

multifit 6.6.70

begin

```
real pi, mod, recmo, G, k, m0, a, c, Ls, Rs, Ms, lTes, L, Te, R, lTe, lL, lR, L0, LV,  
eps, kappa, arb, arbl, U1, U3, cnsqm, epsp, epsn, Lneu, eps4, epsg, dt, dtold,  
testg, testn, q, T, rho, P, T4, qcnew, qcc, Xcc, qout, Xout, beta,  
age, dage, dXmax, DXcc, M, X, Y, Z,  
XomH, YomHe, ZomA, eomua, eomue, tb, rb, sec,  
A1, A2, A3, A4, A5, A6, A01, A02, A08, A09, Atim4, Alp5,  
delta, dd, ddr, dda, gr, Gsum, EGsum, Gsumtot, F, dpar, step, mf,  
gacc, alfa, l, Hp, U, si, t1, tstop, ltb, lpb, slT, slP, C1, C2, C3, C4, dummy, exp  
integer s, t, i, j, nQ, t4, tælqx, ar, tgr, tgrn, argr, grgæt, energi, nfp, fit, nm, l  
UNRW, UNWW, UNWP, UNWD, runnum, nummax, sernum, sermax, almcount, impcount,  
gQ, nDX, nDXfit, intil, autm, mtx4, ccore, two, three, four, five, num, rat, lat  
mode, ztype, ilt, ilp, tsg, tsd, lineout, defout, clout, cdout;  
array k1, y[1:6], Q, XQ, DX, QDX[0:99], XM[1:4], km[1:4, 0:14, 0:20],  
fplkap, ddaalHJU[0:30, 0:20], ara, rpfac, alter[1:4],  
yik[0: 59, 1:4], Gik[1: 20, 1:4], mk[0:20], dGdy[1: 20, 1:4, 1:4];
```

```
switch SW:=timestep, choose, ldef, tdef;
```

```
integer procedure lyn; lyn:=0;
```

```
integer procedure read integer; begin integer a; input(a); read integer:=a end;
```

```
real procedure read real; begin real a; input(a); read real:=a end;
```

```
boolean procedure kbou; kbou:= false;
```

```
procedure writechar(a); value a; integer a; if a=0 then writetext(⟨< ⟩);
```

```
integer procedure select(a); value a; integer a; select:=a;
```

```
real procedure antlog(r); value r; real r; antlog:= exp(r/mod);
```

```
real procedure log(r); value r; real r; log:= mod×ln(r);
```

```
procedure READ LIST;
```

```
begin
```

```
writetext(⟨<
```

```
runnum, sernum, sermax, nummax,
```

```
autm, lineout, defout
```

```
l/Hp:=⟨⟩);
```

```
runnum:= read integer; sernum:= read integer; sermax:= read integer; nummax:= read integer;
```

```
autm:= read integer; lineout:= read integer; defout:= read integer; alfa:= read real;
```

```
if autm ≠ 0 then begin writecr;
```

```
write (⟨-dddd⟩, runnum, sernum, sermax, nummax); writecr;
```

```
write (⟨-dddd⟩, autm, lineout, defout);
```

```
write (⟨-ddd.d⟩, alfa);
```

```
end end READLIST;
```

```
procedure ALARM(text, act); value act; string text; integer act;
```

```
begin s:= select (UNWW);
```

```
writetext (⟨<
```

```
ALARM: ⟩); writetext (text); if act = 1 then WRI;
```

```
dummy:= select(s); almcount:= almcount+1;
```

```
if almcount>10 then goto end ser;
```

```
end ALARM;
```

```
procedure MERSN;
```

comment procedure MERSN performs all integrations;

begin real x0, ho3, eps, eq, gstep; boolean last;
array y0, k3, k4, k5 [1:6];

procedure CRGR ;

comment procedure CRGR determines where transitions between radiative and convective zones occur;

begin integer t; real gtg, teste, qe, qg, qn;
t:= 0 ; gtg:= -testg; qe:= q; teste:= testn; qg:= x0;

iter: if abs(testn)>Atim4^t<15 then
begin if (t>12 ∨ kb on) ^ autm<two then WRI;
qn:=(testexqg-testgxqe)/(teste-testg);
for i:= 1 step 1 until nm do y[i]:= y0[i]; q:= x0; DIFF(k1);
RKM(qn-x0); t:= t+1; DIFF(k4);

if sign(testn)=sign(teste) then begin qe:= qn; teste:= testn end
else begin qg:= qn; testg:= testn end;
go to iter end;

if autm<two then begin WRI;

if mode = four then s:= select (UNWP); writechar (51);

if mode = four then dummy:= select (s) end; DIFF(k1);

if t=15 then ALARM(⟨<CRGR⟩, 1);

if abs(q-mk[intil])>2₁₀⁻⁷ then last:= false;

testg:= testn:= abs(testn)×sign(gtg);

if mode=four then begin if qcnew=0 ∨ qcnew>mk[fit] then begin qcnew:= q;

if qcnew<mk[fit] then begin dXmax:= DXcc:= y[6]/qcnew; nDX:=-1; end else y[6]:= 0;
end procedure CRGR;

procedure RKM (h); value h; real h;

begin ho3:= h/A3;

for i:= 1 step 1 until nm do y[i]:= k1[i]×ho3+y0[i]; q:= x0+ho3;

DIFF(k3); for i:= 1 step 1 until nm do y[i]:= (k1[i]+k3[i])/A2×ho3+y0[i];

DIFF(k3); for i:= 1 step 1 until nm do y[i]:= (k1[i]×.375+k3[i]×1.125)×ho3+y0[i];

q:=x0+h/A2; DIFF(k4);

for i:= 1 step 1 until nm do y[i]:= (k1[i]×A1p5-k3[i]×4.5+k4[i]×6)×ho3+y0[i];

q:= h+x0; DIFF(k5);

for i:= 1 step 1 until nm do y[i]:= ((k1[i]+k5[i])/A2+k4[i]×A2)×ho3+y0[i];

end RKM;

procedure RKMER(h); value h; real h;

begin x0:= q; for i:= 1 step 1 until nm do y0[i]:= y[i];

RKM(h); eps:= 0;

for i:= 1 step 1 until (if nm>four then four else nm) do

begin eq:= abs(k1[i]×A02-k3[i]×A09+k4[i]×A08-k5[i]×A01);

if eq>eps then eps:= eq end; eps:= eps×abs(ho3);

if delta>eps then tsg:= tsg+1 else

begin comment step unacceptable;

if kb on then writetext(⟨<.⟩); tsd:= tsd+1;

for i:= 1 step 1 until nm do y[i]:= y0[i]; q:= x0; go to QR

end;

end RKMER;

comment An integration step is taken;

if ztype=three then nm:= if mode=four then 6 else four;

R: last:=false;

```

if ztype=three ^ abs(step)>.05 then step:= sign(step)×.05;
try last: if ztype=three ^ abs(mk[intil]-q)≤abs(step) then
  begin last:= true; gstep:= step; step:= mk[intil]-q;end;
  if ztype=two ^ y[three]>tstop-A2 then begin if abs(y[three]-y0[three])>10-7
  begin if abs(step)>(q-x0)×(tstop-y[three])/(y[three]-y0[three])
  then begin last:=true; step:=(q-x0)×(tstop-y[three])/(y[three]-y0[three]);e
RKMER(step); DIFF(k1);
if sign(testn)= sign(testg) ^ abs(testn)>Atim4 then testg:= testn else CRGR;
if mode=four ^ qcnew≠0 then begin if ztype=three ^ q≤qout+0.05 then begin
  nDX:= nDX+1; QDX[nDX]:= q; DX[nDX]:= epsp+eps4;end; end;
if kb on v mode=four then WRI;
if ztype=two ^ last ^ abs(y[three]-tstop)>510-7 then go to try last;
if ztype>1 ^ last then begin step:=gstep; go to finis;end;
if ztype=1 ^ y[1]>A01 then go to finis;

comment In the last part of MERSN the new integration step is found;

if eps<delta×.01 then begin step:= A2×step; go to R end;
QR: step:= (delta/eps)×A02×A08×step; go to R;
finis: end MERSN;

```

```

procedure DIFF(K); array K;
begin integer mt,mr,tad;

```

```

comment In the first part of DIFF (if ztype<3) the derivatives
needed in the outer zone are computed;

```

```

if ztype<three then

```

```

begin integer t,p;
real tm, pn, m, n, cmml, cm0, cm1, cnml, cn0, cn1, x2, FP,HJU, PmPr;

```

```

procedure INTP (A,intp1,intp2); real intp1, intp2; array A;
comment INTP interpolates in the packed array A, and stores the results
in intp1 and intp2;
begin integer ir,sgn,ia,ib; real int1,int2;
i:= -2; intp1:= intp2:= 0;
for arb:= cmml,cm0,cm1 do
begin i:= i+1; j:= -2; int1:= int2:= 0;
for arbl:= cnml,cn0,cn1 do
begin j:= j+1; ir:= A[t+i,p+j]; sgn:= sign(ir);
ir:= abs(ir); ia:= entier(ir×Atim4);
int1:= int1+ia×arbl; ib:= ir-ia×10000;
if ib = 9999 then ALARM(⟨<INTP 9999⟩,1);
int2:= int2+sgn×ib×arbl;
end cn forstatement;
intp1:= intp1+int1×10-3×arb;
intp2:= intp2+int2×10-3×arb;
end cm forstatement;
intp2:= exp(intp2/mod);
end INTP;

```

```

procedure MIXING;
begin real U8,U16,fsi,fm,hs,s2;

```

```

comment Here d ln T/ d ln P is computed according to the mixing - length
theory with l:= alfa × Hp;

```

```

l:= alfa×Hp; U:= C3/eomua×Hp×T4/(T×HJU×FP×P/PmPr×kappa×rho×l×l);
U8:= 8×U/9; U16:=A2×U8; hs:= U8×(ddr-dda);
la: s2:= six×si; fsi:= s2×si+U8×s2+U16×U×si-hs;
fm:=A3×s2+U16×(si+U); si:= si-fsi/fm;
if abs(fsi/hs)>10-6 then go to la; dd:= dda+six×si+A2×U×si;

```

end mixing procedure;

P:= exp(q);

if ztype=1 then begin T:=Tex(0.5+0.75xy[1])¹0.25;y[three]:=ln(T) end else T:=exp(y[th

tm:=(y[three]xmod-ltb)/slT; t:= entier(tm);

pn:=(qxmod - (lpb + tmxslP))/slP; p:= entier(pn);

if ilp >twoxp then p:= p+1;

m:= tm -t; n:= pn - p ;

if t < 1 ∨ t > ilt -1 ∨ p < 1 ∨ p > ilp - 1 then begin ALARM(†<FPTAB SMALL†,1);

goto endser end;

cmm1:= m/A2x(m-1); cm0:= 1-mxm; cm1:= m/A2x(m+1);

cnm1:= n/A2x(n-1); cn0:= 1-nxn; cn1:= n/A2x(n+1);

T4:=T×T; T4:= T4×T4; PmPr:= P-a/A3×T4; INTP(fplkap,FP,kappa); rho:= PmPr/FP/C2/T;

dd:= ddr:= C1×kappaxP/(y[two]×T4);

if ztype=1 then K[1]:= kappaxP/gacc else

begin x2:= y[1]xy[1];

Hp:= x2×P/(y[two]×rho×gacc); INTP(ddaalHJU, dda, HJU); testn:=A1/ddr -A1/dda;

if ddr>dda then MIXING;

K[1]:= -Hp/(R×Rs);

K[two]:= -C4×x2×rho×Hp;

K[three]:= dd

end end

In the following second part of DIFF the derivatives needed in the inner zone are computed

else begin real V,r3,T616,T626,T646,T6,t,r,f0,f1,n,p,p1,mnul,m1,Pc,ks;

comment Now rho is determined through the equation of state;

if q<qcc then X:= Xcc else if q>qout then X:= Xout else

begin if q> Q[gQ] then begin for i:= gQ, i+1 while q> Q[i] do gQ:= i; end else

for i:= gQ, i-1 while q< Q[i] do gQ:= i-1;

X:= exp(XQ[gQ]); arb:= exp(XQ[gQ+1]); arb1:= Q[gQ];

X:= X + (q-arb1)/(Q[gQ+1]-arb1)×(arb-X); end;

Y:= 1 - X - Z;

XomH:= X/1.00801; YomHe:= Y/4.0028; ZomA:= Z/16.942;

eomue:= XomH + A2 × YomHe + Z/A2;

eomua:= XomH + YomHe + ZomA;

T:= exp(y[three]); P:= exp(y[four]); V:= T¹3; T4:= V × T;

beta:= A1 - a/A3 × T4/P;

t:= P×beta/(T×k/m0); arb:= sqrt(V); rho:= t/(eomua+eomue×F);

ks:= X+7×Z+A3; r3:= cnsqm×ks¹3; i:= 0;

DEGL: Pc:= sqrt(r3×rho/T)×rho; i:= i+1; if i>15 then ALARM(†<EQ. STATE†, 1);

t:= (P×beta+Pc)/(T×k/m0); f1:= rho;

f0:= rho/(9.0476₁₀-9×arb)×eomue; r:= (f0 - 30)/30;

F:= (((0.109744×r-0.165376)×r+0.151420)×r-0.462698)×r+3.296236)×r+5.187420;

rho:= t/(eomua + eomue×F);

if abs((f1-rho)/rho)>₁₀-5 then go to DEGL ;

V:= V/rho; dda:= A1/(A4-A1p5×beta×beta/(A4-A3×beta)); t:= modxy[three];

comment In the following statement the energy generation rate is determined;

if X<1₁₀-7 then eps:= eps4:= 0 else

begin T6:= T×₁₀-6; T616:= T6¹(1/6); T626:= T616¹2; T646:= T626¹2;

ks:= 0.1332₁₀9×sqrt(ks/V);

arb:= 2.378₁₀-16×(1+0.00922×T626)×exp(-A3×ks+99.96/T626 -30)×exp30×(X/(1-X-Z))¹2;

arb1:= sqrt(1+arb); arb:= (arb1+arb/A2-1)/arb;

epsp:= 4.199₁₀6×arb×(1+0.0123×T626+0.00781×T646+0.00067×T6)/T646×

exp(ks-33.809/T626)×rho×X¹2;

```

if T6<12 then eps4:= 0 else
eps4:= 4.4581027×(1+0.00274×T626)/T646×exp(7×ks-152.31/T626)×rho×X×Z;
arb:= 1/(1+2.4281016×X/(1+X)×(1+0.0041×T626)/T616×exp(A4×ks-102.64/T626));
eps:= epsp×(0.980+(0.964×arb-1.044)/(arb1+A3))+eps4×0.936;
epsn:=epsp+eps4-eps;
if T6<7 then eps:= if T6>6.5 then eps×(T6-A6) else eps/A2; end;
epsg:= 0; eps:= eps+epsg;

```

comment In the following the opacity is determined by interpolation in tables;

```

r:= mod×ln(rho); arb:= (r-rb-A3×(t-tb))×A5; arb1:= (t-tb)×A5;
mt:= entier(arb1); mr:= entier(arb);
if mr< 0 ∨ mr>ka-two then begin ALARM (⊥<KAP INTP⊥, 1);
mr:= if mr<0 then 0 else ka-two end;
tad:= if X<XM[two] then 1 else if X<XM[three] then two else three;
p:= (X-XM[tad])/(XM[tad+1]-XM[tad]); p1:= 1-p;
f0:= km[tad,mt,mr]×p1 + km[tad+1,mt,mr]×p;
f1:= km[tad,mt,mr+1]×p1 + km[tad+1,mt,mr+1]×p;
n:= arb-mr; mnul:= f0+n×(f1-f0);
f0:= km[tad,mt+1,mr]×p1 + km[tad+1,mt+1,mr]×p;
f1:= km[tad,mt+1,mr+1]×p1 + km[tad+1,mt+1,mr+1]×p;
m1:= f0+n×(f1-f0); n:= arb1-mt;
kappa:= exp((mnul+n×(m1-mnul))/mod);

```

comment Now the derivatives are easily written down;

```

r:= exp(y[1]); r3:= r3; arb:= Ms×M/r2;
K[1]:= U1×M/(r3×rho); K[two]:= Ms×M×eps/L0;
if mode=four then begin K[five]:= Ms×M×epsn/L0; K[6]:= epsp + eps4;end;
K[four]:= -G×arb×q/P/(A4×pi)×arb;
ddr:= U3/M×kappa×y[two]×L0×P/(q×T4); testn:= A1/ddr-A1/dda;
K[three]:= K[four]×(if ddr<dda then ddr else dda);
end diff for innerzone;
end procedure DIFF;

```

```

procedure UD(A, k); value k; integer k; array A;
begin integer i; writecr;
write(if k=four then ⊥-dddd⊥ else⊥-ddd.dddd⊥,A[1],A[two],A[three],A[four]);
end;

```

```

procedure OUT4(A, j,k); value j, k; integer j, k; array A;
begin integer i;
for i:= 1 step 1 until four do
ara[i]:= A[j,i]/(if k=four then Atim4 else 1); UD(ara,k)
end for OUT4;

```

```

procedure WRI;
begin if mode=four then begin clout:=clout + 1; if clout≠lineout then go to FIN
else clout:=0; s:=select(UNWP);end;

```

writecr;

comment Now a NORMAL OUTPUT LINE is produced;

```

write(⊥-d.dddd⊥, if ztype=three then q else y[two],
if ztype=1 then 1 else if ztype=two then y[1] else exp(y[1]-lR/mod)/Rs,
if ztype=three then y[two]×L0/(antlog(lL)×Ls) else 1,
log(T), A01×log(P), log(rho), log(kappa), log(ddr),
if ztype=1 then y[1] else if ztype=two then log(dda)
else (if eps>10-9 then log(eps) else 0),
if ztype=1 then 0 else if ztype=two then log(dd)
else (if eps4>10-9 then log(eps4×0.936) else 0), X);

```

if mode=four then dummy:=select(s);

FIN;

end procedure WRI;

```
procedure ATMOF(lTe, lL); value lTe, lL; real lTe, lL;
begin WTS;
lR:= A2X(lTes - lTe) + lL/A2; Te:= exp(lTe/mod); R:= exp(lR/mod); L:= exp(lL/mod);
gacc:= GXMxMs/(RxsRs)↑2; y[two]:= 1; C1:= U3xLsxL/M; C4:= 4xpi/gaccxG;
X:= Xout; Y:= 1-X-Z; tstop:= mk[nfp]/mod;
eomua:= X/1.00801+Y/4.0028+Z/16.942; C2:= kxeomua/m0;
if kb on then begin
writecr; write(↑-dd.dddd↑, M, lTe, lL, lR, log(gacc), 5040/TeX2↑0.25); end;
nm:= ztype:= 1; q:=(lpb + (log(Tex0.5↑0.25) - ltb)/slTxslP)/mod +0.01; y[1]:=si:= 0
adlP: DIFF(k1); if kb on then WRI;
if k1[1]<0.01 then begin q:= q+1; go to adlP end;
step:=A01; MERSN; if kb on then begin DIFF(k1); WRI end;

ztype:= two; nm:= three; y[three]:= ln(Tex(0.5+0.75xy[1])↑0.25); y[1]:= A1; DIFF(k1);
if kb on then WRI; step:= 0.05; MERSN; if kb on then begin DIFF(k1); WRI end;

y[1]:= ln(y[1]xRsxR); arb:= y[two];
y[two]:= antlog(lL)xLs/L0;
y[four]:= q; F:= A1; q:= arb; ztype:= three; Lneu:= y[five]; y[five]:=0; DIFF(k1);
if kb on then WRI; testg:= testn; step:=A01/k1[four]; WTS
end ATMOF;
```

```
procedure WTS;
begin if kb on then begin writecr; write(↑dddd↑, tsg, tsd);
end end WTS;
```

```
procedure CKUGL(Tc, rhoc); value Tc, rhoc; real Tc, rhoc;
begin
X:= Xcc; gQ:= 0;
XomH:= X/1.00801; YomHe:= (1-X-Z)/4.0028; ZomA:= Z/16.942; eomue:= XomH + A2 x YomHe
q:= mk[0]; y[1]:= A1; y[two]:= Atim4;
ztype:= three; y[three]:= Tc;
T:= exp(Tc); rho:= exp(rhoc);
arb:= F:= rho/(9.047610-9xexp(A1p5xTc))xeomue;
F:= (F-30)/30; F:= (((0.109744xF-0.165376)xF+0.151420)xF-0.462698)xF+3.296236)xF+5.1
if kb on then begin write(↑-ddd.dddd↑, arb, F); writecr end;
y[four]:= ln(k/m0xrhoXTx(XomH+YomHe+ZomA+eomuexF)+a/A3xexp(TcxA4)
-sqrt(cnsqm(X+7xZ+A3)↑3xrho/T)xrho);
DIFF(k1); y[five]:= k1[five]; y[6]:= k1[6];
if kb on v mode=four then WRI;
step:= q:= mk[0];
y[1]:= ln(A3xMxMsxq/(A4xpixrho))/A3;
ddr:= U3xMsxkappaxepsxP/T4;
arb:= -GxMxMsxq/A2xrho/(exp(y[1])xP); y[two]:= epsxMxMsxq/L0;
y[three]:= Tc+arbx(if ddr<dda then ddr else dda); y[four]:= y[four]+arb; DIFF(k1);
y[five]:= (y[five]+k1[five])/A2xq; y[6]:= (y[6]+k1[6])/A2xq;
qcnew:= QDX[0]:= if testn>0 then q else 0; dXmax:= DX[0]:= DXcc:= y[6]/q; nDX:= 0;
testg:= testn; if kb on v mode=four then WRI;
end CKUGL;
```

comment In the following block we first input the PARAMETER LIST, and then tables are read to the arrays fplkap and ddaalHJU;

```

begin integer k;
real t, p, t1;
integer array fp, lkap[0: 30];

procedure TWOLINES(B); array B;
comment TWOLINES reads two lines of numbers from fp - tables and stores
some of these table - values in packed form in integer array B;
begin procedure fpline(A); integer array A;
begin comment fpline reads one line of numbers from fp - tables.
These numbers are multiplied by 1000, and in case the absolute
value of a number then exceeds 9999 it is replaced by ± 9999;
for j:= 0 step 1 until (if k<ilp then k else ilp) do
begin A[j]:= read real×1000; if abs(A[j])>9999 then
A[j]:= sign(A[j])×9999 end;
for j:= ilp+1 step 1 until k do dummy:= read real
end fpline;
fpline(fp); fpline(lkap);
for j:= 0 step 1 until (if k<ilp then k else ilp) do
B[i,j]:= sign(lkap[j])×(fp[j]×10000+abs(lkap[j]));
end TWOLINES, two lines have been read and packed in B;

UNRW:= 16; UNWW:= 17; UNWD:=9; UNWP:= 33; A02:= 0.2;
two:=2; three:=3; four:=4; five:= 5; Atim4:= 10-4;

dummy:= select(UNWW); READ LIST; writetext(⟨<
t1, ilt:= ⟩); t1:= read real; arb:= read real;
dummy:= select(UNRW); ltb:=read real; ilt:=read integer; lpb:=read real; ilp:=read
X:= read real; Y:= read real; Z:= read real;
slT:= read real; dummy:= read real; slP:= read real; dummy:= read real;
skip: t:= read real; p:= read real;
if t < t1-Atim4 then begin
ilp:= read integer; for j:=1 step 1 until 4×(ilp+1) do dummy:= read real;
goto skip end;
ltb:= t1; lpb:= p; ilt:=arb; ilp:=20;
for i:= 0 step 1 until ilt do
begin if i≠0 then begin t:= read real; p:= read real;end;
k:= read integer;
if abs(t-ltb-slT×i)+abs(p-lpb-slP×i)>Atim4 then begin
dummy:= select(UNWW); writetext(⟨<
Error in fp-table⟩); dummy:= lyn; go to stop end;
TWOLINES(fplkap); TWOLINES(ddaalHJU);
end reading of fp tables;
dummy:= select(UNWW); dummy:= lyn; dummy:= select(UNRW); tb:= read real; rb:= read
ka:= read integer; t:= read integer; nm:= read integer;
for i:= 1 step 1 until nm do begin XM[i]:= read real;
for j:= 0 step 1 until t do begin
if abs(read real-tb-j×A02)>Atim4 then begin writetext(⟨<
Error in log kappa table⟩); go to stop end;
dummy:= read real; dummy:= read real; for k:= 0 step 1 until ka-1 do
km[i,j,k]:= read real end end læs kappa;
end tablereading;

comment In the next part of the program values are assigned to many variables;
A1:= mf:= 1;
A2:= 2; A3:= 3; A4:= 4; A5:= 5; A6:= 6;
A01:= 0.1; A08:= 0.8; A09:= 0.9; A1p5:= 1.5;
pi:= 3.14159265; mod:= 0.434294482; recmo:= 2.30258509;
G:= 6.66810-8; m0:= 1.6602610-24; k:= 1.3804610-16; a:= 7.564110-15; c:= 2.9979291010;
Ls:= 3.901033; Rs:= 6.95981010; Ms:= 1.9891033; lTes:= log(Ls/(a×pi×c×Rs2))/A4;
U1:= Ms/(A4×pi); U3:= A3/(16×pi×a×c×G×Ms); cnsqm:= pi/k/8×(4.80293/m0/31030)2/m0; ex
C3:= A6×sqrt(2)×a×c×m0/k; go to ldef;

rep: if autm<three then begin writecr; writecr; write(⟨-dddd⟩, num, sernum, runnum);

```

```

        write({-ddd.ddd}, M, Xout, Z); writecr; lR:= A2*(lTes-lTe) + lL/A2;
        write({-dd.dddd}, lTe, lL, yik[0,three]*mod, yik[0,four]*mod); end;
if impcount>five then ALARM({<IMPROVE}, 0);
if autm = 0 v kb on then go to SW[lyn];
Gsum:= 0; tsg:= tsd:= 0;
for lfp:= 0 step 1 until nfp-1 do begin

intil:= if lfp<fit then lfp+1 else nfp-1-lfp+fit;
if lfp = 0 then CKUGL(yik[0,three], yik[0,four]);
if lfp = fit then begin if mode=four ^ qcnew=0 then begin DXcc:= y[6]; nDX:= -1; qcnew
                    nDXfit:= nDX; ATMOF(lTe,lL) end;

DIFF(k1); testg:= testn; MERSN;
if kb on then begin WRI; OUT4(yik, intil, 1) end;
for i:= 1 step 1 until four do begin arb:= yik[intil,i]; Gik[1+lfp,i]:= arb1:= arb-y[
y[i]:= arb; Gsum:= Gsum + abs(arb1) end;
if autm<two then OUT4(Gik, 1+lfp,four);
end integration statement; Gsumtot:= Gsumtot + Gsum;

writecr; write({-dddddd}, Gsum/Atim4, EGsum/Gsum, tsg, tsd);
if mode = four then go to timestep1 ;

if autm = 0 then begin i:= lyn;
                    if i = 7 then go to matrix else
                    if i = 8 then mf:= EGsum/(EGsum + Gsum)*mf else
                    if i = 9 then mf:= read real else go to SW[i] end else

begin boolean bad; comment automatic mode; bad:= EGsum/Gsum<A4;
if Gsum<0.0010 then go to timestep;
if Gsum<0.0100 ^ mtX4<two then begin mode:= three; go to improve end;
if mode = 8 then begin ALARM({< TIME GSUM}, 0); go to timestep end;

if mode>four then mode:= mode + 1;
if mode = 0 then begin if Gsum + EGsum < A01 then mode:= five end;

mf:= if Gsum>A3 then A02 else
      if Gsum>A1 v mode>five then 0.6 else
      if bad then A08 else A1;

if mtX4<0 then mtX4:= mtX4 + 1 else
begin if bad then mtX4:= mtX4 + 1;
      if mtX4>1 then go to matrix;
      mtX4:= if bad then 1 else 0;
end;
end flow control;

improve: EGsum:= Gsum;
begin comment In this block parameter corrections are determined and the
              parameter values are corrected;

integer i, k;
array A, D, E, F, D1, E1, F1 [1:4], COEFM [1: 4, 1: 5], dyik[0:nfp-1, 1:4];
boolean elimmode;

procedure LLGAUSS(n,R,A,U,delta); value n,delta; integer n; real delta;
array R,A; label U;
begin integer i,j,k,h; real a,b;
k:= 0; Do it again: k:= k+1; a:= A[k,k]; h:= k; for i:= k step 1 until n-1 do
begin b:= A[i+1,k]; if abs(a)>abs(b) then go to S; h:= i+1; a:= b; S: end;
if abs(a)<delta then go to U; if h=k then go to eliminate;
for j:= k step 1 until n+1 do begin b:= A[k,j];A[k,j]:= A[h,j]; A[h,j]:= b end;
eliminate: for i:= k+1 step 1 until n do
begin b:= -A[i,k]/a; for j:= k+1 step 1 until n+1 do A[i,j]:= A[i,j]+A[k,j]*b end;

```

```

if k<n then go to Do it again; next unknown: b:= 0; for j:= n step -1 until k+1 do
b:=R[j]×A[k,j]+b; R[k]:= (A[k,n+1]-b)/A[k,k]; k:= k-1;
if k>0 then go to next unknown
end LLGAUSS;

```

```

procedure ELIMSOL (beg, stop); value beg, stop; integer beg, stop;
begin comment In both elim - and solutionmode the quantities D, E, F
are computed. In solutionmode (-, elimmode) these
quantities are used to calculate parameter corrections;
real procedure DEFSUM (B); array B;
begin comment DEFSUM computes a sum in the formulae (2.27) or (2.29);
integer h; real s;
s:= 0; for h:= 1 step 1 until four do s:= DEFSUM:= s + dGdy [k, i, h]×B[h]
end of DEFSUM;

```

```

for k:= beg step 1 until stop do begin
for i:= 1 step 1 until four do begin
comment Here the values of D, E, and F are computed
according to formulae (2.26) - (2.29);
D 1[i]:= - Gik [k, i] - (if k = beg then 0 else DEFSUM (D));
E 1[i]:= if k = beg then - dGdy [beg, i,three] else - DEFSUM(E);
F 1[i]:= if k = beg then - dGdy [beg, i,four] else - DEFSUM(F);
if -, elimmode then begin
comment Solutionmode: dyik are computed from (2.26) - (2.29);
if k ≤ fit then dyik [k, i]:= D1 [i] + E1 [i]× A[three]+F1 [i]×A [four]
else dyik [nfp - 1 - k + fit +1, i]:= D1[i]+E1 [i]×A [1] +F1 [i]×A [two]
end solutionmode;
end i loop;

```

```

for i:= 1 step 1 until four do begin
D[i]:= D1 [i]; E [i]:= E1 [i]; F [i]:= F1 [i] end;

```

```

end k loop;
end procedure ELIMSOL;

```

comment First the coefficients in the 4 equations (2.30) for the corrections at the center and the surface are computed;

```

elimmode:= true; ELIMSOL (1, fit);
for i:= 1 step 1 until four do begin
COEFM[i,three]:= - E[i]; COEFM [i,four]:= - F[i];
COEFM[i,five]:= D[i] end;
ELIMSOL (1+fit, nfp);
for i:= 1 step 1 until four do begin
COEFM[i, 1]:= E[i]; COEFM[i,two]:= F[i];
COEFM[i,five]:= COEFM[i,five] - D[i] end;
LLGAUSS (four, A, COEFM, EXIT, 10 - 6);

```

comment We can now directly use the formulae (2.26) - (2.29) for the determination of all other parameter corrections;

```

elimmode:= false;
for i:= 1 step 1 until four do dyik [0, i]:= A[i];
ELIMSOL (1, fit); ELIMSOL (1+fit, nfp - 1);

```

```

go to END;
EXIT:begin ALARM(⟨LLGAUSS⟩, 0); go to endser end;

```

END:

```

for i:= 0 step 1 until nfp - 1 do begin
if kbon then begin OUT4(yik,i,1); OUT4(dyik,i,1) end;
for k:= 1 step 1 until four do yik [i, k]:= yik [i, k] + dyik [i, k]×mf end improve;
lTe := yik [0,1]; lL:= yik [0,two];

```

end improve block;

if mode = three then go to timestep;
impcount:= impcount + 1;
if autm = 0 then go to SW[lyn]; go to rep;

comment In the following block the coefficients in the equations for the corrections to the parameters are computed;

matrix: writetext({<
matrix}); mt4:= - four;
begin real GF, gstep; integer from , i;
procedure FORST(s,t); value s,t; integer s,t;
for i:= 1 step 1 until four do dGdy[lfp+1,i,s]:= dGdy[lfp+1,i,s+two]:=
(yik[intil,i]-y[i]-Gik[lfp+1,i])/alter[t];

for lfp:= 0 step 1 until nfp-1 do begin
intil:= if lfp<fit then lfp + 1 else nfp-1-lfp + fit;
from:= if lfp<fit then lfp else intil + 1;
if lfp = 0 then begin CKUGL(yik[0,three]+alter[three],yik[0,four]); MERSN; FORST(1,th
CKUGL(yik[0,three],yik[0,four]+alter[four]); MERSN; FORST(two,four); gstep:=

end else

if lfp = fit then begin ATMOF(lTe+alter[1],lL); MERSN; FORST(1,1);
ATMOF(lTe,lL+alter[two]); MERSN; FORST(two,two); gstep:= step; GF:= F;

end else

begin comment normal case follows;

for i:= 1 step 1 until four do begin for j:= 1 step 1 until four do y[j]:= yik[from,j]
q:= mk[from]; y[i]:= y[i] + alter[i]; F:= GF; step:= gstep; DIFF(k1);

testg:= testn; MERSN;

for j:= 1 step 1 until four do dGdy[lfp+1,j,i]:= (yik[intil,j]-y[j]
-Gik[lfp+1, j])/alter[i];

end testalterations;

end one zone;

GF:= F; gstep:= step;

end lfp forstatement;

end matrixblock;

if autm = 0 then go to SW[lyn]; mf:= A1; go to improve;

ldef: dummy:= select (UNWW); writetext({<
Input definition}); dummy:= lyn; dummy:= select (UNRW); nfp:= read real; fit:= read r
begin real Rt, Lt, y1, y2;

num:= read real; age:= read real; dt:= read real; dtold:= read real;

M:= read real; qcc:= read real; Xcc:= read real; qout:= read real; Xout:= read real;

Z:= read real; LV:= read real; delta:= read real; dpar:= read real;

for i:= 1 step 1 until four do begin rpfac[i]:= read real; alter[i]:= read real end;

nQ:= read real; tgr:= read real; energi:= read real;

for i:= 0 step 1 until nQ do begin Q[i]:= read real; dummy:= read real;

XQ[i]:= read real/mod end;

for i:= 0 step 1 until tgr do begin arb:= read real; arb:= read real end;

for i:= 0 step 1 until nfp do mk[i]:= read real;

for t:= 0, nfp, twoxnfp do

for i:= 0 step 1 until nfp - 1 do begin

arb:= read real; if arb ≠ mk [i] then beginALARM({<DEF. ERROR}, 0);

go to endser end;

y1:= yik [i+t, 1]:= read real; y2:= yik [i+t,two]:= read real;

yik [i+t, three]:= read real/mod; yik [i+t, four]:= read real/mod;

```

if i = 0 then begin
Rt:= antlog (y2/A2 + A2 × (lTes - y1)) × Rs;
Lt:= antlog (y2) × Ls; if t = 0 then L0:= Lt/LV;
      end else begin comment i ≠ 0 normal case follows;
yik [i+t, 1]:= ln (y1 × Rt);
yik [i+t, two]:= y2 × Lt/L0;
      end;

```

```

end i loop and input of parameters;
impcount:= almcount:= mode:= 0; cdout:= defout-1;
EGsum:= A4 ; mt4:= two; lTe:= yik[0,1]; lL:= yik[0,two];
end inputblock; dummy:= select (UNWW);
go to rep;

```

```

tdef: if autm≠0 then begin cdout:=cdout+1;if cdout≠defout then go to FINI elsecdout:
begin integer s; real Rt, Lt, y1, y2;
procedure line print (n,a,b,c,d,e); value n,a,b,c,d,e;
integer n; real a,b,c,d,e;begin switch SW1:= 11,12,13,14,15;
go to SW1[n];
11: write(⟨-dd.ddddd⟩,a,writetext (⟨<,⟩));
goto END;
12: write(⟨-dd.ddddd⟩,a,writetext (⟨<,⟩),b,writetext (⟨<,⟩));
goto END;
13: write(⟨-dd.ddddd⟩,a,writetext (⟨<,⟩),b,writetext (⟨<,⟩),c,writetext (⟨<,⟩));
goto END;
14: write(⟨-dd.ddddd⟩,a,writetext (⟨<,⟩),b,writetext (⟨<,⟩),c,writetext (⟨<,⟩),
d,writetext (⟨<,⟩));
goto END;
15: write(⟨-dd.ddddd⟩,a,writetext (⟨<,⟩),b,writetext (⟨<,⟩),c,writetext (⟨<,⟩),
d,writetext (⟨<,⟩),e,writetext (⟨<,⟩));
END: end procedure lineprint;

```

```

line print (two,nfp,fit,dummy,dummy,dummy);
line print (four,num,age,dt,dtold,dummy);
line print (three,M,qcc,Xcc,dummy,dummy);
line print (three,qout,Xout,Z,dummy,dummy);
line print (three,LV,delta,dpar,dummy,dummy);
for i:=1 step 1 until four do
line print (two,rpfac[i],alter[i],dummy,dummy,dummy);
line print (three,nQ,tgr,energi,dummy,dummy);
writecr;
for i:= 0 step 1 until nQ do
line print (three,Q[i],exp(XQ[i]),modXXQ[i],dummy,dummy);
writecr;
for i:= 0 step 1 until tgr do
line print (two,arb,arb,dummy,dummy,dummy);
for i:= 0 step 1 until nfp do
line print (1,mk[i],dummy,dummy,dummy,dummy);

```

```

for t:= 0, nfp, two×nfp do begin writetext (⟨<
⟩);
for i:= 0 step 1 until nfp - 1 do begin
y1:= yik [i+t, 1]; y2:= yik [i+t, two];
if i = 0 then begin Rt:= antlog (y2/A2 + A2 ×(lTes - y1)) × Rs;
      Lt:= antlog (y2) × Ls end;
line print (five, mk [i], if i = 0 then y1 else exp (y1)/ Rt,
      if i = 0 then y2 else y2 × L0/Lt,
      yik [i+t, three] × mod, yik [i+t, four] × mod)
end i loop;
end t loop and yik output;

```

```
writetext(⟨<
```

```
⟩);
```

```
end af tdef;  
dummy:=select(s);  
FINI:  
go to rep;
```

```
choose: s:= select(UNWW); writetext(⟨<  
choose by letter: ⟩); i:= lyn;  
if i = 53 then mtX4:= read integer;  
if i = 54 then begin mk[0]:= read real; mk[nfp]:= read real end;  
if i = 40 then begin writetext (⟨< qout, Xout, Xcc:= ⟩);  
qout:= read real; Xout:= read real; Xcc:= read real; end;  
if i = 41 then READ LIST;  
if i = 39 then dpar:= read real;  
if i = 18 then go to stop;  
if i = 57 v i = 38 then begin integer ii; dummy:= select(12);  
for ii:= 1 step 1 until nfp do begin if i = 38 then writecr;  
for j := 1 step 1 until four do begin if i = 38 then writecr;  
for t:= 1 step 1 until four do begin  
if i = 57 then dGdy[ii, j, t]:= read real else  
begin write(⟨-ddd.ddddd⟩, dGdy[ii,j,t]);  
writetext(⟨<,⟩); end;  
end t loop;  
end j loop; end ii loop;  
end dGdy in - out block;  
if i = 36 then mf:= read real;  
if i = 49 then autm:= read integer;  
if i = 52 then delta:= read real;  
dummy:= select(s);  
go to SW[lyn]; go to rep;
```

comment In the following the evolution of the model is computed.

We first perform a final integration and output a number of model characteris

```
timestep:  
mode:= four; clout:= lineout - 1; s:=select(49); go to tdef;  
timestep1:  
begin real LUMIN, eps to dX, tstep1, tstep2, tstep3;
```

```
LUMIN:= antlog(1L)×Ls; epstodX:= 3.1558151013/6.39851881018; writetext(⟨<  
Te log g log R/Rs Mbol Lneu/L:  
⟩); write(⟨-dddd.dddd⟩, antlog(1Te), log(gacc), 1R,  
4.72 - 2.5×1L, (Lneu - y[five])×L0/LUMIN);  
dummy:= select(s);
```

```
if num = nummax then go to endser; num:= num + 1; impcount:= 0;
```

comment We now prepare the determination of time - step. The unit of dX is changed and dX - values are reorganised. If age = 0 an X array is created, and if kb on the dX array and the (old) H - profile are output;

```
if qcnew>mk[fit] then dXmax:= DXcc:= (DXcc+y[6])/qcnew;  
dXmax:= dXmax×epstodX; DXcc:= DXcc×epstodX;  
for i:= 0 step 1 until nDX do DX[i]:= DX[i] × epstodX;  
if qcnew>mk[fit] then begin for i:= nDX,i-1 while QDX[i]<qcnew do nDX:= i-1; end;
```

```

if qout>mk[fit] then nDX:= nDX-1;
for i:= 1 step 1 until (nDX-nDXfit)/A2 + A01 do begin
arb:= QDX[nDXfit+i]; arb1:= DX[nDXfit+i]; QDX[nDXfit+i]:= QDX[nDX+1-i];
QDX[nDX+1-i]:= arb; DX[nDXfit+i]:= DX[nDX+1-i]; DX[nDX+1-i]:= arb1; end;
if age=0 then beginqcc:=Q[0]:=Q[1]:=qcnew;
if qcnew=mk[0] then begin Q[1]:=0.02;Q[two]:=0.05 end
else Q[two]:=0.05xentier((qcnew+0.06)/0.05);
nQ:=two;XQ[0]:=XQ[1]:=XQ[two]:=ln(Xout);
ADDnQ: nQ:=nQ+1;
Q[nQ]:=Q[nQ-1]+0.05;XQ[nQ]:=ln(Xout);
if Q[nQ]<qout+Atim4 then go to ADDnQ
end criation of X array;
dummy:= select(UNWW);

if autm=0 then dummy:= lyn; if kb on then begin writecr; writecr;
writetext(⟨qcnew , DXcc×100 : ⟩); write(⟨ddd.dddd⟩, qcnew, 100×DXcc);
writecr; writecr; writetext(⟨ QDX DX×100 Q XQ ⟩);
for i:= 0 step 1 until nDX do begin writecr; write(⟨ddd.dddd⟩, QDX[i], 100×DX[i]);
if i<nQ then begin writetext(⟨ ⟩); write(⟨ddd.dddd⟩, Q[i], exp(XQ[i])); end; end
end kb on;

comment In the following the time - steps given by different criteria
are computed, and the smallest one is choosen;

tstep1:= A01×( if Xcc<0.16 then Xcc1/3 else 0.3 ) /dXmax;
tstep2:= (A4×dt - A2×dtold + 0.04×dt/
(yik[0,four] - yik[nfp,four]))/A3;

if Xcc > A01 then dage:=tstep1 else
if Xcc > 10-3 then dage:= (if tstep1<tstep2 then tstep1 else tstep2)
else dage:= tstep2;dage:=dage×dpar;

writetext(⟨<
tstep1, tstep2, dage (106 years) = ⟩); write(⟨ddd.ddd⟩, tstep1, tstep2, dage);

if autm=0 then beginwritecr; i:= lyn; if i≠0 then dage:= read real; end;
age:= age + dage; writetext(⟨<
NEW MODEL WITH AGE: ⟩); write(⟨ddd.ddd⟩,age); writetext(⟨< mill. years⟩);writecr;

comment Now the new X - profile is computed. If kb on the results
of all computation of new X - values are output (in procedure
XNEW). No convective core is indicatet by qcnew = mk[0];

begin integer b;
real Xj, Xjm1, NX, NEWXj, NEWXjm1;
integer array NOnew[1:10];
array Qnew,XQnew[1: 10];
procedure XNEW(q,OLDX, NEWX); value q,OLDX; real q,OLDX,NEWX;
begin integer i; real p;
if q>QDX[gQ] then begin for i:= gQ , i+1 while q>QDX[i] do gQ:=i; end else
for i:= gQ , i-1 while q<QDX[i] do gQ:= i-1;
p:= abs(q-QDX[gQ])/(QDX[gQ+1]-QDX[gQ]);
NEWX:= OLDX - dage×( p×DX[gQ+1] + (1-p)×DX[gQ] );
if kb on then begin writecr; write(⟨ddd⟩, gQ); write(⟨ddd.dddd⟩,
q, OLDX, NEWX); end kb on;
end XNEW;

Q[0]:= qcnew; NEWXj:= Xcc - dage×DXcc; XQ[0]:= ln(NEWXj); gQ:= t:= 0;
if qcnew = mk[0] then begin
b:= 1; Xj:= Xcc; Xcc:= NEWXj;
go to dX for points end;

```

comment Now we consider the case: convective core present;

```
b:= 2; Xj:= exp(XQ[1]); XNEW(Q[1], Xj, NX);XQ[1]:=ln(NX);  
if qcc>mk[0] ^ num ≠ 1 then  
begin comment Normal case.For the first model this section is skipped;
```

```
arb:= Xcc - dage×DX[0]; XNEW(qcc,Xcc,arb1);t:=NOnew[1]:=1;  
Qnew[t]:=qcc-(qcc-qcnew)×(arb-NEWXj) / ((qcc-qcnew)×(NX -NEWXj)  
- (Q[1]-qcnew)×(arb1-arb));  
XQnew[t]:= ln( arb1 - (arb1-arb)×(qcc-Qnew[t])/(qcc-qcnew));end normal case;
```

```
Xcc:= NEWXj; qcc:= qcnew; NEWXj:= NX;
```

```
dX for points: for j:= b step 1 until nQ - 1 do beginXjml:= Xj; Xj:= exp(XQ[j]); NEWXj:=  
XQ[j]:= ln(NEWXj);  
if NEWXjml<NEWXj - 0.02 then begin  
t:= t+1; NOnew[t]:= j; Qnew[t]:= (Q[j-1] + Q[j])/A2;  
XNEW(Qnew[t], (Xjml + Xj)/A2, NX); XQnew[t]:= ln(NX);  
end end DX points j and j-1;
```

```
j:= t;  
for i:= nQ step -1 until 0 do begin if j>0 then begin if NOnew[j]>i+A01 then  
begin Q[i+j]:= Qnew[j]; XQ[i+j]:= XQnew[j]; j:= j-1; end; end;  
Q[i+j]:= Q[i]; XQ[i+j]:= XQ[i]; end;  
nQ:= nQ + t;  
end for X - profile block;
```

comment Extrapolation to new parameter - values followed by initiation of several variables and a jump to the fitting routine;

```
for i:= 0 step 1 until nfp - 1 do for j:= 1 step 1 until four do begin  
arb:= 2×yik[i, j] - yik[i+nfp, j];  
yik[i+nfp, j]:= yik[i, j]×(if i>0^j=2 then L0/LUMIN else 1 );  
yik[i, j]:= arb×(if i>0^j=2 then L0/LUMIN else 1 ) end;  
mode:= 0; EGsum:= A4; L0:= LUMIN; Gsumtot:= 0;  
lTe:= yik[0, 1]; lL:= yik[0,two]; dtold:= dt; dt:= dage;  
end time - step block;
```

```
dummy:= select(UNWW); go to rep;
```

```
endser: if sernum<sermax then  
begin sernum:= sernum + 1; almcount:= 0; go to ldef end;  
stop: end
```