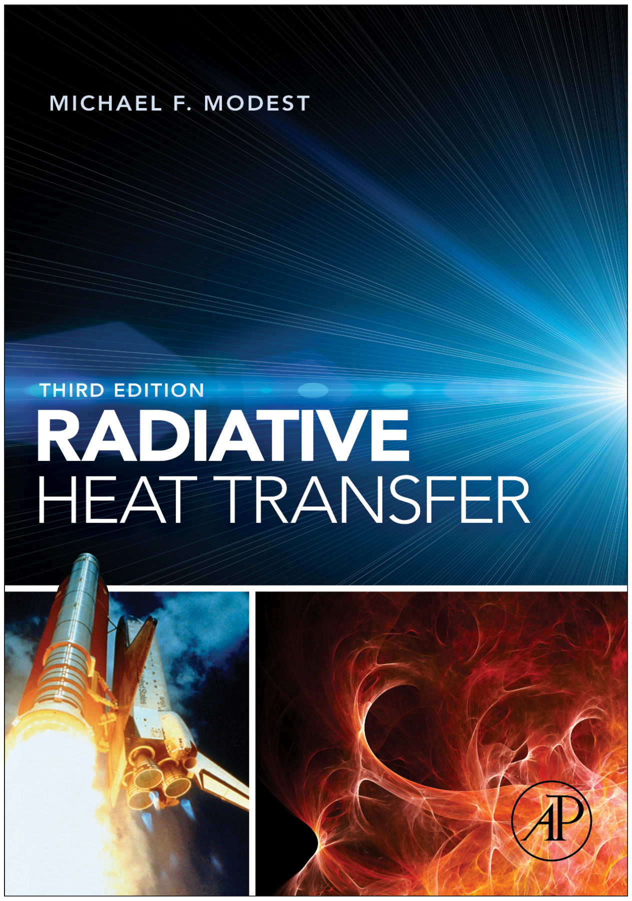
RADIATIVE HEAT TRANSFER



Third Edition

Michael F. Modest
University of California Merced

COMPUTER CODES

Last updated February 4, 2013

Academic Press

New York San Francisco London

This manual/web page contains a listing and brief description of a number of computer programs that may be helpful to the reader of this book, and that can be downloaded from its dedicated website located at http://booksite.elsevier.com/9780123869449. Some of the codes are very basic and are entirely intended to aid the reader with the solution to the problems given at the end of the more basic chapters. Some of the codes were born out of research, but are basic enough to aid a graduate student with more complicated assignments or a semester project. And a few programs are so sophisticated in nature that they will be useful only to the practicing engineer conducting his or her own research. Finally, it is anticipated that the website will be kept up-to-date and augmented once in a while. Thus, there may be a few additional programs not described in this appendix.

It is a fact that most engineers have done, and still do, their programming in Fortran, and the author of this book is no exception. It is also true that computer scientists and most commercial programmers do their work in C++; more importantly, the younger generation of engineers at many universities across the U.S. are now also learning C++. Both compiled languages have in recent years been trumped by MATLAB® [1], which—while an interpreted rather than compiled language—has many convenient mathematical and graphical tools. Since all the programs in this listing were written by the author, either for research purposes or for the creation of this book, they all started their life in Fortran (older programs as Fortran77, and the later ones as Fortran90). However, as a gesture toward the C++ and MATLAB® communities, the most basic codes have all been converted to C++ as well as MATLAB®, as indicated below by the program suffixes `.cpp` and `.m`. If desired, all other programs are easily converted with freeware translators such as `f2c` (resulting in somewhat clumsy, but functional codes). Finally, self-contained programs that have been precompiled for Microsoft Windows have the suffix `.exe`.

The programs are listed in order by chapter in which they first appear. More detailed descriptions, sometimes with an example, can be found on the web site. Third-party codes that are also provided at the web site are listed at the end.

Chapter 1

`bbfn.f`, `bbfn.cpp`, `bbfn.m`
Function `bbfn(x)` calculates the fractional blackbody emissive power, as defined by equation (1.23), where the argument is `x`= $n\lambda T$ with units of μmK.

`planck.f`, `planck.cpp`, `planck.m`, `planck.exe`
`planck` is a small stand-alone program that prompts the user for input (temperature and wavelength or wavenumber), then calculates the spectral blackbody emissive powers $E_{b\lambda}/T^5, E_{b\eta}/T^3$ and the fractional blackbody emissive power $f(\lambda T)$.

Chapters 2 and 3

`fresnel.f`, `fresnel.cpp`, `fresnel.m`
Subroutine `fresnel(n,k,th,rhos,rhop,rho)` calculates Fresnel reflectances from equation (2.113).
Input: `n` (= n) and `k` (= k) are real and imaginary parts of the complex index of refraction, and `th` (= θ) is the off-normal angle of incidence (in radians).
Output: `rhos` (= $\rho_\perp$) and `rhop` (= $\rho_\parallel$) are perpendicular and parallel-polarized reflectance, respectively, while `rho` (= ρ) is the unpolarized reflectance.

Chapter 3

`emdiel.f90, emdiel.cpp`, `emdiel.m`
Function `emdiel(n)` calculates the unpolarized, spectral, hemispherical emissivity of an optical surface of a dielectric material from equation (3.82).
Input: `n` (= n) refractive index of dielectric.

`emmet.f90, emmet.cpp`, `emmet.m`
Function `emmet(n,k)` calculates the unpolarized, spectral, hemispherical emissivity of an optical surface of a metallic material from equation (3.77).
Input: `n` (= n) and `k` (= k) are the real and imaginary parts of the metal's complex index of refraction.

`callemdiel.f90`, `callemdiel.cpp`, `emmet.m`, `callemdiel.exe`
Program `callemdiel` is a stand-alone front end for function `emdiel`, prompting for input (refractive index n) and returning the unpolarized, spectral, hemispherical as well as normal emissivities.

`callemmet.f90`, `callemmet.cpp`, `callemmet.m`, `callemmet.exe`
Program `callemmet` is a stand-alone front end for function `emmet`, prompting for input (complex index of refraction n, k) and returning the unpolarized, spectral, hemispherical as well as normal emissivities.

`dirreflec.f, dirreflec.cpp, dirreflec.m, dirreflec.exe`
Program `dirrecflec` is a stand-alone front end for subroutine `fresnel`, calculating reflectivities for various incidence angles. The user is prompted to input the complex index of refraction, n and k, and the (equal) spacing of incidence angles $\Delta\theta$ (in degrees); the program then returns perpendicular polarized, parallel polarized, and unpolarized reflectivities, as well as unpolarized emissivities.

`totem.f90`, `totem.cpp`, `totem.m`
Program `totem` is a routine to evaluate the total, directional or hemispherical emittance or absorptance of an opaque material, based on an array of spectral data, by 10-point Gaussian quadrature.
Input (by changing data in the heading of function `emlcl(y)`):

`N`	=	number of data points for spectral emittance,
`nrefr`	=	refractive index of adjoining material (`nrefr=1` for vacuum and gases),
`T`	=	temperature of material (for total emittance), or of gray irradiating source (for total absorptance), in K,
`lambda(N)`	=	`N` distinct wavelengths in ascending order, for which the spectral emittance is given, in μm,
`eps(N)`	=	`N` corresponding spectral emittances.

Output (printed to screen):

`emitt`	=	total directional or hemispherical emittance or absorptance.

Case 1: Total, directional emittance (`eps` contains spectral, directional values at temperature T):
From equation (3.8)

$$\begin{aligned}\epsilon'(T,\hat{\mathbf{s}}) &= \frac{1}{n^2\sigma T^4}\int_0^\infty \epsilon'_\lambda(\lambda,T,\hat{\mathbf{s}})E_{b\lambda}(T)\,d\lambda \\ &= \int_0^1 \epsilon'_\lambda\big(\lambda(f),T,\hat{\mathbf{s}}\big)\,df, \end{aligned}\tag{CC-1}$$

where, from equation (1.23)

$$f(n\lambda T) = \int_0^\lambda \frac{E_{b\lambda}d\lambda}{n^2\sigma T^4}. \tag{CC-2}$$

In order to write equation (CC-1) in terms of blackbody fraction f, wavelength must be known as a function of f (for given n and T), i.e., equation (CC-2) must be inverted. The 10 values of $(n\lambda T)$, corresponding to the 10 Gaussian quadrature points $f_i(n\lambda T)$ have been precalculated (using function `bbfn`) and are stored in array `y(i)`. The total emittance is then calculated by expressing equation (CC-1) in quadrature form, or

$$\epsilon'(T,\hat{\mathbf{s}}) \simeq \sum_{i=1}^{10} \epsilon'_\lambda(\lambda_i,T,\hat{\mathbf{s}})w_i, \tag{CC-3}$$

where

$$\lambda_i = y_i/nT, \tag{CC-4}$$

and the w_i are Gaussian quadrature weights. This necessitates that ϵ'_λ must be known at very specific wavelengths, that are not ordinarily part of the given array. The "correct" value for ϵ'_λ is evaluated by linear interpolation between array values, assuming ϵ'_λ = const = `eps(1)` for λ_i <`lambda(1)`, and ϵ'_λ = const = `eps(N)` for λ_i >`lambda(N)`.
Case 2: Total, hemispherical emittance (`eps` contains spectral, hemispherical values at temperature T):

From equation (3.10)

$$\begin{aligned}\epsilon(T) &= \frac{1}{n^2\sigma T^4}\int_0^\infty \epsilon_\lambda(\lambda,T)E_{b\lambda}\,d\lambda = \int_0^1 \epsilon_\lambda\big(\lambda(f),T\big)\,df \\ &\simeq \sum_{i=1}^{10}\epsilon_\lambda(\lambda_i,T)w_i. \end{aligned} \qquad \text{(CC-5)}$$

Thus, the calculation is identical to Case 1.
Case 3: Total, directional absorptance (`eps` contains spectral, directional values at the surface temperature T_s, irradiation is assumed to come from a gray source at temperature T).
From equations (3.23) and (3.31)

$$\begin{aligned}\alpha'(T_s,T,\hat{\mathbf{s}}) &= \frac{1}{n^2\sigma T^4}\int_0^\infty \epsilon'_\lambda(\lambda,T,\hat{\mathbf{s}})E_{b\lambda}(T)\,d\lambda \\ &= \int_0^1 \epsilon'_\lambda\big(\lambda(f),T_s\big)\,df \simeq \sum_{i=1}^{10}\epsilon'_\lambda(\lambda_i,T_s)w_i, \end{aligned} \qquad \text{(CC-6)}$$

and the calculation is again identical.
Case 4: Total, hemispherical absorptance (`eps` contains spectral, hemispherical values at surface temperature T_s; irradiation is assumed to be gray and diffuse with source temperature T).
Then, from equations (3.27) and (3.31)

$$\begin{aligned}\alpha(T_s,T) &= \frac{1}{n^2\sigma T^4}\int_0^\infty \epsilon_\lambda(\lambda,T_s)E_{b\lambda}(T)\,d\lambda \\ &= \int_0^1 \epsilon_\lambda\big(\lambda(f),T_s\big)\,df \simeq \sum_{i=1}^{10}\epsilon_\lambda(\lambda_i,T_s)w_i. \end{aligned} \qquad \text{(CC-7)}$$

Examples
Two examples have been programmed into `totem` (or, rather, function `emlcl`):

1.: The material of Problem 3.1, with a step function in spectral emittance of

$$\epsilon_\lambda = \begin{cases} 0.5, & \lambda < 5\mu\text{m}, \\ 0.3, & \lambda > 5\mu\text{m}, \end{cases}$$

and a temperature of $T = 500$ K. For part *a*) `nrefr=1.0`, and for *b)* `nrefr=2.0` (implemented here) This results in `emitt=0.3435` for *a*) and `emitt=0.4296` for *b*).

2.: Aluminum oxide, as given in Fig. 1-14, discretized into eight equally-spaced values (commented out as given here). For temperature of $T = 500$ K and `nrefr=1.0` this results in `emitt=0.7494`.

Chapter 4 and Appendix D

`view.f90`, `view.cpp`, `view.m`
A function to evaluate any of the 51 view factors given in Appendix D.
Input:

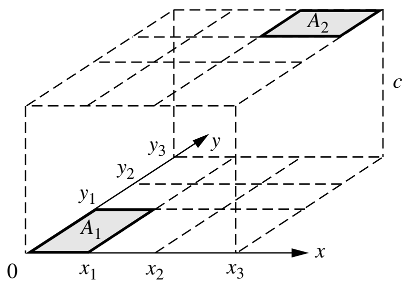
`NO`	=	view factor number, $1 \leq$ `NO` ≤ 51, as given in Appendix D,
`NARG`	=	number of arguments required for view factor,
`ARG`	=	vector of order `NARG` containing the arguments in alphabetical order (Greek characters following the Roman alphabet).

For example, for view factor 14, we have `NO=14`, `NARG=3` and `ARG=`(h,l,r). Upon return the function returns F_{i-j} (except for the infinitesimal view factors 1–9, in which case dF_{d1-d2}/dX is returned, with dX the nondimensional dimension of dA_2).

`parlplates.f90`, `parlplates.cpp`, `parlplates.m`
Contains function `PARLPLTF(X1,X2,X3,Y1,Y2,Y3,Z)` to evaluate the view factor between two displaced parallel plates, as given by equation (4.42).

Input:

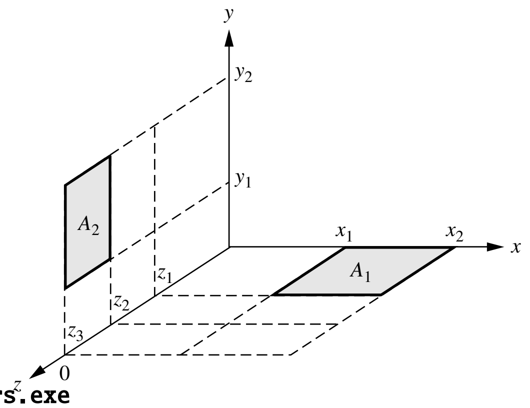
`X1` = Dimension x_1 as given in adjacent sketch (length units)
`X2` = Dimension x_2 as given in adjacent sketch (length units)
`X3` = Dimension x_3 as given in adjacent sketch (length units)
`Y1` = Dimension y_1 as given in adjacent sketch (length units)
`Y2` = Dimension y_2 as given in adjacent sketch (length units)
`Y3` = Dimension y_3 as given in adjacent sketch (length units)
`Z` = Dimension c as given in adjacent sketch (length units)



`perpplates.f90`, `perpplates.cpp`, `perpplates.m`
Contains function `PERPPLTF(X1,X2,Y1,Y2,Z1,Z2,Z3)` to evaluate the view factor between two displaced perpendicular plates, as given by equation (4.41).

Input:

`X1` = Dimension x_1 as given in adjacent sketch (length units)
`X2` = Dimension x_2 as given in adjacent sketch (length units)
`Y1` = Dimension y_1 as given in adjacent sketch (length units)
`Y2` = Dimension y_2 as given in adjacent sketch (length units)
`Z1` = Dimension z_1 as given in adjacent sketch (length units)
`Z2` = Dimension z_2 as given in adjacent sketch (length units)
`Z3` = Dimension z_3 as given in adjacent sketch (length units)



`viewfactors.f90`, `viewfactors.cpp`, `viewfactors.m`, `viewfactors.exe`
A stand-alone front end to functions `view`, `parlplates` and `perpplates`. The user is prompted to input configuration number and arguments; the program then returns the requested view factor.

Chapter 5

`graydiff.f90`, `graydiff.cpp`, `graydiff.m`:
Subroutine `graydiff` provides the solution to equation (5.38) for an enclosure consisting of N gray-diffuse surfaces. For each surface the area, emittance, external irradiation and either heat flux or temperature must be specified. In addition, the upper triangle of the view factor matrix must be provided (F_{i-j}; $i = 1,N$; $j = i,N$). For closed configurations, the diagonal view factors F_{i-i} are not required, since they can be calculated from the summation rule. The remaining view factors are calculated from reciprocity. On output, the program provides all view factors, and temperatures and radiative heat fluxes for all surfaces.
Input:

`N`	=	number of surfaces in enclosure
`iclsd`	=	closed or open configuration identifier
		`iclsd`= 1: configuration is closed; diagonal F_{i-i} evaluated from summation rule
		`iclsd`$\neq$ 1: configuration has openings; F_{i-i} must be specified
`A(N)`	=	vector containing surface areas, [m^2]
`EPS(N)`	=	vector containing surface emittances
`HO(N)`	=	vector containing external irradiation, in [W/m^2]
`F(N,N)`	=	vector containing view factors; on input only F_{i-j} with $j > i$ (`iclsd`=1) or $j \geq i$ (`iclsd`$\neq$ 1) are required; remainder are calculated
`ID(N)`	=	vector containing surface identifier:
		`ID`=0: surface heat flux is specified, in [W/m^2]
		`ID`=1: surface temperature is specified, in [K]
`PIN(N)`	=	vector containing surface emissive powers (`id`=1) and fluxes (`id`=2)

Output:

`POUT(N)`	=	vector containing unknown surface fluxes (for surfaces with `id`=1) and emissive powers (for surfaces with `id`=0)

`graydiffxch.f90`, `graydiffxch.cpp`, `graydiffxch.m`
Program `graydiffxch` is a front end for subroutine `graydiff`, generating the necessary input parameters for a three-dimensional variation to Example 5.4 (making the four surfaces of finite length ℓ, and introducing front and back surfaces A_5 and A_6, both at the same conditions as the left and right sides, i.e., $T_5 = T_6 = 600\,\mathrm{K}$ and $\epsilon_5 = \epsilon_6 = 0.8$), primarily view factors calculated by calls to function `view`. This program may be used as a starting point for more involved radiative exchange problems.

Chapter 6

`graydifspec.f90`, `graydifspec.cpp`, `graydifspec.m`
Subroutine `graydifspec` provides the solution to equation (6.23) for an enclosure consisting of N diffusely emitting surfaces with diffuse and specular reflectance components. For each surface the area, emittance, external irradiation and either heat flux or temperature must be specified. In addition, the upper triangle of the view factor matrix must be provided (F^s_{i-j}; $i = 1, N$; $j = i, N$). For closed configurations, the diagonal view factors F^s_{i-i} are not required, since they can be calculated from the summation rule. The remaining view factors are calculated from reciprocity. On output, the program provides all view factors, and temperatures and radiative heat fluxes for all surfaces.
Input:

`N`	=	number of surfaces in enclosure
`iclsd`	=	closed or open configuration identifier `iclsd`= 1: configuration is closed; diagonal F^s_{i-i} evaluated from summation rule `iclsd`≠ 1: configuration has openings; F^s_{i-i} must be specified
`A(N)`	=	vector containing surface areas, [m^2]
`EPS(N)`	=	vector containing surface emittances
`RHOs(N)`	=	vector containing surface specular reflectance components
`HOs(N)`	=	vector containing external irradiation, in [W/m^2]
`Fs(N,N)`	=	vector containing view factors; on input only F^s_{i-j} with $j > i$ (`iclsd`=1) or $j \geq i$ (`iclsd`≠ 1) are required; remainder are calculated
`ID(N)`	=	vector containing surface identifier: `ID`=0: surface heat flux is specified, in [W/m^2] `ID`=1: surface temperature is specified, in [K]
`PIN(N)`	=	vector containing surface emissive powers (`id`=1) and fluxes (`id`=2)

Output:

`POUT(N)`	=	vector containing unknown surface fluxes (for surfaces with `id`=1) and emissive powers (for surfaces with `id`=0)

`grspecxch.f90`, `grspecxch.cpp`, `grspecxch.m`
Program `grspecxch` is a front end for subroutine `graydifspec`, generating the necessary input parameters for a three-dimensional variation to Example 6.7 (making the four surfaces of finite length ℓ, and introducing front and back surfaces A_5 and A_6, both diffusely reflecting at the same conditions as the left and right sides, i.e., $T_5 = T_6 = 600\,\mathrm{K}$ and $\epsilon_5 = \epsilon_6 = 0.8$), primarily view factors calculated by calls to function `view`. This program may be used as a starting point for more involved radiative exchange problems.

Chapter 7

`semigray.f90`, `semigray.cpp`, `semigray.m`,
`semigraydf.f90`, `semigraydf.cpp`, `semigraydf.m`
Subroutine `semigray` provides the solution to equations (7.5) for an enclosure consisting of N diffusely emitting surfaces with diffuse and specular reflectance components, considering two spectral ranges (one for external irradiation, one for emission). For each surface the area, emittance and specular reflectance (two values each), external irradiation and either heat flux or temperature must be specified. In addition, the upper triangle of the view factor matrix must be provided for both spectral ranges (F^s_{i-j}; $i = 1, N$; $j = i, N$). For closed configurations,

the diagonal view factors F^s_{i-i} are not required, since they can be calculated from the summation rule. The remaining view factors are calculated from reciprocity. On output, the program provides all view factors, and temperatures and radiative heat fluxes for all surfaces.

Input:

`N`	=	number of surfaces in enclosure
`iclsd`	=	closed or open configuration identifier `iclsd`= 1: configuration is closed; diagonal F^s_{i-i} evaluated from summation rule `iclsd`≠ 1: configuration has openings; F^s_{i-i} must be specified
`A(N)`	=	vector containing surface areas, [m^2]
`EPS(2,N)`	=	vector containing surface emittances for 2 spectral ranges
`RHOs(2,N)`	=	vector containing surface specular reflectance components for 2 spectral ranges
`HOs(N)`	=	vector containing external irradiation, in [W/m^2]
`Fs(2,N,N)`	=	vector containing view factors for 2 spectral ranges; on input only F^s_{i-j} with $j > i$ (`iclsd`=1) or $j \geq i$ (`iclsd`≠ 1) are required; remainder are calculated
`ID(N)`	=	vector containing surface identifier: `ID`=0: surface heat flux is specified, in [W/m^2] `ID`=1: surface temperature is specified, in [K]
`PIN(N)`	=	vector containing surface emissive powers (`id`=1) and fluxes (`id`=2)

Output:

`POUT(N)`	=	vector containing unknown surface fluxes (for surfaces with `id`=1) and emissive powers (for surfaces with `id`=0)

Subroutine `semigraydf` is a simplified version of subroutine `semigray` by assuming all surfaces to be diffuse, and input is changed by requiring `HO(N)` and `F(N,N)` (and no reflectance) instead of `RHOs(2,N)`, `HOs(N)` and `Fs(2,N,N)` (note that diffuse view factors do not depend on reflectance properties).

`semigrxch.f90`, `semigrxch.cpp`, `semigrxch.m`,
`semigrxchdf.f90`, `semigrxchdf.cpp`, `semigrxchdf.m`

Program `semigrxch` is a front end for subroutine `semigray` providing the necessary input for Example 7.1, primarily view factors calculated by calls to function `view`; similarly, program `semigrxchdf` is a front end for subroutine `semigraydf`. These programs may be used as a starting point for more involved radiative exchange problems.

`bandapp.f90`, `bandapp.cpp`, `bandapp.m`,
`bandappdf.f90`, `bandappdf.cpp`, `bandapp.m`

Subroutine `bandapp` provides the solution to equations (7.6) for an enclosure consisting of N diffusely emitting surfaces with diffuse and specular reflectance components, considering M spectral bands. For each surface the area, emittance, specular reflectance and external irradiation (one value for each spectral band), and either heat flux or temperature must be specified. In addition, the upper triangle of the view factor matrix must be provided for each spectral band (F^s_{i-j}; $i = 1,N$; $j = i,N$). For closed configurations, the diagonal view factors F^s_{i-i} are not required, since they can be calculated from the summation rule. The remaining view factors are calculated from reciprocity. On output, the program provides all view factors, and temperatures and radiative heat fluxes for all surfaces.

Input:

`M`	=	number of spectral bands
`N`	=	number of surfaces in enclosure
`iclsd`	=	closed or open configuration identifier `iclsd`= 1: configuration is closed; diagonal F^s_{i-i} evaluated from summation rule `iclsd`≠ 1: configuration has openings; F^s_{i-i} must be specified
`A(N)`	=	vector containing surface areas, [m^2]
`EPS(M,N)`	=	matrix containing surface emittances for M spectral ranges
`RHOs(M,N)`	=	matrix containing surface specular reflectance components for M spectral ranges
`HOs(M,N)`	=	matrix containing external irradiation for M spectral ranges, in [W/m^2]
`Fs(M,N,N)`	=	matrix containing view factors for M spectral ranges; on input only F^s_{i-j} with $j > i$ (`iclsd`=1) or $j \geq i$ (`iclsd`≠ 1) are required; remainder are calculated

`ID(N)`	=	vector containing surface identifier: `ID`=0: surface heat flux is specified, in [W/m^2] `ID`=1: surface temperature is specified, in [K]
`q(N)`	=	vector containing known surface fluxes (only for surfaces with `id`=2)
`T(N)`	=	vector containing known surface temperatures (only for surfaces with `id`=1)

Output:

`q(N)`	=	vector containing known surface fluxes (for all surfaces)
`T(N)`	=	vector containing known surface temperatures (for all surfaces)

Subroutine `bandappdf` is a simplified version of subroutine `bandapp` by assuming all surfaces to be diffuse, and input is changed by requiring `HO(M,N)` and `F(N,N)` (and no reflectance) instead of `RHOs(M,N)`, `HOs(M,N)` and `Fs(M,N,N)` (note that diffuse view factors do not depend on reflectance properties).

bandmxch.f90, bandmxch.cpp, bandmxch.m, bandmxchdf.f90, bandmxchdf.cpp, bandmxch.m

Program `bandmxch` is a front end for subroutine `bandapp` providing the necessary input for Example 7.2, primarily view factors calculated by calls to function `view`; similarly, program `bandmxchdf` is a front end for subroutine `bandappdf`. These programs may be used as a starting point for more involved radiative exchange problems.

Chapter 8

MCintegral.f90

`MCintegral` is a little program that evaluates the integral $\int_a^b f(x)\,dx$ for any specified function by the Monte Carlo method, as outlined in equation (8.10).

Input:

`F(x)`	=	The function to be integrated
`a`	=	Lower limit of integration
`b`	=	Upper limit of integration
`varmax`	=	Maximum relative standard deviation allowed

Output:

`no. of bundles`	=	Number of statistical samples taken (with a minimum of 10000 built in)
`integral`	=	Best estimate of the value of the desired integral
`std dev`	=	Absolute standard deviation for the result
`rel.err(%)`	=	Estimated relative error (in %), based on one standard deviation.

The number of statistical bundles is broken up into `numsmpls` realizations of `N` samples each. Using these different realizations, variances are calculated according to equation (8.8), and the relative variance is compared to `stddevmax`; if it exceeds it the number of bundles is doubled, the `numsmpls` realizations are compacted into half that many, and `numsmpls/2` new realizations (with twice as many samples) are generated (giving `numsmpls` realizations with twice as many samples as before), etc., until the convergence criterion is met. For example, `F(x)=sin(x)`, `a=0.`, `b=pi/2.`, and `varmax=0.002` results in (the correct answer being 1):

```
no. of bundles    integral    std dev   rel.err(%)
         10000  1.0024E+00  4.8714E-03   0.49
         20000  9.9957E-01  2.8855E-03   0.29
         40000  1.0001E+00  1.4426E-03   0.14
```

Chapter 11

voigt.f

Fortran77 subroutine `voigt(S,bL,bD,deta,keta)` calculates the spectral absorption coefficient for a Voigt-shaped line based on the fast algorithm by Humlíček [2].

Input:

`S`	=	is the line intensity S, in cm^{-2},
`bL`	=	is the Lorentz line width b_L, in cm^{-1},
`bD`	=	is the Doppler line width b_D, in cm^{-1},
`deta`	=	is the spectral distance $\Delta\eta$ away from the line center, at which κ_η is to be evaluated.

Output:

`keta`	=	is the spectral absorption coefficient of the Voigt line κ_η at $\eta = \eta_0 \pm \Delta\eta$, where η_0 is the wavenumber of the line center.

`nbkdistdb.f90`

Program `nbkdistdb` is a Fortran90 code to calculate narrow band k-distributions for a number of temperatures and a number of wavenumber ranges, for a gas mixture containing CO_2, H_2O, CH_4 and soot. The spectral absorption coefficient is calculated directly from the HITRAN or HITEMP databases.

Input:

`Tmin`	=	minimum temperature for which a k-distribution is to be calculated, in K,
`Tmax`	=	maximum temperature for which a k-distribution is to be calculated, in K,
`numT`	=	number of different temperatures to be considered; equally spaced between `Tmin` and `Tmax`,
`P`	=	total pressure of gas mixture, bar,
`xmfr(3)`	=	mole fraction vector; `xmfr(1)`= mole fraction of CO_2, `xmfr(2)`= mole fraction of H_2O, `xmfr(3)`= mole fraction of CH_4,
`fvsoot`	=	volume fraction of soot,
`nsoot`, `ksoot`	=	complex index of refraction for the soot; its absorption coefficient is assumed linear in wavenumber, using equation (12.112),
`wvnm_b`	=	minimum wavenumber considered, cm^{-1},
`wvnm_e`	=	maximum wavenumber considered, cm^{-1},
`wvnmbuf`	=	line wing influence of spectral lines centered in the wavenumber range `wvnmbuf` cm^{-1} below `wvnm_b` and above `wvnm_e` are considered in the absorption coefficient calculation, cm^{-1},
`wvnmst`	=	wavenumber step (equally spaced) with which the absorption coefficient for the mixture is calculated from the HITRAN or HITEMP database, cm^{-1},
`kdrnge`	=	wavenumber range for individual k-distributions; `wvnm_e-wvnm_b` should be an integer multiple of `kdrnge`, in cm^{-1},
`n_pwrk`	=	number of different k-bin values considered in the construction of the k-distribution,
`pwr`	=	exponent for k-bin values spacing: k-bins are equally spaced in $\mathtt{k}^{pwr}$ between `kmin` (=minimum k to be considered) and `kmax` (=maximum absorption coefficient across spectrum).
`nq`	=	number of quadrature points for radiative calculations, i.e., the number of RTE evaluations to be performed before spectral integration (over cumulative k-distribution g),
`iwr`	=	absorption coefficient switch: `iwr=0` to make a single complete run, i.e., evaluating κ_η from HITRAN or HITEMP (without storing them), followed by generation of k-distributions, `irw=1` same, but absorption coefficient is stored for future use, and `iwr=2`: precalculated absorption coefficients are read in and k-distributions are generated.
`ipl`	=	linear vs. pressure-based absorption coefficient switch: `ipl`=0: calculate linear absorption coefficient, in cm^{-1} `ipl`=1: calculate pressure-based absorption coefficient (allowed only for single absorbing gas!), in cm^{-1} bar^{-1}; if the pressure-based absorption coefficient for a dilute gas is desired, set `xmfr=1.d-3` (=0.1%)
`ipr`	=	output switch: see under output

Output:

`ipr=0`: For each of the `numkd=wvnm_e-wvnm_b/kdrnge` narrow band ranges only the `nq` quadrature points, weights, and $k(T, g)$ (for all temperatures) are printed: the first line of the output file, called `nbkvsgq.dat` by default, contains the first and last wavenumbers of the first narrow band range, followed by `nq` lines containing `gq` (the i-th quadrature point), `wq` (the i-th quadrature weight), and `numT` values of kq [$= k(T_j, g_i)$; one for each temperature T_j]. This is followed by a line containing the first and last wavenumbers of the second narrow band range, etc.

`ipr=1`: Besides the output for `ipr`=0 a second output file is prepared with the complete k-distribution information, i.e., for each narrow band and each temperature all `n_pwrk` values of k, f and g are printed: the first line of the output file, called `nbkvsg.dat` by default, contains the first temperature and first and last wavenumbers of the first narrow band range, followed by `n_pwrk-1` lines containing `k` (the i-th k-bin value), `ff` [its k-distribution value $f(k)$], and `gg` [its cumulative k-distribution value $g(k)$]. This is followed by a line containing the second temperature and first and last wavenumbers of the first narrow band range, etc., looping over all temperatures and narrow band ranges.

ipr=2: Only the complete k-distribution information is printed, i.e., only output file `nbkvsg.dat` is generated.

Example:
We consider a set of narrow band k-distributions for a linear absorption coefficient (`ipl`=0) of pure CO_2, for a mole fraction of 10% (`xmfr(3)=(/0.1d0,0.d0,0.d0/)`). The absorption coefficient is calculated in this run (`iwr`=1), and is stored in file `C:\absco\absctmp.dat` (for a wavenumber range from 2320 cm^{-1} to 2380 cm^{-1}, but also considering lines centered at wavenumbers as low as 2315 cm^{-1} and as high as 2385 cm^{-1}, `wvnmbuf=5.`) with a $\delta\eta = 0.001\,cm^{-1}$. We will calculate the k-distributions for 4 temperatures, equally spaced between T_{min} = 300 K and T_{max} = 1200 K (`numT`=4): this results in the 4 temperatures of 300 K, 600 K, 900 K and 1200 K. Each k-distribution will be over a range of $\Delta\eta = 10\,cm^{-1}$ wavenumbers (`kdrnge=10.`), i.e., there will be 6 narrow bands. We will use 500 k-bins (`n_pwrk`=500) with `pwr`=0.1 (this spreads the k-bins over many orders of magnitude, but places more and more bins into large magnitudes; see output file). We also set `klmin`=10^{-9} (cm^{-1}), i.e., we will consider absorption coefficient contributions as small as 10^{-9} cm^{-1}. Finally, we set `ipr`=1 and `nq`=10, i.e., besides truncated k-distributions ready-made for numerical quadrature, using 10 quadrature points, we want to also print to file the full k-distributions. The top of the program with input parameters, therefore, looks like this:

```
  MODULE Key
   IMPLICIT NONE
!HITRAN/HITEMP DATABASE
   INTEGER  :: lu
   INTEGER,PARAMETER  :: rows=1400000
   DOUBLE PRECISION,PARAMETER   :: wvnm_b=2320.d0,wvnm_e=2380.d0,wvnmbuf=5.d0,wvnmst=0.001d0
   DOUBLE PRECISION   :: data(rows,6),wvnm_l=wvnm_b-wvnmbuf,wvnm_r=wvnm_e+wvnmbuf
  END MODULE Key

   PROGRAM Main
   USE Key
! Input parameters
   INTEGER,PARAMETER :: numT=4,n_pwrk=500,nq=10,iwr=1,ipl=0,ipr=1
   DOUBLE PRECISION,PARAMETER :: P=1.d0,Tmin=300d0,Tmax=1200d0,kdrnge=10.
   DOUBLE PRECISION,PARAMETER :: xmfr(3)=(/0.10d0,0.00d0,0.d0/),pwr=0.1d0,klmin=1.d-9
   DOUBLE PRECISION,PARAMETER :: fvsoot=0.d-6,nsoot=1.89d0,ksoot=0.92d0
```

where we have changed the values for `wvnm_b, wvnm_e, wvnmst`, `numT`, `n_pwrk`, `iwr, ipr, nq, Tmin, Tmax` and `xmfr` to fit our needs. Also, in this simulation we have set file names as

```
! Open output files
   IF(ipr<2) OPEN(7,FILE='nbkvsgqco2.dat',STATUS='unknown')
   IF(ipr>0) OPEN(8,FILE='nbkvsgco2.dat',STATUS='unknown')
! File containing absorption coefficient
   IF(iwr>0) OPEN(9,FILE='C:\absco\absctmp.dat',STATUS='unknown')
```

i.e., the absorption coefficient as calculated here is placed into `c:\absco\absctmp.dat` (and can be reused later by setting `iwr`=2), while the long k-distribution output (500 values for each temperature and narrow band) will be put into `nbkvsgco2.dat`, and the short, quadrature-ready output into `nbkvsgqco2.dat`. Note that the header lines for `absctmp.dat` are formatted such that the absorption coefficient can be plotted from them using the Tecplot drafting package. The other output files will need some reformatting before they can be used for plotting.

We will also assume that Numerical Recipes subroutines are available, leaving the following lines unchanged:

```
! Selection of g-values for numerical quadrature, using a Numerical Recipes routine
! If Numerical Recipes is not available, set nq=12, comment out the following 8 lines of code,
! and uncomment the 5-line REAL declaration following it
   REAL             :: gqs(nq),wqs(nq),kq(numt,nq),gq(nq),wq(nq),gaujac,alf=3.,bet=-.6,sum
! Get quadrature coefficients from Numerical recipies
    sum=0.
    CALL GAUJAC(gqs,wqs,nq,alf,bet)
          do iq=1,nq
            gq(iq)=0.5*(1.-gqs(iq))
            wq(iq)=wqs(iq)/(2.**(alf+bet+1)*gq(iq)**alf*(1.-gq(iq))**bet)
            sum=sum+wq(iq)
          enddo
! Correction to make sum(wq)=1
          wq=wq/sum
! End quadrature coefficients from Numerical recipies
! Selection of precalculated g-values for numerical quadrature, for nq=12,alf=3.,bet=0.
!   REAL              :: kq(numt,nq), &
!         gq(nq)=(/ 5.120075E-02,1.170678E-01,2.015873E-01,3.007074E-01,4.095012E-01,5.225285E-01,  &
!                   6.341280E-01,7.387071E-01,8.310236E-01,9.064499E-01,9.612060E-01,9.925594E-01/),&
!         wq(nq)=(/ 5.556622E-02,7.576839E-02,9.258290E-02,1.048306E-01,1.118451E-01,1.132605E-01,  &
!                   1.090012E-01,9.927844E-02,8.457905E-02,6.563999E-02,4.341329E-02,1.904792E-02/)
```

This will calculate quadrature points `gq` and weights `wq` using Gaussian quadrature of moments (`alf=3` sets 3rd order moments). For users without access to Numerical Recipes the `gq` and `wq` calculated here have been put in data statements and may be used instead by following the guidelines above.

The absorption coefficient placed into `c:\absco\absctmp.dat` has the following form:

```
variables = "wvn" "absco0300K" "absco0600K" "absco0900K" "absco1200K"
zone i=   60001
  2320.00000   0.43878E+00   0.34411E+00   0.33293E+00   0.35420E+00
  2320.00100   0.43694E+00   0.34266E+00   0.33335E+00   0.35600E+00
  2320.00200   0.43512E+00   0.34125E+00   0.33386E+00   0.35783E+00
  2320.00300   0.43333E+00   0.33988E+00   0.33447E+00   0.35968E+00
  2320.00400   0.43157E+00   0.33856E+00   0.33516E+00   0.36155E+00
   .
   .
   .
```

It is formatted for easy plotting using Tecplot, and has 60,001 absorption coefficient values between 2320 cm^{-1} and 2380 cm^{-1}, spaced 0.001 cm^{-1} apart.
The output file `nbkvsgco2.dat` has this form:

```
T= 300.K, wvnm_lft= 2320.000000cm-1, wvnm_rgt= 2330.000000cm-1
      k              f             g
  0.325271D+00  0.615250D-02  0.625249D-02
  0.328970D+00  0.262559D-02  0.887808D-02
  0.332708D+00  0.209533D-02  0.109734D-01
  0.336484D+00  0.188093D-02  0.128543D-01
  0.340299D+00  0.183458D-02  0.146889D-01
  .
  .
  .
  0.277993D+02  0.340523D-03  0.997833D+00
  0.280016D+02  0.402225D-03  0.998235D+00
  0.282052D+02  0.521735D-03  0.998757D+00
  0.284102D+02  0.124290D-02  0.100000D+01

T= 600.K, wvnm_lft= 2320.000000cm-1, wvnm_rgt= 2330.000000cm-1
      k              f             g
  0.187475D+00  0.525121D-02  0.535120D-02
  0.189577D+00  0.199556D-02  0.734676D-02
  0.191700D+00  0.138701D-02  0.873377D-02
  .
  .
  .
```

Finally, output file `nbkvsgqco2.dat` contains quadrature *k*-values as:

```
wvnm_lft= 2320.000000cm-1, wvnm_rgt= 2330.000000cm-1
      gq            wq           kq(T1)        kq(T2)    ...
  0.729136D-01  0.813193D-01  0.400407D+00  0.242578D+00  0.183572D+00  0.160547D+00
  0.165015D+00  0.108536D+00  0.541222D+00  0.297547D+00  0.226229D+00  0.204381D+00
  0.280173D+00  0.128592D+00  0.672925D+00  0.335421D+00  0.278275D+00  0.240816D+00
  0.410404D+00  0.139547D+00  0.867542D+00  0.418648D+00  0.336797D+00  0.295848D+00
  0.546441D+00  0.140538D+00  0.118950D+01  0.584868D+00  0.422875D+00  0.361822D+00
  0.678556D+00  0.131471D+00  0.163219D+01  0.902429D+00  0.560377D+00  0.425924D+00
  0.797291D+00  0.112988D+00  0.286678D+01  0.156588D+01  0.795057D+00  0.518610D+00
  0.894140D+00  0.864116D-01  0.739453D+01  0.262995D+01  0.130120D+01  0.731875D+00
  0.962165D+00  0.536406D-01  0.168294D+02  0.783404D+01  0.286791D+01  0.123165D+01
  0.996473D+00  0.169570D-01  0.268487D+02  0.142687D+02  0.658947D+01  0.326066D+01

wvnm_lft= 2330.000000cm-1, wvnm_rgt= 2340.000000cm-1
      gq            wq           kq(T1)        kq(T2)    ...
  0.729136D-01  0.813193D-01  0.716314D+00  0.299593D+00  0.223759D+00  0.171072D+00
  0.165015D+00  0.108536D+00  0.788507D+00  0.371208D+00  0.277792D+00  0.221426D+00
  0.280173D+00  0.128592D+00  0.943415D+00  0.436240D+00  0.339969D+00  0.280705D+00
  .
  .
  .
```

for each of the 6 narrow bands.

Note that the code has an accuracy-checking mechanism built in: an average narrow band absorption coefficient is calculated directly through line-by-line integration of the absorption coefficient, equation (11.60), and is compared with the mean absorption coefficient as calculated from the k-g-distribution. If the discrepancy exceeds 0.5% a message is printed to the screen, warning that k-bin spacing is too coarse (`n_pwrk` too small) to properly resolve the absorption coefficient. For the above example, the choice of `n_pwrk`=500 results in an error larger than 0.5% only for 2340–2350 cm^{-1} narrow band at 300 K (0.52%), as indicated by the warning message.

`nbkdistsg.f90`

Subroutine `nbkdistsg` calculates a single narrow band k-distribution from a given set of spectral absorption coefficients and corresponding wavenumbers.

Input:

`Deta`	= wavenumber range for which a k-distribution is to be calculated, in cm^{-1},
`numk`	= number of absorption coefficient datapoints, equally spaced in wavenumbers,
`n_pwrk`	= number of k-boxes for k-distribution,
`pwr`	= exponent for setting of k-box values; i.e., k-values are chosen in equal steps of k^{pwr},
`nq`	= number of quadrature points for Gaussian quadrature,
`ipr`	= print switch: `ipr=0`: prints k and w (Gaussian quadrature weights) vs. g only for Gaussian A quadrature points; `ipr=1`: prints k and w vs. g for Gaussian quadrature points, as well as k vs. f and g for all `n_pwrk` k-bins; `ipr=2`: prints only k vs. f and g for all `n_pwrk` k-bins.

file named `absco.dat` containing absorption coefficient data is required: The first line must contain `numk` and `Deta` (in I5,F7.4 format); second through (`numk`+1)th lines contain `wvnm,absco` (in e12.4 format).

Output:

nbkvsg.dat: Output file in Tecplot format (if `ipr=1` or `2`), containing one line giving wavenumber range, then k, f, g for `n_pwrk` k-values.

nbkvsgq.dat: Output file in Tecplot format (if `ipr=0` or `1`), containing one line giving wavenumber range, then k, w, g for `nq` Gaussian quadrature points (`nq=12` set as default: see discussion on Gaussian quadrature in `nbkdistdb.f90`).

`nbkdistsg.f90` is a streamlined version of `nbkdistdb.f90` and, thus, much of the discussion in the example for `nbkdistdb.f90` pertains here, as well. As provided, `nbkdistsg.f90` is embedded in a stand-alone program called `nbkdist_sngl.f90`, which first calculates the absorption coefficient data for the mixture in Example 11.5, then calls `nbkdistsg.f90` to determine the k-distribution given in Fig. 11-19.

`wbmxxx.f, wbmxxxcl.f, wbmxxxcl.exe`

Double precision Fortran77 subroutines `wbmxxx(T,PSIr,PHIr)`, where `xxx` stands for the different gases `h2O`, `co2`, `ch4`, `co`, `no` and `so2`, calculate for a given temperature `T` the ratios `PSIr` = $\Psi^*(T)/\Psi^*(T_0)$ [from equations (11.144) and (11.148)] and `PHIr` = $\gamma/\gamma_0 = \sqrt{T_0/T}\Phi(T)/\Phi(T_0)$ [from equation (11.149)], i.e., the functions shown in Figs. 11-23 through 11-25, for all bands given in Table 11.3 in the order as listed (in order of decreasing band center wavelengths). For example, a call to `wbmch4(1200.,PSIr,PHIr)` would produce 4 values each for `PSIr` and `PHIr`, and `PSIr(3)` would contain the value of $\Psi^*(1200K)/\Psi^*(T_0) = 1.29540$ for the 2.4 μm band of methane, etc. The stand-alone programs `wbmxxxcl.f` perform the identical calculations, prompting the user for input (`T`), and printing `PSIr` and `PHIr` to the screen for all bands listed in Table 11.3.

`emwbm.f, ftwbm.f, wangwbm.f`

Double precision Fortran77 functions to calculate the nondimensional total band absorptance A^* from the Edwards and Menard model, Table 11.2 (`emwbm(tau,beta)`), the Felske and Tien model, equation (11.156) (`ftwbm(tau,beta)`), and the Wang model, equation (11.158) (`wangwbm(tau,beta)`).

`wbmodels.f, wbmodels.exe`

Stand-alone double precision Fortran77 front end for the functions `emwbm`, `ftwbm` and `wangwbm`; the user is prompted to input `tau` (= τ_0, optical thickness at band center) and `beta` (= β, overlap parameter); the nondimensional total band absorptance A^* is printed to the screen, as calculated from three band models (Edwards and Menard, Felske and Tien, and Wang models).

`wbkvsg.f`

Double precision Fortran77 subroutine `wbkvsg(beta,kmax,kmin,n,k,g)` calculates the κ^* vs. g^* distribution of

equation (11.170).
Input:

`beta`	=	β, the overlap parameter,
`kmax`	=	κ^*_{max}, the maximum κ^*-value to be output,
`kmin`	=	κ^*_{min}, the minimum κ^*-value to be output,
`n`	=	the number of κ^* and g^* values to be output, [equally spaced in $\ln(\sqrt{\kappa^*})$],

Output:

`k,g`	=	κ^*, g^*, n values each for κ^* and g^*, [equally spaced in $\ln(\sqrt{\kappa^*})$].

The integral in equation (11.170) is evaluated by first transforming the integration variable from κ^* to $a = \ln(\sqrt{\kappa^*})$, or

$$g^* = \int_{\ln(\sqrt{\kappa^*})}^{a_{\text{max}}=\ln(\sqrt{10^5})} \left[\text{erfc}(\sqrt{\beta}\sinh a) - e^{\beta}\text{erfc}(\sqrt{\beta}\cosh a)\right] da,$$

followed by a simple Newton-Cotes integration. Beginning point of the integration is a_{max} and a minimum step size for the numerical integration is determined and used. However, only values for `kmax` $> \kappa^* >$ `kmin` for `n` values equally spaced in a are output to arrays `k` and `g`.
Notes:
(i) Values of `kmax` $> 10^5$ are truncated;
(ii) Program assumes availability of double precision functions `derfc`, `dcosh` and `dsinh`.
As an example we consider the *k*-distribution of Example 10.9. Writing a small Fortran calling program

```
      program callwbkvsg
      integer n,i
      real*8 beta,kmax,kmin,k(1000),g(1000),c1,c2,kdim,deta
      OPEN(9,FILE='wbkvsg.dat',STATUS='unknown')
      beta=0.211d0
      kmax=1.d1
      kmin=1.d-3
      n=40
      c1=54.84*41.2/138.15/100.   ! rho-alpha/omega with kappa in cm-1
      c2=138.15/2.                ! omega/2
      WRITE(9,9)
      call wbkvsg(beta,kmax,kmin,n,k,g)
      DO i=1,n
       kdim=c1*k(i)
       deta=c2*g(i)
       WRITE(9,10) k(i),g(i),kdim,deta
      ENDDO
      CLOSE(9)
 9    FORMAT('    kstar     gstar     kdim      deta')
10    FORMAT(3f10.5,f8.2)
      stop
      end
```

leads to

```
    kstar     gstar     kdim      deta
  10.00000   0.00942   1.63547    0.65
   7.89652   0.01448   1.29146    1.00
   6.23551   0.02141   1.01980    1.48
   4.92388   0.03064   0.80529    2.12
   3.88816   0.04264   0.63590    2.95
   3.07029   0.05791   0.50214    4.00
   2.42446   0.07702   0.39651    5.32
   1.91448   0.10058   0.31311    6.95
   1.51178   0.12924   0.24725    8.93
   1.19378   0.16374   0.19524   11.31
   0.94267   0.20484   0.15417   14.15
   0.74438   0.25339   0.12174   17.50
   0.58780   0.31028   0.09613   21.43
   0.46416   0.37645   0.07591   26.00
   0.36652   0.45290   0.05994   31.28
   0.28943   0.54061   0.04733   37.34
   0.22855   0.64057   0.03738   44.25
   0.18047   0.75370   0.02952   52.06
   0.14251   0.88079   0.02331   60.84
   0.11253   1.02244   0.01840   70.62
   0.08886   1.17894   0.01453   81.44
   0.07017   1.35022   0.01148   93.27
   0.05541   1.53570   0.00906  106.08
   0.04375   1.73426   0.00716  119.79
   0.03455   1.94425   0.00565  134.30
   0.02728   2.16360   0.00446  149.45
   0.02154   2.38997   0.00352  165.09
   0.01701   2.62110   0.00278  181.05
   0.01343   2.85504   0.00220  197.21
   0.01061   3.09039   0.00173  213.47
   0.00838   3.32632   0.00137  229.77
   0.00661   3.56244   0.00108  246.08
   0.00522   3.79859   0.00085  262.39
   0.00412   4.03475   0.00067  278.70
   0.00326   4.27092   0.00053  295.01
   0.00257   4.50708   0.00042  311.33
   0.00203   4.74324   0.00033  327.64
   0.00160   4.97940   0.00026  343.95
   0.00127   5.21557   0.00021  360.27
   0.00100   5.45173   0.00016  376.58
```

`totemiss.f`

Double precision Fortran77 subroutine `totemiss(ph2o,pco2,ptot,Tg,L,epsh2o,epsco2,epstot)` calculates the total emissivity of an isothermal gas mixture, using Leckner's model, equations (11.177) through (11.181).

Input:

`ph2o`	=	p_{H_2O}, partial pressure of water vapor, in bar,
`pco2`	=	p_{CO_2}, partial pressure of carbon dioxide, in bar,
`ptot`	=	p, total mixture pressure, in bar,
`Tg`	=	T_g, gas column temperature, in K,
`L`	=	L, gas column length, in m,

Output:

`epsh2o`	=	ϵ_{H_2O}, total emissivity of water vapor in the mixture,
`epsco2`	=	ϵ_{CO_2}, total emissivity of carbon dioxide in the mixture,
`epstot`	=	$\epsilon_{CO_2+H_2O}$, total emissivity of the mixture, considering overlap effects.

`totabsor.f`

Double precision Fortran77 subroutine `totabsor(ph2o,pco2,ptot,Tg,Tw,L,absh2o,absco2,abstot)` calculates the total absorptivity of an isothermal gas mixture, using Leckner's model, equations (11.177) through (11.181).

Input:

`ph2o`	=	p_{H_2O}, partial pressure of water vapor, in bar,
`pco2`	=	p_{CO_2}, partial pressure of carbon dioxide, in bar,
`ptot`	=	p, total mixture pressure, in bar,
`Tg`	=	T_g, gas column temperature, in K,
`Tw`	=	T_w, wall (or irradiation source) temperature, in K,
`L`	=	L, gas column length, in m,

Output:

`absh2o`	=	α_{H_2O}, total absorptivity of water vapor in the mixture,
`absco2`	=	α_{CO_2}, total absorptivity of carbon dioxide in the mixture,
`abstot`	=	$\alpha_{CO_2+H_2O}$, total absorptivity of the mixture, considering overlap effects.

Note: `totabsor` calls (i.e., requires) subroutine `totemiss`

`Leckner.f, Leckner.exe`

Stand-alone frontend for `totemiss(ph2o,pco2,ptot,Tg,L,epsh2o,epsco2,epstot)` and `totabsor (ph2o,pco2,ptot,Tg,Tw,L` User is prompted to input `ph2o`, `pco2`, `ptot`, `Tg`, `Tw` and `L` (see above), and the corresponding total emissivities and absorptivities are printed to the screen.

Chapter 12

`mmmie.f, mmmiea.f`

Fortran77 programs `mmmie` and `mmmiea` calculate Mie coefficients (scattering coefficients a_n and b_n, efficiencies

$Q_{\text{sca}}, Q_{\text{ext}}$ and Q_{abs}, see Section 12.2 for definitions), and relate them to particle cloud properties (extinction coefficient β, absorption coefficient κ, scattering coefficient σ_s, scattering phase function Φ for specified scattering angles. In addition, program `mmmiea` also calculates the asymmetry factor g, and phase function expansion coefficients A_n, as defined in Section 12.3), but at a severe penalty in cpu time.
The input for both programs is the same, and is done via a data file `MIE.DAT`:

Input:		
`IDSTF`	=	1: single particle size; =2: modified gamma distribution
`IETA`	=	1: single wavenumber; =2: wave number spectrum
`IPRNT`	=	1: print only final results; =2: also intermediate integrations
`CIR`	=	complex index of refraction
`RMIN`	=	minimum particle size in gamma distribution (in μm)
`RMAX`	=	maximum particle size in gamma distribution (in μm)
`AMG`, `BMG`, `ALMG`, `GAMG`	=	constants in gamma distribution, equation (12.34), `FR(R) = AMG*R**ALMG*DEXP(-BMG*R**GAMG);` units: `AMG` [cm$^{-3}\mu$m$^{\text{ALMG+1}}$], `ALMG` [-], `BMG` [μm^{-1}], `GAMG`[-]
`NPV`	=	number of particles per unit volume (in particles/cm^3)
`ETA`	=	wavenumber if single wavenumber is considered (in cm^{-1})
`ETMIN`	=	minimum wavenumber to be considered
`ETMAX`	=	maximum wavenumber to be considered
`NETA`	=	number of wavenumbers to be considered (equally spaced between `ETMIN` and `ETMAX`)
`ERRP`	=	maximum error allowed for absorption/scattering coefficients (and also the asymmetry factor for `mmmiea`)(in %)
`ERRA`	=	maximum absolute error desired for phase function values (`mmmie`) or expansion coefficients A_n (`mmmiea`) (in $10^{-\text{digits}}$)

Note: to allow running the program on machines with relatively little RAM, array sizes have been declared fairly small, limiting calculations to (i) a maximum of 10 different wavenumbers, (ii) relatively small size parameters ($x \lesssim 300$), and (iii) relatively coarse integration intervals (< 500 nodes). More involved problems can be calculated by carefully increasing array limits as indicated in the programs.

Example:

The input file `MIE.DAT` as given in this directory, contains the following data:

```
  2, 1, 2
  (1.30149,-0.1620E-05)
   1.-10   1.+10,
   1.619424-4, 0.740741, 7.6,  1., 74.
10000.
  1.    .005
```

stating that a gamma-distribution of particles is to be considered for a single wavenumber, with detailed output (including intermediate integrations) (first line).
The complex index of refraction of the particles is $m = 1.30149 - 0.1620 \times 10^{-05} i$ (second line).
Particle sizes range from 10^{-10} μm to 10^{+10} μm (third line).
Gamma-function parameters in equation (12.34) are $A = 1.61942410^{-4}, B = 0.740741, \gamma = 7.6, \delta = 1$. The number of particles is given as $N_(T) = 74/\text{cm}^3$ (this number is really not necessary for a gamma distribution, since it can be calculated from equation (12.35), and is only read and printed, but not used) (fourth line).
Since only a single wavenumber is considered, the fifth line contains only one number, $\eta = 100000\,\text{cm}^{-1}$.
Finally, the last line specifies to calculate κ, σ and β to an accuracy of 1% or better, and that the values for Φ or A_n should be calculated to an absolute accuracy of 0.005.

Running `mmmie` produces the following self-explanatory output, placed into file `MIE.RES`:

```
PARAMETERS FOR PARTICLE DISTRIBUTION/SINGLE WAVENUMBER
******************************************************

WAVENUMBER            = 0.100E+05 CM-1
MINIMUM PARTICLE RADIUS= 0.100E-09 MICROM
MAXIMUM PARTICLE RADIUS= 0.100E+11 MICROM
REFRACTIVE INDEX      = 1.3015-0.0000i
PARTICLE DENSITY      = 0.740E+02 PER CM**3
DISTRIBUTION FUNCTION: N(R)=0.16194E-03*R**7.6*EXP(-0.74074E+00*R**1.0)
```

```
   MIE-PARAMETERS ARE CALCULATED FOR  16.00000  < X < 216.00000


   INTEGRATION WITH   9 NODES, AND A DX =25.000


ETA (CM-1)      1.000E+04
KAPPA (CM-1)    1.250E-07
SIGMA (CM-1)    3.675E-04
BETA  (CM-1)    3.676E-04


PHASE FUNCTION
DEG.              PHI
  0             4.835E+03
  1             1.943E+03
  2             2.093E+02
  3             5.329E+01
.
.
.
176             2.264E-01
177             1.503E-01
178             2.086E-01
179             3.508E-01
180             1.364E+00

   INTEGRATION WITH  17 NODES, AND A DX =12.500

ETA (CM-1)      1.000E+04
KAPPA (CM-1)    9.997E-08
SIGMA (CM-1)    3.667E-04
BETA  (CM-1)    3.668E-04


PHASE FUNCTION
DEG.              PHI
  0             4.634E+03
  1             1.851E+03
  2             2.304E+02
  3             4.943E+01
.
.
.
177             2.428E-01
178             3.551E-01
179             3.691E-01
180             9.224E-01

   INTEGRATION WITH  65 NODES, AND A DX = 3.125

ETA (CM-1)      1.000E+04
KAPPA (CM-1)    1.023E-07
SIGMA (CM-1)    3.684E-04
BETA  (CM-1)    3.685E-04


PHASE FUNCTION
DEG.              PHI
  0             4.617E+03
  1             1.847E+03
  2             2.331E+02
  3             6.044E+01
  .
  .
  .
   INTEGRATION DID NOT CONVERGE: MAXIMUM ERROR = 0.18%
```

```
    ERROR FOR SIGMA : 0.18%, ERROR FOR BETA : 0.18%
    ERROR FOR
PHASE(  1): 2.84309
PHASE(  2): 2.45336
PHASE(  3): 1.56688
PHASE(  4): 0.47940
PHASE(  5): 0.23725
.
.
.
PHASE(179): 0.03003
PHASE(180): 0.05414
PHASE(181): 0.10000


 ETA (CM-1)      1.000E+04
 KAPPA (CM-1)    9.785E-08
 SIGMA (CM-1)    3.677E-04
 BETA  (CM-1)    3.678E-04


 PHASE FUNCTION
 DEG.              PHI
   0             4.614E+03
   1             1.845E+03
   2             2.347E+02
   3             6.092E+01
   4             3.153E+01
   5             2.034E+01
   6             1.511E+01
   7             1.234E+01
   8             1.066E+01
   9             9.560E+00
  .
  .
  .
 170             7.660E-02
 171             1.032E-01
 172             1.213E-01
 173             1.069E-01
 174             9.150E-02
 175             1.214E-01
 176             1.629E-01
 177             2.179E-01
 178             2.986E-01
 179             2.761E-01
 180             7.212E-01
```

Running `mmmiea`, on the other hand produces the following output, placed into file `MIEA.RES`:

```
    PARAMETERS FOR PARTICLE DISTRIBUTION/SINGLE WAVENUMBER
    ******************************************************

    WAVENUMBER             = 0.100E+05 CM-1
    MINIMUM PARTICLE RADIUS= 0.100E-09 MICROM
    MAXIMUM PARTICLE RADIUS= 0.100E+11 MICROM
    REFRACTIVE INDEX       = 1.30149-1.62000E-06i
    PARTICLE DENSITY       = 7.400E+01 PER CM**3
    DISTRIBUTION FUNCTION: N(R)=1.61942E-04*R**7.6*EXP(-0.74074E+00*R**1.0)



    MIE-PARAMETERS ARE CALCULATED FOR  16.00000  < X < 216.00000


INTEGRATION WITH   9 NODES, AND A DX =25.000

 ETA (CM-1)      1.000E+04
 KAPPA (CM-1)    1.250E-07
 SIGMA (CM-1)    3.675E-04
 BETA  (CM-1)    3.676E-04
 GCOS  ( -- )    8.691E-01
       A(  1)    2.60744
       A(  2)    4.02359
       A(  3)    4.85462
       A(  4)    5.53582
       A(  5)    6.29942
       A(  6)    6.88010
       A(  7)    7.63828
       A(  8)    8.43823
       A(  9)    9.15186
       .
       .
       .
       A(449)    0.00000
       A(450)    0.00000
       A(451)    0.00000
       A(452)    0.00000


INTEGRATION WITH  33 NODES, AND A DX = 6.250

 ETA (CM-1)      1.000E+04
 KAPPA (CM-1)    1.015E-07
 SIGMA (CM-1)    3.681E-04
 BETA  (CM-1)    3.682E-04
 GCOS  ( -- )    8.716E-01
       A(  1)    2.52586
       A(  2)    3.88357
       A(  3)    4.68158
       A(  4)    5.32619
       A(  5)    6.04063
       A(  6)    6.59512
       A(  7)    7.31633
       A(  8)    8.08751
       A(  9)    8.72379
       A( 10)    9.58797
          .
          .
          .
       A(449)    0.00000
       A(450)    0.00000
       A(451)    0.00000
       A(452)    0.00000


 PHASEFUNCTION
 DEG.              PHI
   0            4.260E+03
   5            1.758E+01
  10            8.615E+00
  15            5.157E+00
  20            4.088E+00
  25            3.059E+00
  30            2.206E+00
  35            1.287E+00
  40            1.089E+00
  45            6.978E-01
  50            7.122E-01
  55            3.592E-01
  60            2.251E-01
  65            1.581E-01
  70            1.343E-01
  75            9.730E-02
  80            8.906E-02
  85            6.900E-02
  90            5.605E-02
  95            4.968E-02
 100            5.518E-02
 105          5.099E-02
 110          4.992E-02
 115          5.291E-02
 120          5.204E-02
 125          8.062E-02
 130          5.287E-02
 135          2.674E-01
 140          2.485E-01
 145          1.552E-01
 150          1.190E-01
 155          1.194E-01
 160          1.216E-01
 165          1.328E-01
 170          1.030E-01
 175          1.690E-01
 180          9.319E-01
```

`coalash.f90, coalash.exe`
This Fortran90 program determines absorption and extinction coefficients κ^*, β^* for the Rayleigh limit, from the Buckius and Hwang [3] model, as well as from the Mengüç and Viskanta [4] model. The user is prompted to input the complex index of refraction n and k as well as the nondimensional size parameter x of the coal/ash particles; results are then printed to the screen.

Chapter 16

`P1sor.f90`, `P1sor.cpp`
Subroutine `P1sor` provides the solution to equation (16.38) with its boundary condition (16.49) for a two-dimensional (rectangular or axisymmetric cylinder) enclosure with reflecting walls and an absorbing, emitting, linear-anisotropically scattering medium.

Input:

`II`	=	Number of nodes in x-direction
`JJ`	=	Number of nodes in y- or r-direction
`KK`	=	`0` for rectangular, `KK=1` for cylindrical enclosure
`IRE`	=	Radiative equilibrium identifier; `IRE=0`: no equilibrium; `IRE=1`: radiative equilibrium
`L`	=	Length of enclosure (in cm)
`R`	=	Height (rectangle) or radius (cylinder) of enclosure (in cm)
`EPSX`	=	Wall emittances, `EPSX(1)` at `X=0, EPSX(2)` at `X=L`
`EPSR`	=	Wall emittances, `EPSR(1)` at `Y=0` (for rectangle only), `EPSY(2)` at `Y,r=R`
`SX`	=	Sources at x-direction walls: `SX(1,j=1,2,...JJ)` source at $x = 0$ for varying y/r-nodes `SX(2,j=1,2,...JJ)` source at $x = L$ for varying y/r-nodes (for a standard, gray application `SX` = $4\sigma T^4$, in W/cm^2)
`SR`	=	Sources at y, r-direction walls: `SR(1,i=1,2,...II)` source at $y = 0$ for varying x-nodes (for rectangle only) `SR(2,i=1,2,...II)` source at $y, r = R$ for varying x-nodes (for a standard, gray application `SR` = $4\sigma T^4$, in W/cm^2)
`KT`	=	Absorption coefficient for all internal nodes (in cm^{-1})
`ST`	=	Scattering coefficient for all internal nodes (in cm^{-1})
`A1`	=	Linear anisotropy factor for all internal nodes
`SS`	=	Sources for all internal nodes (in cm^{-1}) (for a standard, gray application `SS` = $4\sigma T^4$, in W/cm^2)

Output:

`G`	=	Incident radiation for all internal nodes, (in W/cm^2)
`QX`	=	Fluxes at x-direction walls: `QX(1,j=1,2,...JJ)` flux at $x = 0$ for varying y/r-nodes `QX(2,j=1,2,...JJ)` flux at $x = L$ for varying y/R-nodes (positive into positive x-direction, in W/cm^2)

QR = Fluxes at x-direction walls:
QR(1,i=1,2,...II) flux at $y = 0$ for varying x-nodes (for rectangle only)
QR(2,i=1,2,...II) flux at $y, r = R$ for varying x-nodes
(positive into positive r, y-direction, in W/cm^2)

Calculations can be done for a gray medium or, on a spectral basis, for a nongray medium. For a gray medium the user may either specify a temperature field (IRE=0) by supplying SS= $4n^2\sigma T^4$, or radiative equilibrium may be invoked (IRE=1), in which case the heat generation term SS= $\dot{Q}'''$ must be input. Note that radiative equilibrium is not possible on a spectral level.

Width L is broken up into II equally spaced nodes with spacing Δx = L/(II − 1); similarly height/radius R is broken up into JJ equally spaced nodes with spacing Δr = R/(JJ − 1).

For each of the II×JJ nodes each of the radiative properties (κ = KT, σ_s = ST, A_1 = A1) must be input, as well as the local radiative source SS (= $4\pi I_b$ if IRE=0, or = $\dot{Q}'''$ if IRE=1). In addition, for each surface an emittance must be specified [$\epsilon(x = 0)$ = EPSX(1), $\epsilon(x = L)$ = EPSX(2); $\epsilon(y = 0)$= EPSR(1) for rectangular enclosures only, and $\epsilon(r_{\rm or} y = R)$ = EPSR(2)], as well as radiation sources [$4\pi I_{bw}(x = 0)$ = SX(1), $4\pi I_{bw}(x = L)$ = SX(2); $4\pi I_{bw}(y = 0)$ = SR(1) for rectangular enclosures only, and $4\pi I_{bw}(r_{\rm or} y = R)$ = SR(2)]. Insulated boundaries can be treated by setting the emittance of that surface to zero. One-dimensional problems can be treated by setting two opposing emittances to zero; for better efficiency the number of nodes in the cross-direction should be set to one. Thus, EPSR(1) = EPSR(2) = 0 and JJ = 1 makes the problem a one-dimensional slab, while EPSX(1) = EPSX(2) = 0 and II = 1 makes a one-dimensional cylinder.

Upon return P1sor provides the solution array G (incident radiation G for all II×JJ nodes), as well as flux vectors QX (for radiative fluxes at the two surfaces $x = 0$ and $x = L$) and QY (radiative fluxes at $y = 0$ for a rectangle, and $r_{\rm or} y = R$). The solution is found by *successive over-relaxation*, with over-relaxation parameter OM, which is optimized by an implementation of algorithm 9-6.1 given in [5].

Code Details

For a two-dimensional problem equation (16.38) may be rewritten as

$$\begin{aligned}-\frac{1}{3}\frac{1}{r^k}\frac{\partial}{\partial r}\left(\frac{r^k}{\beta^*}\frac{\partial G}{\partial r}\right)+\frac{\partial}{\partial x}\left(\frac{1}{\beta^*}\frac{\partial G}{\partial x}\right) &= \kappa(4\pi I_b - G) \text{ temperature specified,}\\ &= \dot{Q}''' \text{ radiative equilibrium,}\end{aligned} \tag{CC-8}$$

where $\beta^* = \beta - A_1\sigma_s/3$; KK = 0 makes it a rectangular enclosure, and KK = 1 makes it an axisymmetric cylinder. Standard central finite differencing with equal spacing Δr = R/(JJ − 1) and Δx = L/(II − 1) and $\lambda = \Delta x/\Delta r$ produces an equation for each (internal and boundary) node:

$$A_{ij}G_{i-1,j} + B_{ij}G_{i+1,j} + C_{ij}G_{i,j-1} + D_{ij}G_{i,j+1} - E_{ij}G_{ij} = -F_{ij}, \tag{CC-9}$$

where

$$\begin{aligned}
A_{ij} &= \frac{\beta^*_{ij}}{\beta^*_{i-1/2,j}} \simeq \frac{2\beta^*_{ij}}{\beta^*_{i-1,j}+\beta^*_{ij}}\\
B_{ij} &= \frac{\beta^*_{ij}}{\beta^*_{i+1/2,j}} \simeq \frac{2\beta^*_{ij}}{\beta^*_{ij}+\beta^*_{i+1,j}}\\
C_{ij} &= \lambda^2\frac{\beta^*_{ij}}{\beta^*_{i,j-1/2}}\left(\frac{r_{j-1/2}}{r_j}\right)^k \simeq \lambda^2\frac{2\beta^*_{ij}}{\beta^*_{i,j-1}+\beta^*_{ij}}\left(1-\frac{1}{2(j-1)}\right) \text{ since } r_j = (j-1)\Delta r\\
D_{ij} &= \lambda^2\frac{\beta^*_{ij}}{\beta^*_{i,j+1/2}}\left(\frac{r_{j+1/2}}{r_j}\right)^k \simeq \lambda^2\frac{2\beta^*_{ij}}{\beta^*_{ij}+\beta^*_{i,j+1}}\left(1+\frac{1}{2(j-1)}\right)\\
E_{ij} &= \begin{cases} 3\kappa_{ij}\beta^*_{ij}\Delta x^2 + A_{ij} + B_{ij} + C_{ij} + D_{ij} & \text{temperature specified,}\\ A_{ij} + B_{ij} + C_{ij} + D_{ij} & \text{radiative equilibrium,}\end{cases}\\
F_{ij} &= \begin{cases} 3\kappa_{ij}\beta^*_{ij}\Delta x^2 \mathtt{SS}_{ij} & \text{temperature specified } (\mathtt{SS}_{ij} = 4\pi I_{bij}),\\ 3\beta^*_{ij}\Delta x^2 \mathtt{SS}_{ij} & \text{radiative equilibrium } (\mathtt{SS}_{ij} = \dot{Q}'''_{ij}).\end{cases}
\end{aligned}$$

Boundary conditions equation (16.49) are written as, and finite-differenced using artificial nodes ($i = 0$ at $x = 0$, $i =$ `II` at $x = L$, $j = 0$ at $r = 0$ and $j =$ `JJ` at $r = R$)

$$\begin{aligned}
x = 0: &\quad \frac{\partial G}{\partial x} - \mathtt{bx}(1)\beta^* \left[G - \mathtt{SX}(1)\right] = 0 \quad \text{where} \quad \mathtt{bx}() = \frac{3}{2}\frac{\epsilon}{2-\epsilon}, \quad \mathtt{SX}() = 4\pi I_{bw} \\
x = L: &\quad \frac{\partial G}{\partial x} + \mathtt{bx}(2)\beta^* \left[G - \mathtt{SX}(2)\right] = 0 \\
r = 0: &\quad \frac{\partial G}{\partial r} - \mathtt{br}(1)\beta^* \left[G - \mathtt{SR}(1)\right] = 0 \quad \text{(rectangular enclosure, } \mathtt{KK} = \mathtt{0}\text{, only)} \\
r = R: &\quad \frac{\partial G}{\partial r} - \mathtt{br}(2)\beta^* \left[G - \mathtt{SR}(2)\right] = 0
\end{aligned}$$

or, with $\beta^* = \mathtt{BT}$

$$\begin{aligned}
x = 0\ (i = 1): &\quad G_{i-1,j} - G_{i+1,j} + 2\mathtt{bx}(1)\,\mathtt{\Delta x}\,\mathtt{BT}_{ij}\left(G_{ij} - \mathtt{SX}_j(1)\right) = 0 \\
x = L\ (i = \mathtt{II}): &\quad G_{i+1,j} - G_{i-1,j} + 2\mathtt{bx}(2)\,\mathtt{\Delta x}\,\mathtt{BT}_{ij}\left(G_{ij} - \mathtt{SX}_j(2)\right) = 0 \\
r = 0\ (j = 1): &\quad G_{i,j-1} - G_{i,j+1} + 2\mathtt{br}(1)\,\mathtt{\Delta r}\,\mathtt{BT}_{ij}\left(G_{ij} - \mathtt{SR}_i(1)\right) = 0 \quad (\mathtt{KK} = \mathtt{0}\text{ only}) \\
r = R\ (j = \mathtt{JJ}): &\quad G_{i,j+1} - G_{i,j-1} + 2\mathtt{br}(2)\,\mathtt{\Delta r}\,\mathtt{BT}_{ij}\left(G_{ij} - \mathtt{SR}_i(2)\right) = 0
\end{aligned}$$

Eliminating the artificial nodes between internal node and boundary node equations yields the updated values

$$\begin{aligned}
i = 1: &\quad A'_{ij} = 0, B'_{ij} = A_{ij} + B_{ij}, E'_{ij} = E_{ij} + 2\mathtt{bx}(1)\,\mathtt{\Delta x}\,\mathtt{BT}_{ij}A_{ij} \\
&\quad F'_{ij} = F_{ij} + 2\mathtt{bx}(1)\,\mathtt{\Delta x}\,\mathtt{BT}_{ij}A_{ij}\mathtt{SX}_j(1) \\
i = \mathtt{II}: &\quad B'_{ij} = 0, A'_{ij} = A_{ij} + B_{ij};\ E'_{ij} = E_{ij} + 2\mathtt{bx}(2)\,\mathtt{\Delta x}\,\mathtt{BT}_{ij}B_{ij} \\
&\quad F'_{ij} = F_{ij} + 2\mathtt{bx}(2)\,\mathtt{\Delta x}\,\mathtt{BT}_{ij}B_{ij}\mathtt{SX}_j(2) \\
j = 1: &\quad C'_{ij} = 0, D'_{ij} = C_{ij} + D_{ij}, E'_{ij} = E_{ij} + 2\mathtt{br}(1)\,\mathtt{\Delta r}\,\mathtt{BT}_{ij}C_{ij} \\
&\quad F'_{ij} = F_{ij} + 2\mathtt{br}(1)\,\mathtt{\Delta r}\,\mathtt{BT}_{ij}C_{ij}\mathtt{SR}_j(1) \\
j = \mathtt{JJ}: &\quad D'_{ij} = 0, C'_{ij} = C_{ij} + D_{ij}, E'_{ij} = E_{ij} + 2\mathtt{br}(2)\,\mathtt{\Delta r}\,\mathtt{BT}_{ij}D_{ij} \\
&\quad F'_{ij} = F_{ij} + 2\mathtt{br}(2)\,\mathtt{\Delta r}\,\mathtt{BT}_{ij}D_{ij}\mathtt{SR}_j(2)
\end{aligned}$$

For a cylindrical enclosure (`KK` = 1) the boundary condition at $r = 0$ (`J` = 1) becomes

$$r = 0, (j = 1): \frac{\partial G}{\partial r} = 0 \quad \text{or} \quad G_{i,j-1} = G_{i,j+1}.$$

Also, the governing equation (CC-8) becomes indeterminate. Expanding the radial derivative and using De l'Hopital's rule, we obtain

$$\begin{aligned}
\lim_{r\to 0}\frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r}{\beta^*}\frac{\partial G}{\partial r}\right) &= \frac{1}{\beta^*}\frac{\partial^2 G}{\partial r^2} - \frac{1}{\beta^{*2}}\frac{\partial G}{\partial r}\frac{\partial \beta^*}{\partial r} + \lim_{r\to 0}\frac{1}{r\beta^*}\frac{\partial G}{\partial r} = \frac{2}{\beta^*}\frac{\partial^2 G}{\partial r^2} \\
&= \frac{4}{\beta_{i1}\Delta r^2}(G_{i2} - G_{i1})
\end{aligned}$$

Thus, for `KK` = 1 and `J` = 1

$$C_{ij} = 0, \quad D_{ij} = 4\lambda^2$$

`P1-2D.f90`, `P1-2D.cpp`

Program `P1-2D` is a front end for subroutine `P1sor`, setting up the problem for a gray medium with spatially constant radiative properties (dimensions, radiative properties, and sources from known temperatures); may be used as a starting point for more involved applications. After calling `P1sor` the program also generates appropriate output. As given, `P1-2D` simulates the case of a two-dimensional axisymmetric cylinder (`KK=1`) of $R = 10\,\mathrm{cm}$ radius and $L = 20\,\mathrm{cm}$ length, using `JJ=21` nodes in the radial direction and `II=41` nodes in the axial direction (i.e., $\Delta x = \Delta r = 0.5\,\mathrm{cm}$), with a cold ($T_{ij}$ = `TM` = 0) gray medium, with constant absorption and scattering coefficients ($\kappa = \sigma_s = 0.1\,\mathrm{cm}^{-1}, A_1 = 0$); bounding walls are black and cold except for the face at $x = 0$, which is gray (`EPSX(1)=0.5`) and hot (`TX(1)=2000` K). Since the temperature field is specified, we have `IRE=0`. Running `P1-2D` we find from screen output that the calculation requires 97 iterations with a residual 2-norm error of 0.1354×10^{-4}.

The output is in file `P1-2Dsor.dat`, giving:

```
        GENERAL DATA
        ************

        CYLINDER RADIUS (R-DIR):    10.00
        CYLINDER LENGTH (X-DIR):    20.00
        TEMPERATURE AT r=R(j=J):     0.00K,  EMITTANCE  1.00
        TEMPERATURE AT x=0(i=1):  2000.00K,  EMITTANCE  0.50
        TEMPERATURE AT x=L(i=I):     0.00K,  EMITTANCE  1.00

    MEDIUM TEMPERATURE TM (K)

   \J  1     3     5     7     9    11    13    15    17    19    21
  I
  1    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
  3    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
  5    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
  7    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
  9    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 11    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 13    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 15    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 17    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 19    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 21    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 23    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 25    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 27    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 29    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 31    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 33    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 35    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 37    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 39    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
 41    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.

    INCIDENT RADIATION G (W/SQCM)

   \J  1     3     5     7     9    11    13    15    17    19    21
  I
  1  99.6  99.4  99.0  98.3  97.1  95.5  93.1  89.7  84.7  77.0  64.0
  3  76.3  76.2  75.7  75.0  73.7  72.0  69.5  65.9  60.9  53.5  42.7
  5  58.3  58.1  57.7  56.9  55.7  54.0  51.6  48.3  43.7  37.6  29.6
  7  44.3  44.1  43.7  43.0  41.9  40.3  38.2  35.3  31.5  26.7  20.9
  9  33.5  33.4  33.0  32.4  31.4  30.0  28.2  25.8  22.8  19.2  14.9
 11  25.3  25.2  24.8  24.3  23.4  22.3  20.8  18.9  16.6  13.8  10.8
 13  19.0  18.9  18.6  18.2  17.5  16.5  15.3  13.8  12.1  10.0   7.8
 15  14.2  14.2  13.9  13.5  13.0  12.2  11.3  10.2   8.8   7.3   5.7
 17  10.6  10.6  10.4  10.1   9.6   9.1   8.3   7.5   6.5   5.3   4.1
 19   7.9   7.9   7.7   7.5   7.2   6.7   6.1   5.5   4.7   3.9   3.0
 21   5.9   5.9   5.8   5.6   5.3   5.0   4.5   4.0   3.5   2.9   2.2
 23   4.4   4.4   4.3   4.1   3.9   3.7   3.3   3.0   2.6   2.1   1.6
 25   3.3   3.2   3.2   3.1   2.9   2.7   2.5   2.2   1.9   1.5   1.2
 27   2.4   2.4   2.4   2.3   2.1   2.0   1.8   1.6   1.4   1.1   0.9
 29   1.8   1.8   1.7   1.7   1.6   1.5   1.3   1.2   1.0   0.8   0.6
 31   1.3   1.3   1.3   1.2   1.2   1.1   1.0   0.9   0.7   0.6   0.5
 33   1.0   1.0   1.0   0.9   0.9   0.8   0.7   0.6   0.6   0.5   0.3
 35   0.7   0.7   0.7   0.7   0.6   0.6   0.5   0.5   0.4   0.3   0.3
 37   0.5   0.5   0.5   0.5   0.5   0.4   0.4   0.4   0.3   0.2   0.2
 39   0.4   0.4   0.4   0.4   0.4   0.3   0.3   0.3   0.2   0.2   0.1
 41   0.3   0.3   0.3   0.3   0.3   0.2   0.2   0.2   0.2   0.1   0.1
```

```
    WALL FLUXES AT X=O AND X=L (W/SQCM)

    J    1     3     5     7     9    11    13    15    17    19    21

    Q  43.9  43.9  44.0  44.1  44.3  44.6  45.0  45.5  46.4  47.6  49.8
    Q   0.1   0.1   0.1   0.1   0.1   0.1   0.1   0.1   0.1   0.1   0.1


    RADIAL FLUXES TO CYLINDER WALL (W/SQCM)

         I     QR

         1    32.0
         2    25.9
         3    21.3
         .
         .
         .
```

Had we defined `IRE=1` the same case would be calculated, but for radiative equilibrium with $\dot{Q}''' = 0$ (since `TM` was set to zero). This results in (now requiring 137 iterations):

```
         GENERAL DATA
         ************

         CYLINDER RADIUS (R-DIR):    10.00
         CYLINDER LENGTH (X-DIR):    20.00
         TEMPERATURE AT r=R(j=J):     0.00K,  EMITTANCE  1.00
         TEMPERATURE AT x=0(i=1):  2000.00K,  EMITTANCE  0.50
         TEMPERATURE AT x=L(i=I):     0.00K,  EMITTANCE  1.00

    MEDIUM TEMPERATURE TM (K)

   \J   1     3     5     7     9    11    13    15    17    19    21
  I
  1 1611. 1610. 1606. 1600. 1592. 1579. 1563. 1540. 1510. 1466. 1393.
  3 1555. 1554. 1550. 1542. 1532. 1517. 1497. 1470. 1433. 1381. 1302.
  5 1499. 1497. 1493. 1484. 1472. 1455. 1432. 1402. 1361. 1306. 1228.
  7 1442. 1441. 1435. 1426. 1413. 1394. 1370. 1337. 1295. 1238. 1163.
  9 1386. 1384. 1379. 1369. 1355. 1335. 1309. 1276. 1233. 1177. 1105.
 11 1331. 1329. 1323. 1313. 1298. 1278. 1251. 1218. 1175. 1121. 1051.
 13 1277. 1275. 1268. 1258. 1243. 1223. 1196. 1163. 1121. 1068. 1002.
 15 1224. 1222. 1215. 1205. 1190. 1169. 1143. 1110. 1069. 1019.  955.
 17 1172. 1170. 1164. 1153. 1138. 1118. 1093. 1061. 1021.  972.  911.
 19 1122. 1120. 1114. 1104. 1089. 1069. 1044. 1013.  975.  928.  870.
 21 1074. 1072. 1066. 1056. 1041. 1022.  998.  968.  931.  886.  830.
 23 1027. 1025. 1019. 1009.  995.  977.  953.  924.  889.  846.  793.
 25  982.  980.  974.  965.  951.  933.  911.  883.  849.  807.  757.
 27  938.  936.  930.  921.  908.  891.  869.  843.  810.  771.  722.
 29  895.  893.  888.  879.  867.  850.  829.  804.  773.  735.  689.
 31  853.  852.  847.  838.  826.  810.  790.  766.  736.  700.  656.
 33  812.  811.  806.  798.  786.  771.  752.  729.  701.  666.  624.
 35  772.  770.  765.  758.  747.  732.  714.  692.  665.  633.  593.
 37  730.  729.  725.  717.  707.  693.  676.  655.  630.  599.  561.
 39  688.  686.  682.  675.  665.  653.  636.  617.  593.  564.  528.
 41  642.  641.  637.  630.  621.  609.  594.  576.  553.  526.  493.

    INCIDENT RADIATION G (W/SQCM)

   \J   1     3     5     7     9    11    13    15    17    19    21
  I
  1 152.8 152.4 151.1 148.8 145.5 141.1 135.2 127.6 117.8 104.7  85.5
  3 132.7 132.2 130.8 128.4 124.9 120.1 113.9 105.9  95.6  82.4  65.1
  5 114.5 114.0 112.5 110.1 106.5 101.7  95.5  87.6  77.9  65.9  51.5
  7  98.2  97.7  96.3  93.8  90.3  85.7  79.8  72.5  63.7  53.3  41.5
  9  83.8  83.3  81.9  79.6  76.4  72.1  66.7  60.1  52.4  43.6  33.8
 11  71.2  70.7  69.5  67.4  64.4  60.5  55.6  49.9  43.2  35.8  27.7
 13  60.2  59.9  58.7  56.8  54.1  50.7  46.4  41.5  35.8  29.5  22.8
 15  50.8  50.5  49.5  47.8  45.4  42.4  38.7  34.5  29.7  24.4  18.9
 17  42.8  42.5  41.6  40.1  38.1  35.5  32.3  28.7  24.6  20.3  15.6
 19  36.0  35.7  34.9  33.7  31.9  29.6  27.0  23.9  20.5  16.8  13.0
 21  30.1  29.9  29.3  28.2  26.7  24.8  22.5  19.9  17.0  14.0  10.8
 23  25.2  25.0  24.5  23.5  22.3  20.6  18.7  16.6  14.2  11.6   9.0
 25  21.1  20.9  20.4  19.6  18.6  17.2  15.6  13.8  11.8   9.6   7.4
 27  17.5  17.4  17.0  16.3  15.4  14.3  13.0  11.4   9.8   8.0   6.2
 29  14.6  14.4  14.1  13.6  12.8  11.8  10.7   9.5   8.1   6.6   5.1
 31  12.0  11.9  11.7  11.2  10.6   9.8   8.9   7.8   6.7   5.5   4.2
 33   9.9   9.8   9.6   9.2   8.7   8.0   7.3   6.4   5.5   4.5   3.4
 35   8.0   8.0   7.8   7.5   7.1   6.5   5.9   5.2   4.4   3.6   2.8
 37   6.5   6.4   6.2   6.0   5.7   5.2   4.7   4.2   3.6   2.9   2.2
 39   5.1   5.0   4.9   4.7   4.4   4.1   3.7   3.3   2.8   2.3   1.8
 41   3.8   3.8   3.7   3.6   3.4   3.1   2.8   2.5   2.1   1.7   1.3


    WALL FLUXES AT X=O AND X=L (W/SQCM)

    J    1     3     5     7     9    11    13    15    17    19    21

    Q  35.0  35.1  35.3  35.7  36.2  37.0  37.9  39.2  40.8  43.0  46.2
    Q   1.9   1.9   1.9   1.8   1.7   1.6   1.4   1.2   1.1   0.9   0.7


    RADIAL FLUXES TO CYLINDER WALL (W/SQCM)

         I     QR

         1    42.7
         2    37.0
         .
         .
         .
```

Finally, if we set `IRE=1`, `EPSR=0` and `JJ=1`, we obtain the results for a one-dimensional slab at radiative equilibrium:

```
         GENERAL DATA
         ************

         CYLINDER RADIUS (R-DIR):    10.00
         CYLINDER LENGTH (X-DIR):    20.00
         TEMPERATURE AT r=R(j=J):     0.00K,  EMITTANCE  0.00
         TEMPERATURE AT x=0(i=1):  2000.00K,  EMITTANCE  0.50
         TEMPERATURE AT x=L(i=I):     0.00K,  EMITTANCE  1.00

    MEDIUM TEMPERATURE TM (K)

   \J  1
  1 1829.
  3 1809.
  5 1788.
  7 1767.
  9 1745.
 11 1722.
 13 1698.
 15 1673.
 17 1646.
 19 1619.
 21 1590.
 23 1559.
 25 1527.
 27 1492.
 29 1454.
 31 1414.
 33 1369.
 35 1320.
 37 1264.
 39 1201.
 41 1124.

    INCIDENT RADIATION G (W/SQCM)
   \J  1
  1 253.7
  3 242.8
  5 232.0
  7 221.1
  9 210.2
 11 199.3
 13 188.4
 15 177.5
 17 166.7
 19 155.8
 21 144.9
 23 134.1
 25 123.2
 27 112.3
 29 101.5
 31  90.6
 33  79.7
 35  68.9
 37  58.0
 39  47.1
 41  36.2


    WALL FLUXES AT X=O AND X=L (W/SQCM)

    J    1
    Q  18.2
    Q  18.1


    RADIAL FLUXES TO CYLINDER WALL (W/SQCM)

         I     QR

         1     0.0
         2     0.0
         .
         .
         .
```

Of course, the matrix for this case could have easily been inverted by a tridiagonal matrix solver (instead of using 181 iterations as done here), or could have been found analytically using Example 15.5 (but for a gray wall).

`Delta.f90`:
Program `Delta` is a stand-alone program to calculate the rotation matrix $\Delta^n_{mm'}(\alpha,\beta,\gamma)$ required for the boundary conditions of higher-order P_N-approximations, as given by equations (16.64) through (16.67); here set for $2l = N-1 = 4$ (P_5). Results for the case of a backward rotation with $-\gamma(=$ `alpha`$) = -\pi/2$, $-\beta(=$ `beta`$) = \pi/2$, $-\alpha(=$ `gamma`$) = \pi/2$ (a surface at $y = \text{const}$ facing toward larger y, with $\bar{x} = x$) are calculated and stored in `delta.dat`. For incorporation into a general P_N-code the stand-alone program can easily be converted into a subroutine calculating a single or all rotation Δ-values for a given set of angles α,β,γ.

`pnbcs.f90`:
Program `pnbcs` is a stand-alone program to calculate the Legendre half-moments $p^m_{n,j}$ and coefficients $u^m_{li}, v^m_{li}, w^m_{li}$, which are required for the boundary conditions of higher-order P_N-approximations, as given by equations (16.71) through (16.72). Calculations use the recursion relationships described in [6], Eqs. (27) through (32). As provided, $N =$ `NN` $= 5$, i.e., the $p^m_{n,j}$, u^m_{li}, v^m_{li} and w^m_{li} are calculated up to $n = 5$ (P_5-approximation). Output is directed to `PNbc.dat`, containing all the $p^m_{n,j}$ data for Table 16.2 (i.e., normalized by 10^{-m}), and the corresponding u, v, w. Higher orders may be implemented by changing `NN` (however, output format would need adjustment beyond P_{19}).

Chapter 19

transPN.f90

Program `transPN` calculates energy from a pulsed collimated laser source transmitted through an absorbing, isotropically scattering slab as a function of time, using the P_1 and $P_{1/3}$ methods. Following Example 19.3 the equations for the P_1- and $P_{1/3}$-approximations for a nonemitting and isotropically scattering, one-dimensional medium, reduce to

$$\frac{\partial G}{\partial t^*} + \frac{\partial q}{\partial \tau} = -(1-\omega)G + \omega G_c, \tag{CC-10}$$

$$3a\frac{\partial q}{\partial t^*} + \frac{\partial G}{\partial \tau} = -3q, \tag{CC-11}$$

where $a = 1$ for P_1 and $a = 1/3$ for $P_{1/3}$, and G and q have been normalized as $G = G_d/q_o$ and $q = q_d/q_o$. These two equations are subject to the initial and boundary conditions

$$t^* = 0: \qquad G(0,\tau) = q(0,\tau) = 0, \tag{CC-12}$$

$$\tau = 0: \quad -2q(t^*,0) = G(t^*,0), \tag{CC-13}$$

$$\tau = \tau_L: \quad +2q(t^*,\tau_L) = G(t^*,\tau_L). \tag{CC-14}$$

The normalized isotropic scattering source is immediately found from equations (19.25) and (19.18) for a nonreflecting boundary. For the top-hat profile of Example 19.3 this results in a total nondimensional pulse energy of t_p^* and

$$G_c(t^*,\tau) = \left[H(t^*-\tau) - H^*(t^*-\tau-t_p^*)\right]e^{-\tau}. \tag{CC-15}$$

If a clipped Gaussian source is used [7], then

$$q_0(0,t) = q_0\left[H(t) - H(t-2t_c)\right]\exp\left[-\left(\frac{t-t_c}{t_p}\right)^2\right], \tag{CC-16}$$

and the total nondimensional pulse energy is

$$\begin{aligned}\int_0^\infty \frac{q_0(0,t)}{q_0}\beta c\,dt &= \int_0^\infty \left[H(t^*) - H(t^*-2t_c^*)\right]\exp\left[-\left(\frac{t^*-t_c^*}{t_p^*}\right)^2\right]dt^* \\ &= \int_0^{2t_c^*} \exp\left[-\left(\frac{t^*-t_c^*}{t_p^*}\right)^2\right]dt^* = \sqrt{\pi}t_p^*\,\mathrm{erf}\left(\frac{t_c^*}{t_p^*}\right).\end{aligned} \tag{CC-17}$$

Thus, to run `transPN` with equal pulse strengths, one must use

$$t_{p,\mathrm{TH}}^* = \sqrt{\pi}\,\mathrm{erf}\left(\frac{t_c^*}{t_{p\mathrm{G}}^*}\right)t_{p\mathrm{G}}^* \simeq \sqrt{\pi}t_{p\mathrm{G}}^*, \tag{CC-18}$$

the latter assuming $t_c \gtrsim 2t_{p\mathrm{G}}$. For the clipped Gaussian pulse the source term then becomes

$$G_c(t^*,\tau) = \left[H(t^*-\tau) - H(t^*-2t_c^*-\tau)\right]\exp\left[-\tau-\left(\frac{t^*-t_c^*-\tau}{t_p^*}\right)^2\right]. \tag{CC-19}$$

The hyperbolic nature of this set of equations becomes obvious, if q is eliminated from them (by differentiating the first with respect to t^* and the second with respect to τ), leading to

$$\frac{\partial^2 G}{\partial t^{*2}} - \frac{1}{3a}\frac{\partial^2 G}{\partial \tau^2} + \left(1-\omega+\frac{1}{a}\right)\frac{\partial G}{\partial t^*} + \frac{1-\omega}{a}G - \frac{\omega}{a}G_c - \omega\frac{\partial G_c}{\partial t^*} = 0, \tag{CC-20}$$

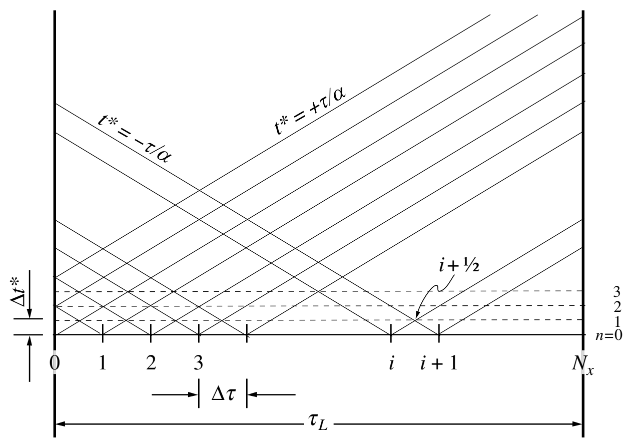


FIGURE 1
Time-space nodal system for `transPN.f90`.

which has a signal velocity of $\alpha = 1/\sqrt{3a}$ (nondimensional in terms of speed of light, c), as already indicated in the formulation for the P_a methods. Eliminating q also from initial and boundary conditions gives

$$t^* = 0: \quad G(0,\tau) = \frac{\partial G}{\partial t^*}(0,\tau) = 0, \tag{CC-21}$$

$$\tau = 0: \quad 3\left(G(t^*,0) + a\frac{\partial G}{\partial t^*}(t^*,0)\right) - 2\frac{\partial G}{\partial \tau}(t^*,0) = 0, \tag{CC-22}$$

$$\tau = \tau_L: \quad 3\left(G(t^*,0) + a\frac{\partial G}{\partial t^*}(t^*,0)\right) + 2\frac{\partial G}{\partial \tau}(t^*,0) = 0. \tag{CC-23}$$

This second-order hyperbolic equation is readily solved by the method of characteristics [8] along the characteristic lines $\tau = \pm\alpha t^*$. Using subscript notation, i.e., $G_x = \partial G/\partial\tau$, etc., equation (CC-20) may be rewritten as

$$G_{tt} - \alpha^2 G_{xx} + (1-\omega)G_t + 3\alpha^2\left[G_t + (1-\omega)G - \omega G_c'\right] = 0, \tag{CC-24}$$

where $G_c' = G_c + \partial G_c/\partial t^*$. Along the two characteristic lines $\tau = \pm\alpha t^*$ we have [8]

$$\pm\alpha dG_t - \alpha^2 dG_x \pm \left\{(1-\omega)G_t + 3\alpha^2\left[G_t + (1-\omega)G - \omega G_c'\right]\right\}d\tau = 0 \tag{CC-25}$$

and the total differential is

$$dG = G_t dt^* + G_x d\tau. \tag{CC-26}$$

We will break up the thickness of the slab, L, into N_x equally-spaced nodes of width $\Delta x = L/N_x$, or $\Delta\tau = \tau_L/N_x$. In t^*-τ-space the characteristics then are straight lines as shown in Fig. 1Time-space nodal system for `transPN.f90`figure.0.1, with the lines going up to the right corresponding to the upper sign in equation (CC-25), and the lines going down to the right to the lower sign. As time step $\Delta\tau$ we take the time it takes to move along the characteristics from adjacent points (n,i) and $(n,i+1)$ to their intersection at $(n+1, i+{}^1\!/\!_2)$ as shown in the figure. During that time the signal moves a distance $\pm\Delta x/2$, so that

$$\Delta t^* = \Delta\tau/2\alpha. \tag{CC-27}$$

We can finite-difference equations (CC-25) and (CC-26) along the characteristics by using $d\phi = \phi^n_{i+1/2} - \phi^{n-1}_i$ for the left-to-right characteristics, and $d\phi = \phi^n_{i+1/2} - \phi^{n-1}_{i+1}$ for the right-to-left characteristics, where ϕ stands for any of the variables τ, G, G_t and G_x. In the finite differencing we distinguish between odd time steps (all nodes, such as $i + {}^1\!/\!_2$, are internal) and even time steps (all nodes are at integer locations, including two boundary nodes $i = 0$ and $i = N_x$).

Odd Time Steps (n odd) All new positions are at $i + 1/2$ ($i = 0, 1 \dots N_x - 1$); all old positions are at i ($i = 0, N_x - 1$) for the left-to-right characteristics, and at $i + 1$ ($i + 1 = 1, N_x$) for the right-to-left characteristics. Thus,

$$\alpha(G_{t,i+1/2} - G_{t,i}) - \alpha^2(G_{x,i+1/2} - G_{x,i}) + \Big\{(1-\omega)(G_{t,i+1/2} + G_{t,i}) + 3\alpha^2\left[G_{t,i+1/2} + G_{t,i} + (1-\omega)(G_{i+1/2} + G_i) - \omega(G'_{c,i+1/2} + G'_{c,i})\right]\Big\}\frac{\Delta\tau}{4} = 0, \quad \text{(CC-28)}$$

where we have used averaged values, $\phi = \frac{1}{2}(\phi^n_{i+1/2} + \phi^{n-1}_i)$ for the terms within braces, and have omitted the time superscripts, since the distinction between new and old is clear. Bringing all unknown quantities at the new time to the left-hand side we get

$$B_p G_{t,i+1/2} - C_4 G_{x,i+1/2} + C_2 G_{i+1/2} = -B_m G_{t,i} - C_4 G_{x,i} - C_2 G_i + C_3(G'_{c,i+1/2} + G'_{c,i}) = E_1, \quad i = 0, N_x - 1, \quad \text{(CC-29)}$$

where

$$B_p = \alpha + (1 - \omega + 3\alpha^2)\frac{\Delta\tau}{4}, \quad B_m = \alpha - (1 - \omega + 3\alpha^2)\frac{\Delta\tau}{4},$$
$$C_2 = 3\alpha^2(1-\omega)\frac{\Delta\tau}{4}, \quad C_3 = 3\alpha^2\omega\frac{\Delta\tau}{4}, \quad C_4 = \alpha^2. \quad \text{(CC-30)}$$

Similarly, we obtain for the right-to-left characteristics, by switching the signs in equation (CC-25) and replacing i by $i + 1$:

$$B_p G_{t,i+1/2} + C_4 G_{x,i+1/2} + C_2 G_{i+1/2} = -B_m G_{t,i+1} + C_4 G_{x,i+1} - C_2 G_{i+1} + C_3(G'_{c,i+1/2} + G'_{c,i+1}) = E_2, \quad i = 0, N_x - 1. \quad \text{(CC-31)}$$

We now have two equations in the three unknowns $G_{t,i+1/2}$, $G_{x,i+1/2}$ and $G_{i+1/2}$: one more relation is needed and will come from equation (CC-26), which may be finite-differenced from the left or from the right as

$$\begin{aligned} G_{i+1/2} &= G_i + \frac{1}{2}(G_{t,i+1/2} + G_{t,i})\Delta t^* + \frac{1}{2}(G_{x,i+1/2} + G_{x,i})\frac{\Delta\tau}{2}, && l \to r \\ &= G_{i+1} + \frac{1}{2}(G_{t,i+1/2} + G_{t,i+1})\Delta t^* - \frac{1}{2}(G_{x,i+1/2} + G_{x,i+1})\frac{\Delta\tau}{2}, && r \to l. \end{aligned} \quad \text{(CC-32)}$$

For better accuracy, we take the average, or

$$-\frac{\Delta t^*}{2}G_{t,i+1/2} + G_{i+1/2} = \frac{1}{2}(G_i + G_{i+1}) + \frac{\Delta t^*}{4}(G_{t,i} + G_{t,i+1}) + \frac{\Delta\tau}{8}(G_{x,i} - G_{x,i+1}) = D_2. \quad \text{(CC-33)}$$

Subtracting equation (CC-29) from (CC-31) leads to

$$G_{x,i+1/2} = (E_2 - E_1)/2C_4, \quad i = 0, N_x - 1, \quad \text{(CC-34)}$$

while adding them gives

$$B_p G_{t,i+1/2} + C_2 G_{i+1/2} = \frac{1}{2}(E_1 + E_2) = D_1, \quad \text{(CC-35)}$$

which, together with equation (CC-33) leads to

$$G_{i+1/2} = \frac{D_1 \Delta t^*/2 + D_2 B_p}{C_2 \Delta t^*/2 + B_p}, \quad G_{t,i+1/2} = \frac{D_1 - C_2 D_2}{C_2 \Delta t^*/2 + B_p}, \quad i = 0, N_x - 1.$$

Even Time Steps (n even) Even time steps are a little more difficult to handle, because two of the nodes lie on the boundaries, and for them the boundary conditions must be invoked. Internal nodes, on the other hand, are the same as those for odd n, except that nodes are displaced by half a node. Replacing every i by $i - 1/2$ we obtain

$$G_{x,i} = (E_2 - E_1)/2C_4, \quad G_{t,i} = \frac{D_1 - C_2 D_2}{C_2 \Delta t^*/2 + B_p},$$
$$G_i = \frac{D_1 \Delta t^*/2 + D_2 B_p}{C_2 \Delta t^*/2 + B_p}; \qquad i = 1, N_x - 1, \tag{CC-36}$$

where

$$\begin{aligned}
E_1 &= -B_m G_{t,i-1/2} - C_4 G_{x,i-1/2} - C_2 G_{i-\frac{1}{2}} + C_3(G'_{c,i} + G'_{c,i-1/2}) \\
E_2 &= -B_m G_{t,i+1/2} + C_4 G_{x,i+1/2} - C_2 G_{i+\frac{1}{2}} + C_3(G'_{c,i} + G'_{c,i+1/2}) \\
D_1 &= \frac{1}{2}(E_1 + E_2) \\
D_2 &= (G_{i-1/2} + G_{i+1/2}) + \frac{\Delta t^*}{4}(G_{t,i-1/2} + G_{t,i+1/2}) + \frac{\Delta\tau}{8}(G_{x,i-1/2} - G_{x,i+1/2})
\end{aligned}$$

At the left boundary, $i = 0$, equation (CC-29) is not valid and must be replaced by the boundary condition, slightly rewritten as

$$G_{x,i} = \frac{3}{2}G_i + \frac{1}{2\alpha^2}G_{t,i}. \tag{CC-37}$$

Sticking this into equation (CC-31) (with $i + 1/2$ replaced by i) gives

$$f_1 G_{t,i} + f_2 G_i = E_2; \quad f_1 = B_p + \frac{C_4}{2\alpha^2} = B_p + \frac{1}{2}; \quad f_2 = C_2 + \frac{3}{2}C_4. \tag{CC-38}$$

Also, for the total derivative we can only use the $r \to l$ form, or

$$G_i = G_{i+1/2} + \frac{1}{2}(G_{t,i} + G_{t,i+1/2})\Delta t^* - \frac{1}{2}(G_{x,i} + G_{x,i+1/2})\frac{\Delta\tau}{2}, \tag{CC-39}$$

or, after eliminating $G_{x,i}$ through equation (CC-37)

$$f_3 G_{t,i} + f_4 G_i = D_2, \quad f_3 = \frac{\Delta\tau}{8\alpha^2} - \frac{\Delta t^*}{2}; \quad f_4 = 1 + \frac{3\Delta\tau}{8}, \tag{CC-40}$$

and, thus,

$$G_i = \frac{f_3 E_2 - f_1 D_2}{f_3 f_2 - f_1 f_4}; \quad G_{t,i} = \frac{f_2 D_2 - f_4 E_2}{f_3 f_2 - f_1 f_4}, \tag{CC-41}$$

and $G_{x,i}$ from equation (CC-37).

Similarly, for $i = N_x$ equation (CC-31) is not valid and must be replaced by the boundary at $\tau = \tau_L$, and for the total derivative the $l \to r$ version must be used, leading to very similar expressions.

Finally, transmissivity and reflectivity of the slab are simply the absolute value of the nondimensional fluxes at the boundaries, i.e.,

$$\begin{aligned}
\text{Reflectivity} &= \left|q(t^*, 0)\right| = \frac{1}{2}G(t^*, 0) \\
\text{Transmissivity} &= q(t^*, \tau_L) + q_c(t^*, \tau_L) = \frac{1}{2}G(t^*, \tau_L) + G_c(t^*, \tau_L).
\end{aligned} \tag{CC-42}$$

Input:

`Nx`	=	Number of equally-spaced nodes across slab,
`a`	=	P_a-approximation switch: `a` = 1 for P_1-approximation, `a` = 1/3 for $P_{1/3}$-approximation,
`L`	=	Thickness of slab, in m,
`beta`	=	Extinction coefficient β, in m^{-1},

`omga`	=	single scattering albedo, ω,
`tmax`	=	Maximum $t^*_{\max}$ to be considered in calculation,
`tps`	=	Total nondimensional pulse energy,
`tme`	=	Starting time for calculation; `tme` = `0` will start top-hat pulse at $t^* = 0$, `tme` = `−tps/2` will center top-hat pulse at $t^* = 0$, etc.
`tc,tp`	=	Pulse parameters for clipped-Gaussian pulse; note that `tp` = `tps`/ $\sqrt{\pi}$ results in a total pulse energy of `tps` (i.e., the same as for the top-hat pulse).

Output:

For every even time step the program prints out the value for `tme` = t^*, Transmissivity and Reflectivity as defined in equation (CC-42). Total pulse energy, total time integrated reflectivity and transmissivity are also printed out, which — for $\omega = 1$ — gives a check of truncation error and the proper choice for `tmax` to simulate the entire pulse.

Example: As an example we will analyze a slab of 1 m width using the $P_{1/3}$-approximation ($a = {}^1/_3$), with an extinction coefficient of $\beta = 5\,\mathrm{m}^{-1}$ (leading to an optical thickness of $\tau_L = 5$), and a scattering albedo of $\omega = 1$ (or 100%). Thus, we call the output file `transP3rd-5-100.dat`. We will use a top-hat laser pulse centered at $t = 0$, with a nondimensional pulse length of $t^*_p = 0.3$. Finally, we will use a spatial resolution of 200 nodes and, since it takes the signal 5 nondimensional time units to penetrate the slab and pure scattering will bounce around the beam for much longer, we choose a maximum t^* of 80. Thus, the beginning of the program looks as follows: (i) in the fifth line we have set `Nx=200`, (ii) under "pulse shape" we have uncommented the 4 'top-hat' lines, and (iii) we have fashioned the numbers below 'Input data' to fit our needs:

```
    program transPN
! Program to calculate energy transmitted as a function of time
! from a pulsed collimated laser source, through absorbing-scattering slab,
! using P1 and P1/3
    IMPLICIT NONE
    INTEGER, PARAMETER :: Nx=200
    INTEGER :: i,n
    DOUBLE PRECISION    :: L,tp,tps,beta,omga,tauL,dx,dt,trmsv,reflc,Bp,Bm,tme,tc
    DOUBLE PRECISION    :: G(0:Nx),Gx(0:Nx),Gt(0:Nx),G5(0:Nx),Gx5(0:Nx),Gt5(0:Nx)
    DOUBLE PRECISION    :: alf,c1,c2,c3,c4,Gc,Gc5,Gcp,Gcp5,Heav,y,E1,E2,D1,D2,f1,f2,f3,f4
    DOUBLE PRECISION    :: tmax,a,sumpls,sumtrn,sumref
    Heav(y)=FLOAT(INT(1.+.5*y/(abs(y)+1.d-15)))
! ********************   Pulse shape  ***********************************
! uncomment only one set of laser data below!!
! the following 4 lines simulate a top hat laser starting at n*dt=0
    Gc(n,i)=(Heav(n*dt-i*dx)-Heav(n*dt-i*dx-tps))*exp(-i*dx)
    Gc5(n,i)=(Heav(n*dt-(i+.5)*dx)-Heav(n*dt-(i+.5)*dx-tps))*exp(-(i+.5)*dx)
    Gcp(n,i)=Gc(n,i)
    Gcp5(n,i)=Gc5(n,i)
! the following 6 lines simulate a clipped Gaussian laser centered at n*dt=tc
!     Gc(n,i)=exp(-i*dx-((n*dt-i*dx-tc)/tp)**2) &
!             *(Heav(n*dt-i*dx)-Heav(n*dt-i*dx-2.*tc))
!     Gcp(n,i)=Gc(n,i)*(1.-2.*a*(n*dt-i*dx-tc)/tp**2)
!     Gc5(n,i)=exp(-(i+.5)*dx-((n*dt-(i+.5)*dx-tc)/tp)**2) &
!             *(Heav(n*dt-(i+.5)*dx)-Heav(n*dt-(i+.5)*dx-2.*tc))
!     Gcp5(n,i)=Gc5(n,i)*(1.-2.*a*(n*dt-(i+.5)*dx-tc)/tp**2)
!
! ********************   Output file  ***********************************
    open(unit=8,file='transP3rd-5-100.dat',status='unknown')
! ****************   Input data  *****************************************
    a=1.d0/3.d0 ! =1 for P1, =1/3 for P1/3 approximation
    L=1.        ! m
    beta=5.     ! 1/m
    omga=1
    tmax=80.    ! maximum t* to be considered
! pulse data: make sure to uncomment only 1 starting time "tme"
! pulse width for top-hat laser
    tps=0.3             ! total pulse duration = total pulse power
    tme=-tps/2.         ! non-zero value moves beginning of pulse; -tps/2 centers pulse at 0
! pulse shape for clipped Gaussian laser
    tc=0.5
    tp=tps/1.77245d0    ! total pulse power/sqrt(pi)
!   tme=-tc
! ****************  End of input data  *********************************
```

This leads to the following results stored in:

```
VARIABLES = tme,trmsv,reflc
zone
  -0.125   0.0000E+00   0.2536E-03
  -0.100   0.0000E+00   0.8391E-03
  -0.075   0.0000E+00   0.1675E-02
  -0.050   0.0000E+00   0.2744E-02
  -0.025   0.0000E+00   0.4027E-02
   0.000   0.0000E+00   0.5507E-02
   0.025   0.0000E+00   0.7167E-02
   0.050   0.0000E+00   0.8993E-02
   0.075   0.0000E+00   0.1097E-01
   0.100   0.0000E+00   0.1308E-01
   0.125   0.0000E+00   0.1533E-01
   0.150   0.0000E+00   0.1768E-01
   0.175   0.0000E+00   0.2012E-01
   0.200   0.0000E+00   0.2237E-01
   0.225   0.0000E+00   0.2444E-01
   0.250   0.0000E+00   0.2632E-01
   0.275   0.0000E+00   0.2804E-01
   0.300   0.0000E+00   0.2960E-01
   0.325   0.0000E+00   0.3103E-01
   0.350   0.0000E+00   0.3232E-01
   0.375   0.0000E+00   0.3348E-01
   0.400   0.0000E+00   0.3453E-01
   0.425   0.0000E+00   0.3548E-01
   0.450   0.0000E+00   0.3633E-01
   0.475   0.0000E+00   0.3708E-01
   0.500   0.0000E+00   0.3775E-01
   0.525   0.0000E+00   0.3835E-01
   0.550   0.0000E+00   0.3887E-01
   0.575   0.0000E+00   0.3932E-01
   0.600   0.0000E+00   0.3971E-01
   0.625   0.0000E+00   0.4004E-01
   0.650   0.0000E+00   0.4032E-01
   0.675   0.0000E+00   0.4055E-01
   0.700   0.0000E+00   0.4074E-01
   0.725   0.0000E+00   0.4088E-01
   0.750   0.0000E+00   0.4098E-01
   0.775   0.0000E+00   0.4105E-01
   0.800   0.0000E+00   0.4109E-01
   0.825   0.0000E+00   0.4109E-01
   0.850   0.0000E+00   0.4107E-01
   0.875   0.0000E+00   0.4102E-01
   0.900   0.0000E+00   0.4095E-01
   0.925   0.0000E+00   0.4086E-01
   0.950   0.0000E+00   0.4074E-01
   0.975   0.0000E+00   0.4061E-01
   1.000   0.0000E+00   0.4046E-01
    .
    .
   4.500   0.0000E+00   0.1518E-01
   4.525   0.0000E+00   0.1509E-01
   4.550   0.0000E+00   0.1501E-01
   4.575   0.0000E+00   0.1492E-01
   4.600   0.0000E+00   0.1484E-01
   4.625   0.0000E+00   0.1476E-01
   4.650   0.0000E+00   0.1468E-01
   4.675   0.0000E+00   0.1459E-01
   4.700   0.0000E+00   0.1451E-01
   4.725   0.0000E+00   0.1443E-01
   4.750   0.0000E+00   0.1436E-01
   4.775   0.0000E+00   0.1428E-01
   4.800   0.0000E+00   0.1420E-01
   4.825   0.0000E+00   0.1412E-01
   4.850   0.6893E-02   0.1405E-01
   4.875   0.7201E-02   0.1397E-01
   4.900   0.7507E-02   0.1390E-01
```

```
 4.925   0.7811E-02   0.1382E-01
 4.950   0.8114E-02   0.1375E-01
 4.975   0.8417E-02   0.1368E-01
 5.000   0.8718E-02   0.1361E-01
 5.025   0.9019E-02   0.1353E-01
 5.050   0.9319E-02   0.1346E-01
 5.075   0.9618E-02   0.1339E-01
 5.100   0.9917E-02   0.1332E-01
 5.125   0.1022E-01   0.1326E-01
 5.150   0.1052E-01   0.1319E-01
 5.175   0.3921E-02   0.1312E-01
 5.200   0.3912E-02   0.1305E-01
 5.225   0.3905E-02   0.1299E-01
 5.250   0.3899E-02   0.1292E-01
 5.275   0.3895E-02   0.1285E-01
 5.300   0.3892E-02   0.1279E-01
 5.325   0.3891E-02   0.1272E-01
 5.350   0.3891E-02   0.1266E-01
 5.375   0.3892E-02   0.1260E-01
 5.400   0.3893E-02   0.1253E-01
 5.425   0.3896E-02   0.1247E-01
 5.450   0.3899E-02   0.1241E-01
 5.475   0.3904E-02   0.1235E-01
 5.500   0.3909E-02   0.1229E-01
   .
   .
 8.000   0.4665E-02   0.8036E-02
 8.025   0.4667E-02   0.8006E-02
 8.050   0.4669E-02   0.7977E-02
 8.075   0.4671E-02   0.7948E-02
 8.100   0.4673E-02   0.7919E-02
 8.125   0.4674E-02   0.7890E-02
 8.150   0.4676E-02   0.7861E-02
 8.175   0.4677E-02   0.7833E-02
 8.200   0.4679E-02   0.7804E-02
 8.225   0.4680E-02   0.7776E-02
 8.250   0.4681E-02   0.7748E-02
 8.275   0.4682E-02   0.7720E-02
 8.300   0.4683E-02   0.7693E-02
 8.325   0.4684E-02   0.7665E-02
 8.350   0.4685E-02   0.7638E-02
 8.375   0.4685E-02   0.7611E-02
 8.400   0.4686E-02   0.7584E-02
 8.425   0.4686E-02   0.7557E-02
 8.450   0.4687E-02   0.7530E-02
 8.475   0.4687E-02   0.7504E-02
 8.500   0.4687E-02   0.7477E-02
 8.525   0.4687E-02   0.7451E-02
 8.550   0.4687E-02   0.7425E-02
 8.575   0.4687E-02   0.7399E-02
 8.600   0.4687E-02   0.7373E-02
 8.625   0.4687E-02   0.7348E-02
 8.650   0.4686E-02   0.7322E-02
 8.675   0.4686E-02   0.7297E-02
 8.700   0.4685E-02   0.7272E-02
 8.725   0.4685E-02   0.7247E-02
 8.750   0.4684E-02   0.7222E-02
 8.775   0.4683E-02   0.7197E-02
 8.800   0.4682E-02   0.7173E-02
 8.825   0.4681E-02   0.7148E-02
 8.850   0.4680E-02   0.7124E-02
 8.875   0.4679E-02   0.7100E-02
 8.900   0.4678E-02   0.7076E-02
 8.925   0.4677E-02   0.7052E-02
 8.950   0.4675E-02   0.7028E-02
 8.975   0.4674E-02   0.7004E-02
 9.000   0.4673E-02   0.6981E-02
   .
   .
```

```
  79.900   0.1443E-04   0.1431E-04
  79.925   0.1440E-04   0.1428E-04
  79.950   0.1437E-04   0.1425E-04
  79.975   0.1434E-04   0.1422E-04
  80.000   0.1431E-04   0.1419E-04
  80.025   0.1428E-04   0.1416E-04

Total transmission:    8.525E-02
Total reflection:      2.394E-01
Total trans+reflec:    3.246E-01
Total pulse enrg:      3.063E-01
```

Note that the transmissivity remains 0 until $t^* = 4.85$, when the beginning of the pulse has reached the opposite end by direct travel, and has its maximum at around $t^* \simeq 8.6$ (while the reflectivity peaks around $t^* \simeq 0.8$. Note that, for the present case of conservative scattering $\omega = 1$, the sum of transmissivity and reflectivity should equal the total pulse energy, or 0.3 (= t_p^*). The departures are due to the relatively coarse grid and the nonconservative nature of the P_a-approximation.

Chapter 20

`fskdist.f90`

Program `fskdist` is a Fortran90 code to calculate full spectrum k-distributions for a number of Planck function temperatures and a single gas property state (temperature, partial and total pressures), for a gas mixture containing CO_2, H_2O, CH_4 and soot; weight functions $a(T, T_{\text{ref}}, g)$ are calculated, as well. The spectral absorption coefficient is either calculated directly from the HITRAN or HITEMP databases, or is supplied by the user. The user should scan the code for `OPEN` statements, identifying input (HITRAN/HITEMP and/or absorption coefficient) and output files.

Input:

`Tref`	=	reference temperature (temperature of gas for evaluation of absorption coefficient, and also used as reference Planck function temperature), in K,
`Tmin`	=	minimum temperature for which a k-distribution and weight functions $a(T, T_{\text{ref}}, g)$ are to be calculated, in K,
`Tmax`	=	maximum temperature for which a k-distribution and weight functions $a(T, T_{\text{ref}}, g)$ are to be calculated, in K,
`numT`	=	number of different temperatures to be considered; equally spaced between `Tmin` and `Tmax`,
`P`	=	total pressure of gas mixture, bar,
`xmfr(3)`	=	mole fraction vector; `xmfr(1)`= mole fraction of CO_2, `xmfr(2)`= mole fraction of H_2O, `xmfr(3)`= mole fraction of CH_4; note that for any `xmfr` $< 10^{-3}$ the specie is neglected.
`fvsoot`	=	volume fraction of soot,
`nsoot, ksoot`	=	complex index of refraction for the soot; its absorption coefficient is assumed linear in wavenumber, using
`wvnm_b`	=	minimum wavenumber considered, cm^{-1},
`wvnm_e`	=	maximum wavenumber considered, cm^{-1},
`wvnmst`	=	wavenumber step (equally spaced) with which the absorption coefficient for the mixture is calculated from the HITRAN or HITEMP database, cm^{-1},
`kdmin`	=	minimum k-value to be considered for k-distribution, cm^{-1},
`kdmax`	=	maximum k-value to be considered for k-distribution (`kdmax` ≤ 0 sets `kdmax=kmax`, i.e., the maximum absorption coefficient found across the spectrum), cm^{-1}; allows for globally fixed k-values independent of k-distribution (useful for mixing),
`n_pwrk`	=	number of different k-bin values considered in the construction of the k-distribution,
`pwr`	=	exponent for k-bin values spacing: k-bins are equally spaced in k^{pwr} between `kdmin` and `kdmax`.
`nq`	=	number of quadrature points for radiative calculations, i.e., the number of (k, g)-pairs desired for RTE evaluations to be performed before spectral integration (over cumulative k-distribution g),
`iwr`	=	absorption coefficient switch: `iwr=0` to make a single complete run, i.e., evaluating κ_η from HITRAN or HITEMP (without storing them), followed by generation of k-distributions, `iwr=1` same but absorption coefficient is stored for future use, and `iwr=2`: precalculated absorption coefficients are read in and k-distributions are generated.

`ipl` = linear vs. pressure-based absorption coefficient switch:
`ipl`=0: calculate linear absorption coefficient, in cm^{-1}
`ipl`=1: calculate pressure-based absorption coefficient (allowed only for single absorbing gas!), in $\mathrm{cm}^{-1}\,\mathrm{bar}^{-1}$; if the pressure-based absorption coefficient for a dilute gas is desired, set `xmfr=1.d-3` (=0.1%)
`ipr` = output switch: see under output.

Output:
`ipr` = 1: a single output file is generated containing a header line (formatted for Tecplot), identifying the variables being printed, and `n_pwrk` data lines, each with 2 × `numT` + 2 numbers: k_i, (`numT` + 1) × $g(T_j, k_i)$, and (slightly smoothened) `numT` × $a(T_j, T_{\text{ref}}, k_i)$ (including T_{ref} for g.)
`ipr` = 2: in addition to the `ipr`=1 output file, a second file is generated, containing a header identifying variables, and `nq` output lines, each with `numT`+3 numbers: w_i, $g_i(T_{\text{ref}}, k_i)$, k_i, and `numT` smoothened $a(T_j, T_{\text{ref}}, k_i)$-values (averaged over its g-range).

Example:
We consider the full-spectrum k-distribution for a pressure-based absorption coefficient (`ipl`=1) of pure H_2O, for a vanishingly small mole fraction (`xmfr(3)=(/0.0d0,1.0d-3,0.d0/)`). Note that x_{H_2O} has been set to 10^{-3}: the code, when accessing HITRAN or HITEMP, will assume a specie not to be present whenever $x_i < 10^{-3}$. The absorption coefficient has been calculated in a previous run (`iwr`=2), and has been stored in file `absch2o-0p-2000K.dat` (for a wavenumber range from 50 cm^{-1} to 12000 cm^{-1} with a $\Delta\eta = 0.005\,\mathrm{cm}^{-1}$). We will calculate the k-distributions for 5 temperatures: a reference temperature $T_{\text{ref}} = T_0 = 2000\,\mathrm{K}$ (at which the absorption coefficient has been evaluated) and 4 equally spaced (Planck function) temperatures between $T_{\min} = 0\,\mathrm{K}$ and $T_{\max} = 1500\,\mathrm{K}$ (`numT`=4): this results in the 4 temperatures of 300 K, 500 K, 1000 K and 1500 K (the first temperature is not 0 K, because temperatures below 300 K are not accepted: any temperature below it is set to 300 K). We will use 500 k-bins (`n_pwrk`=500) with `pwr`=0.1 (this spreads the k-bins over many orders of magnitude, but places more and more bins into large magnitudes; see output file). We also set `kdmin`=10^{-7} ($\mathrm{cm}^{-1}\,\mathrm{bar}^{-1}$) and `kdmax` = $-20 < 0$ ($\mathrm{cm}^{-1}\,\mathrm{bar}^{-1}$), i.e., we will consider k-values between 10^{-7} and the maximum value found among the absorption coefficient values. Finally, we set `ipr`=2 and `nq`=12, i.e., besides the general k-distributions we want to also generate truncated k-distributions ready-made for numerical quadrature, using 12 quadrature points. The top of the program with input parameters, therefore, looks like this:

```
  MODULE Key
   IMPLICIT NONE
!HITRAN/HITEMP DATABASE
   INTEGER  :: lu
   INTEGER,PARAMETER  :: rows=1400000
   DOUBLE PRECISION,PARAMETER :: wvnm_b=50.d0,wvnm_e=12000.d0,wvnmst=0.005d0, &
                                 kdmin=1.d-7,kdmax=-20.d0
   DOUBLE PRECISION   :: data(rows,6)
  END MODULE Key

   PROGRAM Main
   USE Key
! Input parameters
   INTEGER,PARAMETER :: numT=4,n_pwrk=1000,iwr=2,ipl=1,ipr=2,nq=12
   DOUBLE PRECISION,PARAMETER :: P=1.d0,Tref=2000d0,Tmin=000d0,Tmax=1500d0
   DOUBLE PRECISION,PARAMETER :: xmfr(3)=(/0.0d0,1.0d-3,0.d0/)
   DOUBLE PRECISION,PARAMETER :: klmin=1.d-9,pwr=0.1d0
   DOUBLE PRECISION,PARAMETER :: fvsoot=0.d-6,nsoot=1.89d0,ksoot=0.92d0
```

where we have changed the values for `wvnm_b, wvnm_e, wvnmst, kdmin, kdmax, numT, n_pwrk, iwr, ipl, ipr, nq, Tref, Tmin, Tmax, xmfr` and `pwr` to fit our needs. Also, in this simulation we have set file names as

```
! Set output file name
   character(256), parameter :: kvsgFile='kvsgh2o-0p-2000K.dat'
   character(256), parameter :: kvsgqFile='kvsgqh2o-0p-2000K.dat'
   character(256), parameter :: abscFile='absch2o-0p-2000K.dat'
! Open output files
   OPEN(7,FILE=kvsgFile)
! Header formatted for TECPLOT, for a numT of 4
   write(7,6)
 6 format('VARIABLES = k,g0,g1,g2,g3,g4,a1,a2,a3,a4')
   IF(ipr==2) THEN
       OPEN(8,FILE=kvsgqFile,STATUS='unknown')
! Header formatted for readability, for a numT of 4
        write(8,8)
   ENDIF
 8 format('    wq',9x,'gq',9x,'kq',8x,'aq1',8x,'aq2',8x,'aq3',8x,'aq4')
! File containing absorption coefficient
   IF(iwr>0) OPEN(9,FILE=abscFile,STATUS='unknown')
```

i.e., the previously calculated absorption coefficient is located in `absch2o-0p-2000K.dat`, while the long *k*-distribution output (500 values) will be put into `kvsgh2o-0p-2000K.dat`, and the short, quadrature-ready output into `kvsgqh2o-0p-2000K.dat`. Note that the header lines for the output files are formatted for `numT=4` (see the two `format` statements above): they will need to be rewritten for different values of `numT`.

We will also assume that Numerical Recipes subroutines are available, leaving the following lines unchanged:

```
! Selection of g-values for numerical quadrature, using a Numerical Recipes routine
! If Numerical Recipes is not available, set nq=12, comment out the following 6 lines of code,
! and uncomment the 5-line REAL declaration following it
! Get quadrature coefficients from Numerical Recipes
   REAL              :: gqs(nq),wqs(nq),kq(nq),aq(numt,nq),gq(nq),wq(nq),gaujac,alf=3.,bet=0.,sum
    sum=0.
    CALL GAUJAC(gqs,wqs,nq,alf,bet)
          do iq=1,nq
            gq(iq)=0.5*(1.-gqs(iq))
            wq(iq)=wqs(iq)/(2.**(alf+bet+1)*gq(iq)**alf*(1.-gq(iq))**bet)
            sum=sum+wq(iq)
          enddo
! Correction to make sum(wq)=1
          wq=wq/sum
! End quadrature coefficients from Numerical Recipes
! Selection of precalculated g-values for numerical quadrature, for nq=12,alf=3.,bet=0.
!   REAL   :: kq(nq),aq(numt,nq), &
!     gq(nq)=(/ 5.120075E-02,1.170678E-01,2.015873E-01,3.007074E-01,4.095012E-01,5.225285E-01,  &
!               6.341280E-01,7.387071E-01,8.310236E-01,9.064499E-01,9.612060E-01,9.925594E-01/),&
!     wq(nq)=(/ 5.556622E-02,7.576839E-02,9.258290E-02,1.048306E-01,1.118451E-01,1.132605E-01,  &
!               1.090012E-01,9.927844E-02,8.457905E-02,6.563999E-02,4.341329E-02,1.904792E-02/)
```

This will calculate quadrature points `gq` and weights `wq` using Gaussian quadrature of moments (`alf=3` sets 3rd order moments). For users without access to Numerical Recipes the `gq` and `wq` calculated here have been put in data statements and may be used instead by following the guidelines above.

The previously calculated absorption coefficient in `absch2o-0p-2000K.dat` has the following form:

```
variables = "absco"
zone i= 2390001
#    50.00000 12000.00000     0.00500
   0.51219E-04
   0.51323E-04
   0.51428E-04
   0.51534E-04
   0.51642E-04
   0.51750E-04
   .
   .
   .
```

It is formatted for easy plotting using Tecplot, and has 2,390,001 absorption coefficient values between 50 cm^{-1} and 12000 cm^{-1}, spaced 0.005 cm^{-1} apart.
The output file `kvsgh2o-0p-2000K.dat` has this form (with the columns for `a3` and `a4` omitted to fit on the page):

```
VARIABLES = k,g0,g1,g2,g3,g4,a1,a2,a3,a4
  1.11334746D-07  7.96747373D-02  5.23869989D-04  2.26395097D-05  1.28391640D-03  2.13171234D-02  5.29826143D-05  2.63832240D-03  1.00906462D-01  4.55777755D-01
  1.36266566D-07  8.52815350D-02  5.23869989D-04  2.26735350D-05  1.50824331D-03  2.36291115D-02  1.00848658D-04  4.08710454D-03  1.19960706D-01  4.79094048D-01
  1.66115877D-07  9.00036377D-02  5.23869990D-04  2.27003552D-05  1.68726471D-03  2.55265665D-02  2.16767785D-04  7.38497566D-03  1.56250238D-01  5.22293603D-01
  2.01726786D-07  9.37613925D-02  5.23869990D-04  2.27194809D-05  1.82135979D-03  2.69993935D-02  4.37361553D-04  1.32407710D-02  2.06442469D-01  5.79223754D-01
  2.44067583D-07  1.02974254D-01  5.23934221D-04  8.52859416D-05  5.10437162D-03  3.41894314D-02  8.06580759D-04  2.24516206D-02  2.66272096D-01  6.42366201D-01
  2.94246088D-07  1.12533644D-01  5.24022055D-04  1.63032520D-04  8.87822956D-03  4.20105578D-02  1.36144304D-03  3.55584825D-02  3.31229083D-01  7.04508996D-01
  3.53526541D-07  1.23839293D-01  5.24099683D-04  2.30932151D-04  1.26253234D-02  5.07848713D-02  2.11775362D-03  5.25013589D-02  3.97103002D-01  7.60165878D-01
  4.23348170D-07  1.41672092D-01  5.24204831D-04  3.15420522D-04  1.79307268D-02  6.44793111D-02  3.06402561D-03  7.24871772D-02  4.60296729D-01  8.06256909D-01
  5.05345557D-07  1.63682856D-01  6.79267024D-04  4.18076075D-03  3.46648315D-02  8.58464331D-02  4.16735706D-03  9.41999954D-02  5.18053043D-01  8.41979615D-01
  6.01370944D-07  1.82498123D-01  7.95460318D-04  6.98729223D-03  4.71695621D-02  1.03117054D-01  5.38432553D-03  1.16228009D-01  5.68674321D-01  8.68168095D-01
  7.13518620D-07  1.99342127D-01  8.91344620D-04  9.21172573D-03  5.75711493D-02  1.18257717D-01  6.66353924D-03  1.37375476D-01  6.11507507D-01  8.86483146D-01
  8.44151550D-07  2.14065460D-01  9.59688608D-04  1.08096384D-02  6.56348712D-02  1.30896677D-01  7.93423318D-03  1.56618822D-01  6.46421228D-01  8.98637601D-01
  9.95930395D-07  2.28639506D-01  1.08555898D-03  1.32219150D-02  7.52270046D-02  1.44043846D-01  9.09287889D-03  1.72823102D-01  6.72970982D-01  9.05811610D-01
  1.17184510D-06  2.43521364D-01  1.29626952D-03  1.68374407D-02  8.70200012D-02  1.58100939D-01  1.00091993D-02  1.84611821D-01  6.89951229D-01  9.08452972D-01
  1.37524925D-06  2.57984408D-01  1.53627326D-03  2.08886000D-02  9.96127365D-02  1.72243921D-01  1.05612324D-02  1.90674644D-01  6.95893739D-01  9.06568830D-01
  1.60989730D-06  2.71057492D-01  1.69859458D-03  2.36947584D-02  1.09121994D-01  1.84158478D-01  1.06844430D-02  1.90389205D-01  6.90296094D-01  9.00338299D-01
  1.87998501D-06  2.83372640D-01  1.85241517D-03  2.63590759D-02  1.18246410D-01  1.95447797D-01  1.04050520D-02  1.84318068D-01  6.74688551D-01  8.90640135D-01
  2.19019319D-06  2.94707018D-01  1.96152698D-03  2.82804768D-02  1.25440138D-01  2.05304290D-01  9.83672164D-03  1.74204744D-01  6.52715674D-01  8.79138308D-01
  2.54573496D-06  3.04817787D-01  2.03859256D-03  2.96742175D-02  1.31095187D-01  2.13677119D-01  9.14489155D-03  1.62446894D-01  6.29124161D-01  8.67876894D-01
  2.95240684D-06  3.14363969D-01  2.10218069D-03  3.08225009D-02  1.36202937D-01  2.21508242D-01  8.50562312D-03  1.51354772D-01  6.08270704D-01  8.58652439D-01
  3.41664389D-06  3.24015430D-01  2.16685257D-03  3.20098365D-02  1.41584254D-01  2.29613988D-01  8.09474616D-03  1.42583906D-01  5.92969114D-01  8.52530310D-01
  3.94557900D-06  3.32537431D-01  2.22666684D-03  3.31359103D-02  1.46613815D-01  2.36881980D-01  8.14093680D-03  1.36979489D-01  5.84160699D-01  8.49726486D-01
  4.54710684D-06  3.41328215D-01  2.28089386D-03  3.41619508D-02  1.51377700D-01  2.44180400D-01  9.06733490D-03  1.34847804D-01  5.81367038D-01  8.49835656D-01
  5.22995258D-06  3.49624273D-01  2.34055944D-03  3.52255005D-02  1.56195879D-01  2.51245912D-01  1.17210749D-02  1.36508281D-01  5.83534641D-01  8.52222591D-01
  6.00374561D-06  3.58035652D-01  2.40321007D-03  3.63548190D-02  1.61112578D-01  2.58434724D-01  1.76275522D-02  1.42894888D-01  5.89833169D-01  8.56375019D-01
  6.87909880D-06  3.66653849D-01  2.47755172D-03  3.76261434D-02  1.66416406D-01  2.65944430D-01  2.91050097D-02  1.55922207D-01  6.00116335D-01  8.62094495D-01
  7.86769331D-06  3.74674767D-01  2.53082128D-03  3.85922339D-02  1.70965183D-01  2.72759681D-01  4.89979201D-02  1.78317154D-01  6.14936243D-01  8.69494644D-01
  8.98236949D-06  3.82226610D-01  2.58843812D-03  3.95972865D-02  1.75407365D-01  2.79220074D-01  7.98603097D-02  2.12745245D-01  6.35149421D-01  8.78842614D-01
  1.02372242D-05  3.89227742D-01  2.64363067D-03  4.06129729D-02  1.79956200D-01  2.85480820D-01  1.22726326D-01  2.60415217D-01  6.61281164D-01  8.90321322D-01
.
.
.
  1.95195872D+03  9.99998942D-01  1.00000000D+00  1.00000000D+00  9.99999851D-01  9.99999312D-01  4.61364475D-09  1.77221960D-04  1.40712612D-01  6.49679776D-01
  1.98997421D+03  9.99998962D-01  1.00000000D+00  1.00000000D+00  9.99999854D-01  9.99999326D-01  4.61364475D-09  1.77221960D-04  1.40712612D-01  6.49679776D-01
  2.02865475D+03  9.99998983D-01  1.00000000D+00  1.00000000D+00  9.99999857D-01  9.99999339D-01  4.61364475D-09  1.77221960D-04  1.40712612D-01  6.49679776D-01
  2.06801066D+03  9.99999004D-01  1.00000000D+00  1.00000000D+00  9.99999860D-01  9.99999353D-01  4.61364475D-09  1.77221960D-04  1.40712612D-01  6.49679776D-01
  2.10805239D+03  9.99999026D-01  1.00000000D+00  1.00000000D+00  9.99999863D-01  9.99999367D-01  4.61364474D-09  1.77221960D-04  1.40712612D-01  6.49679776D-01
  2.14879055D+03  9.99999048D-01  1.00000000D+00  1.00000000D+00  9.99999866D-01  9.99999381D-01  4.61364474D-09  1.77221960D-04  1.40712612D-01  6.49679776D-01
  2.19023589D+03  9.99999070D-01  1.00000000D+00  1.00000000D+00  9.99999869D-01  9.99999396D-01  4.61364473D-09  1.77221960D-04  1.40712612D-01  6.49679776D-01
  2.23239930D+03  9.99999093D-01  1.00000000D+00  1.00000000D+00  9.99999872D-01  9.99999411D-01  4.61364469D-09  1.77221959D-04  1.40712612D-01  6.49679776D-01
  2.27529180D+03  9.99999116D-01  1.00000000D+00  1.00000000D+00  9.99999876D-01  9.99999426D-01  4.61364458D-09  1.77221957D-04  1.40712612D-01  6.49679775D-01
  2.31892460D+03  9.99999140D-01  1.00000000D+00  1.00000000D+00  9.99999879D-01  9.99999441D-01  4.61364431D-09  1.77221951D-04  1.40712610D-01  6.49679773D-01
  2.36330903D+03  9.99999164D-01  1.00000000D+00  1.00000000D+00  9.99999882D-01  9.99999457D-01  4.61364375D-09  1.77221940D-04  1.40712607D-01  6.49679768D-01
  2.40845656D+03  9.99999188D-01  1.00000000D+00  1.00000000D+00  9.99999886D-01  9.99999472D-01  4.61364270D-09  1.77221919D-04  1.40712602D-01  6.49679760D-01
  2.45437885D+03  9.99999213D-01  1.00000000D+00  1.00000000D+00  9.99999889D-01  9.99999489D-01  4.61364091D-09  1.77221882D-04  1.40712592D-01  6.49679745D-01
  2.50108769D+03  9.99999238D-01  1.00000000D+00  1.00000000D+00  9.99999893D-01  9.99999505D-01  4.61363821D-09  1.77221827D-04  1.40712578D-01  6.49679723D-01
  2.54859502D+03  9.99999264D-01  1.00000000D+00  1.00000000D+00  9.99999896D-01  9.99999522D-01  4.61363456D-09  1.77221753D-04  1.40712558D-01  6.49679694D-01
  2.59691297D+03  9.99999290D-01  1.00000000D+00  1.00000000D+00  9.99999900D-01  9.99999539D-01  4.61363017D-09  1.77221664D-04  1.40712535D-01  6.49679658D-01
  2.64605380D+03  9.99999328D-01  1.00000000D+00  1.00000000D+00  9.99999905D-01  9.99999564D-01  4.61362544D-09  1.77221568D-04  1.40712509D-01  6.49679619D-01
  2.69602995D+03  9.99999375D-01  1.00000000D+00  1.00000000D+00  9.99999912D-01  9.99999594D-01  4.61362089D-09  1.77221476D-04  1.40712485D-01  6.49679582D-01
  2.74685401D+03  9.99999422D-01  1.00000000D+00  1.00000000D+00  9.99999919D-01  9.99999625D-01  4.61361698D-09  1.77221396D-04  1.40712464D-01  6.49679551D-01
  2.79853875D+03  9.99999471D-01  1.00000000D+00  1.00000000D+00  9.99999926D-01  9.99999656D-01  4.61361398D-09  1.77221335D-04  1.40712448D-01  6.49679526D-01
  2.85109710D+03  9.99999520D-01  1.00000000D+00  1.00000000D+00  9.99999932D-01  9.99999688D-01  4.61361191D-09  1.77221293D-04  1.40712437D-01  6.49679509D-01
  2.90454215D+03  9.99999570D-01  1.00000000D+00  1.00000000D+00  9.99999939D-01  9.99999720D-01  4.61361065D-09  1.77221268D-04  1.40712430D-01  6.49679499D-01
  2.95888719D+03  9.99999620D-01  1.00000000D+00  1.00000000D+00  9.99999947D-01  9.99999753D-01  4.61360995D-09  1.77221254D-04  1.40712427D-01  6.49679493D-01
  3.01414566D+03  9.99999672D-01  1.00000000D+00  1.00000000D+00  9.99999954D-01  9.99999787D-01  4.61360961D-09  1.77221247D-04  1.40712425D-01  6.49679491D-01
  3.07033118D+03  9.99999724D-01  1.00000000D+00  1.00000000D+00  9.99999961D-01  9.99999821D-01  4.61360946D-09  1.77221244D-04  1.40712424D-01  6.49679489D-01
  3.12745754D+03  9.99999778D-01  1.00000000D+00  1.00000000D+00  9.99999969D-01  9.99999856D-01  4.61360940D-09  1.77221242D-04  1.40712424D-01  6.49679489D-01
  3.18553874D+03  9.99999832D-01  1.00000000D+00  1.00000000D+00  9.99999976D-01  9.99999891D-01  4.61360938D-09  1.77221242D-04  1.40712423D-01  6.49679489D-01
  3.24458892D+03  9.99999887D-01  1.00000000D+00  1.00000000D+00  9.99999984D-01  9.99999927D-01  4.61360937D-09  1.77221242D-04  1.40712423D-01  6.49679489D-01
  3.30462243D+03  9.99999943D-01  1.00000000D+00  1.00000000D+00  9.99999992D-01  9.99999963D-01  4.61360937D-09  1.77221242D-04  1.40712423D-01  6.49679489D-01
  3.36565380D+03  1.00000000D+00  1.00000000D+00  1.00000000D+00  1.00000000D+00  1.00000000D+00  4.61360937D-09  1.77221242D-04  1.40712423D-01  6.49679489D-01
  3.39642078D+03  1.00000000D+00  1.00000000D+00  1.00000000D+00  1.00000000D+00  1.00000000D+00  4.61360937D-09  1.77221242D-04  1.40712423D-01  6.49679489D-01
```

Finally, output file `kvsgqh2o-0p-2000K.dat` contains quadrature *k*-values as:

```
   wq         gq         kq        aq1        aq2        aq3        aq4
 5.5566E-02 5.1201E-02 1.0056E-07 1.1349E-04 4.4089E-03 1.2272E-01 4.8243E-01
 7.5768E-02 1.1707E-01 3.1802E-07 1.6206E-03 3.9770E-02 3.4270E-01 7.0910E-01
 9.2583E-02 2.0159E-01 7.3344E-07 6.8338E-03 1.3182E-01 5.8699E-01 8.7111E-01
 1.0483E-01 3.0071E-01 2.4012E-06 9.8831E-03 1.7233E-01 6.4875E-01 8.7864E-01
 1.1185E-01 4.0950E-01 1.4519E-05 3.6367E-01 4.8589E-01 7.4587E-01 9.2075E-01
 1.1326E-01 5.2253E-01 6.4322E-05 7.4837E-01 9.0208E-01 1.0097E+00 1.0374E+00
 1.0900E-01 6.3413E-01 2.1014E-04 9.5075E-01 1.1213E+00 1.1166E+00 1.0833E+00
 9.9278E-02 7.3871E-01 6.6275E-04 9.6959E-01 1.2567E+00 1.2799E+00 1.1654E+00
 8.4579E-02 8.3102E-01 1.9721E-03 1.3306E+00 1.6521E+00 1.5328E+00 1.2665E+00
 6.5640E-02 9.0645E-01 6.1860E-03 3.0111E+00 2.5667E+00 1.8180E+00 1.3507E+00
 4.3413E-02 9.6121E-01 2.4663E-02 4.6993E+00 3.4362E+00 2.0650E+00 1.4178E+00
 1.9048E-02 9.9256E-01 1.5850E-01 1.4341E+01 6.5090E+00 2.6153E+00 1.5531E+00
```

Note that the code has an accuracy-checking mechanism built in: if the absorption coefficient is calculated from the HITRAN/HITEMP databases, the Planck-mean absorption coefficient is calculated directly from the database's line intensities,as well as by line-by-line integration of the absorption coefficient, : if the discrepancy exceeds 0.5% a message is printed to the screen, warning that `wvnst` is too coarse to properly resolve the absorption coefficient. The Planck-mean absorption coefficient is also calculated from the *k*-*g*-distribution. Again, if the discrepancy exceeds 0.5% a message is printed to the screen, warning that *k*-bin spacing is too coarse (`n_pwrk` too small) to properly resolve the absorption coefficient. For the above example, the choice of `n_pwrk`=500 results in an error of 1.78%, as indicated by the warning message.

`fskdco2.f90`, `fskdh2o.f90`

These subroutines determine single values of the cumulative *k*-distribution for CO_2 and H_2O, respectively, using the correlations of of Modest and Mehta [9] and of Modest and Singh [10].

Input for `fskdco2.f90`:

`Tg`	=	Gas temperature, i.e., temperature at which the absorption coefficient is evaluated, (in K)
`Tp`	=	Planck function temperature, i.e., temperature at which I_b is evaluated, (in K)
`absco`	=	Pressure-based absorption coefficient, (in $\text{cm}^{-1}\,\text{bar}^{-1}$)

Input for `fskdh2o.f90`: same as for `fskdco2.f90` plus

`x`	=	Mole fraction of water vapor, (–)

Output for both:

`gcal`	=	Cumulative k-distribution for the input conditions, (–).

`fskdco2dw.f90`, `fskdh2odw.f90`

These subroutines determine single values of the cumulative k-distribution for CO_2 and H_2O, respectively, using the correlations of Denison and Webb [11, 12].

Input for `fskdco2dw.f90`:

`Tg`	=	Gas temperature, i.e., temperature at which the absorption coefficient is evaluated, (in K)
`Tp`	=	Planck function temperature, i.e., temperature at which I_b is evaluated, (in K)
`Cabs`	=	Molar absorption cross-section, $R_u T_g k/xp$, (in m^2/mol)

Input for `fskdh2odw.f90`: same as for `fskdco2.f90` plus

`x`	=	Mole fraction of water vapor, (–)

Output for both:

`gcal`	=	Cumulative k-distribution for the input conditions, (–).

`kdistmix.f90`:

Subroutine `kdistmix` finds the cumulative k-distribution for an n-component mixture from a given set of individual species cumulative k-distributions (narrow band, wide band, or full spectrum), employing mixing schemes. Three mixing scheme are implemented, namely superposition, multiplication and uncorrelated mixture (Modest and Riazzi [13]). The mixing model is implemented as an independent module. For $n > 2$ `kdistmix.f90` should be called recursively. To invoke `kdistmix`, the user should call

```
use modkdistmix, only : kdistmix
call kdistmix(k1, g1, k2, g2, k, g, mixmodel, mixNop, mixScheme)
```

Input for subroutine `kdistmix`:

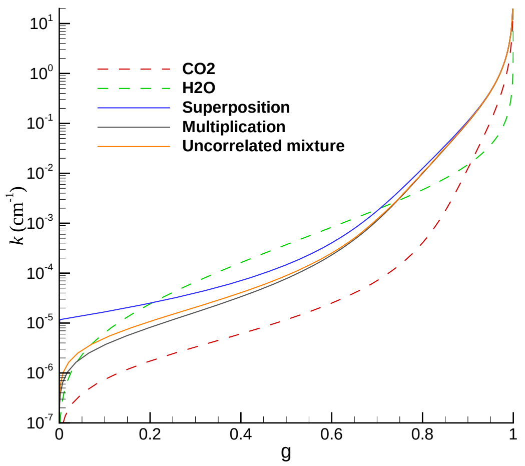
`k1`	=	A double precision array with k-values for the k-distributions of the first species, (in cm^{-1})
`g1`	=	A double precision array with g-values for the k-distributions corresponding to the k-values in array `k1`, (–). The size of `g1` must be the same as `k1`.
`k2`	=	A double precision array with k-values for the k-distributions of the second species, (in cm^{-1}). The size of `k2` may be different from `k1`.
`g2`	=	A double precision array with g-values for the k-distributions corresponding to the k-values in array `k2`, (–). The size of `g2` must be the same as `k2`, but may be different from `g1`.
`k`	=	A double precision array with k-values for the k-distributions of the mixture, (in cm^{-1}). The size of `k` may be different from `k1` and/or `k2`.
`mixmodel`	=	An optional integer scalar to specify the mixing model. Valid model numbers are 1 for superposition, 2 for multiplication and 3 for uncorrelated mixture (Modest and Riazzi). If not given, the uncorrelated mixture model will be used.
`mixNop`	=	An optional integer scalar to specify the minimum number of points for internal calculations. If not given, a value of 256 will be used. This number is only needed for the uncorrelated mixture model.
`mixScheme`	=	An optional integer scalar to specify the integration scheme for the uncorrelated mixture model and is only used for this model. If not given, a value of 0 for the default integration scheme will be used. Currently only the integration model is implemented. This number is reserved for future development.

Output for subroutine `kdistmix`:

`g`	=	A double precision array of rank one with g-values for the mixed k-distribution corresponding to the k-values, (–).

Example:

Consider a mixture of CO_2 and H_2O with mole fraction of 0.2 and 0.2, respectively. The mixture has a total pressure of 1 bar and temperature of 800K. The Planck function temperature is 1000K. The full-spectrum k-distribution data are determined from correlation tables. The following program finds the full-spectrum k-distributions of the mixture using three different mixing models (superposition, multiplication, and uncorrelated). The results are compared in the figure below:



This example also contains a function `kPowerLaw` to generate a list of *k*-values from a power law between minimum and maximum values.

Input for function `kPowerLaw`:

`kmin`	=	Minimum value (in cm^{-1}).
`kmax`	=	Maximum value (in cm^{-1}).
`n`	=	Number of *k*-values desired.
`pwr`	=	Exponent for *k*-value spacing (see also `fskdist.f90`).

Output for function `kPowerLaw`:

`k`	=	An array of rank one and size `n` that contains a list of *k*-values (in cm^{-1}). A sequence k_i for $i = 1, \ldots, n$ from a power law of power p (`pwr`) with a minimum $k_{\min}$ and a maximum $k_{\texttt{max}}$ has k_i^p equally distributed values between $k^p_{\min}$ and $k^p_{\max}$. A value of 0.1 for the power p is suggested.

```
program mixTest
use modkdistmix, only : kdistmix
implicit none

! export nb db
real(8),parameter :: P=1.d0,T=800.d0,xCO2=0.2d0,xH2O=0.2d0
real(8),parameter :: Trad=1000.d0
integer :: erflag=0, ik, ib
real(8) :: x, bb1,bb2
integer, parameter :: nop = 128, m=3
real(8),dimension(nop) :: k, gCO2, gH2O, gSup, gMul,gMR
real(8):: kp
real(8), parameter :: kmin=1.d-9, kmax=1.d2
k = kPowerLaw(kmin, kmax, nop, 0.1d0)
gCO2=1.d0;gH2O=1.d0
    do ik=1, nop
        kp=k(ik)/xCO2
        call fskdco2(T, Trad, kp, gCO2(ik))
        ! k in correlation is pressure based
    end do
    do ik=1, nop
        kp=k(ik)/xH2O
        call fskdh2o(T, Trad, kp, xH2O, gH2O(ik))
        ! k in correlation is pressure based
    enddo

call kdistmix(k, gCO2, k, gCO2, k, gSup, 1)
call kdistmix(k, gCO2, k, gCO2, k, gMul, 2)
call kdistmix(k, gCO2, k, gCO2, k, gMR, 3)
open(60, file='fskgMix.dat')
do ib = 1, nop
write(60,'(6f12.8)') k(ib),gCO2(ib), gH2O(ib), gSup(ib), gMul(ib), gMR(ib)
enddo
close (60)

contains
function kPowerLaw (kmin,kmax, n, pwr) result(k)
! function generate a list of k-values between kmin and kmax
! according to power law with power "pwr"
```

```
integer, parameter :: dp = kind(1.d0)
real(dp),  intent(in):: kmin, kmax, pwr
integer, intent(in) :: n
real(dp), dimension(n) :: k
real(dp) :: pwrk_min, pwrk_max, pwrk_step
integer :: i
pwrk_min = kmin**pwr
pwrk_max = kmax**pwr
pwrk_step = (pwrk_max-pwrk_min)/real(n-1, dp)
k = (/(pwrk_min+real(i-1,dp)*pwrk_step, i=1, n)/)
k = k**(1.d0/pwr)
end function kPowerLaw
end program
```

fskdistmix.f90:

This self-contained Fortran module finds the full spectrum cumulative *k*-distribution for a CO_2–H_2O mixture, employing the correlations of Modest and Mehta [9] and Modest and Singh [10], using one of three mixing schemes described by equations (20.162) (superposition), (20.163) (multiplication), or (20.167) (uncorrelated mixture).

To invoke the model, the user calls

```
use modfskdistmix, only : fskdistmix
call fskdistmix(xCO2, xH2O, Tg, Tp, kq, gq, m, nop,errflag)
```

Input for subroutine `fskdistmix`:

`xCO2`	=	CO_2 mole fraction, (–).
`xH2O`	=	H_2O mole fraction, (–).
`Tg`	=	Gas temperature (in K).
`Tp`	=	Planck function temperature (in K).
`m`	=	Integer to specify mixing model. m = 1 for superposition, 2 for multiplication and 3 for uncorrelated mixture (Modest and Riazzi)
`nop`	=	Integer to specify number of points for internal calculation.
`gq`	=	A double precision array for *g*-values (quadrature points).

Output for subroutine `fskdistmix`:

`kq`	=	A double precision array of *k*-values for the quadrature points specified by `gq`. *kq* is linear based and has the same size as `gq`.
`errflag`	=	Error flag. errflag = 0 if no error, errflag = 1 if error is found, such as a wrong model number.

This module also provides a subroutine for quadrature point calculation, generating Gaussian or Chebychev quadrature self-contained Fortran moduleonts between 0 and 1 and open at both ends. The corresponding quadrature weights are also calculated.

To invoke the quadrature subroutine, the user calls

```
use modfskdistmix, only : quadgen2
call quadgen2(Cheb, g, w, nq)
```

Input for subroutine `quadgen2`:

`Cheb`	=	A logical scalar to switch between Gaussian and Chebychev quadrature schemes. Should be set to True for Chebychev quadrature, False for Gaussian quadrature.
`nop`	=	An integer scalar specifying the number of quadrature points.

Output for subroutine `quadgen2`:

`g`	=	An array of size `nop` containing quadrature points.
`w`	=	An array of size `nop` containing quadrature weights.

Example:

In this example we consider a gas mixture with a total pressure of 1 bar, temperature of 800K. It contains 20% of CO_2 and 20% of H_2O by mole. The following program finds the full-spectrum *k*-distribution of a this mixture subject to 1000K Planck function temperature, using correlation tables and compares results between different mixing models.

```
program mixTest
use modfskdistmix, only : fskdistmix, quadgen2
implicit none
real(8),parameter :: P=1.d0,T=800.d0,xCO2=0.2d0,xH2O=0.2d0
real(8),parameter :: Trad=1000.d0
integer :: erflag=0, ib
integer, parameter ::nq = 16, nopcorr = 1024
real(8),dimension(nq) :: gq, wq
real(8),dimension(nq) :: kqSup, kqMul, kqMR
call quadgen2(.false., gq,wq, nq)
call fskdistmix(xCO2, xH2O, T, Trad, kqSup, gq, 1,nopcorr, erflag)
call fskdistmix(xCO2, xH2O, T, Trad, kqMul, gq, 2,nopcorr, erflag)
call fskdistmix(xCO2, xH2O, T, Trad, kqMR, gq, 3,nopcorr, erflag)
```

```
open(60, file='fskgCorr.dat')
do ib = 1, nq
write(60,'(5f12.5)') gq(ib),wq(ib), kqSup(ib), kqMul(ib), kqMR(ib)
enddo
close (60)
end program
```

The output quadrature g points, quadrature weights w, and k-values from three mixing models are listed below:

```
     gq          wq          k sup       k mul       k MR
  0.07051694  0.13911035  0.00010040  0.00001174  0.00001545
  0.20568663  0.13129793  0.00021668  0.00006478  0.00007703
  0.33189367  0.12078145  0.00046038  0.00021092  0.00023517
  0.44797743  0.11146496  0.00095065  0.00056075  0.00058797
  0.55303706  0.09846713  0.00190269  0.00132135  0.00144293
  0.64644658  0.08844043  0.00371492  0.00288260  0.00292762
  0.72786419  0.07436563  0.00716137  0.00600981  0.00634272
  0.79723565  0.06447459  0.01384539  0.01230641  0.01296693
  0.85479181  0.05076401  0.02734917  0.02538019  0.02569302
  0.90104015  0.04183752  0.05637761  0.05402394  0.05620029
  0.93675064  0.02984988  0.12386378  0.12137227  0.12518867
  0.96293624  0.02263203  0.29454263  0.29238274  0.30489881
  0.98082827  0.01353204  0.76224969  0.76083830  0.76602318
  0.99184741  0.00862024  2.17053771  2.16991559  2.16884325
  0.99757068  0.00328520  7.21939997  7.21923941  7.21288088
  0.99969530  0.00107660 35.31945789 35.31944491 35.28831336
```

The results are identical to the ones given in the previous example for `kdistmix.f90`, since k-distributions in that figure were obtained from the correlations.

Chapter 21

`mocacyl.f`, `rnarray.f`

Program `mocacyl` is a Monte Carlo routine for a nongray, nonisothermal, isotropically scattering medium confined inside a two-dimensional, axisymmetric cylindrical enclosure bounded by gray, diffusely emitting and reflecting walls. Temperature and radiative properties are assumed known everywhere inside the enclosure and along the walls. Requires use of program `rnarray` to set up random number relationships (locations and wavenumbers of emission vs. random numbers). Calculates local radiative fluxes to the walls q_w^R. `mocacyl.f` is supplied in two slightly different versions: `mocacyl_std.f` uses standard Monte Carlo for absorption, i.e., an absorption length is picked from equation (21.17) and the bundle is traced until this point is reached (and its energy is deposited at that point) or to a point on a wall, where it is absorbed (i.e., picking a random number $R_\alpha < \alpha$), whichever comes first. In the other version, `mocacyl_ep.f`, the energy partitioning scheme of Sections 8.7 and 21.7 is employed, i.e., the bundle's energy is gradually attenuated by absorption along its path, and by the fraction α, whenever the bundle hits (and is reflected from) a wall, until the bundles energy becomes negligible. This method is somewhat more expensive per bundle, but should in many instances give converged results with a lot less bundles. At the present time only `mocacyl_std.f` also calculates the internal radiative source $-\nabla \cdot \mathbf{q}^R$ in addition to wall fluxes.

The package consists of the following files:

- the main programs `mocacyl_std.f` and `mocacyl_ep.f` ,
- the program preparing random number relationships for medium emission, `rnarray.f`,
- file `mocasub.f`, which contains simple versions of subroutine `PROPS` and function `ABSCO`, as well as a poor man's random number generator called `RNUM`, all of which the user can (and should) replace,
- a file `splines.f` for monotonic splines,used by both `mocacyl.f` and `rnarray.f`, and
- sample output files `datlam.dat` and `results.dat`.

Program **`rnarray`**

This program prepares random number relationships for photon emission locations within the cylindrical medium, using equations (21.9) and (21.11).

Input:

`NRP`	=	Number of radial nodes for medium emission random number relationships

NZP	=	Number of axial nodes for medium emission random number relationships
NNP	=	Number of random numbers for medium emission relationships
RL	=	Radius of cylinder, (cm)
ZL	=	Length of cylinder, (cm)
AN	=	Refractive index of medium (AN=1.0 for gases)
STN	=	Refractive index of soot
STK	=	Absorptive index of soot
IGRAY	=	Gray/nongray medium switch: IGRAY=0 nongray, IGRAY=1 gray (ignoring contribution from gases; absorption coefficient = PAC)
LU	=	Logical unit number for output: LU=6 sends output to screen, other (legal) values send output to file datlam.dat

Output:
File datlam.dat contains random number relationships generated by RNARRAY:

ETOTAL	=	Total energy emitted (per unit time) by entire volume, in W
PLMCL(I),I=1,NZP	=	Planck-mean absorption coefficient along centerline, in cm^{-1}
RRA(J),J=1,NNP	=	Emission radial location as f(random#), in cm
ZR(K,J),K=1,NRP,J=1,NNP	=	Emission axial location as $f(r,$ random#), in cm
WVE2(K,I,J),K=1,NRP,I=1,NZP,J=1,NRNP1	=	Emission wavelength as $f(r,z,$ random#) (IGRAY=0 only), in μm

These arrays are used by mocacyl.f to determine emission location and wavelength, using single (r), double (z), and triple (λ) linear interpolation between tabulated values.

Note: this program requires two user-supplied subroutines, SUBROUTINE PROPS and FUNCTION ABSCO. Subroutine PROPS(R,Z,T,SVF,PCO2,PH2O,PAC,PSC), upon inputting radial position R (in cm) and axial position Z (in cm), must return local values of T (temperature in K), SVF (soot volume fraction, –), PCO2 (partial pressure of CO_2, in bar), PH2O (partial pressure of H_2O, in bar), PAC (nonsoot particle background absorption coefficient, in cm^{-1}), and PSC (nonsoot particle background scattering coefficient, in cm^{-1}). As provided here, the subroutine produces a uniform field, i.e., SVF=0., T=1000., PH2O=.1, PCO2=.1, PAC=.01, PSC=0.
Function ABSCO(SVF,PCO2,PH2O,PAC,W,T), upon inputting SVF (soot volume fraction, –), PCO2 (partial pressure of CO_2, in bar), PH2O (partial pressure of H_2O, in bar), PAC (nonsoot particle background absorption coefficient, in cm^{-1}), W (wavelength in μm), and T (temperature in K), must return ABSCO, the absorption coefficient of the medium (in cm^{-1}). As provided, function ABSCO calculates the gas absorption coefficient from the wide-band model [with an approximate evaluation of $\alpha(T)/\alpha_0$ in equation (11.144)] assuming strong overlap ($\beta \to \infty$), and the soot absorption coefficient is calculated from equation (12.123).
Function ABSCO should return ABSCO=PAC if W < 0 (gray medium). Both, PROPS and ABSCO, must contain the common statement line COMMON RL,ZL,AN,STN,STK,NRR,NZL,NRN.

Program **mocacyl**
Program mocacyl requires the following input:

NRP	=	Number of radial nodes for medium emission random number relationships
NZP	=	Number of axial nodes for medium emission random number relationships
NNP	=	Number of random numbers for medium emission relationships
NR	=	Number of radial nodes for surface flux calculations
NZ	=	Number of axial nodes for surface flux calculations
T3(NZ)	=	Temperature of liner wall ($r = R$), (K)
EPS(3)	=	Surface emittances: EPS(1)=inlet, EPS(2)=exit, EPS(3)=liner
RL	=	Radius of cylinder, (cm)
ZL	=	Length of cylinder, (cm)
AN	=	Refractive index of medium (AN=1.0 for gases)
STN	=	Refractive index of soot
STK	=	Absorptive index of soot
NTOTAL	=	Total number of photon bundles emitted from medium (number of bundles for surface emission are chosen automatically as function of NTOTAL)
IGRAY	=	Gray/nongray medium switch: IGRAY=0 nongray, IGRAY=1 gray (ignoring contribution from gases and soot; absorption coefficient = PAC)
IWALL	=	Wall emission switch: IWALL=0 only considers medium emission; IWALL=1 also considers surface emission
LU	=	Logical unit number for output: LU=6 sends output to screen, other (legal) values send output to file results.dat

File `datlam.dat` Contains random number relationships generated by `rnarray`

Note: The program **does not check** for consistency of `datlam.dat`, i.e., whether identical input values have been chosen in both `rnarray` and `mocacyl`!

Output:
Upon output relevant input data are displayed, as well as

`QW(1,1..NR)`	=	Axial radiative heat flux for `NR` radial nodes at inlet (W/cm^2)
`QW(2,1..NR)`	=	Axial radiative heat flux for `NR` radial nodes at exit (W/cm^2)
`QW(3,1..NZ)`	=	Radial radiative heat flux for `NZ` axial nodes at liner (W/cm^2)

Note: `QW` > 0 implies that the flux goes into wall, while for `QW` < 0 the flux is out of the wall.

Example:
We will use routines `PROPS` and `ABSCO` as provided, and also the input data as stated in `mocacyl` (and similar in `rnarray`):

```
 C C GENERAL DATA C C T3=liner temperature (K),
EPS=emittances(inlet,exit,liner), C RL=radius (cm), ZL=length (cm),
AN=refractive index of medium (-), C STN= soot refractive index, STK= soot
absorptive index, C NRR=# radial nodes, NZL=# axial nodes, NRN=# wavelength
nodes C
      DATA T3/7*1000./
      DATA EPS/1.,1.,.5/
      RL=10.
      ZL=10.
      TW0=1000.
      TWL=0.
      AN=1.
      STN=1.89
      STK=0.92
      NTOTAL=500000
      IGRAY=0
      IWALL=1
      LU=7
```

This models an isothermal, cylindrical, nongray medium with temperatures of 1000 K for medium, inlet and liner, only the exit being cold at 0 K. For the simulation we will use 500,000 bundles for medium emission, and corresponding numbers for surface emission (such that all bundles carry roughly identical energies). Executing `rnarray` produces the required data file `datlam.dat` and, running `mocacyl`, the results are contained in `results.dat` as:

```
         GENERAL DATA
         ************

              BURNER RADIUS                  10.00 CM
              BURNER LENGTH                  10.00 CM
              EMITTANCES:              INLET:  1.00
                                        EXIT:  1.00
                                       LINER:  0.50
              REFRACTIVE INDEX                 1.00
              NUMBER OF BUNDLES: MEDIUM: 500000
                                  INLET: 474561
                                   EXIT:      1
                                  LINER: 474562



    WALL TEMPERATURES (DEG.K)

      TW0   TWL  T3:I= 1     2     3     4     5     6     7
     1000.   0.     1000. 1000. 1000. 1000. 1000. 1000. 1000.



    PROPERTY VALUES ALONG CENTER LINE:

         I     Z    T      FR   PCO2   PH2O    K-PL  ABSC   SCAT
              CM   DEG.K    %    ATM    ATM    CM-1  CM-1   CM-1

         1    0.0  1000.  0.00  0.100  0.100  0.032  0.010  0.000
         2    0.5  1000.  0.00  0.100  0.100  0.032  0.010  0.000
         3    1.0  1000.  0.00  0.100  0.100  0.032  0.010  0.000
         4    1.5  1000.  0.00  0.100  0.100  0.032  0.010  0.000
         5    2.0  1000.  0.00  0.100  0.100  0.032  0.010  0.000
         6    2.5  1000.  0.00  0.100  0.100  0.032  0.010  0.000
         7    3.0  1000.  0.00  0.100  0.100  0.032  0.010  0.000
         8    3.5  1000.  0.00  0.100  0.100  0.032  0.010  0.000
         9    4.0  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        10    4.5  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        11    5.0  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        12    5.5  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        13    6.0  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        14    6.5  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        15    7.0  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        16    7.5  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        17    8.0  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        18    8.5  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        19    9.0  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        20    9.5  1000.  0.00  0.100  0.100  0.032  0.010  0.000
        21   10.0  1000.  0.00  0.100  0.100  0.032  0.010  0.000


    RADIAL HEAT FLUXES AT NODES I (W/SQCM)

         I     Z   QWALL
              CM   W/SQCM

         1    0.7   -0.4
         2    2.1   -0.4
         3    3.6   -0.5
         4    5.0   -0.7
         5    6.4   -0.8
         6    7.9   -1.0
         7    9.3   -1.2


    AXIAL HEAT FLUXES AT NODES J (W/SQCM)

 J       1     2     3     4     5
 Q0    -2.4  -2.3  -2.1  -1.9  -1.6
 QL     5.9   5.8   5.8   5.6   5.4
```

Note that the fluxes for the three hot walls are slightly negative (surfaces are losing heat, while the cold exit surface experiences strong positive heat fluxes. Also note that the code does not provide error estimates, i.e., it should be run for different values of NTOTAL to get an idea of variances.

FwdMCcs.f90

Program FwdMCcs is a standard forward Monte Carlo code for a narrow collimated, cylindrical beam (centered at $x = y = 0$) penetrating through a nonabsorbing, isotropically scattering slab, calculating the flux onto a small, directionally-selective detector located at $x_0 < x < x_0 + dx$, $0 < y < dy$, $z = L$, as shown in Fig. 21-6 and described in Example 21.3. (FwdMCck1 and FwdMCck2 are forward Monte Carlo codes for the same problem, but also allow for absorption in the medium; FwdMCck1 uses standard ray tracing, while FwdMCck2 uses energy partitioning; see Example 21.4.)

Input:

L	=	thickness of layer, L (m);
sig	=	scattering coefficient of medium, σ_s (m^{-1});
QT	=	total energy contained in collimated beam, (W);
R	=	radius of collimated beam, R (m);
x0	=	displacement of left end of detector from center of beam, x_0 (m);
dx	=	width of detector in x-direction, (m);
dy	=	width of detector in y-direction, (m);
thd	=	opening angle of detector, $\theta_{\max}$ (degrees);
N	=	minimum number of photon bundles to be traced for each "sample;"
numsmpl	=	number of numerical "samples" collected for the determination of a variance;
stddevmax	=	maximum relative variance allowed for the calculation of q_{det}, the flux hitting detector, $= \sigma_m/q_{\text{det}}$ from equation (8.8).

The values for input parameters are assigned in sequence near the top of the program. As distributed, a 1 m thick layer with a scattering coefficient of $\sigma_s = 1\text{m}^{-1}$ is modeled, for a 10 cm radius beam of 100 W strength. The (rather large) detector is 10 cm × 10 cm displaced by 20 cm from the center of the beam, and has an acceptance angle of 10°. `numsmpl`= 10 numerical samples will be taken, initially each containing `N`= 100,000 bundles, to be increased (if necessary) until the relative variance falls below `stddevmax`= 0.05 or 5%.

```
    open(unit=8,file='fwdmccs.dat',status='unknown')
    write(8,1)
    write(*,1)
  1 format('no. of bundles     q_det     variance    rel.var.(%)')
!
    L=1.        ! m
    sig=1.      ! 1/m
    QT=100.     ! W
    R=.1        ! m
    x0=.2       ! m
    dx=0.10     ! m
    dy=0.10     ! m
    thd=10.     ! deg
    N=100000
    numsmpl=10
    stddevmax=0.05
!
```

The program consists of two major parts. The first is a double loop over `numsmpl*N` photon bundles, tracing their paths, as described in Example 21.3. In the second part an average value for the detector irradiation is determined, as well as its relative standard deviation, based on the `numsampl` data points. If the relative standard deviation is too large (`stddev>stddevmax`) the `numsampl` samples of q_det (based on N bundles) are combined into ½×`numsampl` samples (with $2N$ bundles each), the number of bundles is doubled to $2N$, and an additional ½×`numsampl` samples are obtained (with $2N$ bundles each). Thus, after going through the bundle-tracing part one more time, we have again `numsmpl` samples, but each based on twice as many photon bundles. This procedure is repeated until the convergence criteria are met.

For the given case that leads to the following output, stored in `fwdmccs.dat`,

```
no. of bundles     q_det     variance    rel.var.(%)
       1000000  0.3200E-02  0.4899E-03      15.31
       2000000  0.3500E-02  0.3944E-03      11.27
       4000000  0.3100E-02  0.3055E-03       9.86
       8000000  0.2963E-02  0.1468E-03       4.95
```

i.e., for this large detector 8,000,000 photon bundles are needed to attain a relative variance of less than 5%. If a smaller detector was chosen, the necessary number of bundles would be roughly inversely proportional to the detector area!

FwdMCck1.f90

Program `FwdMCck1.f90` is identical to `FwdMCcs.f90`, except that the medium also absorbs radiation (besides isotropically scattering it). Therefore, the input is:

Input:

`L`	=	thickness of layer, L (m);
`sig`	=	scattering coefficient of medium, σ_s (m^{-1});
`kap`	=	absorption coefficient of medium, κ (m^{-1});
`QT`	=	total energy contained in collimated beam, (W);
`R`	=	radius of collimated beam, R (m);
`x0`	=	displacement of left end of detector from center of beam, x_0 (m);
`dx`	=	width of detector in x-direction, (m);
`dy`	=	width of detector in y-direction, (m);
`thd`	=	opening angle of detector, θ_max (degrees);
`N`	=	minimum number of photon bundles to be traced for each "sample;"
`numsmpl`	=	number of numerical "samples" collected for the determination of a variance;
`stddevmax`	=	maximum relative variance allowed for the calculation of q_det, the flux hitting detector, $= \sigma_m/q_\text{det}$ from equation (8.8).

As distributed, a 1m thick layer with a scattering coefficient of 1m^{-1} and an absorption coefficient of $\kappa = 1\text{m}^{-1}$ is modeled, for a 10 cm radius beam of 100 W strength. The (rather large) detector is 10 cm × 10 cm displaced by 20 cm from the center of the beam, and has an acceptance angle of 10°. `numsmpl`= 10 numerical samples will be taken, initially each containing `N` = 100,000 bundles, to be increased (if necessary) until the relative variance falls below `stddevmax` = 0.05 or 5%:

```
    open(unit=8,file='fwdmck1.dat',status='unknown')
    write(8,1)
    write(*,1)
  1 format('no. of bundles     q_det     variance    rel.var.(%)')
!
    L=1.         ! m
    sig=1.       ! 1/m
    kap=1.       ! 1/m
    QT=100.      ! W
    R=.1         ! m
    x0=.2        ! m
    dx=0.10      ! m
    dy=0.10      ! m
    thd=10.      ! deg
    N=100000
    numsmpl=10
    stddevmax=0.05
!
```

For the given case that leads to the following output, stored in `fwdmck1.dat`:

```
no. of bundles     q_det     variance    rel.var.(%)
       1000000  0.1100E-02  0.2333E-03      21.21
       2000000  0.7500E-03  0.2007E-03      26.76
       4000000  0.6500E-03  0.1067E-03      16.42
       8000000  0.6875E-03  0.7512E-04      10.93
      16000000  0.6438E-03  0.4375E-04       6.80
      32000000  0.6781E-03  0.4396E-04       6.48
      64000000  0.6766E-03  0.3424E-04       5.06
     128000000  0.7102E-03  0.3011E-04       4.24
```

i.e., 128,000,000 photon bundles are required, or – making allowance for the slightly different variance – about 10 times as many as for the purely scattering medium. Clearly, with a minimum optical thickness of $\sqrt{1^2 + 0.2^2} = 1.02$ many photon bundles, that would otherwise be scattered toward the detector, become absorbed first.

FwdMCck2.f90

Program `FwdMCck2.f90` is identical to `FwdMCck1.f90`, except that energy partitioning is employed, i.e., photon bundles are emitted and have paths identical to the simulation in `FwdMCcs.f90`, but the bundles' strengths are attenuated exponentially along their way according to Beer's law. Input is identical to `FwdMCck1.f90`, as are the as-distributed input parameters. However, the output (stored in `fwdmck2.dat`) now looks like this:

```
no. of bundles     q_det     variance    rel.var.(%)
       1000000  0.9003E-03  0.1610E-03      17.89
       2000000  0.9382E-03  0.1109E-03      11.82
       4000000  0.8160E-03  0.8774E-04      10.75
       8000000  0.7927E-03  0.3710E-04       4.68
```

i.e., `FwdMCck2.f90` converges at he same rate as the no-absorption case `FwdMCcs.f90`, demonstrating the power of the energy partitioning approach.

FwdMCps.f90

Program `FwdMCps` is a standard forward Monte Carlo code for radiative energy emitted by a point source penetrating through a nonabsorbing, isotropically scattering slab, calculating the flux onto a small, directionally-selective detector. It is a variation of `FwdMCcs.f90`, considering a purely scattering slab, but replacing the collimated beam by an internal point source at $x = 0$, $y = 0$, $z = z_{ps}$. Thus, the simulation is almost identical to that of `FwdMCcs.f90`, except that all photon bundles are now emitted from a single point, but into random directions. The input is also identical to `FwdMCcs.f90`, with `R` replaced by `zps`:

Input:

`L`	=	thickness of layer, L (m);
`sig`	=	scattering coefficient of medium, σ_s (m^{-1});
`QT`	=	total energy contained in point source, (W);
`zps`	=	z-coordinate of point source, (m);
`x0`	=	displacement of left end of detector from point source, x_0 (m);
`dx`	=	width of detector in x-direction, (m);

dy	=	width of detector in y-direction, (m);
thd	=	opening angle of detector, $\theta_{\max}$ (degrees);
N	=	minimum number of photon bundles to be traced for each "sample;"
numsmpl	=	number of numerical "samples" collected for the determination of a variance;
stddevmax	=	maximum relative variance allowed for the calculation of $q_{\det}$, the flux hitting detector, $= \sigma_m/q_{\det}$ from equation (8.8).

The as-delivered case also is the same as for `FwdMCcs.f90`, with the 10 cm-radius beam replaced by a points source at `zps` = 0.5(m). Thus, the input section reads:

```
    open(unit=8,file='fwdmcps.dat',status='unknown')
    write(8,1)
    write(*,1)
  1 format('no. of bundles     q_det     variance    rel.var.(%)')
!
    L=1.         ! m
    sig=1.       ! 1/m
    QT=100.      ! W
    zps=.1       ! m
    x0=.2        ! m
    dx=0.10      ! m
    dy=0.10      ! m
    thd=10.      ! deg
    N=100000
    numsmpl=10
    stddevmax=0.05
!
```

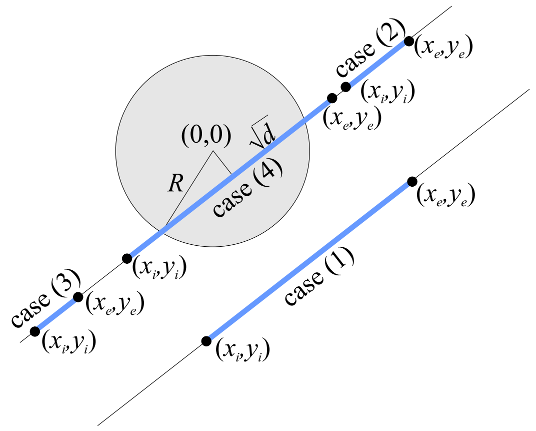
The resulting output, stored in `fwdmcps.dat`, is:

```
no. of bundles     q_det     variance    rel.var.(%)
       1000000  0.5000E-02  0.7303E-03      14.61
       2000000  0.4550E-02  0.4561E-03      10.02
       4000000  0.4800E-02  0.3958E-03       8.25
       8000000  0.4850E-02  0.3226E-03       6.65
      16000000  0.4850E-02  0.2143E-03       4.42
```

`RevMCcs.f90`

This program is a reverse or backward Monte Carlo implementation of the problem solved by `FwdMCcs.f90`, i.e., a narrow collimated beam hitting a purely scattering slab, and scattered toward a small, directionally-selective detector. Input and output are identical to `FwdMCcs`, except that the default minimum number of photon bundles is much smaller, here set to `N=1000`.

Again, the program consists of two parts, a double loop tracing over `numsmpl`×`N` photon bundles, and a module calculating detector power and its standard deviation. Tracing follows the rules of equation (21.84), and the detector flux from equation (21.79) (with $\epsilon'_\lambda = 1$ for the black detector). The coding is self-explanatory except when, between two scattering events, the bundle starts and ends outside the collimated beam column, $x^2 + y^2 = r^2 > R^2$. There are four different possibilities, as illustrated in the sketch below:



1. The path of the bundle misses the beam altogether,
2. the path intercepts the beam, but the bundle is moving away from the beam,

3. the path intercepts the beam and moves toward it, but is scattered anew before reaching it, and finally

4. the bundles path intersects the beam.

These four possibilities are tested by calculating the partial distances l_1 and l_2 (`lpart`) from the starting point (projected into the x-y-plane) (x_i, y_i) toward (x_e, y_e) (the final point) to where the path enters and exits the beam. Thus, using the x- and y-components of the vector equation

$$\mathbf{r}_i + l\hat{\mathbf{s}} = \mathbf{R}$$

leads to

$$x_i + ls_x = x_c,$$
$$y_i + ls_y = y_c,$$

where (x_c, y_c) is a point on the outer limit of the collimated beam, $r = R$. Squaring and adding the latter two equations gives

$$x_i^2 + y_i^2 + 2(x_i s_x + y_i s_y)l + (s_x^2 + s_y^2)l^2 = R^2$$

or

$$l_{1,2} = \frac{-b \pm \sqrt{d}}{c},$$

where

$$a = x_i^2 + y_i^2 - R^2$$
$$b = x_i s_x + y_i s_y$$
$$c = s_x^2 + s_y^2$$
$$d = b^2 - ac.$$

Thus the above four scenarios correspond to

1. If $d < 0$ the roots are complex, i.e., there is no intersection,

2. If $l_1 < 0$ the bundle moves away from beam

3. If $l_1 < l_\sigma$ the bundle is scattered again before reaching the beam, and

4. otherwise both intercepts are calculated to determine $I_{\lambda n}$.

After the tracing of photon bundles is completed, average values and standard deviations are calculated as in the forward Monte Carlo codes, e.g., `FwdMCcs`. Using the as-supplied input (same as for `FwdMCcs`, but with `N=1000`) leads to the output, stored in `revmccs.dat`:

```
no. of bundles     q_det      variance    rel.var.(%)
        10000  0.2882E-02  0.1192E-03       4.14
```

Thus, a better variance is achieved with only 10,000 bundles, as opposed to the 8,000,000 bundles used in the forward simulation (and this ratio would become correspondingly more extreme for smaller detector areas and acceptance angles).

`RevMCck1.f90`, `RevMCck2.f90`
These programs are backward Monte Carlo Implementations of `FwdMCck1` and `FwdMCck2`, respectively, as also discussed in Example 21.4. They have identical inputs (except the much lower minimum number of photon bundles, here set to `N=1000` as default); and their outputs also follow the format of their counterparts. Ray tracing for `RevMCck1` is the same as for `RevMCcs`, except for the slight modification demanded by equation (21.85). If energy partitioning is used, attenuation along the entire path length of the photon bundle must be considered, as explained in the last equation of Example 21.4. For the as-supplied cases the output from `RevMCck1`, stored in `revmck1.dat`, is:

```
no. of bundles     q_det      variance    rel.var.(%)
        10000  0.7261E-03  0.5045E-04       6.95
        20000  0.7540E-03  0.3091E-04       4.10
```

i.e., 20,000 bundles are required (as opposed to 128,000,000 used by `FwdMCck1`). For `RevMCck2` the output, stored in `revmck2.dat`, is:

```
no. of bundles     q_det      variance    rel.var.(%)
         10000  0.7623E-03  0.3061E-04       4.02
```

i.e., 10,000 bundles are required (as opposed to 8,000,000 used in `FwdMCck2`, or 20,000 used in `RevMCck1`).

`RevMCps.f90`

This program is the backward Monte Carlo equivalent of `FwdMCps`, with identical input and output format (again, with the exception of a much smaller base line value for the number of bundles). In the backward Monte Carlo simulation, the detector flux again consists of a direct and a scattered component. In the code it is assumed that the direct component is zero, this time because all direct radiation hits the detector at an angle larger than the acceptance angle (this could, of course, be easily changed). As for collimated irradiation backward Monte Carlo also becomes inefficient if the radiation source comes from a very small surface or volume. The trick is again to break up intensity into a direct component (intensity coming directly from the source without scattering or wall reflections), and a multiply-scattered and reflected part, as given by Modest [14] and briefly described here. Again, we let I_d satisfy the radiative transfer equation without the inscattering term, or,

$$\hat{\mathbf{s}} \cdot \nabla I_d(\mathbf{r}, \hat{\mathbf{s}}) = S_d(\mathbf{r}, \hat{\mathbf{s}}) - \beta(\mathbf{r}) I_d(\mathbf{r}, \hat{\mathbf{s}}),$$

which has the simple solution

$$I_d(\mathbf{r}, \hat{\mathbf{s}}) = \int S_d(\mathbf{r}', \hat{\mathbf{s}}) \exp\left[-\int_{\mathbf{r}\to\mathbf{r}'} (\kappa + \sigma_s)\, ds'\right] ds, \tag{CC-43}$$

where the main integral is along a straight path from the boundary of the medium to point $\mathbf{r}$ in the direction of $\hat{\mathbf{s}}$. For example, if there is only a simple point source at $\mathbf{r}_0$ with total strength Q_0, emitting isotropically across a tiny volume δV, equation (CC-43) becomes

$$I_d(\mathbf{r}, \hat{\mathbf{s}}) = \frac{Q_0}{4\pi|\mathbf{r}_0 - \mathbf{r}|^2} \exp\left[-\int_{\mathbf{r}_0\to\mathbf{r}} (\kappa + \sigma_s)\, ds'\right] \delta(\hat{\mathbf{s}} - \hat{\mathbf{s}}_0), \tag{CC-44}$$

where $\hat{\mathbf{s}}$ is a unit vector pointing from $\mathbf{r}_0$ toward $\mathbf{r}$, and use has been made of the fact that

$$\delta V = \delta A\, \delta s = \delta\Omega_0 |\mathbf{r}_0 - \mathbf{r}|^2\, \delta s,$$

where $\delta\Omega_0$ is the solid angle, with which δV is seen from $\mathbf{r}$. Equation (CC-44) can be used to calculate the direct contribution of Q_0 hitting a detector, and it can be used to determine the source term for the RTE of the scattered radiation as

$$\begin{aligned} S_1(\mathbf{r}, \hat{\mathbf{s}}) &= \frac{\sigma_s(\mathbf{r})}{4\pi} \int_{4\pi} I_d(\mathbf{r}, \hat{\mathbf{s}}')\Phi(\mathbf{r}, \hat{\mathbf{s}}', \hat{\mathbf{s}})\, d\Omega' \\ &= \frac{\sigma_s(\mathbf{r}) Q_0}{16\pi^2|\mathbf{r}_0 - \mathbf{r}|^2} \exp\left[-\int_{\mathbf{r}_0\to\mathbf{r}} (\kappa + \sigma_s) ds'\right] \Phi(\mathbf{r}, \hat{\mathbf{s}}_0, \hat{\mathbf{s}}). \end{aligned} \tag{CC-45}$$

The rest of the solution proceeds as before, with $I_n(\mathbf{r}_i, -\hat{\mathbf{s}}_i)$ found from equations (CC-45Chapter 21equation.0.0.45) and (21.82) as

$$I_n\left(\mathbf{r}_i, -\hat{\mathbf{s}}_i\right) = \frac{\sigma_s Q}{16\pi^2} \sum_j \int_{l_{\sigma,j}} \frac{e^{-\sigma_s|\mathbf{r}_0 - \mathbf{r}|}}{|\mathbf{r}_0 - \mathbf{r}|^2}\, dl', \tag{CC-46}$$

where the $l_{\sigma,j}$ are the straight paths the bundle travels between scattering events. Equation (CC-46Chapter 21equation.0.0.46) must be integrated numerically, and this can be done using a simple Newton-Cotes scheme. Alternatively, the integral can be obtained statistically from

$$I_n\left(\mathbf{r}_i, -\hat{\mathbf{s}}_i\right) = \frac{\sigma_s Q}{16\pi^2} \sum_j \frac{l_{\sigma,j}}{N_{\text{int}}} \sum_{k=1}^{N_{\text{int}}} \frac{e^{-\sigma_s|\mathbf{r}_0 - \mathbf{r}_k|}}{|\mathbf{r}_0 - \mathbf{r}_k|^2},$$

where the $\mathbf{r}_k$ are N_{int} random locations chosen uniformly along path $l_{\sigma,j}$. This was implemented in `RevMCps.f90`, choosing N_{int} (= `numint`) to be inversely proportional to the distance of the integration point from the source (or proportional to its relative importance). Results for detector flux for the as-supplied case (same as for `FwdMCps.f90`) are stored in `revmcps.dat` as:

```
no. of bundles     q_det     variance    rel.var.(%)
         10000  0.4614E-02  0.1057E-03       2.29
```

i.e., with only 10,000 bundles we achieved a much better variance then by using 16,000,000 bundles in `FwdMCps.f90`.

`RevMCps.f90`
The backward Monte Carlo equivalent of `FwdMCps`.

The documentation for this routine is not available.

Software Packages

`MONT3D`
This code, developed at Colorado State University by Burns et al. [15–19], calculates radiative exchange factors for complicated, three-dimensional geometries by the Monte Carlo method, as given by equations (8.15) and (8.21). Diffuse and specular view factors may be calculated as special cases. We provide here only a link to the Colorado State University web site, where documentation and codes are kept up-to-date: `http://www.colostate.edu/~pburns/monte.html`

`VIEW3D`
This code, developed at National Institute of Standards and Technology (NIST) by Walton [20], calculates radiative view factors with obstructions by adaptive integration. The package, as posted here, consists of 4 files:

1. The official NIST publication (`NISTIR-6925.pdf`),
2. A compressed file containing the program executables, help files, etc. (`v3d32exe.zip`),
3. A compressed file containing the program documentation (`V3D32doc.zip`), and
4. A compressed file containing sample data files (`IEA22dat.zip`).

For problems with and/or feedback for this package please address them directly to the author, George Walton (`gwalton@mailserver.nist.gov`).

`RADCAL`
This code, developed at NIST by Grosshandler [21,22] is a narrow band database for combustion gas properties, using tabulated values and theoretical approximations. The package consist of two files:

1. A user manual (NIST Technical Note `TN 1402.pdf`), and
2. a compressed file containing the program Fortran file and sample input and output files (`RADCAL.zip`).

For problems with and/or feedback for this package please address them directly to the author, William Grosshandler (`wgrosshandler@nist.gov`).

`EM2C`
This package contains a number of Fortran codes, developed at the Ecole Centrale de Paris by Soufiani and Taine [23], calculating statistical narrow band properties as well as narrow band *k*-distributions for CO_2 and H_2O, using the HITRAN92 database together with some proprietary French high-temperature extensions. The entire package is provided in the form of a compressed file containing the program Fortran files, data files and documentation (`em2c.zip`). For problems with and/or feedback for this package please address them directly to the authors, Anouar Soufiani (`soufiani@em2c.ecp.fr`) and/or Jean Taine (`taine@em2c.ecp.fr`).

`NBKDIR`
still under development
This package contains a number of Fortran codes, developed at the Pennsylvania State University and the University of California at Merced by the author and his students/postdocs A. Wang, G. Pal, and J. Cai, for the assembly of full spectrum *k*-distributions from a narrow band *k*-distributions database. At the time of printing `NBKDIR` contained data for five species (CO_2, H_2O, CO, CH_4, C_2H_4), as well as nongray soot, for temperatures up to 3000 K and pressures up to 80 bar. Spectroscopic data are taken from the HITEMP 2010 (CO_2, H_2O, CO) [24] and HITRAN 2008 (CH_4, C_2H_4) [25].

FVM2D
This Fortran77 code, developed at the University of Minnesota and Nanyang Technological University by Chai and colleagues [26–28], calculates radiative transfer in participating media using the finite-volume method of Chapter 17 for a two-dimensional, rectangular enclosure with reflecting walls and an absorbing, emitting, anisotropically scattering medium. For each surface the emittance and blackbody intensities must be specified; for the medium spatial distributions of radiation properties and blackbody intensities must be input. Calculated are internal incident radiation (G) and wall flux (q) fields. Can be used for gray problems or on a spectral basis. The package consists of two files:

1. A user manual (`RAT.pdf`), and
2. a compressed file containing the program Fortran files (`RATcode.zip`).

Four modules are needed to run `FVM2D`. These are `PARAM.FOR`, `COMMON.FOR`, `RATmain.FOR` and `ADAPT.FOR`. In this nomenclature, `RATmain.FOR` and `ADAPT.FOR` are the invariant part and the adaptation portion of the program, respectively. `COMMON.FOR` contains all the common block related variables, while `PARAM.FOR` contains the parameters for the program. These files are all contained in `RATcode.zip`, providing the 6 different versions of `ADAPT.FOR` corresponding to the 6 examples described in the manual. The manual as given is preliminary, i.e., two more examples dealing with irregular geometry and non-gray media, respectively, will be added at a later time.
For problems with and/or feedback for this package please address them directly to the author, John (Chee Kiong) Chai (`MCKChai@ntu.edu.sg`).

References

1. *MathWorks* MATLAB website, http://www.mathworks.com/products/matlab/.
2. Humlíček, J.: "Optimized computation of the Voigt and complex probability functions," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 27, p. 437, 1982.
3. Buckius, R. O., and D. C. Hwang: "Radiation properties for polydispersions: Application to coal," *ASME Journal of Heat Transfer*, vol. 102, pp. 99–103, 1980.
4. Mengüç, M. P., and R. Viskanta: "On the radiative properties of polydispersions: A simplified approach," *Combustion Science and Technology*, vol. 44, pp. 143–159, 1985.
5. Hageman, L. A., and D. Young: *Applied Iterative Methods*, Academic Press, 1981.
6. Modest, M. F.: "Further developments of the elliptic P_N-approximation formulation and its boundary conditions," *Numerical Heat Transfer – Part B: Fundamentals*, vol. 62, no. 2–3, pp. 181–202, 2012.
7. Wu, C. Y., and N. R. Ou: "Differential approximations for transient radiative transfer through a participating medium exposed to collimated irradiation," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 73, pp. 111–120, 2002.
8. Ferziger, J. H.: *Numerical Methods for Engineering Application*, 2nd ed., John Wiley & Sons, New York, 1981.
9. Modest, M. F., and R. S. Mehta: "Full spectrum k-distribution correlations for CO_2 from the CDSD-1000 spectroscopic databank," *International Journal of Heat and Mass Transfer*, vol. 47, pp. 2487–2491, 2004.
10. Modest, M. F., and V. Singh: "Engineering correlations for full spectrum k-distribution of H_2O from the HITEMP spectroscopic databank," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 93, pp. 263–271, 2005.
11. Denison, M. K., and B. W. Webb: "An absorption-line blackbody distribution function for efficient calculation of total gas radiative transfer," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 50, pp. 499–510, 1993.
12. Denison, M. K., and B. W. Webb: "Development and application of an absorption line blackbody distribution function for CO_2," *International Journal of Heat and Mass Transfer*, vol. 38, pp. 1813–1821, 1995.
13. Modest, M. F., and R. J. Riazzi: "Assembly of full-spectrum k-distributions from a narrow-band database; effects of mixing gases, gases and nongray absorbing particles, and mixtures with nongray scatterers in nongray enclosures," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 90, no. 2, pp. 169–189, 2005.
14. Modest, M. F.: "Backward Monte Carlo simulations in radiative heat transfer," *ASME Journal of Heat Transfer*, vol. 125, no. 1, pp. 57–62, 2003.
15. Burns, P. J.: "MONTE–a two-dimensional radiative exchange factor code," Technical report, Colorado State University, Fort Collins, 1983.
16. Maltby, J. D.: "Three-dimensional simulation of radiative heat transfer by the Monte Carlo method," M.S. thesis, Colorado State University, Fort Collins, CO, 1987.
17. Burns, P. J., and J. D. Maltby: "Large-scale surface to surface transport for photons and electrons via Monte Carlo," *Computing Systems in Engineering*, vol. 1, no. 1, pp. 75–99, 1990.
18. Maltby, J. D., and P. J. Burns: "Performance, accuracy and convergence in a three-dimensional Monte Carlo radiative heat transfer simulation," *Numerical Heat Transfer – Part B: Fundamentals*, vol. 16, pp. 191–209, 1991.
19. Zeeb, C. N., P. J. Burns, K. Branner, and J. S. Dolaghan: "User's manual for Mont3d – Version 2.4," Colorado State University, Fort Collins, CO, 1999.
20. Walton, G. N.: "Calculation of obstructed view factors by adaptive integration," Technical Report NISTIR–6925, National Institute of Standards and Technology (NIST), Gaithersburg, MD, 2002.
21. Grosshandler, W. L.: "Radiative transfer in nonhomogeneous gases: A simplified approach," *International Journal of Heat and Mass Transfer*, vol. 23, pp. 1447–1457, 1980.

22. Grosshandler, W. L.: “RADCAL: a narrow-band model for radiation calculations in a combustion environment,” Technical Report NIST Technical Note 1402, National Institute of Standards and Technology, 1993.
23. Soufiani, A., and J. Taine: “High temperature gas radiative property parameters of statistical narrow-band model for H_2O, CO_2 and CO, and correlated-k model for H_2O and CO_2,” *International Journal of Heat and Mass Transfer*, vol. 40, no. 4, pp. 987–991, 1997.
24. Rothman, L. S., I. E. Gordon, R. J. Barber, H. Dothe, R. R. Gamache, A. Goldman, V. I. Perevalov, S. A. Tashkun, and J. Tennyson: “HITEMP, the high-temperature molecular spectroscopic database,” *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 111, no. 15, pp. 2139–2150, 2010.
25. Rothman, L. S., I. E. Gordon, A. Barbe, D. C. Benner, P. F. Bernath, M. Birk, V. Boudon, L. R. Brown, A. Campargue, J.-P. Champion, K. Chance, L. H. Coudert, V. Dana, V. M. Devi, S. Fally, J.-M. Flaud, R. R. Gamache, A. Goldman, D. Jacquemart, I. Kleiner, N. Lacome, W. J. Lafferty, J.-Y. Mandin, S. T. Massie, S. N. Mikhailenko, C. E. Miller, N. Moazzen-Ahmadi, O. V. Naumenko, A. V. Nikitin, J. Orphal, V. I. Perevalov, A. Perrin, A. Predoi-Cross, C. P. Rinsland, M. Rotger, M. Simeckova, M. A. H. Smith, K. Sung, S. A. Tashkun, J. Tennyson, R. A. Toth, A. C. Vandaele, and J. V. Auwera: “The HITRAN 2008 molecular spectroscopic database,” *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 110, pp. 533–572, 2009.
26. Chai, J. C., H. S. Lee, and S. V. Patankar: “Finite volume method for radiation heat transfer,” *Journal of Thermophysics and Heat Transfer*, vol. 8, no. 3, pp. 419–425, 1994.
27. Chai, J. C., H. S. Lee, and S. V. Patankar: “Treatment of irregular geometries using a Cartesian coordinates finite-volume radiation heat transfer procedure,” *Numerical Heat Transfer – Part B: Fundamentals*, vol. 26, pp. 225–235, 1994.
28. Chai, J. C., G. Parthasarathy, H. S. Lee, and S. V. Patankar: “Finite volume method radiative heat transfer procedure for irregular geometries,” *Journal of Thermophysics and Heat Transfer*, vol. 9, no. 3, pp. 410–415, 1995.