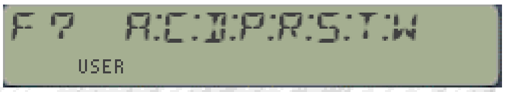
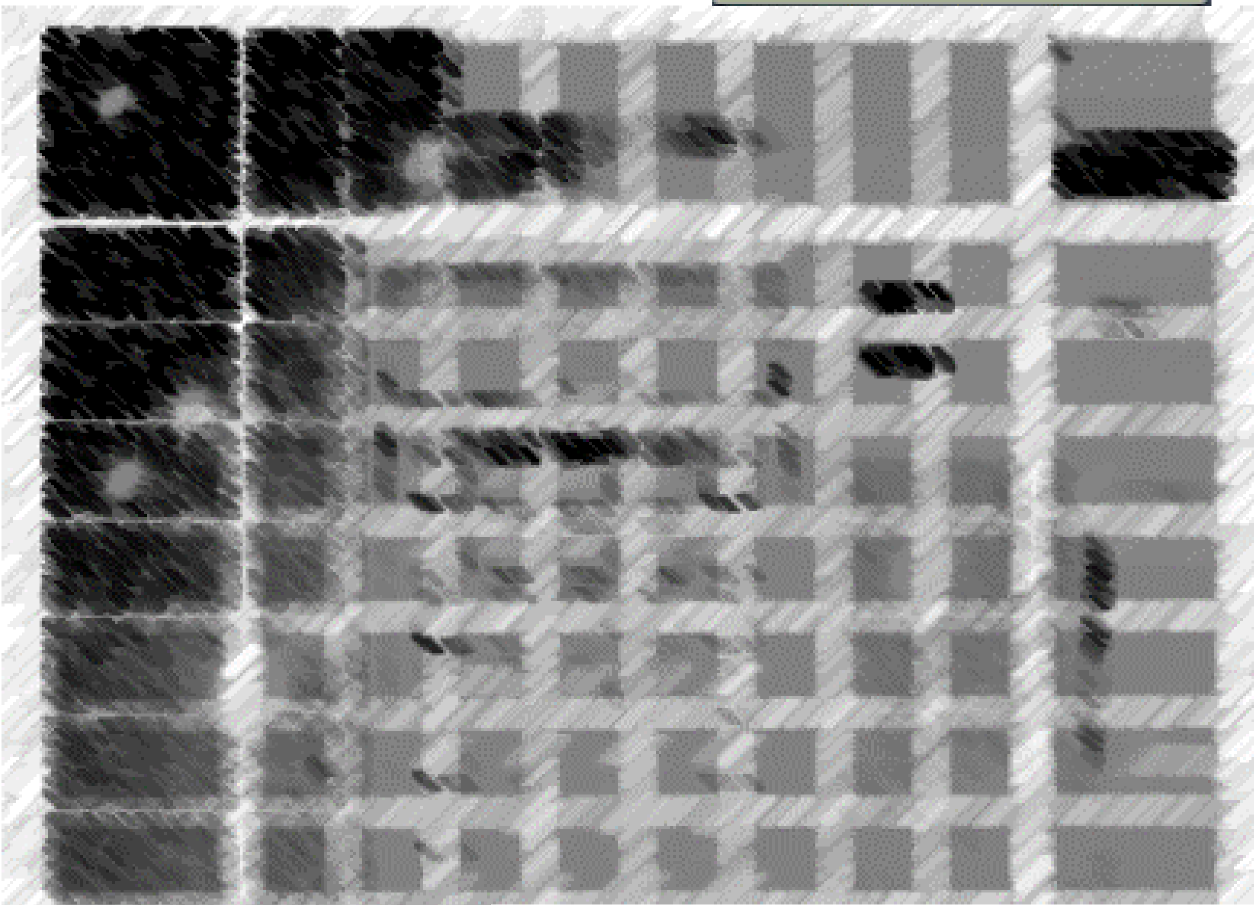
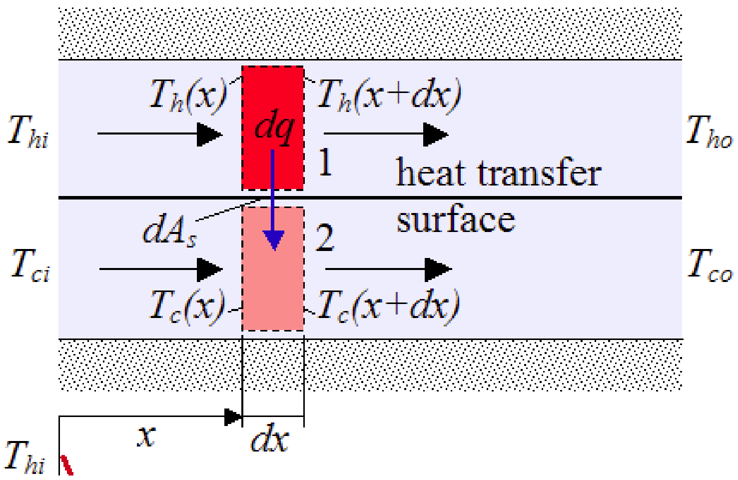
Heat Transfer Applications ROM

HP-41 Module

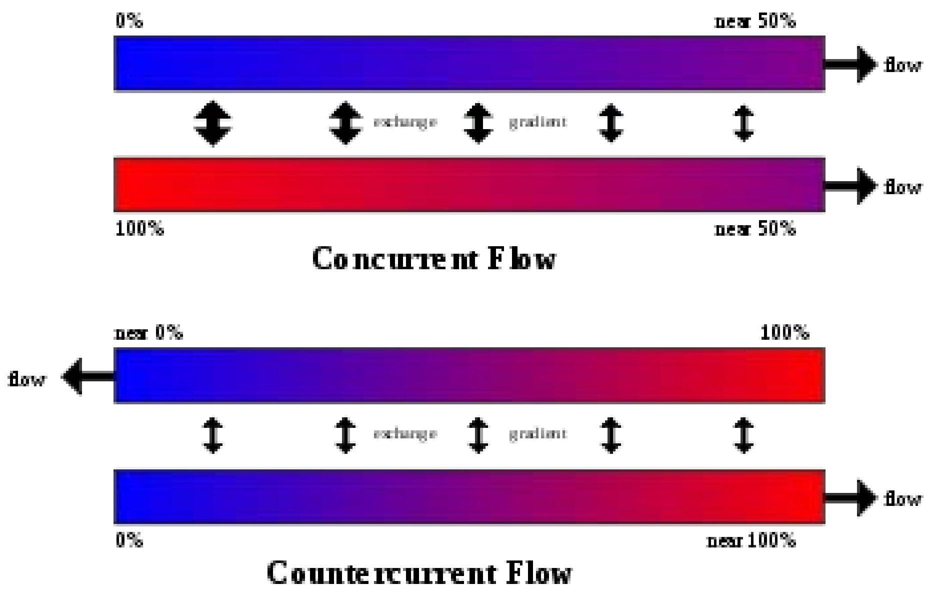






Ángel M. Martin Cañas. --- *February 2020*

This compilation revision 1.1.1



Front cover image taken from: https://www.dreamstime.com/royalty-free-stock-photography-mathematics-background-image20849947

Screen captures taken from V41, Windows-based emulator developed by Warren Furlow.
See www.hp41.org

Introduction

The purpose of this module is to make available a few enhancements and improvements made to several Heat Transfer programs available in other sources, such as the author's ETSII Collection s, and HP's Thermal & Transport Science Pack.

The enhancements include several MCODE functions that replace equivalent FOCAL routines with faster execution and more accurate results. The manual also pays more attention to the theoretical background for several approaches used, related to techniques and applicability of the results.

As usual in this line of work, the magnitude units play an important role in the resolution of the problems and often get in the way. This module works together with the Unit Conversion ROM, holding the Unit Management System and other utilities.

Besides the FOCAL routines I've also included the listings of the MCODE functions called by them because they're crucial to understanding the program flow and the methodology used in the calculations. Feel free to ignore them if MCODE is not your cup of tea – but it's always interesting to have a complete documentation of the work.

Module sections

The main sections in the module are :

The First two are modified versions of the programs in the Thermal Pack, using dedicated MCODE functions and the Unit Conversion ROM.

The Transient analysis in Cylinders provides a parallel study to the plaques included in the ETSII Collection. It requires the SandMath for the Bessel functions support.

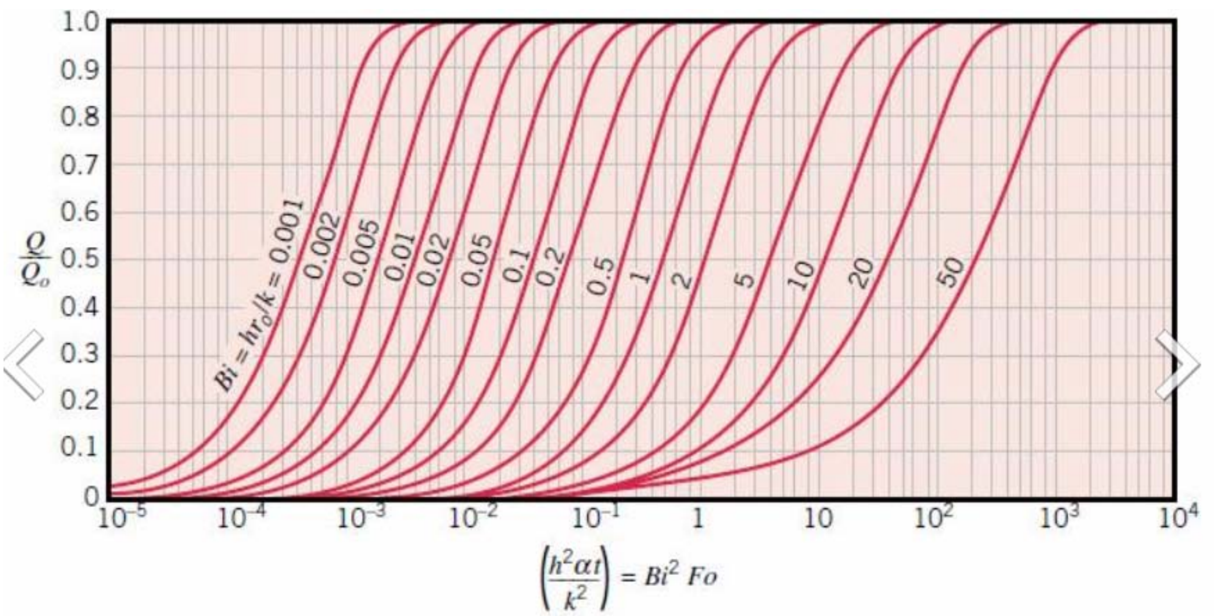
The View Factors library comprises a collection of 23 view factors for combinations of the basic geometric shapes (discs, annular rings, cylinders, spheres, plates and planes). This is an extensive enlargement of the material in the ETSII Collection as well.

ROM Function Tables.

Without further ado, let's see the functions included in the module. Refer to the individual function descriptions later on for details on the syntax and use instructions.

XROM	Function	Description	Input	Author
16,00	**-HEAT EXCH**	*Section Header*	*n/a*	*n/a*
16,01	**"HXIN"**	Heat Exchanger Input	Prompts for data	*Ángel Martin*
16,02	**"HXOUT"**	Heat Exchengers Output	Under prgm control	*Ángel Martin*
16,03	**QC**	Tranferred Heat – Counterflow	k1,k2,T1,T2 in stack	*Ángel Martin*
16,04	**QP**	Transferred Heat - Parallel	k1, k2, T1, T2 in stack	*Ángel Martin*
16,05	**T1C**	Temp. Fluid 1- Counterflow	k1, k2, T1, T2 in stack	*Ángel Martin*
16,06	**T1P**	Temp. Fluid 1- Parallel	k1, k2, T1, T2 in stack	*Ángel Martin*
16,07	**T2C**	Temp. Fluid 2 - Counterflow	k1, k2m T1, T2 in stack	*Ángel Martin*
16,08	**T2P**	Temp. Fluid 2 - Parallel	k1, k2, T1, T2 in stack	*Ángel Martin*
16,09	**?RTN**	Checks for subroutines	YES/NO Skip if true	*Ángel Martin*
16,10	**"HEATX"**	HP's Heat Exchangers	*See Thermal Pack Man.*	*HP Co.*
16,11	**MEM?**	Returns Available memory	none	*Ángel Martin*
16,12	**-RV FACTORS**	*Section Header*	*n/a*	*n/a*
16,13	**ΣVF _**	Function Launcher	F? A:C:D:P:R:S:T:W	*Ángel Martin*
16,14	**AAE**	Annular Ring to Ring		*Ángel Martin*
16,15	**CAE**	Cylinder to Annular Ring		*Ángel Martin*
16,16	**CCF**	Cylinder to parallel Cylinder		*Ángel Martin*
16,17	**CCCF**	Concentric Cylinder to Cylinder		*Ángel Martin*
16,18	**CC11F**	Coaxial Cylinders - Itself		*Ángel Martin*
16,19	**CC12F**	Coaxial Cylinders - Interior		*Ángel Martin*
16,20	**CSTF**	Cylinder to Frontal Strip		*Ángel Martin*
16,21	**CWCF**	Cylinder Wall to Cap		*Ángel Martin*
16,22	**DAF**	Disc to Annular Ring		*Ángel Martin*
16,23	**DCSF**	Disc to Cylinder Surface		*Ángel Martin*
16,24	**DDF**	Disk to Disk		*Ángel Martin*
16,25	**DSF**	Disk to Sphere		*Ángel Martin*
16,26	**PAF**	Patch tp Annular Ring (Parall.)		*Ángel Martin*
16,27	**PPRF**	Rectangular Plate to Plate		*Ángel Martin*
16,28	**PPSF**	Square Plate to Plate – Same		*Ángel Martin*
16,29	**PQSF**	Square Plate to Plate - Unequal		*Ángel Martin*
16,30	**RDF**	Rod to Concentric Disc		*Ángel Martin*
16,31	**RR90F**	Perpend. to Horiztl. Rectangles		*Ángel Martin*
16,32	**SDF**	Sphere to Disk		*Ángel Martin*
16,33	**SSF**	Sphere to Sphere		*Ángel Martin*
16,34	**SSCF**	Concentric Sphere to Sphere		*Ángel Martin*
16,35	**TPSF**	Tilted Plane to Sphere		*Ángel Martin*
16,36	**WCF**	Wire to Cilynder		*Ángel Martin*

16,37	**-CYL TEMP**	Section Header	*n/a*	*n/a*
16,38	**A<>RG _ _**	Swap ALPHA and Registers		*Ángel Martin*
16,39	**ST<>RG _ _**	Swap Stack and Registers	Prompts for Reg. index	*Ángel Martin*
16,40	**"?"**	Prompting Routine	Reg. Index in X	*Ángel Martin*
16,41	**"ΣI"**	Infinite Sum	F. name in ALPHA	*Ángel Martin*
16,42	**"aJ"**	General Term	n in X	*Ángel Martin*
16,43	**"BIOT"**	Roots of Bessel Equation	Bi in Y, #Roots in X	*JM Baillard*
16,44	**"JN"**	Function to Solve	n in Y, x in X	*Ángel Martin*
16,45	**"TRT"**	Transients in Cylinders	Prompts for Data	*Ángel Martin*
16,46	**-BLACK BODY**	Section header	*n/a*	*n/a*
16,47	**"BLKBDY"**	*Black-body Thermal Radiation*	Prompts for data	*Ángel Martin*
16,48	**"EbL"**	Emissive Power at WVL	Driver for EVL	*Ángel Martin*
16,49	**EVL**	Auxiliart for EbL	T in Y, λ in X	*Ángel Martin*
16,50	**PLNK**	Planck Integral	Lower limit in X	*Ángel Martin*
16,51	**SZE?**	Returns Available Sizes	none	*Ángel Martin*



Black Body Thermal Radiation

The program included in the HEATEX module is an enhanced version of the original from HP's "Thermal Pack". The modifications are as follows:

- The use of the Unit Conversion ROM is not only for the flexible input/output unit selection – but also as a *direct entry of the physical constants* involved in the expressions.
- MCODE version of the *Energy density function*, faster and more accurate than the FOCAL counterpart in the original ROM
- MCODE version of the *Planck's Integral calculation* routine, faster and more accurate that the original FOCAL routine.

The program uses the same well-known relationships for the calculation of Planck's integral, as described below. An alternative version using the numerical integration **FINTG** is also included

Black-body radiation Law

It has been found that at a uniform temperature above absolute zero an enclosure always emits radiation as a result of atomic and molecular agitation. The total Energy,M(T), emitted over all wavelengths in a *unit time* from a *unit area* of the wall on the enclosure is a function of its temperature only and is

M(T) = σ. T^4 [W/m^2],

$$M = \left(\frac{2\pi^5 k^4}{15h^3c^2}\right)T^4 \quad \mathrm{W\ m^{-2}}$$

where T is the absolute temperature in degrees Kelvin. This is known as the Stefan-Boltzmann Fourth Power Law and σ the Stefan-Boltzmann constant. σ = 2π^4 k^4 / 15 h^3 c^2

The distribution of emissive power is described by the function, Bλ(T) and we say that the energy in thewavelength range λ to (λ + dλ) is: Bλ(T).dλ, and therefore: M(T) = ∫(0,∞)Bλ(T).dλ

It has also been found experimentally thatfor a given value of absolute temperature T,Bλ(T) has a maximum at wavelength λ**m,** and that this maximum is proportional to the fifth power of **T,** that is:

[Bλ(T)]max = β.T^5

whereβ is the proportionality constant. The wavelength where this maximum occurs has a definiterelationship for all temperatures:

λm T = Wien's Displacement constant

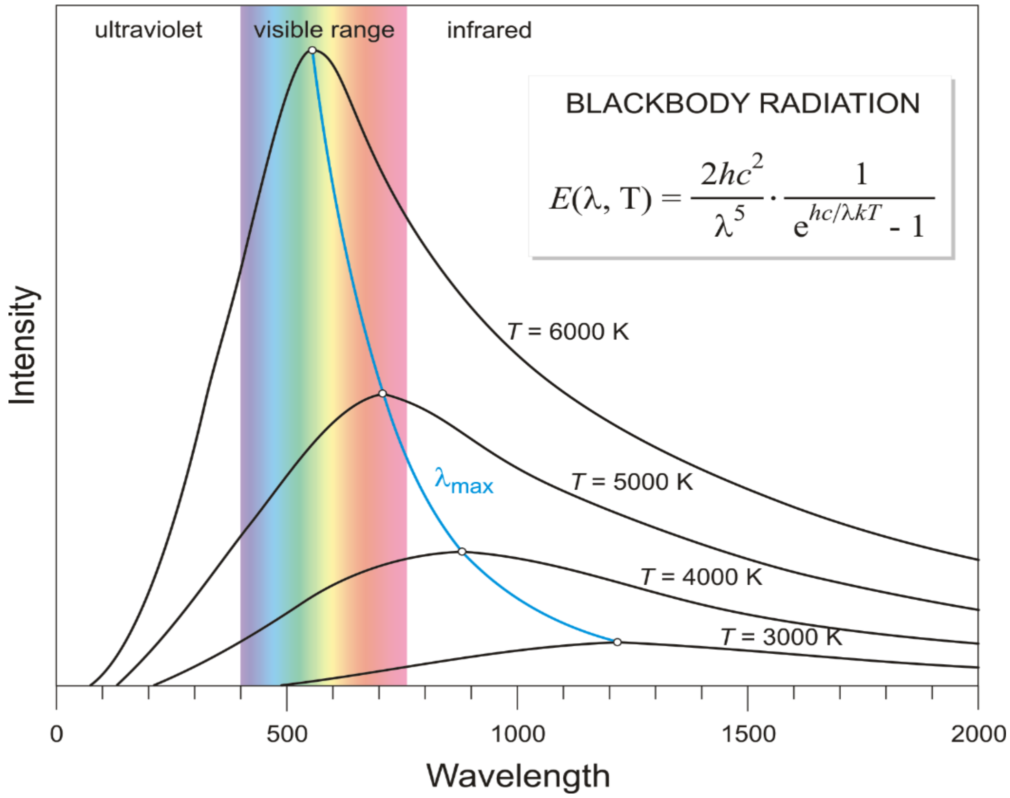
It is evident that these maximums shift toward the shorter wavelengths as the temperature rises.

From Wien's displacement Law to Planck's Law.

Wien's Displacement law says that the distribution of emissive power(per unit volume) with wavelength obeys the following relationship:

$$B_{\lambda,T} = \lambda^{-5} F\left(\frac{1}{\lambda T}\right)$$

Rayleigh and Jeans, applying the laws of classical mechanics, obtained an expression for F() that agreed well with experiments in the longer wavelength region – but it was not until 1901 that Planck, using a quantum hypothesis, was able to obtain an expression accurate over the whole length of the black-body spectrum.



For convenience, the constants are grouped together to form the first and second radiation constants c1 and c2 as follows:

c_1 = 2π h.c^2 (watts. Cm^2) = 2h.c^2 (watts.cm^2/stereo-rad)
c_2 = h.c/k (cm.K)

such that: $B_{\lambda,T} = \frac{c_1 \lambda^{-5}}{\left(e^{c_2/\lambda T} - 1\right)}$ $\qquad B_{\lambda,T} = \frac{2\pi hc^2 \lambda^{-5}}{\left(e^{hc/\lambda kT} - 1\right)}$,

The emissive power per unit volume per unit *wavelength*per unit solid angle(removes π) is given by the expression on the left, whereas using a per unit *frequency* instead is on the right, using ν= c / λ :

$$B_\lambda(T) = \frac{2hc^2/\lambda^5}{exp(hc/\lambda kT) - 1} \quad ;\quad B_\nu(T) = \frac{2h\nu^3/c^2}{\exp(h\nu/kT) - 1}$$

Warning: Bν and Bλ do not peak at the same place, because ν and λ are not linearly related! This is a common source of confusion and somehow counterintuitive - but that's quantum mechanics for you ;-)

Program description.

The following variables are obtained by this enhanced version of the program:

1. Total emissive power, Eb, at a temperature T –> Eb = σ T^4
2. Wavelength λm for maximum emissivity -> λm = c3 /T (*)
3. Emissive power, EbL, at a temperature T and wavelength λ
4. Radiant output Eb12 in a given wavelength range, [λ1, λ2], defined as:

$$\mathrm{W/m^2} = (\pi)\int_{\lambda_2}^{\lambda_1} \frac{2hc^2/\lambda^5}{e^{hc/\lambda kT}-1}d\lambda$$

(*) c3 is the Wien's displacement constant, and its value is derived from Planck's law to be:

c3 = h.c / k [5 + **W**(-5.e^-5)] ~= 2.8977684 E-3 [m.K/s^2]

where **W** is the *Lambert function*. See the appendix at the end of this section for a detailed calculation

The MCODE routine **EVL** calculates the emissive power per unit volume per unit wavelength assuming all magnitudes are entered in SI units. Thus, the mission of routine "**EbL**" is preparing the correct units first and then call the MCODE function **EVL**.

Unit Conversion and Selection

Looking at the expressions involved we can already see that the units play an important role in the program, as they directly affect the constants in the equations. The Unit Conversion ROM comes to the rescue, and we'll be using the **INPUT** and **OUTPUT** routines appropriately. Also remember that the MCODE function **EVL** (energy per unit volume per unit wavelength) requires that all magnitudes are expressed in SI units, therefore we'll use the function **>SI** to do the conversions before calling **EVL**.

The original program calculates the constants c1 and c2 "on the go", i.e. applying the conversion for each magnitude as they are being entered. The intermediate results are placed in data registers R13 and R15, which eventually hold the final value of the constants as well.

Integrating the Plank equation

To calculate the radiant output for a wavelength range we need to integrate the emissive power EbL between the range limit wavelengths λ1 and λ2:

Eb12 = $\int$(λ1, λ2) Eb(λ) dλ = $\int$(λ1,∞) - $\int$(λ2,∞)

The program uses the following relationships to calculate the value of the improper integral:

$\int(x,\infty)\frac{t^3}{e^t-1}dt$ = Σ [exp(-nx)[(nx)^3+3(nx)^2+6(nx)+6]/n^4] ; n=1,2...

= Σ {nx.[nx.(nx + 3) + 6] + 6} / n^4 . exp(nx) ; n=1,2,...

A dedicated MCODE function **PLNCK** does this calculation in the program. Note that it'll also use SI units, and that the integration limits need to align with the *change of variable*required to match the formulas above, from λto u = h.c/T.K.λ, thus:dλ = - (h.c) du /T.K. u^2

$$\text{Eb12} = 2\pi\frac{(K.T)^4}{c^2.h^3}[\int(1/u2,\infty)\ f(u)\ du - \int(1/u1,\infty)\ f(u)\ du\ ]\ ,$$

with $f(u) = \frac{u^3}{e^u - 1}$and integration limits: u1= hc/KT.λ1 ; u2 = hc/KT.λ2

Note: An alternative to the Planck's integral formula could be to use integrate the expression numerically with **FINTG** (in the SandMath, or **INTEG** in the Advantage Pac). The results are equivalent, but it takes much longer to complete, especially with high accuracy settings (number of decimal points in FIX). The only advantage is shorter code – but it's outweighed by the dependency introduced on the other module.

Appendix: Deriving the Wien's Displacement constant.

We can use Planck's equation to derive Wien's Displacement law. To do that we'll obtain the maximun on the wavelength by calculating derivatrive over the wavelenght and equal it to zero, as follows:

$$B(l,T) = \frac{c1}{l^5}.\left(e^{\frac{c2}{lt}} - 1\right)^{-1}$$

$$\frac{dB}{dl} = -5c1\ l^{-6}\left(e^{\frac{c2}{lT}} - 1\right)^{-1} + c1.l^{-5}.\left(e^{\frac{c2}{lT}} - 1\right)^{-2} e^{\frac{c2}{lT}}.\frac{c2}{T\ l^2}$$

Making it zero it results:

$$5\ l^{-6}\left(e^{\frac{c2}{lT}} - 1\right) = \frac{c2}{T} l^{-7} e^{\frac{c2}{Tl}}$$

$$5\left(e^{\frac{c2}{lT}} - 1\right) = \frac{c2}{Tl} e^{\frac{c2}{Tl}}$$

And making the change of variable:u = c2/Tλ

$$5(e^u - 1) = u.e^u$$

This equation has the solution based on Lambert's W function, as follows:

u = 5 + W(-5.e^{-5}), therefore λ = c2/T.u = c2 / T.(5+W(-5.e^{-5}))

and recalling the deficinion of c2:

λm = h.c / k.T(5+W(-5.e^{-5})) →λm.T = h.c / k.(5+W(-5.e^{-5}))

Numerically: wien = 2,897768447 E-3 (m.K/s^2)

Example.

What percentage of the radiant output of a lamp is in the visible range (0.4 to 0.7 microns) if the filament of the lamp is assumed to be a black-body at 2200 C? What is the percentage at 2300 C? What is the ratio of light for thee two temperatures?

XEQ "BLKBDY"	UNITS E?
J, R/S	UNITS WVL?
MIC, R/S	UNITS AREA?
CM2, R/S	UNITS TIME?
S, R/S	T=?
2200, R/S	UNITS T?
C, R/S	Eb=212.141 J/S*CM2
R/S	LMAX=1.172 MIC
R/S	L1=?
0.4, R/S	EbL=1.764 J/S*CM2*MIC
R/S	L2=?
0.7, R/S	EbL=54.745 J/S*CM2*MIC
R/S	Eb12=6.663 J/S*CM2
R/S	T=?
2300, R/S	UNITS T?
C, R/S	Eb=248.589 J/S*CM2
R/S	LMAX=1.126 MIC
R/S	L1=?
R/S	L2=?
R/S	Eb12=9.704 J/S*CM2

To find the percentage, divide the emissive power in the visible range by the total power:

At 2200 C:
6.663, ENTER^, 212.141, / ,100, * => 3.141 %

At 2300: C
9.704, ENTER^, 248.589, / , 100, * => 3.904 %

To find the ratio of power at 2200 C to that at 2300 C, divide the visible range powers:

6.663, ENTER^, 9.704, / => 0.6867

Program Listing:

```
1   *LBL "BLKBDY"
2   SF 00
3   SF 21          ; halting AVIEW
4   SIZE?          ; current size
5   20             ; required size
6   X>Y?           ; have enough?
7   PSIZE          ; no, resize
8   CLA
9   ASTO 02
10  "J"            ; unit: Joule
11  ASTO 01
12  "E"            ; Energy
13  RCL 16         ; currentvalue
14  XEQ 01         ;  Energyunits
15  STO 12         ; [E] SI-factor
16  STO 15         ; [E] SI-factor
17  RDN            ; unit string
18  STO 16         ; save for later
19  "M"            ; unit: Meters
20  ASTO 01
21  "WVL"          ; wavelength
22  RCL 17         ; currentvalue
23  XEQ 01         ; wavelengthunits
24  STO 10         ; [λ] SI-factor
25  STO 13         ; [λ] SI-factor
26  STO 14         ; [λ] SI-factor
27  X^2            ; [λ]^2
28  X^2            ; [λ]^4
29  ST* 12         ; [E].[λ]^4
30  RDN            ; unit string
31  STO 17         ; save for later
32  "M2"
33  ASTO 01        ; unit: square meter
34  "AREA"         ; area
35  RCL 18         ; current value
36  XEQ 01         ; AREA units
37  ST/ 12         [E].[λ]^4/[A]
38  ST/ 15         [E]/[A]
39  RDN            ; unit string
40  STO 18         ; save for later
41  "S"            ; unit: second
42  ASTO 01
43  "TIME"         ; time
44  RCL 19         ; current value
45  XEQ 01         ; TIME units
46  ST/ 12         ; [E].[λ]^4/[A].[t]
47  ST/ 15         ; [E]/[A] / [t]
48  RDN            ; unit string
49  STO 19         ; save for later
50  <H>            ; Planck's const.
51  <c>            ; speed of light
52  *
53  <K>            ; Boltzmann const.
54  /
55  STO 11         ; h.c/K
56  ST* 13         ; [λ].h.c/K
57  289777 E-9     ; Wien's displcmt.
58  ST* 14         ; [λ]. wien
59  SG             ; Stefan-Boltzmann
60  ST* 15         ;σ. [E]/[A] / [t]
61  *LBL A         ; new calculation
62  CLA
63  ASTO 02
64  "K"            ; unit:Kelvin
65  ASTO 01
66  2
67  STO 00         ; will store in R03 (!)
68  "T"            ; Temperature
69  XROM"INPUT"; save T in R02
70  X^2            ; T^2
71  X^2            ; T^4
72  RCL 15         ;σ. [E]/[A] / [t]
73  *  ;Ebl result
74  "Eb"           ; EBL string
75  XEQ 03 ; output value
76  FC? 22         ; entered?
77  GTO 00         ; no, skip
78  "LMAX="
79  RCL 14         ; [λ]. Wiens
80  RCL 03         ; T
81  /
82  ARCL X         ; append result
83  >" "           ; buffer
84  ARCL 17        ; append units WVL
85  AVIEW          ; show result
86  ADV            ; prnter advance
87  *LBL 00
88  7
89  STO 00         ; will store in R08
90  "L1"
91  SF 01          ; needed by INPUT
92  XROM "INPUT"; input λ1 in R08
93  XROM"EbL"      ; do the math
94  XEQ 02 ; output EbL1 result
95  "L2"
96  SF 01
97  XROM "INPUT"; input λ2 in R09
```

```
XROM "EbL"       ; do the math
XEQ 02           ; output EbL2 result
 RCL 11          ;h.c/K
RCL 03           ; T
/                ;h.c/K.T
RCL 10           ; [λ] SI-factor
 *
 STO 00          ;[λ].h.c/K.T
RCL 09           ; λ2
/                ; lower limit
 PLNK            ; improper integral
 RCL 00          ; [λ].h.c/K.T
 RCL 08          ; λ1
/                ; lower limit
PLNK             ; improper integral
 -
 ABS             ; ensure positive
 PI
 ST+ X           ; 2π
*
RCL 03           ; T
<K>              ; Boltzmann const.
*          ; T.K
 X^2             ; (Tk)^2
 X^2             ; (TK)^4
*
<c>              ; speed of light
<H>              ; Planck's const.
 *         ;c.h
 X^2             ; (c.h)^2
 /               ; (T.K)^4 / (c.h)^2
<H>              ; Planck's const.
/                ; (T.K)^4/K.c^2.h^3
 RCL 12          ; [E].[λ]^4/[A]
 *
 RCL 10          ; [λ] SI-factor
 X^2             ; [λ]^2
 X^2             ; [λ]^4
/
 "Eb12"          ; band power
SF 22            ; forced entry
XEQ 03           ; output result
GTO A            ; repeat calculation
*LBL "EbL"
RCL 03           ; T
 RCL 10          ; [λ] SI-factor
 /
 X<>Y            ; λ
 EVL             ; SI conditions
 RCL 12          ; E].[λ]^4
*
RTN
*LBL 02          ; outputsEbL
 SF 01
 "EbL"           ; forEbL
*LBL 03          ; outputs ALPHA
FC? 22
 CF 01
 FC? 22
 RTN
>"="
 ARCL X          ; append value
 AVIEW           ; show it
 CLA
 ARCL 16         ; Energy units
>"/"             "[E]/"
 ARCL 19         ; Time units
>"*"             "[E]/[t]*"
ARCL 18          ; Area units
FC?C 01
 GTO 00
>"*"             "[E]/[t]*[A]*"
 ARCL 17         ; WVL units
*LBL 00
 AVIEW           ;"[E]/[t]*[A]*[A]"
 ADV             ; printer advance
RTN
*LBL 01          ; input value & Units
 E               ; amount = 1
 ASTO T          ; save magnitude in T
"UNITS "
ARCL T           ; append magnitude
> "?"
 AON
 CF 23           ; reset data entry
 PROMPT          ; halt for input
 AOFF
FC? 23           ; entered?
CLA              ; no, reset
FC? 23           ; entered?
 ARCL Y          ; no, append default
 ASTO Y          ; savestring
> "-"            ; cpnvertto:
 ARCL 01         ; SI unitsstring
 SF 25           ; error traping
SI>              ; convertto SI
CLA              ; reset ALPHA
ARCL T           ; magnitude
FC?C 25          ; did error?
 GTO 01          ; yes, try again
 END
```

MCODE Listing for EVL and PLNCK

Emissive Energy Routine

Header	AA1C	08C	"L"	Emmisive Energy
Header	AA1D	016	"V"	per unit volume
Header	AA1E	005	"E"	
EVL	AA1F	0F8	READ 3(X)	λ
	AA20	2EE	?C#0 ALL	zero data?
	AA21	3A0	?NC RTN	yes, abort.
	AA22	10E	A=C ALL	
in SI units (!)	AA23	0B8	READ 2(Y)	T
	AA24	351	?NC XQ	(includes SETDEC)
	AA25	050	->14D4	[CHK_NO_S1]
	AA26	135	?NC XQ	λ.T
	AA27	060	->184D	[MP2_10]
	AA28	04E	C=0 ALL	
	AA29	130	LDI S&X	
	AA2A	098	CON:	100-98 = 02
	AA2B	276	C=C-1 XS	this SETS carry!!!
	AA2C	158	M=C ALL	S&X in M
	AA2D	35C	PT= 12	
	AA2E	050	LD@PT- 1	
	AA2F	110	LD@PT- 4	
	AA30	0D0	LD@PT- 3	
	AA31	210	LD@PT- 8	
	AA32	1D0	LD@PT- 7	
	AA33	1D0	LD@PT- 7	h.c/k
	AA34	150	LD@PT- 5	
	AA35	050	LD@PT- 1	
	AA36	0D0	LD@PT- 3	
	AA37	150	LD@PT- 5	
	AA38	210	LD@PT- 8	
	AA39	1D0	LD@PT- 7	0.0143877513587408914698598607
	AA3A	110	LD@PT- 4	
	AA3B	24D	?NC XQ	{C,M} / {A,B}
	AA3C	060	->1893	[X/Y13]
	AA3D	048	SETF 4	e^x-1
	AA3E	035	?NC XQ	
	AA3F	068	->1A0D	[EXP13]
	AA40	089	?NC XQ	exp(hc/kλT)-1
	AA41	064	->1922	[STSCR]
	AA42	0F8	READ 3(X)	λ
	AA43	10E	A=C ALL	
	AA44	135	?NC XQ	λ^2
	AA45	060	->184D	[MP2_10]
	AA46	13D	?NC XQ	λ^4
	AA47	060	->184F	[MP1_10]
	AA48	0F8	READ 3(X)	λ
	AA49	13D	?NC XQ	λ^5
	AA4A	060	->184F	[MP1_10]
	AA4B	0D1	?NC XQ	
	AA4C	064	->1934	[RCSCR]
	AA4D	149	?NC XQ	λ^5.[exp(hc/kλT) - 1]
	AA4E	060	->1852	[MP2_13]

	AA4F	04E	C=0 ALL	
	AA50	130	LDI S&X	
	AA51	084	CON:	100-84 = 16
	AA52	276	C=C-1 XS	this SETS carry!!!
	AA53	158	M=C ALL	S&X in M
	AA54	35C	PT= 12	
	AA55	0D0	LD@PT- 3	
	AA56	1D0	LD@PT- 7	
	AA57	110	LD@PT- 4	2π.h..c^2 =
	AA58	050	LD@PT- 1	3.741771182054-16
	AA59	1D0	LD@PT- 7	
	AA5A	1D0	LD@PT- 7	
	AA5B	050	LD@PT- 1	
	AA5C	050	LD@PT- 1	
	AA5D	210	LD@PT- 8	
	AA5E	090	LD@PT- 2	
	AA5F	010	LD@PT- 0	
	AA60	150	LD@PT- 5	3.7417711820578882504205510154
	AA61	110	LD@PT- 4	
	AA62	24D	?NC XQ	{C,M} / {A,B}
	AA63	060	->1893	[X/Y13]
	AA64	331	?NC GO	Overflow, DropST, FillXL & Exit
	AA65	002	->00CC	[NFRX]

$$F(\lambda)= \frac{2\pi hc^2}{\lambda^5\left(exp\left(\frac{hc}{k\lambda T}\right)-1\right)}$$

Input parameters: Temperature T in Y-register and wavelengthλ in X-register.

No data registers are used.

Warning: The calculation assumes SI units for T and λ

Planck Integral Routine.

Single Input parameter is the Temperature in the X-register.

Note how the iterations will continue until the relative difference between them is less that 10-8, regardless of the number of decimal digits set in the machine.

Header	ABFF	08B	"K"	
Header	AC00	00E	"N"	Integ(x,infinity) t^3/(e^t-1) dt
Header	AC01	00C	"L"	Σ [exp(-nx)[(nx)^3+3(nx)^2+6(nx)+6]/n^4]; n=1,2…
Header	AC02	010	"P"	= {nx.[nx.(nx + 3) + 6] + 6} / n^4 . exp(nx)
PLNK	**AC03**	**2A0**	**SETDEC**	
	AC04	04E	C=0 ALL	
	AC05	128	WRIT 4(L)	*initial index*
	AC06	070	N=C ALL	*initial result*
NXTIND	AC07	138	READ 4(L)	current index
	AC08	1E1	?NC XQ	*Increment C*
	AC09	100	->4078	*[INCC10]*
	AC0A	128	WRIT 4(L)	*increment index*
	AC0B	0F8	**READ 3(X)**	*x*
	AC0C	13D	?NC XQ	
	AC0D	060	->184F	*[MP1_10]*
	AC0E	089	?NC XQ	*nx*
	AC0F	064	->1922	*[STSCR]*
	AC10	04E	C=0 ALL	
	AC11	35C	PT=12	*build "3" in C*
	AC12	0D0	LD@PT- 3	
	AC13	025	?NC XQ	*nx + 3*
	AC14	060	->1809	*[AD1_10]*
	AC15	0D1	?NC XQ	
	AC16	064	->1934	*[RCSCR]*
	AC17	149	?NC XQ	*nx.(nx + 3)*
	AC18	060	->1852	*[MP2_13]*
	AC19	04E	C=0 ALL	
	AC1A	35C	PT=12	*build "6" in C*
	AC1B	190	LD@PT- 6	
	AC1C	025	?NC XQ	*nx.(nx + 3) + 6*
	AC1D	060	->1809	*[AD1_10]*
	AC1E	0D1	?NC XQ	
	AC1F	064	->1934	*[RCSCR]*
	AC20	149	?NC XQ	*nx.[nx.(nx + 3) + 6]*
	AC21	060	->1852	*[MP2_13]*
	AC22	04E	C=0 ALL	
	AC23	35C	PT=12	*build "6" in C*
	AC24	190	LD@PT- 6	
	AC25	025	?NC XQ	*nx.[nx.(nx + 3) + 6] + 6*
	AC26	060	->1809	*[AD1_10]*
	AC27	0A9	?NC XQ	*nx*
	AC28	064	->192A	*[EXSCR]*
	AC29	035	?NC XQ	*exp(nx)*
	AC2A	068	->1A0D	*[EXP13]*
	AC2B	0D1	?NC XQ	*nx.[nx.(nx + 3) + 6] + 6*
	AC2C	064	->1934	*[RCSCR]*
	AC2D	24D	?NC XQ	*{C,M} / {A,B}*
	AC2E	060	->1893	*[X/Y13]*
	AC2F	089	?NC XQ	
	AC30	064	->1922	*[STSCR]*
	AC31	138	READ 4(L)	*n*
	AC32	10E	A=C ALL	
	AC33	135	?NC XQ	*n^2*
	AC34	060	->184D	*[MP2_10]*
	AC35	13D	?NC XQ	*n^4*
	AC36	060	->184F	*[MP1_10]*
	AC37	0D1	?NC XQ	
	AC38	064	->1934	*[RCSCR]*
	AC39	24D	?NC XQ	*{C,M} / {A,B}*
	AC3A	060	->1893	*[X/Y13]*
	AC3B	0B0	C=N ALL	*previous result*
	AC3C	025	?NC XQ	*updated result*
	AC3D	060	->1809	*[AD1_10]*
	AC3E	0F0	C<>N ALL	*new result to N, previous to C*
	AC3F	2BE	C=-C-1 MS	*sign change*
	AC40	025	?C XQ	*delta diffference*
	AC41	061	->1809	*[AD1_10]*
	AC42	10E	A=C ALL	*put in A for compares*
	AC43	351	?NC XQ	*compared to 1.E-9*
	AC44	128	->4AD4	*[TOLER4]*
	AC45	2FE	?C#0 MS	tolerance?
	AC46	20B	JNC -63d	*no, loop back*
	AC47	0B0	C=N ALL	*yes, we're done.*
	AC48	331	?NC GO	*Overflow, DropST, FillXL & Exit*
	AC49	002	->00CC	*[NFRX]*

Alternative version using FINTG:

```
*LBL "BLKBDY"
SF 00          ; default
SF 21          ; halting AVIEW
SIZE?          ; current size
20             ; required size
X>Y?           ; Enough?
PSIZE          ; no, adjust
CLA
ASTO 02
"J"            ; Joules
ASTO 01
"E"            ; Energy
RCL 16
XEQ 01         ; input units
STO 12
STO 15
RDN
STO 16
"M"            ; Meters
ASTO 01
"WVL"          ; wavelength
RCL 17
XEQ 01         ; input units
STO 10
STO 13
STO 14
X^2
X^2
ST* 12
RDN
STO 17
"M2"
ASTO 01
"AREA"
RCL 18
XEQ 01         ; input units
ST/ 12
ST/ 15
RDN
STO 18
"S"
ASTO 01
"TIME"
RCL 19
XEQ 01         ; input units
ST/ 12
ST/ 15
RDN
STO 19
<H>            ; Planck's const.
<c>            ; speed of light
*
<K>            ; Boltzmann's Const.
/
ST* 13
2897777 E-9
ST* 14
SG             ; Stefan-Boltzmann
ST* 15
*LBL 05
CLA
ASTO 02
"K"            ; unit string
ASTO 01        ;
2
STO 00         ; will use R03
"T"            ; magnitude
XROM "INPUT"
X^2
X^2
RCL 15
*
"Eb"
XEQ 03         ; output result
FC? 22
GTO 00
"LMAX="
RCL 14
RCL 03
/
ARCL X
>" "
ARCL 17
AVIEW
ADV
*LBL 00
7
STO 00         ; will store in R08
"L1"
SF 01
XROM "INPUT"
XROM "EbL"
XEQ 02         ; output EbL1
"L2"
SF 01
XROM "INPUT"
XROM "EbL"
```

```
XEQ 02          ; output Ebl2
CLD
CLST
RCL 08          ; λ1
RCL 09          ; λ2
"EbL"           ; integrand fnc.
FINTG           ; numerical integ
"Eb12"
SF 22
XEQ 03          ; output Eb12
GTO 05 ; new case
LBL "EbL"
X=0?
RTN
RCL 03          ; T
RCL 10          ; [λ] SI-factor
/
X<>Y            ; λ
EVL             ; SI conditions
RCL 12
*
RTN
*LBL 02         ; outputs EbL
SF 01
"EbL"
*LBL 03         ; outputs ALPHA
FC? 22
CF 01
FC? 22
RTN
>"="
ARCL X
AVIEW
CLA
ARCL 16
>"/"
ARCL 19
>"*"
ARCL 18
FC?C 01
GTO 00
>"*"
ARCL 17
*LBL 00
AVIEW
ADV
RTN
*LBL 01
E
ASTO T
"UNITS "
ARCL T
>"?"
AON
CF 23
PROMPT
AOFF
FC? 23
CLA
FC? 23
ARCL Y
ASTO Y
>"-"
ARCL 01
SF 25
SI>
CLA
ARCL T
FC?C 25
GTO 01
END.
```

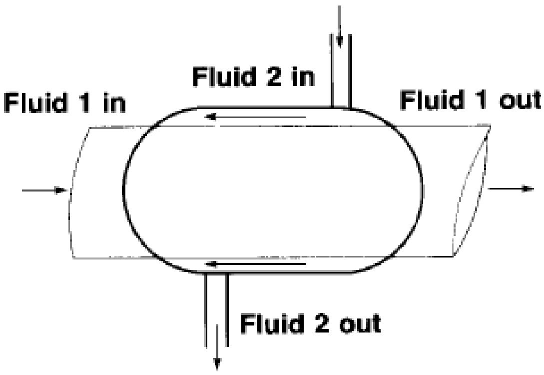
The code for the Planck's integral calculation is much simpler as we don't need to worry about any change of variable, nor to adjust the integration limits. Note also that there's no need to separate the definite integral into two improper ones, but the flipside is a longer execution time in the iterative process – and the accuracy depends on the decimal points setting (FIX).

Heat Exchangers revisited.

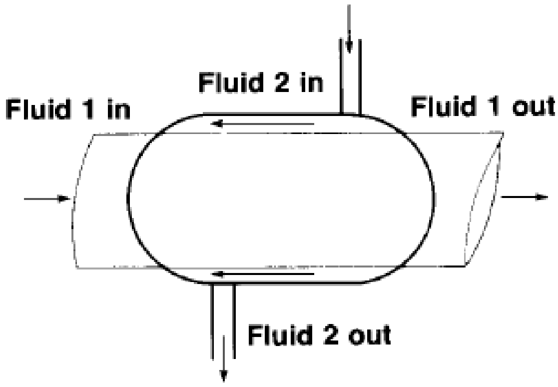
The HEATEX module includes an enhanced version of the Heat Exchangers program originally featured in the ETSII Collection – for counterflow and parallel flow designs. The enhancement basically consists of MCODE versions of the routines that calculate the unknown transferred heat and temperatures – either input or output depending on the known variables.

Base equation: **Q' = U. A. ΔTLM**

Where A is the total surface area for heat exchanger, U is the overall heat transfer coefficient, and ΔTLM is the Log mean temperature difference between the two fluids involved in the heat transfer



Counterflow



Parallel flow

It comes without saying that the mass flows and the specific heat coefficients will have a critical influence on the amount of exchanged heat. The specific design also makes the differences between types and models, but one general criteria concerns the cooling and heat fluid to have parallel flows or counterflow: the temperature difference is more abrupt in the counterflow case and that'd suggest higher efficiency –all other variables being the same.

The program uses a simplified approach for the number of variables, combining the products of mass flow and specific heat coefficients (m'.Cp) on one hand, and the total exchange area and model coefficient (A.U) on the other.

Transferred Heat equations

Let's denote by "i" the input and "o" the output of the heat exchanger. Let m1' and m2' be the mass flows for the *hot and cold fluids respectively,* and cp1 and Cp2 their specific heat coefficients. Making

$$k12 = \frac{m1'.\ cp1}{m2'.\ cp2}$$

Then the total transferred heat Qt is given by the expressions below:

$$Qt = m1' Cp1 \frac{T1i - T2i}{1+k12}\left(exp\left(-\frac{UA(1+k12)}{cp1.m1"}\right) - 1\right)$$ - Parallel flow

$$Qt = m1' Cp1 \frac{T1i - T2o}{1-k12}\left(exp\left(-\frac{UA(1-k12)}{cp1.m1"}\right) - 1\right)$$ - Counter flow

For an intermediate position placed at a distance x from the entry, let A(x) be the area exchange up until that distance, and U the global exchanger coefficient. Then thetemperaturesof each fluid T1(x) and T2(x) at a distance x from the input is as follows:

a. For parallel flow:

$$T1(x) = \left(\frac{1}{1+k12}\right).\left((T2i + k12.T1i) + (T1i - T2i)exp\left(-\frac{UA(x)(1+k12)}{cp1.m1"}\right)\right)$$

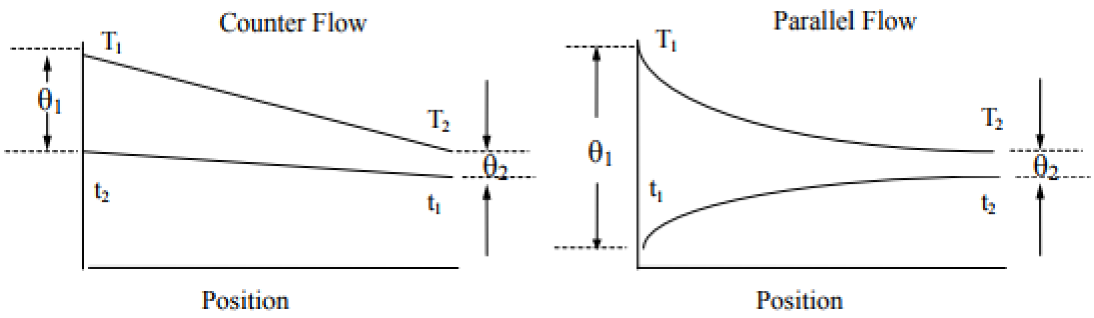
$$T2(x) = T2i - k12.(T1(x) - T1i)$$

b. For counter flow:

$$T1(x) = \left(\frac{1}{1-k12}\right).\left((T2o - k12.T1i) + (T1i - T2o)exp\left(-\frac{UA(x)(1-k12)}{cp1.m1"}\right)\right)$$

$$T2(x) = T2o + k12.(T1(x) - T2i)$$

Obviously, the temperatures at the output of the exchanger T1o and T2o can be calculated by the formulas above simply making A(X) = A, the total exchange area



Six new MCODE functions.

The new functions correspond to the temperatures and transferred heat for the cases shown below:

Function	Stage	Magnitude	Flow
T1C	Fluid-1 (hot)	Temperature	Counter
T2C	Fluid-2 (cold)	Temperature	Counter
QC	Combined	Heat	Counter
T1P	Fluid-1 (hot)	Temperature	Parallel
T2P	Fluid-2 (cold)	Temperature	Parallel
QP	Combined	Heat	Parallel

Using them removes the math from the main program, which turns into a driver for data entry and output results – also preparing the needed input variables to call the MCODE functions.

Example 1.

If a counterflow exchanger with an area of 1,000 ft^2 and an overall heat transfer coefficient of 27 BTU/hr.F.ft^2 is available, how close will the outlet temperature of the oil be to 110 F? What will the total heat transfer and outlet water temperature be?

Knowns:	T1i = 200F ;	T2o = 143 F
	m1'= 37,000 lbm/hr ;	m2' = 20,000 lbm/hr
	cp1 = 0,53 BTU/lbm.F;	cp2 =1 BTU/lbm.F

Solution:

we can approach his directly with the **QC**and **T1C** routines, using the following input parameters:

R00:	AU	= 27,000 BTU/hr.F	= 14,243.25600J/s..K
T:	m2'. Cp2	= 20,000 * 1	= 10,550.56000 J/s.K
Z:	m1'. Cp1	= 37,000 * 0.53	= 10,344.82408 J/s.K
Y:	T2o	= 143 F	= 334.816667K
X:	T1i	=200 F	= 366.483333 K

XEQ "QC"	**-1.518523706 E6** BTU/hr	= -445.035430kJ/s
X<>L, XEQ "T1C"	122.564 F	= 323.463 K

And executing the driver program HXIN:

XEQ "HXIN"	FLOW? C:P
C	M1'CP1=?
19610, R/S	M2'CP2=?
20000, R/S	T2-OUT=?
143, R/S	T1-IN=?
200, R/S	U*A(X)=?
27000, R/S	T1(X)=122.564 – outlet T1 = 323.463K
R/S	T2(X)=218.926 – outlet T2 = 376.998 K
R/S	Q=-1,518,523.706 = -445.035430 kJ/s

Example 2.

Calculate the output temperature of the oil and the total transferred heat in a parallel flow water-oil heat exchanger with AU= 115.8185 kcal/h.°C, when the mass flows are m'(water) = 5 kg/min and m'(oil) = 8 kg/min. if the inlet temperatures are Twater(I) = 20 °C and Toil(I) = 90 °C.Use the following for the specific heat capacities: Cp(water) = 1 kcal/kg.C; Cp(oil) = 0.9671 kcal/kg.C.

The results are:

Q(L)= 100 kcal/min
T1(O) =39,999996°C ;
T2(O) =77.0744 °C

Driver Program listing:

```
*LBL "HXIN"      ;Data Entry
?RTN             ; subroutine?
GTO 00           ; yes, skip data entry
"FLOW? CP"       ; Flow Type
PMTK             ; choose C/P
CF 00            ; flagsparallel
E
X=Y?
SF 00            ; flags counter
RCL 03           ; current value
"M1'CP1=?"       ; label 1
PROMPT
STO 03           ; cp1.m1'
RCL 04           ;current value
"M2'CP2=?"       ; label 2
PROMPT
STO 04           ; cp2.m2'
RCL 02           ; current
"T2-"            ; label
FC? 00           ; parallel?
"`IN"
FS? 00           ; counter?
>"OUT            ; finish string"
>"=?"
PROMPT
STO 02           T2i or T2o
RCL 01           ; current
"T1-IN=?"        ; label
PROMPT
STO 01           ; T1i
*LBL A           ; new UA
RCL 00           ; current
"U*A(X)=?"       ; label
PROMPT
STO 00           ; U/A
*LBL 00
RCL 04           ; cp2.m2'
RCL 03           ; cp1.m1'
RCL 02           ; T2i or T2o
RCL 01           ; T1i
FS? 00           ; counterflow?
GTO 02           ; yes, skip
T1P              ; parallel T1o
STO 05 ; store T1o result
X<> L            ; T1i
T2P              ; parallel T2o
STO 06           ; store T2o result
X<> L            ; T1i
QP               ; parallel flow heat
STO 07           ; store Q result
GTO 03           ; output results
*LBL 02
T1C              ; T1o
STO 05 ; store T1o result
X<> L            ; T1i
T2C              ; counter flow T2o
STO 06 ; store T2o result
X<> L            ; T1i
QC               ; counterflow heat
STO 07           ; store Q result
*LBL 03; output
X<> L            ; T1i
?RTN             ; subroutine?
RTN              ; yes, end here.
*LBL "HXOUT"; Output result
"T1(X)="         ; label
ARCL 05          ; value to ALPHA
AVIEW            ; show
"T2(X)="         ; label
ARCL 06          ; value to ALPHA
AVIEW ; show
"Q="             ; label
ARCL 07          ; value to ALPHA
AVIEW ; show
GTO A            ; new UA
END
```

Note that the input variables for the six MCODE functions are always expected as follows:

Register	T1C	T2C	QC	T1P	T2P	QP
R00	AU	AU	AU	AU	AU	AU
T	Cp2.m2'	Cp2.m2'	Cp2.m2'	Cp2.m2'	Cp2.m2'	Cp2.m2'
Z	Cp1.m1'	Cp1.m1'	Cp1.m1'	Cp1.m1'	Cp1.m1'	Cp1.m1'
Y	T2o	T2o	T2o	T2i	T2i	T2i
X	T1i	T1i	T1i	T1i	T1i	T1i

MCODE Listings for the utility functions.

Listed below are the MCODE utilities used by the driver program. There are six functions grouped by their functional category:

- Output Temperatures Hot fluid, parallel & counterflow
- Output Temperatures Cold Fluid, parallel & counterflow
- Transferred Heat, parallel and counterflow

CPU flag 8 is used to differentiate Parallel flows (F8 Clear) from Counter flows (F8 Set)

Header	A06C	090	"P"	
Header	A06D	032	"2"	
Header	A06E	014	"T"	T: m2'. Cp2
T2P	A06F	104	CLRF 8	Z: m1'. Cp1
	A070	02B	JNC +05	Y: T2e or T2s
Header	A071	083	"C"	X: T1e
Header	A072	032	"2"	
Header	A073	014	"T"	
T2C	A074	108	SETF 8	
BOTH	A075	179	?NC XQ	Checks 4 stack regs
	A076	100	->405E	[CHKST4] - sets DEC
	A077	375	PORT DEP:	perfrom the calculations
	A078	03C	XQ	for Temp
	A079	096	->A096	[T12PC]
	A07A	0F8	READ 3(X)	T1e
	A07B	2BE	C=-C-1 MS	sign change
	A07C	000	NOP	let carry settly
	A07D	025	?NC XQ	
	A07E	060	->1809	[AD1_10]
	A07F	0B0	C=N ALL	
	A080	13D	?NC XQ	
	A081	060	->184F	[MP1_10]
	A082	0B8	READ 2(Y)	T2e or T2s
	A083	025	?NC XQ	
	A084	060	->1809	[AD1_10]
	A085	07B	JNC +15d	[ADIOS]
Header	A086	090	"P"	
Header	A087	031	"1"	
Header	A088	014	"T"	T: m2'. Cp2
T1P	A089	104	CLRF 8	Z: m1'. Cp1
	A08A	02B	JNC +05	Y: T2e or T2s
Header	A08B	083	"C"	X: T1e
Header	A08C	031	"1"	
Header	A08D	014	"T"	
T1C	A08E	108	SETF 8	
BOTH	A08F	179	?NC XQ	Checks 4 stack regs
	A090	100	->405E	[CHKST4]
	A091	375	PORT DEP:	perfrom the calculations
	A092	03C	XQ	for Temp
	A093	096	->A096	[T12PC]
ADIOS	A094	331	?NC GO	Overflow, DropST, FillXL & Exit
	A095	002	->00CC	[NFRX]

All temperature calculations require subroutine [**T12PC**] below. Note that in turn it also calls the routine **[QPC**] listed in next page, shared by the transferred heat calculation main programs.

T12PC	A096	379	PORT DEP:	perfrom subroutine
	A097	03C	XQ	common to Temp & Heat
	A098	0E6	->A0E6	[QPC]
	A099	081	?NC XQ	pre-selects Chip0
	A09A	064	->1920	[STSCR*]
	A09B	0F8	READ 3(X)	T1e
	A09C	10E	A=C ALL	
	A09D	0B8	READ 2(Y)	T2e or T2s
	A09E	2BE	C=-C-1 MS	sign change
	A09F	000	NOP	let carry settly
	A0A0	01D	?NC XQ	
	A0A1	060	->1807	[AD2_10]
	A0A2	0D1	?NC XQ	
	A0A3	064	->1934	[RCSCR]
	A0A4	149	?NC XQ	
	A0A5	060	->1852	[MP2_13]
	A0A6	089	?NC XQ	
	A0A7	064	->1922	[STSCR]
	A0A8	0B0	C=N ALL	k12 = m1'. Cp1 / m2'. Cp2
	A0A9	10E	A=C ALL	
	A0AA	0F8	READ 3(X)	T1e
	A0AB	135	?NC XQ	T1e . K12
	A0AC	060	->184D	[MP2_10]
	A0AD	0B8	READ 2(Y)	T2e
	A0AE	025	?NC XQ	
	A0AF	060	->1809	[AD1_10]
	A0B0	0D1	?NC XQ	
	A0B1	064	->1934	[RCSCR]
	A0B2	031	?NC XQ	
	A0B3	060	->180C	[AD2_13]
EXIT	A0B4	089	?NC XQ	
	A0B5	064	->1922	[STSCR]
	A0B6	0B0	C=N ALL	
	A0B7	1E1	?NC XQ	Increment C
	A0B8	100	->4078	[INCC10]
	A0B9	0D1	?NC XQ	
	A0BA	064	->1934	[RCSCR]
	A0BB	24D	**?NC GO**	{C,M} / {A,B}
	A0BC	062	->1893	[X/Y13]

Finally, see in next page the Heat Transfer routines, also using the 13-digit math subroutines from the OS and a couple of calls to the Library#4 (in dark-blue background).

Header	A0BD	090	"P"	
Header	A0BE	011	"Q"	
QP	A0BF	104	CLRF 8	
	A0C0	023	JNC +04	[BOTH]
Header	A0C1	083	"C"	
Header	A0C2	011	"Q"	
QC	A0C3	108	SETF 8	
BOTH	A0C4	179	?NC XQ	Checks 4 stack regs
	A0C5	100	->405E	[CHKST4]
	A0C6	375	PORT DEP:	perfrom the calculations
	A0C7	03C	XQ	for Heat
	A0C8	0E6	->A0E6	[QPC]
	A0C9	009	?NC XQ	
	A0CA	060	->1802	[SUBONE]
	A0CB	081	?NC XQ	pre-selects Chip0
	A0CC	064	->1920	[STSCR*]
	A0CD	0F8	READ 3(X)	T1e
	A0CE	10E	A=C ALL	
	A0CF	0B8	READ 2(Y)	T2e or T2s
	A0D0	2BE	C=-C-1 MS	sign change
	A0D1	000	NOP	let carry settly
	A0D2	01D	?NC XQ	
	A0D3	060	->1807	[AD2_10]
	A0D4	0D1	?NC XQ	
	A0D5	064	->1934	[RCSCR]
	A0D6	149	?NC XQ	
	A0D7	060	->1852	[MP2_13]
	A0D8	078	READ 1(Z)	m1'. Cp1
	A0D9	13D	?NC XQ	
	A0DA	060	->184F	[MP1_10]
	A0DB	089	?NC XQ	
	A0DC	064	->1922	[STSCR]
	A0DD	0B0	C=N ALL	
	A0DE	1E1	?NC XQ	Increment C
	A0DF	100	->4078	[INCC10]
	A0E0	0D1	?NC XQ	
	A0E1	064	->1934	[RCSCR]
	A0E2	24D	?NC XQ	{C,M} / {A,B}
	A0E3	060	->1893	[X/Y13]
	A0E4	331	?NC GO	Overflow, DropST, FillXL & Exit
	A0E5	002	->00CC	[NFRX]
QPC	A0E6	2A0	SETDEC	
	A0E7	078	READ 1(Z)	m1'. Cp1
	A0E8	10E	A=C ALL	
	A0E9	046	C=0 S&X	
	A0EA	270	RAMSLCT	
	A0EB	038	READATA	m2'. Cp2
	A0EC	261	?NC XQ	k12 = m1'. Cp1 / m2'. Cp2
	A0ED	060	->1898	[DV2_10]
	A0EE	10C	?FSET 8	
	A0EF	01B	JNC +03	
	A0F0	2BE	C=-C-1 MS	sign change
	A0F1	11E	A=C MS	ditto in 13-digit form
	A0F2	070	N=C ALL	save k12 for later use
	A0F3	001	?NC XQ	
	A0F4	060	->1800	[ADDONE]
	A0F5	078	READ 1(Z)	m1'. Cp1
	A0F6	269	?NC XQ	
	A0F7	060	->189A	[DV1-10]
	A0F8	271	?NC XQ	Fetch R00 value
	A0F9	130	->4C9C	[READ00]
	A0FA	13D	?NC XQ	
	A0FB	060	->184F	[MP1_10]
	A0FC	2BE	C=-C-1 MS	sign change
	A0FD	11E	A=C MS	ditto in 13-digit form
	A0FE	035	?NC GO	
	A0FF	06A	->1A0D	[EXP13]

Transients in long cylinder with step temperature change

Extension of the TXT program from the ETSII Collection

This program calculates the temperature T(r) in points r of an infinite cylinder of radius R, after experiencing a thermal shock – or sudden change of ambient temperature, from To initial to Tf final.

Similar to the flat plate case, the Biot number is calculated from its constituent factors. The same data entry process is used like in the infinite plate, only now it is cylindrical symmetry instead.

The resulting temperature is expressed as an infinite sum as follows:

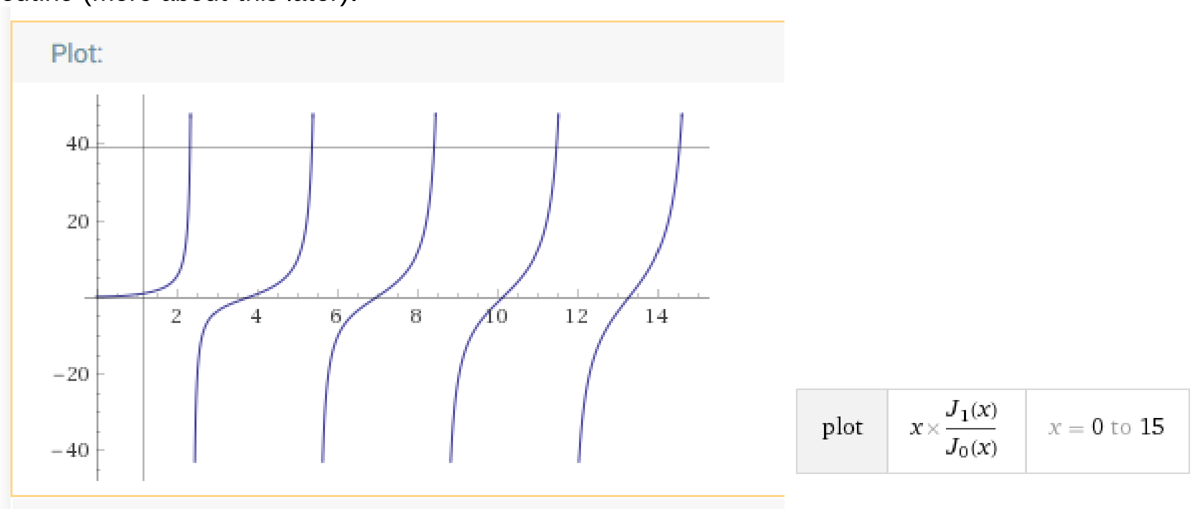
$$T(x,t) = Tf + (T0-Tf) \Sigma \{ (2/\lambda n). f(n, r). \exp[-\alpha.t.(\lambda n/R)^2]\} ; n = 1,2,...$$

$$\text{With: } f(n,r) = J1(\lambda n). J0(\lambda n.r/R) / [J1^2(\lambda n) + J0^2(\lambda n)]$$

Andwhereλnare the n roots of the equation defined by:

$$(\lambda n) J1(\lambda n) = Bi\ J0(\lambda n)$$

Which, leaving the Biot number alone in the second term, can be expressed as the intersection of the Biot number with the function x.J1(x)/J0(x), shown in the graphic below - where the asymptotic boundaries will provide a reasonable criteria for the estimations needed bythe root-finding routine (more about this later).



Example.

A very long metal rod of radius R=0.14 m has a uniform temperature of 1,000 deg C. It is suddenly immersed into a cooling fluid stream at 50 deg C. Calculate the temperature in its center and outer boundary 15, 30 and 60 minutes after the sudden step temperature change. The physicalproperties of the material are given below:

α = 1.66 E-6 m^2/s
h = 20,000 kcal/H.m^2.C = 23,260.0 W/m^2 K
K = 100 kcal/h.m.C = 116.30 W/m.K

The result temperatures are shown in the table below: (warning: very slow convergence)

Point	t = 15 min	t = 30 min	t = 1 hour
center (x=0 cm)	945.7185485	704.2922460	343.4690201
Outer edge (x=0.14 m)	102.5288706	80.51769740	63.05841690

A few remarks regarding the implementation.

By direct inspection of the plot in previous page, it's clear that this case is much more demanding on the root-finder algorithm than the previous one. As the Biot number value increases, the intersection with the graphic will occur in zones with a very steep slope, making the identification of the root very tricky – so much so that the FOCAL routine "**SLV**" is not adequate and misses the roots, even if very fine-tuned search intervals are provided – which is also a difficult affair.

To search for each of theλnroots, the program uses symmetric intervals centered at the initial value, and distance "one", i.e:

[n*(λn)init - 0.5 ; n*(λn)init + 0.5]

With: (λn)init = $\frac{3}{2}.\left(sqrt\left(1+\frac{4}{3}.Bi\right)-1\right)$

In this version we've used **FROOT** instead, also included in the SandMath - which was already required for the Bessel functions, so no more dependencies are added. The estimation for the initial guesses become very important for the successful root identification, and the execution times – which are going to be very long regardless; better crank up your turbo emulator for this one!

Another important remark is that repeating the calculations for different values of (t, r) (analysis time and distance to the cylinder axis) *has been expedited dramatically for subsequent runs with longer times than previous executions*). *In that case there's no need to re-calculate or find additional Mn roots beyond those already identified*, as the contribution of the terms to the infinite sum will be smaller due to the larger argument in the inverse exponential function:

f(n, r) . *exp*[- α.t.(λn/R)^2]}

This of course is not so straight-forward as one may think, because the series is alternating the sign of its terms, so the contributions are not always in the same direction.

The program stores the successive roots found in an X-memory file, to be reused when the analysis is repeated with longer values of cooling time. In this way, if the X-mem value is zero then the corresponding root needs to be sought for.

The program listing is given below. Note that the ALPHA registers are used by the infinite sum routine to calculate the partials and to store the current term. Because the MCODE function JBS also uses the ALPHA registers for scratch, we'll use the function A<>RG to preserve ALPHA in {R17-R20} while the general term is being calculated.

XROM "?" is a simple data-entry utility functions to save bytes. Be careful if you use arithmetic functions with the value in X – that would alter the expected stack configuration and may be disruptive to the program.

Program listing:

```
01  *LBL "TRT"
02  SIZE?
03  21
04  X>Y?
05  PSIZE
06  E
07  CF 04
08  "SAVE-RT? YN"
09  PMTK         ; in OS/X Module
10  X#Y?         ; don't save?
11  GTO 04       ; skip file stuff
12  SF 04        ; flag case
13  "λN"         ; file Name
14  SF 25
15  PURFL        ; purge if exists
16  9            ; up to nine roots
17  CF 25
18  CRFLD        ; create new file
19  *LBL 04      ; data input
20  "TINI"
21  11
22  XROM "?"
23  "TEND"       ; var name
24  E1
25  XROM "?"     ; prompt & save
26  ST- 11
27  "H"          ; var name
28  E
29  XROM "?"     ; prompt & save
30  STO 14
31  "K"          ; var name
32  2
33  XROM "?"     ; prompt & save
34  ST/ 14
35  "RO"         ; var name
36  3
37  XROM "?"
38  ST* 14       Bi
39  "ALPHA"      ; var name
40  0
41  XROM "?"     ; prompt & save
42  *LBL C       new case
43  0
44  FS? 04
45  SEEKPT
46  "R"          ; variable name
47  4
48  XROM "?"     ; prompt & save
49  "TIME"
50  12
51  XROM "?"     ; prompt & save
52  RCL 14
53  0,75         ; 3/4
54  /
55  E
56  +
57  SQRT
58  E
59  -
60  1.5          3/2
61  *
62  SQRT
63  STO 13
64  "aJ"
65  ASTO 06
66  XROM "ΣI"    infinite SUM
67  ST+ X
68  RCL 11
69  *
70  RCL 10
71  +
72  "T(X,T)="
73  ARCL X
74  PROMPT
75  GTO C
76  *LBL "aJ"    ; function to sum...
77  A<>RG        ; preserve alpha
78  17           ; in {R17 - R20}
79  FC? 04       ; roots in X-Mem?
80  GTO 04       ; no, need search
81  GETX         ; yes, get current
82  X#0?         ; valid root?
83  GTO 05       ; yes!
84  RCLPT        ; no, backtrack ptr.
85  E
86  -
87  SEEKPT
88  *LBL 04      ; need to search
89  RCL 18       ; n
90  RCL 13       ; delta
91  *
92  RCL X
93  ,5
94  ST- Z
95  +
96  "JN"         ; function to solve
97  FROOT        ; root-finding
98  FS? 04       ; save option?
99  SAVEX        ; yes, oblige
100 *LBL 05
101 STO 07       ; λn
102 VIEW X
103 E
104 X<>Y
105 JBS          ; J1
106 STO 08
107 0
108 RCL 07       ; λn
109 JBS          ; Jo
110 X^2          ;Jo^2
```

```
111  RCL 08      ; J1
112  X^2         ; J1^2
113  +           ; Jo^2+J1^2
114  STO 09
115  0
116  RCL 07      ; λn
117  RCL 03      ; R0
118  /           ; λn/R0
119  RCL 04      ; r
120  *
121  JBS         ; Jo(λn.r.R0)
122  RCL 08      ; J1
123  *
124  RCL 09      ; Jo^2+J1^2
125  /
126  RCL 07      ; λn
127  /
128  LASTX       ; λn
129  RCL 03      ; R0
130  /           λn /R0
131  X^2         ; (λn/Ro)^2
132  RCL 12      ; t
133  *
134  RCL 00      ; α
135  *           α.t(λn/Ro)^2
136  CHS
137  E^X
138  *
139  A<>RG       restore ALPHA
140  17          from {R17-R10}
141  RTN
142  *LBL "JN"   function to solve...
143  E
144  X<>Y
145  JBS
146  X<>Y
147  ST+ X
148  ,
149  X<>Y
150  JBS
151  ST/ Z
152  RDN
153  ST+X
154  *
155  RCL 14      Bi
156  -
157  END
```

and:

```
01*LBL "ΣI"        ; aN name in R06
02 CLA             ; reset all
03 STO O           ; argument
04*LBL 00          ; current term
05 E               ; starts at n=1
06 ST+ N           ; increase counter
07 XEQ IND 06      ; calls aN
08 X#0?            ; non-zero term?
09 GTO 01          ; yes, skip over
10 RCL O           ; argument
11 X=0?            ; zero?
12 RTN             ; yes, end.
13 GTO 00          ; re-do
14*LBL 01
15 RCL M           ; partial sum
16 RCL M
17 X<> Z           ; n-th. term
18 +               ; new partial sum
19 STO M           ; store
20 X=YR?           ; same as previous?
21 RTN             ; yes, done!
22 GTO 00          ; no, next term
23*LBL "?"         ; prompt & save
24 RCL IND X
25 >"="
26 ARCL X
27 >"?"
28 CF 22
29 PROMPT
30 FS?C 22
31 STO IND Z
32 END
```

Appendix.- Roots of Bessel Equation x.J1(x) - Bi,J0(x) = 0

The program below was contributed by Jean-Marc Baillard. It calculates the roots of the Bessel equation from the previous section, ie. x.J1(x) - Bi,J0(x) = 0, and it does so using a dedicated approach – not relying on general-purpose root finders.

To use it, enter the Biot number in Y and the number of roots desired in X. The different roots will be obtained sequentially, press R/S to continue with the next one.

Program listing:

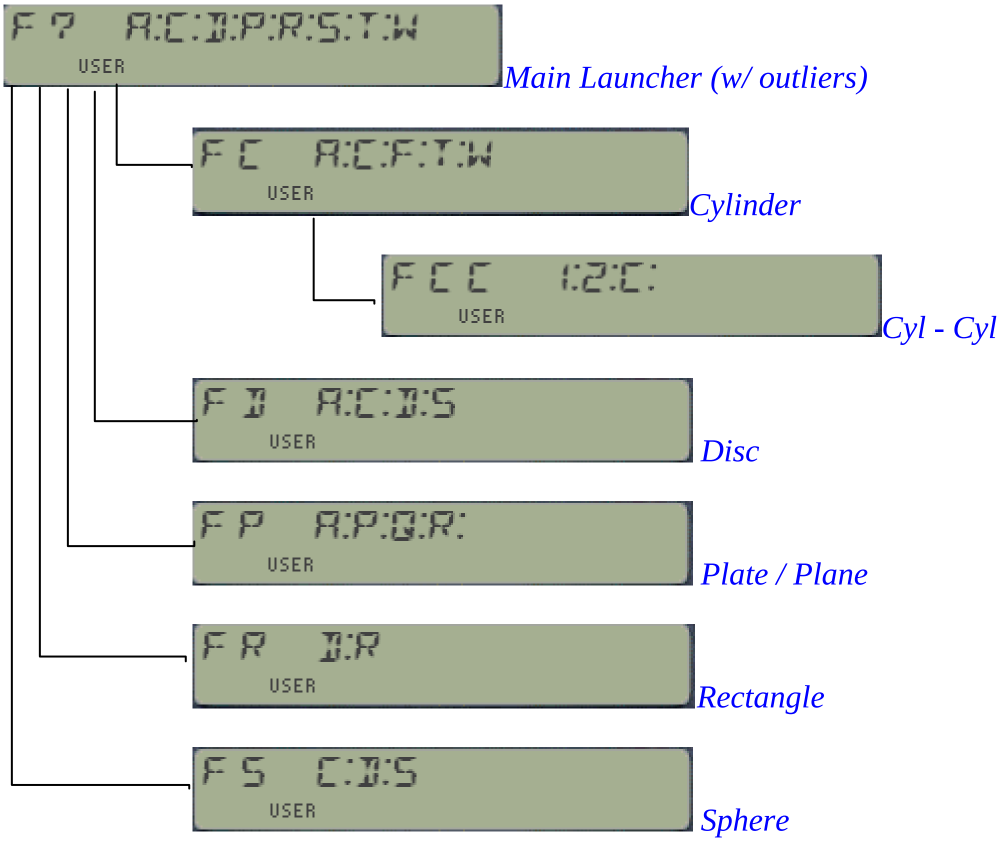
```
1    *LBL "BIOT"
2    STO 08      ; #
3    STO 12
4    X<>Y ; Bi
5    ENTER^
6    STO 09
7    E
8    +
9    /
10   2.4
11   *
12   PI
13   -
14   STO 11
15   13
16   STO 10
17   GTO 14
18   *LBL 10
19   VIEW 11 iteration
20   CLST
21   RCL 11
22   XEQ 12
23   STO 07
24   CLX
25   SIGN
26   RCL 11
27   XEQ 12
28   STO 06
29   RCL 11
30   *
31   RCL 09
32   RCL 07
33   *
34   -
35   RCL 11
36   RCL 07
37   *
38   RCL 06
39   RCL 09
40   *
41   +
42   /
43   ST- 11
44   RCL 11
45   X#0?
46   /
47   ABS
48   E-8
49   X<Y?
50   GTO 10
51   RCL 11
52   RTN
53   *LBL 12
54   X#0?
55   GTO 00
56   X#Y?
57   RTN
58   SIGN
59   RTN
60   *LBL 00
61   STO 01
62   ABS
63   5
64   +
65   X<>Y
66   STO 00
67   X<Y?
68   X<>Y
69   INT
70   ST+ X
71   STO 03
72   ST- 00
73   CLST
74   STO 04
75   STO 05
76   SIGN
77   *LBL 01
78   STO Z
79   RCL 03
80   ST+ X
81   *
82   RCL 01
83   /
84   X<>Y
85   -
86   ISG 00
87   STO 02
88   ST+ 04
89   RCL 05
90   X<> 04
91   STO 05
92   RDN
93   DSE 03
94   GTO 01
95   RCL 05
96   ST+ X
97   X<>Y
98   -
99   RCL 02
100  X<>Y
101  /
102  RTN
103  *LBL 14
104  PI
105  ST+ 11
106  XEQ 10
107  STO IND 10
108  ISG 10
109  CLX
110  DSE 12
111  GTO 14
112  RCL 10
113  E
114  -
115  E3
116  /
117  13
118  +
119  CLD
120  END
```

Appendix. ALPHA <> Stack Exchange Utilities

Header	AD99	087	"G"	
Header	AD9A	012	"R"	**ALPHA <> REG**
Header	AD9B	03E	">"	Prompting
Header	AD9C	23C	"<"	
Header	AD9D	201	"A"	Ken Emery
A<>RG _ _	**AD9E**	**088**	**SETF 5**	
	AD9F	043	JNC +08	
Header	ADA0	087	"G"	
Header	ADA1	012	"R"	
Header	ADA2	03E	">"	**Stack <> REG**
Header	ADA3	03C	"<"	Prompting
Header	ADA4	214	"T"	
Header	ADA5	213	"S"	Ángel Martin
ST<>RG _ _	**ADA6**	**084**	**CLRF 5**	
	ADA7	04C	?FSET 4	
	ADA8	01F	JC +03	SST'ing a program
	ADA9	2CC	?FSET 13	
	ADAA	01B	JNC +03	RUN'ing a program
	ADAB	179	?NC XQ	Get Parameter from NextLine
	ADAC	10C	->435E	[GETRG#]
	ADAD	130	LDI S&X	Returns with it in A[S&X]
	ADAE	004	CON: 4	
	ADAF	0E6	C<>B S&X	add adjust factor to RG#
	ADB0	219	?NC XQ	Check Reg Existence
	ADB1	10C	->4396	[EXISTS]
	ADB2	186	A=A-B S&X	back out the adjustment
	ADB3	39C	PT= 0	To use as a counter
	ADB4	046	C=0 S&X	Starts in register 0(T)
	ADB5	08C	?FSET 5	
	ADB6	01B	JNC +03	
	ADB7	130	LDI S&X	
	ADB8	005	CON: 5	Starts in register 5(M)
	ADB9	0A6	A<>C S&X	Source block address in A
A<>RG	ADBA	270	RAMSLCT	Select register Block
	ADBB	0E6	C<>B S&X	Save rg. Block addr in B
	ADBC	038	READATA	Read Source Register
	ADBD	070	N=C ALL	store register content in N
	ADBE	0AE	A<>C ALL	recall stack rg addr
	ADBF	270	RAMSLCT	Select stack block
	ADC0	0AE	A<>C ALL	save stack rgi n A
	ADC1	038	READATA	read stack rg content
	ADC2	0F0	C<>N ALL	exchange stack content w/ RG co
	ADC3	2F0	WRTDATA	Write RG content in stack rg
	ADC4	166	A=A+1 S&X	decrease stack rg. Pointer
	ADC5	0E6	C<>B S&X	recall RG block add to C
	ADC6	270	RAMSLCT	select RG block again
	ADC7	226	C=C+1 S&X	increase RG block addr
	ADC8	0E6	C<>B S&X	save RG block addr in B
	ADC9	0F0	C<>N ALL	
	ADCA	2F0	WRTDATA	write stack content to RG block
	ADCB	0E6	C<>B S&X	
	ADCC	3DC	PT=PT+1	increase counter
	ADCD	054	?PT= 4	Carry if finished
	ADCE	360	?C RTN	
	ADCF	35B	JNC -21d	

View Factors Library

This module contains a small library with the most frequently used View Factors. They're grouped by types of geometry, and can all be accessed from the main launcher SVF:



or arranged in tabulated view:

ΣVF	FC		FD	FP	FR	FS
AAF						
FC	CAF					
FD	**FCC**	CC11F	DAF			
FP	CCF	CC21F	DCSF	PAF		
FR	CSTF	CCCF	DDF	PPSF	RDF	
FS	CWCF		DSF	PQSF	RR90F	SSCF
TPSF				PPRF		SDF
WCF						SSF

A total of 23 functions grouped logically by the geometries involved. Let's describe them in detail, including the input parameters required and the stack arrangement.

Individual View Factors

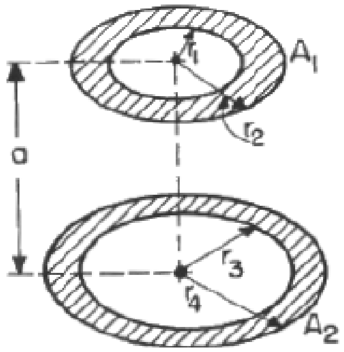
AAF : Annular Ring to Annular Ring,

T:	R1	Inner radius ring #1
Z:	R2	Outer radius, ring #1
Y:	R3	Inner radius ring #2
X:	R4	Outerradius, ring#2
L:	H	Distance between centers

21 Ring to parallel coaxial ring

$H = a/r_1$; $R_2 = r_2/r_1$ Applicable to both the cases.

$R_3 = r_3/r_1$; $R_4 = r_4/r_1$



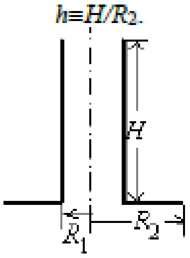
$$F_{1-2} = \frac{1}{2\left(R_1^2 - 1\right)}\left\{\left[\left(R_2^2 + R_3^2 + H^2\right)^2 - \left(2R_3R_2\right)^2\right]^{1/2} - \left[\left(R_2^2 + R_4^2 + H^2\right)^2 - \left(2R_2R_4\right)^2\right]^{1/2} + \left[\left(1 + R_4^2 + H^2\right)^2 - \left(2R_4\right)^2\right]^{1/2} - \left[\left(1 + R_3^2 + H^2\right)^2 - \left(2R_3\right)^2\right]^{1/2}\right\}$$

$F_{1-2} = \frac{1}{2}\left\{R\right.$

CAF : Cylinder to Annular Ring

Z:	H	Cylinder Height
Y:	R1	Inner radius ring
X:	R2	Outer radius, ring

From external lateral surface (1) of a cylinder of radius R_1 and height H, to annular disc of radius R_2, with $r \equiv R_1/R_2$, $h \equiv H/R_2$.

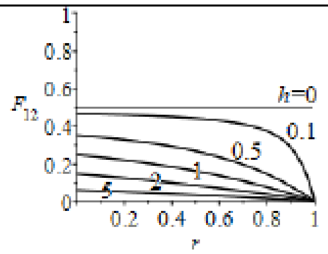


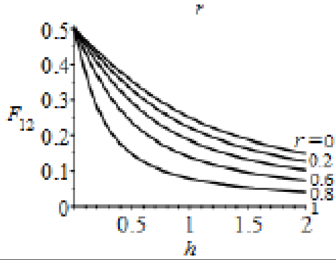
$$F_{12} = \frac{y}{8rh} - \frac{\left(z + x\arcsin(r)\right)}{4\pi rh} + \frac{1}{2\pi}\arccos\left(\frac{x}{y}\right),$$

with $x = h^2 + r^2 - 1$, $y = h^2 - r^2 + 1$,

$$z = \sqrt{(x+2)^2 - 4r^2}\arccos\left(\frac{xr}{y}\right)$$

(e.g. for $R_2=2R_1=2H$, i.e. $r=h=1/2$, $F_{12}=0.268$, $F_{21}=0.178$)

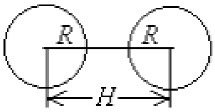




CCF : Cylinder to ParallelCylinder

Z:	H	Cylinder Height
Y:	R1	Inner radius ring
X:	R2	Outer radius, ring

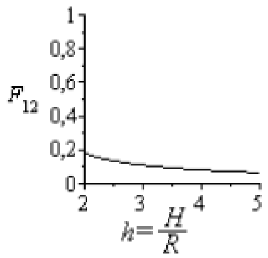
From a cylinder of radius R to an equal cylinder at a distance H between centres (it must be $H>2R$), with $h \equiv H/R$.



Note. See the crossing-string method, above.

$$F_{12} = \frac{\sqrt{h^2-4} - h + 2\arcsin\frac{2}{h}}{2\pi}$$

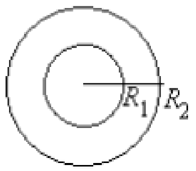
(e.g. for $H=2R$, $F_{12}=1/2-1/\pi=0.18$)



CCCF: Concentric Cylinder to Cylinder

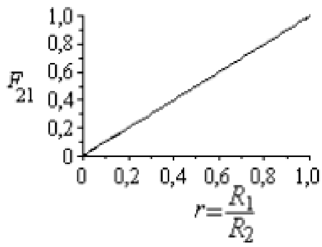
Y:	R1	Inner Cylinder radius
X:	R2	Outer Cylinder radius

Between concentric infinite cylinders of radii R_1 and $R_2>R_1$, with $r \equiv R_1/R_2<1$.



$F_{12}=1$
$F_{21}=r$
$F_{22}=1-r$

(e.g. for $r=1/2$, $F_{12}=1$, $F_{21}=1/2$, $F_{22}=1/4$)

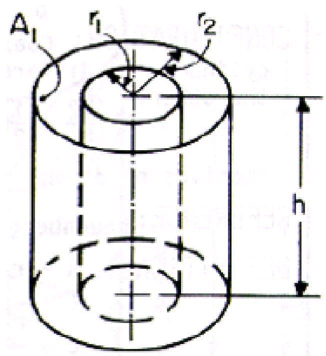


CC11F :Coaxial Cylinders – Itself

Z:	H	Cylinder Height
Y:	R1	Inner Cylinder radius
X:	R2	Outer Cylinder radius

25 Interior of finite length right circular coaxial cylinder to itself $R_1 = r_1/h$; $R_2 = r_2/h$

Interior of Outer Cylinder

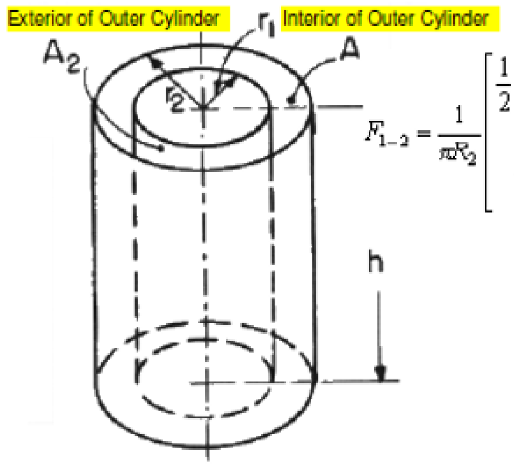


$$F_{1-1} = \frac{1}{\pi R_2}\left\{ \begin{array}{l} \pi(R_2 - R_1) + \cos^{-1}\left(\frac{R_1}{R_2}\right) \\ -\left(1+4R_2^2\right)^{1/2}\tan^{-1}\frac{\left[\left(1+4R_2^2\right)\times\left(R_2^2-R_1^2\right)\right]^{1/2}}{R_1} \\ +2R_1\tan^{-1}\left[2\left(R_2^2-R_1^2\right)^{1/2}\right] \end{array} \right\}$$

(use principal values in evaluating all inverse trig functions)

CC12F :Coaxial Cylinders - Itself

26 Interior of outer right circular cylinder of finite length to exterior of coaxial inner right circular cylinder



$R_1=r_1/h$; $R_2=r_2/h$; $A=R_2+R_1$; $B=R_2-R_1$

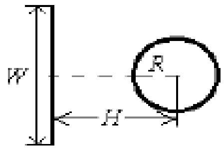
$$F_{1-2} = \frac{1}{\pi R_2}\left[\frac{1}{2}\left(R_2^2 - R_1^2 - 1\right)\cos^{-1}\frac{R_1}{R_2} + \pi R_1 - \frac{\pi}{2}AB - 2R_1\tan^{-1}\left(R_2^2 - R_1^2\right)^{1/2} + \left\{\left(1+A^2\right)\left(1+B^2\right)\right\}^{1/2}\tan^{-1}\left\{\frac{\left(1+A^2\right)B}{\left(1+B^2\right)A}\right\}^{1/2}\right]$$

Z:	H	Cylinder Height
Y:	R1	Inner Cylinder radius
X:	R2	Outer Cylinder radius

CSTF : Cylinder to Frontal Strip

Z:	W	Strip width
Y:	H	Distance
X:	R	Cylinder radius

From frontal surface of strip (1) of width W to a cylinder (2) of radius R at a distance H between centres, with $v=W/(2R)$ and $h=H/R>1$.

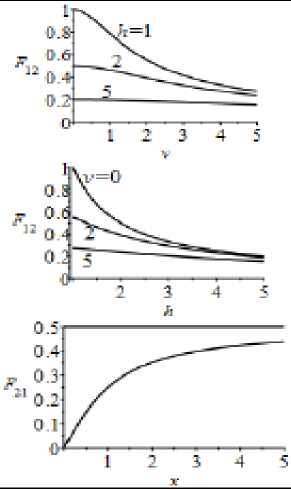


$$F_{12} = \frac{\arctan\left(\frac{v}{h}\right)}{v}$$

$$F_{21} = \frac{\arctan\left(\frac{v}{h}\right)}{\pi} = \frac{\arctan(x)}{\pi} \text{ with}$$

$x=v/h=(W/2)/H$

(e.g. for $W=R=H$, i.e. $v=1/2$, $h=1$, $x=1/2$, $F_{12}=0.927$, $F_{21}=0.148$)



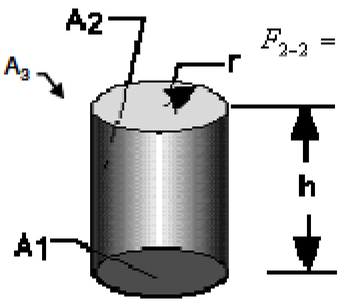
CWCF : Cylinder Wall to Cap

17 View Factor of Cylinder Wall w.r.t. Individual Cap

H = h/2r

$$F_{1-2} = 2H\left[\left(1+H^2\right)^{1/2} - H\right]$$

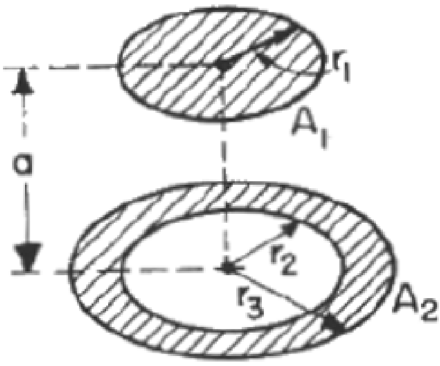
$$F_{2-2} = (1+H) - \left(1+H^2\right)^{1/2}$$



Y:	H	Cylinder height
X:	R	Cylinder radius

DAF : Disc to Annular Ring

T:	H	Distance between centers
Z:	R1	Disc radius
Y:	R2	Inner radius, ring
X:	R3	Outer radius, ring
L:	-	n/a

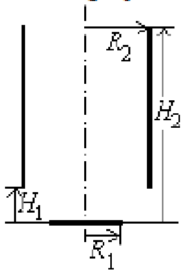


$$F_{1\text{-}2} = \frac{1}{2}\left\{R_3^2 - R_2^2 - \left[\left(1 + R_3^2 + H^2\right)^2 - 4R_3^2\right]^{1/2} + \left[\left(1 + R_2^2 + H^2\right)^2 - 4R_2^2\right]^{1/2}\right\}$$

DCSF :Disc to Cylinder Surface

T:	H1	Distance 1	**Y:**	R1	Disc 1 radius
Z:	H2	Distance 2	X:	R2	Disc 2 radius

From disc (1) of radius R_1 at a distance H_1, to internal lateral surface (2) of a coaxial cylinder of radius $R_2>R_1$ and height H_2-H_1, with $h_1\equiv H_1/R_1$, $h_2\equiv H_2/R_1$, $r\equiv R_2/R_1$.



$$F_{12} = \frac{1}{2}\left[\left(x_2 - x_1\right) - \left(y_2 - y_1\right)\right]$$

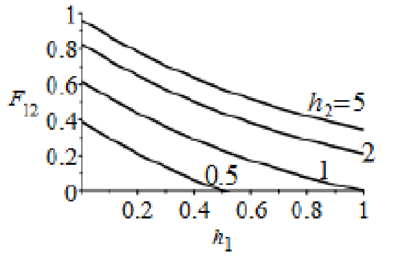
with

$$x_1 = 1 + r^2 + h_1^2, \quad y_1 = \sqrt{x_1^2 - 4r^2}$$

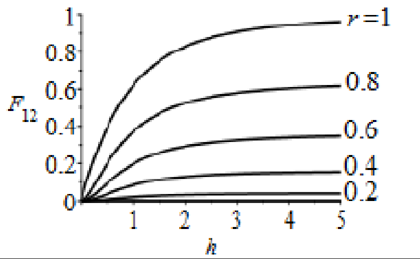
$$x_2 = 1 + r^2 + h_2^2, \quad y_2 = \sqrt{x_2^2 - 4r^2}$$

(e.g. for $H_1=0$ & $H_2=R_2=R_1$, i.e. for $r=1$, $h_1=0$ & $h_2=1$, $F_{12}=0.62$)

Disc of same radius (i.e. $R_2=R_1$ and $r=1$)



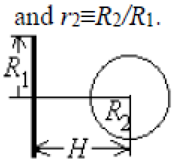
Disc at the base (i.e. $H_1=0$, with $h=h_2$)



Disk to Sphere

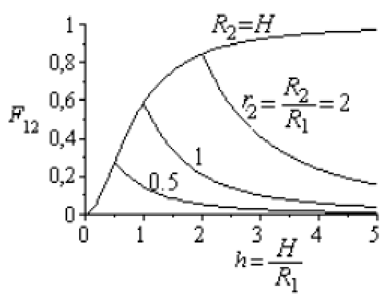
Z:	H	Distance to center
Y:	R1	Disc Radius
X:	R2	Cylinder radius

From a disc of radius R_1 to a frontal sphere of radius R_2 at a distance H between centres (it must be $H>R_1$), with $h\equiv H/R_1$ and $r_2\equiv R_2/R_1$.



$$F_{12} = 2r_2^2\left(1 - \frac{1}{\sqrt{1 + \frac{1}{h^2}}}\right)$$

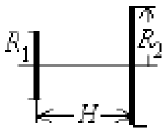
(e.g. for $h=r_2=1$, $F_{12}=0.586$)



DDF : Disk to Disk

Z:	H	Distance to center
Y:	R1	Disc 1 Radius
X:	R2	Disc 2 radius

From a disc of radius R_1 to a coaxial parallel disc of radius R_2 at separation H, with $r_1=R_1/H$ and $r_2=R_2/H$.

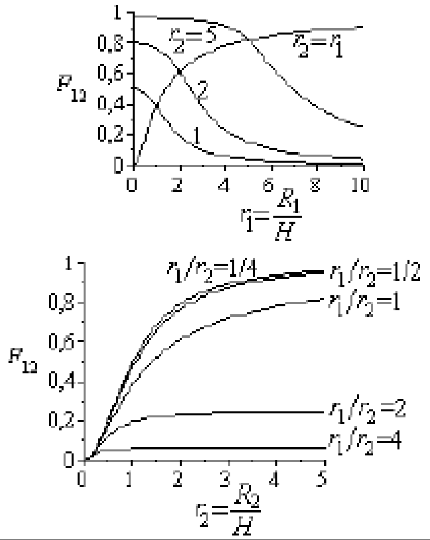


$$F_{12}=\frac{x-y}{2}$$

with $x=1+1/r_1^2+r_2^2/r_1^2$ and

$$y=\sqrt{x^2-4r_2^2/r_1^2}$$

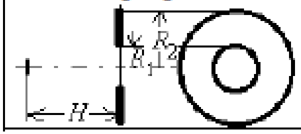
(e.g. for $r_1=r_2=1$, $F_{12}=0.382$)



PAF : Patch to annular Ring

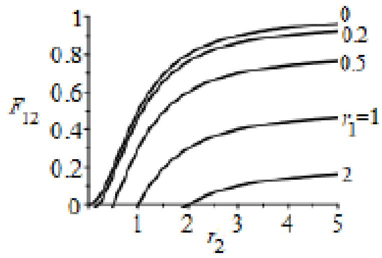
Z:	H	Distance to center
Y:	R1	Inner Radius
X:	R2	Outer radius

From a patch to a parallel and concentric annulus of inner radius R_1 and outer radius R_2, at distance H, with $r_1=R_1/H$ and $r_2=R_2/H$.



$$F_{12}=\frac{r_2^2}{1+r_2^2}-\frac{r_1^2}{1+r_1^2}$$

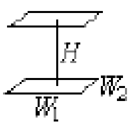
(e.g. for $r_1=1$ and $r_2=2$, $F_{12}=0.30$)



PPRF : Plate to Parallel Plate

Z:	H	Distance to center
Y:	W1	Side 1 lengths
X:	W2	Side 2 length

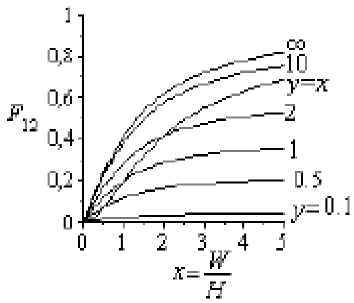
Between parallel equal rectangular plates of size $W_1 \cdot W_2$ separated a distance H, with $x=W_1/H$ and $y=W_2/H$.



$$F_{12}=\frac{1}{\pi xy}\left[\ln\frac{x_1^2y_1^2}{x_1^2+y_1^2-1}+2x\left(y_1\arctan\frac{x}{y_1}-\arctan x\right)+2y\left(x_1\arctan\frac{y}{x_1}-\arctan y\right)\right]$$

with $x_1\equiv\sqrt{1+x^2}$ and $y_1\equiv\sqrt{1+y^2}$

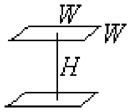
(e.g. for $x=y=1$, $F_{12}=0.1998$)



PPSF :Square Plate to Parallel Plate, Equal sides.

Y:	H	Distance to center
X:	W	Side length

Between two identical parallel square plates of side L and separation H, with $w=W/H$.

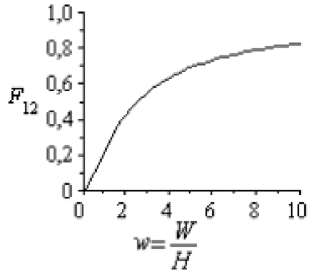


$$F_{12}=\frac{1}{\pi w^2}\left(\ln\frac{x^4}{1+2w^2}+4wy\right)$$

with $x\equiv\sqrt{1+w^2}$ and

$$y\equiv x\arctan\frac{w}{x}-\arctan w$$

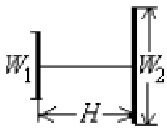
(e.g. for $W=H$, F_{12}=0.1998)



PQSF : Squared Plate to Plate, Unequal sides.

Z:	H	Distance to center
Y:	W1	Side 1 lengths
X:	W2	Side 2 length

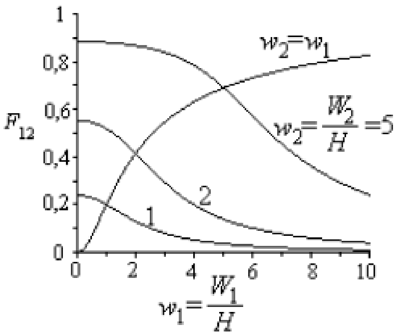
From a square plate of side W_1 to a coaxial square plate of side W_2 at separation H, with $w_1=W_1/H$ and $w_2=W_2/H$.



$$F_{12}=\frac{1}{\pi w_1^2}\left(\ln\frac{p}{q}+s-t\right), \text{ with}$$

$$\begin{cases} p\equiv\left(w_1^2+w_2^2+2\right)^2 \\ q\equiv\left(x^2+2\right)\left(y^2+2\right) \\ x\equiv w_2-w_1, \quad y\equiv w_2+w_1 \\ s\equiv u\left(x\arctan\frac{x}{u}-y\arctan\frac{y}{u}\right) \\ t\equiv v\left(x\arctan\frac{x}{v}-y\arctan\frac{y}{v}\right) \\ u\equiv\sqrt{x^2+4}, \quad v\equiv\sqrt{y^2+4} \end{cases}$$

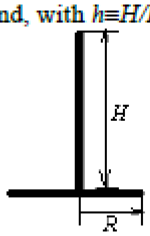
(e.g. for $W_1=W_2=H$, F_{12}=0.1998)



RDF : Rod to Concentric Disc

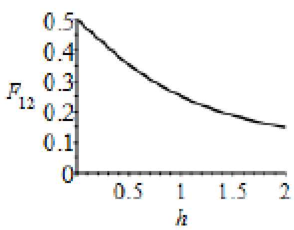
Y:	H	Distance to center
X:	W	Side length

Thin rod (1) of height H, to concentric disc (2) of radius R placed at one end, with $h\equiv H/R$.



$$F_{12}=\frac{1}{4}-\frac{1}{2\pi}\arcsin\frac{h^2-1}{h^2+1},$$

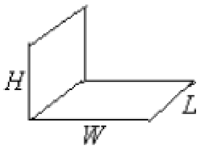
(e.g. for $R=H$, i.e. h=1, F_{12}=1/4)



RR90F : Perpendicular to Horizontal Rectangles,

Z:	H	Rect-1 height
Y:	W	Rect.-2 Height
X:	L	Common Width

From a horizontal rectangle of $W \cdot L$ to adjacent vertical rectangle of $H \cdot L$, with $h=H/L$ and $w=W/L$.

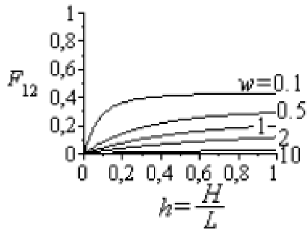


$$F_{12}=\frac{1}{\pi w}\left[h\arctan\left(\frac{1}{h}\right)+w\arctan\left(\frac{1}{w}\right)-\sqrt{h^2+w^2}\arctan\left(\frac{1}{\sqrt{h^2+w^2}}\right)+\frac{1}{4}\ln\left(ab^{w^2}c^{h^2}\right)\right]$$

$$\text{with } a=\frac{(1+h^2)(1+w^2)}{1+h^2+w^2},$$

$$b=\frac{w^2(1+h^2+w^2)}{(1+w^2)(h^2+w^2)},\quad c=\frac{h^2(1+h^2+w^2)}{(1+h^2)(h^2+w^2)}$$

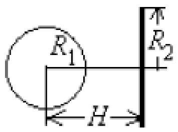
(e.g. for $h=w=1$, $F_{12}=0.20004$)



SDF : Sphere to Disc

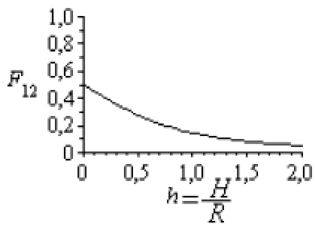
Z:	H	Rect-1 height
Y:	W	Rect.-2 Height
X:	L	Common Width

From a sphere of radius R_1 to a frontal disc of radius R_2 at a distance H between centres (it must be $H>R_1$, but does not depend on R_1), with $h \equiv H/R_2$.



$$F_{12}=\frac{1}{2}\left(1-\frac{1}{\sqrt{1+\frac{1}{h^2}}}\right)$$

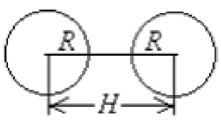
(e.g. for $R_2=H$ and $R_1 \leq H$, $F_{12}=0.146$)



SSF : Sphere to Sphere

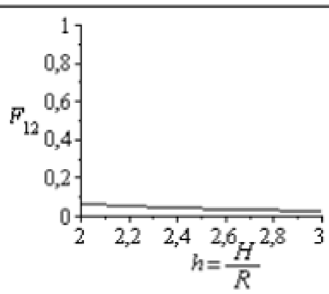
Y:	H	Distance btw. Centers
X:	R	Sphere Radius

From a sphere of radius R to an equal sphere at a distance H between centres (it must be $H>2R$), with $h \equiv H/R$.



$$F_{12}=\frac{1}{2}\left(1-\sqrt{1-\frac{1}{h^2}}\right)$$

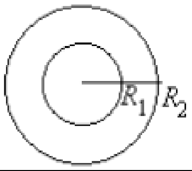
(e.g. for $H=2R$, $F_{12}=0.067$)



SSCF : Concentric Sphere to Sphere

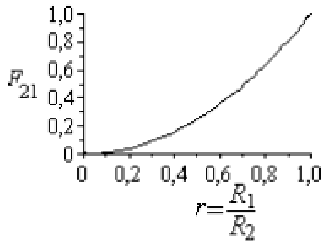
Y:	R1	Inner Sphere radius
X:	R2	Outer Sphere Radius

Between concentric spheres of radii R_1 and $R_2>R_1$, with $r\equiv R_1/R_2<1$.



$F_{12}=1$
$F_{21}=r^2$
$F_{22}=1-r^2$

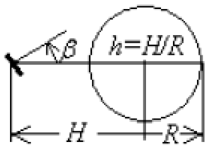
(e.g. for $r=1/2$, $F_{12}=1$, $F_{21}=1/4$, $F_{22}=3/4$)



TPSF : Tilted Plane to Sphere

Z:	H	Distance to center
Y:	R	Sphere radiust
X:	β	Tilt angle

From a small planar plate tilted to a sphere of radius R, at a distance H from centres, with $h\equiv H/R$; the tilting angle β is between the normal and the line of centres.



-if $|\beta|<\pi/2-\arcsin(1/h)$ (i.e. $h\cos\beta>1$),

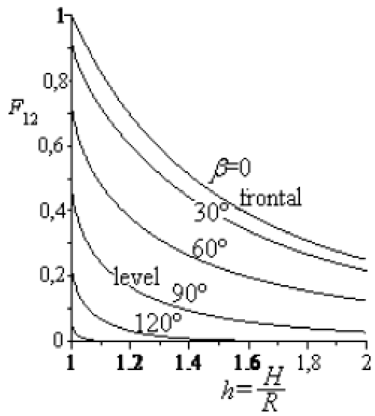
$$F_{12}=\frac{\cos\beta}{h^2}$$

-if not,

$$F_{12}=\frac{1}{\pi h^2}\left(\cos\beta\arccos y-x\sin\beta\sqrt{1-y^2}\right)+\frac{1}{\pi}\arctan\left(\frac{\sin\beta\sqrt{1-y^2}}{x}\right)$$

with $x\equiv\sqrt{h^2-1},\quad y\equiv -x\cot(\beta)$

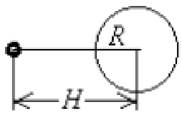
(e.g. for $h=2$ and $\beta=\pi/4$ (45°), $F_{12}=0.177$)



WCF : Wire to Cylinder

Y:	H	Distance to Center
X:	R	Sphere Radius

From a small infinite long cylinder to an infinite long parallel cylinder of radius R, with a distance H between axes, with $h\equiv H/R$.



$$F_{12}=\frac{\arcsin\frac{1}{h}}{\pi}$$

(e.g. for $H=R$, $F_{12}=1/2$)