Brightness Temperature for MOLPOP-CEP
Moshe, October 4–30, 2013

Definitions: For intensity I_ν, brightness temperature $T_{\rm br}$ is defined from $B_\nu(T_{\rm br}) = I_\nu$, where B is the Planck function. Carl Heiles states that "radio astronomers who observe mm-wave lines, for which the RJ approx is not valid, generally quote the antenna temperature as if the RJ approx were valid." That is, if RJ is the Rayleigh Jeans form of B and $\Delta T_{\rm RJ}$ is the temperature obtained from this definition then

$$RJ(T_{\rm RJ}) = B_\nu(T_{\rm br}) \quad \Rightarrow \quad T_{\rm RJ} = \frac{T_l}{e^{T_l/T_{\rm br}} - 1} \tag{1}$$

where $T_l = h\nu/k$. The limit $T_l \ll T_{\rm br}$ gives $T_{\rm RJ} = T_{\rm br}$, as it should.

Radio observers typically measure intensity I_ν in the direction of a cloud and subtract from it the empty sky measurement, which detects only the CMB. The result can be expressed in brightness temperature $\Delta T_{\rm br}$, defined from

$$B_\nu(\Delta T_{\rm br}) = I_\nu(\text{measured}) - B_\nu(T_{\rm CMB}) \tag{2}$$

The measured intensity includes the cloud emission I_ν, which is the quantity of interest, and the transmitted CMB, namely

$$I_\nu(\text{measured}) = I_\nu + B_\nu(T_{\rm CMB})e^{-\tau} \tag{3}$$

where τ is the cloud optical depth. Therefore

$$B(\Delta T_{\rm br}) = I - B(T_{\rm CMB})\left(1 - e^{-\tau}\right) \tag{4}$$

and the subscript ν can be removed because everything is done at line center. Assuming all quantities, including level populations, to be uniform throughout the source, the intensity emerging perpendicular to its face is

$$I = B(T_{\rm ex})\left(1 - e^{-\tau}\right) \tag{5}$$

so that

$$B(\Delta T_{\rm br}) = [B(T_{\rm ex}) - B(T_{\rm CMB})]\left(1 - e^{-\tau}\right) \tag{6}$$

In both cases, whether assuming uniform conditions or not, $\Delta T_{\rm br}$ obtained from either eq. 4 or 6 can be expressed in terms of an equivalent $\Delta T_{\rm RJ}$ using eq. 1.

MOLPOP Implementation: Modeling the source, an exact MOLPOP calculation (using CEP) will produce a prediction for I. So the model prediction for the observed $\Delta T_{\rm br}$ will be obtained by solving equation 4, where I is the model result for intensity. In the escape probability approximation everything is uniform and one can use equation 6 to get $\Delta T_{\rm br}$.

MOLPOP has a "Planck exponential" function `plexp = 1/[exp(x) - 1]`. Equation 6 is therefore

$$\mathtt{plexp}\left(\frac{T_l}{\Delta T_{\rm br}}\right) = \left[\mathtt{plexp}\left(\frac{T_l}{T_{\rm ex}}\right) - \mathtt{plexp}\left(\frac{T_l}{T_{\rm CMB}}\right)\right]\left(1 - e^{-\tau}\right) \tag{7}$$

which translates to the RJ-equivalent temperature

$$\Delta T_{\rm RJ} = T_l\left[\mathtt{plexp}\left(\frac{T_l}{T_{\rm ex}}\right) - \mathtt{plexp}\left(\frac{T_l}{T_{\rm CMB}}\right)\right]\left(1 - e^{-\tau}\right) \tag{8}$$

For a CEP calculation, all that is needed is to replace $\mathtt{plexp}(T_l/T_{\rm ex})$ in these two expressions with $I * c^2/2h\nu^3$.

I have already added to the escape probability branch of MOLPOP listings of $\Delta T_{\rm br}$, determined from equation 7, and $\Delta T_{\rm RJ}$, determined from equation 8. This is done with a new subroutine `Tbr4Tx(Tl,Tx,taul,Tbr,TRJ)` added in `maths_molpop.f90`. I have also coded a similar subroutine `Tbr4I(nu,I,taul,Tbr,TRJ)` that performs the same calculations from the intensity I. This should be used to tabulate $\Delta T_{\rm br}$ and $\Delta T_{\rm RJ}$ when calculating an exact solution using CEP. Below is the listing of the relevant section in the program. The functions `Tbr_Tx` and `Tbr_I` are still there for compatibility with what may have already been coded be in the CEP part. They should be removed in lieu of the subroutines.

In `maths_molpop.f90`

```
! ================  Stuff related to Planck function  ===================


      double precision function plexp(x)
!--------------------------------------------------------
!     calculates the Planck function, modulo 2h*nu^3/c^2
!     That is:   plexp = 1/[exp(x) - 1]
!--------------------------------------------------------
         implicit none
         double precision x

         if(x .eq. 0.0) then
           write(16,'(6x,a)') 'ERROR! Function plexp called with argument zero.'
           plexp = 1.d100
         else if(x .gt. 50.0) then
           plexp = 0.0
         else if(dabs(x) .lt. 0.001) then
```

```
          plexp = 1.0/x
        else
          plexp = 1.0/(dexp(x) - 1.0)
        end if
        return
      end function plexp


      double precision function Inv_plexp(P)
!-----------------------------------------------------------------
!     Finds the argument of the Planck function given its value P
!     That is, solves the equation P = 1/[exp(x) - 1]
!-----------------------------------------------------------------
        implicit none
        double precision, intent(in) :: P

        if (P > 1.e3) then  ! might as well use small x (RJ) limit
            inv_plexp = 1./P
        else
            inv_plexp = DLOG(1. + 1./P)
        end if
        return
      END function Inv_plexp


      Subroutine Tbr4Tx(Tl,Tx,taul,Tbr,TRJ)
!-----------------------------------------------------------------
!     For a line with temperature-equivalent frequency Tl
!     enter with excitation temperature Tx and optical depth taul
!     calculate brightness temperature from
!
!        B(Tbr) = [B(Tx) - B(Tcmb)]*[1 - exp(-taul)]
!
!     All intensitie are in photon occupation number because
!     we use plexp for B(T); so B(Tbr) is simply TRJ/Tl
!     where TRJ is the Rayleigh Jeans equivalent T
!-----------------------------------------------------------------
        implicit none
        double precision, intent(in)  :: Tl, Tx, taul
        double precision, intent(out) :: Tbr, TRJ
        double precision B
```

```
         integer sgn

         if (Tx == Tcmb) then
            TRJ = 0.
            Tbr = 0.
            return
         end if

         B = (plexp(Tl/Tx) - plexp(Tl/Tcmb)) * (1. - dexp(-taul))

!        negative B means Tx < Tcmb so we get absorption line; negative Tbr
         sgn = 1
         if (B < 0.d0) sgn = -1

         TRJ = Tl*B
         Tbr = sgn*Tl/Inv_plexp(dabs(B))
         return
      END Subroutine Tbr4Tx


      Subroutine Tbr4I(nu,I,taul,Tbr,TRJ)
!------------------------------------------------------------
!     For a line with frequency nu
!     enter with intensity I and optical depth taul
!     calculate brightness temperature from
!
!        B(Tbr) = I - B(Tcmb)*[1 - exp(-taul)]
!
!     All intensities are converted to photon occupation number
!     For B(T) we use plexp, I is converted with 2h*nu^3/c^2
!     Then the RJ tempearture is simply TRJ = Tl*B(Tbr)
!-------------------------------------------------------------
         implicit none
         double precision, intent(in)  :: nu, I, taul
         double precision, intent(out) :: Tbr, TRJ
         double precision B, Tl, Intensity
         integer sgn

         Tl = hPl*nu/Bk
         Intensity = I/(2*hPl*nu**3/cl**2)
```

```
         B = Intensity - plexp(Tl/Tcmb) * (1. - dexp(-taul))

         if (B == 0.d0) then
            TRJ = 0.
            Tbr = 0.
            return
         end if

!        negative B means we get absorption line; negative Tbr
         sgn = 1
         if (B < 0.d0) sgn = -1

         TRJ = Tl*B
         Tbr = sgn*Tl/Inv_plexp(dabs(B))
         return
      END Subroutine Tbr4I



      double precision function Tbr_Tx(Tl,Tx,taul)
!-----------------------------------------------------------------
!     For a line with temperature-equivalent frequency Tl
!     enter with excitation temperature Tx and optical depth taul
!     calculate brightness temperature from
!
!        B(Tbr) = [B(Tx) - B(Tcmb)]*[1 - exp(-taul)]
!
!     All intensitie are in photon occupation number because
!     we use plexp for B(T)
!-----------------------------------------------------------------
         implicit none
         double precision, intent(in) :: Tl, Tx, taul
         double precision B
         integer sgn

         if (Tx == Tcmb) then
            Tbr_Tx = 0.
            return
         end if

         B = (plexp(Tl/Tx) - plexp(Tl/Tcmb)) * (1. - dexp(-taul))
```

```
!        negative B means Tx < Tcmb so we get absorption line; negative Tbr
         sgn = 1
         if (B < 0.d0) sgn = -1

         Tbr_Tx = sgn*Tl/Inv_plexp(dabs(B))
         return
      END function Tbr_Tx


      double precision function Tbr_I(nu,I,taul)
!-------------------------------------------------------------
!     For a line with frequency nu
!     enter with intensity I and optical depth taul
!     calculate brightness temperature from
!
!        B(Tbr) = I - B(Tcmb)*[1 - exp(-taul)]
!
!     All intensities are converted to photon occupation number
!     For B(T) we use plexp, I is converted with 2h*nu^3/c^2
!--------------------------------------------------------------
         implicit none
         double precision, intent(in) :: nu, I, taul
         double precision B, Tl, Intensity
         integer sgn

         Tl = hPl*nu/Bk
         Intensity = I/(2*hPl*nu**3/cl**2)

         B = Intensity - plexp(Tl/Tcmb) * (1. - dexp(-taul))

         if (B == 0.d0) then
            Tbr_I = 0.
            return
         end if

!        negative B means we get absorption line; negative Tbr
         sgn = 1
         if (B < 0.d0) sgn = -1

         Tbr_I = sgn*Tl/Inv_plexp(dabs(B))
```

```
    return
  END function Tbr_I

!==========================================================================
```