1. Comet in the box. The aim of this programme is to compute image of coma for given dust distribution in arbitrary spectral range.

```
program cometbox
  parameter (n = 30)          // size (fineness) of the box (n × n × n)
  parameter (steps = 10)      // number of steps in integration of Planck function
  parameter (f_max = 100)     // maximal number of points in the filter function

  real box(n,n,n)
        // Dust distribution is stored in this box (numbers of particles per volume unit).
  real picture(n,n)     // Comet image is stored in this matrix (n × n pixels).

  ⟨ Variable definitions 2 ⟩
  print *, '*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*'
  print *, '*␣COMETBOX␣-␣program␣for␣integrating␣of␣*'
  print *, '*␣␣␣␣␣␣␣␣␣␣␣␣␣cometary␣dust␣radiation␣␣␣*'
  print *, '*␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣*'
  print *, '*␣1996␣Rosta␣Stork␣␣␣␣␣stork@asu.cas.cz␣*'
  print *, '*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*␣*'
  print *
  ⟨ Reading of temperatures 3 ⟩
  ⟨ Reading of parameters 6 ⟩
  ⟨ Erasing of result matrix 5 ⟩
  do size = 0.5, 3.0, 0.5      // for particle sizes from 0.5 μm to 3.0 μm
    print *
    print 55, size
 55: format ('␣Grain␣size␣=␣', F3.1 , '␣microns') ;
    ⟨ Temperature determination 17 ⟩      // Particles of different size differ in temperature.
    ⟨ Box input 9 ⟩
    print 65, T
 65: format ('␣␣␣␣␣␣␣T␣=␣', F5.1 , '␣K') ;
    do l = 1, n
      do r = 1, n      // Integration runs for n × n pixels.
        ⟨ Integration of dust quantity 11 ⟩
        ⟨ Computing of radiation intensity 13 ⟩
        picture(r, l) = picture(r, l) + intensity*column*size*size      // Pixel brightness
            depends on column dust density (number of particles) and effective area of particles.
      enddo
    enddo
  enddo
  ⟨ Output of result matrix 20 ⟩
end
```

2.

⟨ Variable definitions 2 ⟩ ≡

```
integer l, r, p      // box indexes (p means direction to observer)
real size      // size of dust particles
```

See also sections 4, 8, 10, 12, 14, 16, and 18.

This code is used in section 1.

3.

⟨ Reading of temperatures 3 ⟩ ≡

```
open (unit = 2, file = 'temperat.dat')
do i = 1, 6
   read (2, *) temperat(i)
enddo
close (unit = 2)
print *, 'Temperatures␣read␣(from␣file␣temperat.dat)'
print *
```

This code is used in section 1.

4.

⟨ Variable definitions 2 ⟩ +≡

```
real temperat(6)
```

5.

⟨ Erasing of result matrix 5 ⟩ ≡

```
do l = 1, n
   do r = 1, n
      picture(r, l) = 0.0
   enddo
enddo
```

This code is used in section 1.

6. Reading spectral range. In this section we read spectral range in one of following forms:

- border points of interval
- filter function — table of wavelengths (in μm) and corresponding transmittance

In second case program can read file of any size. But, unfortunately, in Fortran we must declare size of array in advance. So, if your table contains more points than *f_max*, please, change this parameter and recompile.

Also, the filename of result file is read in this section.

⟨ Reading of parameters 6 ⟩ ≡

```
print *, 'Wavelengths␣from,␣to␣(microns)'
print *, '␣or␣type␣0␣0␣for␣reading␣the␣filter␣function␣file'
filter_boolean = F
read *, lambda_start, lambda_finish
if ((lambda_start ≡ 0) ∧ (lambda_finish ≡ 0))  filter_boolean = T
lambda_start = lambda_start * 1 · 10^−6
lambda_finish = lambda_finish * 1 · 10^−6      // We compute in SI.
if (filter_boolean) then
   ⟨ Filter reading 7 ⟩
   lambda_start = filter(1, 1)
   lambda_finish = filter(f_last, 1)
        // We assume that transmittance = 0 out of given filter function.
endif
print *, 'Result␣filename:'
read (*, 20) result
20: format (A) ;
```

This code is used in section 1.

7.

⟨ Filter reading 7 ⟩ ≡

```
      print *, 'Filename␣of␣the␣filter␣function:'
      read (*,20) f_name
      open (unit = 3, file = f_name, status = 'old')
      i = 0
30:   continue
      read (3, *, end = 40) auxiliary
      i = i + 1
      goto 30
40:   continue
      f_last = i
      close (unit = 3)
      if (f_last > f_max) then
         print *, 'File␣', f_name, '␣contains␣', f_last, '␣points.'
         print *, 'Please,␣change␣parameter␣f_max␣and␣recompile.'
         stop
      endif
      open (unit = 3, file = f_name)
      do i = 1, f_last
         read (3,*) filter(i,1), filter(i,2)
         filter(i,1) = filter(i,1)*1 · 10^-6      // conversion to SI units
      enddo
      close (unit = 3)
      print *, 'Filter␣function␣read␣(from␣file␣', f_name, ')'
```

This code is used in section 6.

8.

⟨ Variable definitions 2 ⟩ +≡

```
      real lambda_start, lambda_finish
            // λ_start and λ_finish are limits of given spectral range in μm.
      character*12 result, f_name
      logical filter_boolean
      integer f_last      // index of the last point in the filter function
      real auxiliary      // auxiliary variable
      real filter(f_max, 2)      // array for storing filter function, filter(*,1) are wavelengths and
            filter(*,2) are corresponding transmittance
```

9. Box input. This section reads dust distribution of particle size *size* from relevant file into array *box*.

⟨ Box input 9 ⟩ ≡

```
code = size*10
if (code ≡ 5)  extension = '05'
if (code ≡ 10)  extension = '10'
if (code ≡ 15)  extension = '15'
if (code ≡ 20)  extension = '20'
if (code ≡ 25)  extension = '25'
if (code ≡ 30)  extension = '30'
open (unit = 1, file = 'dust.' || extension)
do p = 1, n
  do l = 1, n
    read (1, *) (box(r, l, p), r = 1, n)
  enddo
enddo
close (unit = 1)
print *, '␣␣␣File␣dust.' || extension, '␣read'
```

This code is used in section 1.

10.

⟨ Variable definitions 2 ⟩ +≡

```
integer code
character extension*2
```

11. Integration of dust quantity. This section integrates dust quantity along line of view. We assume optical thin environment. That is why we can integrate dust quantity in first step and compute brightness according to the Planck law in second step. In more general case (non–thin environment) above mentioned phases cannot be detached.

⟨ Integration of dust quantity 11 ⟩ ≡

```
column = 0.0
do p = 1, n        // adding along line of view
  column = column + box(r, l, p)
enddo
```

This code is used in section 1.

12.

⟨ Variable definitions 2 ⟩ +≡

```
real column        // column density of dust
```

13. Computing intensity of radiation. The intensity of radiation is computed by integration of Planck law from λ_{start} do λ_{finish}. Trapezium method with *steps* steps is used. We understand radiation of unit quantity of dust.

In the case filter function is given we compute transmittance in every integration step and multiply intensity by it.

⟨ Computing of radiation intensity 13 ⟩ ≡

```
step = (lambda_finish − lambda_start)/steps    // setting of integration step
intensity = 0.0
lambda_a = lambda_start
lambda_b = lambda_start + step
do i = 1, steps
  intensity_step = step∗(planck(lambda_a, T) + planck(lambda_b, T))/2    // Area of
      trapezium defined by λa and λb is computed by formula (planck(λa,T)+planck(λb,T))/2 × step.
  if (filter_boolean) then
    lambda_c = (lambda_a + lambda_b)/2
    ⟨ Transmittance determination 15 ⟩
    intensity_step = intensity_step∗transmittance
  endif
  intensity = intensity + intensity_step
  lambda_a = lambda_b
  lambda_b = lambda_b + step
enddo
```

This code is used in section 1.

14.

⟨ Variable definitions 2 ⟩ +≡

```
real step    // size of integration step
real lambda_a, lambda_b    // limits of integrated trapezium
real lambda_c    // center point of integrated interval
integer i    // loop variable (step number)
real intensity    // radiation intensity of unit quantity of dust
real intensity_step    // radiation intensity computed in current step
real transmittance    // transmittance in current integration step
```

15. Transmittance determination. Transmittance in wavelength $\lambda_c = \frac{\lambda_a+\lambda_b}{2}$ is computed by linear interpolation from nearest points $filter(s-1,1)$, $filter(s,1)$ in filter function ($filter(s-1,1) < \lambda_c \leq filter(s,1)$). First we find points $filter(s-1,1)$, $filter(s,1)$ and then compute transmittance by formula:

$$transmittance(\lambda_c) = filter(s-1,2) + div(lambda_c - filter(s-1,1))$$

where

$$div = \frac{filter(s,2) - filter(s-1,2)}{filter(s,1) - filter(s-1,1)}$$

is derivation of filter function in λ_c.

⟨ Transmittance determination 15 ⟩ ≡

```
      s = 1
  80: if (lambda_c ≤ filter(s,1))  goto 90
      s = s + 1
      goto 80
  90: continue
      if (lambda_c ≡ filter(s,1)) then
         transmittance = filter(s,2)
      else
         div = (filter(s,2) − filter(s − 1,2))/(filter(s,1) − filter(s − 1,1))
         transmittance = filter(s − 1,2) + div∗(lambda_c − filter(s − 1,1))
      endif
```

This code is used in section 13.

16.

⟨ Variable definitions 2 ⟩ +≡

```
      integer s      // index variable
      real div       // derivation of filter function
```

17. Determination of temperature. It's necessary to determine temperatures of particles of various sizes. Temperatures are stored in file `temperat.dat`, where they are read from.

$\langle$ Temperature determination 17 $\rangle \equiv$

$$
\begin{aligned}
&\textbf{if } (\boldsymbol{abs}(size - 0.5) < 1 \cdot 10^{-4}) \quad T = temperat(1)\\
&\textbf{if } (\boldsymbol{abs}(size - 1.0) < 1 \cdot 10^{-4}) \quad T = temperat(2)\\
&\textbf{if } (\boldsymbol{abs}(size - 1.5) < 1 \cdot 10^{-4}) \quad T = temperat(3)\\
&\textbf{if } (\boldsymbol{abs}(size - 2.0) < 1 \cdot 10^{-4}) \quad T = temperat(4)\\
&\textbf{if } (\boldsymbol{abs}(size - 2.5) < 1 \cdot 10^{-4}) \quad T = temperat(5)\\
&\textbf{if } (\boldsymbol{abs}(size - 3.0) < 1 \cdot 10^{-4}) \quad T = temperat(6)
\end{aligned}
$$

This code is used in section 1.

18.

$\langle$ Variable definitions 2 $\rangle +\equiv$

real T // Temperature in Kelvins

19. Planck law. Planck law in computed according to the formula

$$planck(\lambda, T) = \frac{2hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{k\lambda T}} - 1}$$

where $h = 6.6 \cdot 10^{-34}$ J s is Planck constant, c is speed of light, $k = 1.4 \cdot 10^{-23}$ J K^{-1} is Boltzmann constant.

```
real function planck(lambda,T)
  real lambda
  real T
  real c, h, k
  parameter (c = 3.0 · 10^8)      // speed of light
  parameter (h = 6.6 · 10^-34)    // Planck constant
  parameter (k = 1.4 · 10^-23)    // Boltzmann constant
  planck = (2*h*c*c/lambda^5)/(exp(h*c/(k*T*lambda)) - 1.0)
end
```

20. Output of result matrix.

⟨ Output of result matrix 20 ⟩ ≡

```
open (unit = 2, file = result)
do l = 1, n
   write (2, *) (picture(r, l), r = 1, n)
enddo
close (unit = 2)
print *
print *, 'File␣', result, '␣saved'
```

This code is used in section 1.

21. Index.

COMMAND LINE: `"C:\EMTEX\FWEB\FWEAVE.EXE -PL cometbox.web"`.

WEB FILE: `"cometbox.web"`.

CHANGE FILE: `(none)`.

GLOBAL LANGUAGE: FORTRAN.

September 2, 1998
19:15