An application of the Lanczos Method

Travis Hoppe Drexel University

In this paper a practical implementation of the Lanczos method is applied to a real-world quantum mechanical system. A two dimensional banded symmetric Hamiltonian is introduced and the first ten eigenpairs are plotted. A brief overview of the Lanczos procedure is also given to elucidate the inner workings of the process.

Introduction:
In the experimental design of studied by Yang, we were introduced to a circuit consisting of two distinct sections, the Joseph-Johnson coupling and the traditional LC circuit. The Hamiltonian can be expressed through the traditional application of Kirchhoff's laws yielding:

$$H = H_{JJ} + H_{LC} + H_{coupling}$$

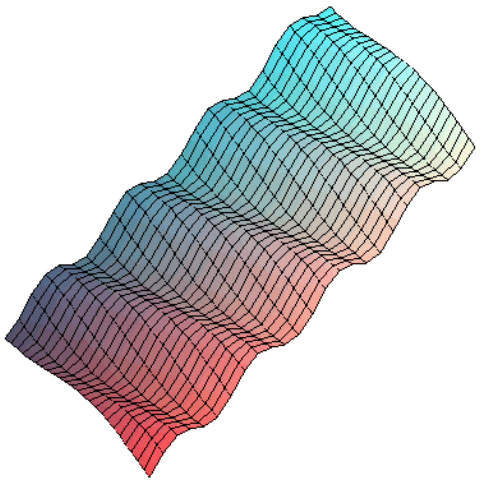
$$H_{JJ} = \frac{p_1^2}{2m_1} + V(\gamma_1)$$

$$H_{LC} = \frac{p_2^2}{2m_2} + \frac{m_2\omega_2\gamma_2^2}{2m_2}$$

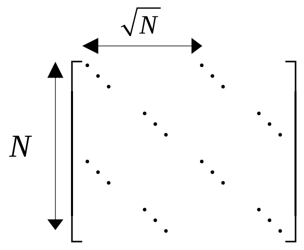
$$H_{coupling} = \varepsilon \frac{p_1 p_2}{\sqrt{m_1 m_2}}$$

The potential, governed by the Joseph-Junction is the so-called 'washboard' potential, due to it's shape.

$$V(\gamma) = \frac{-\phi_0}{2\pi} I_c (I_r + \cos\gamma_1)$$



The Hamiltonian, a function of the two variables γ_1, γ_2, can be mapped conveniently onto a matrix. The expectation value of the this operator <Ψ|H|Ψ > will lead to a corresponding eigenpair. We can map the gird of H using a simple finite difference equation, onto a vector whose elements are defined as, {Ψ00, Ψ10, ..., Ψn0, Ψn1, ..., Ψnn}. When the corresponding eigenvector is found, these elements are then mapped back onto their corresponding gird locations. Schematically, mapping Yang's potential onto H has the form of:



The problem now has been reduced to the standard linear algebra form of finding selected eigenpairs. Those with the smallest eigenvalues are of the most physical interest, as they give the probability density of the ground state and the lowest corresponding excited modes. For notational convenience and consistency with mathematical texts, from this point onward the matrix representing the Hamiltonian will be designated as **A.** Our problem can simply be stated as thus:

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

Iterative Power Method
To fully appreciate the Lanczos routine, it helps to step back and get a perspective on iterative matrix procedures and subsequently, the Krylov subspace. To begin with, we look at the power iteration as a stepping stone to the Lanczos procedure.

In the power iteration, the dominant eigenvalue and it's corresponding eigenvector can be found by simply applying the matrix repeatedly onto any starting vector.

$$\text{choose random } b$$
$$\lim_{n\to\infty} \mathbf{A}^n b = \lambda_{\max}^n v$$

At each successive iteration of the algorithm, the eigenvalues become completely dominated by the largest eigenvalues, hence only the largest survives. While interesting in its own right, the speed of convergence is slow [O(n^2) for each iteration] but can be expedited greatly if the matrix is sparse. Seeking more flexibility we can calculate the smallest eigenvector by taking the inverse of **A**, that is:

$$\lim_{n\to\infty} (\mathbf{A}^{-n}) b = \lambda_{\min}^n v$$

These ideas can even be extended further to calculate the eigenvector of *any* eigenvalue, if the value is known approximately in the spectral region, by employing a clever shift of the matrix:

$$\lim_{n\to\infty} (\mathbf{A} - \mu \mathbf{I})^{-n} b = \lambda^n v'$$

It must be noted that the eigenpair found corresponds to **A** only through a relation of the Rayleigh quotient.

From Krylov subspaces and Rayleigh-Ritz vectors to the Lanczos procedure:
Based off the power iteration, the Krylov subspace is spanned by some initial vector v:

$$\{v, \mathbf{A}v, \mathbf{A}^2 v, ..., \mathbf{A}^n v\}$$

The relation between this subspace and the power iteration is clear. In the Krylov subspace most symmetric matrices can be represented as a tridiagonal matrix $\mathbf{Q}_\mathbf{m}^* \mathbf{A} \mathbf{Q}_\mathbf{m}$, where **Q** is simply the Krylov basis after the Gram-Schmidt orthonormalizing process[1]. This is wonderful news if this new subspace is invariant to our original one, i.e. the eigenpairs in the Krylov subspace are the same as those in our original matrix. Alas, the Krylov basis only approximates the original one due to finite precision, but the relation between the two can be found through the Rayleigh-Ritz procedure. The eigenpairs found in $\mathbf{Q}_\mathbf{m}^* \mathbf{A} \mathbf{Q}_\mathbf{m}$ are now approximations **A**, which get better as the size of our basis *m*, increases. The heart of the Lanczos procedure is to combine the two techniques so that the original matrix is 'reduced' to a tridiagonal one.

The tridiagonal matrix we will be forming at each step of the Lanczos procedure $\mathbf{T}_\mathbf{m} = \mathbf{Q}_\mathbf{m}^* \mathbf{A} \mathbf{Q}_\mathbf{m}$, is **A**'s projection into the Krylov subspace. The diagonal and super/sub diagonals of **T** are designated as α_i, β_i respectively. A verbose way of stating the iterative procedure that comprises the Lanczos procedure is:

$$\mathbf{A}\mathbf{Q}_\mathbf{j} - \mathbf{Q}_\mathbf{j}\mathbf{T}_\mathbf{j} = \mathbf{r}_\mathbf{j}\mathbf{e}_\mathbf{j}^* \qquad \mathbf{r}_\mathbf{j} = \mathbf{q}_{\mathbf{j+1}}\beta_j$$
$$\mathbf{Q}_\mathbf{j}^*\mathbf{Q}_\mathbf{j} = \mathbf{I}$$

Expanding this out to pseudo-code:

```
Choose initial $r$
$\beta_0 = \|r\|$
$q_0 = \mathbf{O}$
$q_1 = r / \beta_0$
Loop for j = 1,2,…m
    $u = \mathbf{A} q_j$
    $r = u - q_{j-1}\beta_{j-1}$
    $\alpha_j = q_j^* r$
    $r = r - q_j \alpha_j$
Reorthogonalize( r )
    $\beta_j = \|r\|$
    $q_{j+1} = r / \beta_j$
```

r need not be random, in fact choosing a starting vector close to desired eigenvectors speeds convergence.

This matrix/vector multiply is the most computationally expensive step, and hence the limiting factor

The orthogonalization can be done in many different ways, running the gamut from none, partial/selective to full.

There are several key points to notice about the algorithm. The first is that only four vectors r,u,q[j],q[j-1] need to be keep in fast storage[2]. Also, our rating limiting step, the matrix/vector multiply can be considerably reduced if the matrix is sparse.

[1] Simply put, this forces all vectors to be orthogonal to each other

[2] It can be done with only two vectors, but comes at a cost of computation

The orthogonalization step has been downplayed as the great failure of the Lanczos procedure. In finite arithmetic, round off errors will force the product $\mathbf{Q}_j^* \mathbf{Q}_j$ to diverge from the identity, which in turn makes in the eigenvectors of **T** diverge. Since one of the key points of eigenvectors is their orthogonality, this represents a serious source of error. Fortunately there are remedies. The solution chosen for this implementation was the slowest, but most accurate, known as the Gram-Schmidt orthogonalization. This can be done simply by looping at each iteration.

$$r_j = r_j - \sum_{v=1}^{j} q_v (q_v^* r_j)$$

As you can see, this get prohibitively more expensive as you attempt to evaluate greater eigenstates.

When an appropriate number of iterations m, have been preformed, you are left with an mxm tridiagonal matrix **T.** These eigenvalues, θ, are approximate eigenvalues, λ ,to **A.** The eigenvectors can be found by:

$$\mathbf{x}_i^{(m)} = \mathbf{Q}_m \mathbf{s}_i$$

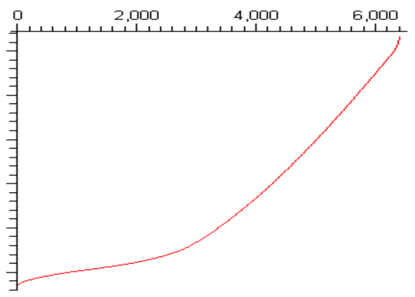
Where **s** is the matrix containing the eigenvectors of **T.** If one were to watch θ, as the iterations proceeded, you would find that the eigenvalues that were converging were the extremal ones. This is a property of the Krylov subspace, and was evident from our Power Iteration example.

Problem setup:
A general Lanczos produce was written in C++ using the above algorithm [Appendix]. In this implementation, all the end user must supply is the matrix/vector routine, everything else in the Lanczos procedure is self-contained. Full orthogonalization was chosen due to it's ease of implementation and high accuracy.

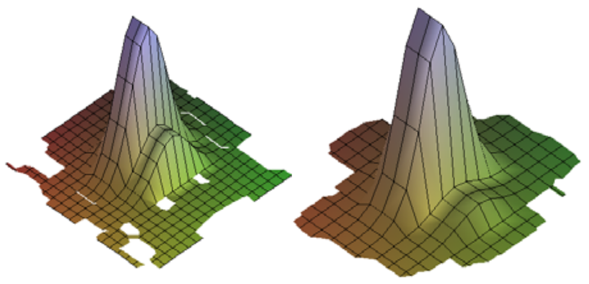
For the banded matrix produced in our problem, calculation of the matrix/vector product required roughly o(6j) ops rather than the traditional o(j^2). Also the Hamiltonian is never explicitly sorted in memory and is only represented as a function call to save memory access.

Numerical Results:
As with all good intentions, the end result of the computation lead down a dark path. The reasoning behind its failure was extremely pedagogic, but unfortunately the eigenvectors that were sought after could not be obtained to any great accuracy.

The Lanczos algorithm was first coded in MAPLE [Appendix], to serve as a check to any advanced code. As a check against the Lanczos routine, the matrix was solved via brute forced for all of its eigenpairs. All results, unless noted, will be from the calculations obtained from the exact solutions. The Lanczos routines, in both MAPLE and C++, produced identical results, giving confidence to the successful implementation of the algorithm. However, only the ground state eigenvector converged for the Lanczos method using a reasonable step size. The reason for this is clear we look at the eigenvalue spectrum:



The Lanczos algorithm is a convergent one. The convergence properties of the Krylov subspace suggest the eigenvalues to converge first are the extremal ones. The convergence is not uniform nor completely understood. Standard practice seems to suggest that given an eigenvalue spectrum, where one end is clustered and the other end diffuse, the diffuse end will converge first. In Yang's potential, it is evident that the spectrum of eigenvalues should converge to the upper eigenpairs first. Considering the physics of the problem, this is not optimal as we desire the lower level eigenstates. If a full iteration is preformed where the Krylov subspace spans that of the original subspace, the eigenpairs do match up, but the computational gain makes the endeavor worthless.

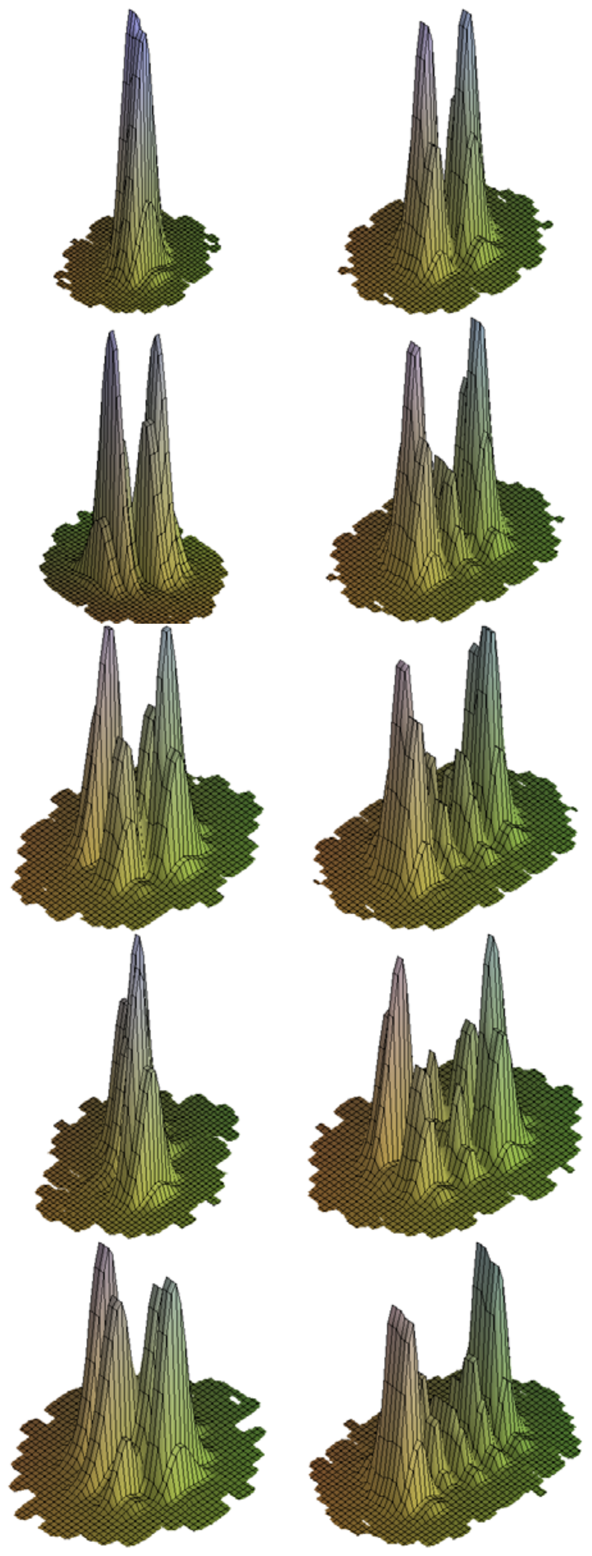


Ground state probability density on a 20x20 grid showing the Lanczos (left) and exact (right).

In an extended project, there is hope to salvage the Lanczos routine to converge to the lower eigenvalues. Extensive work has been done to work on a shift matrix, one where the spectrum of the eigenvalues have been rotated around a point in the space. Roughly, this has the effect of an inverse operation on the spectrum, close values will become separated, etc... However, this new shifted matrix H', is not invariant to our original matrix, and introduces a host of new orthogonality issues.

Physical Results:

While this implementation of the Lanczos routine clearly needs refinement, we can still push the limits of the computation by traditional methods. MAPLE, while not the most efficient code, has the advantage of it's simplicity in coding and results. The two-dimensional washboard potential has a host of interesting features. Approximately, the results could be interpreted as a system of two-dimensional harmonic oscillators of varying heights. Obviously this is a simplified picture, but if we approximated these eigenstates as harmonic oscillator potentials (as Yang did in his initial calculation) the results should have a correspondence. Yang's approximations then serve as a guide to model the washboard potential and seem to give decent qualitative results. The quantitative solutions however are far more interesting. The probability densities for the first ten states are shown below in left to right order using an exact solution with an 80x80 grid size (6400x6400 matrix).

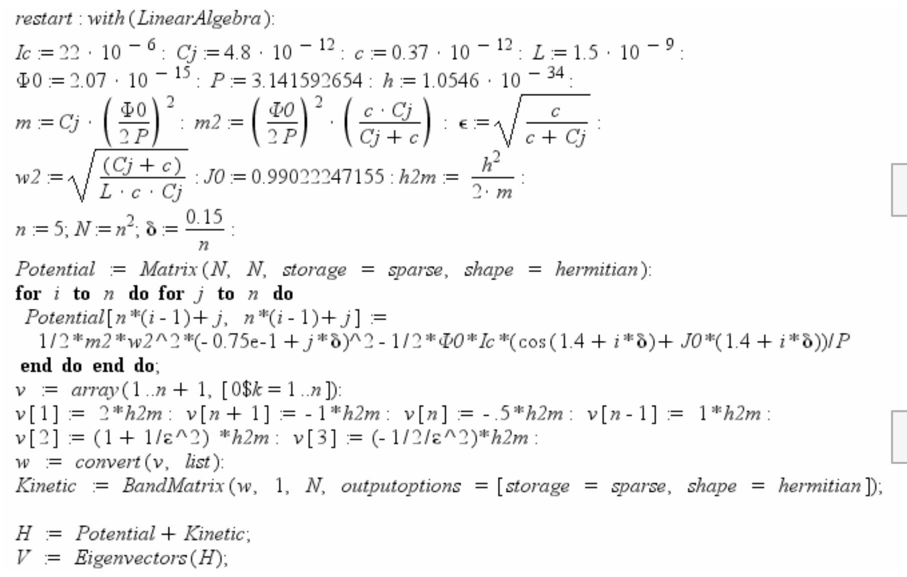


References:

Gilmore Quantum I,II Class Presentations, Yang, Yi
The Symmetric Eigenvalue Problem, Parlett
Matrix Computations, Golub
The Lanczos Method, Komzsik

Appendix

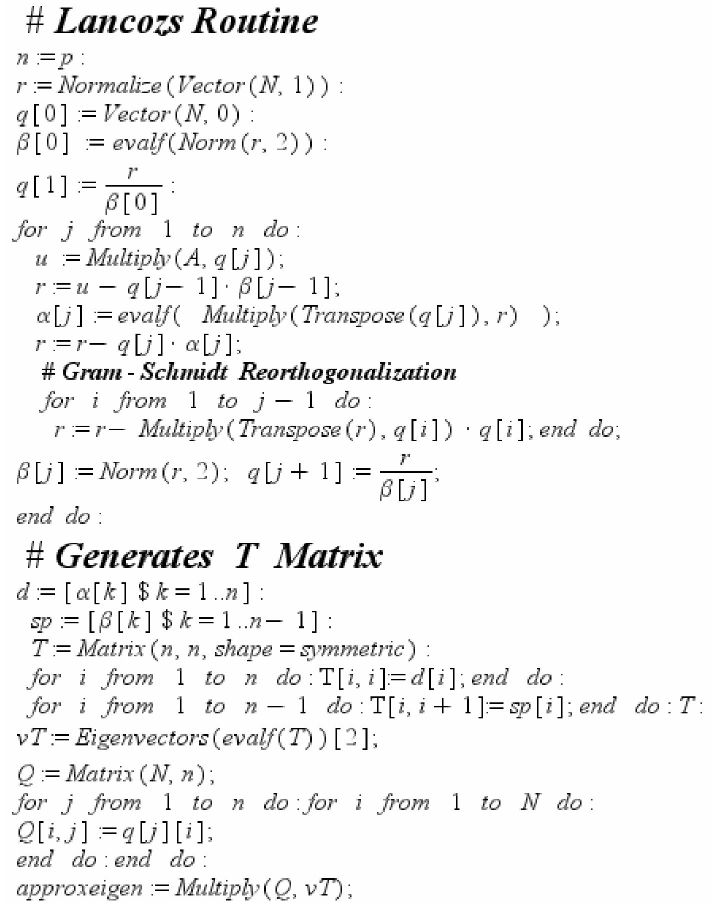
MAPLE Brute force/Exact solution:



```
restart : with(LinearAlgebra):
Ic := 22 · 10^(-6) : Cj := 4.8 · 10^(-12) : c := 0.37 · 10^(-12) : L := 1.5 · 10^(-9) :
Φ0 := 2.07 · 10^(-15) : P := 3.141592654 : h := 1.0546 · 10^(-34) :
m := Cj · (Φ0/(2 P))^2 : m2 := (Φ0/(2 P))^2 · ((c · Cj)/(Cj + c)) : ϵ := sqrt(c/(c + Cj)) :
w2 := sqrt((Cj + c)/(L · c · Cj)) : J0 := 0.99022247155 : h2m := h^2/(2· m) :
n := 5; N := n^2; δ := 0.15/n :
Potential := Matrix(N, N, storage = sparse, shape = hermitian):
for i to n do for j to n do
 Potential[n*(i - 1)+ j, n*(i - 1)+ j] :=
   1/2*m2*w2^2*(- 0.75e-1 + j*δ)^2 - 1/2*Φ0*Ic*(cos(1.4 + i*δ)+ J0*(1.4 + i*δ))/P
 end do end do;
v := array(1..n + 1, [0$k = 1..n]):
v[1] := 2*h2m : v[n + 1] := - 1*h2m : v[n] := - .5*h2m : v[n - 1] := 1*h2m :
v[2] := (1 + 1/ε^2) *h2m : v[3] := (- 1/2/ε^2)*h2m :
w := convert(v, list):
Kinetic := BandMatrix(w, 1, N, outputoptions = [storage = sparse, shape = hermitian]);

H := Potential + Kinetic;
V := Eigenvectors(H);
```

MAPLE Lanczos Method:



```
 # Lancozs Routine
n := p :
r := Normalize(Vector(N, 1)) :
q[0] := Vector(N, 0) :
β[0] := evalf(Norm(r, 2)) :
q[1] := r/β[0] :
for j from 1 to n do :
   u := Multiply(A, q[j]);
   r := u − q[j− 1] · β[j− 1];
   α[j] := evalf( Multiply(Transpose(q[j]), r) );
   r := r− q[j] · α[j];
    # Gram - Schmidt Reorthogonalization
    for i from 1 to j − 1 do :
      r := r− Multiply(Transpose(r), q[i]) · q[i]; end do;
β[j] := Norm(r, 2); q[j + 1] := r/β[j];
end do :
 # Generates T Matrix
d := [α[k] $ k = 1..n] :
  sp := [β[k] $ k = 1..n− 1] :
  T := Matrix(n, n, shape = symmetric) :
  for i from 1 to n do : T[i, i] := d[i]; end do :
  for i from 1 to n − 1 do : T[i, i + 1] := sp[i]; end do : T :
vT := Eigenvectors(evalf(T))[2];
Q := Matrix(N, n);
for j from 1 to n do : for i from 1 to N do :
Q[i, j] := q[j][i];
end do : end do :
approxeigen := Multiply(Q, vT);
```