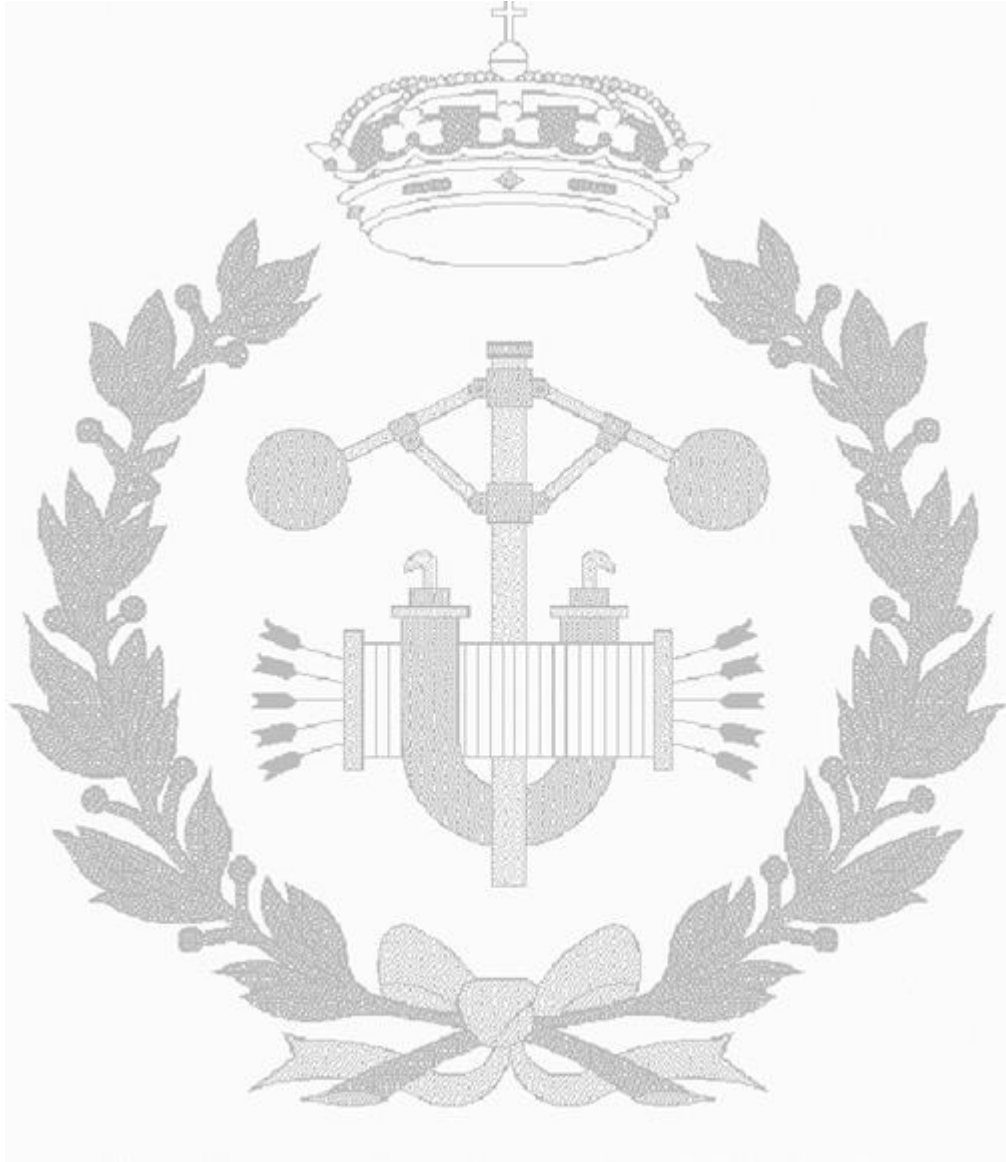


The ETSII Collection
Engineering Programs for the HP-41

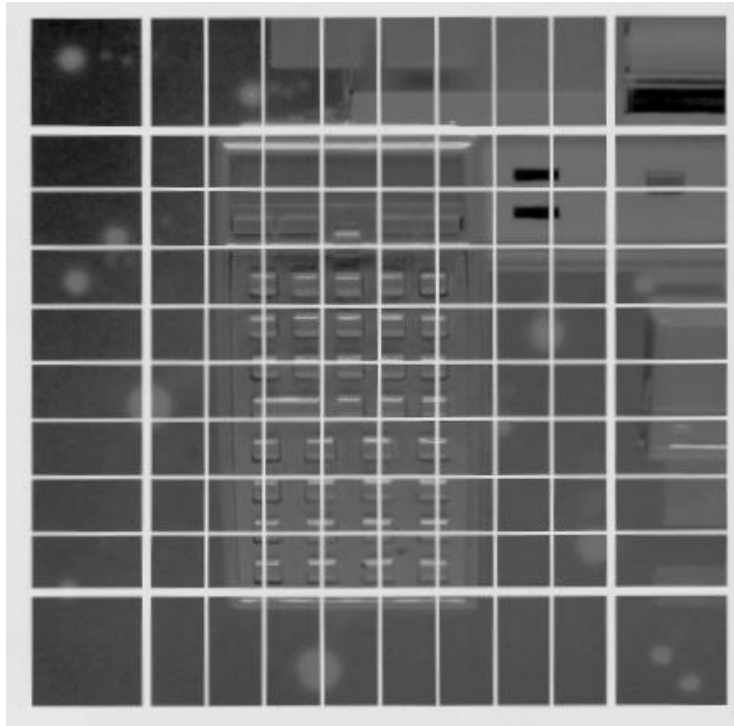


Escuela Técnica Superior de Ingenieros Industriales

Written and programmed by
Ángel M. Martín-Cañas

This compilation revision 1.2.1

Copyright © 2016-2017 Ángel Martin



Published under the GNU software licence agreement.

Original authors retain all copyrights, and should be mentioned in writing by any part utilizing this material. No commercial usage of any kind is allowed.

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

1 - Mechanical Engineering.

1. Orbital Trajectories	9
2. Planar Motion study	10
3. Holzer Method for natural vibrations	11
4. Sag and tension in overhead lines	12
5. RPM-Torque-Power	14
6. Simple beams: Reactions in supports	15
7. Dynamic Balancing in one and two Planes	16
8. 2-D Temperature Distribution in vertical plates	18
9. Transients in wide plates with step temperature change	19
10. Transients in long rods with step temperature change	20
11. Stationary Heat transfer through Fins	23
12. Natural Convection: Nusselt Number	25
13. Radiation View Factors: Rectangles, Discs	26
14. Heat Exchangers basic equation	27
15. Moments of Inertia	27

2. – Thermodynamics and Fluid Mechanics.

1. Gas Liquefaction cycles	31
2. Ideal process of perfect gas w/ $C_p=C_p(T)$	34
3. Non-isentropic expansion into Vapor dome	35
4. Properties of superheated and saturated Steam	36
5. Principle of Corresponding States	39
6. Properties of Refrigerant Gases	41
7. Psychrometric properties of humid Air	46
8. Temperature-Composition for Binary Mixtures	48
9. Hydrodynamic film lubrication	50
10. Stoke's First problem	51
11. Colebrook-White Equation	52
12. Pipes in series with multiple demands	53
13. Rotation speed of multi-nozzle sprinklers	54
14. Lift forces on Joukowski airfoils	55
15. Water hammer transients (simplified)	57

16. Bergeron's method for Pressure pulses	59
17. Association of Pumps in parallel	60
18. Axial velocity at exit of vane profiles	61
19. Centrifugal Pump volute design	62
20. Pipe network analysis – Hardy Cross method	64

3. - *Electrical Engineering.*

1. Association of Resistors / Capacitors	69
2. 2-Port Network Matrix transformations	71
3. 3-Phase Symmetrical Components	72
4. Mutual Induction of coaxial loops	73
5. Power-Flow Equations: Gauss-Seidel	74
6. Synchronous Machine Swing Equation	76
7. Single-phase AC Regulator design	78
8. Electric Circuits Analysis with X-Mem Files	80
9. Backwards Ladder Analysis Program	85
10. Electric Circuits using the Advantage Module	94
11. Interpretation of Logical Networks	97
12. Truth table of logical networks	100
13. Decibel Addition/Subtraction	101
14. Millman's Equivalence Theorem	102
15. Transmission Line Impedances	103
16. Network Frequency Response analysis	104
17. Transfer Function Parameters	107



Introduction.

This collection includes most of the programs written by the author while attending engineering school, many moons ago. The subjects comprise diverse areas in mechanical and electrical engineering, ranging from very simple code snippets to more sophisticated structures and algorithms.

The collection is spread across four plug-in modules, each 8k in size. The modules also include a few programs from the European User's Library and other sources, dealing with similar or complementary topics. Some of these programs (but not all) are also documented in this manual. A top-level rough categorization of the collection sections follows:

- ETSII-3A – General Thermodynamics and Steam properties
- ETSII-3B – Steam properties and Liquefaction Cycles
- ETSII-4A – Fluid Dynamics and Water Pumps
- ETSII-4B – Dynamic Balancing, Mechanical methods and Heat Transfer
- ETSII-5A – Electrical Engineering (mostly power systems)
- ETSII-5B – Circuit and Ladder Analysis
- ETSII-6A – Control Systems and Numerical Methods
- FORFEE – Air Conditioning Loads and Water Well Profiling

Back then documentation wasn't something I spent much time on, so now (30 years later) it's been a bit challenging remembering all the intricacies of the programs. I've tried to include the most relevant points of each program, as well as provide application examples to guide the users. Still, there are a few programs I don't have a real inkling of their exact purpose, let alone how to use them.

Module Dependencies.

Each module is independent from the others, and contains most of the resources needed; such as dedicated MCODE functions and subroutines. The programs make profuse utilization of extended functions; thus you need the **X-Functions** module or (better yet) a CX. Obviously you'll benefit immensely using the 41CL or an emulator in turbo speed, in particular for those programs requiring more number-crunching resources.

Besides that, some programs use routines from the "**Unit Conversion**" module, a stand-alone ROM based on HP's Unit Management Facility (UMF) although strongly enhanced with electrical units and user-friendly catalogs and routines. This is especially useful for subjects involving thermal magnitudes, where the units frequently get in the way of the solution and are a source of errors.

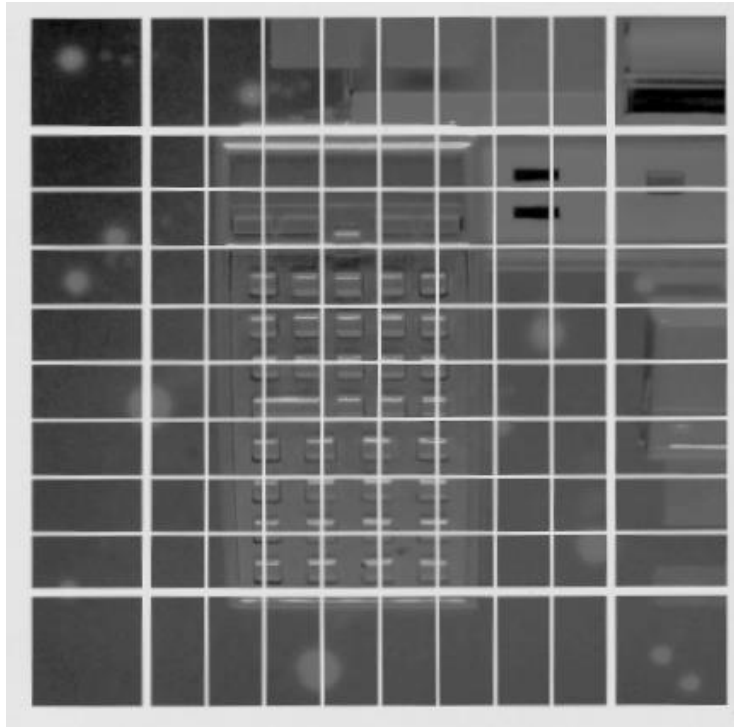
Another common thread is the use of utility functions from the **AMC_OS/X** module – I simply couldn't resist enhancing the U/I and data entry routines. The benefit is not only cosmetic, as the numerous byte savings have been instrumental to add more programs in the modules.

Finally, the **SandMath** module is also required for a few cases, like the Bessel functions used in the Heat transfer section. Note however that the ETSII modules have built-in FOCAL root-finding and integration routines, which (with few exceptions) are used instead of SOLVE/INTEG from the advantage (or FROOT/FINTG from the SandMath).



Mechanical Engineering.



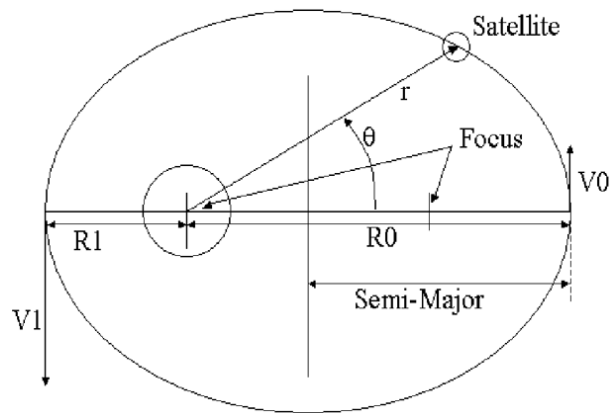


Orbital Trajectories. [GRVTY]

From the author's Engineering Collection, included in the ETSII3 module.

These short routines calculate a few parameters of orbital trajectories when the radius "Ro", gravity constant "go" and distance from earth "r" values are known.

In addition to the planet's radius R_o and surface gravity (g_o), for elliptical orbits typically the known values include the perigee (p) and apogee (h) of the orbit - therefore the ellipsis major axis can be determined with the expression: $a = 2R_o + p + h$



The initial choice is for the type of unknown, either the orbit eccentricity, the period or the velocities – each one of them also requiring additional data input values as per the table below.

Eccentricity "e"	Period "T"	Velocities "C:E:H:P"
Initial angle	Major semi-axis	Major axis -> Ve
Initial velocity		Eccentricity -> Vh

Let "r" the distance to the surface of the planet and "α" the angle of the launch velocity vector and the tangent to the orbit (or horizon from earth). The formulas used are as follows:

$$\text{Eccentricity: } e = \sqrt{1 + [V_o^2 - 2g_o(R_o^2/r)] \cdot [r V_o \cos \alpha / g_o R_o]^2}$$

$$\text{Elliptic orbit: } V_e = \sqrt{2(2a - r) \cdot [g_o R_o^2 / r]}$$

$$\text{Hyperbolic: } V_h = \sqrt{(1+e) \cdot [g_o R_o^2 / r]}$$

$$\text{Parabolic: } V_p = \sqrt{2 g_o R_o^2 / r}$$

$$\text{Circular: } V_c = \sqrt{g_o R_o^2 / r}$$

$$\text{Period: } T = (2\pi / R_o) \sqrt{a^3 / g_o}$$

Example.

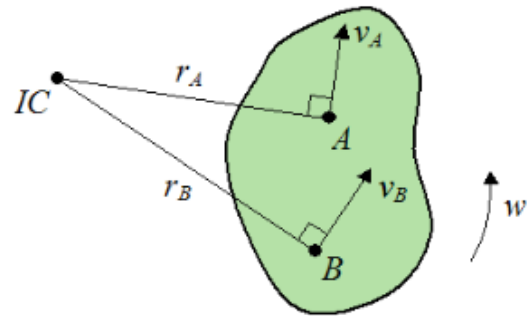
A satellite is orbiting earth in an elliptic orbit with 1,120 km apogee and 120 km perigee. Using $R_o = 6.380$ km determine the new eccentricity if its velocity changes to 9,596 m/s forming an angle of 4.1 deg over the horizontal, when the distance from earth is $r = 6,964.43$ km

The solutions are:

Planar Movement Study. [MVPLN]

From the author's Engineering Collection, included in the ETSII3 module.

For rigid bodies experiencing general plane motion (in two-dimensions), the concept of instant center allows one to conveniently calculate the unknown angular velocity of the rigid body, or unknown linear velocities of points on the rigid body. The instant center is an imaginary point that allows for a mathematical “shortcut” in calculating these unknowns.



The program characterizes the acceleration pole for a two-body configuration, when the kinematic properties are known. The first distinction is whether both the fixed and moving centrodes (i.e. locus of the instant centers of rotation) are on the same side of the common tangent. Other known data values are the rotation speed and the radius of each curve, as well as the acceleration of the instant center.

The results include the succession velocity of the instant center of rotation, V_s , the inversion and inflexion diameters, and the position of the rotation pole (magnitude and angle). The equations used are shown below:

Let R_o and R be the radius of the fixed and moving centrodes, ω the angular velocity of the body, ω' the acceleration at the instant. The formulas used are as follows:

$$\begin{aligned} V_s &= \omega R_o R / (R_o + R) \\ |\text{pole}| &= \omega V_s / \sqrt{(\omega')^2 + \omega^4} \\ \text{Pole}(\angle) &= \text{atan} [\omega^2 / \omega'] \\ D_{\text{inver}} &= 2 (\omega V_s / \omega') \cos \theta \\ D_{\text{inflex}} &= 2 (V_s / \omega) \sin \theta \end{aligned}$$

Example.

Characterize the motion parameters for a planar mechanism with centrodes at different sides, rotation at an angular velocity of 24 rad/s, and an angular acceleration of 139.36 rad./s² if the radius of the fixed and moving centrodes are 12 and 6 m respectively.

The solutions are:

$$V_s = 96$$

$$D_{\text{inver}} = 16.5327 \text{ m}$$

$$D_{\text{inflex}} = 4 \text{ m}$$

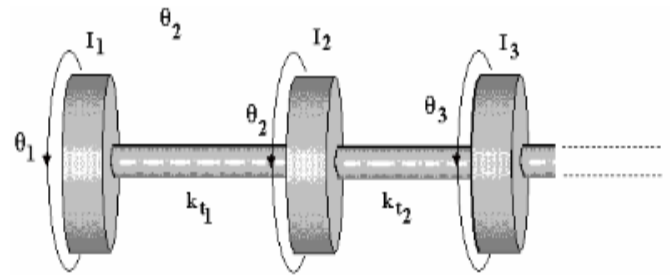
$$\text{Acceleration Pole located at: } 3.8878 \angle 76.3990$$

Holzer method for natural vibrations. [HOLZER]

From the author's Engineering Collection, included in the ETSII4 module.

This program calculates the natural vibration frequencies of a semi-definite mechanical system with N degrees of freedom using the Holzer method.

The vibration can be linear (lineal displacements in the springs) or torsional (angular displacements in the shaft). The vibration modes are also obtained for each natural frequency.



The natural frequencies w are the roots of the frequency function, defined as follows:

$$g(w) = w^2 \sum \{ I_j D_j(w) \} ; j = 1, 2, \dots, N$$

Where D_j is also a function of w and of the previous displacements, according to the expression:

$$D_j = D_{j-1} - [w^2 / K_{j-1}] \sum \{ I_n D_n \} ; n = 1, 2, \dots, j \text{ and } D_1 = 1$$

The terms K_j represent the stiffness constants (elastic or torsional) in the unions between the element masses – typically springs or the shaft depending on the case.

The program offers an initial approximation for the main natural frequency that can be used as guess for the root-finding routine – which is included in the module as well.

Examples.

Calculate the first three natural frequencies and modes of oscillation for a system of 5 rotors connected by a shaft, knowing that the angular momentum is $I = 1 \text{ kg.m}^2$ for all of them. The torsional stiffness of the shaft is $k = 2 \text{ N.m}$

The solutions are shown on the table below:

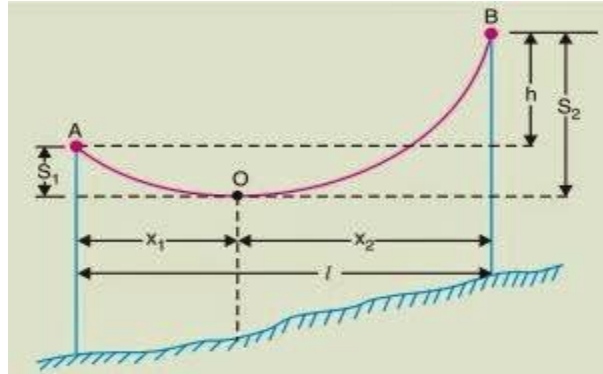
w	m1	m2	m3	m4	m5
w1 = 0.8740	1	0.618	0	-0.618	-1
w2 = 2.2882	1	-1.618	0	1.618.	-1
w3 = 2.6900	1	-2.618	3.2361	-2.61	1

Sag and Tension in Overhead Lines. [CAMELA]

From the author's Engineering Collection, included in the ETSII5 module.

This program calculates the sag and tensions at the supports of overhead line cables, with or without equileveled conditions. The conductor adopts a catenary shape in either case, but the different geometric conditions require different methods to resolve the unknowns.

Besides the slope, posts height and span length the input data includes the minimum (perpendicular) distance to the ground, which occurs at the point of maximum deflection of the cable, thus limiting the maximum sag.



Let V = span length; H = posts height; $m = \tan\theta$ = inclination slope; d = minimum distance (safety)

For level spans the maximum sag occurs at its middle point, with a symmetric catenary curve centred there ($x_f = 0$). Thus the coordinates of posts are $X_a = -L/2$ and $X_b = L/2$. The curve equation in that case is:

$$(H - d) = \alpha [1 + \cosh(-V/2\alpha)]$$

For unlevel spans, the following two equations are used to calculate the values X_a and X_f , the coordinates of the post at lower slope and the point of maximum sag:

$$\begin{aligned} (1) \quad V m &= (x_f / \alpha \sinh m) \{ \cosh(A) \cdot \cosh(B-1) + \sinh(B) \sinh(A) \} \\ (2) \quad f &= m(x_f - x_a) + (x_f / \alpha \sinh(m)) [\cosh(A) - \cosh(\alpha \sinh(m))] \end{aligned}$$

Where: $A = X_a \alpha \sinh(m) / X_f$; $B = V \alpha \sinh m / X_f$; and
 $f = H - d / \cos\theta$ is the maximum sag.

Solving this system for X_a and X_f determines the rest of unknowns, such as $X_b = L - |X_a|$; The resolution is done numerically using "SLV2", a built-in routine to solve non-linear systems of two equations.

The program output includes both geometry and stress results. The geometry results are the X-coordinates of each post referred to the point of maximum deflection ($x=0$), and the alpha parameter of the catenary curve.

The stress results require the unitary weight of the cable (q) , returning the horizontal tension in the supports (T_a , T_b) and maximum sag point (T_0), as well as the total length of the cable (L). The expressions used are derived from the basic catenary, as follows:

$$T = q \alpha \cosh x/\alpha; \text{ and: } L = [\sinh x_b / \alpha - \sinh x_a / \alpha]$$

Example.

Calculate the tensions in the supports for an overhead power line with 100 m span length, with 42 m height posts and a minimum perpendicular distance to ground of 10 m. Do both cases of level span and 20 deg inclined span to compare the results. The unitary weigh is 10 kg/m.

Level span	xa = -50	xb = 50	$\alpha=43.5470$	
q = 10 kg/m	TA=755,4702	TB=755,4702	T0=435,4702	L=123,4667
Inclined 20 deg	XA=-42,8106	XF=8,7983	$\alpha = 44.2816$	
q = 10/kg/m	TA=666,3870	TB=866,3870	T0=442,8156	L=124,2657

Hyperbolic Functions. [SINH, COSH]

Included in the module are stand-alone MCODE routines to calculate the hyperbolic sine and cosine. They use 13-digit math subroutines from the OS for enhanced accuracy. Just enter the argument in X, execute the function and the result is placed in X (stack is lifted) and the original argument is saved in LastX.

Header	AFD0	088	"H"	
Header	AFD1	00E	"N"	$sh(x)=1/2[e^x-e^{-x}]$
Header	AFD2	009	"I"	
Header	AFD3	013	"S"	Ángel Martin
SINH	AFD4	248	SETF 9	
	AFD5	033	JNC +06	[MAIN]
Header	AFD6	088	"H"	
Header	AFD7	013	"S"	$ch(x)=1/2[e^x+e^{-x}]$
Header	AFD8	00F	"O"	
Header	AFD9	003	"C"	Ángel Martin
COSH	AFDA	244	CLRF 9	
MAIN	AFDB	0F8	READ 3(X)	Go noisy!
	AFDC	361	?NC XQ	(this includes SETDEC)
	AFDD	050	->14D8	[CHK_NO_S]
	AFDE	044	CLRF 4	
	AFDF	029	?NC XQ	
	AFE0	068	->1A0A	[EXP10]
	AFE1	089	?NC XQ	e^x
	AFE2	064	->1922	[STSCR]
	AFE3	239	?NC XQ	e^{-x}
	AFE4	060	->188E	[ON/X13]
	AFE5	24C	?FSET 9	true if SINH
	AFE6	01B	JNC +03	
	AFE7	2BE	C=-C-1 MS	Sign change
	AFE8	11E	A=C MS	ditto in A
	AFE9	0D1	?NC XQ	e^x
	AFEA	064	->1934	[RCSCR]
	AFEB	031	?NC XQ	
	AFEC	060	->180C	[AD2-13]
	AFED	04E	C=0 ALL	
	AFEE	35C	PT=12	build "2" in C
	AFEF	090	LD@PT- 2	
	AFF0	269	?NC XQ	
	AFF1	060	->189A	[DV1-10]
	AFF2	331	?NC GO	Overflow, DropST, FillXL & Exit
	AFF3	002	->00CC	[NFRX]

RPM-Torque-Power. [RPMTP]*From the author's Engineering Collection, included in the ETSII4 module.*

HP-41 version of the program first available for HP-29/19C solutions. A classic mini-equation solver for one of the three variables with the other two known.

$P = \omega M$, with:

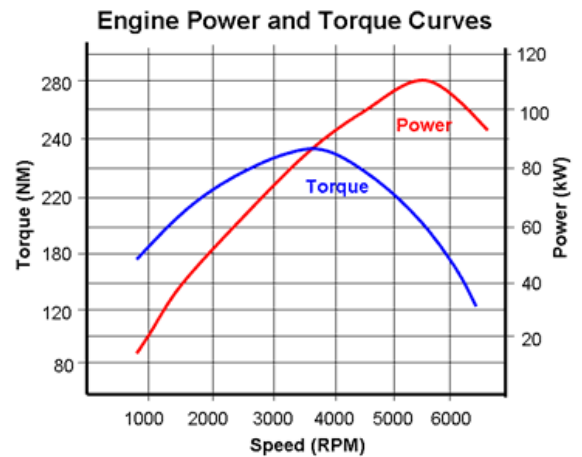
$\omega = \text{rpm} \cdot 2\pi/60$ in rad/s and

$M = \text{torque in N.m}$

Two unit systems are possible: SI and British. Answer Y/N to the "S.I.?" Prompt to select.

S.I.? Y/N
USER ALPHA

RPM TQ PW
USER



Not much to add here, just follow the prompts to select the choices provided by the calculator. The calculation can be repeated for multiple values of the variables and different choices of the unknown.

S.I.	British
rpm	rpm
N.m	ft.lb
W	hp

Example:

Calculate the power in watts for a torque of 20 Nm and angular speed of 50 rpm

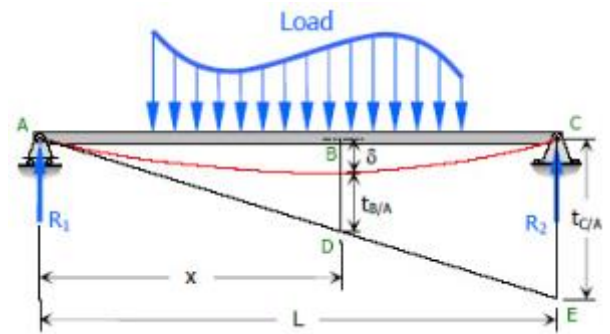
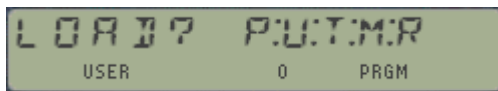
The solution is $P = 11 \text{ kW}$

Note. Using the "Unit Management System" included in the Unit Conversion Module is a vast superior approach to perform unit conversions like this one.

Simple beams: Reactions in Supports. [R2SP]

From the author's Engineering Collection, included in the ETSII4 module.

This program calculates the reactions in the supports of a simple beam subjected to any combination of the following efforts: point, uniform, triangular loads and external moments. Trapezoidal loads can be expressed as a combination of a uniform load plus a triangular one.



The program will prompt for the load type to enter next, with the following message “LOAD? P:U:T:M:R”. Press the corresponding key for each load type and use it as many times as loads exist, then press “R” to calculate the reactions.

The supports can be placed at any two points along the beam's distance - xa, xb - taking the left end as origin of coordinates.

The expressions used by the program are a straight application of statics. Let Ra and Rb be the reaction in the supports; Pj and Mj the different point loads and external moments, applied at a distance xj. (j=1,2...n). Let q be the load per unit length of uniform and triangular loads applied between distances (x1,x2), and “m” the slope of the triangular load. Then we have:

$$(1) \quad \sum F = R_a + R_b + \sum q (x_2 - x_1) + \sum q/2 (x_2 - x_1)^2$$

$$(2) \quad \sum M_j = x_a R_a + x_b R_b + \sum x F_j + \sum q/2 (x_2^2 - x_1^2) + \sum \{ [m (x_2^3 - x_1^3)/3 - (m x_1 / 2) (x_2^2 - x_1^2)] \}$$

Example.

Calculate the reactions in the supports of a simple beam with the following configuration:

Xa = 2 m, Xb = 6m;

External moment M1 = 200 N.m (clockwise is positive)

Uniform loads q1=120 N/m between x1=1 and x2=3

Uniform load q2 = 300 N/m between x1=3 and x2=4

Triangular load between x1=6 and x2=7, with final value of q3=100 N/m

The results are:

R1 = 485, 84 N

R2 = 4, 1667 N

Dynamic Balancing in 1 and 2 planes. [by Eugenio Úbeda]

From the author's Engineering Collection, included in the ETSII4 module.

These programs can be used to characterize the rotating vibrations in trial tests (vector coefficients in g/s) and to calculate the corrective weights to compensate for torsional vibrations in stationary regimes. The programs allow for single or two-plane corrections, where typically the single plane is restricted to systems with shafts not longer than their diameters.

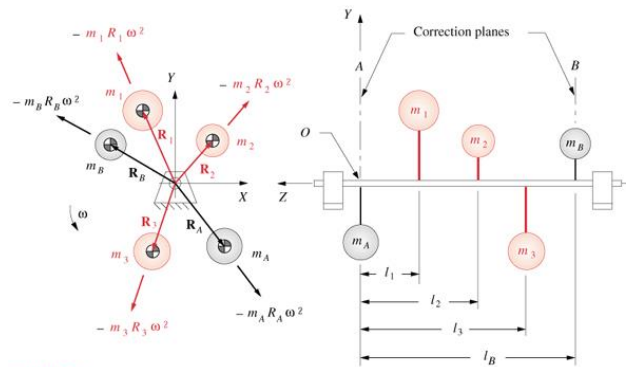


FIGURE 12-3
Two-plane dynamic balancing

For Single-plane balancing the required data are the initial vibration (μm), the trial weight (g < deg) and the resulting trial vibration.

For 2-plane balancing, the required data are the initial vibrations on each plane, but the trial tests are only needed if the system coefficients are not already known. The results obtained from the trial tests can be saved in an X-Memory file and reused repeatedly in successive iterations of the corrective weight calculations (magnitude and position). These iterations can be repeated as often as required until the final vibration is within the accepted limits. The program also offers the possibility to enter the characteristic coefficients matrix manually – should their values are known but not currently in the X-Mem file.

Data entry is expected with the magnitude first, and then the position - separated by ENTER^ . The angles are referred to the chosen origin and must follow a consistent convention as per their orientation. This applies equally to the vibrations (initial and actual) and weights (total and correcting).

Example1.

Using the 1-plane balancing method, calculate the corrective weight and its position to compensate for an initial vibration measured like $155 \mu\text{m}$ at 30° degrees. The trial test was made using a weight of $200 \mu\text{m}$ at 0° deg position, which caused the trial vibration to be $35 \mu\text{m}$ and 120° deg.

The results are shown below:

Vector coeff : $S1 = 1.258634 \text{ <} 342.724356$
Correcting weight: $W1 = 44 \text{ <} 103$

If the measured residual vibration is still $V = 12 \text{ <} 130$.

Running a second iteration results in the additional results below:

Vector coeff : $S2 = 1.892619 \text{ <} 347.860674$
Correcting weight: $W2 = 23 \text{ <} 118$
Total weight: $Wt = 190 \text{ <} 19$

Example2.

Using the 2-plane balancing technique calculate the corrective weights and their positions to compensate for initial vibrations measured on each plane as: $7\ \mu\text{m}$ at 80 degrees and $5\ \mu\text{m}$ at 130 deg. The trial tests were made using weights of $375\ \mu\text{m}$ at 180° deg position on each plane, which caused the trial vibrations to be as shown below:

Trial weights	Plane-1 vibration	Plane-2 vibration
375 <) 180 in Plane 1	10.2 <) 25	8.5 <) 15
275 <) 180 in Plane 2	13 <) 50	9.5 <) 10

Results. The program calculates the system vector coefficients, which get stored in an X-memory file named "COEFFS". This file can be used later instead of the trial tests, as it characterizes the unbalance behavior of the system.

S11 = 64.768616 <) 73.384289	S12 = 39.451436 <) 286.455879
S21 = 58.588379 <) 255.104623	S22 = 42.819398 <) 65.392443

And the correcting weights are shown below:

$$P1' = 472 \text{ <) } 129$$

$$P2' = 283 \text{ <) } 306$$

If the measured residual vibrations are still $V1 = 1 \text{ <) } 85$ and: $V2 = 2.5 \text{ <) } 110$

Running a second iteration results in the additional results below:

$$P1'' = 85 \text{ <) } 77$$

$$P2'' = 53 \text{ <) } 192$$

For an equivalent total corrective weight of:

$$Pt1 = 529 \text{ <) } 122$$

$$Pt2 = 266 \text{ <) } 295$$

Note: The program includes 4 functions to perform arithmetic operations in polar mode, with the complex numbers entered in the stack registers as two pairs of {argument, ENTER^, module}; like in the standard P-R convention of the calculator. Their names are "W+", "W-", "W*", and "W/".

2D Temperature Distribution in vertical plates. [TXY]

From the author's Engineering Collection, included in the ETSII4 module.

This program calculates the temperature distribution $T(x,y)$ within a rectangular vertical plate with dimensions $(b \times h)$; with three sides maintained at a constant temperature T_0 , and with a known temperature distribution on its upper side - either constant T_2 or varying with x - $T(x,h)$.

Therefore it's said that the plate is immersed in a uniform ambient temperature T_0 , while the fourth side is maintained at another constant temperature or temperature distribution.

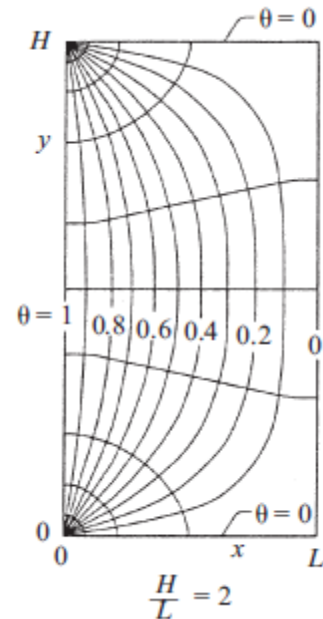
The expression used is based on an infinite sum as follows:

$$T(x,y) = T_0 + 2/b \sum T(n,x) ; n= 1,2,...$$

with the following general term, where $\lambda_n = \pi n / b$

$$T_n(x,y) = \text{sh}(\lambda_n y) \cdot \sin(\lambda_n x) / \text{sh}(\lambda_n h) \text{ INTG } \{ T(x,b) - T_0 \} \sin \lambda_n t \text{ dt ; between } [0, b]$$

The numerical integration is done using the ITG routine also included in the module.



Example:

Calculate the temperature in the points $P(1, 2)$ and $Q(2, 3)$ within a flat plate of dimensions (2×5) m, with a temperature distribution on its top side given by the function: $t(x,5) = x^2 + 10$ deg C. The ambient temperature is $t_0 = 10$ deg C. Compare the result with the case of a constant temperature on the top side $t(x, 5) = 100$ deg C.

The solutions are shown below.

Point	$T(x,5)=100$	$T(x, 5) = x^2 + 10$
P(1, 2)	$T(1, 2) = 10.0239$ C	$T(1, 2) = 10,01357914$
Q(2, 3)	$T(2, 3) = 10$	$T(2, 3) = 10,00$

The temperature function for the second case can be easily programmed:

```

01 LBL "TX"
02 X^2
03 10
04 +
05 END

```

Transients in wide plate with step temperature change [TXT]

From the author's Engineering Collection, included in the ETSII4 module.

This program calculates the temperature $T(x)$ in points x of an infinite flat plate with finite thickness ($2L$), after experiencing a thermal shock - or sudden change of ambient temperature, from T_0 initial to T_f final.

The Biot number is provided indirectly, by means of the heat transfer (or film)

coefficient: $h = Bi \cdot K / L$. The thermal conductivity (K) and thermal diffusivity (α) must also be known, where: $\alpha = K / \rho \cdot Cp$ - i.e. thermal conductivity over the density and specific heat capacity.

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

$$T(x,t) = T_f + 2 (T_0 - T_f) \sum \{ f(x, n) \exp[-\alpha t \cdot (\lambda_n / L)^2] \}; \quad n = 1, 2, \dots$$

$$\text{With: } f(x, n) = \sin(\lambda_n) \cdot \cos(\lambda_n \cdot x / L) / [\lambda_n + \sin(\lambda_n) \cos(\lambda_n)]$$

And (λ_n) are the n roots of the equation defined by: $\tan(\lambda_n) = Bi / (\lambda_n)$

Which in the program has been replaced by its equivalent form:

$$(-1)^n \cos(\lambda_n) + \lambda_n / [\sqrt{\lambda_n^2 + Bi^2}] = 0$$

Solved using the SLV routine, using the truncation of the tangent to its first two terms and using as initial guess: $(\lambda_n)_{init} = \pi(n-1) + \sqrt{(3/2) [\sqrt{1+4h/3} - 1]}$

Example.

A 20 cm thick wide plate 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 one, two and three hours after the sudden step temperature change. The physical properties of the material are given below:

$$\alpha = 1.66 \text{ E-6 m}^2/\text{s}$$

$$h = 20,000 \text{ kcal/H.m}^2\text{.C} = 23,260.0 \text{ W/m}^2\text{ K}$$

$$K = 100 \text{ kcal/h.m.C} = 116.30 \text{ W/m.K}$$

The results are shown in the table below: (warning: very slow convergence!)

Point	t = 1 hour	t = 2 hours	t = 3 hours
center (x=0 cm)	366.5400	133.0300	71.8500
Outer edge (x=0.1 m)	73.5600	56.2700	51.7100

Transients in long cylinder with step temperature change [TRT]

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 T_0 initial to T_f final.

Similar to the previous 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:

$$T(x,t) = T_f + (T_0 - T_f) \sum (2/\lambda_n) \{ f(n, r) \exp[-\alpha \cdot t \cdot (\lambda_n/R)^2] \} ; \quad n = 1, 2, \dots$$

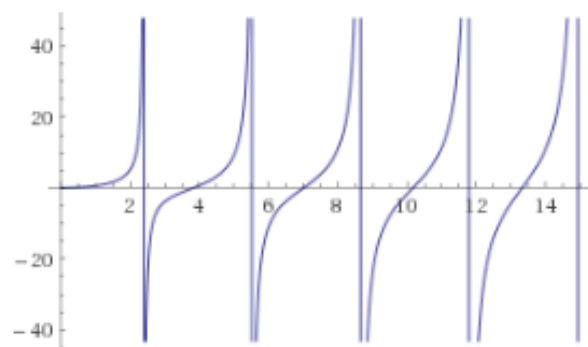
$$\text{With: } f(n,r) = J_1(\lambda_n) \cdot J_0(\lambda_n \cdot r/R) / [J_1^2(\lambda_n) + J_0^2(\lambda_n)]$$

And (λ_n) are the n roots of the equation defined by: $(\lambda_n) J_1(\lambda_n) = Bi J_0(\lambda_n)$

Which, leaving the Biot number alone in the second term, can be expressed as the intersection of the Biot number with the function $x \cdot J_1(x)/J_0(x)$, shown in the graphic on the right, where the asymptotic boundaries will provide a reasonable criteria for the estimations needed by the root-finding routine, as follows:

(λ_1) is between $]2, 4]$
 (λ_{n+1}) is between $](\lambda_n)+1, (\lambda_n)+4]$

Plot:



Example.

A very long metal rod of radius $R=0.14$ has a uniform temperature of 1,000 dec 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 physical properties of the material are given below:

$$\alpha = 1.66 \text{ E-6 m}^2/\text{s}$$

$$h = 20,000 \text{ kcal/H.m}^2\text{.C} = 23,260.0 \text{ W/m}^2 \text{ K}$$

$$K = 100 \text{ kcal/h.m.C} = 116.30 \text{ W/m.K}$$

The results 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 about 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 the λ_n roots, the program uses the symmetric intervals centered at the initial estimation and with distance “one”:

$$[n^*(\lambda_n)_{\text{init}} - 0.5 ; n^*(\lambda_n)_{\text{init}} + 0.5]$$

$$\text{With: } (\lambda_n)_{\text{init}} = \text{sqr}\{ (3/2) [\text{sqr}(1+4\text{Bi}/3) - 1]$$

In this version we've used **FROOT** instead, also included in the SandMath - which was already required for the Bessel functions, so no new dependencies are added. The estimation for the initial guesses becomes very important for the successful root identification, and the execution time – which is 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 times (i.e. longer than a previous execution). In that case there's no need to find additional λ_n roots beyond those already identified, as the contribution of the series terms to the infinite sum will be smaller due to the larger argument in the inverse exponential function:

$$f(n, r) \cdot \exp[-\alpha \cdot t \cdot (\lambda_n/R)^2]$$

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.

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.

1	LBL “?”	6	CF 22
2	RCL IND X	7	PROMPT
3	“ =“	8	FS?C 22
4	ARCL X	9	STO IND X
5	“ -?”	10	END

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.

1	LBL "TRT"		53	,75		105	STO 08	
2	SIZE?		54	/		106	,	
3	21		55	E		107	RCL 07	λn
4	X>Y?		56	+		108	JBS	J_0
5	PSIZE		57	SQRT		109	X^2	J_0^2
6	E		58	E		110	RCL 08	J_1
7	CF 04		59	-		111	X^2	J_1^2
8	"SAVE-RT? YN"		60	1,5		112	+	$J_0^2+J_1^2$
9	PMTK	in OS/X ROM	61	*		113	STO 09	
10	X#Y?		62	SQRT		114	,	
11	GTO 04		63	STO 13	delta	115	RCL 07	λn
12	SF 04		64	"aJ"		116	RCL 03	R0
13	" μN "		65	ASTO 06		117	/	$\lambda n \cdot R0$
14	SF 25		66	XROM "ΣI"	infinite sum	118	RCL 04	r
15	PURFL	purge if there	67	ST+ X		119	*	
16	9	file size	68	RCL 11		120	JBS	$J_0(\lambda n \cdot r \cdot R0)$
17	CF 25		69	*		121	RCL 08	J_1
18	CRFLD	create anew	70	RCL 10		122	*	
19	LBL 04		71	+		123	RCL 09	$J_0^2+J_1^2$
20	"TINI"		72	"T(X,T)="		124	/	
21	11		73	ARCL X		125	RCL 07	λn
22	XROM "?"		74	PROMPT		126	/	
23	"TEND"		75	GTO C		127	LASTX	λn
24	10		76	LBL "aJ"		128	RCL 03	R0
25	XROM "?"		77	A<>RG	preserve	129	/	$\lambda n / R0$
26	ST- 11		78	17	alpha	130	X^2	$(\lambda n / R0)^2$
27	"H"		79	FC? 04	roots in X-Mem?	131	RCL 12	t
28	E		80	GTO 04	no, need to search	132	*	
29	XROM "?"		81	GETX	yes, get current	133	RCL 00	a
30	STO 14		82	X#0?	valid root?	134	*	$\alpha.t.(\lambda n / R0)^2$
31	"K"		83	GTO 05	yes!	135	CHS	
32	2		84	RCLPT	no, backtrack pointer	136	E^X	
33	XROM "?"		85	E		137	*	
34	ST/ 14		86	-		138	A<>RG	restore
35	"RO"		87	SEEKPT		139	17	alpha
36	3		88	LBL 04	need to search	140	RTN	
37	XROM "?"		89	RCL 18	n	141	LBL "JN"	
38	ST* 14	Bi	90	RCL 13	delta	142	E	
39	"ALPHA"		91	*		143	X<>Y	
40	,		92	RCL X		144	JBS	
41	XROM "?"		93	,5		145	X<>Y	
42	LBL C		94	ST- Z		146	ST+ X	
43	0		95	+		147	,	
44	FS? 04	save?	96	"JN"		148	X<>Y	
45	SEEKPT	reset pointer	97	FROOT		149	JBS	
46	"R"		98	FS? 04	save option?	150	ST/ Z	
47	4		99	SAVEX	yes, oblige	151	RDN	
48	XROM "?"		100	LBL 05		152	ST+X	
49	"T"		101	STO 07	λn	153	*	
50	12		102	E		154	RCL 14	Bi
51	XROM "?"		103	X<>Y		155	-	
52	RCL 14		104	JBS	J_1	156	END	

Stationary Heat flow through Fins. [ANULAR, TRIANG, TRAPEZ]

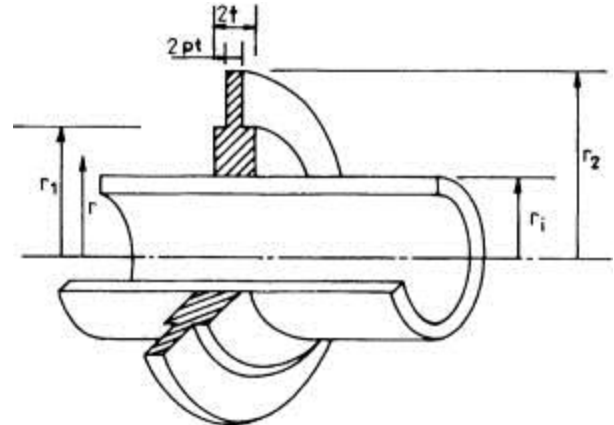
From the author's Engineering Collection, included in the ETSII4 module.

Annular Fins, with thickness w and r_1 , r_2 the internal & external radius respectively.

Let $n = \sqrt{2h / Kw}$,

with h the heat transfer (film) coefficient and K the thermal conductivity.

Let T_0 be the temperature difference between the base ($r = r_1$) and the surrounding cooling fluid.



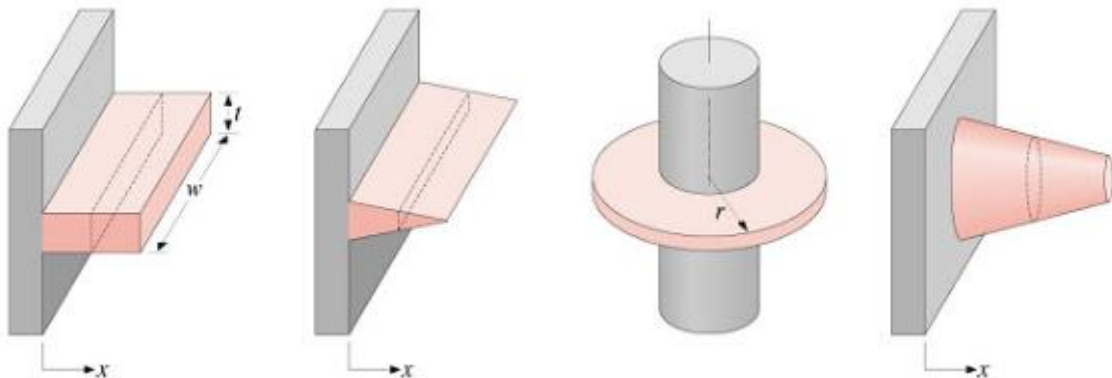
Assuming there's no heat transfer at the fin's tip, the expression for the temperature at a distance r , ($r_1 \leq r \leq r_2$) is given below:

$$T(r)/T_0 = [I_0(nr). K_1(nr_2) + K_0(nr). I_1(nr_2)] / [I_0(nr_1). K_1(nr_2) + I_1(nr_2). K_0(nr_1)]$$

where I , K are the modified Bessel functions of first and second kind.

The expression for the dissipated heat is in this case:

$$Q = 2\pi nKw r_1 T_0 [I_1(nr_2). K_1(nr_1) - K_1(nr_2). I_1(nr_1)] / [I_0(nr_1). K_1(nr_2) - K_0(nr_1). I_1(nr_2)]$$



Straight Fins with trapezoidal or triangular section profiles, with base thickness w and distance " d " to its (fictitious) triangular end point. Taking that end point as origin of coordinates, let x_e be distance to the end of the fin, and the base $x_b = d$

- For a trapezoidal fin the actual length is: $L = (d - x_e)$.
- For a triangular fin $x_e = 0$; and its length is $L = d$

Let $f = \sqrt{1 + (w/2d)^2}$; and: $p = 2 \sqrt{2f \cdot h \cdot d / K \cdot w}$

Let T_0 be the temperature difference between the base and the surrounding fluid (air).

The expression for the corrected temperature (or difference) $T(x)$ at a distance $x \geq x_e$ is given below, denoting $x^* = \sqrt{x}$

$$T(x)/T_0 = [I_0(p \cdot x^*) \cdot K_1(p \cdot x_e^*) + I_1(p \cdot x_e^*) \cdot K_0(p \cdot x^*)] / [I_0(p \cdot d^*) \cdot K_1(p \cdot x_e^*) + I_1(p \cdot x_e^*) \cdot K_0(p \cdot d^*)]$$

Assuming there's no heat transfer at the fin's tip, the expression for the dissipated heat (per unit of depth) is in this case:

$$Q = -(A) [I_1(p \cdot d^*) \cdot K_1(p \cdot x_e^*) - I_1(p \cdot x_e^*) \cdot K_1(p \cdot d^*)] / [I_0(p \cdot d^*) \cdot K_1(p \cdot x_e^*) + I_1(p \cdot x_e^*) \cdot K_0(p \cdot d^*)]$$

with $A = K \cdot w \cdot p (T_b - T_0) / \sqrt{x(d)}$

Note: if you prefer using the base as origin of coordinates, simply replace x by $(d - x)$ in the above expressions.

These programs use the Modified Bessel functions from the SandMath module, which needs to be plugged in the calculator as well.

Examples.

Calculate the temperature at the edge and the total dissipated heat for the following conditions: surrounding temperature $T_{inf} = 30 \text{ deg C}$, base temperature $T_b = 200 \text{ deg C}$. Physical properties: $h = 34.89 \text{ [W/K.m}^2\text{]} ; K = 53.498 \text{ [W/K.m]}$

- a) an annular fin with $r_1=8 \text{ cm}$, $r_2=14 \text{ cm}$; $w=1 \text{ cm}$
- b) a trapezoidal fin with $w=1 \text{ cm}$, $d=14 \text{ cm}$; $x_e=4 \text{ cm}$
- c) a triangular fin with $w=1 \text{ cm}$; $d=14 \text{ cm}$

The results for the corrected temperature ($T(x) - T_{inf}$) are given in the table below:

Fin type	$T_c \text{ (deg C)}$	$Q \text{ [J/s]}$
Annular, $r = 0.14 \text{ m}$	131.1090	409.3259
Trapezoidal ($x=4 \text{ cm}$)	81.5435	799.7711
Triangular ($x=0$)	29.6077	855.8098

Natural Convection Nusselt numbers. [NATCNV]

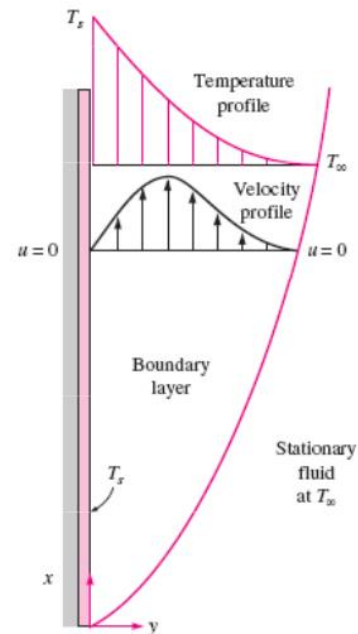
From the author's Engineering Collection, included in the ETSII4 module

This program calculates the Nusselt dimensionless number and the film coefficient (h) in a natural convection situation for the following three cases:

- Vertical plate or cylinder
- Horizontal plate
- Horizontal Cylinder or Sphere.

The program requires the Grashof (Gr) and Prandtl (Pr) numbers - or each of their constituent factors when they're not known to obtain the needed values. Then its product (i.e. the Raileigh number Ra) is used as a criteria for the different sections of the boundary layer conditions, as follows:

$$\begin{aligned} \text{Gr} &= [g \beta L^3 (T_p - T_{\text{inf}})] / \nu^2 \\ \text{Pr} &= \mu C_p / K_f \\ \text{Ra} &= \text{Gr} * \text{Pr} \end{aligned}$$



Case 1: $LowLimit < Ra < 1 E4$

Where the low limit being 0.1 for vertical plates/Cylinders, or 1 E-5 for horizontal Cylinder/Sphere. Here a fourth-degree polynomial approximation is used as follows:

1.a. Vertical Plate / Cylinder:

$$Nu = 0.161771563 + 0.127972027 Ra + 1.153845962 E-2 Ra^2 - 2.797201424 E-3 Ra^3 + 4.662002506 E-4$$

1.b. Horizontal Cylinder / Sphere:

$$Nu = 5.949883478 E-2 + 0.1274378392 Ra + 9.986887925 E-3 Ra^2 + 2.865190955 E-4 Ra^3 + 2.185315948 E-5 Ra^4$$

Case 2: $1 E4 < Ra < 1 E9$

Vertical Plate/Cylinder:	$Nu = 0.59 / Ra^{1/4}$
Horizontal Cylinder / Sphere:	$Nu = 0.525 / Ra^{1/4}$

Case 3: $1 E9 < Ra < 1 E12$

For all cases covered in the program: $Nu = 0.129 / Ra^{1/3}$

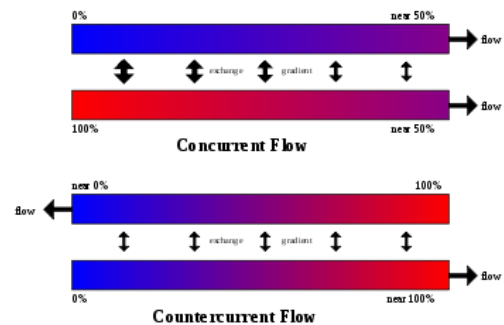
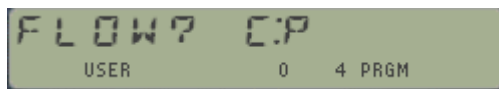
Finally the film coefficient is calculated using the definition expression as function of the thermal conductivity (K) and the characteristic dimension (length or diameter) of the body:

$$h = Nu K / L$$

Heat Exchangers Basic Equation. [HEATX]

From the Author's Engineering Collection, included in the PSYCHRO module.

This program calculates the transferred heat and exit temperatures of the fluids in a heat exchanger when all the other parameters are known, including the total area of exchange "A", and the global heat conduction coefficient "U". Both cases concurrent (parallel) and counter flow configurations are possible – the initial prompt will select the chosen configuration:



The input data parameters include the following:

- Mass flows for each fluid, m_1' and m_2'
- Specific heat capacity for both fluids, C_{p1} and C_{p2}
- Input temperature for cold fluid - $T_1(I)$
- Either input or output temperature for hot fluid; $T_2(I)$ or $T_2(O)$

Note: If only the product $\{A.U\}$ is known you can enter $U=1$ and the value of $U.A.$ at the respective prompts. Each time a new set of results will be obtained.

Let $k_{12} = m_1'.C_{p1} / m_2'.C_{p2}$. The equations used for the *Temperatures at a distance "x" from the inlet* and the *Total transferred heat $Q(L)$* are as follows:

1. Parallel (concurrent) flow.

$$T_1(x) = 1 / (1 + k_{12}) \{ T_2(I) + k_{12}.T_1(I) + [T_1(I) - T_2(I)].\exp[- U.A(x).(1+k_{12}) / m_1'.C_{p1}] \}$$

$$T_2(x) = T_2(I) - k_{12}[T_1(x) - T_1(I)]$$

$$Q(L) = m_1'.C_{p1}[T_1(I) - T_2(I)] / (1 + k_{12}).\{ \exp[-U.A(L).(1+k_{12}) / m_1'.C_{p1}] - 1 \}$$

2. Counter flow.

$$T_1(x) = 1 / (1 - k_{12}) \{ T_2(O) + k_{12}.T_1(I) + [T_1(I) - T_2(O)].\exp[-U.A(x).(1 - k_{12}) / m_1'.C_{p1}] \}$$

$$T_2(x) = T_2(O) - k_{12}[T_1(x) - T_1(I)]$$

$$Q(L) = m_1'.C_{p1} [T_1(I) - T_2(O)] / (1 - k_{12}).\{ \exp [-U.A(L).(1 - k_{12}) / m_1'.C_{p1}] - 1 \}$$

Example.

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

The results are: $Q(L) = 5.999,998790 \text{ kcal/min}$
 $T_1(O) = 39,999996^\circ\text{C}$; and $T_2(O) = 77.0744^\circ\text{C}$

Radiative View Factors. [FDD, FRR]

From the author's Engineering Collection, included in the ETSII4 module

This program obtains the view factors used in radiative calculations. The driver programs prompt for the geometrical dimensions of the shapes (radius, base, height, and separation distances) returning the solution after a short calculation time.

The formulas used are as follows:

1. From a disc of radius R1 to a coaxial parallel disc of radius R2 at separation H, with $r_1=R_1/H$ and $r_2=R_2/H$.

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

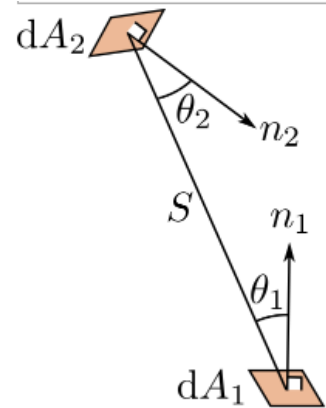
with $x = 1 + 1/r_1^2 + r_2^2/r_1^2$ and

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

2. Between parallel equal rectangular plates of size W1·W2 separated a distance H, with $x=W_1/H$ and $y=W_2/H$.

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

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



Examples.

Calculate the view factors between two parallel coaxial disks of radius $R_1 = 2.25$, and $R_2 = 1.75$ separated a distance $d = 3$. Do the same for two equal rectangular plates of dimensions 2.25×1.75 separated the same distance.

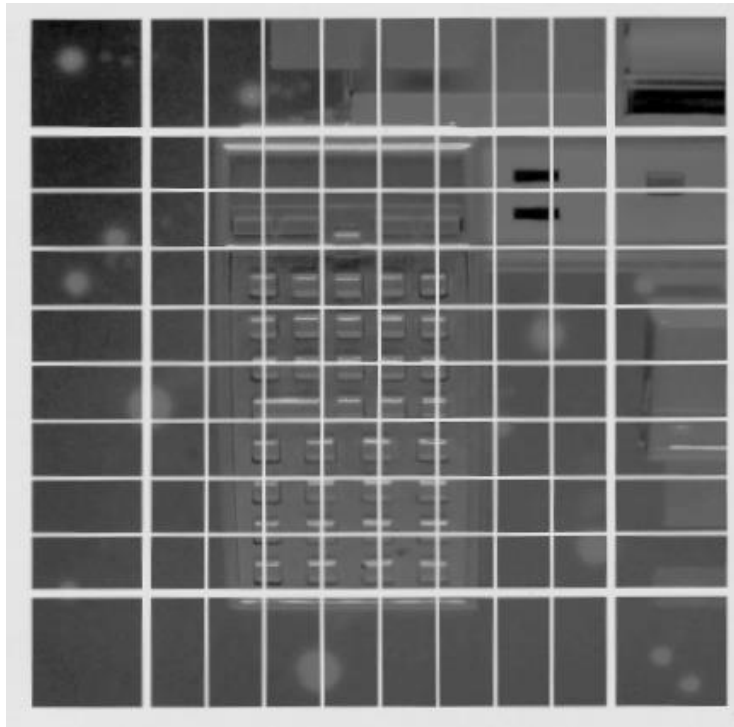
The results are shown below:

Coaxial Discs, $F_{12} = 0.1894$
 Rectangular plates $F_{12} = 0.1088$



Thermodynamics & Fluid Mechanics





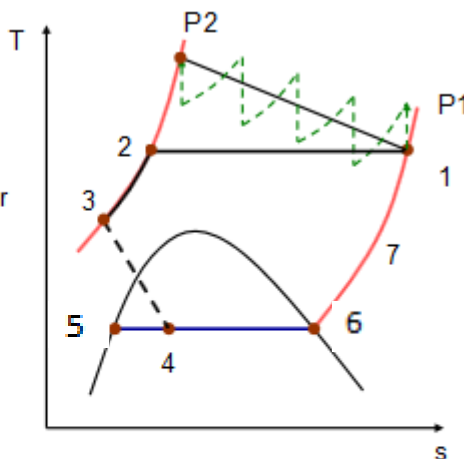
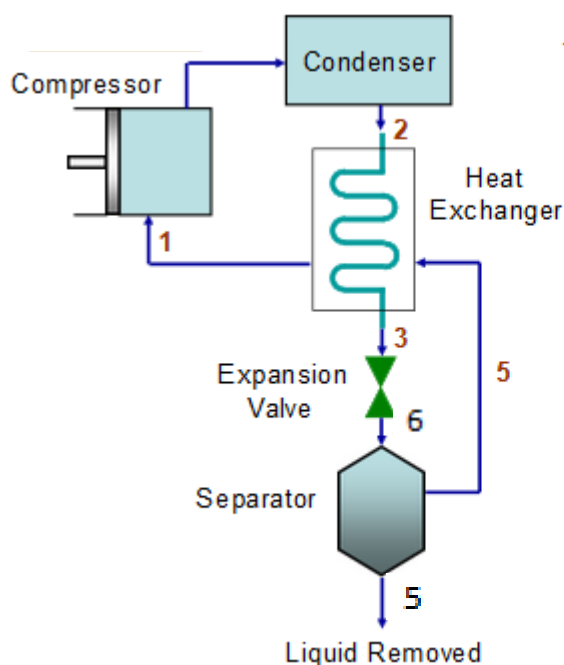
Gas Liquefaction Cycles. [LINDE, CLAUDE, HEYLND]

From the author's Engineering Collection, included in the ETSII3 module

This program calculates the enthalpy at all the significant stages of the most common liquefaction cycles: Linde, Heylandt or Claude. Program prompts for some input data such as the enthalpy of the saturated liquid and vapor and the entry conditions of the gas. In addition, the turbine isentropic efficiency is also required for the Heylandt and Claude cases. The program also calculates the liquefied fraction per circulating mol of gas.

The enthalpies at the points of known conditions must be obtained using the tables corresponding to the gas used in the cycle. The compressors used in the cycle are assumed to be isothermal - or at least that the final temperature is the same as the initial if an association of several compressors in series is used.

Linde Cycle.



The equations used are as follows:

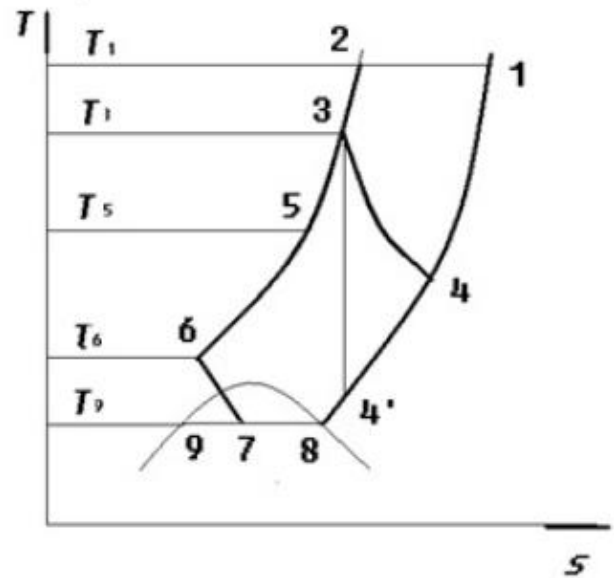
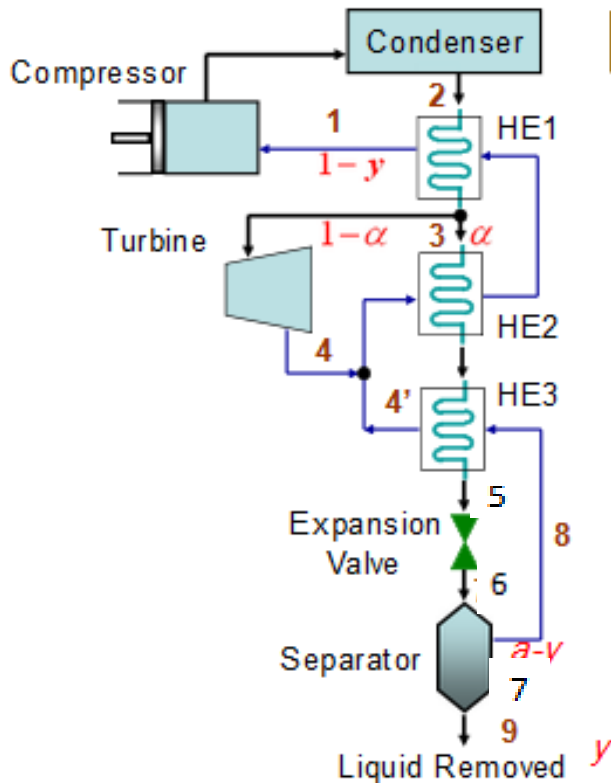
$$y = (h_2 - h_1) / (h_5 - h_1)$$

$$h_4 = y \cdot h_5 + (1-y) \cdot h_6$$

Example:

Calculate the liquid fraction extracted per mol in a Linde cycle with the following input conditions {h1, h2, h5, h6}.= The results are also given in the table.

H1	419.600	<u>Results:</u>	
H2	380.600	H3	183.5878
H5	0.000	H4	183.5878
H6	202.400	y	0.0929

Claude and Heylandt Cycles.

The equations used are as follows:

$$\begin{aligned} x \cdot h_4 &= y \cdot h_7 + (x-y) \cdot h_9'' \\ x \cdot h_6 &= y \cdot h_7 + (x-y) \cdot h_8 \\ (1-y) \cdot h_{10} &= h_3 + h_1 (1-y) - h_2 \end{aligned}$$

And liquid fraction extracted per mol: $y \cdot (h_1 - h_7) = (h_1 - h_2) + (1-x) \cdot h_3 - (1-x) \cdot h_9''$

Examples:

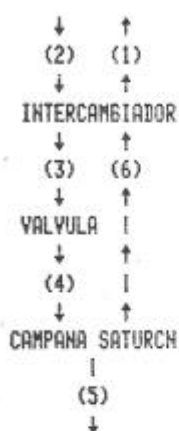
Calculate the liquid fraction extracted per mol in a Claude cycle with the following input conditions $\{h_1, h_2, h_5, h_6\}$. The fraction thru the turbine is $(1-\alpha) = 0.55$; and the isentropic efficiency of the turbine is $r = 0.7$. The results are also given in the table.

Input Data		Results:	
h_1	423.9000	y	0.2971
h_2	384.9000	h_3	77.0744
h_6	0.0000	h_4	67.9579
h_7	200.000	h_5	67.9579
h_8	174.140	h_8''	226.8300

More examples are shown in the next page – taken from a printout using the thermal printer. They include a sketch with the cycle components and (more importantly) the numbered points within the cycle with the convention used in the data entry prompts. The components are labeled in Spanish – a good opportunity to dust off your language skills ;-)

Campana Saturación = Vapor Dome; Intercambiador = Heat Exchanger

XEQ "LINDE"
C. LINDE



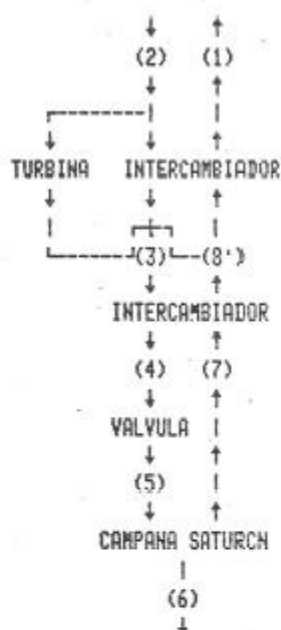
H1? 419,6000 RUN
H2? 380,6000 RUN
H5? .0000 RUN
H6? 202,4000 RUN

Y=0,0929

TABLA DE ENTALPIAS:

H1=419,6000
H2=380,6000
H3=183,5878
H4=183,5878
H5=0,0000
H6=202,4000

XEQ "HEYLANDT"
C. HEYLANDT



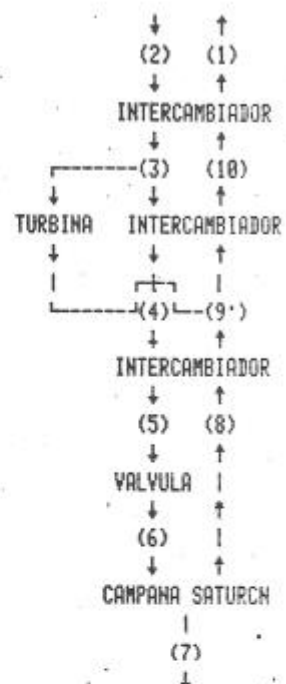
H1? 423,9000 RUN
H2? 384,9000 RUN
H6? .0000 RUN
H7? 200,0000 RUN
H8? 174,1400 RUN
X? .4500 RUN
RENDMT? .7500 RUN

Y=0,2971

TABLA DE ENTALPIAS:

H1=423,9000
H2=384,9000
H3=77,0744
H4=67,9579
H5=67,9579
H6=0,0000
H7=200,0000
H8'=226,8300

XEQ "CLAUDE"
C. CLAUDE



H1? 423,9000 RUN
H2? 413,5000 RUN
H7? .0000 RUN
H8? 200,0000 RUN
X? .2000 RUN
RENDMT? .7500 RUN
H3? 300,0000 RUN
H9? 183,4000 RUN

FRACCION LICUADA / MOL :
Y=0,1896

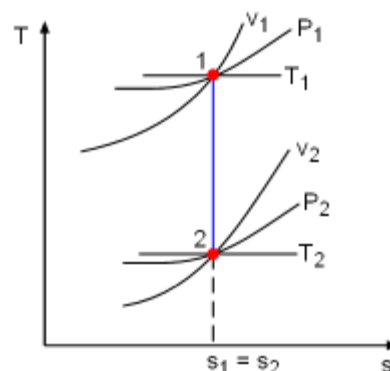
TABLA DE ENTALPIAS:

H1=423,9000
H2=413,5000
H3=300,0000
H4=11,0813
H5=10,4270
H6=10,4270
H7=0,0000
H8=200,0000
H9'=212,5500
H10=283,8504

Ideal processes of Perfect Gases with $C_p = C_p(T)$. [T2, P2, DS]

From the author's Engineering Collection, included in the ETSII3 module

For a perfect gas which specific heat at constant pressure (C_p) is a polynomial expression in the temperature (of any degree) - this program calculates the unknown T2, P2, ΔS final value after experiencing an ideal process of temperature change. The initial state is to be known, with {T1, P1} always known, and also either {T2, P2}, or {P2, ΔS }, or {T2, ΔS } depending on the case.



Let $C_p = \sum \{A_k T^k\}$; in [cal/mol.K] ; with $k = 1, 2, \dots n$.

The main expression used is the following:

$$\Delta S = A_0 \ln(T_2 / T_1) - R \ln(P_2 / P_1) + \sum (A_k / k) [T_2^k - T_1^k] ; k = 1, 2, \dots n$$

Whereby the final pressure is directly obtained as well; and the final temperature requires an iterative process using a root-finding algorithm (routine "SLV" within the Module).

$$P_2 = P_1 \cdot \exp \left\{ (1/R) [A_0 \ln(T_2 / T_1) - \Delta S + \sum (A_k / k) [T_2^k - T_1^k]] \right\} ; k = 1, 2, \dots n$$

Examples.

Characterize the complete final state of a perfect gas under ideal processes from $T_1 = 300^\circ\text{C}$ and $P_1 = 1 \text{ atm}$, with partial final data known shown in the table below; if its C_p is given by the polynomial expressions:

a) $C_p = 5.183 + 0.028 T - 0.000054 T^2$ [Cal/mol.K]

Case	T2 (deg)	P2 (atm)	ΔS (Cal/Mol.K)
XEQ "DS"	654.03 deg	50.00	DS = 5.2788
XEQ "P2"	654.03 deg	P2 = 50.0158	5.2788
XEQ "T2"	T2 = 654.0071	50.00	5.2788

b) $C_p = 5$ [Cal/mol.K]

Case	T2 (deg)	P2 (atm)	ΔS (Cal/Mol.K)
XEQ "DS"	654.03 deg	0.0954	DS = 8.5603
XEQ "P2"	654.03 deg	P2 = 0.0954	8.56
XEQ "T2"	T2 = 654.0578	0.0954	8.56

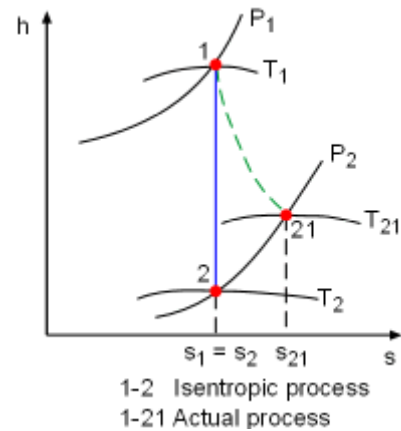
Note; This program uses the "Unit Management System", make sure you have the Unit Conversion module plugged into the calculator as well.

Non-isentropic expansion into Vapor Dome. [ENIVH]

From the author's Engineering Collection, included in the ETSII3 module

This short program calculates the final enthalpy in a non-isentropic expansion of a gas with final conditions inside of the vapor dome, like it's the case of steam turbines in power plants.

Obviously the isentropic efficiency of the turbine will be needed. Other required input data include the initial {P,T} conditions, as well as one of these two in the final stage. With these we'll obtain the enthalpy and entropy using the substance charts, which will be used by the program.



The results include the vapor quality at the exit of the turbine, as well as the corresponding enthalpy if the expansion was isentropic.

Let x the fraction of vapor in the final condition, 2l and 2v the points corresponding to the liquid and vapor ends of the vapor dome at the T₂ temperature.

The formulas used are as follows:

$$x = (S_1 - S_{2,liq}) / (S_{2,vap} - S_{2,liq})$$

$$H_2 = x H_{2,vap} + (1-x) H_{2,liq}$$

$$H_2'' = H_1 + \eta_T \{ H_{2,liq} + H_1 + [(S_1 - S_{2,liq}) / \Delta S_{2,vap}] \cdot \Delta H_{2,vap} \}$$

Example:

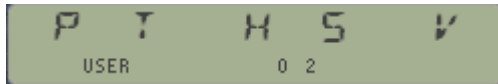
Calculate the final enthalpy in a non-isentropic expansion of a gas from initial conditions $H_1 = 3,240$ kJ/kg; $S_1 = 6.939$ kJ/kg.K into a final condition given by $H_{2,v} = 220.6867$ kJ/kg; $H_{2,liq} = 2.307.5094$ kJ/kg ; entropy of $S_{2,vap} = 7.25$ kJ/kg.K; $S_{2,liq} = 3.7564$. Use the value 0.75 for the turbine efficiency.

The results are:	$x = 0.91$ kg/mol;	vapor title
	$H_2 = 406.46$ kJ/kg; and	isentropic value
	$H_2'' = 1,114.84$ kJ/kg	non-isentropic result

Properties of Superheated & Saturated Steam. [by Michel Le Mero]

From the User's Program Library Europe #10341, included in the ETSII3 module

With a pair of properties being known {P,T} or {P,S}, or {P,H} – this program first determines if the point is in the dry or saturated region and outputs the superheat or the quality of steam. Then upon pressing the corresponding user's keys any of the unknown properties are calculated. The program uses British units internally but the unit conversion module is required – so that you can specify the input and output units with the UMS facility.



Superheated region. The subroutine “PT” uses the well-known formulation of Keenan and Keyes to derive H, S, and V. The higher order term has been omitted from the formulas for H and S. The Subroutines “PH” and “PS” use the following iterative procedure to compute T:

- Estimation of T and Cp, the specific heat
- Calculation of an approximate H or S
- Correction of estimated T as a function of Cp and H or S.
- The iteration stops when the correction factor for T is less than 1 degree F.
- All properties are then derived from the final P and T.

Equations and variables: with P expressed in absolute atm, and T in degrees Kelvin

$$V = 0.0160185 (4.55504 T/P + B)$$

$$H = F + 0.043557 (F0.P + B'(-B6 + B0(B2 - B3 + 2B7.B')))$$

$$S = 0.809691 \log(T) - 0.253801 \log(P) + a1.T - b1/T - 0.355579 - 0.0241983 \beta$$

where:

$$B = B0(1 + (B0.P/T^2)(B2 - B3 + (B0.P/T^2)(B4 - B5)B0.P))$$

$$B' = (1/2) B0(P/T)^2$$

$$B4 = 0.21828.T$$

$$B0 = 1.89 - B1;$$

$$B1 = (2641.62 / T). 10^{(80870/T^2)}$$

$$B2 = 82.546;$$

$$B3 = 162460 / T;$$

$$B5 = 126970 / T;$$

$$B6 = b0.B3 - 2.F0(B2 - B3);$$

$$B7 = 2F0(B4 - B5) - B0.B5;$$

$$F0 = 1.89 - B1(2 + 372420 T^2)$$

$$a1 = 1.8052 E-3;$$

$$b1 = 11.4276$$

$$F = 775.596 + 0.63296 T + 1.62467E-3 T^2 + 47.3635 \log(T)$$

$$\beta = (1/T)((B0 - F0).P + B'(B6 + B'.B0(B4 - B5 - 2B7)))$$

In the superheated region the program will yield accurate results for $S \geq 1.4$ BTU/lb.F

Saturated Region. The gas properties Hg, Sg and Vg are computed using “PT”, with P and T the saturation temperature, Tsat. The fluid properties Hf, Sf, and Vf are calculated as high order polynomial regressions of P. Knowing P and either H or S the steam quality is easily computed – for instance with H being known: $Q = (H - Hf)/(Hg - Hf)$. The remaining properties can then be calculated using Q. In the example below $S = Q.Sg + (1 - Q).Sf$

Polynomial regressions: Let $p = \log(P)$

In the saturated region, polynomial regressions have been made for $1024 \text{ psi} \geq P \geq 1 \text{ psi}$

$$T_{\text{sat}} = \sum \{t_k \cdot p^k\}; k=0..4 \quad - \quad S_f = \sum \{s_k \cdot p^k\}; k=0..6$$

$$H_f = \sum \{h_k \cdot p^k\}; k=0..5 \quad - \quad V_f = \sum \{v_k \cdot p^k\}; k=0..4$$

With coefficients as follows:

K	tk	sk	hk	vk
0	101.6904213	0.1325214898	69.8248945	1.612664461 E-2
1	22.99931254	0.04112950801	23.38590621	5.504415112 E-6
2	1.307138044	1,332706491 E-3	0.5953773184	7.900287375 E-5
3	-0.01038755447	1,055114953 E-4	0.2785143	-1.486233751 E-5
4	9.537213359 E-3	4.068809803 E-5	-0.03427505869	1.236908782 E-6
5		-4.357212264 E-6	2.468448779 E-3	
6		2.038138825 E-7		

All constants will be automatically loaded by the program the first time they're required for the calculations.

Example.

A steam turbine operates at the following conditions; Inlet: 650 psi, 790 deg F; exhaust: 2psi. Determine the inlet enthalpy and specific volume. Assuming a 10% pressure drop in the inlet valves, what is the available energy?

Executing "PT" with P = 650 psi and T = 790 F

```

WAIT...      "SELECT KEY:" ... "P T H S V"
XEQ "C"      =>  "UNITS H?", "BTU/LBM"
              =>  "H=1,349,5 BTU/LBM"

R/S          =>  "SELECT KEY:" ... "P T H S V"
XEQ "E"      =>  "UNITS V?", "FT3/LBM"
R/S          =>  "V=1,08 FT3/LBM"
```

Executing "PH" now with P = 0.9*650 = 580, and the same temperature:

```

WAIT...      "SELECT KEY:" ... "P T H S V"
XEQ "D"      =>  "UNITS S?", "BTU/LBM*K"
R/S          =>  "S=1.620 BTU/LBM*F"
```

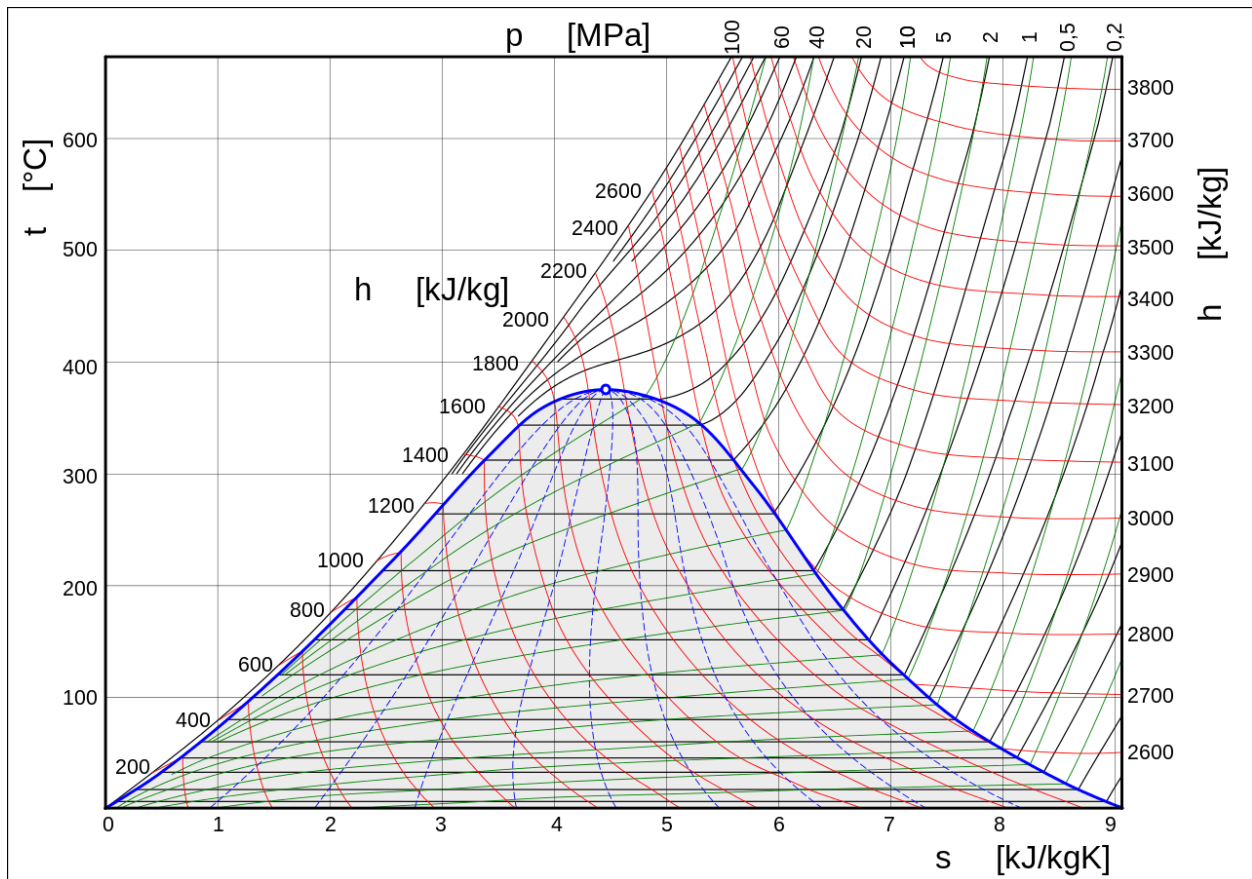
With that value of the entropy known, and the exhaust pressure P = 2 we can execute "PS" to obtain the exhaust enthalpy:

```

WAIT...      "SELECT KEY:" ... "P T H S V"
XEQ "C"      =>  "UNITS H?", "BTU/LBM"
              =>  "H=946.8 BTU/LBM"
```

Finally, the available energy is the difference between the inlet and exhaust enthalpy:

$$U = H_{\text{out}} - H_{\text{in}} = 1,349.5 - 946.8 = 402,72 \text{ BTU/lb}$$



Example (con't).

Suppose the turbine having 80% efficiency. What are the exhaust quality, specific volume and temperature?

Result: the exhaust available energy is now reduced to the 80% of the value obtained before, i.e.: $U' = 0.8$, $U = 322.176 \text{ BTU/lb}$

Subtracting it from the exhaust enthalpy obtained previously:

$$H_2' = 1,349,5 - 322.176 = 1.027,324 \text{ BTU/lbm}$$

Which can be used as input data for another iteration of "PH", using again $P_2 = 2 \text{ psi}$:

WAIT...

XEQ "E"

R/S

XEQ "B"

"Q = 0.927"

"SELECT KEY:"

...

"P T H S V"

"UNITS H?"

"V = 161,07 FT³/L"

"T=126,00 F"

Principle of Corresponding States. [MARTIN]

From the author's Engineering Collection, included in the ETSII3 module

This program evaluates the third of the {P,V,T} properties in the vapor-liquid region of a non-perfect gas when the other two are known. It uses the Principle of Corresponding States (PCS) with a modification of the equation of state proposed by Joseph Martin, expressed in reduced form and cubic in the volume; and given by the expression:

$$Pr = Tr / [Zc.Vr - B] - A / \{ Tr^n [Zc.Vr - B + 1/8]^2 \}$$

The Critical constants {Tc, Pc, Vc} are widely available in the technical reference data banks. Zc is the experimental compressibility factor, typically smaller than the theoretical one (Zt = Pc.Vc/R.Tc). The constants {A, B, n} are dimensionless ("n" is *not* the number of moles) and specific to each substance. They are obtained using semi-empirical methods, which description is beyond the scope of this manual. Approximate values for A, B can be taken as A = 27/64, and: B = 0.72.Zc - 0.152

The table below shows the values for a few gases that you can use to check the program.

Parameter	R-40 CH3Cl	R-50 CH4	R-170 C2H6	R-290 C3H8	R-764 SO2	R-744 CO2
Zc	0.268	0.291	0.27844	0.2701	0.268	0.274
A	0.421875	0.421875	0.421875	0.421875	0.421875	0.421875
B	0.0495	0.0575	0.05739	0.0511	0.051	0.0453
n	0.85	0.75	0.45	0.40	0.50	0.50
Tc (C)	143.1	-82.3	32.0908	96.85	157.19	30.978
Pc (kp/cm2)	68.0997	97.6104	50.3	43.4	80.2996	75.2245
Vc (cm3/g)	2.7027	6.200	4.7619	4.4248	1.9084	2.1359
PM (g/mol)	50.491	16.044	30.07	44.09	64.06	44.01
w (acentric factor)	0.156	0.011	0.105	0.152	0.2510	0.228

The programs work internally with the units reflected in the table but you can input and output the values in any other units. It comes without saying that these programs make extensive use of the Unit Management System (UMS), therefore the unit Conversion Module needs to be plugged into the calculator.

This program provides a subset of the functionality of the Thermodynamic Properties of Refrigerants, to be described in the next section. Here the three magnitudes are treated in an interchangeable solutions form – so you can enter with any two of them known to obtain the third.



- The specific volume is the largest (real) root of the cubic equation in V, obtained by the auxiliary routine "ROOT3", included in the module.
- Calculation of the temperature requires the root-finding routine "SLV", also included in the module (no additional dependencies).

Examples.

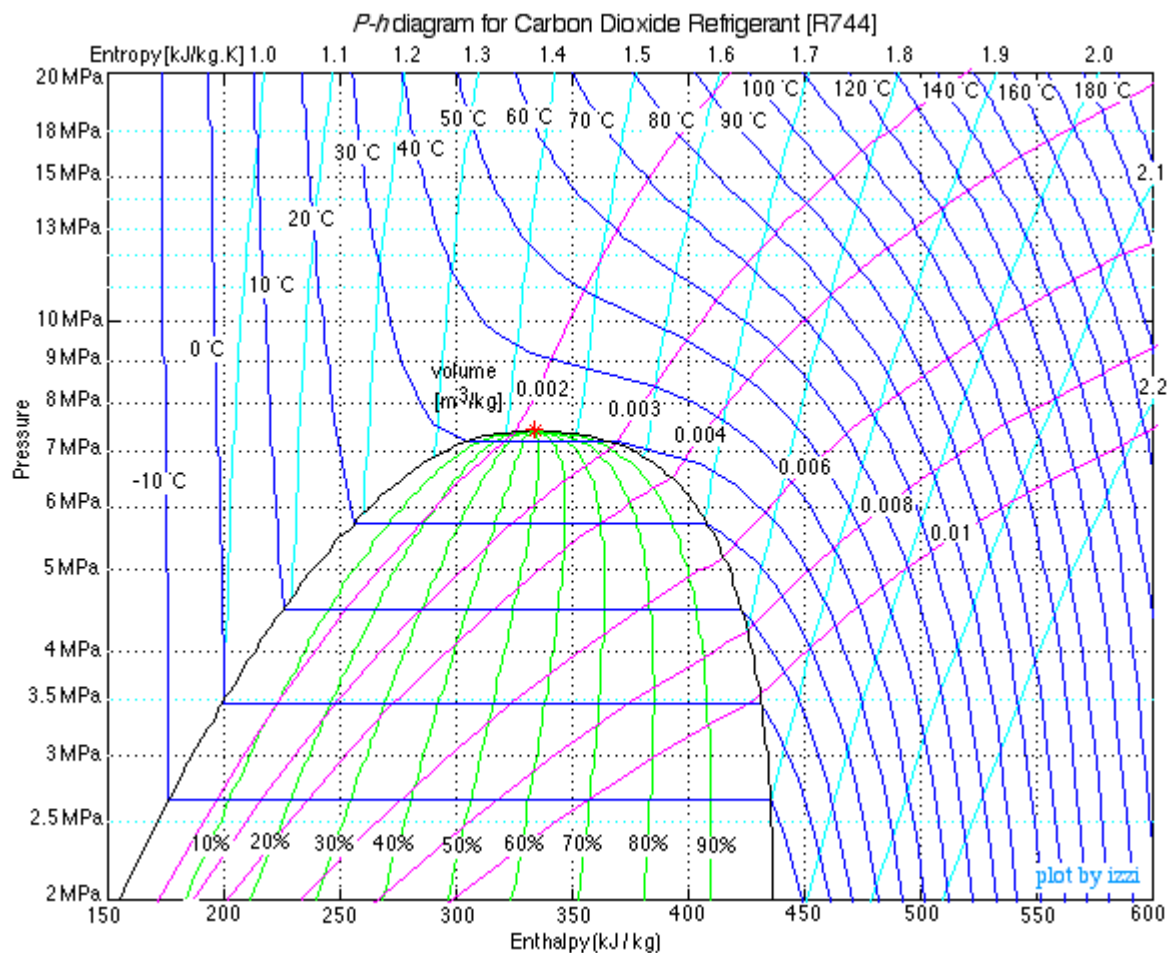
Calculate the specific volume of CO₂ in the saturated region knowing its Martin constants and critical value as per the table above. The initial conditions point data are $T = 0 \text{ deg C}$, and $P = 35.4861 \text{ kp/cm}^2$. Check the result obtained using it as data point for a reversed calculation of P (with same T) and T (with same P).

Result: $V_e = 0.0105 \text{ m}^3/\text{kg}$

Feeding this result as initial input:

$P(V_e; 125 \text{ C}) = 35,4861 \text{ kp/kg}$

$T(V_e; 50.9825 \text{ kp/cm}^2) = 2,0000\text{E-}7 \text{ deg C}$



Thermodynamic Properties of Refrigerants. [FREON, R12, R22, NH3]

From the author's Engineering Collection, included in the ETSII3 module

These programs provide a replacement for the different refrigerant chart sheets – using a semi-empirical approach that combines the Martin Equation of State, the Antoine's Equation, and polynomial expressions (in inverse 1/T) for the isobaric specific heat capacity of the liquid and the gas - valid within the application range, *defined as* $P < P_c$.

The Martin Equation of State uses *reduced magnitudes*, and it's cubic in the Volume:

$$Pr = Tr / [Zc.Vr - B] - A / \{ Tr^n [Zc.Vr - B + 1/8]^2 \}$$

The data entry process involves all the parameters required, as obtained by the semi-empirical method used. The table below lists the parameters for the main refrigerant gases. Note that the parameters A, B, and N are dimensionless ("n" is *not* the number of moles). However all coefficients for the specific heats have dimensions - as required by the inversed polynomial expression: $C_p = \sum a_k / T^k$; with C_p [Energy/Mass*Temperature]

The programs work internally with the units reflected in the table but you can input and output the values in any other units. It comes without saying that these programs make extensive use of the Unit Management System (UMS), therefore the unit Conversion Module needs to be plugged into the calculator.

Parameter	R-11 CFCI3	R-12 CF2CI2	R-13 CF3CI	R-22 CHCIF2	R-113 CCI2FCCI2	R-717 NH3
Zc	0.2766	0.2790	0.27739	0.267	0.2560	0.242004
A	0.421875	0.421875	0.421875	0.421875	0.421875	0.421875
B	0.05598	0.0578	0.0566	0.0488	0.0323	0.03
n	-1.1	0.900	0.60	1.00	0.75	-1.1
Tc (C)	197.99	112.00	28,7708	96.00	214.09	132.2808
Pc (kp/cm2)	44.6003	41.9613	39.460	50.300	34.80	115.0036
Vc (cm3/g)	1.8018	1.79168	1.7212	1.9041	1.7301	4.247
PM (g/mol)	137.38	120.90	104.47	86.50	187.39	17.032
w	0.188	0.1760	0.180	0.215	0.252	0.253
Ln(Pv) = x1 - x2/T ; with T in K and Pv in kp/cm2						
x1	10.84436	10.1760	9.96485	10.6316	11.13964	11.71516
x2	3209.743	2469.5656	1901.3441	2460.1029	3568.1032	2803.591
Cp,v = Σ {ak / T^k} ; k= 0,1,...4 ; with T in K and Cp in kcal/kg.K						
a0,v (kcal/kg.K)	0.1886	0.25659	0.30905	0.3485	0.25732	0.95239
a1,v	-8.0924	-49.1350	-70.3763	-102.14	-15.946	-2.92521E2
a2,v	-3.8118E3	5405.60	1.0052E4	1.6546E4	-1.6728E4	6.2791E4
a3,v	4.7327E5	-239860.0	-5.6219E5	-1.003E6	4.9196E6	-4.725E6
a4,v	0.0000	0.0000	0.0000	0.0000	-4.1739E8	0.0000
Cp,l = Σ {ak / T^k} ; k= 0,1,...4						
a0,l	0.56642	9.9361	11.190	14.4110	4.5917	0.24694
a1,l	-2.3715E2	-9.445E3	-8.4336E3	-1.3625E4	-5.058E3	-2.3079E4
a2,l	5.2605E4	3.4338E6	2.4332E6	4.9211E6	2.2046E6	8.4508E6
a3,l	-4.0107E6	-5.5272E8	-3.1189E8	-7.8839E8	-4.2602E8	-1.3705E9
a4,l	0.0000	3.3182E10	1.4964E10	4.7189E10	3.0626E10	8.2854E10

Equations and memory register map:-

Let $Pr = P/P_c$, $Tr = T/T_c$, and $Vr = V/V_c$ the reduced pressure, temperature and volume over the corresponding critical values. Let $To = 273$ K, and $Po = Pv(273)$.

The pressure is directly calculated as:

$$Pr = Tr / [Z_c.Vr - B] - A / \{ Tr^n [Z_c.Vr - B + 1/8]^2 \}$$

The specific volume is obtained as root of the third degree equation given by:

$a_3.V^3 + a_2.V^2 + a_1.V + a_0 = 0$; where the coefficients are defined as:

$$\left\{ \begin{array}{l} a_0 = A.B + (1/8 - B)^2 [(Tr)^{n+1} + B.Pr.(Tr)^n] \\ a_1 = (Z_c/V_c). \{ A + Pr.(Tr)^n [3B^2 - B/2 + 1/64] - (1/8 - B)^2 (Tr)^{n+1} \} \\ a_2 = (Z_c/V_c)^2 . [(Tr)^n Pr.(1/4 - 3B) - (Tr)^{n+1}] \\ a_3 = (Z_c/V_c)^3 Pr (Tr)^n \end{array} \right.$$

The density of the saturated liquid is calculated by the formula:

$V_{s,l} = V_{sc}. V_g (1-w.\Gamma)$; with the following auxiliary definitions:

$$V_{sc} = (R.T_c/PM.P_c).(0.292 - 0.0967 w)$$

$$\Gamma = 0.29607 - 0.09045 Tr - 0.048442 Tr^2$$

And V_g depends on the actual value of Tr , as follows:

- if: $Tr < 0.2$ or $Tr > 1 \Rightarrow$ *not in liquid phase*
- if: $0.2 \leq Tr < 0.8$
 $V_g = 0.33593(1-Tr) + 1.51941 Tr^2 - 2.02512 Tr^4 + 1.11422 Tr^4$
- if: $0.8 \leq Tr \leq 1$
 $V_g = 1 + 1.3 \sqrt{1-Tr} . \log(1-Tr) - 0.50879(1-Tr) - 0.91534(1-Tr)^2$

Enthalpy of Vapor: Defined as: $H(T,P) = H^o(T,P) + [\Delta^o H(T,P) - \Delta^o H(T_o,P_o)]$

where Δ indicates *discrepancy from the perfect gas*, denoted itself as “o”.

$$H(T,P) = U(T,P) - RT (Z-1) = U(T,P) - (P.V - R.T)$$

This is calculated using the *Helmholtz Free Energy as auxiliary magnitude* instead of the internal energy, defined as: $F = U - T.S \Rightarrow U = F + TS$

$$F^o(T,P) = R.T.Ln(Z) + ITG\{(P - R.T/V).dV\};$$

$$\Delta H^o(T,P) = [\Delta F^o(T,P) + T.\Delta S^o(T,P)] - \Delta Q(T,P); \text{ with: } Q = P.V - R.T/PM$$

hence, ***moving along the isobaric*** and using a reference 100 cal/kg at 0 °C (the “dead state”):

$$H^o(T,P) = H^o(T_o) + ITG\{C_{pv}.dT\}; \text{ between } \{T_o \text{ and } T\};$$

$$H(T,P) = \{100 + H^o(T_o) + ITG[C_{pv}]\} + [(F-F_o) - (Q-Q_o) + T.(S-S_o)]$$

The integral of the specific heat is easily obtained by direct integration:

$$ITG\{C_{pv}\} = a_0(T-T_o) + a_1.Ln(T/T_o) - a_2(1/T - 1/T_o) - a_3(1/T^2 - 1/T_o^2)/2 - a_4(1/T^3 - 1/T_o^3)/3$$

Entropy of Vapor: Defined as: $S(T,P) = S^o(T,P) + [\Delta^o S(T,P) - \Delta^o S(T_o,P_o)]$;

Let $T_o = 273 \text{ }^\circ\text{K}$, and $P_o = P_v(273) = \exp(x_1 + x_2/T_o)$

$S^o(T,P) = S^o(T_o,P_o) - R \cdot \ln(P/P_o) + \text{ITG} \{ (C_{pv}/T) \cdot dT \}$; between $\{T_o \text{ and } T\}$

hence, using an origin reference of 1 Cal/kg.K at $0 \text{ }^\circ\text{C}$ (a.k.a the “dead state”):

$$S(T,P) = \{1 + H_v^o(T_o)/T_o + \text{ITG}[C_p^v/T]\} - (R/PM) \cdot \ln(P/P_o) + [\Delta S(T,P) - \Delta S(T_o,P_o)]$$

The integral of the specific heat over the temperature is easily obtained by direct integration of the inversed-polynomial expression:

$$\text{ITG}\{C_{pv}/T\} = a_0 \cdot \ln(T/T_o) - a_1(1/T - 1/T_o) - a_2(1/T^2 - 1/T_o^2) / 2 - a_3(1/T^3 - 1/T_o^3) / 3 - a_4(1/T^4 - 1/T_o^4) / 4$$

The vaporization enthalpy is obtained using the Pitzer correlation shown below (per kg), where w is the acentric factor of the substance:

$$H_v^o(T_o) = (R \cdot T_c/PM) \{ 10.95w \cdot [(1 - T_o/T_c)^{0.456}] + 7.08 \cdot (1 - T_o/T_c)^{0.354} \}$$

The discrepancy of Entropy and Helmholtz's Free Energy are calculated with the expressions shown below, with: $Z^o = P \cdot V/R \cdot T$; and: $Z_c^o = P_c \cdot V_c/R \cdot T_c$; in [mol]

$$\Delta F^o(T,P) = R \cdot T \cdot \ln(Z) + \text{ITG}\{(P - R \cdot T/V) \cdot dV\} = (R \cdot T/PM) \cdot \ln(PM \cdot Z^o) + P_c \cdot V_c \{ A / [Z_c \cdot T_r^n \cdot (Z_c \cdot V_r + 1/8 - B)] - T_r \cdot [\ln(V_r)/PM \cdot Z_c^o - \ln(Z_c \cdot V_r - B)/Z_c] \}$$

$$\Delta S^o(T,P) = \partial \Delta F^o / \partial T = (R/PM) \{ (n+1) + n \cdot A \cdot (PM \cdot Z_c^o / Z_c) / [Z_c \cdot V_r + 1/8 - B] \cdot T_r^{(n+1)} - \ln(PM \cdot Z^o) + \ln[V_r / (V_r - B/Z_c)^{(PM \cdot Z_c^o / Z_c)}] \}$$

Finally, the registers map is shown below.1

R00	pointer	R18	w (acentric fac.)	R36	ΔS_o
R01	units-1	R19	x1 Antoine	R37	ΔS
R02	units-2	R20	x2 Antoine	R38	ΔF_o
R03	Z_c	R21	$C_p(0)-V$	R39	ΔF
R04	A	R22	$C_p(1)-V$	R40	T
R05	B	R23	$C_p(2)-V$	R41	P
R06	n	R24	$C_p(3)-V$	R42	ICPV or ICPL
R07	T_c	R25	$C_p(4)-V$	R43	ICPVT or ICPLT
R08	P_c	R26	$C_p(0)-L$	R44	$\Delta F_o + Q_o + T_o \cdot \Delta S_o$
R09	V_c	R27	$C_p(1)-L$	R45	$\Delta F + Q + T \cdot \Delta S$
R10	work V	R28	$C_p(2)-L$	R46	100+IHVo+ ICPV+[$\Delta F + Q + T \cdot \Delta S$] - [$\Delta F_o + Q_o + T_o \cdot \Delta S_o$]
R11	work P	R29	$C_p(3)-L$	R47	1+IHVo/ T_o - DSO+ICPVT T_o +DS}- R/PM. $\ln(P/P_o)$
R12	work T	R30	$C_p(4)-L$		
R13	PM	R31	P_o		
R14	scratch	R32	V_o		
R15	scratch	R33	V		
R16	scratch	R34	IHV o		
R17	T(Pv)]	R35	IHV		

Example1.

Calculate the Freon-12 properties for the following (P,T) conditions:

T1 = -25 °C, and P1 = 1.2616 kp/cm2;

T2 = 25 °C, P2 = 0.2 atm; and

T3 = 0 °C, P3 = 4.3135 kp.cm2

The program execution is shown below. Note that each case represents a point in different areas of the diagram: saturated vapor (dome), Vapor, and Liquid. The results in the vapor dome include both the saturated vapor and liquid phases.

Note. You can type "R12" at the initial prompt to get all constants loaded automatically.

<pre> FREON-12 Zc=0,2790 A=0,4219 B=0,0578 N=0,9000 Tc=385,1592 K Pc=4115000,000 PA Vc=0,0287 FT3/LBM PM=120,9000 G/MOLE W=0,1760 -ECC. ANTOINE X1=10,1760 X2=2,469,5656 -CP VAPOR a0=0,2566 a1=-49,1350 a2=5,405,6000 a3=-239,860,0000 a4=0,0000 -CP LIQUIDO a0=9,9361 a1=-9,445,0000 a2=3,433,800,000 a3=-552,720,000,0 a4=3,3182E10 </pre>	<pre> T=-25,0000 C P=1,2616 KGF/CM2 -CAMPANA- RESULTADOS VESPV=0,1315 M3/KG HV=132,684,2316 CAL/KG SV=1,133,1725 CAL/KG*K DENSL=1,467,5406 KG/M3 HL=93,940,8542 CAL/KG SL=991,3216 CAL/KG*K </pre>	<pre> T=25,0000 C P=0,2000 ATM -VAPOR- RESULTADOS VESPV=1,0054 M3/KG HV=139,692,2963 CAL/KG SV=1,187,9291 CAL/KG*K T=0,0000 C P=4,3135 KGF/CM2 -LIQUIDO- RESULTADOS DENSL=1,359,7224 KG/M3 HL=104,883,7763 CAL/KG SL=1,012,6644 CAL/KG*K </pre>
---	--	--

Example2.

Calculate the Freon-22 properties for the following (P,T) conditions:

T = -40 deg C, P1 = 1.0760 kp/cm2; P2 = 0.5 atm ; and P3 = 2 atm

The program execution is shown below. Note that each case represents a point in different areas of the diagram: saturated vapor (dome), Vapor, and Liquid. The results in the vapor dome include both the saturated vapor and liquid phases.

Note. You can type "R22" at the initial prompt to get all constants loaded automatically.

<pre> FREON-22 Zc=0,2670 A=0,4219 B=0,0488 N=1,0000 Tc=96,0000 C Pc=50,3000 KGF/CM2 Vc=0,0305 FT3/LBM PM=86,5000 G/MOLE W=0,2150 -ECC. ANTOINE X1=10,6316 X2=2,460,1029 -CP VAPOR a0=0,3485 a1=-102,1400 a2=16,546,0000 a3=-1,003,000,000 a4=0,0000 -CP LIQUIDO a0=14,4110 a1=-13,625,0000 a2=4,921,100,000 a3=-788,390,000,0 a4=4,7189E10 </pre>	<pre> T=-40,0000 C P=1,0760 KGF/CM2 -CAMPANA- RESULTADOS VESPV=0,2031 M3/KG HV=142,750,9776 CAL/KG SV=1,188,5211 CAL/KG*K DENSL=1,356,2689 KG/M3 HL=87,969,0278 CAL/KG SL=987,9965 CAL/KG*K </pre>	<pre> T=-40,0000 C P=0,5000 ATM -VAPOR- RESULTADOS VESPV=0,4312 M3/KG HV=142,837,4507 CAL/KG SV=1,205,1779 CAL/KG*K T=-40,0000 C P=2,0000 ATM -LIQUIDO- RESULTADOS DENSL=1,313,6531 KG/M3 HL=96,398,2331 CAL/KG SL=1,007,4991 CAL/KG*K </pre>
---	--	--

Psychrometric Properties of humid Air. [by Alfredo Quijano]

From the User's Program Library Europe, included in the ETSII3 module

This program calculates the psychrometric properties of the air, replacing the use of the substance charts with faster and more accurate results. The input data *can be one of the following three options*:

- Dry-bulb temperature (T_s) and relative humidity (f)
- Dry-bulb temperature (T_s) and specific humidity (w)
- Dry-bulb (T_s) and wet-bulb (T_h) temperatures.

The equations used are as follows:

Vapor pressure (Pa):	$P_v = \Phi P_{vs}$, with Φ the relative humidity
Saturated vapor pressure (Pa):	$\log P_{vs} = [2.7858 + 7.5.T_s/(239.3 + T_s)]$
Specific humidity (per-unit):	$w = 0.622.P_v / (101300 - P_v)$
Dew point temperature (C):	$T_r = 237.3.k / (7.5 - k)$, with: $k (237.3 + T_s) = 7.5.T_s + (237.3 + T_s).log(\Phi)$
Specific enthalpy (kCal/Kg dry):	$H = 0.24.T_s + W.(0.44.T_s + 597)$
Specific Volume (m ³ /kg dry air):	$V_e = 462(T_s + 237.15) (0.622 + w) / 101300$
Wet-bulb temperature (C):	$T_h = T_s - 597.(w_{sh} - w) / (0.44w + 0.24)$; with:
Wsh: w for saturated air	$w_{sh} = 0.622.P_{sh} / (101300 - P_{sh})$; and:
Psh: P _v saturated air at T _h	$\log(P_{sh}) = 2.7858 + 7.5.T_h / (237.3 + T_h)$

With specific humidity known (w), the relative humidity is given by the expression:

$$f = 101300.w / P_{vs}(0.622 + w)$$

With wet-bulb temp known (T_h) the specific humidity is obtained with the formula:

$$w = [597.w_{sh} + 0.24(T_h - T_s)] / [597 - 0.44 (T_h - T_s)]$$

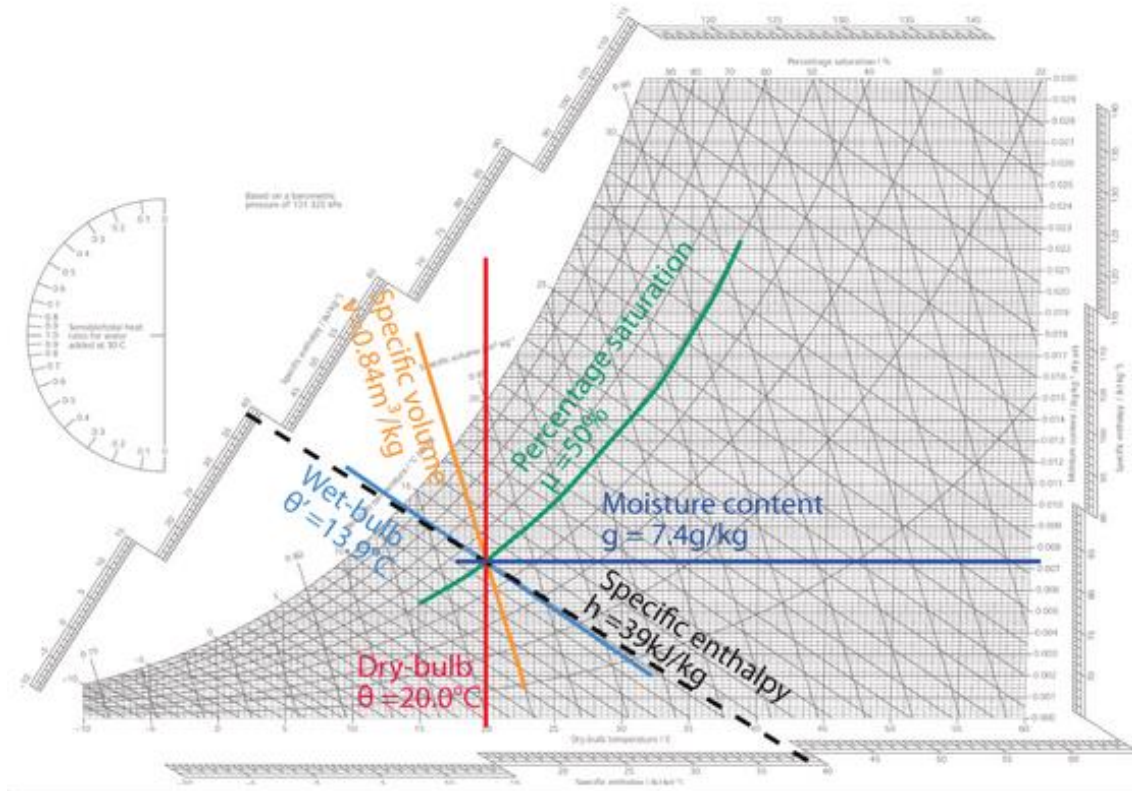
Examples.

Calculate the unknown factors for the three cases shown below.

- $T_s = 32$ deg C and $\Phi = 0.68$.
- $T_s = 35$ deg C and $w = 28.89$ E-3 kg/kg,
- $T_s = 32$ deg C and $T_h = 30$ deg C

The results are summarized in the table below:

Parameter	Case a)	Case b)	Case c)
T_s (C)	32	35	32
T_h (C)	26.95	31.79	30
Φ	0.68	799.9 E-3	865.8 E-3
w	20.50 E-3	28.89 E-3	26.34 E-3
T_r (C)	25.34	31.02	29.47
Hh(kCal/Kg)	20.21	26.09	23.78
V_e (m ³ /kg)	894.2 E-3	914.7 E-3	902.3 E-3



Notes on Data Entry and Output.

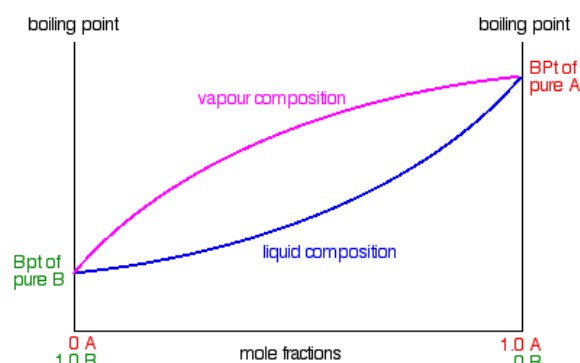
The program will ask for the input data in a serial sequence of prompts. You must enter the value for T_s in all cases, and then either the known value or simply [R/S] to skip the unknowns. If no values are introduced (i.e. you pressed [R/S] to all prompts) then the program will show the last calculated results. The data output will always show all the seven result values for the case – including the known inputs.

The calculation of the wet-bulb temperature T_h is done using the root-finding routine “SLV” also included in the module – thus no additional dependencies are introduced.

Temperature-Composition for binary mixtures. [WILSON, VANLAAR]

From the author's Engineering Collection, included in the ETSII3 module

This program obtains tabular representation of the composition-boiling point temperature diagram of a binary mixture in non-ideal conditions (where Raoult's law won't apply), under constant pressure conditions. The Antoine constants for each component must be known. Two models are available, using either Van-Laar or Wilson dissolution constants and equations. The program also calculates the vapor-liquid composition diagram. The tabulations can also be plotted as graphic curves on the thermal printer if available.



The expressions used for the calculations are shown below. Note that in both cases the solution requires *solving the equation for the value of the temperature "T" for each value of the liquid fractions (x1,x2) of both components.*

Let and $\{A_1, B_1, C_1\}$ and $\{A_2, B_2, C_2\}$ the Antoine constants for each component as per the corresponding sub-index. The main equations are written as follows:

$$\begin{aligned} x_1 \exp [Z_1] &= P - x_2 \exp [Z_2] && \text{; where } x_1 + x_2 = 1 \\ y_1 &= (x_1/P) \exp [Z_1] && \text{; molar fraction of vapor, : } y_1 = y_1(T, x_1) \end{aligned}$$

Let $\{\alpha, \beta\}$ the Van-Laar constants for the dissolution; then we have:

$$\begin{aligned} Z_1 &= A_1 - B_1/(T+C_1) + \alpha / (1 + \alpha \cdot x_1/\beta \cdot x_2)^2; \text{ and} \\ Z_2 &= A_2 - B_2/(T+C_2) + \alpha / (1 + \alpha \cdot x_1/\beta \cdot x_2)^2 \end{aligned}$$

Let $\{G_{12}, G_{21}\}$ the Wilson constants for the dissolution; then we have:

$$\begin{aligned} Z_1 &= A_1 - B_1/(T+C_1) - \ln(x_1+G_{12} \cdot x_2) + x_2 [(G_{12}/(x_1+G_{12} \cdot x_2) - G_{21}/(x_2+G_{21} \cdot x_1))] \\ Z_2 &= A_2 - B_2/(T+C_2) - \ln(x_2+G_{12} \cdot x_1) + x_1 [(G_{12}/(x_1+G_{12} \cdot x_2) - G_{21}/(x_2+G_{21} \cdot x_1))] \end{aligned}$$

The pressure remains constant. The program assumes an atmospheric pressure of 760 mm Hg. This can be changed by modifying the value in program lines .151, .300, and .366

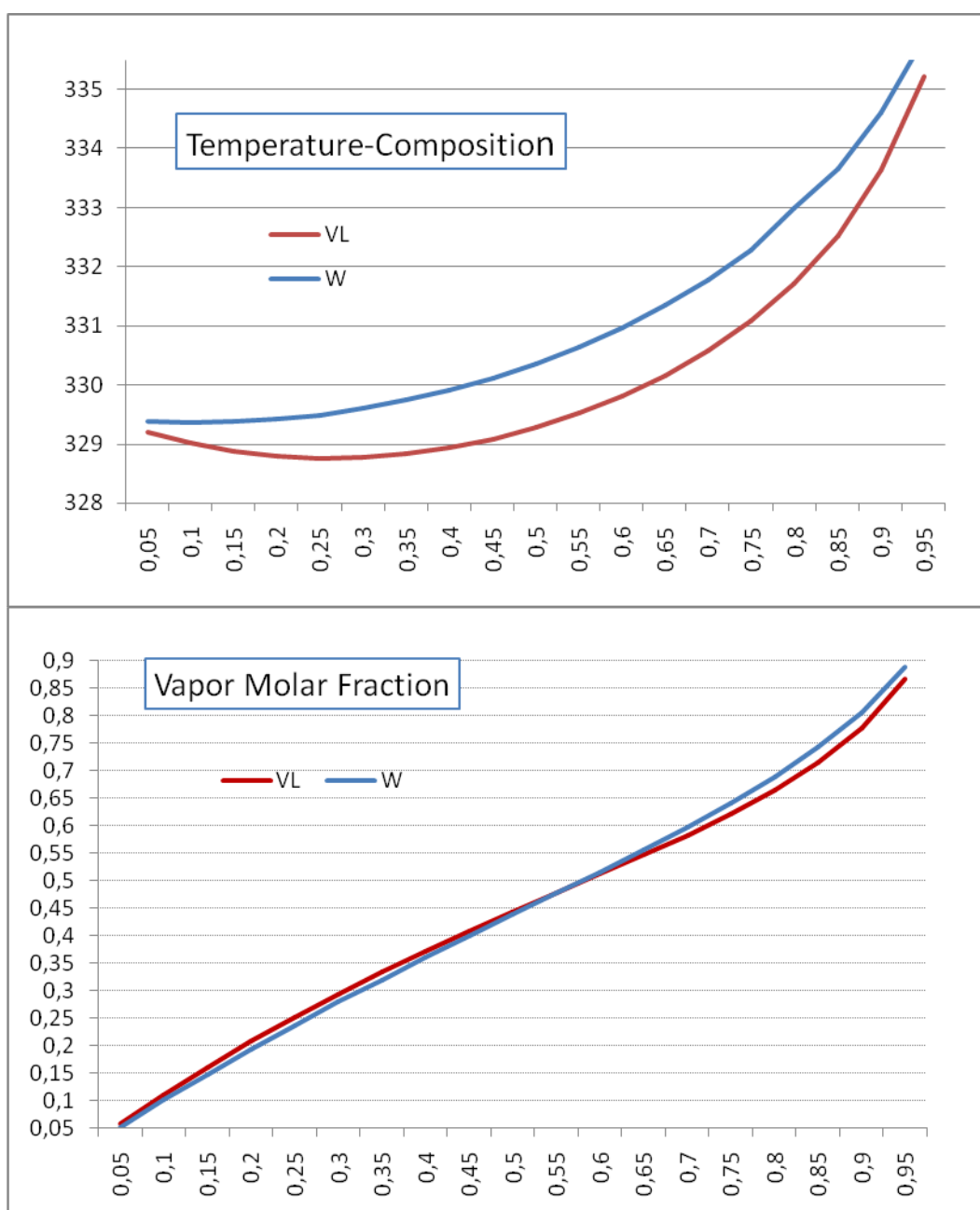
Example.

Tabulate and represent the "Temperature-Composition" and "vapor-liquid" diagrams for a non-ideal mixture of methanol and acetone, with the following data known:

Methanol	Acetone	Disso - Van-Laar	Disso -Wilson
A1 = 18.5875	A1 = 16.6513	$\alpha = 0.5076$	G12 = 1.2847
B1 = 3,626.55	B1 = 2,940.46	$\beta = 0.962536$	G21 = 0.3661
C1 = -34.29	C1 = -35.93		

The results are shown in the tables below, as well as represented in the charts:

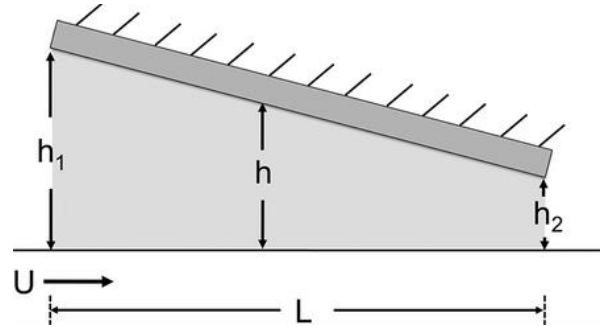
Van-Laar			Wilson		Van-Laar			Wilson	
x1	y1	T	y1	T	x1	y1	T	y1	T
.05	0,0574	329,1995	0,0514	329,3898	.55	0,4771	329,5147	0,4761	330,6398
.10	0,1107	329,0104	0,1004	329,366	.60	0,5115	329,8039	0,5151	330,9648
.15	0,1606	328,8766	0,1475	329,3758	.65	0,5463	330,1511	0,5548	331,3404
.20	0,2074	328,7946	0,1927	329,4184	.7	0,5823	330,5686	0,596	331,7767
.25	0,2515	328,7616	0,2362	329,4931	.75	0,6208	331,0765	0,6398	332,2893
.30	0,2932	328,7754	0,2784	329,5998	.80	0,6637	331,7091	0,6876	332,982
.35	0,3328	328,8343	0,3194	329,7387	.85	0,7143	332,5254	0,7418	333,6535
.40	0,3707	328,9374	0,3594	329,9105	.90	0,778	333,6293	0,8063	334,6074
.45	0,4072	329,0845	0,3987	330,1165	.95	0,8656	335,2137	0,8879	335,8748
.50	0,4425	329,2762	0,4375	330,3585	1				



Hydrodynamic Film lubrication. [MICHELL]

From the author's Engineering Collection, included in the ETSII4 module

This program calculates the required flow Q and outer edge thickness (h_1) on a film lubrication inclined pad with a perpendicular load F and moving at a uniform speed U_0 . The geometric data required may be either the slope angle (α) or the thickness of the oil film at the lower end (h_2).



The formula for the radial force is a non-explicit equation on h_1 , the thickness on the higher end of the fluid film. This can be resolved with a root-finder routine like SLV, also included in the module. The expression is given below, where μ_0 is the fluid viscosity:

$$Fr = [2 \mu_0 U_0 L / (h_1 - h_2)] [2 \ln (h_1/h_2) - 3 (h_1 - h_2)/(h_1 + h_2)]$$

And the flow required:

$$Q = U_0 h_1 h_2 / (h_1 + h_2)$$

Examples.

Calculate Q and h_1 for a Michell pad moving at 15 m/s and bearing a perpendicular load of 200,000 N. The geometric parameters of the pad are 0.2 m deep x 0.4 m long, and the taper slope is 0.0573 deg. The fluid viscosity is $\mu = 0.01 \text{ N/s m}^2$

The solutions are shown below:

Geometry

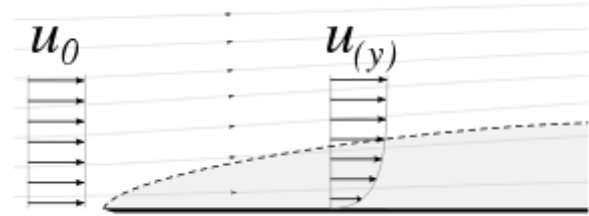
$h_1 = 99,327\text{E-}6$
 $h_2 = 59,327\text{E-}6$
 $L = 0,4000$
 $\text{SLOPE}(\angle) = 0,0001$

Work Data:

$\mu = 0,0100$
 $U_0 = 15,0000$
 $F = 1.000.000,000$
 $FR = 823,0323$
 $Q = 0,0006$

Stokes' First Problem. [P1STOKE]*From the author's Engineering Collection, included in the ETSII4 module.*

This program calculates the velocity at a point placed at a distance Y from the bottom and an instant t in an unsteady viscous boundary layer flow. The bottom is suddenly imposed at $t=0$ a constant velocity U_0 and the fluid has a kinematic viscosity ν . Vertical distances (y) are measured from the bottom ($y=0$) up.



The expression for the instant velocity at a distance y can be related to the cumulative probability function of a normal distribution as follows:

$$U(y,t) = 2 U_0 [1 - F(y / \sqrt{2 \nu t})]$$

Example:

for $U_0 = 1 \text{ m/s}$, $\nu = 10 \text{ m}^2/\text{s}$; $Y = 0.5 \text{ m}$ and $t = 1 \text{ s}$

the result is: $U(y,t) = 0.8744 \text{ m/s}$

The original version of this program used a polynomial approximation to calculate F , with an accuracy limited to 4 to 6 decimal places, depending on the value of the argument. A modern version based on the ERF implementation on the SandMath brings that to at least 8 decimal places and a much faster execution – thanks to the MCODE and the improved algorithm used.

$$U(y,t) = U_0 [1 - \text{erf} \{ [y / 2 \sqrt{\nu t}] \}]$$

Below you can see the program listing using the new approach. Note that R00-R03 are used by ERF:

01 LBL "P1STOKE"	11 "T=?"	21 +
02 "U0=?"	12 PROMPT	22 RCL 04
03 PROMPT	13 RCL 05	23 *
04 STO 04	14 *	24 "U="
05 "NU=?"	15 ST+ X	25 ARCL X
06 PROMPT	16 SQRT	26 PROMPT
07 STO 05	17 /	27 GTO 00
08 LBL 00	18 ERF	28 END
09 "Y=?"	19 CHS	
10 PROMPT	20 1	

Colebrook-White Equation. [MOODY, BROOK]

From the author's Engineering Collection, included in the ETSII4 module.

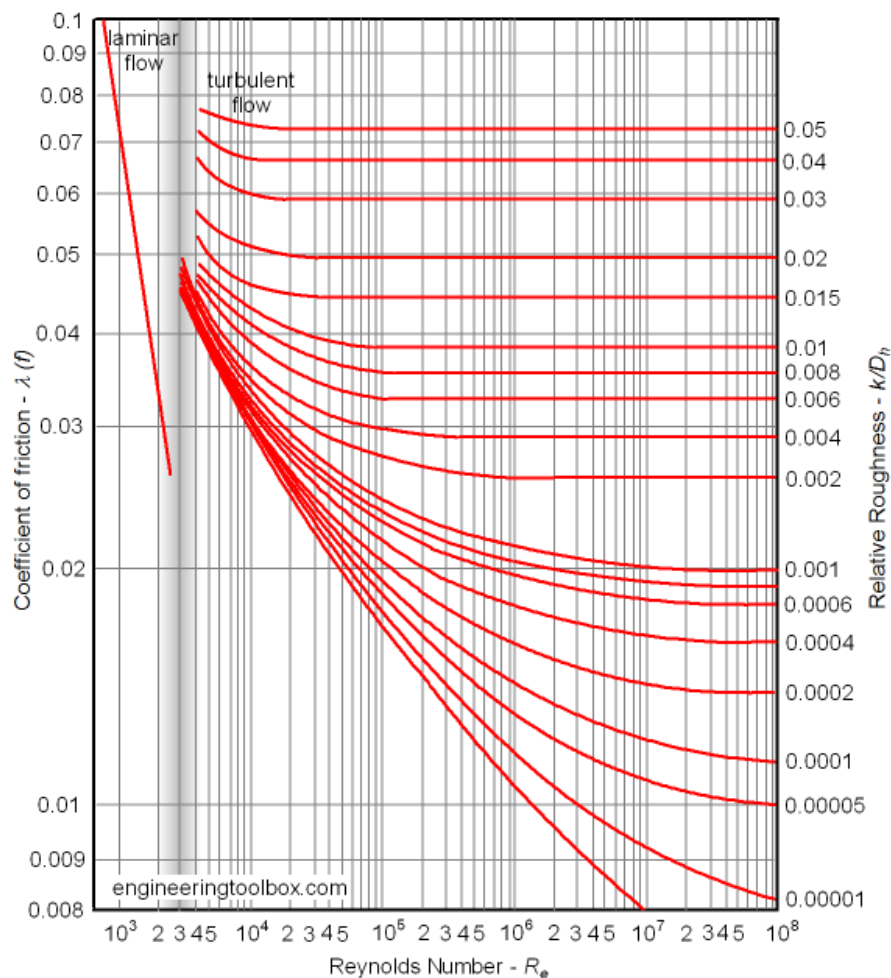
This program solves the Colebrook equation using an iterative approach. The input data may be either the relative roughness and Reynolds number directly (program BROOK); or indirectly based on the flow conditions: mass flow, inside diameter, cinematic viscosity, and absolute roughness (program MOODY). The result is the Darcy-Weisbach friction coefficient “f”, given by the implicit expression on $\text{sqr}(f)$:

$$1/\text{sqr}(f) = -0.86 \ln [\varepsilon_r / 3.71 + 2.51 / \text{Re} \text{sqr}(f)]$$

Example:

Calculate the friction factor for a flow with $\text{Re} = 1 \text{ E}6$ and relative roughness $\varepsilon_r = 0.005$

Solution: $f = 0.031051$



Pipes in Series with multiple demands. [TUBSER]

From the author's Engineering Collection, included in the ETSII4 module.

This program has two modes of utilization. In DESIGN mode, it calculates the optimal diameters D_i for a set of pipes in series with multiple demands, when the costs relation is known. In OPERATION mode it calculates the total head required to meet the demand with known pipe diameters.

The equation for optimal diameter d_i for a pipe with friction loss factor f_i and flow q_i is shown below:

$$d_i = (\lambda) [f_i q_i^2]^{1/(a+5)}; \text{ with a cost relation given as } C_i = A d_i^a$$

where λ reflects the combined configuration in series,

$$\lambda = [k/\Delta H \sum \{ L_i (f_i q_i^2)^{a/(a+5)} \}]^{1/5}; \quad i = 1, 2, \dots, n$$

Where ΔH is the available total head (or piezometric height between ends of the pipes), and k is a constant defined as $k = 8/g\pi^2$

Note that Q_i is the *flow through each of the pipes* – not the discharged flow at the point of demand.

Whereas the formula for the total available head when the diameters are known (i.e. not the design case) is given by the expression below:

$$\Delta H = k \sum [f_i L_i q_i^2 / D_i^5]; \quad i = 1, 2, \dots, n$$

Example.

Obtain the optimal diameters for the pipeline with 15 m of total available head, formed by 4 individual pipes of fibrocement (friction loss factor = 0.018, and cost relation given by $C = 0.228 D^{0.567}$); with the following conditions:

Pipe	1	2	3	4
Length (m)	500	200	315	180
flow q (l/s)	100	94	86	74
The results are also provided below:				
Diameter (m)	D1=0.2589	D2=0.2532	D3=0.2453	D4=0.2324

Rotation speed of multi-nozzle sprinkles. [SPRNKL]

From the author's Engineering Collection, included in the ETSII4 module.

This program calculates the rotation speed of a custom sprinkler with N nozzles, located at different distances R_i from the center of a linear pipe-like frame, each inclined at an angle β_i . The water intake is at the center of rotation. The flows Q and areas A are assumed to be the same for all nozzles.

The program prompts for the data values needed, such as the number of nozzles and their distances and orientation angles measured from the sprinkler rod. The rotation speed is obtained using the following formulas, with ρ the density of the fluid, and depending on whether the resistant torque M_r is constant or proportional to the rotation speed:

$$\text{Constant:} \quad w * \sum R_i^2 = (M_r / \rho Q) - [(Q/A) \sum R_i \sin \beta_i] ; \quad i=1,2,\dots,n$$

$$\text{Proportional:} \quad w = - [Q \sum R_i \sin \beta_i] / \{ A [(M_r/\rho Q) + \sum R_i^2] \} ; \quad i = 1,2,\dots,n$$

where the negative sign denotes counter clockwise rotation. Angles are positive in the same counter clockwise direction from the reference (sprinkler arm).

The program uses the AMC_OS/X functions ARCLI and PMTK during the data entry process.

Example.

Assuming that there's no resistant torque; calculate the rotation speed of an irrigation sprinkler with 4 nozzles situated as shown in the table below. Each nozzle has an external diameter of 10 mm, and an exit flow of 7.5 l/min.

Nozzle	1	2	3	4
R_i (m)	1.5	3	-1.5	-3
β_i (deg)	45	45	-90	-90

$$Q = 1.25 \text{ E-4 m}^3/\text{s}$$

$$A = 1.25 \text{ E-4 m}^2$$

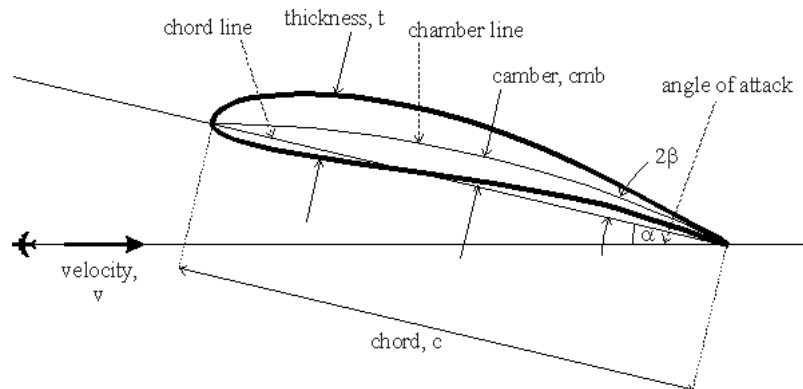
The result is $w = 0.5434 \text{ rad/s}$ counter-clockwise.

What would it be if there is a resistant torque $M_r = 1 \text{ Nm}$?

$$\text{Density} = 1 \text{ gr/cm}^3 \Rightarrow w = 0.2519 \text{ rad/s counter-clockwise.}$$

Lift forces on Joukowski aerofoils. [PERJOW, JOW, AJOW]

From the author's Engineering Collection, included in the ETSII4 module.



These programs calculate the sustentation forces on a Joukowski aerofoil immersed in a uniform flow U with an angle of incidence α , and the aerofoil is characterized by its thickness τ and camber angle 2ε .

The lift force L in the direction perpendicular to the oncoming stream and upwards is given by the product of the density, the potential flow and the circulation around the aerofoil Γ , that is:

$$L = -\rho U \Gamma$$

According to the Kutta condition that requires the velocities be finite at the aerofoil's trailing edge, there's just one possible value of the circulation and it's expressed as follows, where "a" is half the distance to the edge of the foil:

$$\Gamma = -4\pi U a \tau \sin(\alpha + \beta);$$

Therefore the lift force is expressed as: $L = 4\pi \rho U^2 a \tau \sin(\alpha + \varepsilon)$;

The main driver program prompts for all the required data automatically. The camber angle β is expressed as a function of the maximum camber within the profile (at point of null x-coordinate) and X-coordinate of the tail point (cusp), using the formulas

$$2\varepsilon = \arcsin \left\{ \frac{2 f_{xt}}{(f^2 + xt^2)} \right\}; \text{ and}$$

The aerofoil is parameterized into its image cylinder using the Joukowski transform (with parameter $a = xt/2$) by the subroutine "JW", and the results are output sequentially including the transformation parameters (circle origin coordinates and radius), the circulation and the xy components of the lift force.

Programs JOW and AJOW are also available to calculate the equivalent coordinates between the image plane (circles) and back to the original plane (Joukowski aerofoils) – using the transform equations:

$$z = z' + a^2/z' ; \text{ and}$$

$$z' = z/2 \pm \sqrt{(z/2)^2 + a^2} \quad - \text{ using the root } \geq a$$

Examples.

Calculate the lift forces on a Joukowski aerofoil immersed in a uniform flow of density 1.2 kg/m^3 , and uniform velocity 150 km/h with an angle of attack of 45 deg . The maximum camber of the profile (where $x=0$) is $f=0.214 \text{ m}$, and the coordinates of the head and tail points are: $x_h = -0.904 \text{ m}$, $x_t = 0.8 \text{ m}$

The obtained results are as follows:

$\varepsilon=14,9760 \text{ deg}$	$FX=-8.814,1304 \text{ N}$
$R'=0,5499 \text{ m}$	$FY=8.814,1304 \text{ N}$
$X0'=-0,1312 \text{ m}$	$SUST=12.465,062 \text{ N}$
$Y0'=0,1421 \text{ m}$	
$CIR=-249,3013 \text{ N s m}^2 / \text{kg}$	

Note. The routines JW and AJW can be re-written much more efficiently using functions from the 41Z module, (pity it wasn't around 30 years ago ;-)) as shown below. The data is expected in the stack registers on entry as shown: R00: parameter a, Y: Imaginary part, X: Real part

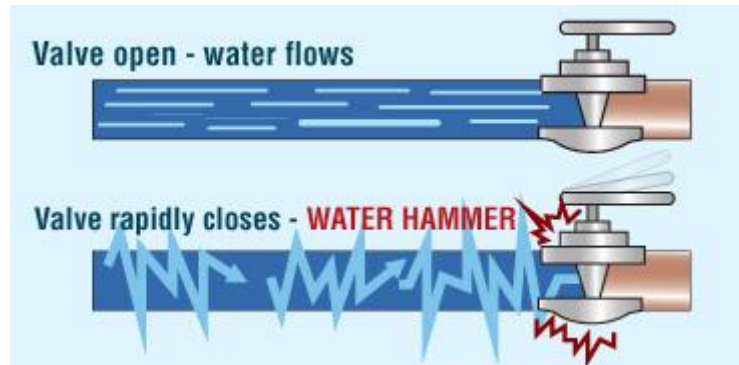
01 LBL "JWZ"	01 LBL "AJWZ"
02 Z^2	02 ZENTER^
03 ZENTER^	03 1/Z
04 ZENTER^	04 ZENTER^
05 RCL 00	05 0
06 X^ 2	06 RCL 00
07 +	07 X^2
08 ZSQRT	08 Z*
09 Z+	09 Z+
10 Z<>W	10 END
11 LASTZ	
12 Z-	
13 END	

Water Hammer Transients. [ARIETE]

From the author's Engineering Collection, included in the ETSII4 module.

This program calculates a simplified transient analysis of the pressure inside of a forced pipe, when the exit valve follows a given closing curve.

The program does not take into account the friction so the pressure waves would continue to transient indefinitely.



Let H_0 be the stationary head pressure inside of the pipe, let “a” be the speed at which the pressure wave propagates, L the pipe length, and $v(t)$ the valve closing speed equation.

In ideal conditions (no friction dampening) the instant pressure change at the analysis point placed at a distance d from the valve (exit), is modelled by the following expression:

$$H(t) = H_0 + a/g \sum (-1)^{\text{int}(k/2)} [v_0 - v(t'')] ; k=1,2..n \quad (1)$$

where t'' is an auxiliary time defined as: $t'' = t - t_0 \text{int}(k/2) + (-1)^k t_d$

and: $n = 2 \text{int}[(t+t_d)/t_0]$ iif $t < t_c = t_d + t_0 \text{int}[(t+t_d)/t_0]$, or: $n = n-1$ iif $t \geq t_c$

$t_0 = 2L/a$ is the time it takes for the pressure wave to return to the valve for the first time;

$v_0 = v(t_0)$ is the initial speed before the pressure waves affect the analysis point.

$t_d = d/a$ is the time it takes for the pressure wave to reach the analysis point for the first time;

The valve closing speed equation is supposed to be relatively slow – as referenced to the characteristic parameters t_0 and t_d . The equation needs to be entered as a separate program using a global label. It can also have discontinuities, used to simulate sudden closing steps. This is handled by providing the *left-value in the Y register*, and the *right-value in the X register*. Because of this, two pressure results per time instant will be calculated, one before (H_- in the Y register) and another after (H_+ in the Z register) the instant t (in the X register).

The user must provide a step size value to calculate the successive pressure values at the point of analysis. These can be plotted on the thermal printer using the PRPLOT program as well. The program uses the OS/X function PMTA to prompt for the closing equation label name.

Note: Equation (1) above was deduced from a conceptual approach, extending the simple cases to a general-purpose scheme valid for any time instant. To the author's knowledge there is no reference in the literature for an equivalent expression. The closer I've seen is the Bergeron Method, used to calculate reflection of electrical signals.

Example.

Study the pressure transients at the closing valve (thus $d=0$) during a time interval of 10 seconds, in a forced pipe with 1,200 m length. Assume that the pressure wave moves at 800 m/s (hence $td = 2L/a = 2,400/800 = 3$ s). For the gravity acceleration use $g = 10$ m/s² to simplify the numeric results.

The closing valve equation is given as follows:

$$\begin{aligned} v(t) &= 1.2 \text{ m/s} & \text{if } 0 \leq t < 1 \\ v(t) &= 0.8 \text{ m/s} & \text{if } 1 \leq t < 3 \\ v(t) &= 0.4 \text{ m/s} & \text{if } 3 \leq t < 4 \\ v(t) &= 0 & \text{if } t \geq 4 \end{aligned}$$

Which has been programmed under the global label “V<T>” using the “from right to left” rule, as follows: (for your convenience, the ETSII4 module includes pre-programmed this valve closing equation, and another one that you can also use to test the operation - under the labels “V<T>” and “V(T)”).

Using a step size of 0.25 s the results for the pressure changes are shown in the table below:

t	H(-)	H(+)	t	H(-)	H(+)	t	H(-)	H(+)	t	H(-)	H(+)	t	H(-)	H(+)
0	0	0	2	32	32	4	64	32	6	32	-32	8	-32	-32
0.25	0	0	2.25	32	32	4.25	32	32	6.25	-32	-32	8.25	-32	-32
0.50	0	0	2.5	32	32	4.5	32	32	6.5	-32	-32	8.5	-32	-32
0.75	0	0	2.75	32	32	4.75	32	32	6.75	-32	-32	8.75	-32	-32
1	0	32	3	32	64	5	32	32	7	-32	-32	9	-32	32
1.25	32	32	3.25	64	64	5.25	32	32	7.25	-32	-32	9.25	32	32
1.5	32	32	3.5	64	64	5.5	32	32	7.5	-32	-32	9.5	32	32
1.75	32	32	3.75	64	64	5.75	32	32	7.75	-32	-32	9.75	32	32

The critical instants occur at $t=1, 3, 4, 6$, and 9 ; with the maximum pressure change occurring at $t=6$.

01	LBL V<T>”	17	GTO 02	33	0,4
02	4	18	X=Y?	34	RTN
03	X<>Y	19	GTO 01	35	LBL 04
04	X>Y?	20	1.2	36	0,4
05	GTO 06	21	RCL X	37	RCL X
06	X=Y?	22	RTN	38	RTN
07	GTO 05	23	LBL 01	39	LBL 05
08	3	24	1.2	40	0,4
09	X<>Y	25	0,8	41	0
10	X>Y?	26	RTN	42	RTN
11	GTO 04	27	LBL 02	43	LBL 06
12	X=Y?	28	,8	44	0
13	GTO 03	29	RCL X	45	RCL X
14	1	30	RTN	46	END
15	X<>Y	31	LBL 03		
16	X>Y?	32	0,8		

Bergeron Graphic Method [by E. Thvin]*From the User's Program Library Europe #25674, included in the ETSII4 module*

This program calculates the flows and pressure changes on a reservoir discharge pipe when the exit valve closing follows a linear speed. The geometric data include the pipe length (L), its section (S), and the section of the output tube with the valve open (σ). The other required inputs are the propagation speed of the sound waves (a), the initial mass flow before the closing begins (Q_0), and the actual section closing rate (m) -which must follow a linear curve, thus the equation is: $\sigma(t) = \sigma - m \cdot t$.

The over-pressure calculations at a distance from the valve "x" and instant "t" are defined as:

$$\Delta P = [p(x,t) - p(0,0)] / \rho$$

Where ρ is the density of the liquid, and $p(0,0)$ is the initial pressure at the valve.

The sound waves move at the propagation speed given by:

$$a = \sqrt{1 / [\rho (1/\lambda + \delta / \varepsilon E)]},$$

with δ the diameter and ε the thickness; λ the compressibility factor of the fluid and E the elasticity module of the pipe's material. For water, the expression can also be written as:

$$a = 9900 / \sqrt{48.3 + k \delta / \varepsilon},$$

where k is a constant of the material of the pipe (typical values are k=0.5 for copper, k= 1 for alloy, k=1 for lead, k=5 for foundry mixes, etc.)

Example.

Calculate the pressure values at a distance $x = 260$ m from the closing valve in a discharge pipe with 1 km length, of cross section 1 m². Assume the wave propagation to be 1,000 m/s and the initial mass flow $Q_0 = 10$ m³/s. The valve nozzle has an open section $s = 0.1$ m² and closes at a linear rate of 20% during 4 seconds (fully shut). Study the evolution during 8 seconds, with a step size $dt = 1$ s.

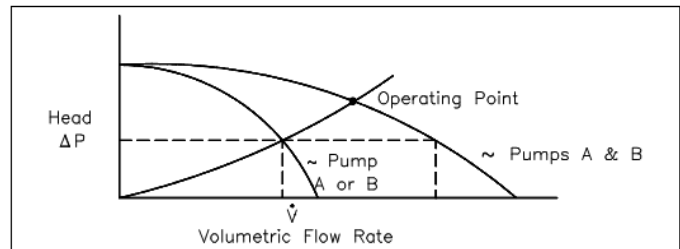
The results are shown in the table below:

t	Q(t)	$\Delta p(t)$	t	Q(t)	$\Delta p(t)$
1	9.1868	813.2241	5	1.5076	102.0886
2	7.5469	1,913.8012	6	1.7399	-1,195.3174
3	4.9568	2,077.6468	7	2.3252	-66.8310
4	2.3360	1,779.1259	8	2.1550	787.9555

Association of Pumps in Parallel. [PH<Q>, PQ<H>]

From the author's Engineering Collection, included in the ETSII4 module.

With these programs the combined head (H) and capacity (Q) values for centrifugal pumps associated in parallel are calculated using the individual head/capacity curves for each pump – modelled as second degree polynomials.



The H<Q> program will check for minimal capacity error condition if one of the pumps would run into stall mode with the given combined flow.

In addition to the driver programs with automated data entry, two subroutines QH and HQ can be used for manual operation or other checks.

Let H1 and H2 be the head-capacity curves for each of the associated pumps, P1 and P2. The general expressions are modelled as second degree curves, with $a_2 < 0$, $b_2 < 0$ for the case of pumps:

$$H_1(Q) = a_0 + a_1 Q + a_2 Q^2 ;$$

$$H_2(Q) = b_0 + b_1 Q + b_2 Q^2 ;$$

The program PH<Q> calculates the combined head for a given capacity, and PQ<H> does the reciprocal job, i.e. obtaining the combined capacity for a given head. The programs use the MCODE function QROOT (also included in the ETSII4 module for your convenience) to calculate the roots of the quadratic equations, and provide logic to select the meaningful roots for the physical problem.

Example.

Obtain the combined flow provided by two pumps in parallel with the height-capacity equations given below, when the combination generates a total head of 20 m

$$H_1(Q) = 30 + 2Q - 0.096 Q^2; \quad Q \text{ in l/s}$$

$$H_2(Q) = 20 + 0.3 Q - 0.015 Q^2; \quad \text{with } Q \text{ in l/s}$$

The solution using PQ<H> is: $Q=25,0000 \text{ l/s}$

Axial velocity at exit of vane profiles. [V2M]

From the author's Engineering Collection, included in the ETSII4 module.

This program calculates the axial velocity at a radius r of an impeller for axial pumps, characterized by the ideal head performance equation: $H_t = a + b r$ and within the boundaries of the blades.

The general expression for the axial velocity is given below:

$$V_m^2 = V_{mi}^2 + 2gb(r-r_i) - 2b \left(\frac{g}{w} \right)^2 \left[a \left(\frac{1}{r_i} - \frac{1}{r} \right) + b \ln \left(\frac{r}{r_i} \right) \right]$$

Where V_{mi} (a constant value) is the exit axial velocity at the root of the blade, and can be determined as a boundary condition when the flow is known, by means of the following numerical integration:

$$Q = 2\pi \int_{R_i}^{R_e} \{ r V_m(r) \} dr ; \quad \text{between } R_i \text{ and } R_e$$

Thus this program will first use a numerical integration routine to obtain the value of V_{mi} , and with it, it'll take upon solving for r in the V_m equation. The nested arrangement explains the long execution times, as the integral needs to be calculated at each iteration of the root finder!

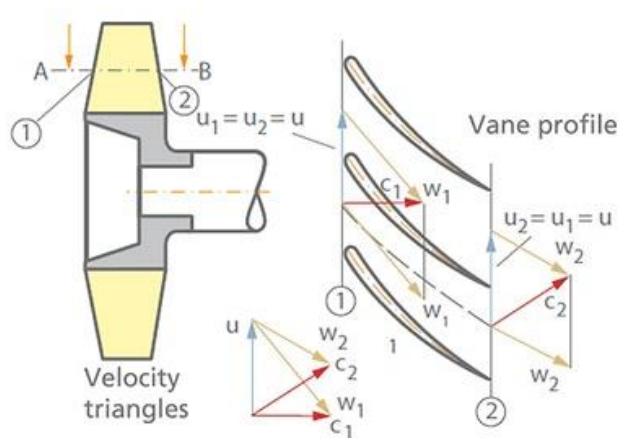
Both the numeric integration and root-finding routines are included in the ETSII4 module for your convenience, so there are no additional dependencies.

Example.

Calculate the exit axial velocity at radius $r = .25$ for an axial pump with the following characteristics:

- $r_i = 0,17$ – interior impeller radius
- $r_e = 0,4$ – exterior impeller radius
- $W = 600$ rpm
- $Q = 2,20$ m³/s
- $H_t = 1 + 2 r$ – theoretical head

The solution is: $V_{2M} = 5,1657$ m/s
 And $V_{mi} = 4,8792$; $V_{me} = 5,6857$

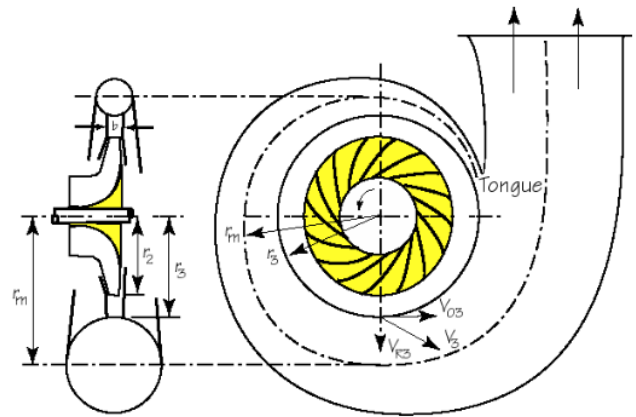


Centrifugal Pump Volute Design. [VOLUTE]

From the author's Engineering Collection, included in the ETSII4 module.

This program calculates the geometric dimensions of centrifugal pump Volute with Circular, Rectangular or Trapezoidal sections - reflecting a given configuration and performance conditions of the pump, as follows:

- Nominal flow thru the impeller, QR
- Impeller outer diameter, D2
- impeller rotating speed in rpm
- Impeller outer width, b2
- impeller blade exit angle, β_2
- impellers' Pfleiderer's coefficient, μ
(can also be calculated as function of number of blades, ZR -and internal diameter, D1)
- The friction losses coefficient in the volute, λ_c (should be zero for circular volutes)



If a diffuser is installed, the outer diameter of the diffuser, D3

if it has blades, the fluid exit radial speed, V_{3u}

without blades, the friction coefficient in the diffuser, λ_d

For Circular section volutes, the volute width at the tongue, bc

For trapezoidal volutes, the volute width at the tongue, bc -and the aperture angle, δ

Once these data are entered the program will prompt for the angle to calculate the outer radius of the volute. This can be repeated for many angles in a loop to further characterize the volute geometry along the interval $[0, 360]$, i.e. from the tongue to the throat. All angles are entered in degrees.

Data Entry requires functions PMTK and ARCLI from the AMC_OS/X Module.

The PPC root-solving "SLV" routine is used for each calculation of the results in rectangular and trapezoidal section volutes. It is also included in the ETSII module. Routines "R", "T", "T0" are used to describe the equation to solve, depending on the diffuser type.

The routine "NT" (NewType) allows for different types of volutes (Rectangular, Trapezoidal, or circular), and different design parameters known.

Example.

Calculate the dimensions of the three volutes types (use $\delta = 15$ for trapezoidal) for a centrifugal pump with impeller rotating at 1450 rpm, without a diffuser and with the following configuration:

$$Q_r = 52 \text{ l/s}$$

$$D_2 = 350 \text{ mm}$$

$$b_2 = b_c = 10 \text{ mm}$$

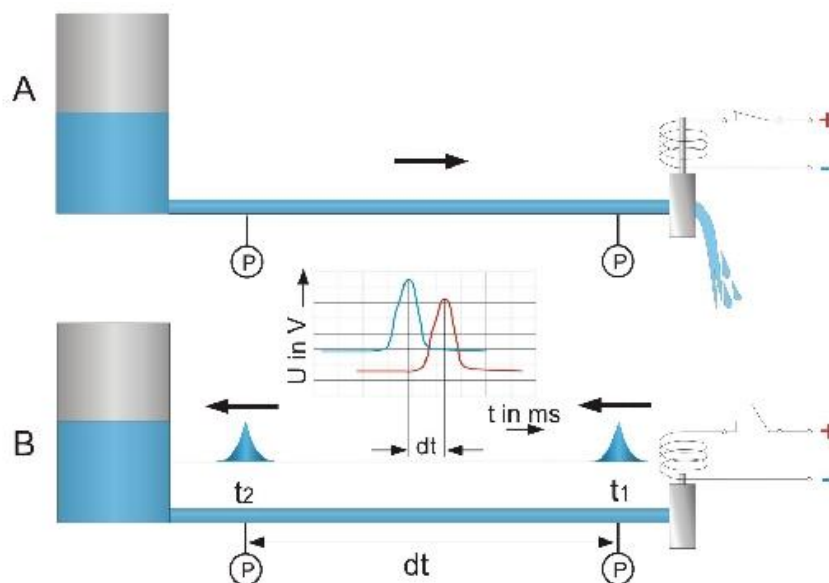
$$\beta_2 = 30 \text{ deg}$$

$$\mu = 0.809$$

$$\lambda_c = 0.04$$

The solutions are given in the table below;

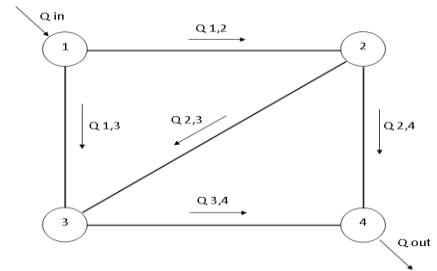
Angle (deg)	Rectangular	Trapezoidal	Circular (center)	Circular radius
0 (tongue)	R=0.1750	R=0.1750	a=0.1750	$\rho=0.0000$
90	R=0.3167	R=0.2230	a=0.1924	$\rho=0.0174$
180	R=0.8965	R=0.2555	a=0.2000	$\rho=0.0250$
270	no convergence	R=0.2843	a=0.2060	$\rho=0.0310$
360 (throat)	no convergence	R=0.3115	a=0.2111	$\rho=0.0361$



Pipe network analysis – Hardy Cross method. [by Marc Peraya]

From User's Program Library Europe #26065, included in ETSII4 module

This program calculates the flows, pressure drops and velocities in a water distribution pipe network using the method of Hardy-Cross. The network size can go up to 253 registers, as a combination of the number of pipes, and loops:
 $\#Reg = 13 + \#P + \#L + \sum (\#P_m)$



Let ε be the pipe roughness in mm and ν the fluid cinematic viscosity (for water equals $1.31 \text{ E-6 m}^2/\text{s}$). Let D be the pipe diameter in meters, Q the flow in m^3/s , and L the pipe length in m. Then the velocity of the fluid is obtained using the Prandtl-Colebrook formula, as function of the per-unit head loss (J) and total head loss $\Delta h = J.L$ – as per the expression below:

$$V = \{ -2 \log [2.51 \nu / D.\text{sqr}(2.g.J.D)] + \varepsilon / 3.71.D \} * \text{sqr}(2.g.J.D)$$

with: $V = Q/\text{Sec} = 4Q / \pi D^2$

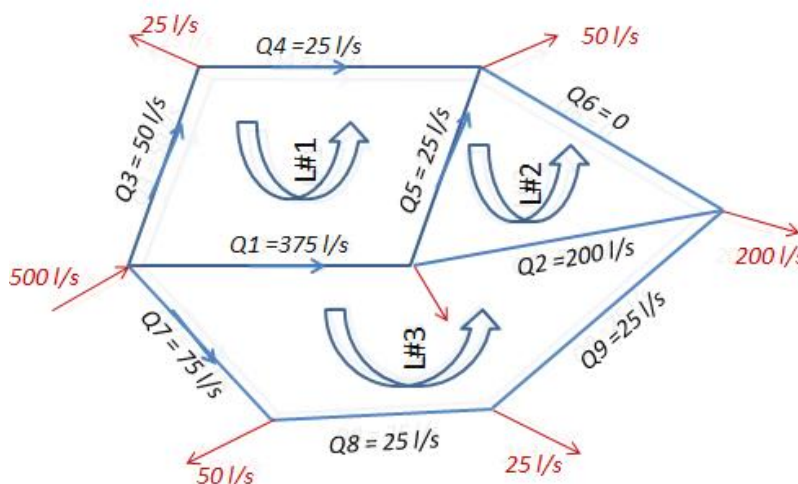
writing: $A = \text{sqr}(2.g.J.D)$, this is solved with iterative calculations given by:

$$A_n (\pi D^2/4Q_n) = 1 / \{ -2 \log [2.51 \nu / D.A_{n-1}] + \varepsilon / 3.71.D \}$$

The user needs to establish an initial distribution of flows through all pipes based on the extractions and insertions at the consumption nodes (junctions). This initial guess is then used as the basis for a new adjusted set of values, applying a correction factor to the previous values.. The program uses a flow correction factor for each loop until it reaches a residual value of 1 E-4 (convergence factor). The flow correction factor is obtained as:

$$\Delta Q = \{ \sum \Delta h / 2 \sum |\Delta h/Q| \}; \quad \text{with the new flow being: } Q_n = Q_{n-1} - \Delta Q$$

Example.



Calculate the flows, velocities and pressure drops for each pipe in the network represented on the left, with 3 loops and 9 pipes in total. Use a guess initial flow distribution Q_0 not all zero. The roughness is $e = 0.1 \text{ mm}$, and the pipe data is tabulated below – together with the initial flow estimation derived from the consumption node information.

Preparing the input data.

There are two types of data: Network Data and Pipe data. Each type is store in a separate file, and once calculated, the obtained results will be included in the pipe data file. Then you should follow the process below:

1. With all the network characterized (extracted flows at the junctions, pipe diameters and lengths), choose an initial guess flow distribution through the pipes (not all zero!)
2. Number the pipes from 1 to “#p”. The order is not unique but must be followed throughout the program.
3. Describe the loops by choosing a reference direction as positive flow. Also try to choose them so that there are a minimum number of pipes per loop - this will accelerate the convergence of the method and reduce the calculation time. Then number the loops.
4. The pipe number within a loop is positive if the flow through that pipe has the same sign as the convention for the loop.

After entering the roughness, the program presents the main menu shown blow. You can always return here by pressing [E] at any time.



Network File. – You create this file pressing [A] in the main menu.

The network file contains the number of pipes that constitute each loop. The first and last data registers contain a zero. Each loop section is separated by a zero-entry in the file. The file size is: $S = 1 + \#m + \sum \#p_m$, where $\#p_m$ is the number of pipes of the m-th loop (some pipes will belong to two loops). The file starts at $R\# = (13 + \#p)$, and ends at $R\# = 13 + \#p + \#m + \sum \#p_m$

Pipes File. – Created pressing [B], modified pressing [b]

The pipes file contains the diameters, flows, lengths and head losses for each pipe. Each pipe requires four data registers; thus its size is $S2 = 4(\#p - \text{starting at } R13)$.

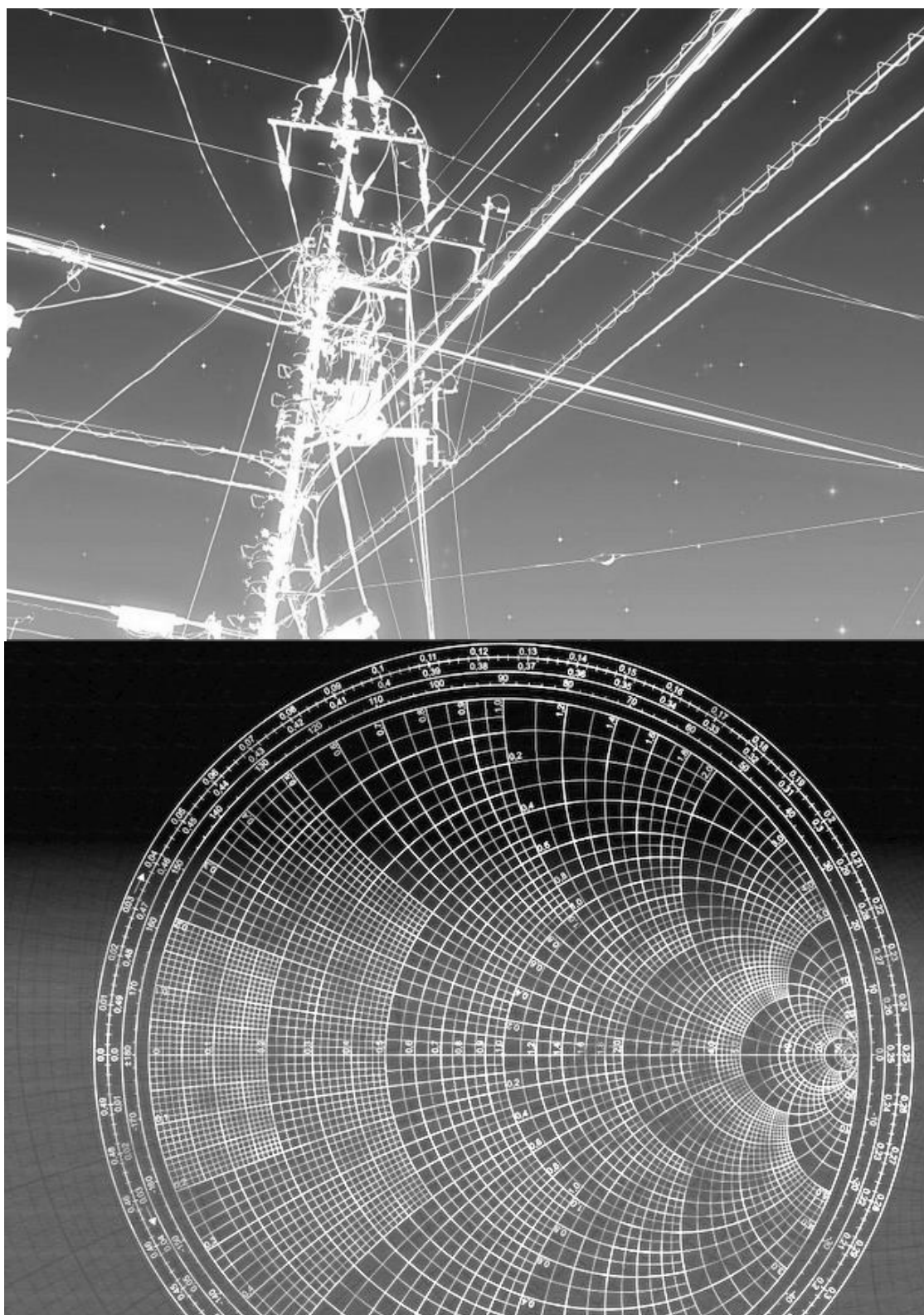
Calculation and presentation of results.

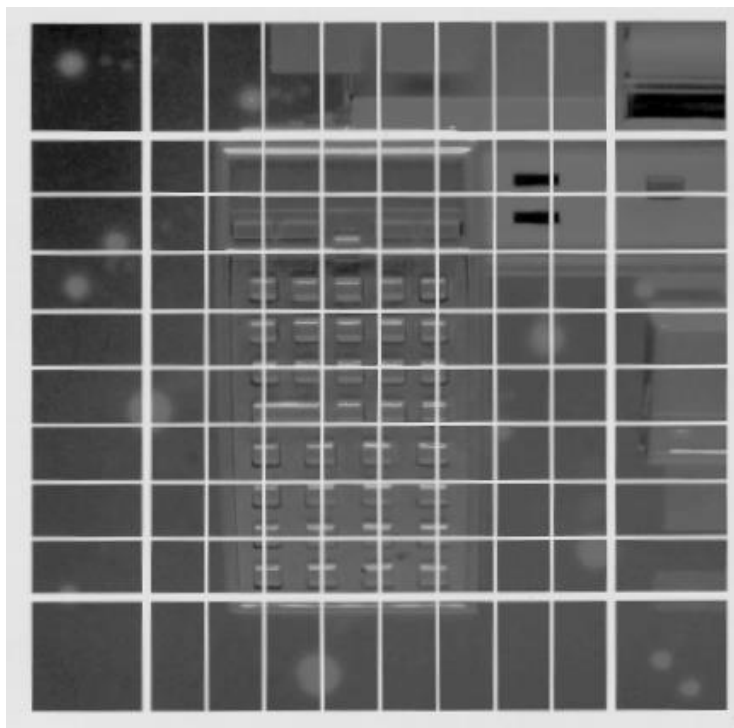
Single pipe results pressing #p, [D], multi-pipe results pressing bbb.eee, [d]

#	D (m)	Qo (m3/s)	L (m)	#	Q (m3/s)	V (m/s)	Δh (m)
1	0.40	0.375	300	1	0.383	3.05	5.42
2	0.40	0.20	300	2	0.195	1.55	1.46
3	0.15	0.50	200	3	0.038	2.13	5.95
4	0.15	0.25	250	4	0.013	0.71	0.93
5	0.20	0.25	200	5	0.039	1.23	1.45
6	0.15	0.00	300	6	0.001	0.06	0.01
7	0.20	0.075	200	7	0.79	2.52	5.79
8	0.20	0.025	250	8	0.29	0.73	1.06
9	0.20	0.00	300	9	0.004	0.13	0.04



Electrical Engineering.



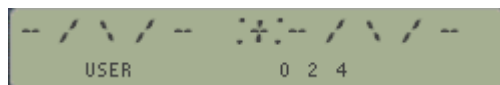
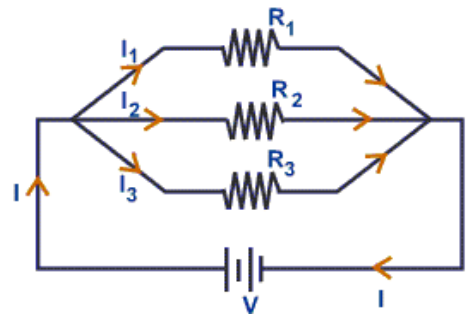


Association of Resistors / Capacitors. $[\Sigma RP, \Sigma CS]$

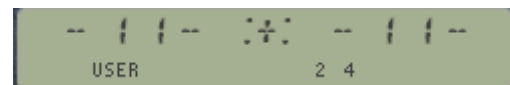
From the author's Engineering Collection, included in the ETSII5 module.

Here's a very simple program to calculate the resultant of association of resistors in parallel or capacitors in series. Easy does it, just a direct application of the well-known formulas for the association. Understanding that the same formulas are used for both on the appropriate case.

The calculator will briefly show a "sketch" of the type of association, as illustrated below:



or:

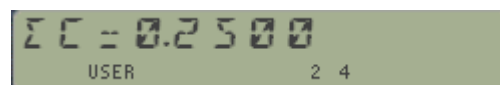


The number of components is prompted first, followed by a loop to enter all the individual values. Finally, the result sum is shown in the display.

Example:

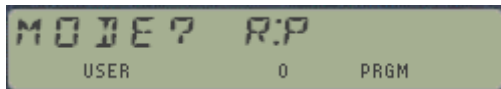
Calculate the equivalent value to 4 capacitors in series, each if 1 pF.

The result is:



Delta-Wye Impedance Transformation. [D<>Y, Y<>D]*From the author's Engineering Collection, included in the ETSII5 module.*

A standard example for any EE class, this program makes the conversion between the Delta and Wye equivalent of a load configuration. The data can be entered in rectangular or polar form, as selected by the user in the initial prompt.

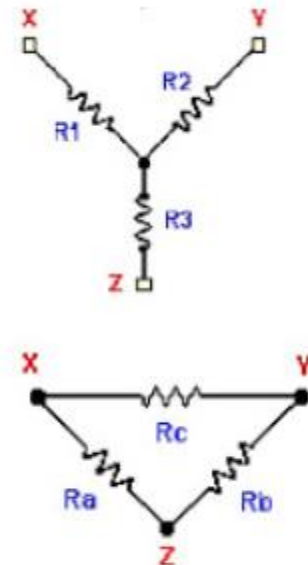


If a peripheral printer is plugged in, the program draws a sketch showing the letter convention for each configuration. It then proceeds with the prompts for the values of the individual impedances.

Let a, b, c the three nodes of the load configuration. We'll call Z_a , Z_b , Z_c the three loads in the Wye arrangement between these nodes and ground; and Z_{ab} , Z_{bc} , Z_{ca} the equivalent loads in the Delta arrangement between each of them.

The expressions used are the well-known formulas shown below:

$$\begin{aligned} Z_{ab} &= (Z_a Z_b + Z_a Z_c + Z_b Z_c) / Z_c & Z_a &= Z_b Z_c / (Z_a + Z_b + Z_c) \\ Z_{bc} &= (Z_a Z_b + Z_a Z_c + Z_b Z_c) / Z_b & Z_b &= Z_a Z_c / (Z_a + Z_b + Z_c) \\ Z_{ca} &= (Z_a Z_b + Z_a Z_c + Z_b Z_c) / Z_a & Z_c &= Z_a Z_b / (Z_a + Z_b + Z_c) \end{aligned}$$

**Example.**

Convert the Delta configuration given below into the Wye equivalent, then back into the original to check the results.

$$\begin{aligned} Z_{ab} &= 1 + 2j ; & Z_a &= 1.01 + 0.57j \\ Z_{bc} &= 3 + 4j ; & Z_b &= 0.67 + 0.33j \\ Z_{ca} &= 5 + 6j ; & Z_c &= 2.00 + 1.67j \end{aligned} \quad < == >$$

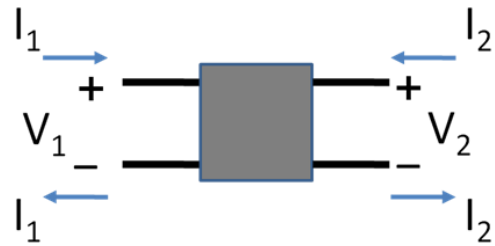
HP-41Z Version. [ZWYE, ZDLT, DYD]

There are two version of the Delta<>Wye conversions included in the module. The second version uses the complex arithmetic functions from the 41Z module, which needs to be plugged in to run properly. The U/I is very similar and the same convention is used to name the loads on each configuration, thus the same user instructions apply.

2-Port Network Matrix Transforms. [ABCD, Y<>Z, etc.]- [by G. Gil]

From Users Program Library Europe #25242, included in the ETSI5 module.

The first set of routines can come handy to calculate the different forms of a 2-port network matrix, when one of them is known. The matrix elements are stored in an X-Memory File, names "HTYZ" so there's no need to re-enter the results to make chain conversions. The Data format is expected to be in rectangular form.



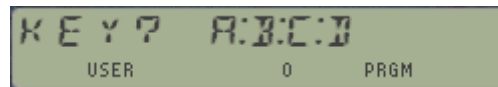
The program ABCD is useful to calculate the fourth element of a 2-port reciprocal network matrix in Transmission Line form ("T"), using the definition relationship between their elements: $AD - BC = 1$. In this case the data format can be either polar or rectangular form, selected by the user at the initial question "MOD? R:P".

Example1.

Calculate the D element of a quadrupole's Transmission matrix with the following known elements: $A = 1 + 2j$; $B = 3 + 4j$; $C = 5 + 6j$

The result is:

$$D = 11.6 + 13.2j$$



Example 2.

Obtain the equivalent matrices for a quadrupole with the initial impedance matrix: [Y]:

$$\begin{matrix} 1+2j & ; & 3+4j \\ 5+6j & ; & 7+8j \end{matrix}$$

The results are shown below

$$\begin{matrix} Z_{11} = -0.500 + j0.438 & ; & Z_{12} = 0.250 - j0.1882 \\ Z_{21} = 0.375 - j0.313 & ; & Z_{22} = -0.125 + j0.063 \end{matrix}$$

$$\begin{matrix} T_{11} = -1.361 + j0.033 & ; & T_{12} = -0.082 + j0.098 \\ T_{21} = 1.574 + j1.311 & ; & T_{22} = -0.279 - j0.066 \end{matrix}$$

$$\begin{matrix} H_{11} = 0.200 - j0.400 & ; & H_{12} = -2.200 + j0.400 \\ H_{21} = 3.400 - j0.800 & ; & H_{22} = -6.400 - j3.200 \end{matrix}$$

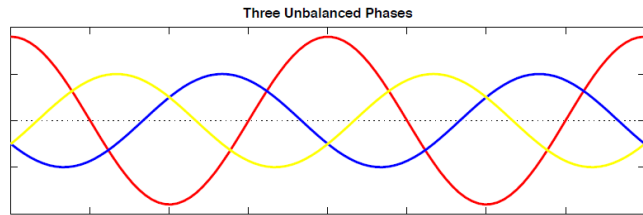
Note that some of the routines like $Y \leftrightarrow H$, $Y \leftrightarrow Z$, $Z \leftrightarrow H$, and $Z \leftrightarrow T$ are reciprocal and can be called twice to undo the conversion – whereas the others are separate routines.

(*) The matrix conversion routines are based on User's Library program UPLE #25242 written by G. Gil. I added the X-Memory file storage to facilitate chained conversions.

3-Phase Systems Symmetrical Components. [SYM3] - [by Ed Borrebach]

From PPCCJ V11N8 p26, included in the ETSII6 module

In electrical engineering, the method of symmetrical components simplifies analysis of unbalanced three-phase power systems under both normal and abnormal conditions. The basic idea is that an asymmetrical set of N phasors can be



expressed as a linear combination of N symmetrical sets of phasors by means of a complex linear transformation. In the most common case of three-phase systems, the resulting "symmetrical" components are referred to as direct (or positive), inverse (or negative) and zero (or homopolar). The analysis of power system is much simpler in the domain of symmetrical components, because the resulting equations are mutually linearly independent if the circuit itself is balanced.

The unbalanced initial system $\{V_a, V_b, V_c\}$ is defined in terms of three balanced systems V_0, V_1, V_2 – where V_0 represents three phasors in direct sequence; V_1 represents three phasors in inverse sequence, and V_2 is the homopolar phasor set. The following relationships define the back and forth conversions between the three sets:

$$\begin{aligned} V_0 &= \frac{1}{3}(V_a + V_b + V_c) \\ V_a &= V_0 + V_1 + V_2 \\ V_b &= V_0 + a^2 V_1 + a V_2 \\ V_c &= V_0 + a V_1 + a^2 V_2 \end{aligned} \quad \begin{aligned} V_1 &= \frac{1}{3}(V_a + a V_b + a^2 V_c) \\ V_2 &= \frac{1}{3}(V_a + a^2 V_b + a V_c) \end{aligned}$$

Where a is the phase operator, defined as: $a = 1 \angle 120^\circ$

The program will first ask for the direction of the conversion, whereby "TO" means from unbalanced system to symmetrical components, and "FROM" is the opposite direction.

FROM/TO?
USER 34 ALPHA

VECT#1 V74
USER 0 34

OR:

ZRO SEQ V74
USER 2 4

Example.

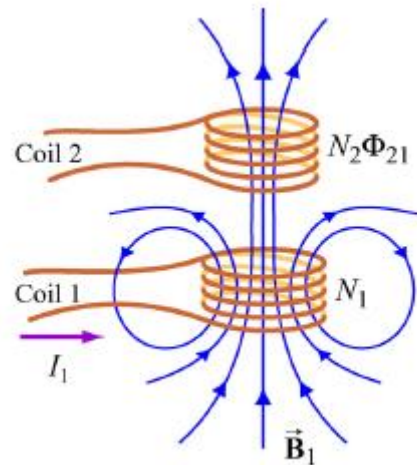
$$\begin{aligned} V_a &= 2.093 \angle 45^\circ & V_0 &= 0.682 \angle -167.89^\circ \\ V_b &= 2.846 \angle -140^\circ & V_1 &= 1.208 \angle 61.94^\circ \\ V_c &= 1.302 \angle -176.48^\circ & V_2 &= 1.674 \angle 19.45^\circ \end{aligned} \quad \Leftrightarrow$$

Mutual Inductance of 2 coaxial loops. [MUTIND]

From the author's Engineering Collection, included in the ETSII6 module.

This program computes the mutual inductance of a pair of coaxial circular coils as a function of the two radii and their axial separation. Using the MCODE complete Elliptic Integral functions results in a faster execution and more accurate results than using other approximations.

This application shows a practical utilization of functions **ELIPK** and **ELIPE** in the SandMath module to calculate the mutual inductance between two coaxial circular coils of radius r_1 and r_2 , separated a distance "d". The program is based on the example taken from page# 83 of the NASA SP-42 document, "Space Resources and Space settlements".



Note the conventions used in the definition, especially for the "k" parameter – not squared!

Example.

Calculate the mutual inductance of 2 coaxial coils with radius $r_1 = 0.2$ m and $r_2 = 0.25$ m, separated a distance of: 0.1 m. How would that change if the separation was $d = 0.2$ m instead?

Test cases: with $r_1 = 0.2$, $r_2 = 0.25$

1. $d = 0.1 \rightarrow MI = 2.48787E-7$
2. $d = 0.2 \rightarrow MI = 1.23957E-7$

These results are in henries.

M = mutual inductance of coil pair (henries)

$$M = \frac{8\pi \times 10^{-7} \sqrt{r_1 r_2}}{k} \left[\left(1 - \frac{k^2}{2}\right) K - E \right]$$

where

$$k^2 = m = \frac{4r_1 r_2}{[(R + r)^2 + x^2]}$$

01 LBL "MIND"	16 4	30 ELIPE (ΣFL# 41)	44 *
02 "R1=?"	17 *	31 STO 09	45 RCL 06
03 PROMPT	18 RCL 06	32 E	46 RCL 07
04 STO 06	19 RCL 07	33 RCL 05	47 *
05 "R2=?"	20 +	34 2	48 RCL 05
06 PROMPT	21 X^2	35 /	49 /
07 STO 07	22 RCL 05	36 -	50 SQRT
08 LBL 00	23 X^2	37 RCL 08	51 *
09 "d=?"	24 +	38 *	52 "MI="
10 PROMPT	25 /	39 RCL 09	53 ARCL X
11 LBL C	26 STO 05	40 -	54 PROMPT
12 STO 05	27 ELIPK (ΣFL# 43)	41 PI	55 GTO 00
13 RCL 07	28 STO 08	42 *	56 END
14 RCL 06	29 RCL 05	43 8 E-7	
15 *			

Power-Flow Equations: Gauss-Seidel. [PFE-GS]

From the author's Engineering Collection, included in the ETSII5 module.

This program calculates the unknown 2n-variables of a “n”-bus power system, when the admittance matrix and the remaining 2n-variables are known.

In a power-flow system there is one main complex equation for each bus relating the active and reactive power to the voltages of all the other buses, as follows:

$$S_i = P_i + jQ_i = V_i \cdot I_i^*$$

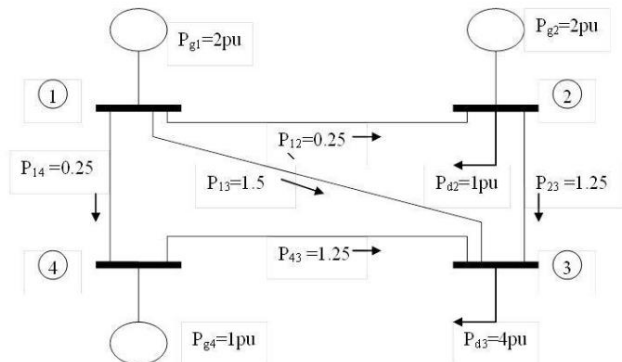
This results in the following 2 power equations per each bus - as function of the voltages on all the other system buses, where α_{ij} are the lag angles between voltage and current and the Y_{ij} are the elements of the admittance matrix of the system:

$$P_i = \sum V_i V_j Y_{ij} \cos(\theta_i - \theta_j - \alpha_{ij}) ; j = 1, 2, \dots, n$$

$$Q_i = \sum V_i V_j Y_{ij} \sin(\theta_i - \theta_j - \alpha_{ij}) ; j = 1, 2, \dots, n$$

The program prompts for the number of the voltage-controlled buses (PV, those with voltage known) –excluding the slack bus (which this program assumes there's only one!). The slack node should be the first ones entered in the bus sequence. This node's voltage is used as reference for all other angles θ_i .

Also the number of Load buses (PQ) is defined as a consequence of the number of PV voltage controlled buses, since the sum of all buses (including the slack) must be equal to the total “n”.



Example 1.

Calculate the voltage angles on each node, the reactive power in the two generation nodes and the active power in the slack node of a three-node power system with the following known variables:

$$\begin{aligned} Y_{11} &= -20j ; & Y_{21} &= 10j ; & Y_{22} &= -20j ; \\ Y_{31} &= 10j ; & Y_{32} &= 10j ; & Y_{33} &= -20j \end{aligned}$$

$$\begin{aligned} PD_1 &= 1; & QD_1 &= 0.5; & |V_1| &= 1 \text{ (slack bus \#1)} \\ PD_2 &= QD_2 = 0; & PG_2 &= 1.5 & |V_2| &= 1 \\ PD_3 &= QD_3 = 1; & PG_3 &= 0 ; & |V_3| &= 1 \end{aligned}$$

The results are as follows:

$$\begin{aligned} PG_1 &= 0.5 \text{ pu.}; & QG_2 &= 0.057 \text{ pu} ; & \arg(V_2) &= 3.822 \text{ deg} \\ QG_1 &= 0.523 \text{ pu} & QG_3 &= 1.036 \text{ pu} ; & \arg(V_3) &= 0.957 \text{ deg} \end{aligned}$$

Example 2.-

A two-bus power system has the following variables known:

$$Y_{11} = 1.841631, \angle = -80.48561$$

$$Y_{12} = 1.904443, \angle = 99.197819$$

$$Y_{22} = Y_{11}$$

$$|V_1| = 1.05, \delta_1 = 0$$

$$SD_1 = 1.15 + 0.31j$$

$$SD_2 = 0.45 + 0.2j$$

a) If $Q_{G2} = 0$, obtain $|V_2|$, δ_2 , P_{G1} and Q_{G1}

b) If $|V_2| = 1$, obtain Q_{G2} , δ_2 , P_{G1} and Q_{G1} (one voltage-controlled bus)

XEQ "DEG"

XEQ "PFE-GS"

Parameter Value	(a)	(b)	Results (a)	Results (b)
#. Buses	2	2		
#.Voltage-Buses	0	1		
$ Y(1,1) $	1.841631			
$\gamma(1,1)$	-80.48561			
$ Y(1,2) $	1.904443			
$\gamma(1,2)$	99.197819			
$ Y(2,2) $	1.841631			
$\gamma(2,2)$	-80.48561			
PD1	1.1500			
PD2	0.4500			
QD1	0.3100			
QD2	0.2000			
PG1	?	?	1.6233	1.6171
PG2 (*)	0	0		
QG1	?	?	0.5357	0.3171
QG2	0	?	0	0.1645
$ V_1 $	1.05	1.05		
δ_1	0	0		
$ V_2 $	?	1	0.8912	1
δ_2	?	?	-13.6293	-12.9669

(*) Assumed to be always zero

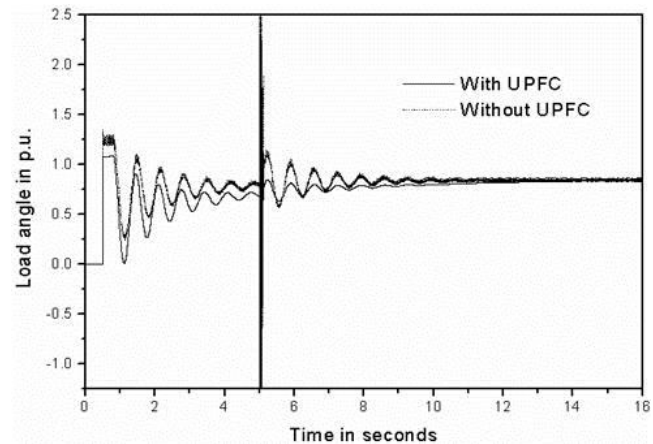
Note.- This was a difficult program to coax within the limitations of the HP-41. Obviously some limitations in the bus definitions needed to be made, but the final result pretty much showcased the capabilities of the calculator – and were very surprising to my power systems class teacher, who wasn't sure how that would be possible!

Synchronous Machine Swing Equation. [SWING]

From the author's Engineering Collection, included in the ETSII6 module.

With this program you can model the transients of a synchronous machine (either as generator or as motor) when a sudden load change is imposed to the system. Typically this is done looking at θ , the electrical angle between a point on the rotor and the synchronous reference frame.

Using this variable, the basic swing equation expressed in system Torque (M_m = mechanical torque, k_d = damping constant, and P_e = electrical power) has the form:



$$(J/2p) \frac{d^2\theta}{dt^2} = M_m + K_d \frac{d\theta}{dt} + P_e \sin \theta / [\omega_1 + (1/2p) \frac{d\theta}{dt}] \quad (1)$$

where $2p$ is the number of poles in the machine and ω_1 is the synchronous rotation speed. Note that the last term cannot be simplified for oscillations of large amplitude; i.e. $\sin \theta$ is not equivalent to the angle, and $d\theta/dt$ is not null. This introduces some complexity but produces more accurate results.

The program uses the 4th-degree Runge-Kutta formula to solve numerically this second-order differential equation. The routine is called for each time instant, and the results are stored in an X-memory data file (named "SWING") as data pairs (t, δ). The stabilization time and the maximum value of the oscillation will occur at t_{max} , determined using the control systems theory – according to the general expression form of a transfer function it follows:

$$t_{max} = \pi / \sqrt{(2p P_e / J \omega_1) - (2p K_d / 2J)^2}$$

$$t_{stb} = 2\pi J / 2p K_d$$

whereas the stationary final value can be easily obtained from the equation (1) above, zeroing the derivatives of the angle:

$$\theta_{finl} = \arcsin(M_n \omega_1 / P_e)$$

Finally, the program uses a step size for the increments as $(h/2) = t_{max} / 60$

Example 1.

Characterize the oscillations when a 28-pole, 600 kW synchronous generator rotating at 22 rad/s experiences a sudden change of the load given by $M_n = 10 \text{ kN.m}$. Assume a dampening constant $K_d = 200 \text{ N.m/rad.s}$, and angular momentum of inertia $J = 444 \text{ kg/m}^2$

Results: The time for maximum angle is $t_m = 0,107755 \text{ s}$

And the stabilization angle:

$$\theta_{fnl} = 0,375424$$

The table below summarizes the values for each instant of time where the calculations have been made:

t (s)	θ (rad)	t (s)	θ (rad)	t (s)	θ (rad)
0,035918	0,172011	0,215510	0,214867	0,395102	0,382101
0,172011	0,482708	0,251429	0,252188	0,431021	0,311437
0,107755	0,634696	0,287347	0,386210	0,466939	0,310819
0,143674	0,546672	0,323266	0,480650	0,502858	0,364791
0,179592	0,343725	0,359184	0,467154	0,538776	0,415264

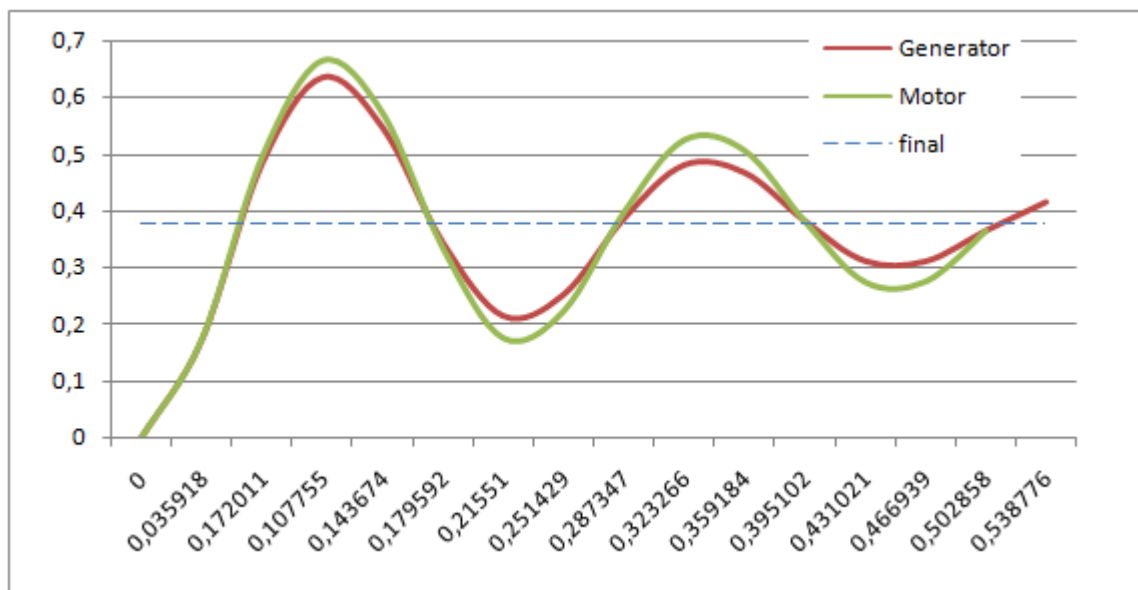
Example 2.

Repeat the previous example but using the synchronous machine as a motor instead.

t (s)	θ (rad)	t (s)	θ (rad)	t (s)	θ (rad)
0,035918	0,172613	0,215510	0,175288	0,395102	0,380103
0,172011	0,492738	0,251429	0,222591	0,431021	0,273351
0,107755	0,662513	0,287347	0,396602	0,466939	0,274786
0,143674	0,570441	0,323266	0,523370	0,502858	0,363795
0,179592	0,333262	0,359184	0,504751	0,538776	

Note that the program will generate data pair results until the SWING file is full, presenting the "END OF FILE" message. At that juncture you can enlarge the file (using RESZFL) and continue getting more results if so desired.

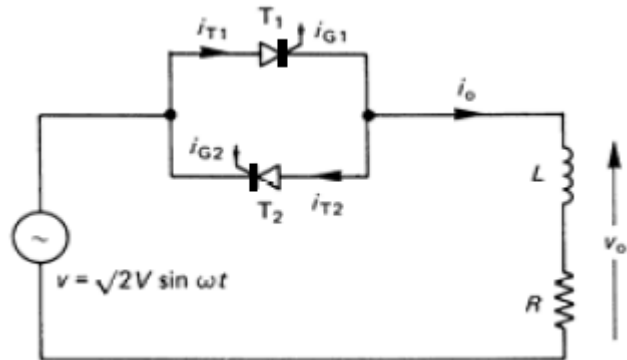
These results can be represented graphically as shown below:



Single-phase AC Regulator. [SPACREG]

From the author's Engineering Collection, included in the ETSII5 module.

This program is useful to calculate the RMS voltage on the output load of a single-phase thyristor AC regulator; based on the thyristor gate delay angle α . Also the current extinction angle β is determined, showing the message "NO REGULA" in not-regulating conditions of the delay angle. The program allows for different combination of load types, resistor and inductance, arranged in series or in parallel.



Let $\omega = 2\pi f$ the input frequency, and $Z = R + jL$ the general load. The expressions used for the four different cases considered are shown below:

1. Resistive Load

Extinction angle: $\beta = \pi$

$$V_{\text{RMS}} = V_{\text{max}} \sqrt{\frac{1}{\pi} [(\pi - \alpha) + \frac{1}{2} \sin 2\alpha]}$$

$$I_{\text{RMS}} = 1/R (V_{\text{RMS}})$$

2. Inductive load

Extinction angle: $\beta = 2\pi - \alpha$

$$V_{\text{RMS}} = V_{\text{max}} \sqrt{\frac{1}{2\pi} [2(\beta - \alpha) + \sin 2\alpha - \sin 2\beta]}$$

$$I_{\text{RMS}}^2 = (1/\pi) \cdot (V_{\text{max}}/\omega L)^2 f(\alpha, \beta)$$

$$\text{Where } f(\alpha, \beta) = \{ (1/2 + \cos^2 \alpha)(\beta - \alpha) + \frac{1}{4}(\sin 2\beta + 3 \sin 2\alpha) - 2 \cos \alpha \sin \beta \}$$

3. R-L load in series.

Let the load natural power angle $\phi = \text{atan}(\omega L/R)$. The extinction angle is obtained solving for β in the equation below:

$$\sin(\beta - \phi) \exp[\beta R/\omega L] = \sin(\alpha - \phi) \exp(\alpha R/\omega L)$$

in addition, for the regulation to occur the delay angle must also be greater than the load natural power angle, that is: $\phi \leq \alpha \leq \pi$

The expression for the load voltage is the same as the case 2) above, but not so for the current RMS, which in this case is a much more elaborate one (no kidding!):

$$I_{RMS}^2 = U_{max}^2 / \pi |z|^2 \{ [\frac{1}{4} (2(\beta - \alpha) + \sin 2(\alpha - \phi) - \sin 2(\beta - \phi)) - \\ - (\omega L / 2R) \sin^2(\alpha - \phi) \exp (2R\alpha / \omega L) \cdot [\exp (-2R\beta / \omega L) - \exp(-2R\alpha / \omega L)] - \\ - 2 \sin (\alpha - \phi) (\omega L / R)^{1/2} \exp [(R / \omega L) (\alpha - \phi)] \cdot f(\alpha, \beta, \phi) \}$$

with:

$$f(\alpha, \beta, \phi) = \exp (-R(\alpha - \phi) / \omega L) [\cos (\alpha - \phi) + R / \omega L \sin (\alpha - \phi)] - \\ - \exp (-R(\beta - \phi) / \omega L) [\cos (\beta - \phi) + R / \omega L \sin (\beta - \phi)]$$

4. R-L load in series.

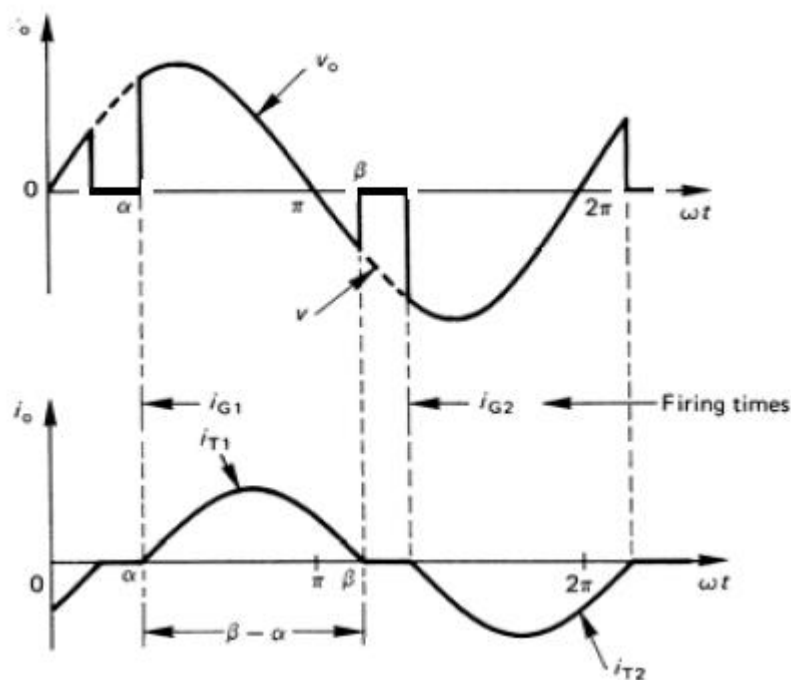
The extinction angle is obtained solving for β in the equation below:

$$\sin \beta \{ 1 + \exp [R(\beta - \alpha - \pi) / \omega L] \} = (R / \omega L) [\cos \beta - \cos \alpha]$$

with the same condition the case before for regulation: $\phi \leq \alpha \leq \pi$

Finally, the expression for the voltage RMS is below:

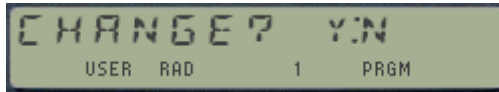
$$V_{RMS} = V_{max} / 2 \sqrt{\pi} \sqrt{ [2(\beta - \alpha) + \sin 2\alpha - \sin 2\beta] + \\ + 2\omega L \sin^2 (\beta - \pi) [1 - \exp [2R(\beta - \pi - \alpha) / \omega L]] }$$



Program Details.

The U/I guides the user during the data entry stage, with several prompts to determine the configuration used. You should use zero values to determine simple resistive or inductive cases, and answer "S/P" for the series or parallel case.

Data can be changed at any time by executing the routine “ND” – new data – which will prompt for a parameters choice until you answer “N” in the “CHANGES?” prompt.



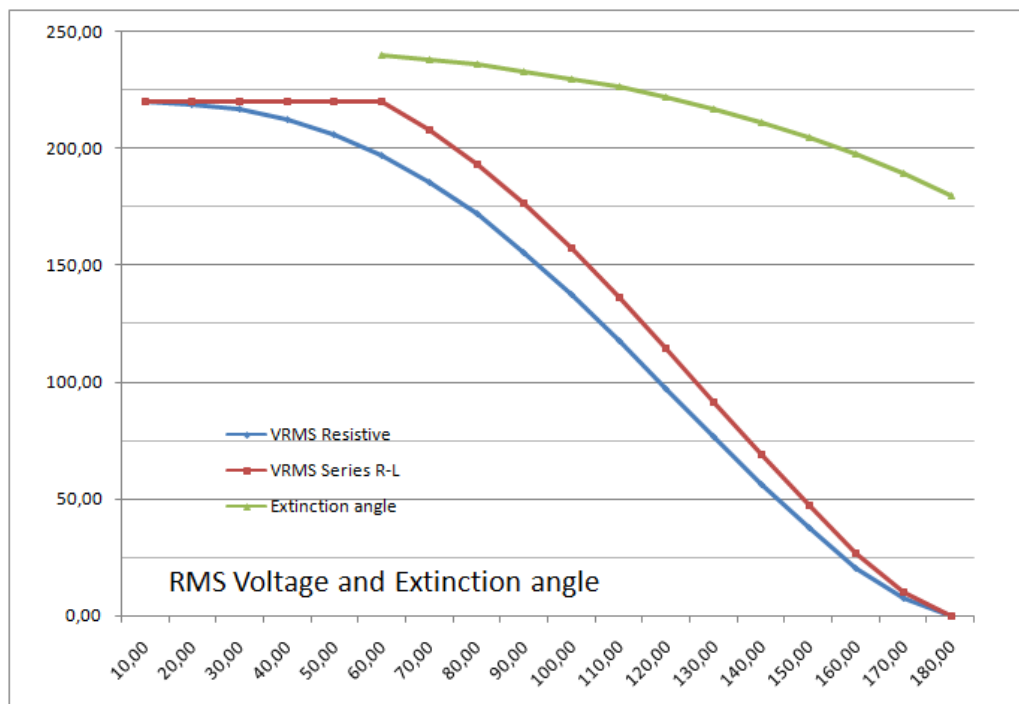
To calculate the extinction angle β the program uses a built-in root-finding routine, “SLV”, also included in the module.

Example .

Obtain the extinction angles and the RMS values of the load voltage for a series configuration with $R = 200 \, \Omega$ and $L = 1.1026 \, \text{H}$ - using trigger angles between 0 and 180, in increments of 10 deg. Compare the results with a simple resistive load case. The network data are: input voltage RMS $V_{\text{inpt}} = 220 \, \text{V}$, and frequency $f = 50 \, \text{Hz}$

The results are tabulated below, and also shown graphically:

Delay angle	Extinction angle	VRMS Series R-L	VRMS Resistive	Delay angle	Extinction angle	VRMS Series R-L	VRMS Resistive
10	NO REG	220	219.88	100	230.01	157.29	137.39
20	NO REG	220	219.038	110	226.27	136.44	117.77
30	NO REG	220	216.81	120	221.94	114.37	97.27
40	NO REG	220	212.67	130	216.97	91.64	76.54
50	NO REG	220	206.26	140	211.31	68.91	56.30
60	240	220	197.33	150	204.89	46.97	37.36
70	238.17	208.01	185.82	160	197.61	26.82	20.65.
80	235.92	193.42	171.82	170	189.37	9.95	7.37
90	233.22	176.41	155.56	180	180.01	9.6 E-5	0.00



Electric Circuits with X-Mem Files. [by Guillermo Castarés]*From the HP-Museum Program Library, included in the ETSII5 module.***Overview**

The program solves an electric circuit of several nodes or several loops, and works with AC using complex matrix representation of the circuit. It solves DC problems as a subgroup of AC problems. *The circuit has to be specified in an ASCII file*, and stored in the extended memory of the calculator. The program reads this file and prepares a complex matrix. This system is solved by the **MSYS** function of the HP-41 Advantage module. Finally, the program shows a complex vector with the results.

This method of entering data allows the user to modify single parameters or components in a circuit and solve the new circuit without having to enter all the data again! It's thus possible to store different circuits to solve in the extended memory. The program accepts all R or $G=1/R$, X or $B=-1/X$, Z or $Y=1/Z$ as well as C and L instead of X.

And to round up this task, *a new data-entry program is also provided*, which builds the ASCII file from the scratch in an interactive way just entering the information in the prompts. The new data-entry program “**EEE**” is completely independent, and its end result is the ASCII file to use as input for “**EEA**”, the circuit solver program. Note that it uses functions from the AMC_OSX module for a more convenient UI, with step-by-step entry sequences.

**Requirements**

The program was developed for a HP-41CX and the HP-41 Advantage module. With all the program routines (EEA, VELs, IELs) loaded in ROM, is possible to solve problems with up to ten Loops/Nodes. The following table shows the number of data register required, according with the number of Nodes/Loops of the circuit:

Loops/Nodes	Data Registers	Elapsed Time to solve*
1	23	1' 30"
2	39	2' 00'
3	63	3' 00"
4	95	4' 30"
5	135	6' 00"
6	183	8' 30"
7	239	12' 00"

(*) Approximated. This elapsed time depends on the number of components.)

<http://www.hpmuseum.org/software/41/41elcirc.htm>

The only limit to the number of elements in the circuit is the memory size of the circuit description file. The elements can be specified freely sorted, or with no sort at all. Another user friendly item of the program is that the number of data registers is automatically set by the **PSIZE** function.

Instructions

The program solves circuits by Nodes Voltages and Loops Currents methods. In the first case, only Current Sources are allowed, in the second case only voltage Sources are allowed. If you have to solve a circuit with mixed sources types, then you have to convert all of them to the same type before to solve the circuit.

The circuit is described in an ASCII file in the extended memory. The name of the file can be freely chosen. The following elements are allowed:

Symbol	Component type
V	Independent Voltage Source
I	Independent Current Source
R	Resistance
G	Conductance
L	Inductance
C	Capacitance
X	Reactance
B	Susceptance
Z	Impedance
Y	Admittance

The following elements have only one parameter: R, G, L, C, X, B. The following elements have two parameters expressed as a complex number in rectangular form: Z, Y, V, I. The values of the parameters should be expressed according to the flags 28 and 29 (decimal symbol and digit groups). Here we assumed that flag 28 is "Clear" and flag 29 is "Set". The first step is to decide if we will solve a Loops Currents problem or a Nodes Voltages problem.

Loops currents

- Identify all the independents loops of the circuit and number them beginning from 1 (one). All the currents are supposed clockwise. In consequence, in each branch (component connecting two nodes) the two currents have opposite courses.
- If a component belongs to a only independent loop, then it's described as belonging to a fictitious loop identified by 0 (zero) (non independent).
- Only voltage sources are allowed. The polarity of them have to be according to the current in the loop

The first record of the input file is: ***In***
where *n* is the number of independent loops.

The following records of the input file describes one component each one:

ncm par1 par2

where:

n is the number of the first loop

m is the number of the second loop

c is the component type

par1 is the value of the component or the real part of it

par2 is the imaginary part of a two parameters component. Assumed 0 if no present.

The last record of the input file is always: **END**

Nodes Voltages

- Node is the join of more than two elements.
- Identify all the nodes of the circuit and number them beginning from 0 (zero). The node numbered zero will be the reference one. The nodes numbered from 1 (one) are the independent nodes.
- Only Current power supplies are allowed. Conventionally is supposed that they are sending current from the first node to the second node.

The first record of the input file is: **Vn**

where *n* is the number of independent nodes.

The following records of the input file describe one component each one:

ncm par1 par2

where:

n is the number of the first node

m is the number of the second node

c is the component type

par1 is the value of the component or the real part of it

par2 is the imaginary part of a two parameters component. Assumed 0 if no present.

The last register of the input file is: **END**

Starting to solve

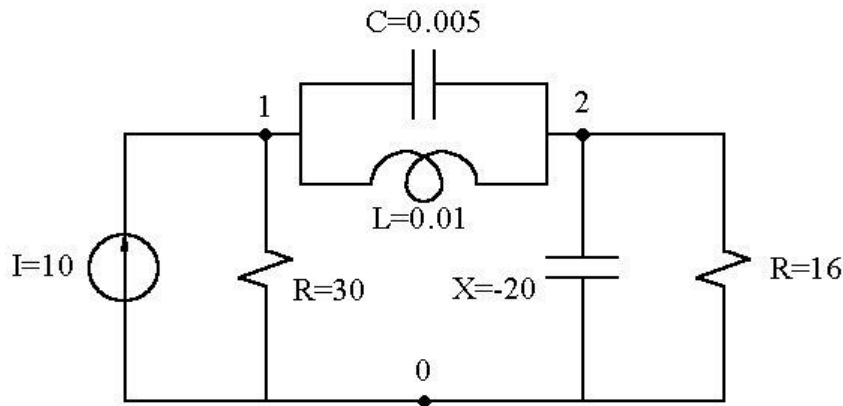
Once the file describing the circuit is ready, it is necessary to put the file name in the alpha register to call the "EEA" program:

Keystrokes:	Display:
[ALPHA]	FileName
[ALPHA]	
[XEQ] [ALPHA]	EEA
[ALPHA]	

If there are capacitances or inductances in the circuit, then the program will ask for the frequency. Finally, the program will show the real and imaginary parts of each current/voltage on the circuit.

Example

To solve the following circuit, we will use the voltage nodes method, because there is a current source.



The first step is to identify the nodes and number them from 0. In this circuit there are a reference voltage node (0) and to nodes 1 and 2. Then we have to create the file with the circuit description:

Keystrokes:	Display:	Comment:
[ALPHA] CIRC1 [ALPHA]	0,0000	file name
10 [XEQ] [ALPHA]	XEQ _	file size
CRFLAS [ALPHA]	10,0000	create file ASCII
[XEQ] [ALPHA]	XEQ _	
ED [ALPHA]	00 ^T	invoke text editor
V2 [R/S]	01 ^T	type of circuit and size
0I1 10 [R/S]	02 ^T	current source
0R1 30 [R/S]	03 ^T	resistance
1C2 0,005 [R/S]	04 ^T	capacitance
1L2 0,01 [R/S]	05 ^T	inductance
0X2 -20 [R/S]	06 ^T	reactance
0R2 16 [R/S]	07 ^T	Resistance
END [R/S]	08 ^T	end of file
[ON]	10,0000	exit editor

Once the circuit is described and stored in the extended memory, we can solve it:

Keystrokes:	Display:	Comment:
[ALPHA] CIRC1 [ALPHA]	10,0000	file name
[XEQ] [ALPHA]	XEQ _	Invoke the program
EEA [ALPHA]	0I1 10	shows first component
	0R1 30	shows second component
	1C2 0,005	...
	FREQ=?	ask for frequency

50 [R/S]	1L2 0,01	shows following component
	0X2 -20	...
	0R2 16	...
	END	end of file
	V(1)X=83,7467	V1 real
[R/S]	V(1)Y=-46,7976	V1 imaginary
[R/S]	V(2)X=82,5013	V2 real
[R/S]	V(2)Y=-41,0423	V2 imaginary
[R/S]	V(1)X=83,7467	shows the first again...

Some Error Messages

- NONEXISTENT: The file contains a component not supported. The circuit type is "I" and contains a current source. The circuit type is "V" and contains a voltage source.
- NO ROOM: There is not enough memory to solve a circuit.
- FL NOT FOUND: The specified circuit file doesn't exist.
- FL TYPE ERR: The specified circuit file isn't an ASCII file.

More Examples.

These are the files for the Math Pac and the Advantage Pac manuals (remember student “AC Dimmer”?), and from Grapevine’s “EE for Students” example – all listed with their respective solutions.

MPAC - Math Pac	ACDIM - Advantage	GRAPEVINE
0 I4	0 I2	0 I3
1 OV1 34 0	1 OV1 5 0	1 OV1 120 0
2 OR1 1	2 OZ1 10	2 OR1 10
3 1R2 1	3 1Z2 0 200	3 1R2 200
4 OR2 1	4 OZ2 0 -30	4 1L2 0.50
5 2R3 1	5 END	5 OC2 1.00 E-5
6 OR3 1		6 OR2 100
7 3R4 1	$I1 = 0.0372 + 0.1311j$	7 2Z3 300 20
8 OR4 2	$I2 = 0.0437 + 0.1543j$	8 OR3 10
9 END		9 OV3 84.85 84.85
		10 END
$I1 = 21$		$I1 = 0.570 + 0.342j$
$I2 = 8$		$I2 = 0.276 + 0.637j$
$I3 = 3$		$I3 = 0.266 + 0.617j$
$I4 = 1$		

Backwards Ladder Analysis Program. [by Gary D. Frey]

From the hp41.org program library, included in the ETSII5 module.

Program description.

One of the simplest ways to analyze a ladder circuit is to assume an output current; then work backwards through the network obtaining all voltages and currents in terms of the assumed output current. for a linear network, gain and impedance throughout the circuit are independent of actual current and voltage levels and the response (all voltages and currents) due to one value of excitation may be linearly scaled to another value of excitation - for example you might want to know all voltages and currents within a circuit for a specific input power in order to determine the voltage and current ratings of all the components.

'BLAP' is a collection of subroutines for the hp-41c which employs the 'backwards ' algorithm. The load current is assumed $1.0+j0$ amperes for convenience. This makes the load voltage R_L volts for resistive load R_L and the load power is R_L watts. Working back toward the generator; if a series impedance is encountered the current is unchanged but the voltage is increased by the drop across the series impedance ($V = V + Z \cdot I$). If a parallel admittance is encountered the voltage is unchanged but the current is increased by the current flowing through the shunt admittance ($I = I + Y \cdot V$).

A library of 28 series/parallel type elements is available (all simple series/parallel RLC combinations. Open and shorted transmission line stubs, and series and shunt impedances). two-port elements may also be included in a ladder circuit. Four two-port elements are provided:

- GB resistive feedback gain block
- BG "backwards" gain block
- TL transmission line
- TF ideal transformer.

The gain block is a reasonable approximation of a single transistor broadband resistive feedback amplifier which is commonly employed in modern circuit design (AVANTEK. OPTIMAX, W-J. ANZAC, etc. amplifiers) and includes the coupling from load to source due to the intentional feedback. BG is the same gain block in the reverse direction which allows analysis in either direction of any ladder circuit, even one including amplifiers.

All element subroutines are given global labels so that they may be called by a separate program which describes the circuit. other elements subroutines may be added to 'BLAP ' -or unused ones may be deleted. the subroutine must complete the input current and voltage in terms of the "known" output current and voltage for the element being modelled.

A brief study of the register usage, the appendix, and the program listings of some of the subroutines used should enable the user to generate his own new elements.

At least two memory modules are required.

Six 'compute and print' commands are available in the BLAP program:

- 'RL' initializes load and 'aviews' frequency
- 'RG' computes and 'aviews' gain
- 'S' computes and 'aviews' forward and input
- 'S' parameters (SF and SI) in DB
- 'Z' computes and 'aviews' $Z = V/I$ at any point
- 'VP' 'aviews' V at any point
- 'IS' 'aviews' I at any point

Except for 'RL' which initializes the circuit (and is the first command usually executed), the V,I data is not disturbed by any of the commands so these may be executed anywhere within the circuit.

“RG” or “S” will normally be the last command executed. in addition to the six commands above, register usage in BLAP is compatible with ' PRPLOT ' in the printer rom making it easy to plot any desired circuit response. 'PRPLOT' supplies the frequency to the circuit description program which in turn returns the computed parameter to ' PRPLOT '.

Using “BLAP”

'BLAP' commands may be manually executed to analyze a given circuit at a single frequency; however most of the time the commands will be stored in a program in order to 'sweep' the selected response versus frequency. the present analysis frequency -in GHz- must be stored in register 08 so the circuit description program will usually be contained within a loop which increments R08 either linearly (additive increments) or logarithmically (multiplicative increments). 'PRPLOT' automatically provides a linearly incremented frequency (x) loop.

'PRPLOT' can be made to provide multiplicative increments by initially specifying a small non zero 'x increment' then multiply R6 by the desired increment in the circuit description program (R06 is 'PRPLOT' X), R17 contents are tacked onto the display name for 'Z', 'VP', or 'IS' to keep track of the output data. usually start with 0 in R17 at load end and increment R17 by one for each new element added.

Begin the program with an 'RL' load initialize command then work toward the generator using the element commands to describe the circuit you may assign often-used commands and elements to user keys to save time. Output commands may be inserted anywhere intermediate results are desired.

The last command within the circuit description loop will normally be either "RG” or “S' to obtain the overall response. Examples of 'BLAP' including using the plotter and both linear and log frequency scales are included along with the program listing to aid the user in creating his own circuit description programs.

Complex number mathematics is usually required for circuit analysis (except at DC or for resistors only). 'BLAP' carries complex numbers in rectangular form for all operations in order to achieve a speed improvement over using R-P and P-R operations.

The routines within 'BLAP' employ only stack registers (X,Y,Z,T,L), R04, and flag 14. "+" and "--" even save 'last X+jY' in the stack (Z+jT). The complex arithmetic commands may be employed for general use outside of 'BLAP' just remember that '1' uses register 04; all other complex operations affect only the stack.

Quick reference guide

#	BLAP Commands	Data Format	Function Performed
1	RL	RL	Initialize Load Resistance
2	RG	RG	Compute Gain for RG Gen.
3	S	RG	Compute SF and SI for RL/RG
4	Z	(use R17 as index marker)	Compute Impedance
5	VP		Compute Voltage to Ground
6	IS		Compute Series Current

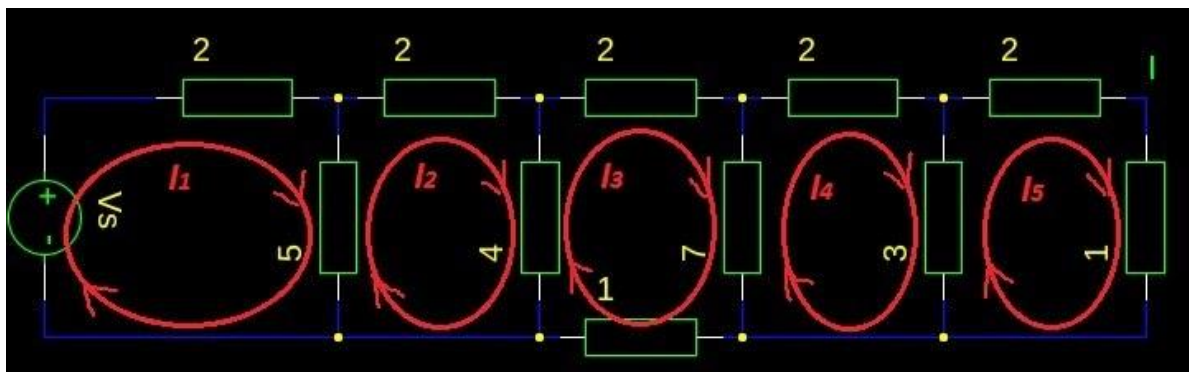
	BLAP Elements	Data Format	Function Performed
1	BG	R0, ENTER^, GDB	Reverse Gain Block
2	GB	R0, ENTER^, GDB	Transistor Gain Block
3	TL	R0, ENTER^, 00, ENTER^, F0	Transmission Line
4	TF	N1, ENTER^, N2	Ideal Transformer
5	PRXS	R, ENTER^, L, ENTER^, C	Parallel RLC in Series
6	PRXP	R, ENTER^, L, ENTER^, C	Parallel RLC in Parallel
7	SRXP	R, ENTER^, L, ENTER^, C	Series RLC in Parallel
8	SRXS	R, ENTER^, L, ENTER^, C	Series RLC in Series
9	PLCS	L, ENTER^, C	Parallel LC in Series
10	PLCP	L, ENTER^, C	Parallel LC in Parallel
11	SLCP	L, ENTER^, C	Series LC in Parallel
12	SLCS	L, ENTER^, C	Series LC in Series
13	PRCS	R, ENTER^, C	Parallel RC in Series
14	PRCP	R, ENTER^, C	Parallel RC in Parallel
15	SRCP	R, ENTER^, C	Series RC in Parallel
16	SRCS	R, ENTER^, C	Series RC in Series
17	PRLS	R, ENTER^, L	Parallel RL in Series
18	PRLP	R, ENTER^, L	Parallel RL in Parallel
19	SRLP	R, ENTER^, L	Serial RL in Parallel
20	SRLS	R, ENTER^, L	Serial RL in Series
21	RP	R in Ohms	R in Parallel
21	RS	R	R in Series
22	LP	L in Nh	L in Parallel
23	LS	L	L in Series
24	CP	C in pF	C in Parallel
25	CS	C	C in Series
26	ZP	R, ENTER^, X	R+jX in Parallel
27	ZS	R, ENTER^, X	R+jX in Series
28	OSTP	R0, ENTER^, 00, ENTER^, F0	Open Stub in Parallel
29	OSTS	R0, ENTER^, 00, ENTER^, F0	Open Stub in Series
30	SSTP	R0, ENTER^, 00, ENTER^, F0	Shorted Stub in Parallel
31	SSTS	R0, ENTER^, 00, ENTER^, F0	Shorted Stub in Series

Complex Math Operator	Operation Performed	Comments
"1"	$X+jY = 1 / (X+jY)$	
"/"	$X+jY = (Z+jT) / (X+jY)$	
"*"	$X+jY = (X+jY) * (Z+jT)$	
"+"	$X+jY = (X+jY) + (Z+jT)$	Last X+jY
"_"	$X+jY = (X+jY) - (Z+jT)$	Saved in Z+jT

Register Use: (Min Size 020, DEG mode, F00-04 Clear)

Register#	Purpose	Register#	Purpose
R00	Plotter Ymax	R10	Plotter Xinc
R01	Plotter Ymin	R11	Plot "Name"
R02	Plot nnn.aaa	R12	Re(V)
R03	Plot character	R13	Im(V)
R04	Scratch Reg.	R14	Re(I)
R05	Plotter "Fix" N	R15	Im(I)
R06	Plotter Frequency	R16	RL
R07	Plotter "X Units"	R17	Index Symbol
R08	Frequency GHz	R18	Scratch Reg.
R09	Plotter Xmax	R19	Scratch Reg.

Example 1.- (From the Math Pac manual).



```

01 LBL "EX"
02 0
03 XEQ "RL"
04 1
05 XEQ "RS"
06 2
07 XEQ "RS"
08 3
09 XEQ "RP"
10 2
11 XEQ "RS"
12 7
13 XEQ "RP"
14 2
15 XEQ "RS"
16 1
17 XEQ "RS"
18 4
19 XEQ "RP"
20 2
21 XEQ "RS"
22 5
23 XEQ "RP"
24 2
25 XEQ "RS"
26 XEQ "VP"
27 END

```

Make sure to have the frequency $f = 50\text{Hz}$ stored as GHz in register 08:
 5 E-8, STO 08

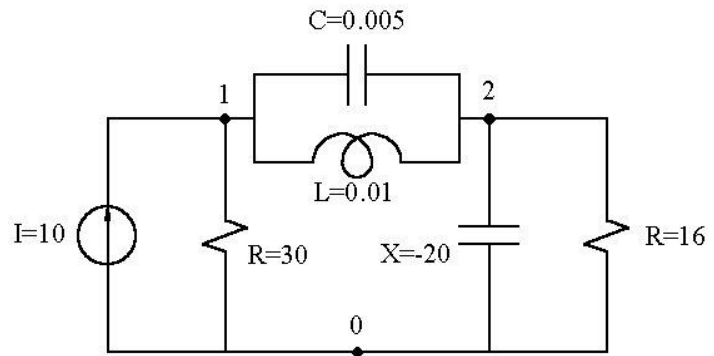
At the end the program will display: $V0 = 56.00 < 0.00$

Since the system is linear you can scale the voltage down to 7 V, the current I will also scale down by a factor of 8, so the actual I is 125 mA, not the 1 A you originally assumed.

Example 2.- (From the EEE program).

```

01 LBL "CI"
02 0
03 XEQ "RL"
04 16
05 XEQ "RS"
06 0
07 -20
08 XEQ "ZP"
09 1 E7
10 5 E9
11 XEQ "PLCS"
12 30
13 XEQ "RP"
14 XEQ "Z"
15 END
    
```



Make sure to have the frequency $f = 50\text{Hz}$ stored as GHz in register 08:
 5 E-8, STO 08

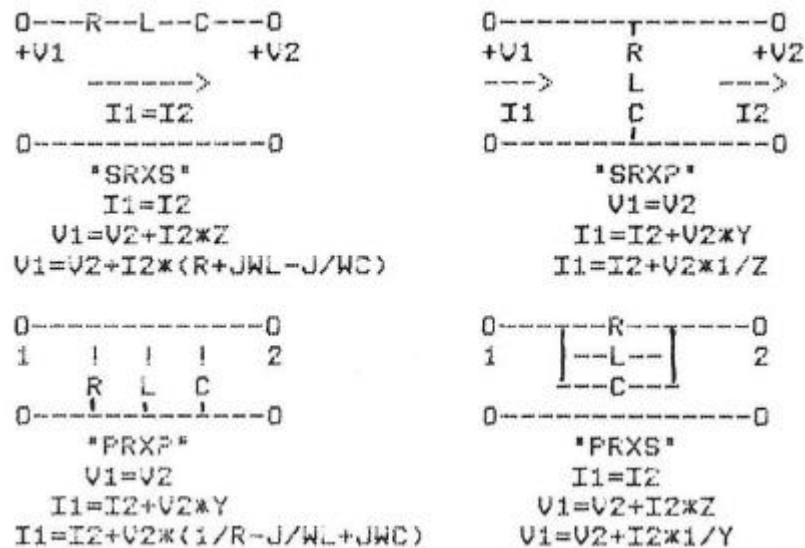
The result is: $Z0 = 9.59 < -29.20$

If we want to calculate V1 we have to multiply this by $I = 10$:
 10, *, P-R

This matches the result provided by “EEE”: $V1 = 83,7467 - 46,7976i$

Appendix.

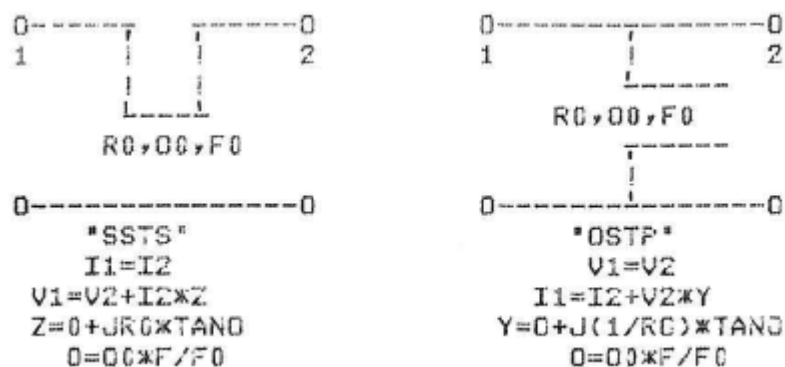
The derivation of some of the 'BLAP' element subroutines is presented in this section to aid the user in creating customized subroutines for his needs. Familiarity with circuit analysis and using the hp-41C are the only prerequisites.

Series – Parallel RLC's

Either the impedance of a series RLC or the admittance of a parallel RLC is computed. 'Z<>Y' transformation -if necessary- is performed using the 'I' command. Flag 04 is used to distinguish between series or parallel "RLC" connection within the element. F04 is set if the first letter of the 'name' is 'P' -. Flag 00 is used to distinguish between series or parallel usage of the element within the ladder network - F00 is set if the last letter of the element 'name' is 'P'. F01, F02, and F03 are set if respectively R, L, or C are present within the block. The flags are tested later to remove any unused elements from the general RLC element. If flags DC and 04 match, the element is used as is, otherwise the '1' command converts Z<>Y.

Stubs.

Transmission line stubs may also simply be represented as impedances or admittances. The shorted stub is represented as an impedance, while the open stub is represented as an admittance.



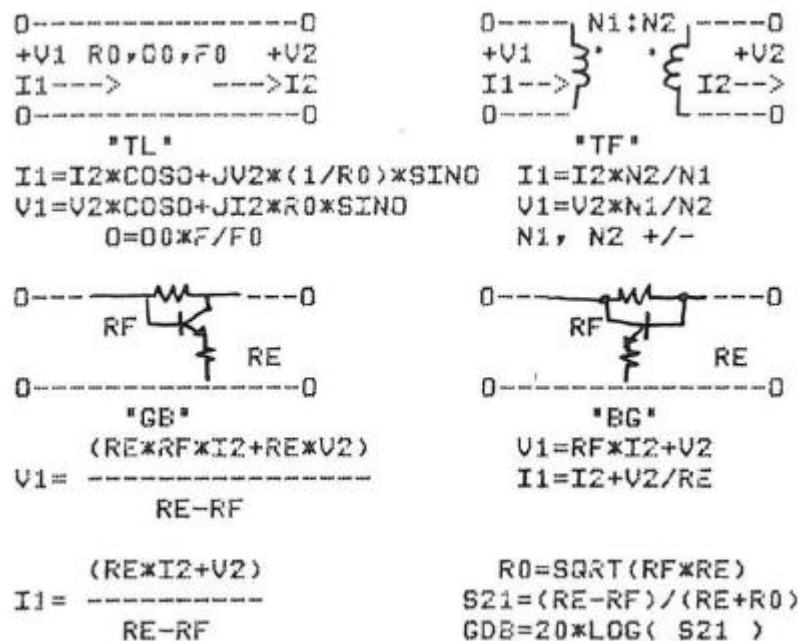
The only difference between the shorted stub impedance -Z- and the open stub admittance -Y- is R_0 versus $1/R_0$. Flag 04 is set for an open stub -first letter 'O' and flag 00 is set for parallel stub usage - last letter 'P'. Once the stub impedance or admittance is determined the element is handed within 'BLAP' just like a lumped impedance or admittance. Almost any passive filter may be analyzed using only the impedance/admittance elements described above. 'BLAP' does not provide all possible RLC combinations; however, and also the user might find it convenient and faster to create combinations of existing elements.

Two examples which might warrant their own commands are a quartz crystal -series RLC with a capacitor in parallel- and a real coil or resistor -series RL in parallel with a capacitor. These elements may be realized using normal 'BLAP' commands only if they are used as parallel elements.

Two-Port Elements.

All of the elements described so far have either identical voltage on either side -parallel element- or identical current on either side -series element. Many very useful elements modify both the current and voltage and must be considered as two-port networks - in fact the elements described above are special trivial cases of two-port networks -. The input/output relationship of a two-port can be described in many equivalent ways - Z, Y, G, H, S, ABCD depending upon the choice from I_1, V_1, I_2, V_2 of the pairs or independent and dependent variable pairs.

In "BLAP" I_2 and V_2 are the independent variables so we are really using "backwards" ABCD parameters. The four two-ports used in BLAP are presented below:



The two-ports included are all 'ideal' elements. The transmission line and transformer are lossless and the gain block is built using an ideal transistor such that the amplifier performance is solely determined by the feedback resistors. The gain block has 180 degrees' phase shift and is a perfect match in a R_0 ohm system.

'BLAP' actually ' designs' each gain block - computes R_F and r_e for specified DB gain and system resistance R_0 -. Simultaneous perfect match with the gain block is only possible if the generator and load impedances are equal. The 'GB ' may always be cascaded with a transformer or matching network to obtain any combination of source and load impedances.

The gain block is unconditionally stable since the input and output match is perfect in a R_0 ohm system and the reverse isolation is greater than the forward gain. Candidates for other two-port elements are limitless. Lossy transmission line, non-ideal transformers, and gain blocks having 'real' transistors are obvious examples.

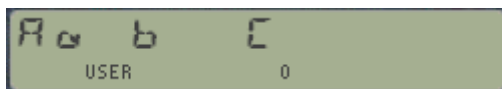
Electrical Circuits using the Advantage. [by C. Coffin and T. Wadman]*From “EE for Students”, Grapevine Editions – included in the ESTI5 module.*

Here’s one of the seminal applications using the Advantage module – The program is fundamentally unchanged but *it features an optimized U/I and enhanced data output routines* for additional convenience. The main attributes of the program are:

- It will solve for complex current, power and gain anywhere in a sinusoidal AC circuit of up to 12 nodes and 27 elements (CX plus two X-Mem modules)
- It allows easy storage and recall of these and other complex calculations, including parallel-to-series and Why-to-Delta conversions; complex arithmetic and polar-rectangular equivalents.
- Allows you to save, expand or alter the circuit analyzed, including the frequency
- Accepts circuit elements V and I (complex sources) and impedances R, L, C, either in units of ohms, henrys and farads – or combined as complex ohms (either in rectangular or polar form)

Cold-Starting the Program.

Executing “ADV-Z” initializes the data structures and resets the values to zero state. It then presents the main menu as follows:

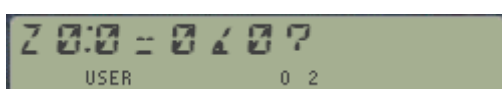


Which indicates the available options to the user showing the letters:

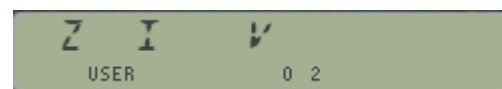
- [A], to Alter the program
- [] [a], to add elements
- [] [b], to build the circuit
- [C], to Calculate results
-
- [D], to enter values in rectangular mode (hidden key)
- [] [d], to enter values in polar mode (hidden key)
- [E], to change to the Equations menu (hidden key)
- [J], to Jump to previous menu (hidden key)

Building the Circuit.

The first three prompts allow you to define the number of nodes, number of elements, and the frequency. Simply answer the values and press [R/S]. Next you need to provide the circuit topology and element data. At each element prompt you’re required to enter its complex value (imaginary and real parts, or argument and module) as well as its “from-to” nodes location. Then pressing [R/S] the element type selection uses a secondary prompt with the three available options:



and:

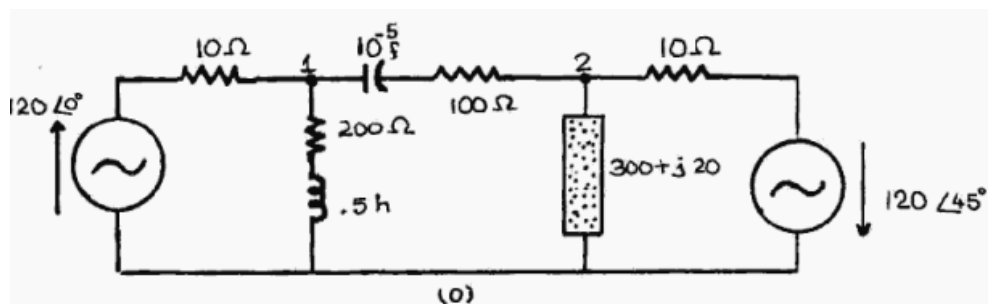


If the element is not the last one in the branch, then the “from-to” location uses a negative sign. Capacitances use a negative value whilst inductances are entered as positive values – this tell them apart from one another for the program to treat them appropriately.

The element data can be introduced in one of three formats: rectangular form, polar form, or “physical” form – which uses the combination of resistors and coils or capacitors directly. The input mode is selected by user flags F0 and F1, toggled using the options [D] and [d] respectively in the main menu. The correspondence between flags and modes is shown below:

Mode	Data format	Flag_00	Flag_01
Rectangular mode	Imag, ENTER^, Real	Clear	Set
Polar mode	Arg, ENTER^, Module	Set	Clear
“Physical” mode	L/C, ENTER^, R	Clear	Clear

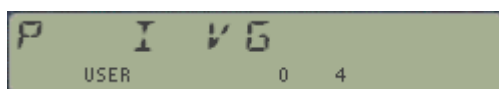
Let’s use the circuit shown below as example for the data entry process, with two nodes (besides the reference) and seven elements. The elements will be introduced sequentially, as follows:



e1: 0, ENTER^, 120, -0.01, [R/S];	Voltage	Polar
e2: 0, ENTER^, 20, 0.01, [R/S];	Impedance	Polar or Rect.
e3: 0.5, ENTER^, 200, 1, [R/S];	Impedance	Physical
e4: -1E-5, ENTER^, 100, 1.02, [R/S]	Impedance	Physical
e5: 20, ENTER^, 300, 2, [R/S]	Impedance	Rectangular
e6: 0, ENTER^, 20, -2, [R/S]	Impedance	Rectangular
e7: 45, ENTER^120, 2, [R/S]	Voltage	Polar

After all elements have been introduced the main menu is displayed again – offering to do alterations (pressing [a]), general review (pressing [A]), or “Commit” to the matrices in memory (pressing [C]). This step is only performed once prior to the calculations of the results.

Once the circuit data has been committed to the matrix structure in memory, the program will present the results options, either Power, Voltages or Currents at a certain node or branch – always using the same references used to enter the circuit topology in the first place.



The results are shown in the table below:- Note that after each result the program pointer is within the “Equations” section, thus we need to press [J] to return to the calculation menu.

Node 1 Voltage (to ground)	116.964 - 2.16 USER 0 4
Node 2 Voltage (to ground)	111.074 - 137.46 USER 0 4
Voltage between nodes 1 and 2	181.094 23.06 USER 0 4
Current 0-1 Thru resistor	1.974 13.07 USER 0 4
Current 1-0 Thru coil and resistor	0.434 - 45.46 USER 0 4
Current 2-0 Thru complex impedance	0.374 - 141.27 USER 0 4
Current 1-2 Thru capacitor and resistor	1.814 23.06 USER 0 4

Warm-Starting the Program and Showing Element Values.

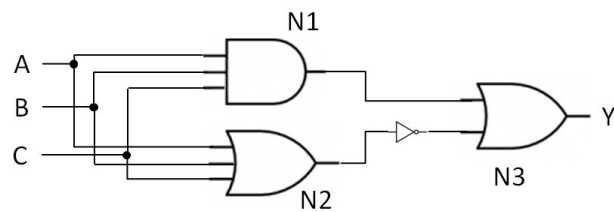
Executing “ADV-R” will start the program without resetting the stored values in memory, i.e. you can use this to resume a previous analysis or to modify / add / remove components and re-run the calculations.

Finally, the auxiliary routine “YZ-A” is used internally by the program to display the element values as a complex number, as well as showing the “from-to” configurations. It will properly interpret the element type (V, I, Z) and whether it’s not the last element in a branch (comma character added to the index).

Interpretation of Logical Networks. [by C. Sture Sjöström]

From the User's Program Library Europe #10574; included in the ETSII5 module.

With this program you may define a logical (digital) network, interpret the definition and calculate (execute) the output signal for a set of input signals to the network. The digital circuit may have AND, OR, XOR and INV gates. It is defined in ALPHA as a RPN-like sequence and can be executed at any time.

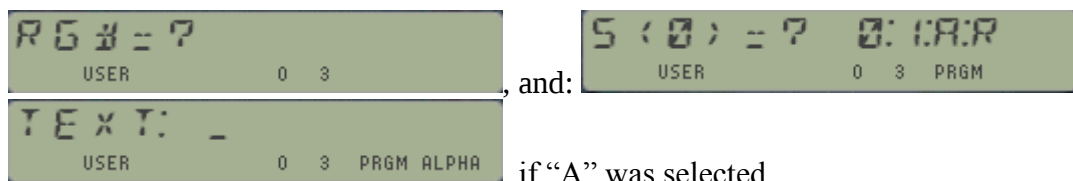


The Network is defined in 10 inputs or “bits” {S0 – S9} - each of them may use two data registers for a total of 20 reserved. Each input may consist of either a constant signal (zero or one) or a description of part of the network. The descriptions may include combinations of the symbols below, always following a number (from 0 to 9) representing the *logical register number* upon which apply the operator.

A: AND
 O: OR
 X: XOR (exclusive OR)
 I: INV

For instance, “34A5O” means “OR between R5 and “AND between R3 and R4”. Each logical input can hold up to 12 characters (split in two data registers with 6 characters each). The interpretation will always start using the logical input S0, involving also others when the description exceeds that 12-char limit – but data entry doesn’t need to follow a sequential register order.

The calculator prompts first for the logical input number (bits 0 to 9), followed by an input value prompt: either 0/1 for constant bit values, “A” for an Alpha string, or “R” to review its value.



This process will continue until you press [R/S] at the “S#” prompt, which indicates that no more inputs are to be used. Note that the data entry can be made in any sequence order. At this point the calculation of the output signal will be made, showing the final result in the display. You can modify the signal values pressing [D], which will invoke the input prompt.

Error Messages.

This program uses functions from the AMC_OS/X module, which therefore needs to be plugged in the calculator or otherwise you’ll get the ‘NONEXISTENT’ message. The program

uses data registers R23 and up as scratch for intermediate calculations. If the SIZE is not large enough the overflow message “OVERFLOW” will be shown and you’ll need to increase the calculator size to continue. Other error messages may be “SYNTAX ERR” if the operator symbols are used improperly, or “ILL. CHR: x” when non-allowed characters are found (i.e. not on the list above).

Saving and restoring Network to/from X-Memory.

You can save the logical network in an X-Memory file using the program option under **[F][a]**. This creates a data file named “LOGIC” with the contents of the first 20 data registers. Note that this will be done automatically every time you perform the calculation of the result. To restore the logical network you can use **[F][b]**. Then pressing **[B]** will calculate the result signal again.

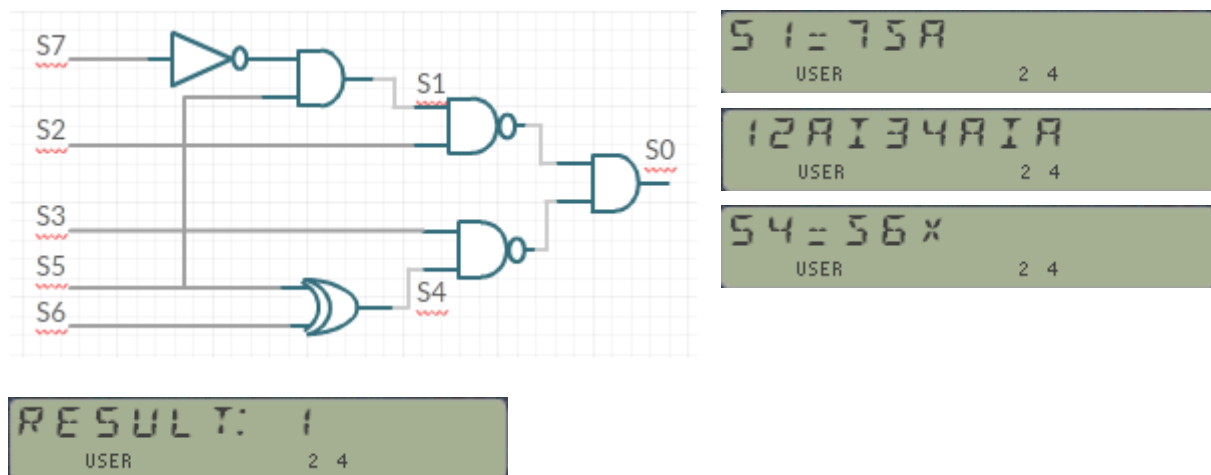
Examples.

The three networks a, b, c below are to be examined. Note that the second is an expansion of the first one. The constant inputs are included in the descriptions below. We also want to know what happens if the S6 value in the third network is changed to a “1”.

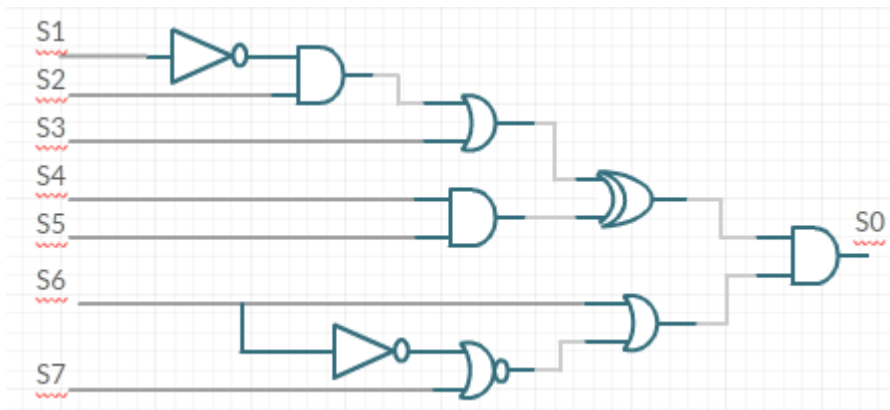
For (a): S1=0, S2 = 1, S3=1, S4=1



For (b): S2=1, S3=1, S5=1, S6=1, S7=0



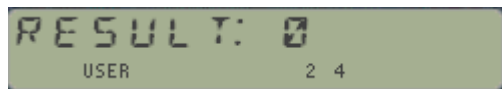
For (c): S1=0, S2=1, S3=1, S4=0, S5=0, S6=0, S7=1



Written as a single statement, the output signal value can be split into the upper part (stored in S8) and the bottom part (stored in S0), as follows:



The other constant values can easily be entered using the same procedure shown in the previous examples. If you do that the final result will be:



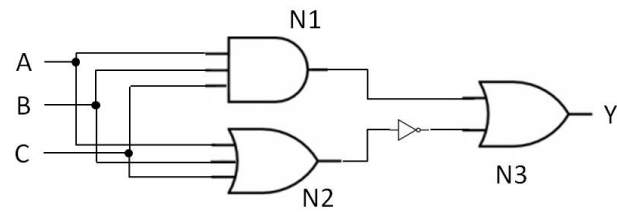
Modifying the value for S6 is very simple: just press [D] and enter “6” at the register number prompt, followed by the new constant value $s6 = 1$

The final result changes to a TRUE one after this bit value change.

Truth Table of Logical Networks. [TRUTH]

From the author's Engineering Collection, included in the ETSII5 module.

This program calculates the complete truth table of a logical network with a generic number of bits. The Network is defined as a user-provided function calling the individual gates subroutines as building blocks – whereby it is assumed the input bits are in the stack registers X and Y. The bit values will be stored in registers R06 and up, thus the minimum size required is $N = 6 + \#b + 1$.



The available gates are shown in the table below:

AND	OR	OREX	NOT
NAND	NOR	AOI = AND-OR-Invert	

The program will sweep the complete bit range, sequentially assigning one's and zero's to their values. For each iteration the calculator will show the result value and the input bits. For example, for a 4-bit input there'll be 16 of those results, from "0.0.0.0" to "1.1.1.1." such as the ones shown below corresponding to "LN1" – Logical network-1:



Having a peripheral printer is the most efficient way to use this program, as without it you'll need to copy the results by hand – not too bad if all you need is a few specific results for fixed input bit values. Set user flag 21 if you need the execution to halt after each display.

Example.

Build the Truth table for the logical network shown above, with three input bits and programmed as follows:

```

01 LBL "LN1"
01 RCL 06
02 RCL 07
03 XROM "AND"
04 RCL 08
05 XROM "AND"
06 STO 10
07 RCL 06
08 RCL 07
09 XROM "OR"
10 RCL 08
11 XROM "OR"
12 XROM "NOT"
13 RCL 10
14 XROM "OR"
15 END

```

A	B	C	Result Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Decibel addition/subtraction. [dB+, dB-]

From the author's Engineering Collection, included in the ETSII6 module.

These two miniature MCODE functions come very handy to calculate decibel addition and subtraction, so you don't have to take anti-logarithms and care about the needed conversions.

The formulas used are the usual suspects:

$$\text{dB}(x) = 10 \log x$$

$$x = 0.1 * \text{alog} [\text{dB}(x)]$$

Examples.

Add 8 and 5 dB. Subtract 3 dB from the result.

The final result is 8.7370 dB.

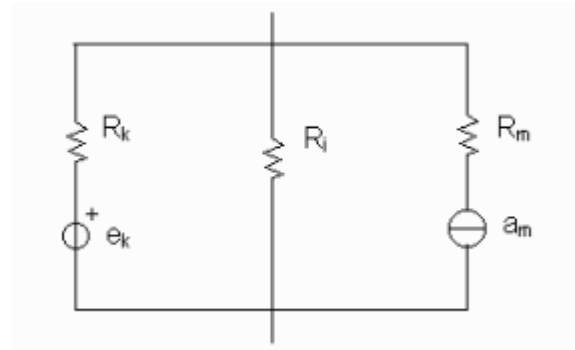
Header	AFA5	0AB	"+"		
Header	AFA6	002	"B"		Decibel addition
Header	AFA7	044	"d"		
dB+	AFA8	244	CLRF 9		Ángel Martin
	AFA9	02B	JNC +05		[MAIN]
Header	AFAA	0AD	"-"		
Header	AFAB	002	"B"		Decibel subtraction
Header	AFAC	044	"d"		
dB-	AFAD	248	SETF 9		Ángel Martin
MAIN	AFAE	0F8	READ 3(X)		
	AFAF	361	?NC XQ		(includes SETDEC)
	AFB0	050	->14D8		[CHK_NO_S]
	AFB1	266	C=C-1 S&X		/10
	AFB2	070	N=C ALL		
	AFB3	044	CLRF 4		
	AFB4	3E1	?NC XQ		
	AFB5	06C	->1BF8		[10TOX]
	AFB6	24C	?FSET 9		
	AFB7	01B	JNC +03		[PLUS]
MINUS	AFB8	2BE	C=C-1 MS		sign change
	AFB9	11E	A=C MS		ditto for 13-digit form
PLUS	AFBA	089	?NC XQ		
	AFBB	064	->1922		[STSCR]
	AFBC	0B8	READ 2(Y)		
	AFBD	361	?NC XQ		(includes SETDEC)
	AFBE	050	->14D8		[CHK_NO_S]
	AFBF	266	C=C-1 S&X		/10
	AFC0	070	N=C ALL		
	AFC1	044	CLRF 4		
	AFC2	3E1	?NC XQ		
	AFC3	06C	->1BF8		[10TOX]
	AFC4	0D1	?NC XQ		
	AFC5	064	->1934		[RCSCR]
	AFC6	031	?NC XQ		
	AFC7	060	->180C		[AD2-13]
	AFC8	01E	A=0 MS		absolute value
	AFC9	088	SETF 5		Decimal Log
	AFCA	121	?NC XQ		
	AFCB	06C	->1B48		[LN13]
	AFC0	226	C=C+1 S&X		x10
	AFCD	000	NOP		let carry settle
	AFCE	369	?NC GO		Final checks & Exit
	AFCF	002	->00DA		[NFRXY]

Millman's equivalence Theorem. [MILLMN]

From the Author's Engineering Collection, included in the ETSII6 module.

This short program calculates the voltage at the ends of a circuit made up of only branches in parallel using the Millman's theorem expression, whereby:

$$v = \frac{\sum \frac{\pm e_k}{R_k} + \sum \pm a_m}{\sum \frac{1}{R_k} + \sum \frac{1}{R_i}}$$



Where **R_i** are the resistences on branches without generator, **R_k** the resistences in branches with generators, **e_k** the voltage generators in those branches, and **a_m** the current sources (which cannot be in series with e_k).

The programs first prompts for the total number of branches, and then proceeds into a loop for individual branch characterization. For each branch the resistance and voltages values are always prompted for. If there are no voltage generators (*signaled by your inputting zero*), then another prompt asks for the current source: enter its value if any, *or zero if none*.



When all branches have been entered the program shows the voltage at the ends. Note that the resistences included in branches with current sources are not used in the formula. You should always follow a consistent sign convention for all voltage generators and current sources.

Example.

Calculate the voltage at ends of a 3-branch circuit (like that shown above) with the following element values:

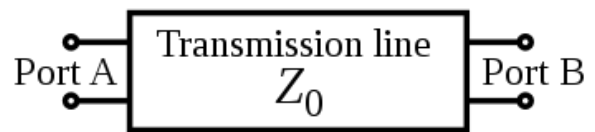
$$\begin{array}{lll} R_1 = 10 \, \Omega ; & R_2 = 20 \, \Omega ; & R_3 = 0.2 \, \text{k}\Omega \\ e_1 = 5 \, \text{V} ; & & a_3 = 2 \, \text{A} \end{array}$$

The result is $V = 16.129$ Volts

Transmission Line Impedances. [by Carter P. Buck]

From PPCCJ V9N7 p16, included in the ETSII6 module

This routine provides a solution for source and load impedances in a transmission line – given a known line impedance Z , the characteristic impedance Z_0 , and the electrical length B_s – in units consistent with the trigonometric mode set.



The electrical length is usually calculated as product of the physical length by the frequency, divided by the speed of light in the medium: $B_s = L \cdot f / s$

The formulas used are as follows:

$$Z_s = Z_0 [(Z_L + j Z_0 \tan B_s) / (Z_0 + j Z_L \tan B_s)] ;$$

$$Z_L = Z_0 [(Z_s - j Z_0 \tan B_s) / (Z_0 - j Z_s \tan B_s)]$$

The routine expects the data loaded in the stack as per the table below. Note that this assumes the characteristic impedance to be just a resistance, which is the most common case.

T:	B_s	Z:	Z_0	Y:	$\text{Im}(Z)$	X:	$\text{Re}(Z)$
----	-------	----	-------	----	----------------	----	----------------

Example 1.

A 65 cm length of 75 Ω line in air operates at a frequency of 465 MHz and terminates at a 100-j50 Ω load. Find the impedance seen from the source assuming lossless conditions.

Solution: Since the media is air the velocity is the speed of light, 30 cm/ns. Therefore, the electrical length is: $B_s = 65 \cdot 465 \text{ E}6 / 30 \text{ E}9 = 1.0075$ wavelengths, or 2.7 degrees. Thus we enter the following data in the stack:

```
DEG, 2.7, ENTER^, 75, ENTER^, 50, CHS, ENTER^, 100
XEQ "ZS"    ->    ZS = 93.86 - j 50.77  $\Omega$ 
```

Example 2.

A 30 cm length of 50 Ω polyethylene line is connected to a 400 MHz generator with an impedance of 75 Ω . Find the impedance of the balanced load.

Solution: The velocity in polyethylene is 2/3 the speed of light, or 20 cm/ns. The electrical length is therefore 0.6 wavelengths, or 1.2 π rad. Thus the data entry sequence is now:

```
RAD, 1.2, ENTER^, PI, *, 50, ENTER^, 0, ENTER^, 75
XEQ "ZL"    ->    ZL = 52.38 - j 20.76  $\Omega$ 
```

Network Frequency Response Analysis. [by Michael Moser]

From the User's Program Library Europe #I0786, included in the EE Filters module.

This program computes the frequency response on a desired interval of a general linear network made up of resistors, capacitors, inductors and voltage-controlled dependent current sources. You define the circuit by keying in the number of nodes (i.e. the order), the types and values of the components, and the nodes they're connected to.

The program consists of three blocks: (1) the input stage, (2) the construction of the admittance matrix, and (3) a matrix reduction stage. Then successive sweeps will loop showing the different electrical outputs with each frequency value.

Let $\omega = 2\pi f$, with f the work frequency. The elements of the admittance matrix are formed as follows for the different components:

Resistors : $Y(R) = 1/R$ with R in ohms
 Capacitors: $Y(C) = j\omega C$, with C in farads
 Inductors: $Y(L) = -j/\omega L$, with L in henries
 Voltg-Cntl'd: $Y(VCS) = g_m$, the transconductance

The following conventions are to be observed:

- Node "0" is the reference or ground node; node "1" is the input, and node "2" is the output. A 1-volt reference source is connected between ground and the input node. the phase output is always between +/-180 deg
- For R, L, C components the "from" node cannot be the ground.
- Problems may occur with inductors in networks at near-zero frequencies. For DC analysis you should redesign the network, shorting all inductors and specify the starting frequency to be zero.
- *Independent current sources are not supported.* All voltage-controlled current sources (VCS) are specified so that the voltage is measured between the node and ground, and the current leaves ground and enters the "To:" node.

Once the complex matrix admittance (NMA) is formed, the following matrix equations can be used to describe the steady-state performance of a network at a given frequency:

$[Y]_{n \times n} [V]_{n \times 1} = [I]_{n \times 1}$, and solving for the voltages:

$$[V]_{n \times 1} = [Y]_{n \times n}^{-1} [I]_{n \times 1}$$

Since for this program is only necessary to find one mode voltage, the complex matrix inversion method is replaced in favor of a faster and simpler matrix-reduction approach. First a partition is made, whereby the NAM is expressed as follows:

$$[Y]_{n \times n} = \begin{array}{c|c} [Y_{11}] & [Y_{12}] \\ \hline [Y_{21}] & [Y_{nn}] \end{array}$$

Y_{11} is a square submatrix $(n-1) \times (n-1)$
 Y_{12} is a column vector $(n-1) \times 1$
 Y_{21} is a row vector for $(n-1)$ -th. element
 Y_{nn} is the selected element of the NAM

This is followed by a order reduction of $[Y]_{n \times n}$ to $[Y^{(1)}]_{(n-1) \times (n-1)}$ as follows:

$$[Y^{(1)}]_{(n-1) \times (n-1)} = [Y_{11}] - [Y_{12}][Y_{21}] [Y_{nn}]^{-1}$$

The order reduction step is repeated n-2 times, until we obtain a 2x2 matrix:

$$[Y^{(n-2)}]_{2 \times 2} = \begin{array}{c|c} [Y'_{11}] & [Y'_{12}] \\ \hline [Y'_{21}] & [Y'_{nn}] \end{array}$$

The required transfer function is then given by the expression:
 $V_2/V_1 = -[Y'_{21}] / [Y'_{22}]$

Which is expressed as a magnitude in dB = 20 log |V₂/V₁|.

Program details.

The input routines offer a very convenient method to enter and change the component parameters. The modifications made use functions PMTK and ARCLI from the AMC_OS/X module, which therefore needs to be plugged in as well. A few program options include:

- Press [H] for a new circuit
- Press [J] to change the increment mode from Linear to Logarithmic
- Press [G] to change the increment step or the number of points per decade
- Press [F] to change the start frequency (Fa) or end frequency (Fe)
- Press [E] to change a component value (error corrections or design changes)

The prompts will request to enter the element type first, followed by the “from-to” node configuration and element value. Voltage-controlled sources differ from that scheme in that they require the dependent voltage instead.

TYPE? RLCVX
USER 4 PRGM

FROM = ?
USER

TO = ?
USER

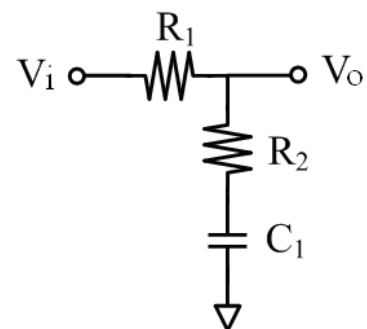
Example1.

For the passive RC-filter shown below, determine the frequency response from 1MHz to 15MHz at 2Mhz increments. The component parameter values are:

$$R_1 = 50 \, \Omega; \quad C = 1 \, nF; \quad R_2 = 20 \, \Omega$$

The analytical expression for the transfer function is:

$$V_{out} / V_{in} = [1 + j \omega C_1 R_2] / [1 + j \omega C_1 (R_1 + R_2)]$$

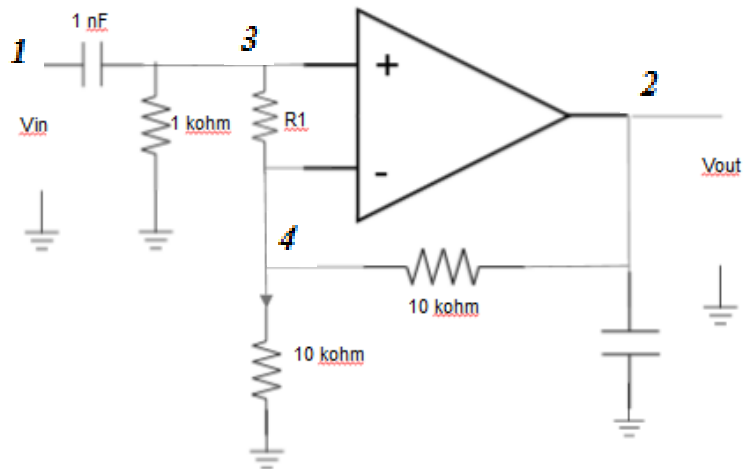


The results are shown in the table below:

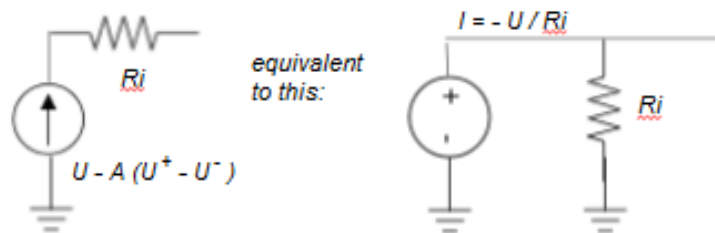
f (Hz)	IGI (dB)	<)G (deg)	f (Hz)	IGI (dB)	<)G (deg)
1 E6	-0.700	-16.5785	9 E6	-8.641	-27.305
3 E6	-3.802	-32.186	11 E6	-9.235	-24.205
5 E6	-6.216	-33.4055	13 E6	-9.630	-21.552
7 E6	-7.714	-30.670	15 E6	-9.903	-19.328

Example2.

Compute the frequency response of the active high-pass filter shown below from 10 Hz to 10 KHz with 3 points per decade. The values of $R1 = 10 \text{ k}\Omega$; $C2 = 50 \mu\text{F}$; $A = 10 \text{ E}6$



There is a problem concerning the Op-Amp in this circuit. An ideal Op-Amp is defined as a voltage-controlled *voltage source*, and this type of source is not supported by the NAM, node admittance model, which can only handle voltage-controlled *current sources*. But it's possible to transform the Op-Amp into that type adding an inner resistance in parallel, R_i . Therefore the Op-Amp can be replaced by the following circuit, where $I = A.(U^+ - U^-) / R_i$



Thus the OpAmp is replaced by two current sources (each controlled by the input voltages of the OpAmp), plus the inner resistance - *all connected between ground and node #2*:

$$I_1 = -A U^{(+)} / R_i ; \quad \text{and:} \quad I_2 = A.U^{(-)} / R_i$$

In closed-loop circuits like this one the value of this resistor is not critical; it just has to be small enough to not have noticeable influence. In this example we've chosen an inner resistance of 0.1Ω – which is very small, compared to the other resistors in the circuit. You can also use an inner resistance of 0.01Ω to check the results, which are practically the same.

The results are shown in the table below:

f (Hz)	IGI (dB)	<G (deg)	f (Hz)	IGI (dB)	<G (deg)
10.000	-18.033	86.405	464.16	5.538	18.926
21.544	-11.428	82.291	1.0000 E3	5.912	9.043
46.416	-5.037	73.741	2.1544 E3	5.997	4.225
100.00	0.539	57.858	4.6416 E3	6.016	1.964
215.44	4.129	36.454	10.000 E3	6.020	0.912

Transfer Function Parameters. [GS1, GS2]

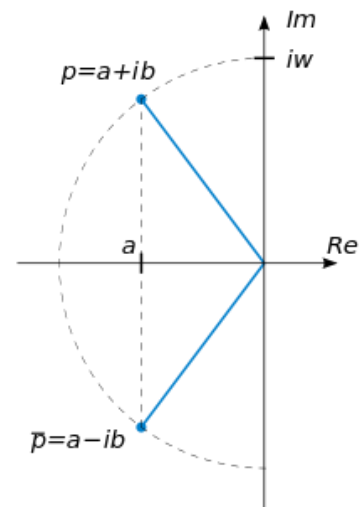
From the Author's Engineering Collection, included in the ETSII6 module.

This program calculates the representative parameters from the transfer function expression of system of first and second order. The input data for the program are coefficients of the transfer function, and the output results are alternative factors used in the description of the systems.

For first order systems the general expression of the transfer function is: $G(s) = b_0 / (a_0 + s)$

Which can be re-written as: $H(s) = k / (1 + s/\tau)$

The program calculates the dampening, the time factor; and the location of the system pole ($S_p = -a_0$)



For 2nd. Order systems, the general expression of the transfer function is:

$$H(s) = \frac{b_0}{s^2 + a_1 s + a_0}$$

which can be re-written as as a function of the natural frequency, dampening and oscillation factors. The growth factor, estabilization time, end time, and maximum value time (if under-damped) are also calculated.

$$\begin{aligned} \zeta &= a_1 / 2.\omega_n; & \sigma &= \zeta.\omega_n; & \omega_p &= \omega_n \sqrt{1 - \zeta^2} \\ t_{max} &= \pi/\omega_p; & t_e &= \pi/\sigma; & t_s &= (1+\zeta)/\omega_n \\ \delta &= \exp(-\sigma.\pi/\omega_p); & \text{poles: } S_{1,2} &= -\sigma \pm j \omega_p \end{aligned}$$

Example.

Determine the characteristic factors for a 1st. order system and a 2nd. order system with the following transfer functions: $G1(s) = 2/(s + 3)$; and: $G2(s) = 2 / (s^2 + 3.s + 2)$

The results are shown in the tables below:

System	$k = b_0/a_0$	$\tau = 1/a_0$
GS1	0.6667	0.3333

Real Pole at: $s = -3.0000$

System	$k = b_0/a_0$	$\omega_n = \sqrt{a_0}$	F.DUMP	F.GROWTH	T_e	T_s
GS2	1.0000	1.4142	1.0607	1.500	2.0944	1.4571

This system is over-damped; therefore $\omega_p = 0$, and no time of maximum is calculated. Real Poles located at: $s_1 = -1.000$; $s_2 = -2.000$

Moments of Inertia. [MOI, MOI+]

From Jean-Marc Baillard's web site, included in the ETSII6 module

This program calculates the moments of inertia of a system of n points $M_k (x_k, y_k, z_k)$ with masses m_k respectively

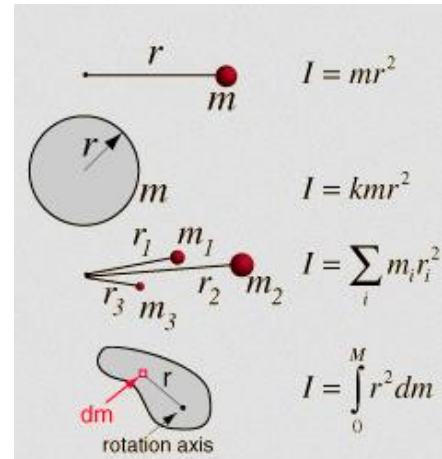
$$I_x = \sum m_k (y_k^2 + z_k^2)$$

$$I_y = \sum m_k (x_k^2 + z_k^2)$$

$$I_z = \sum m_k (x_k^2 + y_k^2)$$

and:

$$I_o = \sum m_k (x_k^2 + y_k^2 + z_k^2) = (I_x + I_y + I_z) / 2$$



The Driver program MOI+ will first prompt for the number of points, “n”; followed by sequential prompts for the point coordinates and the masses values (entered in the stack separated by ENTER^), storing them in a contiguous set of data registers starting at R01. The data entry process completes with the control word 1,00(4n) in X and the execution is transferred to the main routine MOI.



In manual mode (or subroutines) the data is expected in the data registers, and the control works bbb.eee in X. On completion the results are left in the stack, arranged as follows:

STACK	INPUTS	OUTPUTS
T	/	Io
Z	/	Iz
Y	/	Iy
X	bbb.eee	Ix

Example.

With the 5 points:	xk	1	-4	2	-3	5
	yk	2	7	-8	-1	3
	zk	3	1	-2	-4	-7
	mk	7	8	4	5	2

Store, for instance, these 20 numbers into R01 thru R20 (in column order)

Then 1.020 XEQ "MOI" >>>> Ix = 964 ---Execution time = 6s---

RDN Iy = 511

RDN Iz = 945

RDN Io = 1210