Projection methods for oversized linear algebra problems

Jan Brandts

September 1, 2000

Abstract

Everybody who has some experience in doing mathematics knows, that dimensional reduction and projection are useful tools to confront problems that are too complicated to solve without any simplification. Who hasn't, occasionally, but notwithstanding timidly, suggested that perhaps it would be a good idea to study the simple one-dimensional case first before trying to understand the real-world three-dimensional problem? Apparently, it is a wide-spread faith that such simplifications will not damage the essential mathematical or physical truth that is hidden in the original problem. But is this faith founded? Regardless of the answer, one should realize that in many applications there is no plausible alternative, so it would be unfair to judge too harshly on those who solve reduced problems and, with due mathematical care, formulate interesting and strong theorems and hypotheses on the full problem. Among them are the people from the field of numerical linear algebra.

I. Tosca, the rabbit and the physicist

Visiting the shadow theater *Laterna Magica* in Prague in the spring of 1998, I was truly impressed by the skillful way in which the puppeteers moved their heroes in such a way, that their shadows on the screen told me something of which I hardly doubted it was the *full* story. Sure, there was one dimension lacking, and sure, it was all in black and white, but somehow Tosca was still pretty and *il Barone* Vitellio Scarpia still collapsed when the kitchen knife hit target. Arriving back at my office at the *Matematicky Ustav* of the *Akademie Věd České Republiky*, where I was employed at that time, I sat down at my desk, mournfully contemplated on spending

one tenth of a month's salary in one evening, switched on a spotlight, and started to project images on the wall by folding my hands. Rabbit. Camel. Crocodile. All just as I had been taught many years before.

It took me a while before I realized that the shadow theater play and my own brave though unpublicized attempts to produce art of some kind, were two entirely different things. As a matter of fact, they showed *opposite* aspects of the concept of projection: *preserving the reality* as much as possible (the shadow theater), and *creating false images* (there is, after all, a non-negligible difference between a pair of hands and a rabbit). Like many things in life, it all depended on the point of view, or, as a physicist from the previous century put it, on the frame of reference. Indeed, after I moved the spotlight on my desk by a futile twenty centimeters, the shadow on the wall pretty much resembled my very own two hands, folded together in some unnatural manner, but with the recognizable curved little fingers that enable me to hit an octave plus a third on the piano.

II. Oversized problems in linear algebra

In linear algebra, two of the most frequent and important problems that are posed are the *linear system problem* and the *eigenvalue problem*. As soon as students enter university (and if we're lucky, sometimes even before), they are asked to solve the typical $Ax = b$ and $Ax = \lambda x$. Tedious (and not seldom incorrect) calculations follow, techniques like *Gaussian elimination* and finding *roots of polynomial equations* (Ah! Let the degree be not too high or a factorization obvious!) are applied, and answers (not seldom incorrect) are given. Should we tell those students that when Industry knocks on the door, it comes with matrices of size ten thousand times ten thousand? One million times one million? If we do, wouldn't it then be not more than decent to teach them about projection and dimensional reduction?

Subspaces, bases and projections

Having agreed on this, the question is, how does this all work. The answer is simpler than one might think, and the mathematics we need in order to understand the basic principles, is not much more than what we present in this small section. Let's suppose that the matrix from our linear system or eigenvalue problem has size $n \times n$, and choose a k-dimensional subspace $\mathcal{V}$ of $\mathbb{R}^n$. Typically, one should think of k as being much smaller than n. Let V be a matrix with k mutually orthonormal columns $v_1, \cdots, v_k$ spanning $\mathcal{V}$. Then every element of $\mathcal{V}$ can be written as Vu for some vector $u \in \mathbb{R}^k$. The entries of u are the *local coordinates* of Vu with respect to the basis $v_1, \cdots, v_k$ of $\mathcal{V}$. Clearly, by orthonormality of the columns of V and the fact that V^TV has as entries the Euclidean inner products $v_i^T v_j = (v_i, v_j)_{L^2} := v_i^{(1)} v_j^{(1)} + \cdots + v_i^{(n)} v_j^{(n)}$, we have that $V^TV = I$, the identity matrix.

The matrix VV^T is important too: using $V^TV = I$ we see that for all $y \in \mathbb{R}^n$, $V^T(y - VV^Ty) = 0$, and from this it follows that $z = (y - VV^Ty)$ is *orthogonal* to $\mathcal{V}$. After all, it is orthogonal to each of the columns of V:

$$V^T z = \begin{bmatrix} v_1^T \\ \hline \vdots \\ \hline v_k^T \end{bmatrix} z = \begin{bmatrix} v_1^T z \\ \vdots \\ v_k^T z \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^k. \qquad (1)$$

Since $Vu \in \mathcal{V}$ for all $u \in I\!R^k$, we have in particular that $VV^Ty \in \mathcal{V}$ for all $y \in I\!R^n$. So the relation $y - VV^Ty \perp \mathcal{V}$ tells us that VV^Ty is the *orthogonal projection of* y *on* $\mathcal{V}$. And V^Ty are its *local coordinates*. As a matter of fact, VV^Ty can be written as the more familiar looking expansion

$$VV^Ty = (v_1, y)_{L^2} v_1 + \cdots + (v_k, y)_{L^2} v_k. \tag{2}$$

We are now able to explain how to use projections in the approximation of solutions of oversized linear algebra problems.

Projecting from extra-large to medium

Concentrate on the linear system problem $Ax = b$. If we apply A to an element v of $\mathcal{V}$, then of course we cannot expect the result Av to be in $\mathcal{V}$ again. Neither can we expect b to be in $\mathcal{V}$ (unless we deliberately chose $\mathcal{V}$ that way). But both the *projection* of Av on $\mathcal{V}$ and of b on $\mathcal{V}$ are surely in $\mathcal{V}$. So, imagining ourselves in the shadow theater, we could ask ourselves what would happen if, instead of demanding Av and b to be equal, we would demand their *projections* to be equal. Then on the screen, the problem looks solved.

Mathematically, we would be trying to find an element $v \in \mathcal{V}$ such that

$$(P_{\mathcal{V}} \circ A)v = P_{\mathcal{V}} b, \tag{3}$$

where $P_{\mathcal{V}}$ is the projection on $\mathcal{V}$. Or, in matrix language, using again that every $v \in \mathcal{V}$ can be written as Vy for some y, we would be trying to find y such that

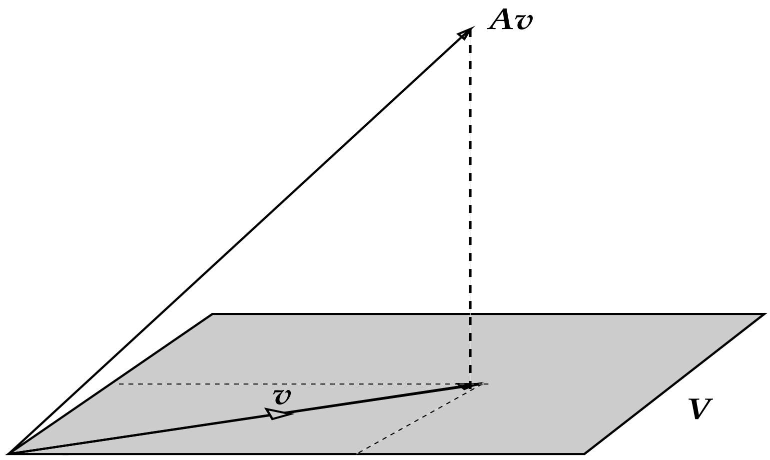
$$V^TAVy = V^Tb. \tag{4}$$

This is a linear system again, for the matrix $M := V^TAV$, which is "only" $k \times k$ big. Note that it is of this size because in (4) we imposed *equality of the local coordinates* of both projections. Now, just download a computer program to perform Gaussian elimination on this system, and it will be solved very quickly. Of course, we are not so much interested in the local coordinates y, but in the vector $v = Vy \in I\!R^n$, which is the element of $\mathcal{V}$ such that the shadow of AVy coincides with the shadow of b. Naturally, the success of this approach depends on the point of view. But, before we go into that, let's first look at the eigenvalue problem.

Extra-large eigenvalue problems

Actually, there is not much difference between projection methods for the linear system problem and the eigenvalue problem. Also in the eigenvalue problem, we can look for an element v in our subspace $\mathcal{V}$ such that the projection of Av coincides with a multiple μv of v. One thing is obviously different: μv does not need to be projected onto $\mathcal{V}$ since it is already in it to start with. Since we will demand *equality of the local coordinates* again, the *projected problem* to solve reads in this case as:

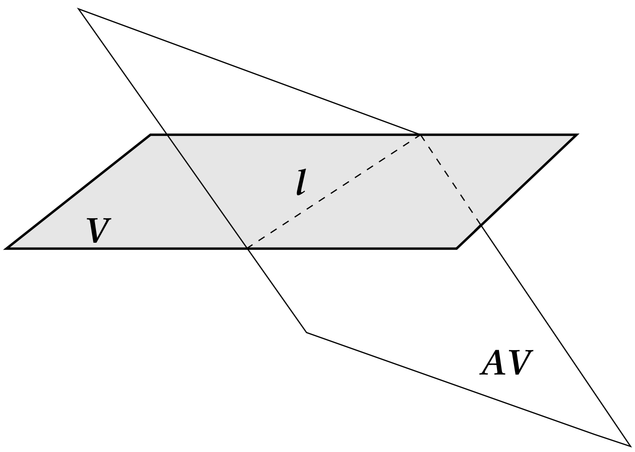
$$V^T AVy = V^T \mu Vy, \quad \text{or, equivalently,} \quad My = \mu y. \tag{5}$$



We wouldn't advice anyone to try and factorize the characteristic polynomial belonging to this eigenvalue problem of size $k \times k$. Better find some software for small to medium size eigenvalue problems, or write it yourself after reading Section V of this paper.

Equal rites

To make the situations for the linear system problem and the eigenvalue problem coincide a bit more, we will from now on assume that the subspace $\mathcal{V}$ is chosen such, that b is in it. Then also for the linear system problem, the right-hand side does not need to be projected into the subspace anymore; in fact, everything will, for both problems, only depend on how much the operator A maps the space $\mathcal{V}$ *outside of itself.*

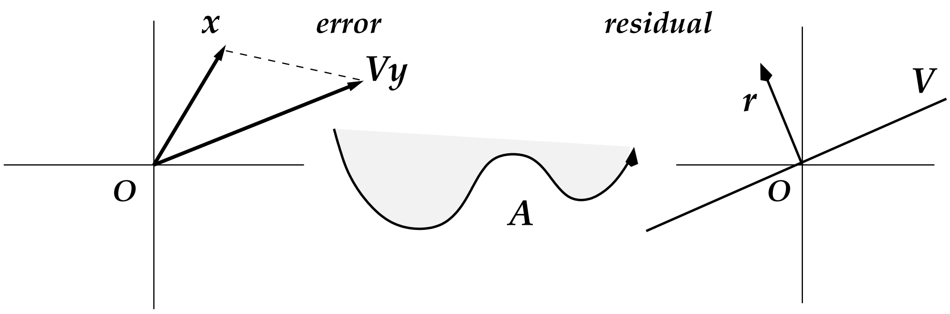


Right or wrong?

Now, look back at Equation (4). Its solution y gives a vector Vy that is in general not equal to x. To get an idea *how far away* it is from x, we define the *residual* for (4) as

$$r := b - AVy = A(x - Vy). \tag{6}$$

As we see, r is not equal to the *error* $x - Vy$, but since it is linearly related to it, we can extract useful information from it. Moreover, without knowing x, the residual can be easily computed.



Apart from that, being the difference between the object and its shadow, it is orthogonal to the screen, to $\mathcal{V}$. As a matter of fact, the projection method is *designed* to yield y, such that r is orthogonal to $\mathcal{V}$. This is exactly what is expressed by the *defining equation* (4) *for* y as $V^T r = V^T(AVy - b) = 0$. For the eigenvalue problem, all is similar. Given an eigenvalue μ of M and a corresponding eigenvector y, just define the *eigenvalue residual* as $r := AVy - \mu Vy$ and note that it can be easily computed. Also, it is orthogonal to $\mathcal{V}$ by definition. In Section VII, we learn a bit more about residuals.

Invariant subspaces, exact solutions

The ideal situation is of course the one in which A *does not* map $\mathcal{V}$ outside of itself; $\mathcal{V}$ is then called an *invariant subspace* which, for invertible A, means that $A\mathcal{V} = \mathcal{V}$. Both the linear system problem as well as (part of) the eigenvalue problem will then be solved exactly by our projection method. To see this, first note that, in matrix language, $A\mathcal{V} = \mathcal{V}$ means that there exists a small square matrix M such that $AV = VM$. Denoting the $k \times k$ matrix entries of M by m_{ij}, this relation states that

$$Av_j = m_{1j}v_1 + m_{2j}v_2 + \cdots + m_{kj}v_k, \tag{7}$$

which indeed expresses that the image under A of a basis vector v_j of $\mathcal{V}$ is a linear combination of basis vectors of $\mathcal{V}$. Assuming that $b \in \mathcal{V}$ and that $AV = VM$, it follows that $r = b - VMy \in \mathcal{V}$. Recall that r is orthogonal to $\mathcal{V}$. Then, since the only element in $\mathcal{V}$ orthogonal to $\mathcal{V}$ is the zero vector, we get $VMy = b$, or $AVy = b$, and hence $x = Vy$. Using the same arguments on (5) for the eigenvalue residual $r = AVy - \lambda Vy$, one can check that if $\mathcal{V}$ is an invariant subspace, all eigenvalues of $M = V^TAV$ are eigenvalues of A, and all eigenvectors y of M yield eigenvectors Vy of A. We leave this as the proverbial exercise to the reader.

The idea to project a problem (either infinite dimensional or almost infinite dimensional) on a relatively small subspace containing the most essential information goes back to Boris Galerkin (1871-1945), John Strutt (1842-1919, better known as Lord Rayleigh) and Walther Ritz (1878-1909).

Lord Rayleigh (1842-1919) and Walther Ritz (1878-1909)

III. Moving the screen around

The fact that reduction of dimension might really preserve the essentials of the object that is projected, is now beyond doubt. The problem that remains to be solved, is where to put the spotlight, or, as we have just seen, *where to put the screen*. In order to keep things simple, we will only consider the case in which the spotlight will project orthogonally on the screen, and then only in our usual daily-life Euclidean geometry[1]. This implies that once you've decided on the position of the screen,

[1] There do exist important methods that do not fall in this category

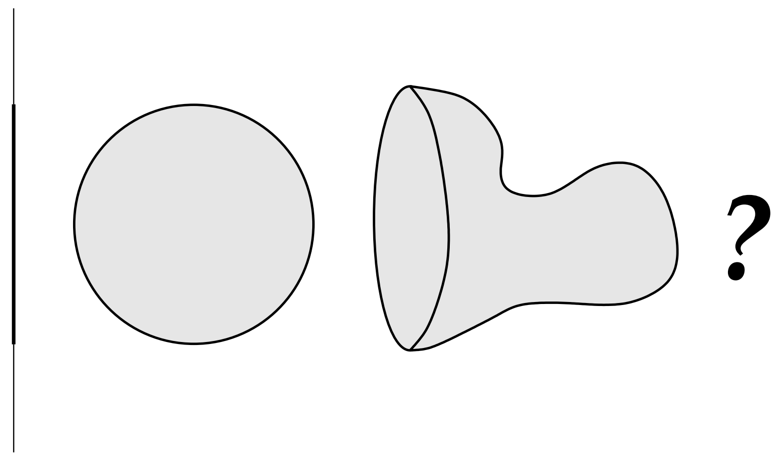
there's only one possible position of the spotlight, and vice versa.

Choosing a subspace $\mathcal{V}$ and performing the projection gives an approximation of our linear algebra problem. Computing the residual gives some idea about how good this approximation is. But what to do next, if we are not satisfied with this approximation? The obvious choice is to *reposition* the screen and the spotlight and look at the object from a different viewpoint. Of course, this repositioning should, if possible, not be at random, but based on some heuristic, or strategy. Another choice is not to reposition the screen, but to make it bigger, to expand it, to *add one or more dimensions* to it. Or, in other words, perform a projection on a space $\mathcal{W}$ of bigger dimension such that $\mathcal{V} \subset \mathcal{W}$. Intuitively, this seems to be more promising, but at the same time it also looks a bit more complicated.

Note! *Regardless of if you want to reposition the space or to expand it, the strategy should be aimed at moving towards an invariant subspace.*

A framework based on expansion

Let's concentrate on the *expansion approach*. Of course, we do not want our projected problem to become too big, so it seems like a good idea to start with a simple one-dimensional subspace. Then we have the opportunity to *strategically* expand this space $k-1$ times by one extra dimension before we arrive at a k-dimensional subspace. This will surely be better then to start off with a random subspace of dimension k, or one chosen by some vague intuition.



It will be fairly straight-forward to maintain an orthonormal basis for the subspace. Assume that V_k spans the current k-dimensional subspace $\mathcal{V}_k$, and that $V_k^T V_k = I$. Then as soon as we have decided in which direction q to expand, we compute $\hat{v} = (I - V_k V_k^T)q$ (this is nothing more than *orthogonalization* of q to V_k), normalize it to length one , $v = \hat{v}/\|\hat{v}\|$, and set $V_{k+1} := (V_k|v)$, which is the matrix V_k with extra column v. Then $V_{k+1}^T V_{k+1} = I$ while the span of V_{k+1} is still equal to the span of $(V_k|q)$. After all, by the orthogonalization, we have only removed components from q that were in $\mathcal{V}_k$ already.

We can also efficiently compute the *projected matrix* $M_{k+1} = V_{k+1}^T A V_{k+1}$ on the expanded space V_{k+1} using the projection M_k on V_k. Recall that this is the matrix that we need in order to solve the reduced problem. It will prove to pay off if we maintain a matrix $W_k := AV_k$ during the expansion process. Each time after the orthogonal expansion v is computed from q, compute $w := Av$ and add this as new column to the current W_k. Then note that

$$M_{k+1} = (V_k|v)^T (W_k|w) = \left(\begin{array}{c|c} M_k & V_k^T w \\ \hline v^T W_k & v^T w \end{array} \right), \tag{8}$$

and that we have M_k still available from the previous projection[2]. This means, that we only need to compute $2k+1$ new entries instead of all $(k+1)^2$ of them.

The resulting *iterative projection algorithm* is given below. We have skipped the indices, which are not really necessary in the algorithm.

> Algorithm 1: **Iterative Projection Method**
>
> **input:** A, V, ε; matrix, first subspace, the desired reduction of the residual
> $W = AV$;
> $M = V^H W$; projected matrix belonging to the subspace $\mathcal{V}$ spanned by V
> $r = s =$*residual of projected problem*;
> **while** $\|r\| > \varepsilon \|s\|$
> q = *expansion vector, strategically obtained ...*;
> $\hat{v}$ = $(I - VV^T)q$; orthogonalization of q against V
> v = $\hat{v}/\|\hat{v}\|$; normalization
> w = Av; compute new column for the matrix W
> $M = \left(\begin{array}{c|c} M & V^T w \\ \hline v^T W & v^T w \end{array} \right)$; efficient implementation of projection $M = (V|v)^T (W|w)$ using previous M
> V = $(V|v)$; expansion of the subspace
> W = $(W|w)$; updating the matrix $W = AV$
> r = *residual of the new projected problem derived from M and V*;
> **end (while)**

Note! *In solving the linear system, the assumption $b \in \mathcal{V}$ forces us to start with $\mathcal{V}$ =span(b) as initial subspace.*

In the algorithm-frame above, we have only sketched the main iteration. We did not specify how to obtain the vector by which to expand the current subspace. Sometimes this is done iteratively as well. We will now present one of the most important and elegant strategies.

IV. Expanding towards an invariant subspace

If the approximation coming from the projection on a subspace does not have the accuracy that is desired, we need to think about how to expand the subspace in such a way, that the next projection will give better results. As we have seen in

[2] Also in the computation of the residual, the vector AVy is needed; this can then be implemented cheaply as $W_k y$

the previous, we know that we would like our new space to be *closer to invariant* than the one we had. At first glance, this seems a frustrating task: we started off, for example, with looking for an eigenvector v (which is a one dimensional invariant subspace), and now our algorithm wants us to iterate towards an invariant subspace of larger dimension that contains the start-vector as well as v. This might be a simpler problem (take the whole space and you're ready), but it looks as if we're only pushing the real problem slightly further away from ourselves [3]. In the linear system case, it seems even worse. All we wanted was to solve a system, and now we have to iterate towards a smallest possible invariant subspace containing b. Isn't this an example of the cure being worse than the disease? Apart from that, what are the odds that this smallest invariant subspace is *not* the whole space?

A straight-forward approach

Question. *Suppose you're given a matrix A and a vector b and you're asked to produce the smallest invariant subspace $\mathcal{V}$ for A such that b is in it. How would you proceed?*

Not hindered by any pre-knowledge, and hoping that the person who put this question to you gave you an easy one, you might as well compute Ab and check if this is, by some funny coincidence, a multiple of b. If this is the case, you can smile relieved and say "Hey, it's an eigenvector!". If, on the other hand, it isn't, then you *do* know that $\mathcal{V}$ must, at least, contain Ab as well. So, putting $\mathcal{V}_1 =< b >$, where the pointed brackets indicate the span of the vectors between them, and $\mathcal{V}_2 =< b, Ab >$, we have replaced the question above by the question how to find the smallest invariant subspace that contains $\mathcal{V}_2$. And, finding an answer for *this* question can be started off similarly. Just compute (a basis for) $A\mathcal{V}_2$ and see if it's in $\mathcal{V}_2$. Here it starts getting interesting, because clearly, we only need to check where the basisvector Ab of $\mathcal{V}_2$ ends up. If $A(Ab)$ is in $\mathcal{V}_2$, we're ready, and if not, we can define $\mathcal{V}_3 =< b, Ab, A(Ab) >$ and proceed[4].

Answer. *If the smallest invariant subspace containing b has dimension m, then it equals $\mathcal{V}_m$. For any k, the space $\mathcal{V}_k :=< b, Ab, \cdots, A^{k-1}b >$ is called the k-th Krylov subspace for the vector b and the operator A. Usually it's denoted by $K^k(A, b)$, and that's why we will do the same.*

As we've just seen, a nice property of Krylov subspaces is that if you apply A to it, you move out of the space in at most one direction. Put mathematically, we have

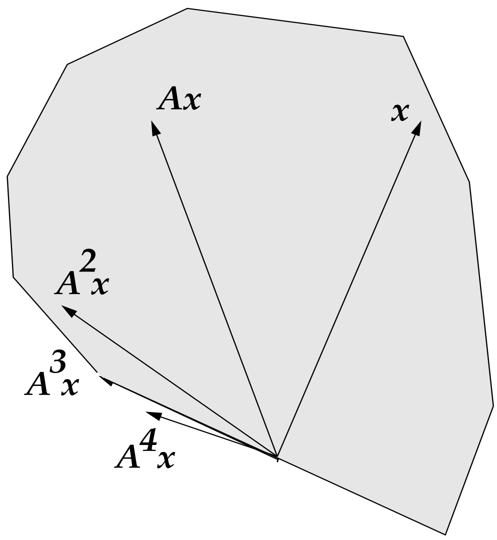
$$k - 1 \leq \dim(K^k(A, v) \cap AK^k(A, v)) \leq k. \tag{9}$$

So, for increasing k, we might say that the Krylov subspace moves, in the relative sense, towards an invariant subspace. This is not too bad in the light of our projection methods. And indeed, the Rayleigh-Ritz-Galerkin projection algorithm combined with Krylov subspaces gives rise to an important class of algorithms in

[3] This way to tackle a problem seems to be very popular in mathematics. It can be very successful if many small pushes add up to a big push

[4] This, actually, is an example in which many slight pushes forward bring you to the answer of a problem

both the linear system and the eigenvalue problem case.



Iterative Krylov subspace projection

In the Rayleigh-Ritz-Galerkin framework, we can try to solve our oversized linear algebra problem by iterating towards an invariant subspace as follows. Starting in the eigenvalue problem case with a random unit vector v, and in the linear system case with $v := b/\|b\|$ we do a first projection. Then, upon entering the **while**-loop, we propose to use Av as *expansion vector* for the current subspace, as suggested by the results of the previous section. But, as it turns out, there is an important alternative giving the same result.

Note! *The residual r resulting from projection on $K^k(A, v)$ is an element of $K^{k+1}(A, v)$. This means that it can be used to expand the subspace, especially since it is already orthogonal to $K^k(A, v)$. The orthogonalization step in our algorithm therefore becomes superfluous with this choice of expansion.*

If the residual of the now two-dimensional projected problem is not small enough, we repeat the procedure.

Note! *At all iteration numbers k, the columns of the matrix $V_k = V$ form an orthonormal basis for $K^k(A, v)$ consisting of the startvector and the consecutive normalized residuals. Moreover, the basis V_k of $K^k(A, v)$ is part of the basis V_{k+1} of $K^{k+1}(A, v)$.*

This last property leads to the following. Since for all k, we have that

$$A : K^k(A, v) \rightarrow AK^k(A, v) \subset K^{k+1}(A, v), \tag{10}$$

it follows that A maps the first j columns of V_k on a linear combination of the first $j+1$ columns of V_{k+1} for all $j \leq k$. Or, in matrix language,

$$A \left[\begin{array}{c} \\ \\ V_k \\ \\ \\ \end{array} \right] = \left[\begin{array}{c|c} & \\ & \\ V_k & v_{k+1} \\ & \\ & \end{array} \right] \left[\begin{array}{ccc} * & \cdots & * \\ \square & \ddots & \vdots \\ 0 & \ddots & * \\ \hline 0 & 0 & \square \end{array} \right]. \tag{11}$$

The most right matrix, with $k+1$ rows and k columns, we'll call $H_{k+1,k}$. This H stands for *Hessenberg*, because matrices (h_{ij}) such that $h_{ij} = 0$ for all indices $i \geq j+2$ are called *upper Hessenberg matrices*. The non-zero elements in column j of $H_{k+1,k}$ are the local coordinates of Av_j with respect to the basis $v_1, \cdots, v_{j+1}$. As a matter of fact, since Av_i is a linear combination of $v_1, \cdots, v_j$ for all $i < j$, the relative magnitude of the entry $h_{j+1,j}$ measures how much V_k is mapped outside itself. Denoting the $k \times k$ upper part of $H_{k+1,k}$ simply by H_k, we can state another important and interesting observation.

Note! *The projected matrix $M_k = V_k^T A V_k$ equals H_k. If A is symmetric, then H_k is also symmetric, and in particular tridiagonal. This means, that in expanding the projected matrix from M_k to M_{k+1}, we know a priori that the new far right column m_{k+1} of M_{k+1} (and by symmetry also the last row) will only have two non-zero elements.*

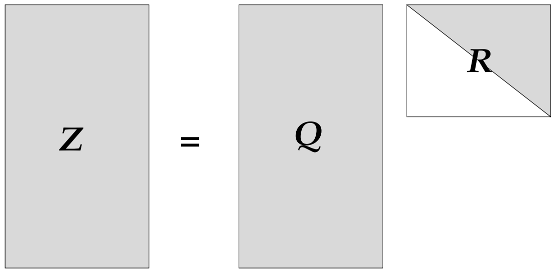
The message is, that in calculating the projected matrix for symmetric A, we can explicitly make use of those known zero elements to reduce the computational costs. Moreover, as we will illustrate in the upcoming section, efficient algorithms are known to solve linear algebra problems with Hessenberg or symmetric tridiagonal matrices, so apart from the fact that the reduced problem is smaller than the original one, it also has a favorable structure.

V. Small and medium problems

For completeness of our toolbox for oversized problems, let's reflect for a while on the small and medium problems, of which we loosely assumed we could just download some software and solve them. Instead of boring you with details on Gaussian elimination, we'll introduce you to the *QR-factorization*. The idea behind it is simple.

QR-factorization. *Let Z be an $n \times k$ matrix of rank k. Then there exist an $n \times k$ matrix Q with $Q^T Q = I$ and a $k \times k$ upper triangular matrix R such that $Z = QR$.*

This factorization can easily be constructed, for example by applying (from the left to the right) the *Gram-Schmidt orthonormalization process* to the linearly independent columns of Z. The orthonormal result is then Q, and the matrix R contains orthogonalization coefficients *above* the diagonal, and normalization coefficients *on* the diagonal.



Linear systems and QR-factorization

Any linear system $Zx = b$ can be solved quite efficiently by QR-factorizing Z and solving $Rx = Q^T b$. The efficiency comes from the fact that $Q^{-1} = Q^T$ and from the comfortable way in which systems with upper triangular matrices can be solved:

$$Rx = Q^T b, \quad \text{or} \quad \begin{bmatrix} * & \cdots & * \\ 0 & \ddots & \vdots \\ 0 & 0 & * \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix} = Q^T \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix} \tag{12}$$

Compute the right-hand side $Q^T b$. Then, just start with solving x_k from the last equation, and proceed to higher rows by substitution.

Another important observation is that for Hessenberg matrices in general, which have already a large amount of zero elements below the diagonal, the QR-factorization can be computed far more cheaply then for general matrices. In particular, for the Hessenberg matrices that arise in our projection algorithms, the QR-decomposition at a certain iteration step can be obtained from the one in in the *previous* iteration step at relatively small costs. This all contributes to the efficiency of the Ritz-Galerkin projection methods on Krylov subspaces.

The QR-algorithm

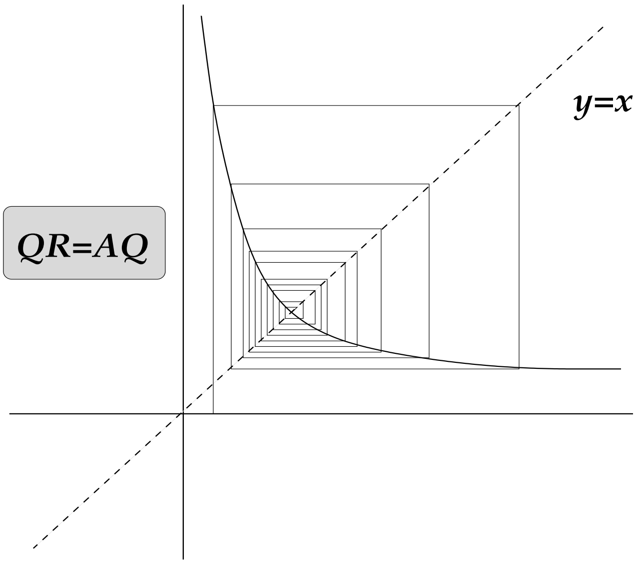
The solution of small and medium size eigenvalue problems can be done using the QR-decomposition as well. Not, of course, by doing one decomposition, but by doing a repetition of them. After all, since eigenvalue problems are equivalent to finding roots of polynomials, this is *necessarily iterative* as soon as the degree of the polynomial exceeds four. Before we proceed, let's first highlight the *Schur form* of a matrix Z. Its existence can be proved by a rather straight-forward induction argument.

Schur canonical form. *Let Z be a square matrix. Then there exists a matrix Q with $Q^T Q = I$ and an upper triangular matrix R such that $ZQ = QR$. The diagonal elements of R are equal to the eigenvalues of Z.*

Please *do* note the similarity (but equally important, the difference) between the Schur form and the QR-decomposition. Doesn't this *ask* for an algorithm to produce the Schur form by means of QR-decompositions? And in fact, a very simple idea would be to do a *Picard iteration* or *successive substitution* on $ZQ = QR$ as follows:

$$\text{Start with } Q_0 = I, \quad \text{iterate} \quad Q_{n+1}R_{n+1} = ZQ_n, \tag{13}$$

where, of course, the left-hand side $Q_{n+1}R_{n+1}$ is obtained by QR-decomposition of the right-hand side ZQ_n.



Clearly, any fixed point of this iteration yields a Schur form. And believe it or not, this simple looking idea is at the foundations of one of the most successful eigenvalue algorithms to compute the Schur form of a small size matrix, the *QR-algorithm*, which is a more advanced implementation of the same idea.

QR-algorithm. *The basic QR-algorithm is Picard iteration on the Schur canonical form, using QR-decomposition at every iteration step.*

To be precise, the QR-algorithm is usually presented as follows:

$$\text{Start with } \hat{Q}_1\hat{R}_1 = Z, \quad \text{iterate} \quad \hat{Q}_{n+1}\hat{R}_{n+1} = \hat{R}_n\hat{Q}_n. \tag{14}$$

In this form, the intuition behind it is less clear, but the algorithm is more robust and cheaper. Some manipulations give that (14) actually produces $\hat{Q}_{n+1}$ and $\hat{R}_{n+1}$ such that

$$\hat{Q}_1 \cdots \hat{Q}_n\hat{Q}_{n+1}\hat{R}_{n+1} = Z\hat{Q}_1 \cdots \hat{Q}_n, \tag{15}$$

which shows that the transformation Q_{n+1} from (13) is generated as a product of transformations $\hat{Q}_j$, and that the upper triangular matrices are in principle equal for both iterations. Nice aspect of this formulation is that the right hand side $\hat{Q}_n\hat{R}_n$ in (14) is always spectrally equivalent to the original matrix Z, which follows immediately from $\hat{R}_1\hat{Q}_1 = \hat{Q}_1^T Z\hat{Q}_1$ and an induction argument. Moreover, in case the algorithm converges, we have that $\hat{Q}_j \to I$, so $\hat{R}_j\hat{Q}_j \to R$, the triangular QR-factor of Z. A last favorable property results from the following.

Note! *If Z is an upper Hessenberg matrix, then so are its orthogonal QR-factor Q and the product RQ. In that case, one only needs to perform the relatively cheap*

QR-decompositions for Hessenberg matrices in each step of the iteration (14).

Therefore, if Z is not upper Hessenberg to start with, it is worthwhile to transform it such, that the result is indeed an upper Hessenberg matrix. Note that this can be realized by transformation to an orthonormal basis of a full Krylov subspace. For symmetric matrices, the situation is even more favorable.

Note! *In our initial iteration (13), even if we assume Z to be a Hessenberg matrix, the right-hand side ZQ_n is not upper Hessenberg anymore. In fact, being the product of two upper Hessenberg matrices, it has one more non-trivial subdiagonal.*

Shifts and lucky guesses

It would go to far to try and explain the convergence behavior of the QR-algorithm. A few words, however, won't harm our cause. First of all, let's reflect on what happens to the QR-decomposition if Z is singular. If we assume that Z is an upper Hessenberg matrix with non-zero elements below the diagonal, then the singularity can only be caused by the last column being a linear combination of the others. Still, a QR-factorization can be made, one in which the last column of Q spans the one remaining dimension (as would have happened without the singularity, of course) but with an upper triangular matrix $R = (r_{ij})$ with $r_{kk} = 0$. It might seem a trivial observation, but then the matrix $Q^T ZQ = RQ$ has last row zero, which *shows that Z has a zero eigenvalue.*

The *QR-algorithm with shifts* exploits this idea to accelerate the original QR-algorithm. First, Z is transformed to an upper Hessenberg matrix Z_1, then it iterates as follows,

$$\text{Start with } \hat{Q}_1\hat{R}_1 = Z_1 - \mu_1 I, \quad \text{iterate} \quad \begin{cases} Z_{n+1} & := \hat{R}_n\hat{Q}_n + \mu_n I, \\ \hat{Q}_{n+1}\hat{R}_{n+1} & := Z_{n+1} - \mu_{n+1} I. \end{cases} \tag{16}$$

The intuition behind this is, that again at each stage, the upper Hessenberg matrix Z_n has the same eigenvalues as the original matrix Z. And if $Z - \mu_n I$ happens to be singular, then the last row of Z_n equals μe_n^T, showing explicitly that Z has an eigenvalue equal to μ. We could then proceed the QR-iteration with the $(k-1) \times (k-1)$ upper-left block of Z, which has the remaining eigenvalues of Z.

If we would be able to a *priori* guess the eigenvalues *exactly right* then we would be earning our money differently than now. But based on a continuity argument, we might hope that if we use a shift that is *close to* an eigenvalue, then for the last row $e_k^T Z_n$ of Z_n we have

$$e_n^T Z_n = \mu e_k^T + h^T, \tag{17}$$

where $h = \alpha e_{k-1} + \beta e_k$ has a relative small norm compared to $|\mu|$. This would make $\hat{\mu} = \mu + \beta$ an approximation of an eigenvalue, while the size of α would indicate *how good* this approximation is. Hoping that it is an *even better* approximation than μ, we could continue the QR-algorithm with next shift equal to $\hat{\mu}$.

Note! *As soon as the subdiagonal element at position $(j+1, j)$ of the matrix Z_n is very small, the problem might be split in two by replacing this small element by zero, and continuing with the remaining $j \times j$ and $(k-j-1) \times (k-j-1)$ blocks.*

A last remark on the QR-algorithm. If Z is a real matrix with complex eigenvalues, then it is clearly impossible to iterate towards a Schur decomposition using real shifts only: every matrix R_n that is produced, will have real entries as long as real shifts are used. Complex and double shifts are remedies for this, but we won't present any details.

VI. Feedback in the eigenvalue algorithm

Let's return to our projection methods for very large linear algebra problems. Recall that they produce small systems or small eigenvalue problems to solve, for which the small matrix is upper Hessenberg or even tridiagonal. Those were exactly the matrices for which the algorithms in the previous section work very efficiently. So it seems as if we're doing fine so far, and that everything fits perfectly together. We will now solely concentrate on the eigenvalue problem and explain a mechanism that will fit in the total framework even more beautifully.

How to start ...

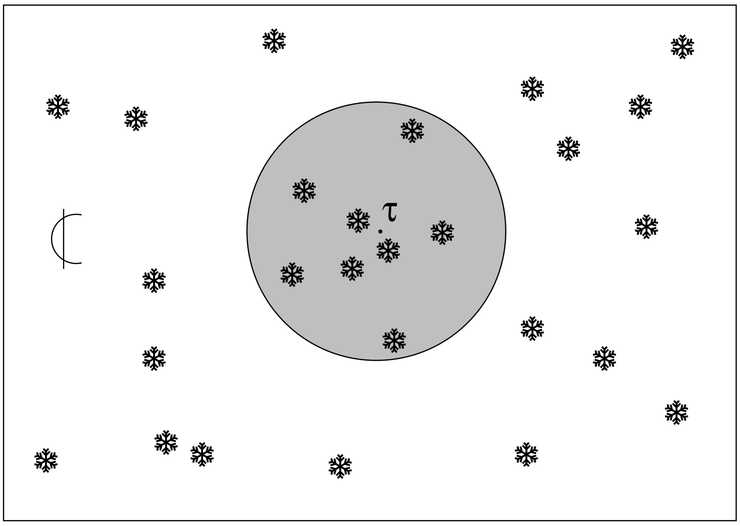
Suppose that you're interested in finding the, say, p eigenvalues of a matrix A that are closest to some target value τ in the complex plane. The ideal situation would be to have a vector available that is a linear combination of five corresponding independent eigenvectors or Schur vectors. Then in five steps, an invariant subspace would be generated, and the problem solved. However, knowing nothing about the spectrum initially, there's not much better we can do than to start our eigenvalue algorithm with some random vector v. Doing a number of iterations and watching the eigenvalue approximations then gives a first idea about the part of the spectrum of A. Since it might be that our startvector has only small components in the directions of the eigenvectors we are interested in, it could very well happen that convergence to *other* eigenvalues than the ones we want, occurs. Regardless of that, the number of iterations should not become too large; the calculations become slower and slower since the work per iteration step increases quadratically, and also the computer memory might become too full to have good performance. So, what to do if we do not want to expand our subspace any further, but we still *haven't found what we're looking for*?

... and how to proceed

Suppose we have done $k > p$ steps of the algorithm, which gives us k approximate eigenvalues. The idea is to divide those into two groups: those we find uninteresting, say $\mu_1, \cdots, \mu_\ell$, because they are relatively far away from the target τ, and the remaining "good" ones, say $\mu_{\ell+1}, \cdots, \mu_k$. Then, compute

$$\hat{v} = \frac{\tilde{v}}{\|\tilde{v}\|}, \quad \text{where} \quad \tilde{v} = \Pi_{j=1}^{\ell}(A - \mu_j)v, \tag{18}$$

and start the algorithm *all over again* but now with $\hat{v}$ as startvector. The philosophy behind this is, that apparently, v contained too large components in uninteresting directions (that is why and how $\mu_1, \cdots, \mu_\ell$ were generated in the first place). By applying the product in (18) to v, we aim to partially *remove* or *filter away* those components from v, so that starting again with $\hat{v}$ hopefully won't yield approximations of those uninteresting eigenvalues again.



The big disadvantage is, that it seems quite an expensive procedure to apply a projection method only to find out *afterwards* that we started with the wrong vector.

Recycling Krylov subspaces

Watching the screen, we find out that it could have been better positioned. We even see that some directions are okay, and some aren't. Nevertheless, the previous section suggests that we roll it up, put it away and start all over again. Now, can't that be done better? Wouldn't it be a shame to throw away this orthonormal basis of the k-dimensional Krylov subspace that we so carefully built and maintained? After all, isn't it true that we only want to get rid of ℓ dimensions, and not of *all of them*? Indeed, for our new startvector $\hat{v}$ from (18), we have that $\hat{v} \in K^{\ell}(A, v)$, which implies that

$$K^{k-\ell}(A, \hat{v}) \subset K^{k}(A, v). \tag{19}$$

Would it be too much to ask if there's a way to *extract* this subspace from $K^k(A, v)$, and while we're at it, also an orthonormal basis for it and the Hessenberg representation of A on this orthonormal basis? Or are we pushing our luck now?

Let's first write down what we have. We have, with our random start v, matrices V_{k+1} and $H_{k+1,k} = (h_{ij})$ such that

$$AV_k = V_{k+1}H_{k+1,k}, \quad \text{or, equivalently,} \quad AV_k = V_kH_k + h_{k+1,k}v_{k+1}e_k^T. \tag{20}$$

Second, let's write down what we *would like* to have: an orthonormal basis $W_{k-\ell+1}$ for the $k-\ell+1$ dimensional subspace $K^{k-\ell+1}(A, \hat{v})$ and an upper Hessenberg matrix $\hat{H}_{k-\ell+1,k-\ell}$ such that, with $q := k - \ell$,

$$AW_q = W_{q+1}\hat{H}_{q+1,q}, \quad \text{or, equivalently,} \quad AW_q = W_q\hat{H}_q + \hat{h}_{q+1,q}w_{q+1}e_{q+1}^T. \tag{21}$$

The reason *why* we would like to have this is, that we would have *mimicked* the first q steps of our algorithm with startvector $\hat{v}$ *without actually performing them*, and from there on we could just *continue* the algorithm by expansion with the residual for the projected problem.

Two important observations regarding bases for Krylov subspaces should be made here[5]. First one is, that when the basis is orthonormal, it is uniquely determined by the first vector, give or take the sign of the basisvectors. Secondly, not only does A reduce to upper Hessenberg form on such a basis, also the *reverse* holds: if $AV_k = V_{k+1}H_{k+1,k}$ with H_k upper Hessenberg, then V_k spans the Krylov subspace $K^k(A, V_k e_1)$. So, in fact, in order to find W_q, all we need to find is the orthogonal transformation Q such that $V_k Q$ has $\hat{v}$ as first column, and such that A has upper Hessenberg form on the first q columns of $V_k Q$. We'll give the elegant solution of this problem in the case $q = k - 1$, which corresponds to filtering away only one component of our original startvector v. If one wants to filter away more components, the generalization is, hopefully, obvious.

The QR-algorithm strikes again

The key to the solution is, surprisingly enough, the QR-algorithm. Just apply one iteration step of it, with shift μ, to the Hessenberg matrix H_k,

$$QR := H_k - \mu I, \quad \text{and} \quad \hat{H}_k := RQ + \mu I. \tag{22}$$

This transformation Q and the upper Hessenberg matrix $\hat{H}_k$ leads to the fulfillment of all the wishful thinking of the previous section. Simply define $W_k := V_k Q$, after which W_k still spans $K^k(A, v)$ since the right-multiplication does not change the column span. Also, note that $W_k^T W_k = I$. Then, using (20) and (22), for the first column $W_k e_1$ of W_k we find,

$$W_k e_1 = V_k Q e_1 = V_k Q \frac{Re_1}{r_{11}} = \frac{1}{r_{11}} V_k (H_k - \mu I) e_1 = \frac{1}{r_{11}} (AV_k - \mu V_k) e_1 = \hat{v}, \tag{23}$$

where in the last step, we used that $V_k e_1 = v$, and that we started, at the left, with a vector of length one. It remains to be shown that for all $j \leq k - 1$, the first j columns W_j of W_k span $K^j(A, \hat{v})$.

Note! *Since W_k does not span $K^k(A, \hat{v})$, there does not exist an upper Hessenberg matrix H such that $AW_{k-1} = W_k H$.*

It will, however, appear to be possible to *alter* the last column of W_k such that we obtain our goal. First note that

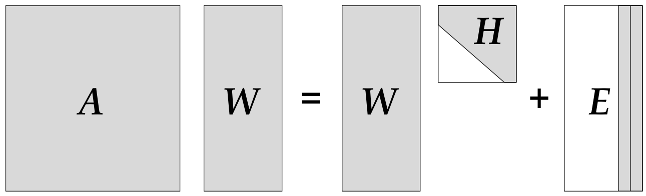
$$V_k H_k Q = V_k (QR + \mu I) Q = V_k Q \hat{H}_k = W_k \hat{H}_k, \tag{24}$$

which is nothing more than applying (22), after which we find, using the right (and right) relation in (20), that

$$\begin{aligned} AW_k = AV_k Q = V_k H_k Q &\quad + h_{k+1,k} v_{k+1} e_k^T Q \\ = W_k \hat{H}_k &\quad + h_{k+1,k} v_{k+1} e_k^T Q. \end{aligned} \tag{25}$$

[5] By basis we mean the inductively built basis

Now, the first $k-2$ columns of the $n \times k$ matrix $E_k := h_{k+1,k} v_{k+1} e_k^T Q$ are zero, because Q, being the orthogonal QR-factor of H_k, is an upper Hessenberg matrix. Both last columns of E_k are *known* multiples of v_{k+1}, let's say $E_k e_{k-1} = \alpha v_{k+1}$ and $E_k e_k = \beta v_{k+1}$.



So, writing $\hat{H}_{k,k-1}$ for the first $k-1$ columns of $\hat{H}_k$, we find by comparing the first $k-1$ columns of (25) that

$$AW_{k-1} = W_k \hat{H}_{k,k-1} + \alpha v_{k+1} e_{k-1}^T. \tag{26}$$

Since $\hat{H}_{k,k-1}$ is an upper Hessenberg matrix, the only equation in which the last column w_k of W_k appears, is

$$AW_{k-1} e_{k-1} = w_k \hat{h}_{k,k-1} + \alpha v_{k+1}. \tag{27}$$

Note that $W_{k-1}^T (w_k \hat{h}_{k,k-1} + \alpha v_{k+1}) = 0$. So, if we *re-define* w_k and $\hat{h}_{k,k-1}$ as a unit-vector and scalar such that $w_k \hat{h}_{k,k-1}$ equals the right-hand side of (27), then redefining $W_k := (W_{k-1} | w_k)$ finally leads to $AW_{k-1} e_{k-1} = w_k \hat{h}_{k,k-1}$ and hence,

$$AW_{k-1} = W_k \hat{H}_{k,k-1}. \tag{28}$$

Summarizing we can say that if we regret to have started the eigenvalue algorithm with startvector v, then we can still do something about it without throwing away everything we have done. The QR-algorithm applied to the small Hessenberg matrix H_k provides us with the unitary transformation that selects a subspace from $K^k(A, v)$ that contains the "most relevant" information.

Note! *As we have seen before, applying a step of the QR-algorithm to H_k with a shift μ that is an eigenvalue of H_k, produces a matrix $\hat{H}_k$ that has μe_k^T as last row. As a consequence we get $\hat{h}_{k,k-1} = 0$ in (27).*

And now, for something completely different ... What about the objects that we try to approximate? Do they let themselves be approximated, or do they give rise to false images on the screen that might just look perfectly okay, like the rabbit that wasn't a rabbit?

VII. The complex minefield

It is well-known that eigenvalues depend continuously on the entries of the matrix. In general, we cannot expect to have higher smoothness, as the example

$$A = \begin{pmatrix} 1 & a \\ \varepsilon & 1 \end{pmatrix}, \quad \lambda_{1,2} = 1 \pm \sqrt{a\varepsilon} \tag{29}$$

clearly shows. The same example (with $\varepsilon = 0$) also illustrates that the *Jordan canonical form*, which is so useful in many theoretical issues, is not even continuous, as a function from the matrix entries. That is why in the previous we preferred to use the *Schur canonical form*, which *is* in fact continuous as a function of the matrix entries, and this partly explains the success of eigenvalue algorithms based on the Schur form. The success becomes even bigger when eigenvalues are simple and *isolated*, which makes them differentiable.

Introducing the characters

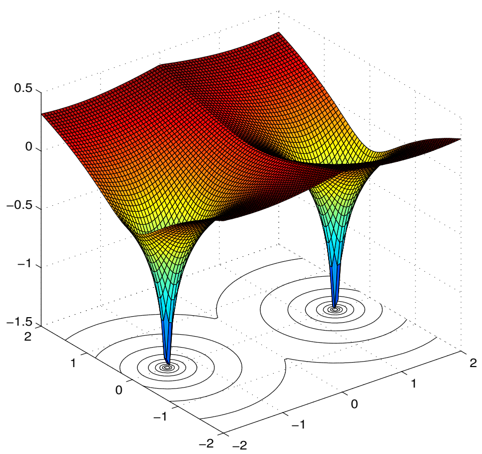
Before we go further, let's briefly recall the main classification of matrices. The nicest group is formed by the *Hermitian* matrices, for which $A^H = A$. It is a subset of the larger group of so-called *normal* matrices. Normal matrices have eigenvectors that form an *orthonormal* basis. In fact, they are exactly the matrices for which A and A^H commute. Clearly, they form a subset of all *diagonalizable* matrices, which are the ones that have eigenvectors forming a basis. Matrices that are *not* diagonalizable are also called *defective*, which refers to a lack of eigenvectors to form a basis. Following our linguistic intuition, we tend to think of defective matrices as the bad guys. As a matter of fact, non-normal matrices in general are often presumed to be a bit suspect as well, even if they're not defective. We'll illustrate why in the following.

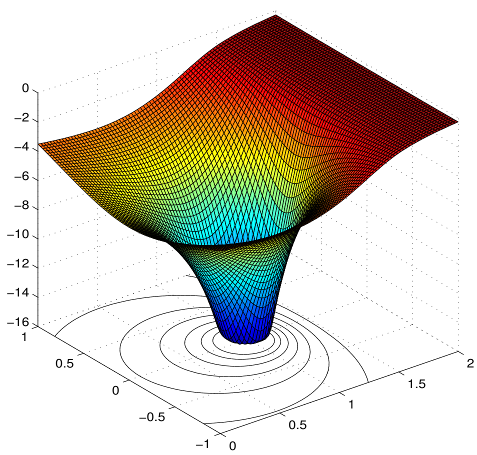
Avoiding the pitfalls

Anything that has to do with eigenvalues, needs to be approached with proper care. After all, eigenvalues indicate *singular behavior*. In order to make things even more singular, people tend to be interested in the following function,

$$R : C \to \mathbb{R} : z \mapsto \|(A - zI)^{-1}\|^{-1}. \qquad (30)$$

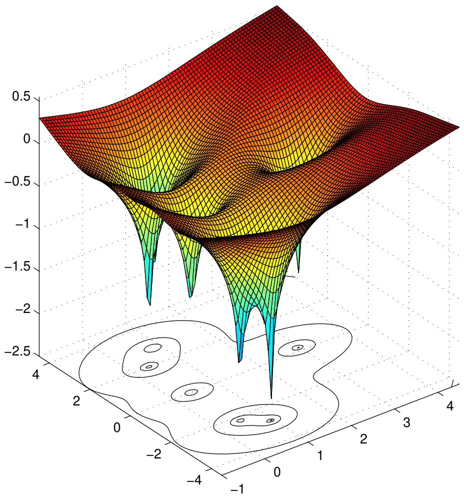
Each eigenvalue corresponds to a singularity, and the characteristics of the singularity depend on how close a matrix is to defective, or how far away from normal. We will illustrate this by drawing a logarithmic plot containing the function R and/or its contourlines in the complex plane.

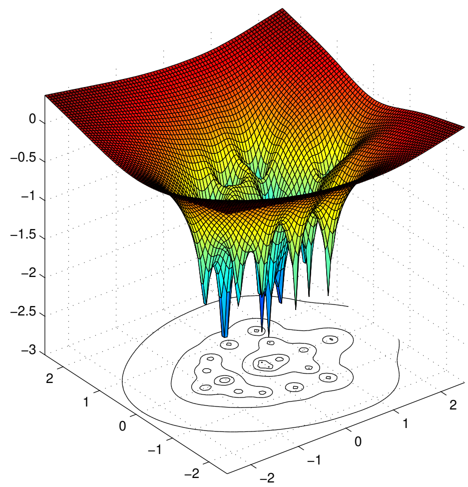




On the left we see two eigenvalues of a normal matrix. On the right we see the only (twenty-fold) eigenvalue of a non-normal matrix of size twenty. The same contours are given on a comparable scale. The inner contour corresponds to $R(z) = 10^{-16}$. In the left picture, the contours are very tight around the eigenvalues, so that we cannot even see them. In the right picture, however, there is a big disk in which $R(z) \leq 10^{-16}$.

Note! *In most practical computations, it is hard to distinguish between* 10^{-16} *and zero. So, the average computer will think that this relatively small matrix has an "eigendisk" with centre 1 and radius 0.1, implying that it is impossible to find the eigenvalue with an accuracy of more than about ten percent.*

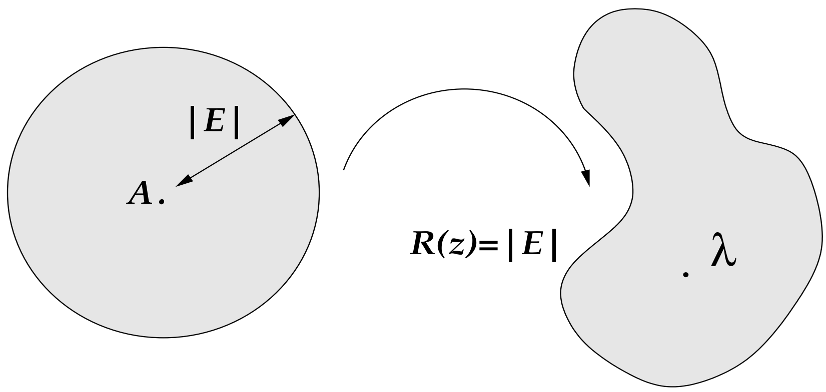




In the two pictures above, we present similar plots for a matrix of size six on the left, and a random matrix of size twenty on the right. What we try to illustrate here is, that a cluster of singularities close to each other can also make the situation more complicated. Imagine yourself somewhere in the complex plane, trying to spot a particular eigenvalue, if there are one million of them around. In particular, if there's another eigenvalue around having a "deep" singularity, our object of interest might just drown in that singularity and remain unnoticed.

Eigen-areas

The message of the previous is clear. It is not the singularity that counts, but the area in the complex plane in which we cannot distinguish anymore what's happening: do we have a singularity here, or not? The exact eigenvalue could be anywhere in this area. Of course, we need to be aware of this if we approximate eigenvalues of any matrix, and those of a very large matrix in particular. As a matter of fact, *each* contourline has a nice interpretation. Suppose we have one at height ε . Then it bounds the region within which an eigenvalue can move if the original matrix A is *perturbed* by a matrix E with $\|E\| \leq \varepsilon$. Or, in other words, for each z with $R(z) \leq \varepsilon$, there exists a matrix E with $\|E\| \leq \varepsilon$, such that z is an eigenvalue of $A+E$.



If ε is small while at the same time the region bounded by the ε-contour of R is large, the eigenvalue is called *sensitive to perturbations*. Analysis learns that the distance of A to a defective matrix and its distance to a normal matrix are important factors that influence the sensitivity of eigenvalues.

Residuals revisited

Let's see how our projection method for eigenvalue problems fits into all this. Recall that we defined the *eigenvalue residual* $r := AV_k y - \mu V_k y$, where μ is an eigenvalue of the projected matrix M_k and y a corresponding eigenvector of length one. The residual was supposed to give us an idea how good μ and $v := V_k y$ approximate an eigenpair of A. We are now able to make this claim more solid.

Consider the surprisingly simple equality,

$$(A - rv^T)v = Av - rv^T v = \mu v. \tag{31}$$

It states that we have found a matrix $E = -rv^T$ such that μ and v are an exact eigenvalue and eigenvector of $A + E$. It all becomes even more interesting when we realize that $\|E\| = \|r\|$ and that we are able to *compute* this norm easily.

Claim! *Let $r = Av - \mu v$ be an eigenvalue residual coming from our projection method. Then μ lies within the contour $R(z) \leq \|r\|$, or equivalently, $R(\mu) \leq \|r\|$.*

Now, before getting too excited about this, let's not forget that computing these contours is not an easy job, and it might very well be that it is much harder then computing μ and v. But there are situations in which you need some guarantees about the quality of μ, and in those cases this result may be very useful.

VIII. Modern developments, ancient ideas

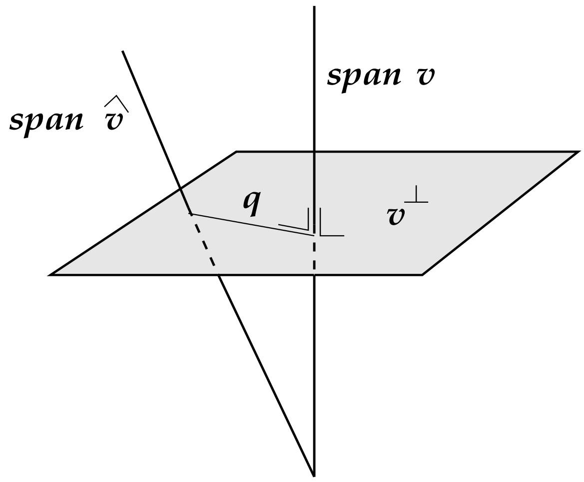
The eigenvalue algorithm that we have sketched in the previous sections, is not the only one that is based on projections. Even though the idea of expanding the subspace towards an invariant subspace is very appealing and natural (not to mention, it often works quite satisfactory), there is another important strategy to expand the subspace. The idea behind it might seem a bit perverse: what about expanding the subspace with *error* of the problem, then we're ready at once.

Moving the problem forward

Hold on, reader! We're not kidding here. Of course, there is no way that we can add the error to the space, since by linearity, having the approximation and the error would mean that we have the solution. But in the case of expansion towards an invariant subspace, the philosophy was the same: we didn't actually ever expect to get there in the first place, we only wanted to do small steps in the right direction and stop when we would be close enough. Our new idea is similar in this respect. It brings a kind of *nestedness* or *recursion* into the algorithm, and by means of a little bit of steering, it might be possible to *find a balance* between the two extremes of *expanding with the error* and *expanding at random*. It is all a matter of *where to put the energy*, and by dividing the energy cleverly, we might prevent the subspace from growing too much while on the other hand preventing to solve the whole problem by finding the perfect expansion vector.

Orthogonal corrections

Suppose you have an initial subspace $\mathcal{V}$ spanned by v and that you're after an eigenvector $\hat{v}$. Assume both v and $\hat{v}$ to be of unit length. Write $\mu = v^H Av$ for the eigenvalue approximation belonging to the first projection, and let $r := Av - v\mu$ be the corresponding residual. Then, if $\hat{v} \notin \mathcal{V}^\perp$, there exists a unique *orthogonal correction* $q \in \mathcal{V}^\perp$ that yields $\hat{v}$, meaning that $q \perp v$ and $\hat{v} = (v+q)/\|v+q\|$.



It can be shown that q is a solution of the *non-linear* equation

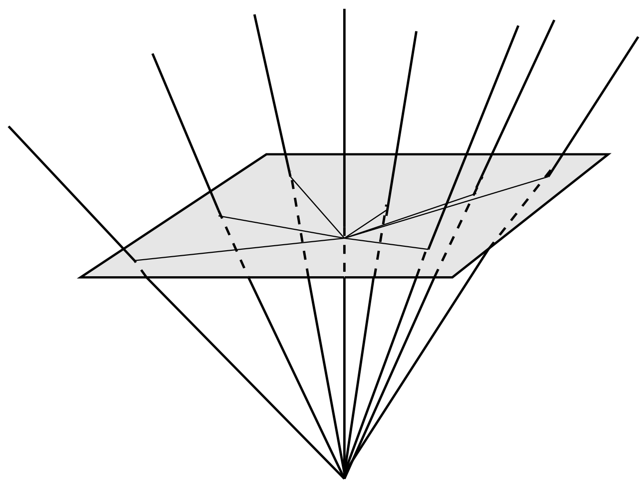
$$q \perp v \quad \text{and} \quad (I - vv^H)(A - \mu I)q = q(v^H Aq) - r. \tag{32}$$

Note that the matrix $(I - vv^H)(A - \mu I)$ in (32) is *singular*, not because of μ, which, after all, is only an *approximation* of an eigenvalue, but because of the projection $(I - vv^H)$ on the orthogonal complement $\mathcal{V}^\perp$ of v. The *orthogonality constraint* $q \perp v$ together with the fact that $r \in \mathcal{V}^\perp$, make (32) into a better posed problem. Indeed, the total right-hand side $q(v^H Aq) - r$ is in $\mathcal{V}^\perp$, and after applying $A - \mu I$ to a vector $w \in \mathcal{V}^\perp$ it is projected back onto $\mathcal{V}^\perp$. This makes (32) resemble a subspace method, with the *very big* subspace $\mathcal{V}^\perp$.

Projection on a giant subspace

You might wonder why we would like to do something like that, since we have stressed all the way, that the dimension of the subspace to project upon should be kept as small as possible. The answer is that this projection is of a *different type*. Its goal is (fortunately!) not so much to reduce the dimension of the problem, but *to get rid of* the dimension that might cause severe computational difficulties. Indeed, if μ is relatively close to an eigenvalue λ of A, we might get into the region in the complex plane where the function $R(z)$ from (30) is indistinguishable from zero. Or, equivalently, there would exist vectors such that $(A-\mu I)^{-1}w$ would be too large to handle. In particular vectors w that make a small angle with the eigenvector $\hat{v}$ belonging to the eigenvalue λ would suffer from this. By making sure that $A-\mu I$ only acts on vectors in $\mathcal{V}^{\perp}$, we hope to *filter out* vector components that might be extremely blown up by $(A-\mu I)^{-1}$, just as we tried to do filtering in our restart strategy for the Krylov subspace method for eigenvalues. So that motivates the projection on the big subspace, or, *working in the orthogonal complement of* v.

The equation (32) still might have several solutions because of the non-linearity. This is of course because the span of *every* eigenvector $\hat{v}$ of A that is *not orthogonal* to v, intersects with the *affine variety* $\{w+v|w^H v=0\}$, each giving rise to a corresponding orthogonal correction q.



Note! *The solution q of (32) with the smallest norm will give rise to the eigenvector $\hat{v}$ of A that has minimal angle to v. Of course, strictly speaking, there might be more than one solution q with minimal norm.*

Inexact solutions of Riccati equations

Equation (32) is, in fact, of a well-known type. It is classified as a special kind of *Riccati Equation*, named after *Il Barone* Jacopo Francesco Riccati. Riccati equations arise in many mathematical fields, like control theory, differential equations, and differential geometry. It is our goal to find approximations of solutions to the Riccati equation above, and use those to expand the subspace. Then, after solving the expanded projected problem, a new approximation of the eigenvector is used to write down a new Riccati equation to solve, and the process can repeat itself.

Jacopo Francesco Riccati (1676-1754)

We will now comment on three strategies to obtain approximations of solutions of the Riccati equation (32).

Neglecting the quadratic term

A first idea is to forget about the quadratic term in (33) and to find the (unique) solution $\hat{q} \perp v$ of

$$(I - vv^H)(A - \mu I)\hat{q} = -r. \tag{33}$$

This is just a linear system, and we might apply the projection method with Krylov subspaces, or any other method, to find approximations to its solution.

Note! *Halfway the nineteenth century, Carl Gustav Jacob Jacobi used, in almost the same context, his Jacobi iteration to find orthogonal corrections to approximations of eigenvectors. But instead of expanding the subspace by the solution, he merely replaced the subspace by the new approximation (he repositioned the screen).*

Carl Gustav Jacob Jacobi (1804-1851)

Of course, Jacobi did not know about Krylov subspace methods, and also the idea to expand subspaces was not known at that time. Mind you, he had to do everything by hand, so his matrices were rather small.

Note! *Since $r \perp v$ and $By := (I - vv^H)(A - \mu I)y \perp v$ for all y, the Krylov subspace $K^k(B, -r)$ is automatically orthogonal to v. So all approximations $\hat{q}_k$ to the solution $\hat{q}$ of* (33) *taken from this space, automatically satisfy $\hat{q}_k \perp v$.*

Since we already introduced an error by throwing away the quadratic term, we might as well be satisfied with an approximation for the solution of (33) that is not very accurate. This leads to the following interesting and surprising observation.

Note! *Using a Krylov subspace of dimension $k = 1$ to approximate the solution of (33) yields the quite crude approximation $\hat{q} \approx \hat{q}_1 := -\alpha r$ for some $\alpha \in \mathbb{R}$, which leads exactly to the Krylov subspace projection method for the eigenvalue problem treated in Section IV.*

Using Picard iteration

As we have seen in the QR-algorithm, the relatively simple concept of *Picard iteration* can be a very useful tool to approximate solutions of non-linear equations. If, instead of the linearization of the previous section, we apply Picard iteration to (32), we could be iterating as follows,

$$q_0 = 0, \quad \text{iterate} \quad q_{n+1} \perp v \quad \text{and} \quad (I - vv^H)(A - \mu I)q_{n+1} = q_n(v^H A q_n) - r. \tag{34}$$

In each iteration step, we have to solve a linear system, so the question is justified if this extra energy is well-spent. If, on the other hand, in finding an approximation $\tilde{q}_1$ for q_1 we have computed, say, a k-dimensional Krylov subspace $K^k(B, -r)$, we might as well *use the same subspace* to approximate the solution $\tilde{q}_2$ of

$$\tilde{q}_2 \perp v \quad \text{and} \quad (I - vv^H)(A - \mu I)\tilde{q}_2 = \tilde{q}_1(v^H A \tilde{q}_1) - r. \tag{35}$$

Note that, since $\tilde{q}_1 \in K^k(B, -r)$, the right-hand side $\tilde{q}_1^H(v^H A\tilde{q}_1) - r$ is also in $K^k(B, -r)$. We can go for an approximation $p \in K^k(B, -r)$ of $\tilde{q}_2 \approx q_2$ such that the projection of Bp on $K^k(B, -r)$ equals $\tilde{q}_1^H(v^H A\tilde{q}_1) - r$. The big advantage of this is, that we still have the projected matrix B available from the computation of $\tilde{q}_1$, or, even better, its QR-factorization. So, following this strategy, we could call the Picard iteration a *projected Picard iteration*, because it takes place completely within $K^k(B, -r)$.

Note! *By linearity, we can write $q_{n+1} = q_1 + w$ where w solves the projected equation*

$$w \perp v \quad \text{and} \quad (I - vv^H)(A - \mu I)w = q_n(v^H A q_n). \tag{36}$$

A non-linear subspace method

Going even one step further, we could try to project (32) directly on the Krylov subspace $K^k(B, -r)$. Suppose we have an orthonormal basis for it in the matrix V, and we write H for $V^H BV$. Then we can try to find $v = Vy \in K^k(B, -r)$ such that

$$V^H(BVy - (Vy)(v^H AVy) + r) = Hy - y(h^H y) - e_1\|r\| = 0, \tag{37}$$

where we wrote $h^H = v^H AV$. If we try to solve this small non-linear equation by Picard iteration and look at the iterates Vy_j, they *coincide* with the q_j from the previous section. So, nothing really new is happening, apart from the fact that we can interpret the small problem (37) as a small eigenvalue problem. Indeed,

$$\left(\begin{array}{c|c} 0 & h^H \\ \hline -\|r\| & \\ \cline{1-1} 0 & H \end{array}\right) \begin{pmatrix} 1 \\ y \end{pmatrix} = (h^H y) \begin{pmatrix} 1 \\ y \end{pmatrix}. \tag{38}$$

This clearly implies that instead of using a Picard iteration, one could use one or a few iterations of any efficient method (the QR-algorithm?) for smaller eigenvalue problems to obtain approximations for the eigenvector $(1, y)^H$ for which y has minimal norm. Such a y corresponds to the minimal norm approximation q of (32).

Note! *Since the Picard iterations typically only converge if v is close to $\hat{v}$, the subspace expansion is a natural way to bring v into this convergence area.*

Expanding with (approximate) solutions of the eigenvalue problem (38) leads to algorithms with generally small subspaces compared to the approach (33), but with more expensive iteration steps. It remains to be found out which implementation of the expansion methods with inexact solutions of the Riccati equation is the most favorable for which situation.

Methods for invariant subspaces

As you might expect, an *invariant subspace* is, in general, *more stable* than a single eigenvector. Therefore, people recently tend to be more and more interested in invariant subspaces, and in ways to find them that are *different* than going vector-by-vector for an eigenvector basis. In fact, such a basis does not necessarily exist in the first place.

The ideas of orthogonal corrections can be applied to invariant subspaces as well. Under similar conditions, one can show that there exists an *orthogonal correction* Q to an approximation X of an invariant subspace $\hat{X}$ (both stored as orthogonal matrices) that satisfies the Riccati equation

$$Q \perp X \quad \text{and} \quad (I - XX^H)AQ - QM = Q(X^H AQ) - R, \tag{39}$$

where $M = X^H AX$ and $R = AX - XM$ is the residual. A difficulty in this equation is that Q and M don't commute, so that the linear operator in the left-hand side that acts on Q, is harder to handle. Nevertheless, the essential ideas of the eigenvalue algorithm of the previous section can still be applied, and a similar projection algorithm with expansion by inexact solutions of the Riccati equation (39) can be written down, including the Picard iterations taking place in a Krylov subspace.

IX. Conclusions

We have introduced projection methods for oversized linear algebra problems. The ideas involve iterative expansion of the space to project upon, either with the aim to move towards an invariant subspace, or to include more and better approximations of the object of interest into the space. In the first case, we explained a way to restart the algorithm without breaking down the whole subspace that was just built. Or, as you wish, we showed how to rotate towards a sub-subspace that contains the most relevant information.

Apart from describing the algorithms, we also made some remarks on stability of eigenvalues and practical problems that may be encountered.

X. Around and about this paper

It was a deliberate choice to write a paper on numerical methods for large linear algebra problems without referring much to the numerical aspects and without presenting numerical tests of the algorithms. Instead, the *geometrical aspects* were highlighted. One of the other goals was to refrain as much as possible from the specialized terminology that has become part of the daily vocabulary of the numerical algebraist, and which might be less common for the reader with a general mathematical background. For such readers, it seemed better to talk about a method projecting on a certain type of subspace and augmented with a clever feedback mechanism, then to call it the *Implicitly Restarted Arnoldi Algorithm.*

Naturally, in practice, also topics like implementation, numerical stability, the effects of finite precision arithmetic, convergence speed, and computational costs are of interest and of importance. The reader who is interested in those aspects is referred to the literature.

Rough historical overview

The idea to project a linear algebra problem on a subspace goes back to Rayleigh, Ritz, and Galerkin at the beginning of this century. In 1950-51, Lanczos and Arnoldi proposed to use Krylov subspaces to project upon. In both cases, though, the algorithms were meant as a way to reduce *the whole matrix* to Hessenberg or tridiagonal form. It took until the seventies before it was recognized by Saad, Kaniel, Page and others that the intermediate projected matrices could be considered as useful approximations of the unprojected ones. Linear system algorithms among this category are *Conjugate Gradients* and the *Full Orthogonalization Method*, while the eigenvalue algorithms are named after Lanczos and Arnoldi.

It should be mentioned here that apart from finding the approximation from the Krylov subspace that gives a residual *orthogonal* to that space (those are the methods treated in this paper), there is another important class of methods that aims to find the approximation from the Krylov subspace that *minimizes* the residual. Clearly, expanding the subspace then gives by construction a smaller residual. For non-Hermitian linear systems, the *Generalized Minimal Residual Method* is the best known. Those methods were developed in the early eighties.

Yet another class of subspace methods arises if *bi-orthogonality* is exploited. The Krylov subspace belonging to A^H has interesting features when used in combination with the space for A itself: methods that aim to find an approximation from the one space, such that the residual is orthogonal to the other, give (when carefully worked out) tridiagonal projected matrices, even if A is not Hermitian. Those methods too, have nice geometrical interpretations in the sense that the two spaces, of course, coincide for Hermitian matrices, and *bifurcate* as A^H *continuously moves away* from A. The methods fall into the class of *Petrov-Galerkin* methods.

In the late fifties, Rutishauser developed a predecessor of the QR-algorithm, and in particular Wilkinson developed the convergence theory for the actual QR-algorithm halfway the sixties. Only in 1992, Sorensen found the connection between the QR-algorithm and restarting the projection algorithm of Arnoldi, which resulted in what is now widely known as the *Implicitly Restarted Arnoldi Method*.

Important contributions to stability issues for eigenvalue problems are due to Bauer and Fike in 1960 and Henrici in 1962. The region in the complex plane where the resolvent R has norm less than ε is called the *ε-pseudospectrum* of the matrix. This concept is due to Trefethen in 1992.

The algorithms based on the inexact solution of Riccati equations are very recent. Neglecting the quadratic term leads to the *Jacobi-Davidson algorithm* (1996), which is named after combining the old orthogonal correction ideas of Jacobi (1845-1846) with the subspace expansion idea based on residuals by Davidson (1975). The projected Picard iteration approach leads to the *Riccati algorithm* (2000). The theory behind the Riccati equation in connection with invariant subspaces goes back to Stewart (1973), who derived perturbation theorems and stability results for invariant subspaces from it.

Instead of giving an exhaustive bibliography, we prefer to mention a few good and modern textbooks, which are, with a single exception, all from the last ten years. In particular [5] treats almost all topics touched in this paper and gives many links to the research literature. For stability issues, we advice [3], [4],[6] and [8]. For an easy and very readable further introduction to general topics in numerical liner algebra, try [11]. For eigenvalue problems, [7] and [9] are modern and readable textbooks. Apart from those books, which do not include the recent developments, we mention references [10] and [2], and the book [1] on the Riccati equation.

About the author, acknowledgments

Since July 1999, Jan Brandts is *Research Fellow of the Royal Netherlands Academy of Arts and Sciences* (KNAW), and performs his research in the field of *Algorithms for and Stability of Very Large Eigenvalue Problems*. Apart from that, he investigates *Finite Element Methods* and *Boundary Element Methods*, which are projection methods for partial differential equations and integral equations respectively. The support of the KNAW is gratefully acknowledged. Also, the detailed comments and suggestions of NAW editors André Ran and Chris Zaal on how to improve the presentation of an earlier version of this manuscript, are very much appreciated.

Author's address: Mathematics Institute, Utrecht University, P.O.Box 80.010, 3508 TA Utrecht, The Netherlands. E-mail: brandts@math.uu.nl.

References

[1] S. Bittanti, A.J. Laub, and J.C. Willems, Eds. (1991). *The Riccati Equation*,

Communications and Control Engineering Series, Springer-Verlag, Berlin.

[2] J.H. Brandts (2000). A Riccati Algorithm for Eigenvalues and Invariant Subspaces, *Preprint nr. 1150 of the Department of Mathematics, Utrecht University, Netherlands.*

[3] F. Chatelin and V. Frayssé (1996). *Lectures on Finite Precision Arithmetic*, SIAM Publications, Philadelphia, PA.

[4] I. Gohberg, P. Lancaster and L. Rodman (1986). *Invariant subspaces of matrices with applications*, Wiley, New York.

[5] G.H. Golub and C.F. van Loan (1996). *Matrix Computations (third edition)*, The John Hopkins University Press, Baltimore and London.

[6] N.J. Higham (1996). *Accuracy and Stability of Numerical Algorithms*, SIAM Publications, Philadelphia, PA.

[7] B.N. Parlett (1998). *The Symmetric Eigenvalue Problem*, reprinted by SIAM Publications, Philadelphia, PA.

[8] G.W Stewart and J.G. Sun (1990). *Matrix Perturbation Theory*, Academic Press, London.

[9] Y. Saad (1992). *Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms*, John Wiley and Sons, New York.

[10] G.L.G. Sleijpen and H.A. van der Vorst (1996). A Jacobi-Davidson iteration method for linear eigenvalue problems, *SIAM J. Matrix Anal. Applic. 17*, pp. 401–425.

[11] L.N. Trefethen and D. Bau III (1997). *Numerical Linear Algebra*, SIAM Publications, Philadelphia, PA.