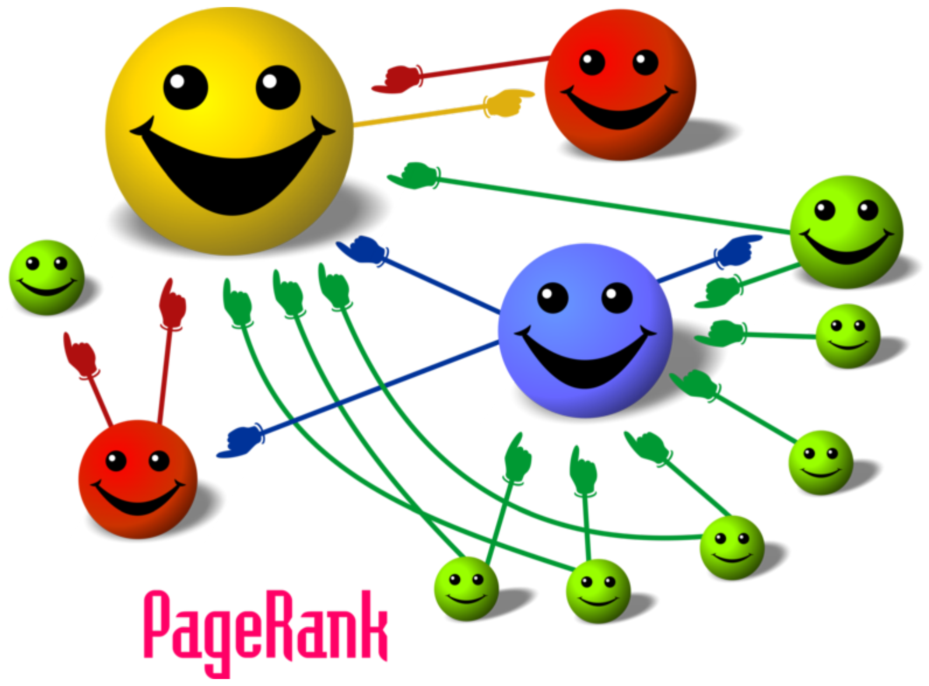
- for $\mathbf{A} \in \mathbb{H}^n$, QR factorization and the matrix multiplication requires $\mathcal{O}(n)$ flops

Hessenberg QR Iteration With Shift

```
Algorithm: Hessenberg QR Iteration With Shift
input: $\mathbf{A} \in \mathbb{C}^{n\times n}$
$\mathbf{H} = \mathbf{Q}^H\mathbf{A}\mathbf{Q}$, $\mathbf{A}^{(0)} = \mathbf{H}$        % Hessenberg reduction for $\mathbf{A}$
$k = 0$
repeat
    choose a shift $\mu^{(k)}$
    $\mathbf{Q}^{(k+1)}\mathbf{R}^{(k+1)} = \mathbf{A}^{(k)} - \mu^{(k)}\mathbf{I}$        % perform QR for $\mathbf{A}^{(k)} - \mu^{(k)}\mathbf{I}$
    $\mathbf{A}^{(k+1)} = \mathbf{R}^{(k+1)}\mathbf{Q}^{(k+1)} + \mu^{(k)}\mathbf{I}$
    $k := k + 1$
until a stopping rule is satisfied
output: $\mathbf{A}^{(k)}$
```

PageRank: A Case Study

- PageRank is an algorithm used by Google to rank the pages of a search result.
- the idea is to use counts of links of various pages to determine pages' importance.



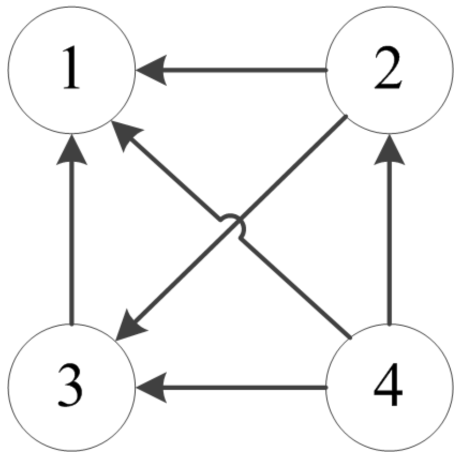
Source: Wiki.

- further reading: **[Bryan-Tanya2006]**

PageRank Model

- Model:
$$\sum_{j \in \mathcal{L}_i} \frac{v_j}{c_j} = v_i, \quad i = 1, \ldots, n,$$
where c_j is the number of outgoing links from page j; $\mathcal{L}_i$ is the set of pages with a link to page i; v_i is the importance score of page i.

- example:



$$\overbrace{\begin{bmatrix} 0 & \frac{1}{2} & 1 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix}}^{\mathbf{A}} \overbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}}^{\mathbf{v}} = \overbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}}^{\mathbf{v}}.$$

- $\mathbf{A}$ is a "weighted adjacency matrix" that contains the structure of the network.
- can be solved as a sparse linear system

PageRank Problem

- let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a matrix such that $a_{ij} = 1/c_j$ if $j \in \mathcal{L}_i$ and $a_{ij} = 0$ if $j \notin \mathcal{L}_i$
- Problem: find a non-negative $\mathbf{v}$ such that $\mathbf{Av} = \mathbf{v}$
  - $\mathbf{A}$ is extremely large and sparse, and we want to use the power method
- Questions:
  - does a solution to $\mathbf{Av} = \mathbf{v}$ exist? Or, is $\lambda = 1$ an eigenvalue of $\mathbf{A}$?
  - does $\mathbf{Av} = \mathbf{v}$ have a non-negative solution? Or, does a non-negative eigenvector associated with $\lambda = 1$ exist?
  - is the solution to $\mathbf{Av} = \mathbf{v}$ unique? Or, would there exist more than one eigenvector associated with $\lambda = 1$?
    - a unique solution is desired for this problem
  - is $\lambda = 1$ the only eigenvalue that is the largest in modulus?
    - this is required for the power method

Some Notations and Conventions

- notation:
  - $\mathbf{x} \geq \mathbf{y}$ means that $x_i \geq y_i$ for all i
  - $\mathbf{x} > \mathbf{y}$ means that $x_i > y_i$ for all i
  - $\mathbf{x} \ngeq \mathbf{y}$ means that $\mathbf{x} \geq \mathbf{y}$ does not hold
  - the same notations apply to matrices

- conventions:
  - $\mathbf{x}$ is said to be non-negative if $\mathbf{x} \geq \mathbf{0}$, and non-positive if $-\mathbf{x} \geq \mathbf{0}$
  - $\mathbf{x}$ is said to be positive if $\mathbf{x} > \mathbf{0}$, and negative if $-\mathbf{x} > \mathbf{0}$
  - the same conventions apply to matrices
  - a square $\mathbf{A}$ is said to be column-stochastic (or a Markov matrix) if $\mathbf{A} \geq \mathbf{0}$ and $\mathbf{A}^T\mathbf{1} = \mathbf{1}$
    - $*$ a column-stochastic $\mathbf{A}$ has every column $\mathbf{a}_i$ satisfying $\mathbf{a}_i^T\mathbf{1} = \sum_{j=1}^{n} a_{ji} = 1$

PageRank Matrix Properties

- in PageRank, $\mathbf{A}$ is column-stochastic if all pages have outgoing links
  - see the literature to see how to deal with cases where some pages do not have outgoing links (dangling nodes)

Property 8. Let $\mathbf{A}$ be column-stochastic. Then,

1. $\lambda = 1$ is an eigenvalue of $\mathbf{A}$
2. $|\lambda| \leq 1$ for any eigenvalue λ of $\mathbf{A}$

- Implications:
  - a solution to $\mathbf{A}\mathbf{v} = \mathbf{v}$ does exist, though it doesn't say if $\mathbf{v} \geq \mathbf{0}$ or not
  - $\lambda = 1$ is an eigenvalue that has the largest modulus, but we don't know if it is the *only* eigenvalue that has the largest modulus

- we resort to non-negative matrix theory to answer the rest of the questions

Non-Negative Matrix Theory

Theorem 8 (Perron-Frobenius Theorem)**.** Let $\mathbf{A}$ be square positive. There exists an eigenvalue ρ of $\mathbf{A}$ such that

1. ρ is real and $\rho > 0$
2. $|\lambda| < \rho$ for any eigenvalue λ of $\mathbf{A}$ with $\lambda \neq \rho$
3. there exists a positive eigenvector (and also left eigenvector) associated with ρ
4. the algebraic multiplicity of ρ is 1 (so the geometric multiplicity of ρ is also 1)

A weaker result for general non-negative matrices:

Theorem 9. Let $\mathbf{A}$ be square non-negative. There exists an eigenvalue ρ of $\mathbf{A}$ such that

1. ρ is real and $\rho \geq 0$
2. $|\lambda| \leq \rho$ for any eigenvalue λ of $\mathbf{A}$
3. there exists a non-negative eigenvector (and left eigenvector) associated with ρ

PageRank Matrix Properties

- further implication by Theorem 9:
  - a non-negative solution to $\mathbf{A}\mathbf{v} = \mathbf{v}$ exists, though it doesn't say if there exists another solution
  - even worse, it is not known if there exists another solution $\mathbf{v}$ such that $\mathbf{v} \ngeq \mathbf{0}$

PageRank Matrix Properties

- PageRank actually considers a modified version of $\mathbf{A}$

$$\tilde{\mathbf{A}} = (1-\beta)\mathbf{A} + \beta \begin{bmatrix} 1/n & \dots & 1/n \\ \vdots & & \vdots \\ 1/n & \dots & 1/n \end{bmatrix}$$

  where $0 < \beta < 1$ (typical value is $\beta = 0.15$)

- $\tilde{\mathbf{A}}$ is positive
- further implications by Theorem 8:
  - $\lambda = 1$ is the *only* eigenvalue that has the largest modulus
  - there exists *only* one eigenvector associated with $\lambda = 1$; that eigenvector is either positive or negative
  - so the power method should work

Generalized Eigenvalue Problem

Problem: given a $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n\times n}$, find a vector $\mathbf{v} \in \mathbb{C}^n$ with $\mathbf{v} \neq \mathbf{0}$ such that

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{B}\mathbf{v}, \qquad \text{for some } \lambda \in \mathbb{C} \tag{$*$}$$

or, equivalently,

$$\begin{aligned} \text{find} \quad & \mathbf{v} \in \mathbb{C}^n \backslash \{\mathbf{0}\} \text{ and } \lambda \in \mathbb{C} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{v} = \lambda \mathbf{B}\mathbf{v} \end{aligned}$$

- $(*)$ is called a generalized eigenvalue problem for matrix pair (matrix pencil) $(\mathbf{A}, \mathbf{B})$
- let $(\mathbf{v}, \lambda)$ be a solution to $(*)$. We call
  - $(\mathbf{v}, \lambda)$ an generalized eigen-pair of $(\mathbf{A}, \mathbf{B})$
  - λ an generalized eigenvalue of $(\mathbf{A}, \mathbf{B})$; $\mathbf{v}$ an generalized eigenvector of $(\mathbf{A}, \mathbf{B})$ associated with λ

References

[Golub-Van Loan'13] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th edition, JHU Press, 2013.

[Bryan-Tanya2006] K. Bryan and L. Tanya, "The $25,000,000,000$ eigenvector: The linear algebra behind Google," *SIAM Review*, vol. 48, no. 3, pp. 569–581, 2006.

[Wilkinson'68] J. H. Wilkinson, "Global convergence of tridiagonal QR algorithm with origin shifts," *Linear Algebra and its Applications*, vol. 1, no. 3, pp. 409–420, 1968.