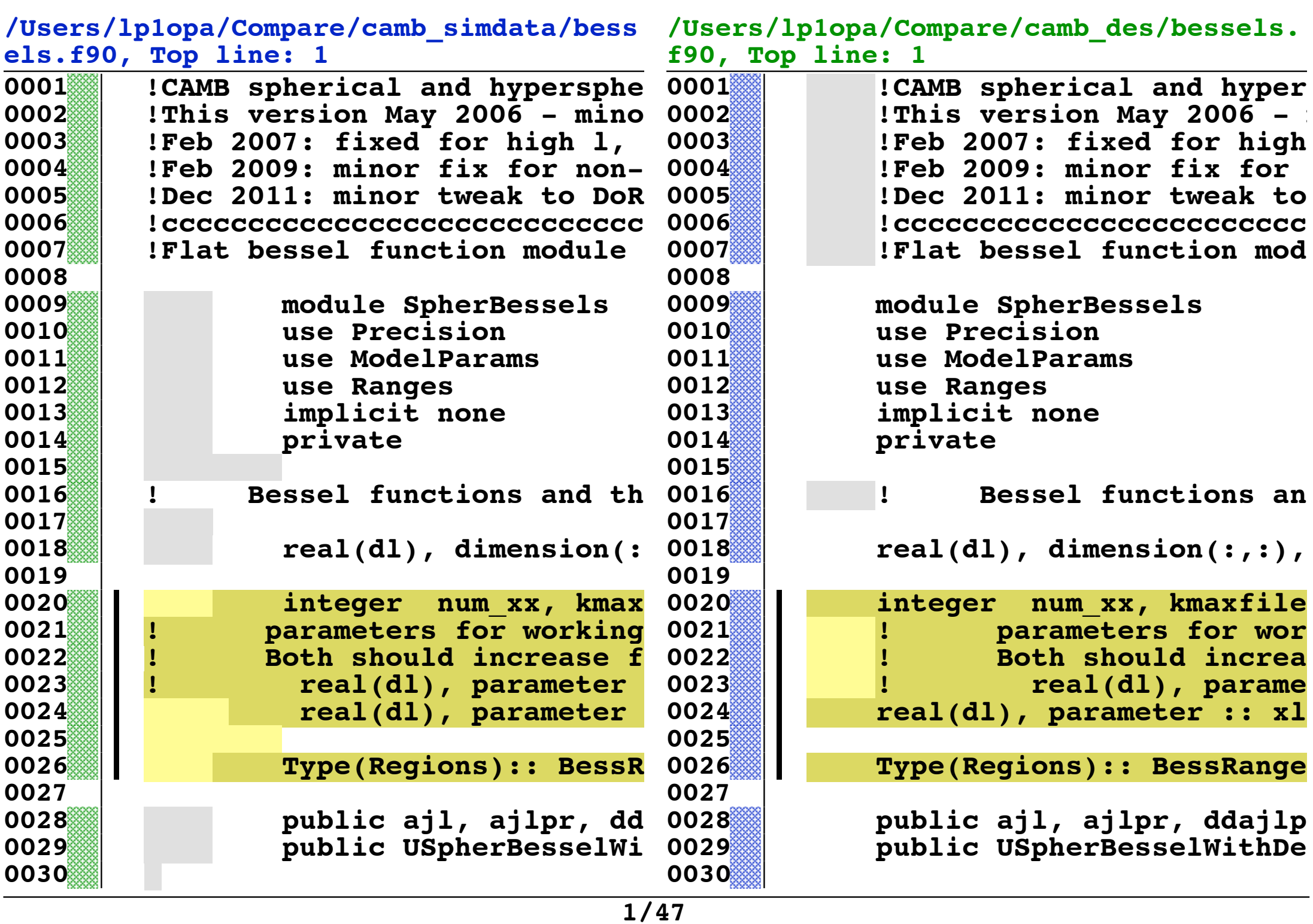
/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 1
/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1
!CAMB spherical and hypersphe
!This version May 2006 - mino
!Feb 2007: fixed for high l,
!Feb 2009: minor fix for non-
!Dec 2011: minor tweak to DoR
!Flat bessel function module
module SpherBessels
use Precision
use ModelParams
use Ranges
implicit none
private
! Bessel functions and th
real(dl), dimension(:
integer num_xx, kmax
! parameters for working
! Both should increase f
! real(dl), parameter
real(dl), parameter
Type(Regions):: BessR
public ajl, ajlpr, dd
public USpherBesselWi
!CAMB spherical and hyper
!This version May 2006 -
!Feb 2007: fixed for high
!Feb 2009: minor fix for
!Dec 2011: minor tweak to
!Flat bessel function mod
module SpherBessels
use Precision
use ModelParams
use Ranges
implicit none
private
! Bessel functions an
real(dl), dimension(:,:),
integer num_xx, kmaxfile
! parameters for wor
! Both should increa
! real(dl), parame
real(dl), parameter :: xl
Type(Regions):: BessRange
public ajl, ajlpr, ddajlp
public USpherBesselWithDe

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 31

```
0031     contains
0032
0033
0034     subroutine InitSpherBes
0035  !  This subroutine reads t
0036     use lvalues
0037     implicit none
0038
0039     !See if already loaded
0040     if (allocated(ajl) .and
0041                    .and. (in
0042
0043     !Haven't made them befo
0044     call GenerateBessels
0045
0046     if (DebugMsgs .and. Fee
0047
0048     end subroutine InitSphe
0049
0050   subroutine GenerateBesse
0051     use lvalues
0052     real(dl) x
0053     real(dl) xlim
0054     integer i,j
0055     integer max_ix
0056     real(dl), parameter ::
0057
0058
0059      if (DebugMsgs .and. F
0060
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 31

```
0031     contains
0032
0033
0034     subroutine InitSpherBesse
0035  !     This subroutine rea
0036     use lvalues
0037     implicit none
0038
0039     !See if already loaded wi
0040     if (allocated(ajl) .and.
0041         .and. (int(min(max_be
0042
0043     !Haven't made them before
0044     call GenerateBessels
0045
0046     if (DebugMsgs .and. Feedb
0047
0048     end subroutine InitSpherB
0049
0050     subroutine GenerateBessel
0051     use lvalues
0052     real(dl) x
0053     real(dl) xlim
0054     integer i,j
0055     integer max_ix
0056     real(dl), parameter :: be
0057
0058
0059     if (DebugMsgs .and. Feedb
0060
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 61

```
0061
0062        file_num1= lSamp%l0
0063        file_l(1:lSamp%l0) =
0064        kmaxfile = int(min(CP
0065        if (do_bispectrum) km
0066
0067
0068        call Ranges_Init(Bess
0069
0070        call Ranges_Add_delta
0071        call Ranges_Add_delta
0072        call Ranges_Add_delta
0073        call Ranges_Add_delta
0074        call Ranges_Add_delta
0075
0076        call Ranges_GetArray(
0077        num_xx = BessRanges%n
0078
0079
0080      max_ix = min(max_besse
0081
0082      if (allocated(ajl)) de
0083      if (allocated(ajlpr))
0084      if (allocated(ddajlpr)
0085      Allocate(ajl(1:num_xx,
0086      Allocate(ajlpr(1:num_x
0087      Allocate(ddajlpr(1:num
0088
0089      !$OMP PARALLEL DO DEFA
0090      do j=1,max_ix
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 61

```
0061
0062        file_num1= lSamp%l0
0063        file_l(1:lSamp%l0) = lSam
0064        kmaxfile = int(min(CP%Max
0065        if (do_bispectrum) kmaxfi
0066
0067
0068        call Ranges_Init(BessRang
0069
0070        call Ranges_Add_delta(Bes
0071        call Ranges_Add_delta(Bes
0072        call Ranges_Add_delta(Bes
0073        call Ranges_Add_delta(Bes
0074        call Ranges_Add_delta(Bes
0075
0076        call Ranges_GetArray(bess
0077        num_xx = BessRanges%npoin
0078
0079
0080        max_ix = min(max_bessels_
0081
0082        if (allocated(ajl)) deall
0083        if (allocated(ajlpr)) dea
0084        if (allocated(ddajlpr)) d
0085        Allocate(ajl(1:num_xx,1:m
0086        Allocate(ajlpr(1:num_xx,1
0087        Allocate(ddajlpr(1:num_xx
0088
0089        !$OMP PARALLEL DO DEFAULT
0090        do j=1,max_ix
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 91

```
0091
0092            do  i=1,num_xx
0093               x=BessRanges%poin
0094               xlim=xlimfrac*lSa
0095               xlim=max(xlim,xli
0096               xlim=lSamp%l(j)-x
0097               if (x > xlim) the
0098                  if ((lSamp%l(j
0099                                 (
0100                     ajl(i,j)=0
0101                  else
0102                   !if ( lSamp
0103                   ! ajl(i,j)
0104                   !else
0105                    call bjl(l
0106                   !end if
0107               end if
0108            else
0109                  ajl(i,j)=0
0110            end if
0111         end do
0112
0113   !     get the interpolation m
0114           call spline(BessRange
0115           call spline(BessRange
0116
0117      end do
0118   !$OMP END PARALLEL DO
0119
0120   end subroutine GenerateB
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 91

```
0091
0092            do  i=1,num_xx
0093               x=BessRanges%poin
0094               xlim=xlimfrac*lSa
0095               xlim=max(xlim,xli
0096               xlim=lSamp%l(j)-x
0097               if (x > xlim) the
0098                  if ((lSamp%l(
0099                     (lSamp%l(
0100                     ajl(i,j)=
0101                  else
0102                     !if ( lSa
0103                     ! ajl(i,j
0104                     !else
0105                     call bjl(
0106                     !end if
0107               end if
0108            else
0109                  ajl(i,j)=0
0110            end if
0111         end do
0112
0113         !     get the interpo
0114           call spline(BessRange
0115           call spline(BessRange
0116
0117      end do
0118   !$OMP END PARALLEL DO
0119
0120   end subroutine GenerateBe
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 121

```
0121
0122      subroutine Bessels_Free
0123
0124        if (allocated(ajl)) de
0125        if (allocated(ajlpr))
0126        if (allocated(ddajlpr)
0127        call Ranges_Free(BessR
0128
0129      end  subroutine Bessels_
0130
0131
0132         SUBROUTINE BJL(L,X,J
0133       !!== MODIFIED SUBROUT
0134       !!== CORRECTED THE SM
0135       !!== CORRECTED THE SI
0136       !!== WORKS FASTER AND
0137       !!== zqhuang@astro.ut
0138       IMPLICIT NONE
0139       INTEGER L
0140       real(dl) X,JL
0141       real(dl) AX,AX2
0142       real(dl),PARAMETER::L
0143       real(dl),PARAMETER::O
0144       real(dl),PARAMETER::P
0145       real(dl),PARAMETER::P
0146       real(dl),parameter::R
0147       real(dl),parameter::G
0148       real(dl),parameter::G
0149       real(dl),PARAMETER::P
0150       real(dl) NU,NU2,BETA,
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 121

```
0121
0122      subroutine Bessels_Free
0123
0124      if (allocated(ajl)) deall
0125      if (allocated(ajlpr)) dea
0126      if (allocated(ddajlpr)) d
0127      call Ranges_Free(BessRang
0128
0129      end  subroutine Bessels_F
0130
0131
0132      SUBROUTINE BJL(L,X,JL)
0133      !!== MODIFIED SUBROUTINE
0134      !!== CORRECTED THE SMALL
0135      !!== CORRECTED THE SIGN O
0136      !!== WORKS FASTER AND MOR
0137      !!== zqhuang@astro.utoron
0138      IMPLICIT NONE
0139      INTEGER L
0140      real(dl) X,JL
0141      real(dl) AX,AX2
0142      real(dl),PARAMETER::LN2=0
0143      real(dl),PARAMETER::ONEML
0144      real(dl),PARAMETER::PID2=
0145      real(dl),PARAMETER::PID4=
0146      real(dl),parameter::ROOTP
0147      real(dl),parameter::GAMMA
0148      real(dl),parameter::GAMMA
0149      real(dl),PARAMETER::PI=3.
0150      real(dl) NU,NU2,BETA,BETA
```

/Users/lp1opa/Compare/camb_simdata/besselS.f90, Top line: 151

```
0151        real(dl) sx,sx2
0152        real(dl) cotb,cot3b,c
0153        real(dl) trigarg,expt
0154
0155        IF(L.LT.0)THEN
0156            write(*,*) 'Can n
0157            STOP
0158        ENDIF
0159        AX=DABS(X)
0160        AX2=AX**2
0161        IF(L.LT.7)THEN
0162            IF(L.EQ.0)THEN
0163                IF(AX.LT.1.D-
0164                    JL=1.D0-A
0165                ELSE
0166                    JL=DSIN(A
0167                ENDIF
0168
0169            ELSEIF(L.EQ.1)THE
0170                IF(AX.LT.2.D-
0171                    JL=AX/3.D
0172                ELSE
0173                    JL=(DSIN(
0174                ENDIF
0175            ELSEIF(L.EQ.2)THE
0176                IF(AX.LT.3.D-
0177                    JL=AX2/15
0178                ELSE
0179                    JL=(-3.0D
0180                ENDIF
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 151

```
0151        real(dl) sx,sx2
0152        real(dl) cotb,cot3b,cot6b
0153        real(dl) trigarg,expterm,
0154
0155        IF(L.LT.0)THEN
0156            error stop 'Can not e
0157        ENDIF
0158        AX=DABS(X)
0159        AX2=AX**2
0160        IF(L.LT.7)THEN
0161            IF(L.EQ.0)THEN
0162                IF(AX.LT.1.D-1)TH
0163                    JL=1.D0-AX2/6
0164                ELSE
0165                    JL=DSIN(AX)/A
0166                ENDIF
0167
0168            ELSEIF(L.EQ.1)THEN
0169                IF(AX.LT.2.D-1)TH
0170                    JL=AX/3.D0*(1
0171                ELSE
0172                    JL=(DSIN(AX)/
0173                ENDIF
0174            ELSEIF(L.EQ.2)THEN
0175                IF(AX.LT.3.D-1)TH
0176                    JL=AX2/15.D0*
0177                ELSE
0178                    JL=(-3.0D0*DC
0179                ENDIF
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 181

```
0181        ELSEIF(L.EQ.3)THE
0182            IF(AX.LT.4.D-
0183                JL=AX*AX2
0184            ELSE
0185                JL=(DCOS(
0186            ENDIF
0187        ELSEIF(L.EQ.4)THE
0188            IF(AX.LT.6.D-
0189                JL=AX2**2
0190            ELSE
0191                JL=(DSIN(
0192            ENDIF
0193        ELSEIF(L.EQ.5)THE
0194            IF(AX.LT.1.D0
0195                JL=AX2**2
0196            ELSE
0197                JL=(DSIN(
0198            ENDIF
0199        ELSE
0200            IF(AX.LT.1.D0
0201                JL=AX2**3
0202            ELSE
0203                JL=(DSIN(
0204                    DCOS(
0205            ENDIF
0206        ENDIF
0207    ELSE
0208        NU=0.5D0+L
0209        NU2=NU**2
0210        IF(AX.LT.1.D-40)T
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 180

```
0180        ELSEIF(L.EQ.3)THEN
0181            IF(AX.LT.4.D-1)TH
0182                JL=AX*AX2/105
0183            ELSE
0184                JL=(DCOS(AX)*
0185            ENDIF
0186        ELSEIF(L.EQ.4)THEN
0187            IF(AX.LT.6.D-1)TH
0188                JL=AX2**2/945
0189            ELSE
0190                JL=(DSIN(AX)*
0191            ENDIF
0192        ELSEIF(L.EQ.5)THEN
0193            IF(AX.LT.1.D0)THE
0194                JL=AX2**2*AX/
0195            ELSE
0196                JL=(DSIN(AX)*
0197            ENDIF
0198        ELSE
0199            IF(AX.LT.1.D0)THE
0200                JL=AX2**3/135
0201            ELSE
0202                JL=(DSIN(AX)*
0203                    DCOS(AX)*
0204            ENDIF
0205        ENDIF
0206    ELSE
0207        NU=0.5D0+L
0208        NU2=NU**2
0209        IF(AX.LT.1.D-40)THEN
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 211

```
0211          JL=0.D0
0212        ELSEIF((AX2/L).LT
0213          JL=DEXP(L*DLO
0214             /NU*(1.D0-
0215        ELSEIF((real(L,dl
0216          BETA=AX-PID2*
0217          JL=(DCOS(BETA
0218             -DSIN(BETA
0219                (NU2-2
0220        ELSE
0221          L3=NU**0.325
0222          IF(AX .LT. NU
0223            COSB=NU/A
0224            SX = DSQR
0225            COTB=NU/S
0226            SECB=AX/N
0227            BETA=DLOG
0228            COT3B=COT
0229            COT6B=COT
0230            SEC2B=SEC
0231            EXPTERM=(
0232               - ( (4
0233               + ((16
0234               + (32.
0235               /NU)/N
0236            JL=DSQRT(
0237
0238          !           /*
0239
0240          ELSEIF (AX .G
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 210

```
0210          JL=0.D0
0211        ELSEIF((AX2/L).LT.5.D
0212          JL=DEXP(L*DLOG(AX
0213             /NU*(1.D0-AX2
0214        ELSEIF((real(L,dl)**2
0215          BETA=AX-PID2*(L+1
0216          JL=(DCOS(BETA)*(1
0217             -DSIN(BETA)*(
0218              (NU2-20.25)/8
0219        ELSE
0220          L3=NU**0.325
0221          IF(AX .LT. NU-1.3
0222            COSB=NU/AX
0223            SX = DSQRT(NU
0224            COTB=NU/SX
0225            SECB=AX/NU
0226            BETA=DLOG(COS
0227            COT3B=COTB**3
0228            COT6B=COT3B**
0229            SEC2B=SECB**2
0230            EXPTERM=( (2.
0231               - ( (4.D0
0232               + ((16.D0
0233               + (32.D0+
0234               /NU)/NU
0235            JL=DSQRT(COTB
0236
0237          !           /*
0238
0239          ELSEIF (AX .GT. N
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 241

```
0241        COSB=NU/A
0242        SX=DSQRT(
0243        COTB=NU/S
0244        SECB=AX/N
0245        BETA=DACO
0246        COT3B=COT
0247        COT6B=COT
0248        SEC2B=SEC
0249        TRIGARG=N
0250                -(
0251                +(
0252        EXPTERM=(
0253                -(
0254        JL=DSQRT(
0255
0256    !           /*
0257
0258    ELSE
0259        BETA=AX-N
0260        BETA2=BET
0261        SX=6.D0/A
0262        SX2=SX**2
0263        SECB=SX**
0264        SEC2B=SEC
0265        JL=( GAMM
0266               -(B
0267               -((
0268               +((
0269               +((
0270               -((
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 240

```
0240        COSB=NU/AX
0241        SX=DSQRT(AX2-
0242        COTB=NU/SX
0243        SECB=AX/NU
0244        BETA=DACOS(CO
0245        COT3B=COTB**3
0246        COT6B=COT3B**
0247        SEC2B=SECB**2
0248        TRIGARG=NU/CO
0249            -((2.0+3.
0250            +(16.D0-(
0251        EXPTERM=( (4.
0252            -(32.D0+(
0253        JL=DSQRT(COTB
0254
0255    !           /*
0256
0257    ELSE
0258        BETA=AX-NU
0259        BETA2=BETA**2
0260        SX=6.D0/AX
0261        SX2=SX**2
0262        SECB=SX**0.33
0263        SEC2B=SECB**2
0264        JL=( GAMMA1*S
0265            -(BETA2/1
0266            -((BETA2-
0267            +(((BETA2
0268            +(((BETA2
0269            -((((BETA
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 271

```
0271                                        BE
0272                       ENDIF
0273                   ENDIF
0274               ENDIF
0275               IF(X.LT.0.AND.MOD(L,2
0276               END SUBROUTINE BJL
0277
0278      !     end module SpherBessels
0279
0280
0281
0282
0283     !ccccccccccccccccccccccccccc
0284     !
0285     ! Calculation of ultraspheric
0286     ! Fortran version of the c pr
0287     ! WKB approx described in ast
0288     !
0289     ! Modifications by Anthony Ch
0290     ! Minor modifications to corr
0291     ! the small chi approximation
0292     ! the quadratic approximation
0293     ! Bug fixed in downwards recu
0294     !
0295     ! The routine phi_recurs uses
0296     ! the functions, which is acc
0297     !   ***NOT STABLE FOR K=1 or
0298     !
0299     ! The routine phi_langer uses
0300     ! uniform first-order asympto
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 270

```
0270                                    BETA2+793
0271                   ENDIF
0272               ENDIF
0273           ENDIF
0274           IF(X.LT.0.AND.MOD(L,2).NE
0275           END SUBROUTINE BJL
0276
0277        !     end module SpherBess
0278
0279
0280
0281
0282        !ccccccccccccccccccccccccc
0283        !
0284        ! Calculation of ultrasph
0285        ! Fortran version of the
0286        ! WKB approx described in
0287        !
0288        ! Modifications by Anthon
0289        ! Minor modifications to
0290        ! the small chi approxima
0291        ! the quadratic approxima
0292        ! Bug fixed in downwards
0293        !
0294        ! The routine phi_recurs
0295        ! the functions, which is
0296        !   ***NOT STABLE FOR K=1
0297        !
0298        ! The routine phi_langer
0299        ! uniform first-order asy
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 301

```
0301    ! and flat cases. This approx
0302    !
0303    ! The routine qintegral calcu
0304    ! to the eikonal integral use
0305    !
0306    ! The routine airy_ai returns
0307    ! passed. It employs a Pade-t
0308    ! a Taylor expansion around z
0309    !
0310    ! The routines polevl and p1e
0311    ! evaluation routines used in
0312    !
0313    !ccccccccccccccccccccccccccc
0314
0315
0316
0317    ! module USpherBessels
0318    ! use Precision
0319    ! implicit none
0320    ! private
0321
0322
0323     !public USpherBesselWithDeri
0324
0325
0326    ! contains
0327
0328          subroutine USpherBess
0329            !returns y1=ujl*sin
0330            !aim for accuracy >
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 300

```
0300        ! and flat cases. This ap
0301        !
0302        ! The routine qintegral c
0303        ! to the eikonal integral
0304        !
0305        ! The routine airy_ai ret
0306        ! passed. It employs a Pa
0307        ! a Taylor expansion arou
0308        !
0309        ! The routines polevl and
0310        ! evaluation routines use
0311        !
0312        !ccccccccccccccccccccccc
0313
0314
0315
0316        ! module USpherBessels
0317        ! use Precision
0318        ! implicit none
0319        ! private
0320
0321
0322         !public USpherBesselWithD
0323
0324
0325        ! contains
0326
0327         subroutine USpherBesselWi
0328         !returns y1=ujl*sinhChi a
0329         !aim for accuracy > 1% fo
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 331

```
0331        real(dl) Chi,beta,y1,
0332        real(dl) sin_K, cot_K
0333        integer l,K
0334        logical, intent(IN) :
0335        logical DoRecurs
0336
0337        if (closed) then
0338            sin_K = sin(Chi)
0339            cot_K= 1._dl/tan(
0340            K=1
0341         else
0342            sin_K=sinh(Chi)
0343            cot_K = 1._dl/tan
0344            K=-1
0345         end if
0346
0347        sinhChi = sin_K
0348        cothChi = cot_K
0349
0350       DoRecurs = ((l<=45*Acc
0351            .or.closed.and.(b
0352
0353    !Deep in the tails the closed
0354    !Added July 2003 to prevent p
0355        if (DoRecurs .and. clo
0356         if  (Chi < asin(sqrt(
0357            if (phi_langer(l,K
0358              call phi_small_c
0359              return
0360            end if
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 330

```
0330        real(dl) Chi,beta,y1,y2,s
0331        real(dl) sin_K, cot_K
0332        integer l,K
0333        logical, intent(IN) :: cl
0334        logical DoRecurs
0335
0336        if (closed) then
0337            sin_K = sin(Chi)
0338            cot_K= 1._dl/tan(Chi)
0339            K=1
0340        else
0341            sin_K=sinh(Chi)
0342            cot_K = 1._dl/tanh(Ch
0343            K=-1
0344        end if
0345
0346        sinhChi = sin_K
0347        cothChi = cot_K
0348
0349        DoRecurs = ((l<=45*Accura
0350            .or.closed.and.(beta*
0351
0352     !Deep in the tails the cl
0353     !Added July 2003 to preve
0354        if (DoRecurs .and. closed
0355            if  (Chi < asin(sqrt(
0356                if (phi_langer(l,
0357                    call phi_smal
0358                    return
0359                end if
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 361

```
0361            end if
0362         end if
0363
0364         if (DoRecurs) then
0365                 !use recursive e
0366            y1=phi_recurs(l,K,be
0367            y2=y1*(l+1)*cothChi
0368            if (.not.closed.or.(
0369                 sqrt(beta**2-(K
0370             !of course we could
0371             !phi_recurs to retu
0372
0373
0374            else !WKB approx
0375            y1=phi_langer(l,K,be
0376            y2=y1*(l+1)*cothChi
0377            if (.not.closed.or.(
0378                 sqrt(beta**2-(K
0379
0380            end if
0381
0382         end subroutine USpher
0383
0384
0385
0386   !Calculates y1,y2 (noramlized
0387   !by integrating up the differ
0388   !in the region in which phi_r
0389   !This allows closed functions
0390      subroutine phi_small_close
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 360

```
0360         end if
0361      end if
0362
0363      if (DoRecurs) then
0364          !use recursive evalua
0365          y1=phi_recurs(l,K,bet
0366          y2=y1*(l+1)*cothChi
0367          if (.not.closed.or.(l
0368              sqrt(beta**2-(K*(
0369          !of course we could g
0370          !phi_recurs to return
0371
0372
0373      else !WKB approx
0374          y1=phi_langer(l,K,bet
0375          y2=y1*(l+1)*cothChi
0376          if (.not.closed.or.(l
0377              sqrt(beta**2-(K*(
0378
0379      end if
0380
0381      end subroutine USpherBess
0382
0383
0384
0385      !Calculates y1,y2 (noraml
0386      !by integrating up the di
0387      !in the region in which p
0388      !This allows closed funct
0389      subroutine phi_small_clos
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 391

```
0391        integer, intent(IN) :: l
0392        real(dl), intent(IN) :: be
0393        real(dl) y1,y2
0394
0395        integer nsteps,i
0396
0397        real(dl) ap1,nu2,dydchi1,d
0398        real(dl) x0, delchi,sh, h6
0399        real(dl) y1_x,y2_x, tmp,xh
0400
0401        nsteps = 200
0402        ap1 = l*(l+1)
0403        x0 = sqrt(ap1)/beta
0404        nu2 = beta**2
0405
0406        if ((beta*chi)**2/l < 0.00
0407      !Series solution
0408
0409          x = chi
0410          sh = sin(x)
0411          tmp=(ap1/sh**2 - nu2)
0412          y1=1e-20
0413          y2 = ((l+1)/x - (nu2-ap1
0414        else
0415
0416        x = max(1d-7,chi - 50._dl/
0417        y1=1e-20
0418        y2 = (l+1)*y1/x
0419
0420        delchi = (chi-x)/nSteps
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 390

```
0390        integer, intent(IN) :: l
0391        real(dl), intent(IN) :: b
0392        real(dl) y1,y2
0393
0394        integer nsteps,i
0395
0396        real(dl) ap1,nu2,dydchi1,
0397        real(dl) x0, delchi,sh, h
0398        real(dl) y1_x,y2_x, tmp,x
0399
0400        nsteps = 200
0401        ap1 = l*(l+1)
0402        x0 = sqrt(ap1)/beta
0403        nu2 = beta**2
0404
0405        if ((beta*chi)**2/l < 0.0
0406          !Series solution
0407
0408          x = chi
0409          sh = sin(x)
0410          tmp=(ap1/sh**2 - nu2)
0411          y1=1e-20
0412          y2 = ((l+1)/x - (nu2-
0413        else
0414
0415          x = max(1d-7,chi - 50
0416          y1=1e-20
0417          y2 = (l+1)*y1/x
0418
0419          delchi = (chi-x)/nSte
```

/Users/lp1opa/Compare/camb_simdata/besselS.f90, Top line: 421

```
0421        h6=delchi/6
0422        hh=delchi/2
0423        sh = sin(x)
0424        tmp=(ap1/sh**2 - nu2)
0425
0426              do i=1,nSteps
0427     ! One step in the ujl integra
0428     ! fourth-order Runge-Kutta me
0429
0430                  dydchi1=y2
0431                  dydchi2=tmp*y1
0432                  xh=x+hh
0433                  yt1=y1+hh*dydchi1
0434                  yt2=y2+hh*dydchi2
0435                  dyt1=yt2
0436                  tmp=(ap1/sin(xh)*
0437
0438
0439                  dyt2=tmp*yt1
0440
0441                  yt1=y1+hh*dyt1
0442                  yt2=y2+hh*dyt2
0443
0444                  dym1=yt2
0445                  dym2=tmp*yt1
0446                  yt1=y1+delchi*dym
0447                  dym1=dyt1+dym1
0448                  yt2=y2+delchi*dym
0449                  dym2=dyt2+dym2
0450
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 420

```
0420        h6=delchi/6
0421        hh=delchi/2
0422        sh = sin(x)
0423        tmp=(ap1/sh**2 - nu2)
0424
0425              do i=1,nSteps
0426                 ! One step in the
0427                 ! fourth-order Ru
0428
0429                  dydchi1=y2
0430                  dydchi2=tmp*y1
0431                  xh=x+hh
0432                  yt1=y1+hh*dydchi1
0433                  yt2=y2+hh*dydchi2
0434                  dyt1=yt2
0435                  tmp=(ap1/sin(xh)*
0436
0437
0438                  dyt2=tmp*yt1
0439
0440                  yt1=y1+hh*dyt1
0441                  yt2=y2+hh*dyt2
0442
0443                  dym1=yt2
0444                  dym2=tmp*yt1
0445                  yt1=y1+delchi*dym
0446                  dym1=dyt1+dym1
0447                  yt2=y2+delchi*dym
0448                  dym2=dyt2+dym2
0449
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 451

```
0451                  x=x+delchi      !e
0452                  sh=sin(x)
0453                  dyt1=yt2
0454                  tmp=(ap1/sh**2 -
0455                  dyt2=tmp*yt1
0456                  y1=y1+h6*(dydchi1
0457                  y2=y2+h6*(dydchi2
0458                 if (y1 > 1d10 .or.
0459
0460                    y1=y1/1d10
0461                    y2=y2/1d10
0462
0463                end if
0464              end do
0465
0466          end if
0467
0468           y1_x = y1; y2_x = y2
0469
0470           delchi = (x0 - chi)/nS
0471           h6=delchi/6
0472           hh=delchi/2
0473
0474              do i=1,nSteps
0475    ! One step in the ujl integra
0476    ! fourth-order Runge-Kutta me
0477
0478                dydchi1=y2
0479                dydchi2=tmp*y1
0480                xh=x+hh
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 450

```
0450                  x=x+delchi      !e
0451                  sh=sin(x)
0452                  dyt1=yt2
0453                  tmp=(ap1/sh**2 -
0454                  dyt2=tmp*yt1
0455                  y1=y1+h6*(dydchi1
0456                  y2=y2+h6*(dydchi2
0457                  if (y1 > 1d10 .or
0458
0459                     y1=y1/1d10
0460                     y2=y2/1d10
0461
0462                end if
0463             end do
0464
0465          end if
0466
0467          y1_x = y1; y2_x = y2
0468
0469          delchi = (x0 - chi)/nStep
0470          h6=delchi/6
0471          hh=delchi/2
0472
0473          do i=1,nSteps
0474             ! One step in the ujl
0475             ! fourth-order Runge-
0476
0477             dydchi1=y2          !d
0478             dydchi2=tmp*y1      !d
0479             xh=x+hh           !mid
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 481

```
0481            yt1=y1+hh*dydchi1
0482            yt2=y2+hh*dydchi2
0483            dyt1=yt2
0484            tmp=(ap1/sin(xh)*
0485
0486
0487            dyt2=tmp*yt1
0488
0489            yt1=y1+hh*dyt1
0490            yt2=y2+hh*dyt2
0491
0492            dym1=yt2
0493            dym2=tmp*yt1
0494            yt1=y1+delchi*dym
0495            dym1=dyt1+dym1
0496            yt2=y2+delchi*dym
0497            dym2=dyt2+dym2
0498
0499            x=x+delchi      !e
0500            sh=sin(x)
0501            dyt1=yt2
0502            tmp=(ap1/sh**2 -
0503            dyt2=tmp*yt1
0504            y1=y1+h6*(dydchi1
0505            y2=y2+h6*(dydchi2
0506            if (y1 > 1d10 .or
0507              y1=y1/1d10
0508              y2=y2/1d10
0509              y1_x = y1_x/1d1
0510              y2_x = y2_x/1d1
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 480

```
0480            yt1=y1+hh*dydchi1   !y
0481            yt2=y2+hh*dydchi2   !y
0482            dyt1=yt2            !d
0483            tmp=(ap1/sin(xh)**2 -
0484
0485
0486            dyt2=tmp*yt1        !d
0487
0488            yt1=y1+hh*dyt1      !y
0489            yt2=y2+hh*dyt2      !y
0490
0491            dym1=yt2            !d
0492            dym2=tmp*yt1        !d
0493            yt1=y1+delchi*dym1  !y
0494            dym1=dyt1+dym1
0495            yt2=y2+delchi*dym2  !y
0496            dym2=dyt2+dym2
0497
0498            x=x+delchi      !end p
0499            sh=sin(x)
0500            dyt1=yt2            !d
0501            tmp=(ap1/sh**2 - nu2)
0502            dyt2=tmp*yt1        !d
0503            y1=y1+h6*(dydchi1+dyt
0504            y2=y2+h6*(dydchi2+dyt
0505            if (y1 > 1d10 .or. y2
0506              y1=y1/1d10
0507              y2=y2/1d10
0508              y1_x = y1_x/1d10
0509              y2_x = y2_x/1d10
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 511

```
0511
0512              end if
0513           end do
0514
0515
0516           tmp = phi_recurs(l,1
0517           y1 = y1_x * tmp
0518           y2 = y2_x * tmp
0519
0520
0521        end subroutine phi_small_c
0522
0523        !**************************
0524        !
0525        ! Calculates Phi(l,beta,chi)
0526        ! See Abbot and Schaefer, ApJ
0527        ! recursion relations and clo
0528        ! (Note: Their variable y is
0529        !
0530        ! When the flag direction is
0531        ! must be used because the up
0532        ! errors. The downwards recur
0533        ! continues downwards to l=1,
0534        ! using the closed form solut
0535        ! Recipes of Bessel functions
0536        !
0537        !**************************
0538
0539        function phi_recurs(l, K, b
0540        !doesn't like values which
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 510

```
0510
0511           end if
0512        end do
0513
0514
0515        tmp = phi_recurs(l,1,beta
0516        y1 = y1_x * tmp
0517        y2 = y2_x * tmp
0518
0519
0520     end subroutine phi_small_
0521
0522     !**************************
0523     !
0524     ! Calculates Phi(l,beta,c
0525     ! See Abbot and Schaefer,
0526     ! recursion relations and
0527     ! (Note: Their variable y
0528     !
0529     ! When the flag direction
0530     ! must be used because th
0531     ! errors. The downwards r
0532     ! continues downwards to
0533     ! using the closed form s
0534     ! Recipes of Bessel funct
0535     !
0536     !**************************
0537
0538     function phi_recurs(l, K,
0539     !doesn't like values whic
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 541

```
0541      integer, intent(IN) :: l, K
0542      real(dl), intent(IN) :: bet
0543      real(dl) phi_recurs
0544      integer j, direction, lstar
0545      real(dl) ell, kay, arg, ans
0546      real(dl) root_K
0547      real(dl) phi0, phi1, phi_pl
0548      real(dl), parameter :: ACC=
0549      real(dl) sin_K, cot_K
0550
0551      ell=dble(l)
0552
0553
0554      ! Test input values
0555
0556      if(l<0) then
0557         write(*,*) "Bessel funct
0558         stop
0559      endif
0560      if(beta<0._dl) then
0561         write(*,*) "Wavenumber b
0562         stop
0563      endif
0564      if ((abs(K)/=1).and.(K/=0))
0565         write(*,*) "K must be 1,
0566         stop
0567      end if
0568
0569      if(K==1) then
0570         ibeta=nint(beta)
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 540

```
0540      integer, intent(IN) :: l,
0541      real(dl), intent(IN) :: b
0542      real(dl) phi_recurs
0543      integer j, direction, lst
0544      real(dl) ell, kay, arg, a
0545      real(dl) root_K
0546      real(dl) phi0, phi1, phi_
0547      real(dl), parameter :: AC
0548      real(dl) sin_K, cot_K
0549
0550      ell=dble(l)
0551
0552
0553      ! Test input values
0554
0555      if(l<0) then
0556         call MpiStop("Bessel
0557      endif
0558      if(beta<0._dl) then
0559         call MpiStop("Wavenum
0560      endif
0561      if ((abs(K)/=1).and.(K/=0
0562         call MpiStop("K must
0563      end if
0564
0565      if(K==1) then
0566         ibeta=nint(beta)
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 571

```
0571        if(ibeta<3) then
0572           write(*,*) "Wavenumbe
0573           stop
0574        endif
0575        if(ibeta<=l) then
0576           write(*,*) "Wavenumbe
0577           stop
0578        endif
0579     endif
0580
0581     if (chi<1/BIG) then
0582        phi_recurs=0
0583        return
0584     end if
0585
0586     kay = dble(K)
0587     arg = beta * chi
0588     beta2 = beta**2
0589
0590     if(K == 0) then
0591        cot_K = 1._dl/chi
0592        sin_K = chi
0593        root_K = beta
0594     else
0595      root_K = sqrt(beta2 -kay*e
0596
0597       if(K == -1) then
0598       cot_K = 1._dl/tanh(chi)
0599       sin_K = sinh(chi)
0600       else
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 567

```
0567        if(ibeta<3) then
0568           call MpiStop("Wav

0569        endif
0570        if(ibeta<=l) then
0571           call MpiStop("Wav

0572        endif
0573     endif
0574
0575     if (chi<1/BIG) then
0576        phi_recurs=0
0577        return
0578     end if
0579
0580     kay = dble(K)
0581     arg = beta * chi
0582     beta2 = beta**2
0583
0584     if(K == 0) then
0585        cot_K = 1._dl/chi
0586        sin_K = chi
0587        root_K = beta
0588     else
0589        root_K = sqrt(beta2 -
0590
0591        if(K == -1) then
0592           cot_K = 1._dl/tan
0593           sin_K = sinh(chi)
0594        else
```

/Users/lp1opa/Compare/camb_simdata/besselsf90, Top line: 601

```
0601        cot_K = 1._dl/tan(chi)
0602        sin_K = sin(chi)
0603        end if
0604
0605      endif
0606
0607
0608      ! Closed form solution for
0609
0610      if (abs(chi) < 1.d-4) then
0611        if (abs(arg)<1.d-4) then
0612         phi0 = 1._dl-chi**2*(bet
0613        else
0614         phi0=sin(arg)/arg
0615        end if
0616      else
0617         phi0 = sin(arg) / (beta
0618      end if
0619
0620      if (l==0) then
0621           phi_recurs=phi0
0622           return
0623       end if
0624
0625
0626        ! Closed form solution f
0627
0628        if((abs(chi) < 1.d-4).an
0629           if(arg < 1.d-4) then
0630              phi1 = chi*sqrt(be
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 595

```
0595            cot_K = 1._dl/tan
0596            sin_K = sin(chi)
0597         end if
0598
0599      endif
0600
0601
0602      ! Closed form solution fo
0603
0604      if (abs(chi) < 1.d-4) the
0605         if (abs(arg)<1.d-4) t
0606            phi0 = 1._dl-chi*
0607         else
0608            phi0=sin(arg)/arg
0609         end if
0610      else
0611         phi0 = sin(arg) / (be
0612      end if
0613
0614      if (l==0) then
0615         phi_recurs=phi0
0616         return
0617      end if
0618
0619
0620      ! Closed form solution fo
0621
0622      if((abs(chi) < 1.d-4).and
0623         if(arg < 1.d-4) then
0624            phi1 = chi*sqrt(b
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 631

```
0631            !beta2 * chi / (3.
0632         else
0633            phi1 = (sin(arg)/a
0634              !(sin(arg)/arg -
0635         end if
0636      elseif ((abs(arg) < 1.d-
0637         phi1 = arg / 3._dl
0638      else
0639         if (K /= 0 ) then
0640            phi1 = sin(arg) *
0641            phi1 = phi1/sqrt(b
0642         else
0643            phi1 = (sin(arg)/a
0644         end if
0645      end if
0646      if(l==1) then
0647         phi_recurs=phi1
0648         return
0649      end if
0650      ! Find recursion directi
0651      !  direction = +1 for up
0652
0653      if(abs(cot_K) < root_K /
0654         direction = 1
0655      else
0656         direction = -1
0657      end if
0658
0659      ! For K=1, must do upwar
0660      ! NOT STABLE for all val
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 625

```
0625            !beta2 * chi / (3
0626         else
0627            phi1 = (sin(arg)/
0628            !(sin(arg)/arg -
0629         end if
0630      elseif ((abs(arg) < 1.d-4
0631         phi1 = arg / 3._dl
0632      else
0633         if (K /= 0 ) then
0634            phi1 = sin(arg) *
0635            phi1 = phi1/sqrt(
0636         else
0637            phi1 = (sin(arg)/
0638         end if
0639      end if
0640      if(l==1) then
0641         phi_recurs=phi1
0642         return
0643      end if
0644      ! Find recursion directio
0645      !  direction = +1 for upw
0646
0647      if(abs(cot_K) < root_K /
0648         direction = 1
0649      else
0650         direction = -1
0651      end if
0652
0653      ! For K=1, must do upward
0654      ! NOT STABLE for all valu
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 661

```
0661
0662        if(K==1) direction = 1
0663
0664        ! Do upwards recursion o
0665
0666
0667        if(direction == 1)then
0668           b_minus = sqrt(beta2
0669           phi_minus = phi0
0670           phi_zero = phi1
0671
0672           do j=2,l
0673
0674              if(K == 0) then
0675                 phi_plus = ((2*
0676              else
0677                 b_zero = sqrt(b
0678                 phi_plus = ((2*
0679                 b_minus = b_zer
0680              end if
0681              phi_minus = phi_ze
0682              phi_zero = phi_plu
0683           end do
0684
0685
0686           phi_recurs=phi_plus
0687
0688
0689           return
0690
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 655

```
0655
0656        if(K==1) direction = 1
0657
0658        ! Do upwards recursion on
0659
0660
0661        if(direction == 1)then
0662           b_minus = sqrt(beta2
0663           phi_minus = phi0
0664           phi_zero = phi1
0665
0666           do j=2,l
0667
0668              if(K == 0) then
0669                 phi_plus = ((
0670              else
0671                 b_zero = sqrt
0672                 phi_plus = ((
0673                 b_minus = b_z
0674              end if
0675              phi_minus = phi_z
0676              phi_zero = phi_pl
0677           end do
0678
0679
0680           phi_recurs=phi_plus
0681
0682
0683           return
0684
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 691

```
0691            ! Do downwards recurs
0692
0693         else
0694            lstart = l + 2 * int(
0695
0696            b_zero = sqrt(beta2 -
0697            phi_plus = 0._dl
0698            phi_zero = 1._dl
0699            answer = 0._dl
0700
0701            do j= lstart - 2,1,-1
0702
0703               if(K == 0) then
0704                  phi_minus = ((2
0705               else
0706                  b_minus = sqrt(
0707                  phi_minus = ((2
0708                  b_zero = b_minu
0709               end if
0710               phi_plus = phi_zer
0711               phi_zero = phi_min
0712
0713               if(j == l) answer
0714               if((abs(phi_zero)
0715                  phi_plus = phi_
0716                  phi_zero = phi_
0717                  answer = answer
0718               end if
0719            end do
0720
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 685

```
0685            ! Do downwards recurs
0686
0687         else
0688            lstart = l + 2 * int(
0689
0690            b_zero = sqrt(beta2 -
0691            phi_plus = 0._dl
0692            phi_zero = 1._dl
0693            answer = 0._dl
0694
0695            do j= lstart - 2,1,-1
0696
0697               if(K == 0) then
0698                  phi_minus = (
0699               else
0700                  b_minus = sqr
0701                  phi_minus = (
0702                  b_zero = b_mi
0703               end if
0704               phi_plus = phi_ze
0705               phi_zero = phi_mi
0706
0707               if(j == l) answer
0708               if((abs(phi_zero)
0709                  phi_plus = ph
0710                  phi_zero = ph
0711                  answer = answ
0712               end if
0713            end do
0714
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 721

```
0721            ! Normalize answer to
0722
0723            answer = answer*phi1
0724            phi_recurs=answer
0725
0726         end if
0727
0728       end function phi_recurs
0729
0730
0731    !ccccccccccccccccccccccccccc
0732    !
0733    ! Calculates Phi(l,beta,chi)
0734    ! to the first-order WKB appr
0735    ! See C.M. Bender and S.A. Or
0736    ! Scientists and Engineers (M
0737    ! chapter 10.
0738    !
0739    ! Differential equation for n
0740    ! Schrodinger form       \epsi
0741    ! where \epsilon^2 = 1/l(l+1)
0742    ! alpha \equiv beta * epsilon
0743    !
0744    ! In the K= +1 case, the func
0745    ! determined by its value on
0746    ! conditions Phi(chi + pi) =
0747    !              Phi(pi - chi) =
0748    ! This interval contains one
0749    ! can be used.
0750    ! Note that the second condit
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 715

```
0715            ! Normalize answer to
0716
0717            answer = answer*phi1
0718            phi_recurs=answer
0719
0720         end if
0721
0722       end function phi_recurs
0723
0724
0725    !ccccccccccccccccccccccccc
0726    !
0727    ! Calculates Phi(l,beta,c
0728    ! to the first-order WKB
0729    ! See C.M. Bender and S.A
0730    ! Scientists and Engineer
0731    ! chapter 10.
0732    !
0733    ! Differential equation f
0734    ! Schrodinger form      \
0735    ! where \epsilon^2 = 1/l(
0736    ! alpha \equiv beta * eps
0737    !
0738    ! In the K= +1 case, the
0739    ! determined by its value
0740    ! conditions Phi(chi + pi
0741    !              Phi(pi - chi
0742    ! This interval contains
0743    ! can be used.
0744    ! Note that the second co
```

/Users/lp1opa/Compare/camb_simdata/besselsf90, Top line: 751

```
0751     ! condition on beta, which  m
0752     ! eigenvalue(s), the region b
0753     ! enough for the asymptotic s
0754     ! have a small discontinuity
0755     ! this behavior is corrected
0756     ! series around the regular p
0757     ! The exact eigenvalue condit
0758     ! integer >= 3 with beta > 1.
0759     !
0760     !ccccccccccccccccccccccccccc
0761
0762
0763
0764         function phi_langer(l,K
0765         integer l,K,ibeta,kay
0766         real(dl) phi_langer
0767         real(dl) ell,symm, anu,
0768         real(dl) beta,chi,eikon
0769         real(dl) epsilon, alpha
0770
0771         real(dl) cot_K, sin_K
0772         real(dl), parameter ::
0773                                  PI
0774
0775         ell=dble(l)
0776         achi=chi
0777
0778         symm=1._dl
0779     !
0780     ! Test input values
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 745

```
0745         ! condition on beta, whic
0746         ! eigenvalue(s), the regi
0747         ! enough for the asymptot
0748         ! have a small discontinu
0749         ! this behavior is correc
0750         ! series around the regul
0751         ! The exact eigenvalue co
0752         ! integer >= 3 with beta
0753         !
0754         !ccccccccccccccccccccccc
0755
0756
0757
0758         function phi_langer(l,K,b
0759         integer l,K,ibeta,kay
0760         real(dl) phi_langer
0761         real(dl) ell,symm, anu, a
0762         real(dl) beta,chi,eikonal
0763         real(dl) epsilon, alpha,
0764
0765         real(dl) cot_K, sin_K
0766         real(dl), parameter :: PI
0767             PIOVER2=1.57079632d0
0768
0769         ell=dble(l)
0770         achi=chi
0771
0772         symm=1._dl
0773         !
0774         ! Test input values
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 781

```
0781    !
0782          if(l<0) then
0783             write(*,*) "Bessel f
0784             stop
0785          endif
0786          if(beta<0._dl) then
0787             write(*,*) "Wavenumb
0788             stop
0789          endif
0790          if ((abs(K)/=1).and.(K/
0791            write(*,*) "K must be
0792            stop
0793          end if
0794
0795
0796          if(K == 1) then
0797             ibeta=nint(beta)
0798             if(ibeta<3) then
0799                write(*,*) "Waven
0800                stop
0801             endif
0802             if(ibeta<=l) then
0803                write(*,*) "Waven
0804                stop
0805             endif
0806          endif
0807
0808          kay=K
0809
0810
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 775

```
0775    !
0776          if(l<0) call MpiStop("Bes
0777          if(beta<0._dl) call MpiSt
0778          if ((abs(K)/=1).and.(K/=0
0779              call MpiStop("K must
0780          end if
0781
0782
0783          if(K == 1) then
0784             ibeta=nint(beta)
0785             if(ibeta<3) call MpiS
0786             if(ibeta<=l) call Mpi
0787          endif
0788
0789          kay=K
0790
0791
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 811

```
0811    ! For closed case, find equiv
0812    !
0813        if(K==1) then
0814           achi=achi-2._dl*Pi*i
0815           if(achi>PI) then
0816              achi=2._dl*PI-ach
0817              if(2*(l/2).eq.l)
0818                 symm=symm
0819              else
0820                 symm=-symm
0821              endif
0822           endif
0823           if(achi>PI/2._dl) th
0824              achi=PI-achi
0825              if(2*((ibeta-l-1)
0826                 symm=symm
0827              else
0828                 symm=-symm
0829              endif
0830           endif
0831        endif
0832
0833    ! Definitions
0834        if(K == 0) then
0835            sin_K = achi
0836        else
0837              if(K == -1) then
0838                 sin_K = sinh(a
0839               else
0840                 sin_K = sin(ac
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 792

```
0792        ! For closed case, find e
0793        !
0794           if(K==1) then
0795              achi=achi-2._dl*Pi*in
0796              if(achi>PI) then
0797                 achi=2._dl*PI-ach
0798                 if(2*(l/2).eq.l)
0799                    symm=symm
0800                 else
0801                    symm=-symm
0802                 endif
0803              endif
0804              if(achi>PI/2._dl) the
0805                 achi=PI-achi
0806                 if(2*((ibeta-l-1)
0807                    symm=symm
0808                 else
0809                    symm=-symm
0810                 endif
0811              endif
0812           endif
0813
0814        ! Definitions
0815           if(K == 0) then
0816              sin_K = achi
0817           else
0818              if(K == -1) then
0819                 sin_K = sinh(achi
0820              else
0821                 sin_K = sin(achi)
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 841

```
0841        end if
0842      endif
0843
0844    ! Closed form solution for l=
0845    !
0846      if(l == 0) then
0847        arg=beta*achi
0848
0849        if((abs(achi)<1.d-4)
0850          if(abs(arg)<1.d-4
0851            wkb=1._dl-achi
0852          else
0853            wkb=sin(arg)/a
0854          endif
0855        else if((abs(arg)<1.
0856          wkb=1._dl-arg*arg
0857        else
0858          wkb=sin(arg)/(bet
0859        endif
0860        phi_langer=symm*wkb
0861        return
0862      endif
0863    !
0864    ! Closed form solution for l=
0865    !
0866      if(l==1) then
0867        arg=beta*achi
0868
0869        if((abs(achi)<1.d-4)
0870          if(abs(arg)<1.d-4
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 822

```
0822        end if
0823      endif
0824
0825    ! Closed form solution fo
0826    !
0827      if(l == 0) then
0828        arg=beta*achi
0829
0830        if((abs(achi)<1.d-4).
0831          if(abs(arg)<1.d-4
0832            wkb=1._dl-ach
0833          else
0834            wkb=sin(arg)/
0835          endif
0836        else if((abs(arg)<1.d
0837          wkb=1._dl-arg*arg
0838        else
0839          wkb=sin(arg)/(bet
0840        endif
0841        phi_langer=symm*wkb
0842        return
0843      endif
0844    !
0845    ! Closed form solution fo
0846    !
0847      if(l==1) then
0848        arg=beta*achi
0849
0850        if((abs(achi)<1.d-4).
0851          if(abs(arg)<1.d-4
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 871

```
0871                  wkb=achi*sqrt(
0872                else
0873                   wkb=(sin(arg)/
0874                endif
0875             else if((abs(arg)<1.
0876                wkb=arg/3._dl
0877             else
0878              if(K/=0) then
0879                if(K==1) then
0880                   cot_K=1._dl/ta
0881                else
0882                   cot_K=1._dl/ta
0883                endif
0884                wkb=sin(arg)*cot_
0885                wkb=wkb/sqrt(beta
0886             else
0887                wkb=(sin(arg)/arg
0888             endif
0889             end if
0890             phi_langer=symm*wkb
0891             return
0892          endif
0893       !
0894       ! Closed form solution for K=
0895       !
0896          if((K==1).and.(ibeta ==
0897             wkb=(sin_K**ell)* &
0898                  sqrt(sqrt(2._dl
0899             wkb=wkb*(1+0.1875d0/
0900             phi_langer=symm*wkb
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 852

```
0852                  wkb=achi*sqrt
0853                else
0854                   wkb=(sin(arg)
0855                endif
0856             else if((abs(arg)<1.d
0857                wkb=arg/3._dl
0858             else
0859                if(K/=0) then
0860                   if(K==1) then
0861                      cot_K=1._
0862                   else
0863                      cot_K=1._
0864                   endif
0865                   wkb=sin(arg)*
0866                   wkb=wkb/sqrt(
0867                else
0868                   wkb=(sin(arg)
0869                endif
0870             end if
0871             phi_langer=symm*wkb
0872             return
0873          endif
0874          !
0875          ! Closed form solution fo
0876          !
0877          if((K==1).and.(ibeta == (
0878             wkb=(sin_K**ell)* &
0879                  sqrt(sqrt(2._dl*P
0880             wkb=wkb*(1+0.1875d0/e
0881             phi_langer=symm*wkb
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 901

```
0901        return
0902     endif
0903
0904  ! Very close to 0, return 0 (
0905  !
0906     if(abs(achi)<1.d-8) the
0907        phi_langer=0._dl
0908        return
0909     endif
0910
0911
0912  ! For closed case, find corre
0913  !
0914     if(K==1) then
0915        anu=dble(ibeta)-1._d
0916     else
0917        anu=beta
0918     endif
0919  !
0920  ! Evaluate epsilon using asym
0921  !
0922     if(l<20) then
0923        epsilon=1._dl/sqrt(e
0924     else
0925        epsilon=1._dl/ell-0.
0926     endif
0927
0928     alpha=epsilon*anu
0929  !
0930  ! Calculate the turning point
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 882

```
0882        return
0883     endif
0884
0885     ! Very close to 0, return
0886     !
0887     if(abs(achi)<1.d-8) then
0888        phi_langer=0._dl
0889        return
0890     endif
0891
0892
0893     ! For closed case, find c
0894     !
0895     if(K==1) then
0896        anu=dble(ibeta)-1._dl
0897     else
0898        anu=beta
0899     endif
0900     !
0901     ! Evaluate epsilon using
0902     !
0903     if(l<20) then
0904        epsilon=1._dl/sqrt(el
0905     else
0906        epsilon=1._dl/ell-0.5
0907     endif
0908
0909     alpha=epsilon*anu
0910     !
0911     ! Calculate the turning p
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 931

```
0931    ! Function in question has on
0932    !
0933         if(K==-1) chi0=log((1._
0934         if(K==0) chi0=1._dl/alp
0935         if(K==1) chi0=asin(1._d
0936
0937
0938    ! Very close to chi0, use usu
0939    !
0940         if(abs(achi-chi0)<1.d-5
0941
0942    ! Calculate coefficients of l
0943    ! in the neighborhood of the
0944    ! Q(chi)=a*(chi0-chi)+b*(chi0
0945            alpha2=alpha*alpha
0946
0947            if(K==-1) then
0948               a=2._dl*alpha2*sq
0949               b=3._dl*alpha2**2
0950            endif
0951            if(K==0) then
0952               a=2._dl*alpha2*al
0953               b=3._dl*alpha2**2
0954            endif
0955            if(K==1) then
0956               a=2._dl*alpha2*sq
0957               b=3._dl*alpha2**2
0958            endif
0959
0960    ! Dependent variable x for wh
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 912

```
0912         ! Function in question ha
0913         !
0914         if(K==-1) chi0=log((1._dl
0915         if(K==0) chi0=1._dl/alpha
0916         if(K==1) chi0=asin(1._dl/
0917
0918
0919         ! Very close to chi0, use
0920         !
0921         if(abs(achi-chi0)<1.d-5)
0922
0923            ! Calculate coefficie
0924            ! in the neighborhood
0925            ! Q(chi)=a*(chi0-chi)
0926            alpha2=alpha*alpha
0927
0928            if(K==-1) then
0929               a=2._dl*alpha2*sq
0930               b=3._dl*alpha2**2
0931            endif
0932            if(K==0) then
0933               a=2._dl*alpha2*al
0934               b=3._dl*alpha2**2
0935            endif
0936            if(K==1) then
0937               a=2._dl*alpha2*sq
0938               b=3._dl*alpha2**2
0939            endif
0940
0941         ! Dependent variable
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 961

```
0961      ! x>0 is the evanescent regio
0962      !
0963            x=chi0-achi
0964      !
0965      ! Argument of Airy function
0966      !
0967            arg=(x+b*x*x/(5._dl*
0968      !
0969      ! Evaluate Airy function
0970      !
0971            wkb=airy_ai(arg)
0972      !
0973      ! Rest of functional dependen
0974      !
0975            wkb=wkb*(1._dl-b*x/(
0976      !  Normalization factor:
0977
0978            wkb=wkb*symm*ROOTPI*
0979            phi_langer=wkb
0980
0981            return
0982         endif
0983
0984
0985      ! Langer approximation.
0986      !
0987      ! Transport factor:
0988      !
0989         tmp=sqrt(abs(1._dl/(sin
0990
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 942

```
0942            ! x>0 is the evanesce
0943            !
0944            x=chi0-achi
0945            !
0946            ! Argument of Airy fu
0947            !
0948            arg=(x+b*x*x/(5._dl*a
0949            !
0950            ! Evaluate Airy funct
0951            !
0952            wkb=airy_ai(arg)
0953            !
0954            ! Rest of functional
0955            !
0956            wkb=wkb*(1._dl-b*x/(5
0957            !  Normalization fact
0958
0959            wkb=wkb*symm*ROOTPI*(
0960            phi_langer=wkb
0961
0962            return
0963         endif
0964
0965
0966         ! Langer approximation.
0967         !
0968         ! Transport factor:
0969         !
0970         tmp=sqrt(abs(1._dl/(sin_K
0971
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 991

```
0991     ! Eikonal factor
0992     !
0993          eikonal=qintegral(sin_K
0994
0995
0996
0997          arg=(1.5d0*eikonal/epsi
0998
0999          arg2=arg*arg
1000          if(achi>chi0) arg2=-arg
1001
1002     ! Multiply factors together
1003
1004          wkb=airy_ai(arg2)*symm*
1005          phi_langer=wkb
1006
1007          end function phi_langer
1008
1009     !ccccccccccccccccccccccccccc
1010     !
1011     ! Evaluates the exact solutio
1012     ! eikonal solution,   \int^x
1013     !
1014     ! In the open case, this inte
1015     ! and a log; its evaluation w
1016     ! of the Phi routine. An anal
1017     ! because the dependence on a
1018     ! region of the integrand con
1019     ! can only save the computati
1020     !
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 972

```
0972          ! Eikonal factor
0973          !
0974          eikonal=qintegral(sin_K,a
0975
0976
0977
0978          arg=(1.5d0*eikonal/epsilo
0979
0980          arg2=arg*arg
0981          if(achi>chi0) arg2=-arg2
0982
0983          ! Multiply factors togeth
0984
0985          wkb=airy_ai(arg2)*symm*RO
0986          phi_langer=wkb
0987
0988          end function phi_langer
0989
0990          !ccccccccccccccccccccccc
0991          !
0992          ! Evaluates the exact sol
0993          ! eikonal solution,   \in
0994          !
0995          ! In the open case, this
0996          ! and a log; its evaluati
0997          ! of the Phi routine. An
0998          ! because the dependence
0999          ! region of the integrand
1000          ! can only save the compu
1001          !
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 1021

```
1021    ! The integrals are very blan
1022    ! be precomputed and cached t
1023    ! on a relatively small numbe
1024    !
1025    ! Note that for the closed ca
1026    ! and alpha; for arg > alpha,
1027    ! integral to this case.
1028    !
1029    !ccccccccccccccccccccccccccc
1030          function qintegral(sin_
1031          implicit none
1032          real(dl) qintegral, sin
1033          integer K
1034          real(dl) alpha,exact,ar
1035
1036          real(dl), parameter ::
1037                                    PI
1038
1039          arg=alpha*sin_K
1040
1041          if(K==0) then
1042             if(arg>1._dl) then
1043                exact=sqrt(arg*ar
1044                qintegral=exact
1045                return
1046             else
1047                root1=sqrt(1._dl-
1048                exact=log((1._dl+
1049                qintegral=exact
1050                return
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1002

```
1002          ! The integrals are very
1003          ! be precomputed and cach
1004          ! on a relatively small n
1005          !
1006          ! Note that for the close
1007          ! and alpha; for arg > al
1008          ! integral to this case.
1009          !
1010          !ccccccccccccccccccccccc
1011          function qintegral(sin_K,
1012          implicit none
1013          real(dl) qintegral, sin_K
1014          integer K
1015          real(dl) alpha,exact,arg,
1016
1017          real(dl), parameter :: PI
1018              PIOVER2=1.570796327d0
1019
1020          arg=alpha*sin_K
1021
1022          if(K==0) then
1023              if(arg>1._dl) then
1024                  exact=sqrt(arg*ar
1025                  qintegral=exact
1026                  return
1027              else
1028                  root1=sqrt(1._dl-
1029                  exact=log((1._dl+
1030                  qintegral=exact
1031                  return
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 1051

```
1051        endif
1052     else if(K==-1) then
1053        if(arg>1._dl) then
1054           root1=sqrt(arg*ar
1055           root2=sqrt(arg*ar
1056           exact=alpha/2._dl
1057                 2._dl*root1
1058                 (alpha*root
1059           qintegral=exact
1060           return
1061        else
1062           root1=sqrt((1._dl
1063           exact=alpha/2._dl
1064                 1._dl))+0.5
1065                 arg*arg*(1.
1066                 alpha*alpha
1067           if(2._dl*arg*arg+
1068              exact=exact-al
1069           endif
1070           qintegral=exact
1071           return
1072        endif
1073     else
1074        if(arg>1._dl) then
1075           root1=sqrt(arg*ar
1076           root2=sqrt(alpha*
1077           exact=alpha/2._dl
1078              (2._dl*arg*arg
1079              atan(-root2/(r
1080           if(2._dl*arg*arg-
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1032

```
1032        endif
1033     else if(K==-1) then
1034        if(arg>1._dl) then
1035           root1=sqrt(arg*ar
1036           root2=sqrt(arg*ar
1037           exact=alpha/2._dl
1038                2._dl*root1*r
1039                (alpha*root1)
1040           qintegral=exact
1041           return
1042        else
1043           root1=sqrt((1._dl
1044           exact=alpha/2._dl
1045                1._dl))+0.5d0
1046                arg*arg*(1._d
1047                alpha*alpha))
1048           if(2._dl*arg*arg+
1049              exact=exact-a
1050           endif
1051           qintegral=exact
1052           return
1053        endif
1054     else
1055        if(arg>1._dl) then
1056           root1=sqrt(arg*ar
1057           root2=sqrt(alpha*
1058           exact=alpha/2._dl
1059              (2._dl*arg*ar
1060              atan(-root2/(
1061           if(2._dl*arg*arg-
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 1081

```
1081                    exact=exact+al
1082                 endif
1083              else
1084                 root1=sqrt((1._dl
1085                 dummyarg=alpha*(1
1086                 exact=0.5d0*log((
1087                  alpha/2._dl*log(
1088                  (alpha*alpha-1._
1089              endif
1090              qintegral=exact
1091              return
1092           endif
1093
1094           end function qintegral
1095
1096        !ccccccccccccccccccccccccccc
1097        !
1098        !       Airy function
1099        !
1100        ! Modified from original rout
1101        ! as part of the Cephes libra
1102        ! Modifications: eliminates c
1103        ! and translation to Fortran
1104        !
1105        !ccccccccccccccccccccccccccc
1106        !
1107        ! DESCRIPTION:
1108        !
1109        ! Solution of the differentia
1110        !
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1062

```
1062                    exact=exact+a
1063                 endif
1064              else
1065                 root1=sqrt((1._dl
1066                 dummyarg=alpha*(1
1067                 exact=0.5d0*log((
1068                     alpha/2._dl*l
1069                     (alpha*alpha-
1070              endif
1071              qintegral=exact
1072              return
1073           endif
1074
1075           end function qintegral
1076
1077        !ccccccccccccccccccccccccccc
1078        !
1079        !       Airy function
1080        !
1081        ! Modified from original
1082        ! as part of the Cephes l
1083        ! Modifications: eliminat
1084        ! and translation to Fort
1085        !
1086        !ccccccccccccccccccccccccccc
1087        !
1088        ! DESCRIPTION:
1089        !
1090        ! Solution of the differe
1091        !
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 1111

```
1111    !       y"(x) = xy.
1112    !
1113    ! The function returns the tw
1114    ! and their first derivatives
1115    !
1116    ! Evaluation is by power seri
1117    ! by rational minimax approxi
1118    !
1119    !
1120    !
1121    ! ACCURACY:
1122    ! Error criterion is absolute
1123    ! when function > 1, except *
1124    ! For large negative x, the a
1125    ! For large positive x, the r
1126    !
1127    ! Arithmetic  domain   functi
1128    ! IEEE        -10, 0     Ai
1129    ! IEEE          0, 10    Ai
1130    ! IEEE        -10, 0     Ai'
1131    ! IEEE          0, 10    Ai'
1132    ! IEEE        -10, 10    Bi
1133    ! IEEE        -10, 10    Bi'
1134    ! DEC         -10, 0     Ai
1135    ! DEC           0, 10    Ai
1136    ! DEC         -10, 0     Ai'
1137    ! DEC           0, 10    Ai'
1138    ! DEC         -10, 10    Bi
1139    ! DEC         -10, 10    Bi'
1140    !
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1092

```
1092    !       y"(x) = xy.
1093    !
1094    ! The function returns th
1095    ! and their first derivat
1096    !
1097    ! Evaluation is by power
1098    ! by rational minimax app
1099    !
1100    !
1101    !
1102    ! ACCURACY:
1103    ! Error criterion is abso
1104    ! when function > 1, exce
1105    ! For large negative x, t
1106    ! For large positive x, t
1107    !
1108    ! Arithmetic  domain   fu
1109    ! IEEE        -10, 0
1110    ! IEEE          0, 10
1111    ! IEEE        -10, 0
1112    ! IEEE          0, 10
1113    ! IEEE        -10, 10
1114    ! IEEE        -10, 10
1115    ! DEC         -10, 0
1116    ! DEC           0, 10
1117    ! DEC         -10, 0
1118    ! DEC           0, 10
1119    ! DEC         -10, 10
1120    ! DEC         -10, 10
1121    !
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 1141

```
1141    !
1142    ! Cephes Math Library Release
1143    ! Copyright 1984, 1987, 1989
1144    ! Direct inquiries to 30 Fros
1145    !
1146
1147        function airy_ai(x)
1148        implicit none
1149        real(dl) airy_ai
1150        real(dl) x,z, zz, t, f,
1151        real(dl) ak
1152        real(dl) AN(8),AD(8),AF
1153        real(dl), parameter ::
1154        real(dl), parameter ::
1155        real(dl), parameter ::
1156
1157
1158        AN(1)=3.465381015256290
1159        AN(2)=1.200759527396458
1160        AN(3)=7.627960536152345
1161        AN(4)=1.680892249346305
1162        AN(5)=1.597563913501644
1163        AN(6)=7.053609068404441
1164        AN(7)=1.402646911633896
1165        AN(8)=9.999999999999999
1166
1167        AD(1)=5.675945326387702
1168        AD(2)=1.475625625848472
1169        AD(3)=8.451389701414746
1170        AD(4)=1.773180881454004
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1122

```
1122    !
1123    ! Cephes Math Library Rel
1124    ! Copyright 1984, 1987, 1
1125    ! Direct inquiries to 30
1126    !
1127
1128        function airy_ai(x)
1129        implicit none
1130        real(dl) airy_ai
1131        real(dl) x,z, zz, t, f, g
1132        real(dl) ak
1133        real(dl) AN(8),AD(8),AFN(
1134        real(dl), parameter :: AM
1135        real(dl), parameter :: c1
1136        real(dl), parameter :: sq
1137
1138
1139        AN(1)=3.4653810152562903
1140        AN(2)=1.2007595273964580
1141        AN(3)=7.6279605361523451
1142        AN(4)=1.6808922493463057
1143        AN(5)=1.5975639135016441
1144        AN(6)=7.0536090684044418
1145        AN(7)=1.4026469116338966
1146        AN(8)=9.9999999999999954
1147
1148        AD(1)=5.6759453263877021
1149        AD(2)=1.4756256258484720
1150        AD(3)=8.4513897014147463
1151        AD(4)=1.7731808814540046
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 1171

```
1171      AD(5)=1.642346928715297
1172      AD(6)=7.147784008255756
1173      AD(7)=1.409591356078340
1174      AD(8)=1.000000000000000
1175
1176      AFN(1)=-1.3169632341833
1177      AFN(2)=-6.2645654443191
1178      AFN(3)=-6.9315803603693
1179      AFN(4)=-2.7977998154511
1180      AFN(5)=-4.9190013260950
1181      AFN(6)=-4.0626592359488
1182      AFN(7)=-1.5927649623926
1183      AFN(8)=-2.7764910815523
1184      AFN(9)=-1.6778769848911
1185
1186      AFD(1)=1.33560420706553
1187      AFD(2)=3.26825032795224
1188      AFD(3)=2.673670409414 99
1189      AFD(4)=9.187074029072 59
1190      AFD(5)=1.475291467716 66
1191      AFD(6)=1.156871737951 88
1192      AFD(7)=4.402916416152 11
1193      AFD(8)=7.547203482874 14
1194      AFD(9)=4.518500929705 80
1195
1196      AGN(1)=1.97339932091685
1197      AGN(2)=3.91103029615688
1198      AGN(3)=1.06579897599595
1199      AGN(4)=9.39169229816650
1200      AGN(5)=3.51465656105547
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1152

```
1152      AD(5)=1.6423469287152 9701
1153      AD(6)=7.1477840082557 5695
1154      AD(7)=1.4095913560783 4029
1155      AD(8)=1.0000000000000 0000
1156
1157      AFN(1)=-1.31696323418331 7
1158      AFN(2)=-6.26456544431912 3
1159      AFN(3)=-6.93158036036933 5
1160      AFN(4)=-2.79779981545119 1
1161      AFN(5)=-4.91900132609500 3
1162      AFN(6)=-4.06265923594885 4
1163      AFN(7)=-1.59276496239262 0
1164      AFN(8)=-2.77649108155232 9
1165      AFN(9)=-1.67787698489114 6
1166
1167      AFD(1)=1.33560420706553 24
1168      AFD(2)=3.26825032795224 61
1169      AFD(3)=2.67367040941499 55
1170      AFD(4)=9.18707402907259 62
1171      AFD(5)=1.47529146771666 41
1172      AFD(6)=1.15687173795188 04
1173      AFD(7)=4.40291641615211 20
1174      AFD(8)=7.54720348287414 29
1175      AFD(9)=4.51850092970580 37
1176
1177      AGN(1)=1.97339932091685 67
1178      AGN(2)=3.91103029615688 27
1179      AGN(3)=1.06579897599595 59
1180      AGN(4)=9.39169229816650 23
1181      AGN(5)=3.51465656105547 61
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 1201

```
1201        AGN(6)=6.33888919628925
1202        AGN(7)=5.85804113048388
1203        AGN(8)=2.82851600836737
1204        AGN(9)=6.98793669997260
1205        AGN(10)=8.1178923955438
1206        AGN(11)=3.4155178476592
1207
1208        AGD(1)=9.308929080774 41
1209        AGD(2)=1.983529287183 12
1210        AGD(3)=1.556466289328 64
1211        AGD(4)=5.476860694229 75
1212        AGD(5)=9.542936116189 61
1213        AGD(6)=8.645808263523 92
1214        AGD(7)=4.126565238242 22
1215        AGD(8)=1.012590851165 09
1216        AGD(9)=1.171667332144 13
1217        AGD(10)=4.91834570062 93
1218      !
1219      ! Exponentially tiny for larg
1220      !
1221        if(x>AMAXAIRY) then
1222           airy_ai=0._dl
1223           return
1224        endif
1225      !
1226      ! Pade fit for large negative
1227      !
1228        if(x<-2.09d0) then
1229           t=sqrt(-x)
1230           zeta=-2._dl*x*t/3._d
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1182

```
1182        AGN(6)=6.338891962892549
1183        AGN(7)=5.858041304838845
1184        AGN(8)=2.828516008367 3701
1185        AGN(9)=6.987936699972 6096
1186        AGN(10)=8.11789239554 3892
1187        AGN(11)=3.41551784765 9236
1188
1189        AGD(1)=9.308929080774 4197
1190        AGD(2)=1.983529287183 1214
1191        AGD(3)=1.556466289328 6461
1192        AGD(4)=5.476860694229 7549
1193        AGD(5)=9.542936116189 6188
1194        AGD(6)=8.645808263523 9219
1195        AGD(7)=4.126565238242 2260
1196        AGD(8)=1.012590851165 0913
1197        AGD(9)=1.171667332144 1352
1198        AGD(10)=4.91834570062 9300
1199        !
1200        ! Exponentially tiny for
1201        !
1202        if(x>AMAXAIRY) then
1203           airy_ai=0._dl
1204           return
1205        endif
1206        !
1207        ! Pade fit for large nega
1208        !
1209        if(x<-2.09d0) then
1210           t=sqrt(-x)
1211           zeta=-2._dl*x*t/3._dl
```

/Users/lp1opa/Compare/camb_simdata/besselsf90, Top line: 1231

```
1231            t=sqrt(t)
1232            ak=sqpii/t
1233            z=1._dl/zeta
1234            zz=z*z
1235            uf=1._dl+zz*polevl(z
1236            ug=z*polevl(zz,AGN,1
1237            theta=zeta+0.25d0*PI
1238            f=sin(theta)
1239            g=cos(theta)
1240            airy_ai=ak*(f*uf-g*u
1241            return
1242         endif
1243      !
1244      ! Pade fit for large positive
1245      !
1246         if(x>=2.09) then
1247            t=sqrt(x)
1248            zeta=2._dl*x*t/3._dl
1249            g=exp(zeta)
1250            t=sqrt(t)
1251            ak=2._dl*t*g
1252            z=1._dl/zeta
1253            f=polevl(z,AN,7)/pol
1254            airy_ai=sqpii*f/ak
1255            return
1256         endif
1257      !
1258      ! Taylor series for region ar
1259      !
1260
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1212

```
1212            t=sqrt(t)
1213            ak=sqpii/t
1214            z=1._dl/zeta
1215            zz=z*z
1216            uf=1._dl+zz*polevl(zz
1217            ug=z*polevl(zz,AGN,10
1218            theta=zeta+0.25d0*PI
1219            f=sin(theta)
1220            g=cos(theta)
1221            airy_ai=ak*(f*uf-g*ug
1222            return
1223         endif
1224         !
1225         ! Pade fit for large posi
1226         !
1227         if(x>=2.09) then
1228            t=sqrt(x)
1229            zeta=2._dl*x*t/3._dl
1230            g=exp(zeta)
1231            t=sqrt(t)
1232            ak=2._dl*t*g
1233            z=1._dl/zeta
1234            f=polevl(z,AN,7)/pole
1235            airy_ai=sqpii*f/ak
1236            return
1237         endif
1238         !
1239         ! Taylor series for regio
1240         !
1241
```

/Users/lp1opa/Compare/camb_simdata/besselsf90, Top line: 1261

```
1261        f=1._dl
1262        g=x
1263        t=1._dl
1264        uf=1._dl
1265        ug=x
1266        ak=1._dl
1267        z=x*x*x
1268       do while (t>ACC)
1269           uf=uf*z
1270           ak=ak+1._dl
1271           uf=uf/ak
1272           ug=ug*z
1273           ak=ak+1._dl
1274           ug=ug/ak
1275           uf=uf/ak
1276           f=f+uf
1277           ak=ak+1._dl
1278           ug=ug/ak
1279           g=g+ug
1280           t=abs(uf/f)
1281        end do
1282
1283
1284        airy_ai=c1*f-c2*g
1285
1286        end function airy_ai
1287
1288    !cccccccccccccccccccccccccc
1289    !  Evaluate polynomial
1290    !cccccccccccccccccccccccccc
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1242

```
1242        f=1._dl
1243        g=x
1244        t=1._dl
1245        uf=1._dl
1246        ug=x
1247        ak=1._dl
1248        z=x*x*x
1249        do while (t>ACC)
1250            uf=uf*z
1251            ak=ak+1._dl
1252            uf=uf/ak
1253            ug=ug*z
1254            ak=ak+1._dl
1255            ug=ug/ak
1256            uf=uf/ak
1257            f=f+uf
1258            ak=ak+1._dl
1259            ug=ug/ak
1260            g=g+ug
1261            t=abs(uf/f)
1262        end do
1263
1264
1265        airy_ai=c1*f-c2*g
1266
1267        end function airy_ai
1268
1269    !cccccccccccccccccccccccccc
1270    !  Evaluate polynomial
1271    !cccccccccccccccccccccccccc
```

/Users/lp1opa/Compare/camb_simdata/besselS.f90, Top line: 1291

```
1291     !  DESCRIPTION:
1292     !
1293     !  Evaluates polynomial of deg
1294     !
1295     !                          2
1296     !  y  =  C  + C x + C x  +...+
1297     !         0    1     2
1298     !
1299     !  Coefficients are stored in
1300     !
1301     !  coef(1) = C  , ..., coef(N+
1302     !             N
1303     !
1304     !  The function p1evl() assume
1305     !
1306     !  omitted from the array.  It
1307     !  otherwise the same as polev
1308     !
1309     !
1310     !  SPEED:
1311     !
1312     !  In the interest of speed, t
1313     !  of bounds arithmetic.  This
1314     !  the functions in the librar
1315     !  equipment features, the use
1316     !  program in microcode or ass
1317     !
1318     !  Cephes Math Library Release
1319     !  Copyright 1984, 1987, 1988
1320     !  Direct inquiries to 30 Fros
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1272

```
1272     !  DESCRIPTION:
1273     !
1274     !  Evaluates polynomial of
1275     !
1276     !                          2
1277     !  y  =  C  + C x + C x  +
1278     !         0    1     2
1279     !
1280     !  Coefficients are stored
1281     !
1282     !  coef(1) = C  , ..., coe
1283     !             N
1284     !
1285     !  The function p1evl() as
1286     !
1287     !  omitted from the array.
1288     !  otherwise the same as p
1289     !
1290     !
1291     !  SPEED:
1292     !
1293     !  In the interest of spee
1294     !  of bounds arithmetic.
1295     !  the functions in the li
1296     !  equipment features, the
1297     !  program in microcode or
1298     !
1299     !  Cephes Math Library Rel
1300     !  Copyright 1984, 1987, 1
1301     !  Direct inquiries to 30
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 1321

```
1321      !
1322            function polevl(x,coef,
1323            implicit none
1324            real(dl) polevl
1325            real(dl) x,ans
1326            real(dl) coef
1327            integer N,i
1328
1329            dimension coef(N+1)
1330
1331            ans=coef(1)
1332            do i=2,N+1
1333               ans=ans*x+coef(i)
1334            end do
1335            polevl=ans
1336
1337            end function polevl
1338
1339      !
1340      !
1341      ! Evaluate polynomial when co
1342      ! Otherwise same as polevl.
1343      !
1344            function p1evl(x,coef,N
1345            implicit none
1346            real(dl) p1evl
1347            real(dl) x,coef,ans
1348            integer N,i
1349            dimension coef(N)
1350
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1302

```
1302          !
1303            function polevl(x,coef,N)
1304            implicit none
1305            real(dl) polevl
1306            real(dl) x,ans
1307            real(dl) coef
1308            integer N,i
1309
1310            dimension coef(N+1)
1311
1312            ans=coef(1)
1313            do i=2,N+1
1314                ans=ans*x+coef(i)
1315            end do
1316            polevl=ans
1317
1318            end function polevl
1319
1320          !
1321          !
1322          ! Evaluate polynomial whe
1323          ! Otherwise same as polev
1324          !
1325            function p1evl(x,coef,N)
1326            implicit none
1327            real(dl) p1evl
1328            real(dl) x,coef,ans
1329            integer N,i
1330            dimension coef(N)
1331
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 1351

```
1351        ans=x+coef(1)
1352        do i=2,N
1353           ans=ans*x+coef(i)
1354        end do
1355        p1evl=ans
1356
1357        end function p1evl
1358
1359
1360
1361     end module SpherBessels !US
1362
1363
1364
1365        SUBROUTINE BJL_EXTERN
1366        use SpherBessels
1367        use Precision
1368        !!== MODIFIED SUBROUT
1369        !!== CORRECTED THE SM
1370        !!== CORRECTED THE SI
1371        !!== WORKS FASTER AND
1372        !!== zqhuang@astro.ut
1373        IMPLICIT NONE
1374        INTEGER L
1375        real(dl) X,JL
1376
1377        call BJL(L,X,JL)
1378
1379        END SUBROUTINE BJL_EX
1380
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1332

```
1332        ans=x+coef(1)
1333        do i=2,N
1334           ans=ans*x+coef(i)
1335        end do
1336        p1evl=ans
1337
1338        end function p1evl
1339
1340
1341
1342     end module SpherBessels !
1343
1344
1345
1346        SUBROUTINE BJL_EXTERNAL(L
1347        use SpherBessels
1348        use Precision
1349        !!== MODIFIED SUBROUTINE
1350        !!== CORRECTED THE SMALL
1351        !!== CORRECTED THE SIGN O
1352        !!== WORKS FASTER AND MOR
1353        !!== zqhuang@astro.utoron
1354        IMPLICIT NONE
1355        INTEGER L
1356        real(dl) X,JL
1357
1358        call BJL(L,X,JL)
1359
1360        END SUBROUTINE BJL_EXTERN
1361
```

/Users/lp1opa/Compare/camb_simdata/bessels.f90, Top line: 1381

```
1381  |
```

/Users/lp1opa/Compare/camb_des/bessels.f90, Top line: 1362

```
1362  |
```