Numerical Methods in Linear Algebra

Dénes Sexty

University of Graz

2020-2021, winter term

UNIVERSITÄT GRAZ
UNIVERSITY OF GRAZ

1. Examples: Bridge, multidim Newton's method, QM, vibrational eigenmodes
2. Linear equations
3. Gauss elimination, Pivoting
4. Numerical errors, condition number
5. LU decomposition, iterative improvement
6. Householder reduction
7. iterative solution: Gauss-Seidel, Successive overrelaxation (SOR), Conjugate gradient (CG) and others
8. Eigenvalues Subspace methods, power iteration, Lánczos method, Krylov Schur
9. Diagonalization with Fourier transformation
10. Eigenvalues transformation method

We want to get to the conjugate gradient method, let's first discuss the Gradient method (a.k.a Steepest descent, Richardson's method)
Take

$$N = \frac{1}{\alpha}, \quad P = \frac{1}{\alpha} - A \tag{1}$$

with $\alpha > 0$. This leads to

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha(\mathbf{b} - A\mathbf{x}_k) \tag{2}$$

This converges for $0 < \alpha < 2/\lambda_{max}(A)$ for a positive definite A and might not converge for a non positive definite A

Suppose A is symmetric (hermitic) and positive definit ($\forall \mathbf{x}: \ \mathbf{x}^T A\mathbf{x} > 0$) define

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b} \tag{3}$$

Since A is positive definit, this has a minimum. There we have $\nabla f(\mathbf{x}) = 0$

$$\nabla f(\mathbf{x}) = A\mathbf{x} - \mathbf{b} = 0 \tag{4}$$

Minumum of $f(\mathbf{x})$ solves the eq. $A\mathbf{x} = \mathbf{b}$

The iteration

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha(\mathbf{b} - A\mathbf{x}_k) \tag{5}$$

steps along the negative gradient of $f(\mathbf{x})$.
If α is small enough ($0 < \alpha < 2/\rho(A)$) than $f(\mathbf{x})$ decreases $\rightarrow$ we might get to the minimum

if we take small steps it's going to take long to get to the minimum.
Idea: choose α such that $f(\mathbf{x} + \alpha(\mathbf{b} - A\mathbf{x}))$ is minimized $\rightarrow$ optimal stepsize

$$f(\mathbf{x} + \alpha\mathbf{r}) = \frac{1}{2}(\mathbf{x} + \alpha\mathbf{r})^T A(\mathbf{x} + \alpha\mathbf{r}) - b^T(x + \alpha\mathbf{r}) \tag{6}$$

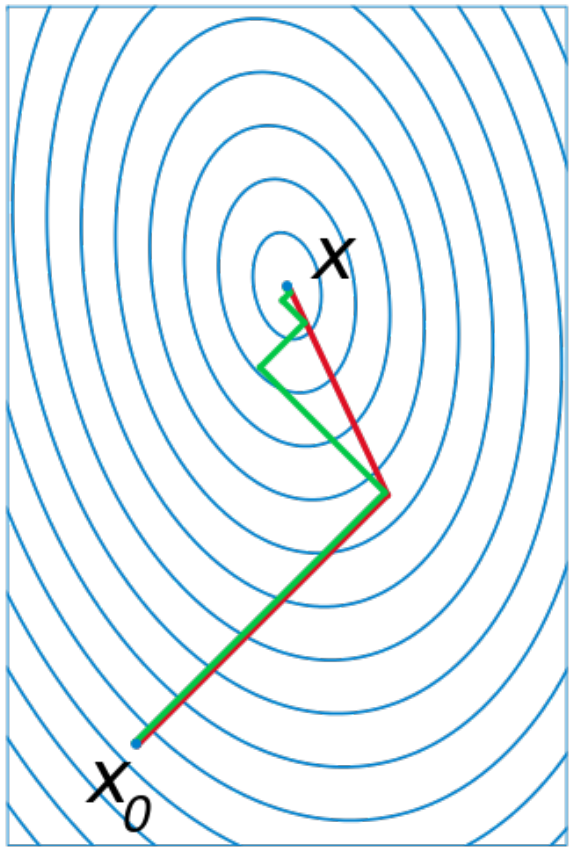
We get minimum from $\partial_\alpha f(\mathbf{x} + \alpha\mathbf{r}) = 0$

$$\implies \alpha = \frac{r^T r}{r^T A r} \tag{7}$$

Green: Gradient method with optimal stepsize

Can still take many steps to converge

Teaser: Red corresponds to Conjugate Gradient, which converges in two steps (in 2d)



Some definitions first:
The vector space spanned by $\{\mathbf{r}, A\mathbf{r}, \ldots, A^k\mathbf{r}\}$ is the **Krylov subspace** corresponding to A and $\mathbf{r}$, denoted by K_k
k_0 is the **Krylov critical dimension** if $K_{k_0} = K_{k_0+1}$

two vectors u and v are **conjugate** with respect to A if we have $u^T Av = 0$.

Consider a gradient method with whatever stepsize

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k(\mathbf{b} - A\mathbf{x}_k) = \mathbf{x}_k + \alpha_k\mathbf{r}_k \tag{8}$$

Then $\mathbf{r}_k \in K_k$ and $\mathbf{x}_{k+1} - \mathbf{x}_0 \in K_k$
Shown by $\mathbf{b} - A\mathbf{x}_{k+1} = \mathbf{b} - A\mathbf{x}_k - \alpha A\mathbf{r}_k \implies \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha A\mathbf{r}_k$

The conjugate gradient method (CG) searches for the minumum of $f(\mathbf{x}_{k+1})$ in the affine subspace $[\mathbf{x}_0 + K_k]$ (this means $\mathbf{x}_{k+1} - \mathbf{x}_0 \in K_k$), where K_k is the krylov subspace of $\mathbf{r}_0$

The Conjugate Gradient satisfies:

1. $\mathbf{x}_{k+1}$ is choosen such that $r_{k+1} \perp K_k$
2. $\mathbf{x}_{k+1}$ is choosen such that it minimizes $f(\mathbf{x})$ in $[\mathbf{x}_0 + K_k]$

This means that CG converges in at most n iterations. (But usually one needs less for a good solution)

The Algorithm for finding that minimum is:

$$\begin{aligned}
&\texttt{Initialization:} \quad \mathbf{p}_0 = \mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0 \\
&\texttt{Loop:} \quad \alpha_i = \mathbf{r}_i^T \mathbf{r}_i / (\mathbf{p_i}^T A \mathbf{p_i}) \\
&\qquad \mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i \\
&\qquad \mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i A \mathbf{p}_i \\
&\qquad \beta_i = \mathbf{r}_{i+1}^T \mathbf{r}_{i+1} / (\mathbf{r}_i^T \mathbf{r}_i) \\
&\qquad \mathbf{p}_{i+1} = \mathbf{r}_{i+1} + \beta_i \mathbf{p}_i
\end{aligned}$$

one can show by induction:

$$\forall i \neq j : \mathbf{r}_i^T \mathbf{r}_j = 0, \quad \mathbf{p}_i^T A \mathbf{p}_j = 0, \quad \mathbf{r}_i \mathbf{p}_j = 0 \tag{9}$$

Also, α_i is choosen such that $f(\mathbf{x}_i + \alpha_i \mathbf{p}_i)$ is minimized in α_i.
$\mathbf{p}$ is sometimes called the "search direction"

since $\forall i \neq j : \mathbf{r}_i^T \mathbf{r}_j = 0$, the CG converges in at most n iterations in an n dimensional vectors space (less
However, rounding errors lead to $\mathbf{r}_{n+1} \neq 0$, so we should use a small ϵ and stop when

$$\frac{\|\mathbf{r}_k\|}{\|\mathbf{r}_0\|} < \epsilon \tag{10}$$

for large systems, n and $k_0 \sim 10^4$ or larger, typically one needs much less than n iterations.

Costs: n iterations maximally.
One iteration: matrix-vector product: $O(n^2)$, linear combinations of vectors $O(n)$
Total: $\approx n^3$ slightly worse than Gauss or LU decomp.
But: usually needs less than n iterations.
Sparse matrices: multiplication costs only $O(n)$ operations

One can show: The norm of $\mathbf{e}_i = \mathbf{x}_i - \mathbf{x}$ is reduced in each iteration by the factor at least

$$\frac{\sqrt{\text{cond}_2 A} - 1}{\sqrt{\text{cond}_2 A} + 1} \tag{11}$$

Preconditioning

given a matrix M such that:

$M^{-1}\mathbf{v}$ is easy to calculate

$\text{cond}_2(M^{-1}A) < \text{cond}_2(A)$

We can solve the **preconditioned system** $M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}$ instead of $A\mathbf{x} = \mathbf{b}$, converges faster.

`Initialization:` $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0, \mathbf{z}_0 = M^{-1}\mathbf{r}_0, \mathbf{p}_0 = \mathbf{z}_0$

`Loop:`
$\alpha_i = \mathbf{z}_i^T \mathbf{r}_i / (\mathbf{p_i}^T A \mathbf{p_i})$
$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i$
$\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i A \mathbf{p}_i$
$\mathbf{z}_{i+1} = M^{-1}\mathbf{r}_{i+1}$
$\beta_i = \mathbf{z}_{i+1}^T \mathbf{r}_{i+1} / (\mathbf{z}_i^T \mathbf{r}_i)$
$\mathbf{p}_{i+1} = \mathbf{z}_{i+1} + \beta_i \mathbf{p}_i$

What happens if we have A non-symmetric positive definite?

Use $B = AA^+$: $\mathbf{v}^+ B\mathbf{v} = \mathbf{v}^+ AA^+\mathbf{v} = \left\|A^+\mathbf{v}\right\| > 0$ if A is non-singular
solve $B\mathbf{y} = \mathbf{b}$ using CG, this means $AA^+\mathbf{y} = \mathbf{b}$
$\implies$ $\mathbf{x} = A^+\mathbf{y}$ solves $A\mathbf{x} = \mathbf{b}$

Further iterative methods:
BiConjugate Gradient (BiCG, BiCGStab), Generalized minimal residual (GMRES): A does not need to be positive def, not guaranteed to converge, but often does.
Block-CG: we want to solve $A\mathbf{x}_k = \mathbf{b}_k$ for several right hand side $\mathbf{b}$ vectors. We do the iterations the same time where the search directions for different k "communicate", such that the we look for a minimum on a larger Krylov space, ensuring faster convergence
Multishift-CG: we want to solve $(A + m_i\mathbf{1})\mathbf{x}_i = \mathbf{b}_i$. These ahve all the same Krylov space. One can solve all of them in the same iteration with just one matrix-vector multiplication and many linear combinations of vectors.

So far we had $A\mathbf{x} = \mathbf{b}$, now we look at $A\mathbf{x} = 0$
General eigenvalue problem:

$$A(\lambda)\mathbf{x} = 0 \tag{12}$$

solutions at $\det(A(\lambda)) = 0$, giving eigen values λ_i and eigenvectors $\mathbf{x}_i$.

Regular eigenvalue problem: $A(\lambda) = A - \lambda\mathbf{1}$, this gives the usual form:

$$A\mathbf{x} = \lambda\mathbf{x} \tag{13}$$

$\det(A - \lambda\mathbf{1}) =$ polynomial of degree $n \implies \lambda \in \{\lambda_1, \ldots, \lambda_n\}$

Eigenvectors are determined up to a factor $\implies$ choose x_1 at will and solve for x_j, $j > 1$ using methods detailed before.

In practice looking for the roots of a large polynomial is hard (no direct method for $n > 4$, even calculating the coefficients of the polynomial is hard for large n)

1. Subspace methods
   Aim at finding a few eigenvalues with high precision by keeping track of a subspace using a few vectors, and iteratively improving the precision.
2. Transformation methods
   If we find Q such that

$$Q^{-1}AQ = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \tag{14}$$

   than λ_i are the eigenvectors,
   $AQ = Q\mathrm{diag}(\lambda_1, \ldots, \lambda_n) \implies$ columns of Q are the eigenvectors
   Usually Q is built iteratively $Q = Q_1 Q_2 \ldots Q_n \ldots$, such that the non diagonal elements of the transformed A decrease
3. Fourier transformation Works in some cases: for a certain class of matrices: $M_{j,k} = \sum_i \alpha_i \delta_{j,k+n_i}$, using periodic boundary conditions
   These types of matrices are often come up in physics, discretization of PDEs, etc.

Consider the matrix

$$A = \begin{pmatrix} 0 & \dots & \dots & 0 & \epsilon \\ 1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix} \tag{15}$$

Let's choose $\epsilon = 10^{-n}$, with n even. The characteristic polinomial for this is:

$$p_A(\lambda) = \det(A - \lambda \mathbf{1}) = (\lambda^n - \epsilon) \tag{16}$$

$\implies \lambda_i = 10^{-1}$ All eigenvalues are the same.
However, if we take $\epsilon = 0$ we have $\lambda_i = 0$
$\implies$ in some cases small variations of the matrix give a large change in the eigenvalues.

We define the eigenvalue condition number

$$\Gamma(A) = \inf_{P^{-}1AP=\mathrm{diag}} \mathrm{cond}(P), \quad \text{with } \mathrm{cond}(P) = \|P\| \left\|P^{-1}\right\| \tag{17}$$

This means $\Gamma(A) \geq 1$
For symmetric (hermitian) matrices we have $\Gamma_2(A) = 1$, as we $\|P\|_2 = 1$ for orthogonal (unitary) P matrices

Bauer-Fike Theorem:
If we perturbe the matrix $A \to A + \delta A$, the eigenvalues change $\lambda_i \to \lambda_i'$
We have an upper estimate of their change:

$$|\lambda_i - \lambda_i'| \leq \Gamma(A) \, \|\delta A\| \tag{18}$$

For a norm which satisfies $\|\mathrm{diag}(d_1, \ldots, d_n)\| = \max|d_i|$

aka. von Mises Method – simplest subspace method
Suppose we have a symmetric (Hermitian) matrix, such that the eigenvalues are non degenerate (we really need the largest absolute value is non-degenerate)

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n| \tag{19}$$

$\Longrightarrow$ any vector can be written as a linear combination of the eigenvectors $\mathbf{x}_i$

$$\mathbf{v} = \sum \alpha_i \mathbf{x}_i \tag{20}$$

We can start applying A: $\mathbf{v}_0 = v$, $\mathbf{v}_1 = A\mathbf{v}_0$, and so on: $\mathbf{v}_k = A\mathbf{v}_{k-1} = A^k\mathbf{v}_0$.
We can see:

$$\mathbf{v}_1 = \sum \alpha_i \lambda_i \mathbf{x}_i \tag{21}$$
$$\mathbf{v}_k = \sum \alpha_i \lambda_i^k \mathbf{x}_i$$

Our assumption means that at large k the first eigenvector will dominate

Use the following iteration:

$$\mathbf{v}_0 = \mathbf{v}/\left\|\mathbf{v}\right\| \tag{22}$$
$$\mathbf{v}_{k+1} = A\mathbf{v}_k/\left\|A\mathbf{v}_k\right\|$$

We can write $\mathbf{v}_k = \alpha_i^{(k)}\mathbf{x}_i$
We can also write

$$\mathbf{v}_k = A^k\mathbf{v}_0/\left\|A^k\mathbf{v}_0\right\| = \frac{\frac{1}{\lambda_1^k}A^k\mathbf{v}_0}{\left\|\frac{1}{\lambda_1^k}A^k\mathbf{v}_0\right\|} = \frac{\alpha_1\mathbf{x}_1 + \sum_{i=2}^{n}\left(\frac{\lambda_i}{\lambda_1}\right)^k\mathbf{x}_i}{\left\|\alpha_1\mathbf{x}_1 + \sum_{i=2}^{n}\left(\frac{\lambda_i}{\lambda_1}\right)^k\mathbf{x}_i\right\|} \tag{23}$$

$\implies$ convergence rate given by $|\lambda_2|/|\lambda_1|$
$\mathbf{v}_k$ converges to the eigenvector x_1
Using $\mathbf{d}_k = \mathbf{x}_k - \mathbf{x}_{k-1}$

$$A\mathbf{d}_k = A\mathbf{x}_k - A\mathbf{x}_{k-1} = A\mathbf{x}_k - \lambda\mathbf{x}_k \tag{24}$$

for some λ (as we calculate $\mathbf{x}_k$ by normalizing $A\mathbf{x}_{k-1}$
$\implies$ $\|A\mathbf{d}_k\|$ helps judge how close we are to an eigenvalue.
We can get the eigenvalue by: $\mathbf{e}_k^T A\mathbf{v}/(\mathbf{e}_k^T\mathbf{v})$ (e.g. picking a component of a vector, preferably a large one)

Sometimes we need the smallest eigenvale:

$$\begin{aligned} \mathbf{v}_0 &= \mathbf{v} / \|\mathbf{v}\| \\ \text{solve} \quad A\mathbf{y_k} &= \mathbf{v}_{k-1} \text{ for } \mathbf{y}_k \\ \mathbf{v}_{k+1} &= \mathbf{y}_k / \|\mathbf{y}_k\| \end{aligned} \tag{25}$$

Is equivalent to power iteration with A^{-1} , which has eigenvalues $\{1/\lambda_1, \ldots, 1/\lambda_n\}$

$$\left|\frac{1}{\lambda_1}\right| < \left|\frac{1}{\lambda_2}\right| < \ldots < \left|\frac{1}{\lambda_n}\right| \tag{26}$$

$\Longrightarrow$ power iteration will give the smallest eigenvalue, $1/\lambda_n$
Can be efficiently carried out by calculating the LU decomposition of A

If we know the largest eigenvalue λ_1 and eigenvector $\mathbf{x}_1$
make our vector orthogonal to that:

$$\mathbf{v}_0 = \mathbf{v} / \|\mathbf{v}\| \tag{27}$$

$$\mathbf{y_k} = A\mathbf{v}_{k-1}$$

$$\mathbf{v}_{k+1} = \mathbf{y}_k - \mathbf{x}_1(\mathbf{x}_1^T \mathbf{y}_k) \tag{28}$$

$\Longrightarrow$ This converges to the eigenvector of the second largest eigenvalue.

If we know there is an eigenvalue close to μ:

$$|\lambda_i - \mu| < |\lambda_j - \mu| \text{ for } j \neq i \tag{29}$$

We can use power iteration of $(A - \mu\mathbf{1})^{-1}$:

$$\mathbf{v}_0 = \mathbf{v} / \|\mathbf{v}\| \tag{30}$$

$$\text{solve} \quad (A - \mu\mathbf{1})\mathbf{y_k} = \mathbf{v}_{k-1} \text{ for } \mathbf{y}_k$$

$$\mathbf{v}_{k+1} = \mathbf{y}_k / \|\mathbf{y}_k\|$$

$\Longrightarrow$ This converges to $\mathbf{x}_i$ $\Longrightarrow$ λ_i is found

What if we need more than a few eigenvectors? We should not throw away all but the last vector in the krylov space

$$K_k = \{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, \ldots, A^k\mathbf{v}\} \tag{31}$$

Build the following base of the Krylov space (with A symmetric (hermitian) matrix):

$$\mathbf{v}_1 = \frac{\mathbf{v}}{\|\mathbf{v}\|} \tag{32}$$

$$\mathbf{v}_k = \frac{\mathbf{y_k}}{\|\mathbf{y_k}\|}, \quad with \quad \mathbf{y}_k = A\mathbf{v}_{k-1} - \sum_{i=1}^{k-1}((A\mathbf{v}_{k-1})^+\mathbf{v}_i)\mathbf{v}_i$$

In words: take the new vector $A\mathbf{v}_{k-1}$, make it orthogonal to previous vectors and normalize it.
This is equivalent to using the Gram-Schmidt procedure on the Krylov space if $\mathbf{v}_k$ is zero for some k, than we have exhausted the Krylov space of the vector $\mathbf{v}$ and the the iteration is stopped.

$$\mathbf{y}_k = A\mathbf{v}_{k-1} - \sum_{i=1}^{k-1}((A\mathbf{v}_{k-1})^+\mathbf{v}_i)\mathbf{v}_i \tag{33}$$

if A is symmetric we can write

$$((A\mathbf{v}_{k-1})^+\mathbf{v}_j) = (\mathbf{v}_{k-1}^+ A\mathbf{v}_j) = \underbrace{(\mathbf{v}_{k-1}^+\mathbf{y}_{j+1})}_{=0 \text{ if } j<k-2} + \sum_{i=1}^{j}((A\mathbf{v}_j)^+\mathbf{v}_i)\ \underbrace{(\mathbf{v}_{k-1}^+\mathbf{v}_i)}_{=0 \text{ for } j<k-1} \tag{34}$$

So we only need to make the new vector orthogonal for the last two vectors

$$\mathbf{y}_k = A\mathbf{v}_{k-1} - \sum_{i=k-2}^{k-1}((A\mathbf{v}_{k-1})^+\mathbf{v}_i)\mathbf{v}_i \tag{35}$$

Rewriting the last equation we have:

$$A\mathbf{v}_{k-1} = \mathbf{v}_k \|\mathbf{y}_k\| + ((A\mathbf{v}_{k-1})^+ \mathbf{v}_{k-1})\mathbf{v}_{k-1} + \|\mathbf{y}_{k-1}\| \mathbf{v}_{k-2} \tag{36}$$

We collect k column vectors into a matrix V_k, than this equation says:

$$AV_k = V_k T_k + \mathbf{y}_{k+1} e_k^T \tag{37}$$

with $e_k^T = (0, \ldots, 0, 1)$ the row vector of length k
i.e. If we apply A to our column vectors we get a linear combination of our column vectors (the coefficients are in the T_k, which is a $k \times k$ matrix), except for the last vector, which has an extra contribution that goes into the last column
This also means if we multiply with V_k^T from the left:

$$V_k^T A V_k = T_k \tag{38}$$

and $V_k^T V_k = \mathbf{1}_{k \times k}$ since V_k is built from orthonormal vectors
Looking at the coefficients above, we also see that T_k is tridiagonal (and symmetric)

Why is this useful?
T_k is a $k \times k$ matrix: $V_k^T A V_k = T_k$: can think of it as A restricted to the vector space given by the basis V_k
Eigenvalues of T_k are related to eigenvalues of A
One can show: For any eigenvalue λ_T of T_k there exists an eigenvalue λ_A of A such that

$$|\lambda_T - \lambda_A| \leq \|\mathbf{y}_{k+1}\| \tag{39}$$

One can show even more: if y is the eigenvector of T_k corresponding to λ_T than

$$|\lambda_T - \lambda_A| \leq \|\mathbf{y}_{k+1}\| \frac{e_k^T y}{\|y\|} \tag{40}$$

Which means we have a particularly good approximation if the last element of the eigenvector y is small

Generalization for non-hermitian: **Arnoldi method.** T_k is upper Hessenberg in this case.

Lanczos is very useful, but we have no way to control which eigenvalues we get. With large k the method becomes intractable. restarting can improve one eigenvalue, but what if we need more?

Krylov-Schur

We start with an Arnoldi iteration

$$AV_k = V_k T_k + \mathbf{y}_{k+1} e_k^T \tag{41}$$

With unitary transformations on the $\mathbf{v}_k$ and T_k we can achive a partition:

$$A(V_1 V_2) = (V_1 V_2) \begin{pmatrix} B_{11} & B_{12} \\ 0 & B_{21} \end{pmatrix} + \mathbf{u}_{k+1}(b_1^+ b_2^+) \tag{42}$$

Such that V_1 contains the eigenvectors corresponding to eigenvalues we want to calculate, V_2 contains uninteresting eigenvalues. We can now simply drop V_2 and start iterating again from

$$AV_1 = V_1 B_{11} + \mathbf{u}_{k+1} b_1^+ \tag{43}$$

We can do the expand-drop cycle until eigenvalues and eigenvectors converge
See details e.g. in: G.W. Stewart: Matrix Algorithms Vol II: Eigensystems