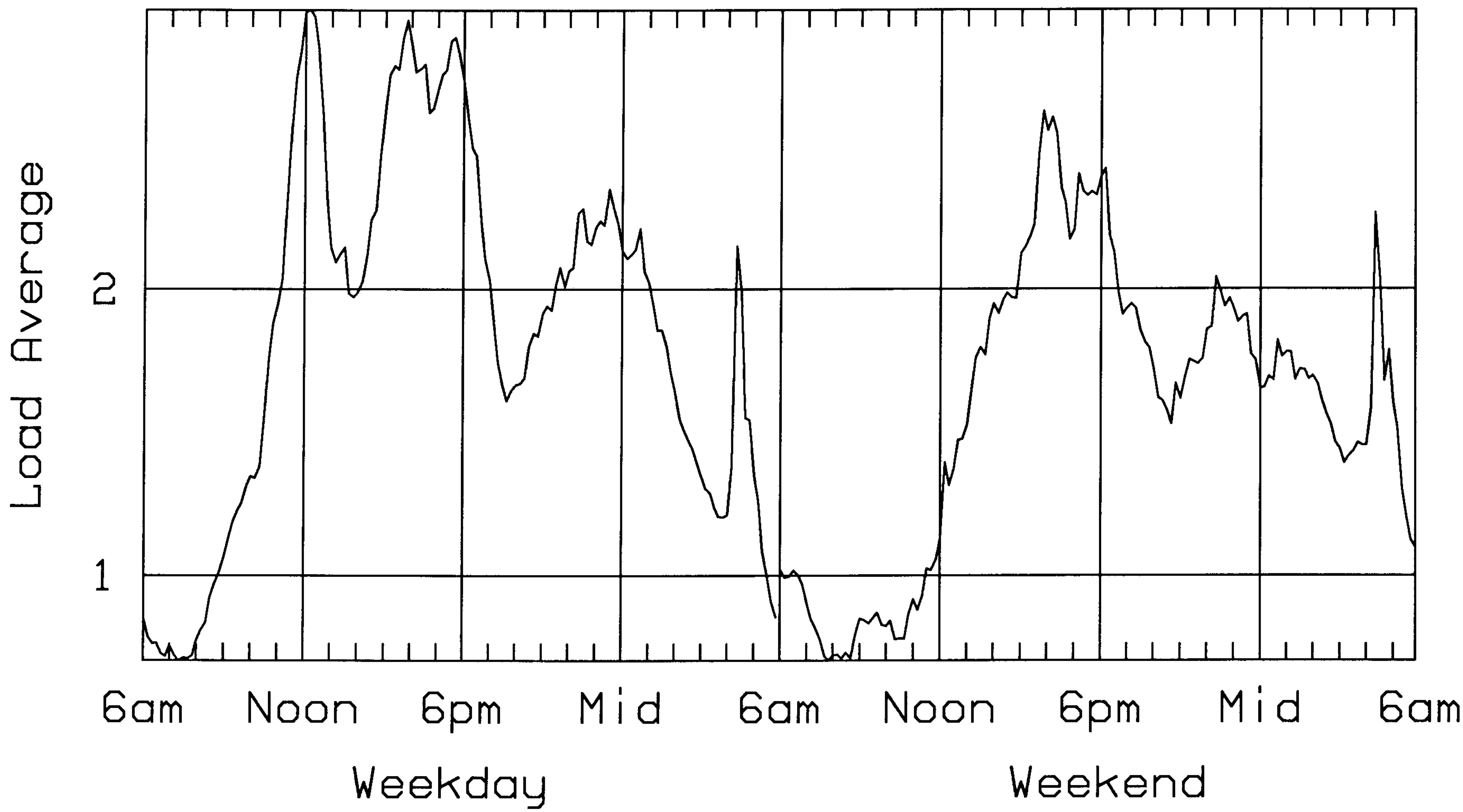
Tutorials

Jon F. Claerbout

INTRODUCTION

The tutorial articles included here are derived from material that appeared in earlier SEP reports. The NMO paper is a complete rewrite, for reference more than for tutorial purposes. The conjugate gradient paper is a revision of an earlier tutorial. The revision is only about 25%, but I deem it worth reprinting because of my belief that we should see increased use of this method.

Load average on Hanauma, sampled every 10 minutes and averaged over the last 2 months



System accounting is done at 4AM. People often wander over to the student union for dinner at around 5PM, or go home for a few hours and come back late in the evening. It is common practice to start something running before going off to lunch. Things pick up slowly after lunch. Quite a few people work on weekends, but not in the morning. Last chance to get a snack at the student union is around 11:30PM.

Pseudounitary NMO Tutorial

Jon F. Claerbout

It is often desirable to work with transformations that are as near as possible to being unitary, i.e. their transpose is their pseudoinverse. Such transformations will be called pseudounitary. The practical value is that iterative transformation, as may be a part of some estimation procedure (such as missing data or deconvolution), does not lead to continued degradation.

Let us make NMO into a pseudounitary transformation. With linear interpolation, the matrix operator defining NMO has two bands. The matrix $T = NMO^T \; NMO$ is tridiagonal, and we need to factor it into bidiagonal parts, say $T = B^T B$. Such factorization is called Cholesky decomposition. It is remniscent of spectral factorization. Then we'll define *pseudounitary* $NMO = UNMO = NMO \; B^{-1}$. To confirm the unitary property, examine $UNMO^T \; UNMO = B^{-T} NMO^T \; NMO \; B^{-1} = B^{-T} B^T BB^{-1} = I$.

(Consider $UNMO = NMO \; B^{-1}$. Time series experts will recall that an all-pass filter is a ratio of two terms, both with the same color, the denominator minimum phase and the numerator not. So Cholesky factorization gives the "minimum-phase" denominator.)

First notice that:

$$
T = \begin{pmatrix} b_1 & a_1 & & \\ a_1 & b_2 & a_2 & \\ & a_2 & b_3 & a_3 \\ & & a_3 & b_4 \end{pmatrix} = \begin{pmatrix} d_1 & & & \\ e_1 & d_2 & & \\ & e_2 & d_3 & \\ & & e_3 & d_4 \end{pmatrix} \begin{pmatrix} d_1 & e_1 & & \\ & d_2 & e_2 & \\ & & d_3 & e_3 \\ & & & d_4 \end{pmatrix} = B^T \; B
$$

Writing out the terms of the above product suggests a direct recurrence for the components of the bidiagonal matrix.

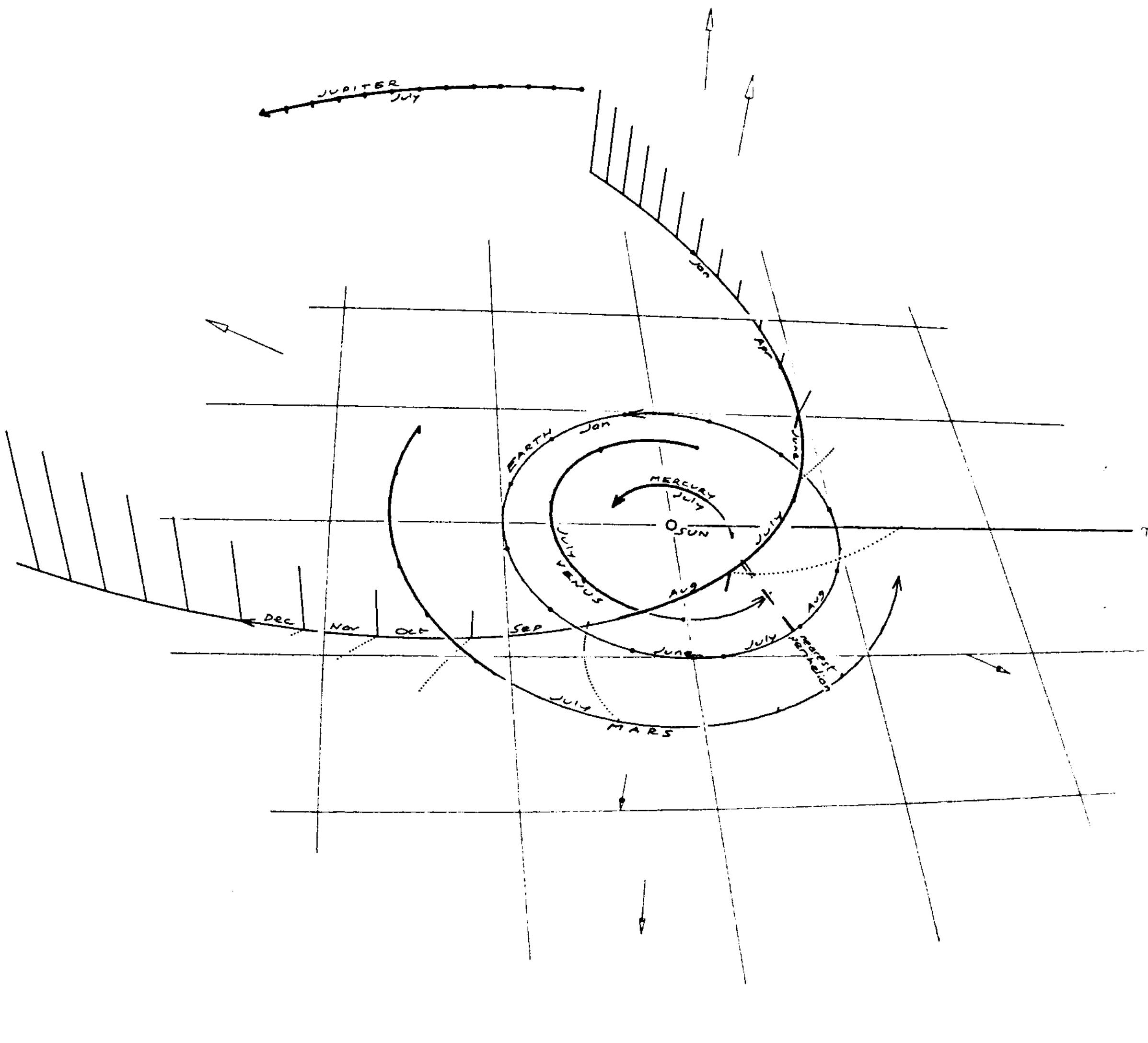
```
d(1) = sqrt(b(1))
do i=1, n-1 {
      e(i) = a(i) / d(i)
      d(i+1) = sqrt( b(i+1) - e(i)*e(i) )
      }
```

To apply B^{-1} to data use backsubstitution. The reason it is *pseudo*unitary is that a number of the matrix elements vanish. (These cluster at the upper left corner of the matrix). It can be confirmed that the components that cannot be computed because of an implied zero divide may simply be set to zero. For reference, a program is included below. To within machine accuracy, it passes the $(A'y)'x = y'(Ax)$ test for random values of x and y.

```
# trainv=0: zz(iz) = [pseudounitary NMO] tt(it)
# trainv=1: tt(it) = [pseudoinverse NMO] zz(iz)

subroutine unmoix( trainv, mktab, slow, x, t0, dt, nt, tt, zz,
                        itab, dd, ee, w0, w1 )
integer trainv, it, nt, iz, mktab, itab(nt)
real slow(nt), t0, dt, tt(nt), zz(nt), t, x, z, tm, tpart, xs, arg,
        w0(nt), w1(nt), dd(nt), ee(nt), bb(4000), cc(4000), zt(4000)
if( mktab == 1 ) {          # tabulate pointers and weights.
        do it = 1, nt {
                cc(it) = 0.
                bb(it) = 0.
                }
        z = t0 + nt * dt
        t = z

      do iz = nt, 1, -1 {
              xs = x * slow(iz)
              arg = z * z + xs * xs
              # next line replaceable by: t = sqrt ( arg )
              t = (arg + t * t) / (t + t)
              it = (t - t0) / dt + .00001
              tm = t0 + it * dt
              tpart = t - tm
              w0(iz) = (dt - tpart) / dt
              w1(iz) = 1. - w0(iz)
              itab(iz) = 0
              if ( it+1 <= nt ) {          # interior
                      itab(iz) = it
                      bb(it)   = bb(it)   + w0(iz) * w0(iz)
                      cc(it)   = cc(it)   + w1(iz) * w0(iz)
                      bb(it+1) = bb(it+1) + w1(iz) * w1(iz)
                      }
              else if ( it <= nt ) {       # at edge
                      itab(iz) = - it
                      bb(it)   = bb(it)   + w0(iz) * w0(iz)
                      cc(it)   = cc(it)   + w1(iz) * w0(iz)
                      }
              else                         # off end
                      itab(iz) = 0
              z = z - dt
              }
```

```
        dd(1) = sqrt( bb(1) )              # Cholesky factorization.
        do it = 1, nt-1 {
                if( dd(it) != 0. )
                        ee(it) = cc(it) / dd(it)
                else
                        ee(it) = 0.
                dd(it+1) = sqrt( bb(it+1) - ee(it)*ee(it) )
                }
        }

if( trainv == 0 ) {    # Operator itself
        if( dd(nt) != 0. )                 # Divide bidiagonal
                zt(nt) = tt(nt) / dd(nt)
        else
                zt(nt) = 0.
        do it = nt-1, 1, -1
                if( dd(it) != 0. )
                        zt(it) = (tt(it) - ee(it) * zt(it+1)) / dd(it)
                else
                        zt(it) = 0.

        do iz = 1, nt                      # Linear interpolate
                zz(iz) = 0.
        do iz = 1, nt {
                it = itab(iz)
                if( it > 0 ) {
                        zz(iz) = zz(iz) + w0(iz) * zt(it)
                        zz(iz) = zz(iz) + w1(iz) * zt(it+1)
                        }
                else if( it < 0 ) {
                        it = -it
                        zz(iz) = zz(iz) + w0(iz) * zt(it)
                        }
                }
        }

else {                 # pseudoinverse
        do it = 1, nt
                zt(it) = 0.
        do iz = 1, nt {                    # Linear interpolate
                it = itab(iz)
                if( it > 0 ) {
                        zt(it)   = zt(it)   + w0(iz) * zz(iz)
                        zt(it+1) = zt(it+1) + w1(iz) * zz(iz)
                        }
                else if( it < 0 ) {
                        it = -it
                        zt(it)   = zt(it)   + w0(iz) * zz(iz)
                        }
                }

        if( dd(1) != 0. )                  # Divide bidiagonal
                tt(1) = zt(1) / dd(1)
        else
                tt(1) = 0.
        do it = 2, nt
                if( dd(it) != 0. )
                        tt(it) = (zt(it) - ee(it-1) * tt(it-1)) / dd(it)
                else
                        tt(it) = 0.
        }
return;                end
```

Halley's comet, 2061:
Another too-symmetric visit, but the exact opposite of this year's. We won't come very close, but we get a good look at the comet when it is intrinsically at its brightest. If you have great faith in the progress of medical technology, you may be interested to know that in the NEXT apparition after this one, Halley's will come VERY close to Earth, halting our attempts to peer further into the future due to the large errors tiny uncertainties at that distant point in time create. In the other direction, we can extrapolate backwards in time all the way to 1404BC.



Reprinted with permission from Mankind's Comet, by Guy Ottewell and Fred Schaaf