C.C.Tscherning,
Geodetic Institute, Geodetic Dep. I.,
Nørre Farimagsgade 1,1; 1364 Cph. K.

2655

September 1970.

ALGOL - procedure potential.

KEY WORDS AND PHRASES: geophysics, spherical harmonics, recursion algorithm
CR CATEGORIES: 5.12

```
real procedure potential(GM,axis,n,C,theta,lambda,radius,dx,dy,dz);
comment References:
James, R.W.  Geophys. J.R.Astr.Soc.,(1969) 17,305.
Heiskanen, W.A. and H. Moritz, Physical Geodesy, 1967, Chp.1.

The procedure computes the value of a potential due to internal sources
and its gradient in a point in space. The point has spherical coordinates
theta (the complement to the geocentric latitude), lambda (the longitude)
and radius (the distance from the orign).

The potential must be given in a form usually used in the geophysical
sciences, V=GM×(sum(i),sum(j) of axis↑i/radius↑(i+1)×P[i,j](cos(theta))
×(a[i,j]×cos(j×lambda)+b[i,j]×sin(j×lambda))), where GM is the product
of the mass and the gravitation constant, axis is the equatorial axis,
P[i,j](cos(theta))  are the quasinormalized spherical harmonics and a[i,j],
b[i,j] are the coefficients of  the series. These must be stored on list
form in the array C: C[0]=a[0,0] (= 1 due to the selected representation),
C[1]=a[1,0], C[2]=a[1,1], C[3]=b[1,1] etc. C must be declared with the
lover bound 0 and the upper bound n↑2+2×n+4, where n is the highest degree
of the spherical harmonics used. (Quasinormalized means fully normalised,
multiplied by sqrt(2×i+1)).

The procedure might easily be modified so as to compute a potential due to
external sources or  higher derivates. If only the potential is to be
evaluated, James has given simpler formulas in earlier papers.

Remark, that logical constructions are avoided by means of declaring
arrays a bit bigger than nescessary and multiplying by zero;

value theta,lambda,radius,GM,axis,n; integer n;
real theta,lambda,radius,GM,axis,dx,dy,dz; array C;
begin
     integer i,iplus1,j,jplus1,iplusj,iminj1,s,n2;
     comment Variables containing i or j and s control the recursion
     algorithm;

     real u,v,w,pot,pot0,dx0,dy0,dz0,
     alfa,alfa2,beta,gamma,ala0,alb0,bea2,beb2,
     a0,a1,a2,da0,da1,da2,b0,b1,b2,db0,db1,db2,aj,bj,
     factor,adivr,adivri,arfac,
     sq2,d,d0,d1,d2;

     comment description of the variables.
     line 1: u,v,w coordinates of a unit vector in the direction of the point
             of evaluation. pot,pot0 etc. holds the partial sums of
             potential,dx,dy and dz.
     line 2: coefficients in the recursion formulas see R.W. James (4).
     line 3: a0 - a2, b0 - b2 etc. hold  the 3 actual variabels in the
             recursion formulas.
     line 4-5: working variabels (axis divided by radius etc.);

     real array A,B[0:n+2],F,G[0:2×n+3];
     comment A and B hold the (quasinormalized) spherical harmonics
     of degree i  (multiplied by factorial) in the point of evaluation.
```

In F and G are stored the coefficients of the recursion formulas.
If the procedure is to be evaluated several times, G and F should be declared and computed outside the procedure;

```
n2 := n+2;
for i :=0 step 1 until n2 do A[i] := B[i] := 0; n2 := 2×n+3;
for i :=0 step 1 until n2 do
begin F[i] := sqrt(i×(i-1)); G[i] := sqrt(i);
end;

u := sin(theta); v := sin(lambda)×u;
u := cos(lambda)×u; w := cos(theta);
sq2 := sqrt(2); adivr := axis/radius;
pot := dx := dy := dz := 0;
A[0] := factor := adivri := 1;

for i :=0 step 1 until n do
begin
   a1 :=b1 :=b2 :=da1 :=db1 :=db2 :=dx0 :=dy0 :=dz0 :=pot0 :=d :=0;
   s := i↑2; d2 := d1 := 1; d0 := sq2;
   a2 := A[0]; da2 := C[s];
   iplusj := i; iplus1 := i+1; iminj1 := i+2; s := s+1;
   arfac := adivri/factor;

   for j :=0 step 1 until iplus1 do
   begin
      iplusj := iplusj+1; iminj1 := iminj1-1; jplus1 := j+1;
      alfa := F[iplusj]×d1; alfa2 := alfa×d2;
      beta := F[iminj1]×d0; gamma := G[iminj1]×G[iplusj];

      pot0 := pot0+a2×da2+b2×db2;

      a0 := a1; a1 := a2; a2 := A[jplus1];
      b0 := b1; b1 := b2; b2 := B[jplus1];
      ala0 := alfa2×a0; bea2 := beta×a2;
      alb0 := alfa×b0;  beb2 := beta×b2;
      A[j] := aj := (u×(ala0-bea2)-v×(alb0+beb2))/2+gamma×a1×w;
      B[j] := bj := (u×(alb0-beb2)+v×(ala0+bea2))/2+gamma×b1×w;

      da0 := da1; da1 := da2; da2 := C[s];
      db0 := db1; db1 := db2; db2 := C[s+1];
      ala0 := alfa2×da0; bea2 := beta×da2;
      alb0 := alfa×db0;  beb2 := beta×db2;
      dx0 := dx0+(ala0-bea2)×aj+(alb0-beb2)×bj;
      dy0 := dy0+(ala0+bea2)×bj-(alb0+beb2)×aj;
      dz0 := dz0+gamma×(da1×aj+db1×bj);

      s := s+2; d1 := 1/d0; d2 := 2-d; d0 := d := 1;
   end j-loop;

   comment the contributions from the degree i are accumulated. Other
   summation strategies may be convenient, depending on the numerical
   properties of the coefficients. Remark, that the gradient is posi-
   tive outward;

   pot := pot+arfac×pot0;
   factor := factor×iplus1; arfac := adivri/factor;
   dx := dx-arfac×dx0; dy := dy-dy0×arfac; dz := dz-dz0×arfac;
   adivri := adivri×adivr;
 end i-loop;
 potential := pot×GM/r; a0 :=GM/radius↑2;
 dx := dx×a0/2; dy := dy×a0/2; dz := dz×a0;
end potential;
```