Notes on Jim's paper: PhL 10.16.04
Automatic Computation of Zeros of Bessel Functions ...
~ Aug 2000 SIAM SISC

2. The general method. The method depends on the existence of a 3-term recurrence relation like that shown in (2.1) where g(x) is independent of n. At this point, notice that the constant coefficients A,B and C are independent. If you look on page 782 of A&S you see that for all ortho polys, you do have this kind of relation with g(x) = x.

Where do recurrence relations come from? When you solve an ODE with a power series and require termination of the series, you get certain recurrence relations on the coefficients of the powers, and probably these are what lead to recurrence relations on the solution functions. I realize that I do not own a good book on ODE's, M&M is really all I have. But I am very familiar with recurrence relations in the world of special functions.

Now, it is possible to rescale the A,B,C coefficients to get the result shown in (2.3) where now the new C and A coefficients are no longer independent. I proved this to be true, and I show the way you must define the γ_n to make it so.

Now the recurrence relation set can be written in the form of a matrix equation with a simple symmetric real tridiagonal matrix (which is real symmetric, hence diagonalizable). I agree 100% so far with the development. The equation looks like a quantum mechanics $H\psi = E\psi$ eigenvalue equation, but there is an extra term as shown in (2.4) and (2.7). If you restrict the variable x to values x_i which are the zeros of $\phi_{N+1}(x)$, then you really do get a simple eigenvalue matrix equation and then the $g(x_i)$ are in fact the eigenvalues and the $\phi_n(x_i)$ are the eigenvectors. We must also make sure that $\phi_{-1}(x) = 0$.

So what is going on here? We are saying that the zeros of $\phi_{N+1}(x)$, which we call x_i, are the eigenvalues of this little equation. Therefore, maybe we can compute the zeros by solving this simple matrix equation somehow. Getting the eigenvalues of the matrix equation gets you the $g(x_i)$, and this is where you need the extra assumption that g must be invertible, so you can then get $x_i = g^{-1}[g(x_i)]$.

Example #1: $\phi_n(x) = \sin[(n+1)x]$ $\phi_{-1}(x) = 0$ as required
$g(x) = \cos(x)$ $\alpha = 1/2$ and $\beta = 0$.
$\phi_{N+1}(x) = \sin[(N+2)x]$

The zeros of $\phi_{N+1}(x)$ are located at $(N+2)x = i\pi$ so $x_i = i\pi/(N+2)$ $i = 0,1,2..N$

But suppose we did not know where the zeros were? We could solve the above equation for the eigenvalues $\cos(x_i)$ and from those numbers we could get the x_i .

How do you actually solve for the eigenvalues, call them $\lambda_i = \cos(x_i)$? You have to diagonalize the matrix and then those diagonal elements are the eigenvalues. Jim does not discuss how this is done in this example. I want to do a trivial case to see what happens.

Assume then the above #1 example with N = 1 so that we have a 2x2 matrix. The T matrix is then given by:

$$T = \begin{pmatrix} 0 & 1/2 \\ 1/2 & 0 \end{pmatrix}$$

Since this is a real symmetric matrix, it can be diagonalized by a similarity transformation of the form

$$Q = \begin{pmatrix} Cx & Sx \\ -Sx & Cx \end{pmatrix}$$ // This is the general form for an orthogonal Q

I did this by hand and found out that the eigenvalues are ± 1/2 and the eigenvectors are these:

$$f_0 = \begin{pmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} \qquad \lambda_0 = -1/2$$

$$f_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} \qquad \lambda_1 = 1/2$$

and Maple confirms these simple results using the eigenvectors(A) call. Now consider:

$\cos(x_0) = -1/2 \quad \Rightarrow \quad x_0 = 2\pi/3$
$\cos(x_1) = 1/2 \quad \Rightarrow \quad x_1 = 1\pi/3$

so in this indirect way, we can conclude that the zeros of the function

$\phi_{N+1}(x) = \sin[(N+2)x] = \phi_2(x) = \sin[(3)x]$ are at the values above, and we verify:

$\sin[3 * 2\pi/3] = \sin(2\pi) = 0$
$\sin[3 * 1\pi/3] = \sin(\pi) = 0$

and we have demonstrated, then, the idea of doing matrix algebra to find the zeros of a function! Notice that we never had to evaluate a sine function anywhere to get this result. But we did have to invert the g(x) function.

3. Application to Bessel functions. Here we follow the above plan. We write out the recursion formula, but for a reason not yet known to me, Jim iterates it to get a different one with 2 units spacing. He then defines the ϕ_n in the strange way shown in (3.2) and this gives our "canonical" form recursion relation where we have the α and β constants as he shows, and where $g(x) = 1/x^2$. As usual, we have our little extra vector with $\phi_{-1}(x)$ which turns out to be exactly a multiple of $J_\upsilon(x)$, part of his manipulations to this point. So if we choose $x = x_i$ as the zeros of the Bessel function, then this one extra thing goes away, as desired.

But what about the other piece? Jim argues that it is very small for large but finite N, and in that case he can use perturbation theory to find the adjustment of the infinite-N eigenvalue which is written as $1/x_i^2$. The eigenvalue after perturbation (ie, large but finite N) is called λ_i. He comes up with a formula for the difference and calls it R. This is a form of perturbation theory I don't remember doing, because you are not really perturbing your matrix (the Hamiltonian) as we do in Q.M, but you are adding a small vector to the eigenvalue equation. I don't know offhand where to find the theory for this, but let's assume Jim has done it right, and we then get the correction (3.8).

So far, however, I don't know either the uncorrected results $1/x_i^2$ or the correction R. Well, I guess the uncorrected results are that $x_i = (i + \upsilon/2 - 1/4)\pi$ since $J_\upsilon(x)$ acts as a sine for large index. This must be then what we correct with R. In computing R, he uses a fancy asymptotic expression for the J(x) and I think the overall answer for R is given by the right equation in (3.11). Thus, all you have to do is evaluate this R thing which contains a few square roots and an exponential.

So does it work? He first selects N = 100, which determines the size of his matrix (which is 101 x 101 in that case). This involves Bessel functions up to $\upsilon = 202$. He then considers the function $J_0(x)$ and looks at the Mth zero. He claims that this approximation for the first 55 zeros is accurate to 14 significant figures, but for larger M things get worse. So presumably if you needed larger M, you would pick a larger N (but he does not state this).

The work on page 6 is really estimating how accurate the result is. The actual computation is showin the FORTRAN code at the end. In that code I think he would set nt = 50 to get his 100x100 matrix more or less, since then nmax = 100. He installs all the α and β values, and then hands off the tridiagonal matrix to an eispack routine called tq12 which computes the eigenvalues of the matrix. This routine overwrites your input d vector with the computed eigenvalues, I guess to save space. So this TQL2 is where all the "real work" is done, the bump and grind. The eigenvalues are $\lambda = 1/x^2$ and so you just invert and takes square root to get your x_i. The results are then sorted and printed out.

So looking at the code on page 8 is very helpful. I verified that the α and β are correctly entered in terms of the f1 thing. The EISPACK is some dusty deck package of eigenvalue system routines, and includes one called TQL2 which is used. This routine uses "the QL method" to find the eigenvalues,

" The **QL** method is used on a tridiagonal matrix. The basic idea behind the **QL** method is that of decomposing the matrix **A** in the form

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{L}$$

where **Q** is *orthogonal* and **L** is *lower triangular*."

4. Zeros of other Special Functions

(1) <u>The Coulomb Wave Functions</u>. The variable is ρ instead of x. The index is L, and η is some second index. Things are scaled as usual and we get our α and β. In this application, the matrix ruins from $L = L_0+1$ up to some large L_{max}. $g(\rho) = 1/\rho$ in this case. Same idea.

(2) <u>Associated Legendre Functions</u>. Now we have a funny index $n = \upsilon - \mu$ and all the β's are 0. And we have g(x) = x, very simple. So in this case, the eigenvalues ARE the zeros.

My comments on this paper. This is somewhat of a "boutique" algorithm that applies in certain cases. It is very efficient in computation time. But the Newton method techniques always give the exact answers although they require (perhaps much) more compute time. So this algorithm is really only useful in some application where you have to compute the zeros perhaps in real time. But why not just pre-compute them by the old methods and put them in a table?

The paper does not say *how much* faster this method is than standard methods. The idea is nevertheless novel I think.

I wonder how on earth Jim got involved with this obscure topic? Nelson Beebe is a "research professor" in the math department. Interests range from TeX and PDF to Hartree-Fock physics stuff.