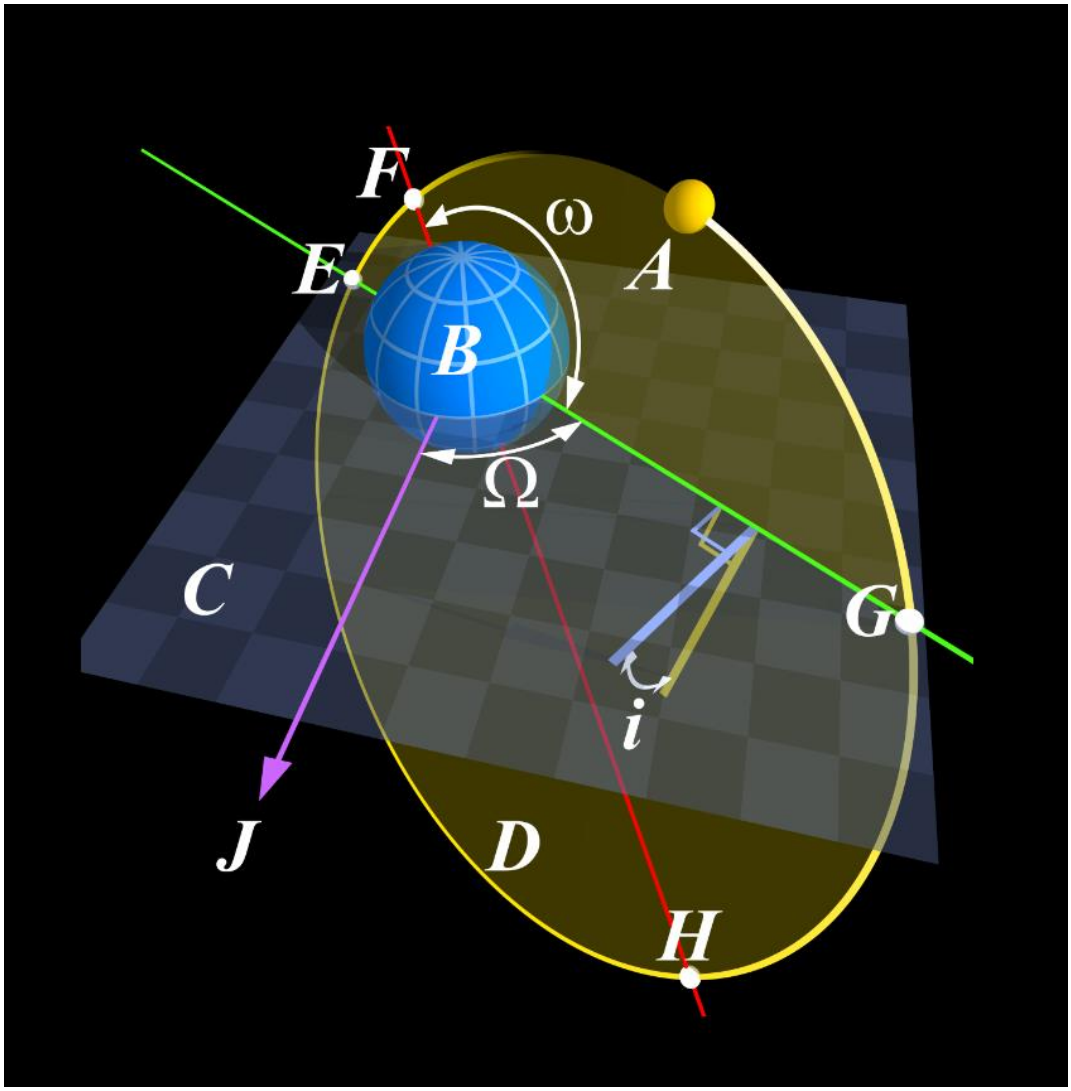
ELLIPTIC APPLICATIONS ROM & ORBITAL MECHANICS

HP-41 Module



Programmed by Ángel M. Martin
April 2018

This compilation revision 1.4.1

Thou shall know well your equations:

$$sin(\alpha+\beta)=sin(\alpha)cos(\beta)+cos(\alpha)sin(\beta)$$

$$sin(\alpha-\beta)=sin(\alpha)cos(\beta)-cos(\alpha)sin(\beta)$$

$$cos(\alpha+\beta)=cos(\alpha)cos(\beta)-sin(\alpha)sin(\beta)$$

$$cos(\alpha-\beta)=cos(\alpha)cos(\beta)+sin(\alpha)sin(\beta)$$

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

Acknowledgments.- Thanks to Jean-Marc Baillard and Poul Kaarup for their contributions to this compilation.

ELLIPTIC APPLICATIONS ROM

Table of Contents

Introduction.Elliptic Applications Module

This HP-41 Module includes a collection of MCODE functions and short driver FOCAL programs on the subject of 'Elliptic Applications". Despite the section labelled "Orbital Mechanics", this is not for the most part an astronomy related module – as the subjects are mostly approached from a geometry point of view.

It could be said that the module was prepared while learning about the subjects, as an instrument to verify assumptions and experiment. Therefore, don't expect to find advanced treatment of the subjects. Yet it's possible that you'll enjoy the way the material is put together, sometimes an un-biased approach is a more refreshing way to proceed.

Without further ado, here's the function index for your reference:

XROM	Function	Description	Input / Output	Author
16,00	**-ELLIP APPS**	Section Header	n/a	*n/a*
16,01	**BRHM**	Area of Cyclic Quadrilateral	Data in R01-R04	*JM Baillard*
16,02	**CIRCLE**	Circle through three points	P1, P2, P3 in Registers R01-R06	*Ángel Martin*
16,03	***"ECS+"***	Driver for ECS	Prompts for values	*Poul Kaarup*
16,04	**ECS**	Elliptical Central Sector	{a,b,e} in R00-R02, α in X	*Ángel Martin*
16,05	**EFS-**	Ellliptical Left Focal Sector	{a,b,e} in R00-R02, α,β in R03-04	*Ángel Martin*
16,06	***"EFS+"***	Elliptical Right Focal Sector	{a,b,e} in R00-R02, α,β in R03-04	*Ángel Martin*
16,07	**HERON**	Area of Triangle from sides	Data in X,Y,Z	*Ángel Martin*
16,08	***"K2-"***	Kepler 2nd. Law – Left Sector	Prompts for values	*Ángel Martin*
16,09	***"K2+"***	Kepler 2nd. Law – Right Sector	Prompts for values	*Ángel Martin*
16,10	***"LCS"***	Central sector Arc Length	Prompts for values	*Ángel Martin*
16,11	***"LF+"***	Right Focal sector Arc Length	Prompts for values	*Ángel Martin*
16,12	**RAMA2**	Ramanujan's 2nd. Approx.	a,b, in Y,X	*Ángel Martin*
16,13	**TOF**	Time of Flight	(T, e, θ) in { Z,Y,X}	*Angel Martin*
16,14	**ZELIP1**	Complex Elliptic Intg. 1st. kind	z in (Z,Y), m in X	*Ángel Martin*
16,15	**ZELIP2**	Complex Elliptic Intg. 2nd. kind	z in (Z,Y), m in X	*Ángel Martin*
16,16	**-ORBITS 101**	Section Header	n/a	*n/a*
16,17	**E>M**	Eccentric to Mean anomaly	(e, E) in Y,X	*Ángel Martin*
16,18	**E>T**	Eccentric to True Anomaly	(e, E) in Y, X	*Ángel Martin*
16,19	**M>E**	Mean to Eccentric anomaly	(e, M) in Y,X	*JM Baillard*
16,20	**T>E**	True to Eccentric Anomaly	(e, T) in Y,X	*Ángel Martin*
16,21	***"ORBIT"***	Delta-V Orbit Simulator	Under program control	*hp Co.*
16,22	***"TA<T>"***	True Anomaly via Kepler	New Position w/Kepler equation	*Ángel Martin*
16.23	***"TA+"***	True Anomaly Direct Mode (+)	New Position using Right sectors	*Ángel Martin*
16.24	***"TA-"***	True Anomaly Direct Mode (-)	New Position using Left sectors	*Ángel Martin*
16.25	***"<)T"***	Subroutine for Solve	Under program control	*Ángel Martin*
16.26	***"V<R>"***	Velocity "Vis Vivas"	Velocity at distance – Left focus	*Ángel Martin*
16.27	***"VR+"***	Velocity at Position R+	Velocity at distance - right focus	*Ángel Martin*
16.28	***"VR-"***	Velocity at Position R-	Velocity at distance - Left focus	*Ángel Martin*
16.29	**T(μ)**	Period from Gravitational data	Mass in Y, semi-major axis in X	*Ángel Martin*
16.30	**μ1/2**	Gravitational Parameter	Period in Y, semi-major in X	*Ángel Martin*

XROM	Function	Description	Input / Output	Author
16.31	**-AUX FNS**	Section Header	Doubles as QROUT	*n/a*
16.32	**<)C-R**	Central angle to Radius	{a,b,e} in R00-R02, α in X	*Ángel Martin*
16.33	**<)C-XY**	Central angle to coordinates	{a,b,e} in R00-R02, α in X	*Ángel Martin*
16.34	**<)C0+**	Central angle for Focal 90°	(a b,e) in R00-R02. Lifts stack	*Ángel Martin*
16.35	**<)C>F**	Central to Focal angle	{a,b,e} in R00-R02, α in X	*Ángel Martin*
16.36	**<)F>C**	Focal to Central angle	{a,b,e} in R00-R02, α in X	*Ángel Martin*
16.37	**<)F0+**	Right focal angle at x=0	(a,b,e) in R00-R02. Lifts stack	*Ángel Martin*
16.38	**<)F0-**	Left Focal angle at x=0	(a,b,e) in R00-R02. Lifts stack	*Ángel Martin*
16.39	**<)F>R-**	Left Focal angle to radius	{a,b,e} in R00-R02, α in X	*Ángel Martin*
16,40	**<)F>R+**	Right focal angle to radius	{a,b,e} in R00-R02, α in X	*Ángel Martin*
16,41	***"?ab"***	Prompts for semi-axis	New value or R/S for current	*Ángel Martin*
16,42	***"?<)-"***	Prompts for left-angles	New value or R/S for current	*Ángel Martin*
16,43	***"?<)+"***	Prompts for right-angles	New value or R/S for current	*Ángel Martin*
16,44	***"?P"***	Prompts for Period	New value or R/S for current	*Ángel Martin*
15,45	**QROOT**	Quadratic Eq. Roots	c2, c1, c0 in Z,Y,X	*Ángel Martin*
15.46	**V(X)**	Velocity at abscissa X	Period in Y, abccissa in X	*Ángel Martin*
15.47	**-SOLAR SYS**	Section Header	Doubles as ZOUT	*n/a*
15.48	**EARTH**	Planet orbital data	Puts data in R00-R02 & R06	*Ángel Martin*
15.49	**JUPITER**	Planet orbital data	Puts data in R00-R02 & R06	*Ángel Martin*
15.50	**MARS**	Planet orbital data	Puts data in R00-R02 & R06	*Ángel Martin*
15.51	**MERCURY**	Planet orbital data	Puts data in R00-R02 & R06	*Ángel Martin*
15.52	**MOON**	Satellite orbital data	Puts data in R00-R02 & R06	*Ángel Martin*
15.53	**NEPTUNE**	Planet orbital data	Puts data in R00-R02 & R06	*Ángel Martin*
15.54	**PLUTO**	Planet orbital data	Puts data in R00-R02 & R06	*Ángel Martin*
15.55	**SATURN**	Planet orbital data	Puts data in R00-R02 & R06	*Ángel Martin*
15.56	**URANUS**	Planet orbital data	Puts data in R00-R02 & R06	*Ángel Martin*
15.57	**VENUS**	Planet orbital data	Puts data in R00-R02 & R06	*Ángel Martin*
15.58	**+VH12**	Hohmann Transfers v1 & v2	M, R1, R2 in (Z,Y,X)	*Ángel Martin*
15.59	**+VB123**	Bi-Elliptic Transfer V1, V2. V3	(M, r1, r2, rb) in {T,Z,Y,X}	*Angel Martin*
15.60	**+VEa12**	Elliptic Apogee Transfers V12	(r1, r2, r3, r4,M) in Stack & LastX	*Ángel Martin*
15,61	**+VEb12**	Elliptic Perigee Transfers V12	(r1, r2, r3, r4,M) in Stack & LastX	*Ángel Martin*
15,62	**"CORV"**	Co-Orbital Rendezvous	Prompts for Data	*Ángel Martin*
16,63	**"CPRV"**	Co-Planar Rendezvous	Prompts for Data	*Ángel Martin*

Module Dependencies.

The functionality included in this applications module builds on the ELLIPTIC module, and expands the examples included there, like the ellipse eccentricity and perimeter. It requires the ELLIPTIC module plugged in the calculator.

In addition to that, some MCODE functions use routines from the Library#4, which therefore also needs to be plugged to ensure proper operation and correct results.

Elliptic Sector Areas and Arc Lengths.

This section describes a set of functions to calculate sector areas and arc lengths of an ellipse. As it is, the Ellipse is an unassuming conic that presents more difficulty than intuitively expected when first approached.

Let's consider an ellipse with major and minor semi-axis "a" and "b" respectively. Then its eccentricity is given by the expression: e = sqrt(1-b^2) / a^2. We also know that the distance from the foci to the origin is given by: OF = a.e

1.1. Central Sector Areas.

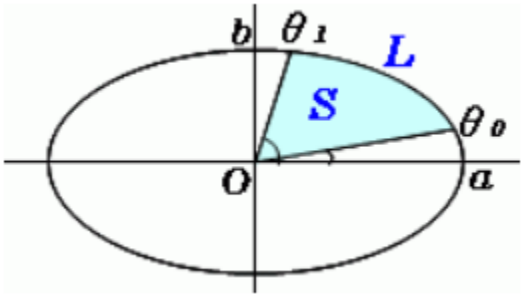
By Central sector we define the area comprised between the arc of the ellipse and the segments that link two points in the ellipse with its center, situated at the origin of coordinates x=y=0.

The origin of angles is the horizontal axis, measured counter-clockwise. Thus a null angle ($\theta = 0$) corresponds to the point x=a, y=0, and an angle of 90° ($\theta = 90$) will represent the point x=0, y=b where {a, b} are the major and minor semi-axis of the ellipse respectively.

With this convention, the formula that gives the areas is shown below, where: $\theta 0 < \theta 1$, and $0 <= \theta <= 90°$

$$S = F(\theta_1) - F(\theta_0)$$

$$F(\theta) = \frac{ab}{2}\left[\theta - tan^{-1}\left(\frac{(b-a)\sin 2\theta}{b+a+(b-a)\cos 2\theta}\right)\right]$$



Examples. Obtain the area of the central sector between the angles $\phi 0 = 45°$ and $\phi 1 = 90°$, for an ellipse with semi-axis values a=3 and b=2.

The central sector area is calculated by the MCODE function **ECS.** It expects the ellipse data already stored in the data registers R00 to R02, and the delimiting angles in the stack - as follows:

Value	Symbol	Register
Eccentricity	e	R00
Major Semi-axis	a	R01
Minor semi-axis	b	R02
Lower angle	$\phi 1$	Y
Upper angle	$\phi 2$	X

So for this example we type:

RAD
3, STO 01, 2, STO 02, XEQ "ECC" =>0.745355993 , the eccentricity
STO 00, PI, 4, /, PI, 2, / , XEQ "ECS" =>1.764007811 , the sector area.

The driver program **ECS+** makes the data input/output a much simpler affair; just enter the values at the corresponding prompts. If you want to re-use the existing data in the registers just press R/S to bypass the input. The input order is not critical, as the program will ensure that a>b and $\phi 1 > \phi 0$

Here's the program listing, a simple application of the auxiliary functions included in the ROM:

01	LBL "ECS+"		06	"A(c)="	
02	XROM "?ab"	semi-axis	07	ARCL X	
03	LBL 00		08	PROMPT	shows the result
04	XROM "<)+"	sector angles	09	GTO 00	other angles
05	**ECS**	does the math	10	END	

1.2. Central Sector Arc Length

Also from the same reference, the formula for the arc length is given by the expression:

(2) *elliptical arch*:

$$L = aE\left(\frac{x(\theta_0)}{a}, k\right) - aE\left(\frac{x(\theta_1)}{a}, k\right)$$

$$x(\theta) = r(\theta)\cos\theta,\ k = \sqrt{1-\left(\frac{b}{a}\right)^2},\quad a \geq b,\ \frac{\pi}{2} \geq \theta \geq 0$$

$E(x,k)$: *2nd incomplete elliptic integral*

Where the radius "r" is the distance from the origin to the point:

$$r(\theta)^2 = \frac{a^2b^2}{b^2\cos^2\theta + a^2\sin^2\theta}$$

The arc length is calculated using the incomplete Elliptic Integral of second kind, which as always, is going to require careful attention to the conventions used - in order to use the functions from the Elliptic Module. Function **LEI2** can be used once we change the parameters to a suitable form, as follows:

E (x, k) = LEI2 (asin(x) ; e^2) ; with e the eccentricity of the ellipse.

In order to apply **LEI2** directly we also need to free the data registers R00-R04, which are used by the program. This is achieved using **REGMOVE** twice, right before calling LEI2 and after it completes.

Example.- Calculate the arc length for the sector used in the previous example, defined by 45 and 90 degrees. The solution is 1.709841

XEQ "LCS"	a? b?
2, ENTER^, 3, R/S	∠(?,?)=? (+)
PI, 4, /, PI, 2, /, R/S	RUNNING... "
	L (+) = 1.709841

The program **LCS** ("Length of Central Sector") is listed below. Note that it's also prepared to calculate arc lengths for focal sectors, which will be the subject of another section later on…

1	LBL "LF+"		1	LBL "<)C-XY"	
2	SF 04		2	XROM "1/R^2"	
3	GTO 04		3	1/X	
4	LBL "LCS"		4	SQRT	r1
5	CF 04		5	ENTER^	
6	LBL 04		6	ENTER^	
7	XROM "?ab"		7	RCL N	sin E1
8	LBL 01		8	*	r1 . sin E1
9	XROM "?<)+"		9	X<>Y	
10	RCL 04	$\alpha 1$	10	RCL M	cos E1
11	?FS 04		11	*	r1 . cos E1
12	XROM "F2C"	E1	12	RTN	
13	STO 11		13	LBL "1/R^2"	
14	RCL 03	$\alpha 2$	14	ENTER^	
15	FS? 04		15	COS	cos E1
16	XROM "F2C"		16	STO M	
17	STO 10	E2	17	RCL 01	a
18	XEQ 03		18	/	
19	STO 12		19	X^2	
20	STO 12		20	X<>Y	
21	RCL 11	E1	21	SIN	sin E1
22	XEQ 03		22	STO N	
23	ST- 12		23	RCL 02	b
24	RCL 12		24	/	
25	RCL 01	a	25	X^2	
26	*		26	+	
27	"L="		27	END	
28	ARCL X				
29	PROMPT				
30	GTO 01				
31	LBL 03				
32	XROM "<)C-XY"	x = r.cos α			
33	RCL 01	a			
34	/				
35	ASIN	asin (x/a)			
36	RCL 00	e			
37	X^2	e^2			
38	X<>Y				
39	5,000005				
40	REGMOVE				
41	RDN				
42	XROM "LEI2'				
43	0,005005				
44	REGMOVE				
45	RDN				
46	END				

The auxiliary function **<)C-XY** calculates the coordinates (x,y) of the point in the ellipse situated at the central angle Φ. In turn, it also uses **1/R^2**, another auxiliary routine to obtain the radius at that point, i.e. the segment connecting it with the origin of coordinates.

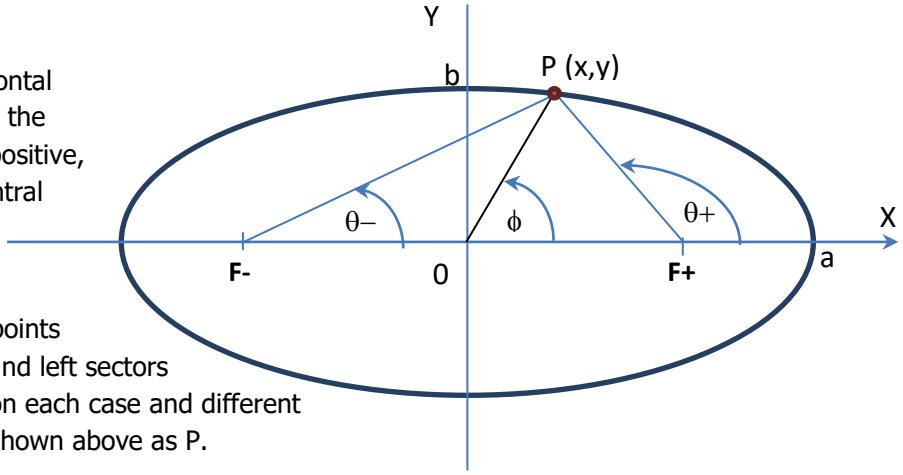
2.1. Focal Sector Areas.

By Focal sector we define the area comprised between the arc of the ellipse and the segments that link two points in the ellipse with one of its foci. This type is very relevant in areas of orbital mechanics and the Kepler laws – as we'll see later in the examples.

From this definition we see that there are two different focal sectors, depending on which focus is used as its "vertex", either **F-** or **F+**. Therefore, we'll call

- Left Focal sector to the one using the left focus as vertex, situated at F- = (-a.e ; 0) , and
- Right Focal sector to the one with the right focus as vertex, at F+ = (a.e ; 0).

For either case we'll use the horizontal axis y=0 as origin for angles, with the counter-clockwise convention as positive, i.e. the same one used for the central sectors.



The angle θ = 90 represents the points P' and P" in figure 2 (next page) and left sectors respectively; i.e. totally different on each case and different as well from the central case, as shown above as P.

The formulas for the focal sector areas for each case are given below:

a) *Right Focal Sector Area*
(formula from: http://bado-shanai.net/Platonic%20Dream/pdAreaofEllipticSector.htm)

$$S_6 = \tfrac{1}{2}a^2\sqrt{1-e^2}(\psi_B - \psi_A) - ea^2\sqrt{1-e^2}\sin(\tfrac{1}{2}[\psi_B - \psi_A])\cos(\tfrac{1}{2}[\psi_B + \psi_A]).$$

Where the parametric angle Ψ is calculated from the azimuth angles θ as follows:

$$\cos\psi_K = \frac{e + \cos\theta_K}{1 + e\cos\theta_K}.$$

b) *Left Focal Sector Area* (formula by David Cantrell).-

S = a.b/2 . [F(θ1) – F(θ0)]

F(θ) = θ + e.Sqrt(1-e^2).sin θ / (1-e.cos θ) + 2.Arctan(e.sinθ / (1-e.cos θ + Sqrt(1-e^2)))

Note that the relationships between left and right focal sectors are obviously given by the symmetrical nature of the ellipse. After all, each case is the mirror reflection of the other – or in terms of the Sun-Earth system, it all depends on the observer's point of view at either side of the solar system plane (from the top or from below), swapping the perihelion and the aphelion.

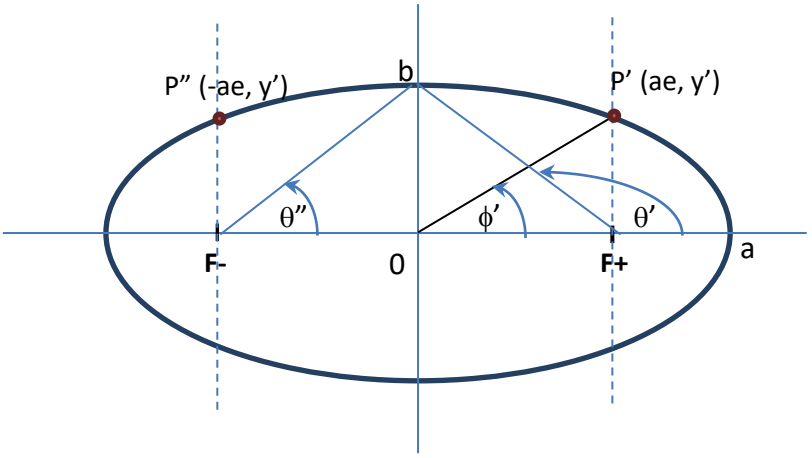


Figure 2. Relevant Central and Focal angles.

Examples. Let the semi-axis be a=3 m. and b=2 m. Calculate the right and left focal sector areas between θ0 = 0 and θ1 = π/2 . Is there any relationship between them?

XEQ "K2+	
2, ENTER^, 3	
0, PI, 2, / ,R/S	A(+) = 0.6985 m^2

Note that you can also enter the semi-axis and the angles in the reverse order, the program will sort them appropriately before saving them in the data registers so that a>b and θ2 >θ1. Note as well that you can just press R/S at the prompts to reuse the existing values in the data registers

XEQ "K2-"	
R/S (uses current)	
R/S (uses current)	A(-) = 8.7263 m^2

The observant reader would notice that A(-) = πab/2 – (A+) =9.4248 – 0.6985 = 8.7263.

The general form expresses that the Left focal sector is the complementary to the semi-ellipse and the Right focal sector. This is just one of the many interdependencies, which will be covered in the next sections.

Assuming θ0 =0, see the table below with the specific relationships (some of them trivial) for the different values of the θ1 angle, between 0 and 2π:

θ1	0	π/2	θ'	π	2π – θ'	3π/2	2π
A+	0	Ao	A'	πab/2	πab – A'	πab – Ao	πab

θ1	0	θ"	π/2	π	3π/2	2π –θ"	2π
A-	0	A"= πab/2 – A'	πab/2 – Ao	πab/2	πab + Ao	πab – A"	πab

ϕ1	0	ϕ'	ϕ"	π/2	π	3π/2	2π
Ac	0	Ao + Δ'	3πab/2 - Δ"	πab/4	πab/2	3πab/4	πab

Sector Delimiting angles.

There are a couple of important border angles in these cases, because they determine a change of the criteria for the calculations – delimiting a sign change in the expressions of the {x,y} coordinates.

- For right Focal sectors, let **θ'** the angle between the X-axis and the segment F(+) to P(0, b).
- For Left focal sectors, let **θ"** the angle between the X axis and the segment F(-) to P(0, b).

  In other words, these are the left and right focal angles corresponding to the point in the ellipse with abscissa zero, x=0.

- We can also find the central angles called ϕ' and ϕ'' corresponding to points on the ellipse with abscissa equal to the foci, i.e. x =a.e and: x= - a.e,

Obviously, we have: $\theta'' = \pi - \theta'$. That and a couple of MCODE functions included in the module can be used to determine the significant angles θ' and θ'', as follows:

- Use **<)C0+** to calculate ϕ', as given by the expression: $\tan \phi' = b.\ sqrt(1-e^2) / a.e$
- Use **<)F0+** to calculate θ', as given by the expression: $\tan (\pi - \theta') = b / a.e$
- Use **<)F0-** to calculate the left focal angle θ'' for x=0, determined by: $\tan \theta'' = b / a.e$

Important remark: these functions expect the eccentricity and semi-major axis values to be already stored in data registers R00 and R01 respectively.

Note:- The Focal angle $\theta = \pi/2$ (i.e. the Central angle $\phi = \phi'$) determines the point **P'**, also known as Gauss' "half parameter" (H) – specifically its Y-value (ordinate).

Example. For the same ellipse with a=3 and b=2, calculate the values of the central and focal border angles. Just in case the contents of the data registers had been altered, we first set the calculator in RAD mode and input the parameters in the data registers (including the eccentricity as well).

3, STO 01, 2, STO 02, XEQ 'ECC"	=>0.745355993 m
STO 00, RAD, XEQ "<)F0+"	=>2.411864997 rad
R-D	=> 138.189685 °
XEQ "<)C0+"	=>0.537683253 rad
R-D	=>30.80698112 °

Therefore, the left focal angle is: $\theta'' = \pi - \theta' = 0.729727657$ rad

Using these as sector upper angles and zero the lower one, the respective areas are:

A+(0, θ') = 2.476321000	-	for the right focal sector,
A- (0, θ") =6.948456965	-	for the left focal sector, and:
Ac(0, ϕ') = 2.189182968	-	for the central sector.

2.2. Focal Sector Arc Length.

The approach followed has been to calculate the central angles corresponding to the points defining the focal sector, and then use the formula for the **central** arc length from the previous section. This allows for a reuse of the same code – namely the application of **ELI2**.

In its initial FOCAL implementation it involved solving a quadratic equation on "d", the focal segment linking the point in the ellipse with the focus (i.e. hypotenuse of the triangle with vertices P-F-0). The quadratic equation is derived from the Cartesian equation of the ellipse: (x/a)^2 + (y/b)^2 = 1

A.d^2 + B.d + C =0, with coefficients as follows:

A = (cos θ / a)^2 + (sin θ /b)^2 = 1/r^2
B = 2.OF.cosθ / a^2 = 2.e.cosθ / a
C = (OF/a)^2 = e^2

The final expression for the central angle depends on the region, with four regions being considered as shown below:

Focal angle	Central angle	Region
θ	ϕ = atan [d.sinθ / (OF + d.cosθ)]	0 <θ<= θ′
θ	ϕ = π - atan|d.sinθ / (OF + d.cosθ) |	θ′<θ<= π
θ	ϕ = π+atan |d.sinθ / (OF + d.cosθ) |	π<θ <= θ″
θ	ϕ = 2π - atan |d.sinθ / (OF + d.cosθ) |	θ″<θ<= 2π

Although explicit, these expressions are not the best for an efficient algorithm implementation. That's why on the current revision they were replaced for more concise approach, based on the parametric expressions linking the variables involved:

$$|OF| + d\cos\theta = r\cos\phi$$
$$d\sin\theta = r\sin\phi$$

Knowing the values o the central radius "r" and the focal segment "d", we can use the R-P function on this pair of equations for the direct and inverse conversions, i.e. from focal to central angle - and back.

Two MCODE functions are included in the module, they are inverse from each other and the input angle can be in degrees or radians:

- **<)F>C** is used to obtain the central angle from the (right) focal angle
- **<)C>F** is used to obtain the (right) Focal angle from the central angle

Finally, the quadratic equation isn't the best method to calculate the focal segment either. This has been further replaced by a direct calculation using the formula below; where θ is the right focal angle:

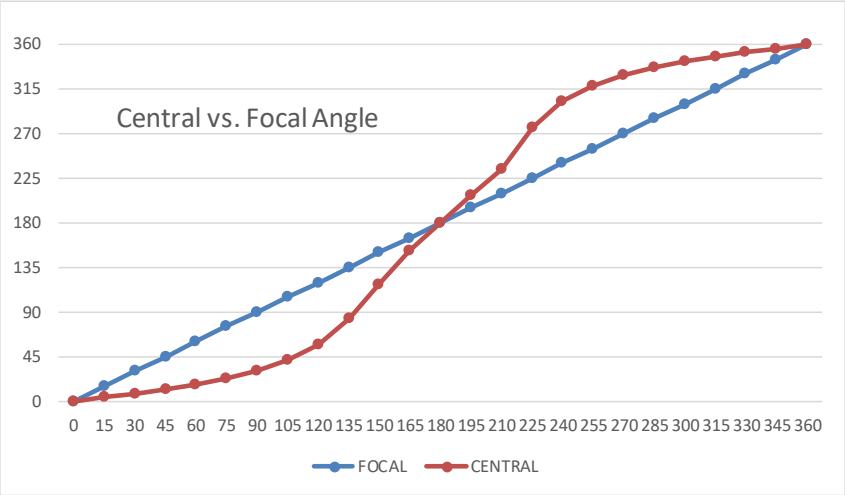
d(+) = a.(1-e^2) / (1 + e. cos θ) for the right segment, and
d (-) = 2a – d(+) for the left segment

Example. Calculate the central angle equivalent to θ' from the previous section.

XEQ "**<)F0+"**	=>	2.41864997 rad
XEQ "**<)F>C"**	=>	1.570796326 = π/2

Note that *this is not the same as ϕ'*, even if both are central angles they certainly aren't the same.

Examples.- For our favorite ellipse with a=3 and b=2, obtain the central angles for the points defined by the focal angles between 0 and 2π, at 15 degrees intervals. The solutions are plotted in the chart below.



Armed with this ammunition, calculating Focal arc lengths is reduced to the same case as Central arc lengths. Let's see a couple of examples to demonstrate the applicability. -

Example. Calculate the arc length of a focal sector with (focal) angles θ1 = 45° and θ2 = 90°

XEQ "LF+"		a? b=?
R/S		(1) (2)?
PI, 4, /, PI, 2, / , R/S		RUNNING...
=>		L(+)=0.954371017

Example. Calculate the arc length of a Central sector with (central) angles ϕ1 = 45° and ϕ2 = 90°

XEQ "LC0"		a? b=?
R/S		(1) (2)?
R/S		RUNNING...
=>		L(+)=1.709841483

Example. Calculate the semi-circumference of the ellipse using the Central or Focal sectors defined by the angles θ1=ϕ1=0° , and θ2=ϕ2 = 180°

XEQ "LF+"		a? b=?
R/S		(1) (2)?
0, PI, R/S		RUNNING...
=>		L(+)=7.932719796
ST+ X	=>	15.86543959 for the complete circumference

FOCAL Routine listing for the "Focal to Central" angle conversion.

As you can see this version uses a couple of MCODE auxiliary functions from the module to reduce the program length and the execution time.

1	LBL "F2C"	*Focal to Central*	28	RCL N	*sin α*
2	CF 00		29	*	
3	CF 01		30	X<>Y	*d*
4	CF 02		31	RCL M	*cos α*
5	X=0?		32	*	
6	RTN		33	RCL 00	*e*
7	<F0+	*α0 = π - atan(b/e.a)*	34	RCL 01	*a*
8	X>Y?	is 0 < α < α0 ?	35	*	*OF*
9	GTO 00	*yes, skip over*	36	+	*OF + d. cos α*
10	SF 00	*flag region*	37	/	*tan E1*
11	CLX		38	FS? 00	
12	PI	*new boundary*	39	ABS	
13	X>Y?	*is α < π ?*	40	ATAN	*E1*
14	GTO 00	*yes, skip over*	41	FC? 00	
15	SF 01	*flag region*	42	RTN	
16	PI		43	FS? 02	
17	X=Y?		44	CHS	
18	RTN	α0 < α <= π	45	FC? 01	
19	RCL M	is α = π ?	46	CHS	
20	+	α1 = π + atan (b/e.a)	47	PI	
21	X<=Y?	is α >= α1 ?	48	+	
22	SF 02	*yes, flag it for later*	49	FC? 02	
23	LBL 00		50	RTN	
24	RDN	*drop boundary*	51	PI	
25	<)F>R+	*right focal angle to radius*	52	+	
26	ENTER^	*d*	53	END	
27	ENTER^				

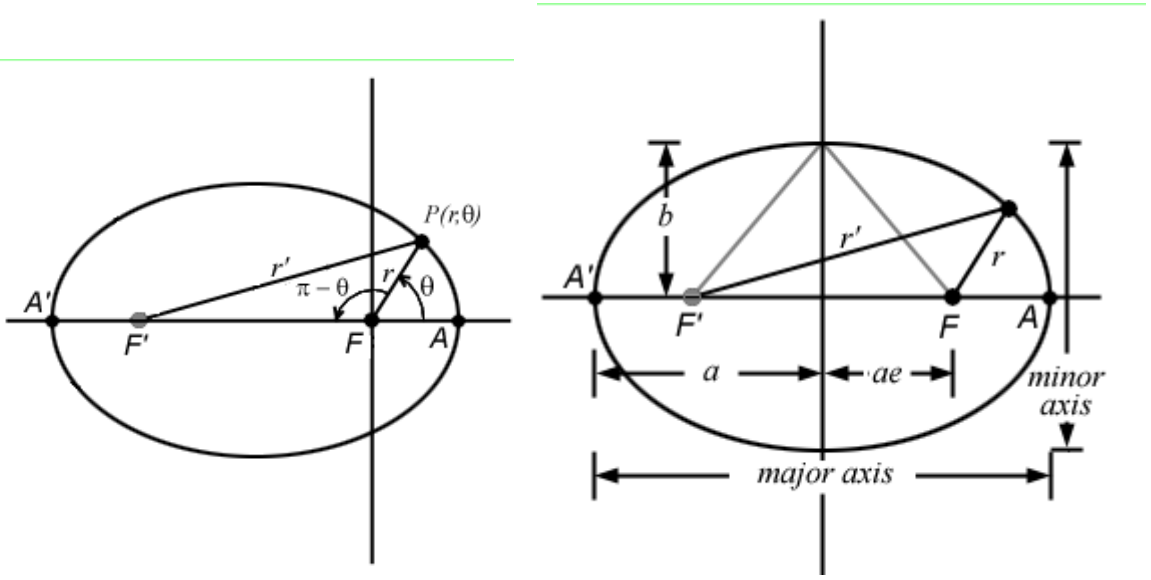
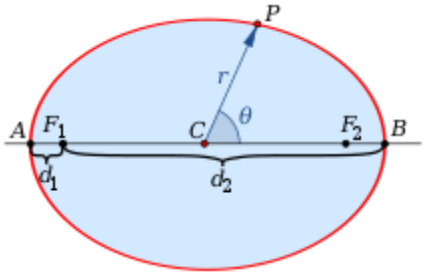


Figure 0 – Basic geometry parameters in the Ellipse

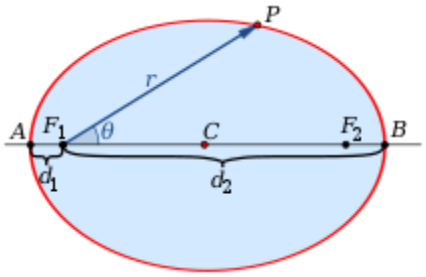
Polar form relative to center



In polar coordinates, with the origin at the center of the ellipse and with the angular coordinate θ theta measured from the major axis, the ellipse's equation is

$$r(\theta) = \frac{ab}{\sqrt{(b\cos\theta)^2 + (a\sin\theta)^2}}$$

Polar form relative to focus



If instead we use polar coordinates with the origin at one focus, with the angular coordinate θ = 0 theta =0 still measured from the major axis, the ellipse's equation is:

$$r(\theta) = \frac{a(1-e^2)}{1 \pm e\cos\theta}$$

where the sign is *plus for RIGHT focal angles* (subtended by right focus segments d1), and *minus when using LEFT focal angles* (subtended by left focus segments d2)

The FOCAL program below makes the conversions between left and right focal angles, θ(-) and θ(+). It's an easy application of the same method used in the Central-to-Focal routines, as per the equations:

r(-). sin θ(-) = r(+). sin θ(+)
r(-). cos θ(-) = 2.e + r(+). cos θ(+)

1	LBL "F+>F-"		12	RCL 00	*e*
2	CF 00		13	RCL 01	*a*
3	<)F>R+	*right radius*	14	*	*e.a*
4	*GTO 00*		15	ST+ X	*2e.a*
5	LBL "F->F+"		16	FS? OO	*left-to-right?*
6	SF 00		17	CHS	*yes, sign change*
7	<)F>R-	*left radius*	18	+	*add to x-coordinate*
8	LBL 00		19	R-P	*back to polar*
9	LASTX	*recall angle*	20	X<>Y	*result to X*
10	X<>Y	*sort locations*	21	END	done.
11	P-R	*to rectangular*			

Note that the angle result is in the X register, and the left or right focal radius is left in the Y register.

Example. Calculate the left focal angle corresponding to a right focal angle of 45 degrees. Check the result using **<)F>R-** Assuming DEG is on, we type:

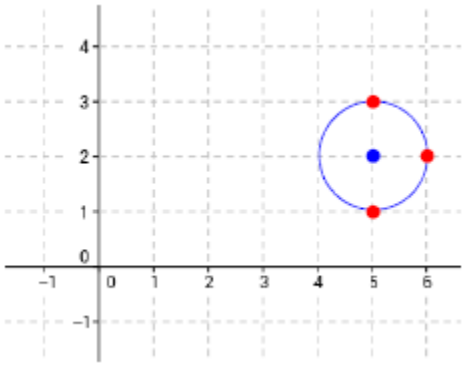
45, XEQ "**F+>F-**"	6,9 16693242	deg
XEQ "**<)F>R-**"	5, 12685465	left-radius
–	-0,00000000	difference

Digression 1: Circles, Triangles & Circumferences.

A short digression on subjects unrelated to the geometry of the ellipses.

- **CIRCLE** calculates the radius of a circle passing thru three data points, using the point x,y coordinates. The values are expected to be stored in R01-R06. Besides that, it'll also return in the Y-register the area of the circumscribed triangle defined by the three points.

  Example: Calculate the radius of the circle passing thru P(5,1), Q(6,2), and R(5,3)



  The results are:

  XEQ "CIRCLE" => r=1,000000000,
  X<>Y => A=1,000000000

  Note that you can use the routines **IN** or **INPUT** to populate the registers automatically. The input sequence starts with the abscissa of P1 in R01.

- **HERON** calculates the area of a triangle knowing its three sides, using Heron's formula. Just enter the sides values in the stack, and execute the function (located in the auxiliary FAT). The result is stored in X, with the original side saved in LastX. The rest of the stack is unchanged.

  Let the triangle ABC with 3 known sides { a , b , c } and s = (a+b+c)/2 the semi-perimeter

  Heron's formula is: $\text{Area} = [\, s(s-a)(s-b)(s-c)\,]1/2$

  Example: a = 2, b = 3, c = 4
  Type: 2, ENTER^, 3, ENTER^, 4, XEQ "HERON" => Area = 2.904737510

  Note: the function **CIRCLE** described above makes use of the HERON formula internally after it first calculates the triangle sides from the point coordinates.

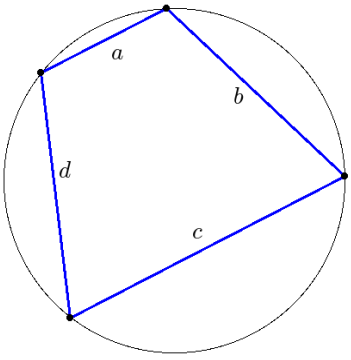
- **BRHM** is related to it, but the calculation for the area of the cyclic quadrilateral - using Brhamagupta's formula. Just enter the four values in the stack and execute the function (in the secondary FAT). The result is stored in X, with the original side saved in LastX. The rest of the stack is unchanged.

  Let a, b, c, and d be its sides lengths, and the semi-perimeter s = (a + b + c + d)/2 .The area A of the cyclic quadrilaterals:

  $A = [(s-a).(s-b).(s-c).(s-d).]1/2$

  Example. Calculate the area for the values:
  a = 4 , b = 5 , c = 6 , d = 7

  Type: 4, ENTER^, 5, ENTER^, 6, ENTER^, 7,
  XEQ "BRHM" => Area = 28.98275349



Circumference of the Ellipse. { **RAMA2** }

The ELLIPTIC module has the routine **ELP** that calculates the accurate value for the perimeter of the ellipse. **ELP** uses the complete Elliptic integral of 2nd. kind, which is based on the AGM implementation – and therefore is relatively fast.

The function **RAMA2** uses Ramanujan's 2nd approximation, which has the advantage of being even faster – albeit the accuracy may not be as good, as shown in the table below for a few examples.

Let h = (a-b)^2 / (a+b)^2, then the formula used is:

$$C = \pi \cdot (a+b) \cdot \left(1 + \frac{3 \cdot h}{10 + \sqrt{4 - 3 \cdot h}}\right)$$

a	**b**	ELP	RAMA2	Error %
3	2	15.86543959	15.86543959	0
4	1	17.15684355	17.15683926	-2.5005E-07
5	3	25.52699886	25.52699886	0
6	2	26.72978556	26.72978556	0
10	2	42.02008908	42.02005330	8.5149820-09

Note that **RAMA2** expects the semi-axis values in the X, Y registers. The input order is indistinct but for **ELP** the order is important, with **b** in Y and **a** in X.

The reason this formula is called the 2nd. approximation is because – you guessed it – Ramanujan had already put forward another expression for the circumference of the ellipse, shown below. It was very accurate for near-circular cases, but as the ellipticity increases the accuracy was lost.

The simplified expression for the circumference is:

$$C = \pi\left[3(a+b) - \sqrt{(a+3b)(3a+b)}\right].$$

Refer to the link below for a comprehensive discussion of the different approximations used historically on this subject.

http://www.ebyte.it/library/docs/math05a/EllipsePerimeterApprox05.html

Digression 2: Complex Elliptic Integrals.

The Complex Elliptic integrals are covered in several FOCAL programs, all included in the 41Z Deluxe module. For your convenience, the Incomplete types are also included in this module. Note that:

- The amplitude can be a complex number but the modulus is expected to be a real value. This method uses dedicated formulas that apply the real expressions on a repeated basis according to changes of variable. Here the program **ZELIP1** corresponds to F(z; m) , and**ZELIP2** corresponds to E(z; m).
- No provision is made for the case where both amplitude and modulus are complex numbers. To check the results you can use the syntax "EllipticF" and "EllipticE" on WolframAlpha using two arguments for incomplete cases or just one argument for complete cases.

Let's see an example next. Be aware that the execution time can range from long to very long depending on the case. You can abort the execution pressing the R/S key at any time.

Example. Calculate the incomplete Elliptic integrals for a= 1-i, m=0.5

1, CHS, ENTER^, CHS, ENTER^, .5, XEQ "ZELIP1" => 0.804+J1.163

EllipticF(1-i, .5): http://www.wolframalpha.com/input/?i=EllipticF%281-i,+.5%29

1, CHS, ENTER^, CHS, ENTER^, .5, XEQ "ZELIP2" => 1.128+J0.789

EllipticE(1-i, .5): http://www.wolframalpha.com/input/?i=EllipticE%281-i,+.5%29

Formulas used (from Abramowitz-Stegun, Section 14.4)

Writing z= (phi+ i psi) then we have for the first kind:

$$F(\phi + i\psi|m) = F(\lambda|m) + iF(\mu|1-m)$$

Where cot^2 (λ) is the positive root of the quadratic equation:

$$x^2-[\cot^2\varphi+m\sinh^2\psi\csc^2\varphi-m_1]x-m_1\cot^2\varphi=0$$
$$\text{and } m\tan^2\mu=\tan^2\varphi\cot^2\lambda-1.$$

And similarly for the second kind integral:

$$E(\varphi+i\psi\backslash\alpha)=E(\lambda\backslash\alpha)-iE(\mu\backslash 90°-\alpha)+iF(\mu\backslash 90°-\alpha)+\frac{b_1+ib_2}{b_3}$$

where now:

$$b_1=\sin^2\alpha\sin\lambda\cos\lambda\sin^2\mu(1-\sin^2\alpha\sin^2\lambda)^{\frac{1}{2}}$$
$$b_2=(1-\sin^2\alpha\sin^2\lambda)(1-\cos^2\alpha\sin^2\mu)^{\frac{1}{2}}\sin\mu\cos\mu$$
$$b_3=\cos^2\mu+\sin^2\alpha\sin^2\lambda\sin^2\mu$$

as you can see an elaborate set of equations that requires a relatively long FOCAL program even if some auxiliary functions really expedite things significantly. Refer to next page for the FOCAL program listing as a reference.

The solution is therefore expressed as a linear combination of the real-variable case for the Elliptic integrals, which are also included in the ELLIPTIC module as functions **ELIPF** and **LEI1** and **LEI2**.

Program Listing: Complex Incomplete Elliptic Integrals.

Data Registers: R00-R08 ;User flag: F1

```
01  LBL "ZELIP1"
02  SF 01
03  GTO 00
04  LBL "ZELIP2"
05  CF 01
06  LBL 00
07  RAD
08  STO  00
09  RDN
10  STO  01
11  RDN
12  SINH
13  X^2
14   /
15  RCL  00
16  E
17  -
18  *
19  E
20  STO T(0)
21  RDN
22  QROOT
23  X<Y?
24  X<>Y
25  STO 02
26  RCL 01
27  TAN
28  X^2
29  *
30  E
31  -
32  RCL 00
33   /
34  SQRT
35  ATAN
36  STO 01
37  E
38  RCL 00
39  -
40  RCL 01
41  ELIPF
42  FS? 01
43  GTO 00
44  STO 01
45  RCL 00
46  RCL 02
47  SQRT
48  1/X
49  ATAN
50  ELIPF
51  STO 00
52  ZRCL 00
53  ZOUT
54  RTN
55  LBL 00
56  STO 08
57  RCL 00
58  RCL 02
59  SQRT
60  1/X
61  ATAN
62  STO 02
63  SIN
64  X^2
65  *
66  E
67  -
68  CHS
69  STO 03
70  SQRT
71  RCL 00
72  *
73  RCL 02
74  E
75  P-R
76  *
77  *
78  RCL 01
79  SIN
80  X^2
81  *
82  RCL 01
83  SIN
84  X^2
85  STO 06
86  RCL 00
87  SQRT
88  ASIN
89  COS
90  RCL 01
91  SIN
92  *
93  X^2
94  CHS
95  E
96  +
97  SQRT
98  RCL 03
99  *
100 RCL 01
101 E
102 P-R
103 *
104 *
105 RCL 02
106 STO 05
107 SIN
108 *
109 X^2
110 RCL 00
111 *
112 RCL 01
113 COS
114 X^2
115 +
116 ST/ 06
117 ST/ 07
118 RCL 08
119 ST+ 07
120 E
121 RCL 00
122 STO 08
123 -
124 RCL 01
125 XROM "LEI2"
126 ST- 07
127 RCL 08
128 RCL 05
129 XROM "LEI2"
130 ST+ 06
131 RCL 07
132 RCL 06
132 ZOUT
133 END
```

Granted, this listing doesn't have much of a complex flavor to it since it really operates on real variable functions. Pulling all stops with the aid of key functions we deflect the complex variable with linear combinations as per the formulas shown before.

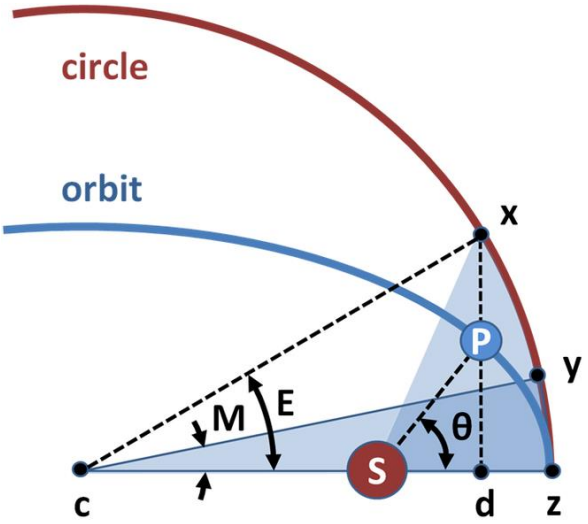
Orbital Position using Kepler Equation.{ **TA<T>** }

Disclaimer: If you're reading this then chances are that you have already forgotten more astronomy than what I have ever learned – so bear that in mind, you're likely going to be presented a newcomer's perspective of otherwise well established stuff.

Because the velocity is not constant during the trajectory, determining the position of a satellite in its orbit at a given time was a daunting problem when Kepler embarked into the search for a solution. He devised ingenious methods and models to approach the task, using the tools known back then (obviously no powerful computers and numerical methods were available). They continue to be taught, although I'd imagine not likely used in real-life scenarios.

Anomalies galore.

Concepts like auxiliary circle and Mean & Eccentric anomalies originate from those days, and are profusely utilized today by astronomer and flight-path planners. Very likely today they use numerical methods and more complex models, bearing higher accuracy in the results.



Transferring the problem to a fictitious planet that moves at constant angular velocity ω in the auxiliary circle is no doubt a great trick; as in there the new position can be easily calculated using the mean velocity ($\omega = 2\pi$ / Period) and the elapsed time of passage, (t1 – to).

Therefore: $\boxed{M' = Mo + \omega.\,(t1 - t0)}$ - Eq.(1)

The question then is how to relate the real focal angle θ (a.k.a. the "true anomaly") to the fictitious one "M", also called 'mean anomaly". The answer is through an auxiliary, intermediate angle "E" - called the "eccentric anomaly". A set of angle transformation MCODE functions are included in the module to perform the conversions, as described below.

- **T>E** and **E>T** will convert between True and Eccentric anomaly back and forth, using the expression:

$$\cos E = \frac{e + \cos \nu}{1 + e \cos \nu}$$

- **E>M** and **M>E** will convert back and forth between Eccentric and mean anomaly, using Kepler's equation: $\boxed{M = E - e.\sin(E)}$. Note that the indirect conversion **E>M** requires an iterative process, implemented as a custom formula for the successive approximations:

  En+1 = En - 2 f(En) / { f '(En) - [f '(En)2 - 2 f(En) f ''(En)] 1/2 } , using E0 = M

  Note also that for accuracy reasons, the **E>M** algorithm expects the inputs in degrees and returns the result also in degrees.

The complete process is rather simple:

1. Initial True position => Eccentric Anomaly => Mean anomaly
2. New position in the auxiliary circle given by Eq.(1) above
3. Mean => Eccentric => True new position
4. Finally, Once the new angle is known, the travelled distance can be quickly obtained from the elliptical arc length expressions from previous sections.

Which has been implemented into the following driver program:

1	**LBL "TA<T>"**		19	RCL 06	*P*
2	**XROM "?ab"**	*semi-axis values*	20	/	*(t1-t0)/P*
3	*"P=?"*	*Period*	21	PI	
4	PROMPT		22	ST+ X	*2 π*
5	STO 06		23	*	
6	*"<)0=? (+)"*	*initial position*	24	+	
7	PROMPT		25	R-D	*required by M>E*
8	STO 03		26	**M>E**	Mean to Eccentric
9	LBL 00		27	D-R	*in rad*
10	*"dT=?"*	*elapsed time*	28	RCL 00	*e*
11	PROMPT		29	X<>Y	*needed by M>E*
12	STO 05		30	**E>T**	*Eccentric to True*
13	RCL 00	*e*	31	STO 04	*final result*
14	RCL 03	*<)0*	32	*"<) ' ="*	
15	**T>E**	*True to Eccentric*	33	ARCL X	
16	**E>M**	*Eccentric to Mean*	34	PROMPT	
17	RCL 05	*t1-t0*	35	*GTO 00*	*new position*
18	-		36	END	

Refer to the QRG in the intro section for the input parameters needed by the anomaly conversion functions.

Example.- A point orbiting the ellipse with a=3, b=2, with a period of 24 hours, is known to be at the periapsis at t=0. What will be its position 2 hours later?

We type:

XEQ "TA<T>"		a ? b = ?
3, ENTER^, 2, R/S		P = ?
24, R/S		∠) 0 = ? (+)
0, R/S		dT = ?
2, R/S	=>	∠) ' = 2.145545841

Therefore, the new focal angle is 122.93 °

Plugging this result as upper angle in **LF+**, the travelled distance results:

L (+) = 2.933965 1138

Orbital Position using Direct Approach. {**TA+** , **TA-**}

Kepler's second law states that the focal areas swept by a satellite are proportional to the times taken to sweep them. Therefore, calculating areas is a great surrogate to determine times, and from there actual positions if we can also link them to the trajectories (arc lengths of the orbit). This is the basis of the direct approach, which puts together many components covered in this manual so far.

Position and travelled distance at a given time.

Say we want to find the position of the satellite at a certain time t1, knowing where it was at an earlier moment t0. During that time interval the satellite has swept an area A(t1-t0), which can be expressed in terms of Kepler's 2^n law as:

$A(t1-t0)=(t1-t0).\ \pi.a.b\ /\ T$; where T is the period of the movement.

As it is, we have an expression for said area, as a function of the focal angles (known as "true anomalies"); in fact we have it both for the left and right focal sectors in case we need to choose.

What's the missing link? Obviously a relationship linking the time elapsed "t1" and the new true anomaly $\theta 1$ – but this can be replaced by an iterative approach to solve the equation for θ, calculating different areas iteratively until it matches Kepler's 2nd law expression:

Guess θ => A(θ) => matches known result? If Yes => done
If NO, modify guess value and try again

In other words, solve: $A(t1) - (t1-t0).\ \pi.a.b\ /\ T = 0$.

Here the tricky part is going to be the expression for the successive approximations of the true anomaly, which will be deferred to the numerical method (Newton, Secant, etc.). For initial guess value we can use a circular estimation based on a mean angular velocity, $\omega o = 2\pi/T$; thus: $\theta o = \omega o.\ t1$. Further refinement can be done to reduce the number of iterations, using some kind of correction through the orbit's eccentricity "e" (not necessarily the eccentric anomaly).

Important remark: The module includes two ways to calculate this, **TA+** and **TA-** ,which internally use right or left focal sector areas respectively. The direct method assumes that the body orbited around (the Sun in the Earth's case) is *located in the RIGHT focus of the ellipse.* Consequently. Both programs expect *the <u>same</u> input values*, including the initial angle - as a **right focal angle**.

Example.- Using the direct approach, calculate the true anomaly in the same new position given by the previous example.

We type:

XEQ "**TA+**"	a? b = ?
3, ENTER^, 2, R/S	P = ?
24, R/S	∠ 0 = ? (+)
0, R/S	dT = ?
2, R/S	RUNNING. . .
=>	∠ (T) = 2.145545841

Example 2.- Using the data from appendix 1, calculate the position of the earth 25 days after the perihelion. Do it first via the Kepler indirect method, and verify it using the direct approach.

We type:

XEQ "**TA<T>**"		αTb=?
1.496 E8, ENTER^, 149579116.1 ,		
R/S		P=?
365.25 , R/S		∠)0=?(÷)
0, R/S		dT=?
25, R/S	=>	∠)'=0.444425550 9

Alternatively:

XEQ "**TA-**"	αTb=?
R/S (will use current)	P=?
365.25, R/S	∠)0=?(÷)
0, R/S	dT=?
25, R/S	∠)T=0.444425550 9

That's to say the new position is at 25.472 degrees away from perihelion.

Plugging this result as upper angle in **LF+**, the travelled distance results:

L(÷)=65,384,908.48 km

The program uses a really simple initial guess for the interval, adding one radian for **TA+** case, or subtracting it in **TA-** case.

Note: This method requires a general-purpose solver. There are a few available in different modules:

Function	Module	XROM	Note
FROOT	SandMath	XROM 02,15	
FROOT	S&I ROM	XROM 27,04	Used here
SOLVE	Advantage	XROM 24,03	

Using the SIROM requires only 4k in the bus – and makes the operation independent from the SandMath. This is aligned with the ELLIPTIC module approach as well.

Program Listing.-

1	LBL "TA+"	Right Sectors	39	LBL "<)T"	*function to solve*
2	CF 04		40	STO 04	*current angle*
3	*GTO 04*		41	FC? 04	*right sectors?*
4	LBL "TA-"	Left Sectors	42	XROM "EFC+"	*yes, calculate it*
5	SF 04		43	FC? 04	*right sectors?*
6	LBL 04		44	*GTO 04*	*yes, skip*
7	XROM "?ab"	*axis data*	45	RCL 03	*upper angle*
8	*"P=?"*	*period*	46	EFS-	*upper left sector*
9	PROMPT		47	STO 07	*temporary store*
10	STO 06		48	RCL 04	*lower angle*
11	PI	*pre-loaded*	49	EFS-	*lower left sector*
12	*"<)0+=?"*	*initial angle*	50	ST- 07	*subtract from upper*
13	PROMPT		51	RCL 07	*sector area*
14	FS? 04	*left sectors?*	52	RCL 01	*a*
15	-	*complementary!*	53	RCL 02	*b*
16	STO 03		54	*	*a.b*
17	LBL 00		55	*	*a.b.A*
18	*"dT=?"*		56	2	
19	PROMPT		57	/	*a.b.A/2*
20	STO 05		58	LBL 04	
21	*"<)T"*	*function name*	59	RCL 05	*dT*
22	RCL 03	*upper guess*	60	RCL 06	*P*
23	RCL 03		61	/	*dT/P*
24	1		62	PI	π
25	FS? 04		63	*	π*.dT/P*
26	CHS		64	RCL 01	*a*
27	+	*lower guess*	65	*	π*.a.dT/P*
28	FROOT	*in the SandMath*	66	RCL 02	*b*
29	FC? 04	*right sectors?*	67	*	π*.a.b.dT/P*
30	*GTO 04*		68	-	*a.b.A -* π*.a.b.dT/P*
31	CHS		69	END	*done.*
32	PI				
33	+				
34	LBL 04				
35	*"<)TA="*				
36	ARCLX				
37	PROMPT				
38	GTO 00	*new position*			

Time of Flight: Time to travel a given arc.

To round up the options, here's the inverse problem: we want to know the time taken to travel a certain distance (also known as Time of Flight) – measured by its new true anomaly.

Here the easiest approach is to move in the auxiliary circle using the mean anomaly, from an initial location (t=0) to a final one (t=TOF). Since the speed is constant in such circle the equation is a trivial one:

$$TOF = \Delta M \cdot To/2\pi = (E - e.\sin E) \cdot To/2\pi$$

The MCODE function **TOF** calculates the Time of Flight from the following input values in the stack:

Register	Input	Output
Z:	To, period	To – remains there
Y:	e, eccentricity	e - remains there
X:	$\Delta\theta$, true anomaly delta	TOF - result
L:	n/a	θ - previous TA delta

This stack arrangement facilitates repeated executions for different true anomalies: simply press RDN and input a new angle (or call LastX to retrieve the original one).

Example. For our friendly test ellipse (a=3, b=2) calculate the time it would take a fictitious body orbiting around the RIGHT focus with an orbital period of 24 hours to move from the point P1 (0, b) to the opposite point P2(0, -b) – that is, the "slow" half-ellipse.

We type:

2, STO 02, 3, STO 03 XEQ "ECC" => 0.745355993
STO 00, 24, X<>Y , XEQ "<)F0+" => 2.411864997
XEQ "TOF" => 3.152949823 hours from perigee to x=0

Therefore the time between θo and the perigee is:

t = 12 -3.152949823 = 8.847050177 hours

And finally the requested time is twice that amount:
ST+ X => 17.694100355 hours between P1-P2

Example: Calculate the time taken by the Earth to move from apogee to $\theta(+)=\pi/2$. Use the data from appendix1: e=0.01671 and T = 365.25 days/

We type:

XEQ "EARTH" (so we don't need to type the numbers…;-)
RCL 06, RCL 00, PI, 2 , /, XEQ "TOF" => 89.371395598 days

Orbital Velocity using Conservation of Energy. {**V<R>**}

Moving on, let's see how to calculate the instantaneous velocity if the orbiting body at a generic position, determined by its true anomaly. As before, we'll devise two methods for the calculation, the first one based on the energy conservation and the second based on the angular moment conservation. The second method also requires knowing the velocity at a reference point, typically the periapsis (aka. perihelion for the earth around the sun, or perigee for satellites around the earth).

The energy conservation states that the sum of kinetic and potential energies is constant. In particular, for a point situated at a distance r from the focus, the formula below determines the magnitude of its velocity – expressed only as a function of the orbital period and the geometry:

V(r) = 2π/T sqrt[(2a/r) -1]

In particular for the periapsis and apoapsis, r = a(1–e) and r= a(1+e) respectively ; thus we have:

Vp = 2π.a/T sqrt [(1+e)/(1-e)]
Va = 2π.a/T sqrt [(1-e)/(1+e)]

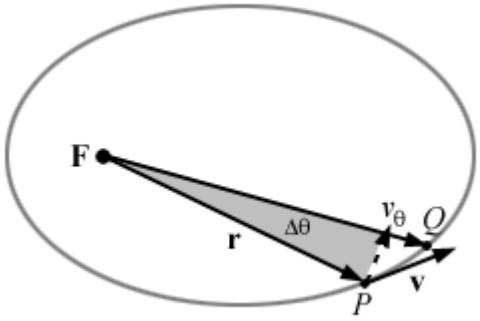
Thus their ratio: Va/Vp = (1-e)(1+e) ; a common parameter of the ellipse.

Vis Viva Equation.

In astrodynamics, the vis-viva equation, also referred to as orbital-energy-invariance law, is one of the equations that model the motion of orbiting bodies. It is the direct result of the principle of conservation of mechanical energy which applies when the only force acting on an object is its own weight. For any Keplerian orbit (elliptic, parabolic, hyperbolic, or radial), the vis-viva equation is as follows:

$$v^2 = GM\left(\frac{2}{r} - \frac{1}{a}\right)$$

Where r is the distance between the two bodies, and μ=G.M is standard gravitational parameter. In our programs we'll use the orbit period as input, thus for an elliptic orbit: μ^2 = 2p/T sqrt(a^3) ;
which is the format we'll use.



The distance r (elliptic radius) varies with the focal angle, so we need to be careful on which one to use. For a LEFT configuration (i.e. with the Sun in the left focus) it also has a closed-form representation:

$$r(\theta) = \frac{a(1-e^2)}{1 - e\cos\theta}$$

Remember: For the program **V<R>**the convention is to *use the LEFT focal angles* – and not the right ones; make sure you input the correct one.

Examples. Using the data from appendix#1, **c**alculate the instantaneous velocity of the Earth in its orbit around the sun for the three "cardinal" points of the semi-orbit: apogee (θ=0), perigee (θ=π) and θ(x=0)

Solution.- We type:

XEQ "V<R>"			
1.496 E8, ENTER^, 149,579,116.1 , R/S			
365.24 , R/S			
0, R/S	=>		km/day
R/S	=>		at the apogee
R/S			
PI, R/S	=>		km/day
R/S	=>		at the perigee
R/S			
XEQ "<F0-",			
R/S	=>		km/day
R/S	=>		at x=0

Program Listing.

1	**LBL "V<R>"**		22	*	a/r
2	**XROM "?ab"**		23	ST+ X (3)	2a/r
3	*"P=?"*		24	FS? 00	*to tell the cases apart*
4	PROMPT		25	1	
5	LBL 00		26	FS? 00	
6	*"<)=? (-)"*		27	-	*(2a/r) -1*
7	PROMPT	*left focal angle*	28	SQRT	*sqrt[(2a/r)-1]*
8	STO 03		29	RCL 01	*a*
9	**<)F-R-**	*left radius*	30	*	*a.sqrt[(2a/r)-1]*
10	STO 05		31	PI	
11	SF 00	first pass	32	ST+ X (3)	*2p*
12	XEQ 02	velocity	33	*	*2p.a.sqrt[(2a/r)-1)]*
13	STO 04	*V(r)*	34	RCL 06	*Period*
14	*"V(<))="*		35	/	*2p.a.sqrt[(2a/r)-1)] / To*
15	ARCL X		36	FS? 00	*first pass?*
16	PROMPT		37	RTN	*yes, return here*
17	CF 00	second pass	38	*"Vc="*	
18	LBL 02	velocity	39	ARCL X	
19	RCL 05		40	PROMPT	*show result*
20	1/X		41	*GTO 00*	*new position*
21	RCL 01	a	42	END	

Orbital Velocity as function of the abscissa.

For a known period of the movement To, it's possible to express the velocity at any point in the orbit as a function of the abscissa, i.e. the x-coordinate of the ellipse. The formula is derived directly from the vis viva equation, replacing the (right) focal segment "r" with the following:

$$r = a.(1 - e.\cos E) \quad ; \text{ where } \cos E = x/a$$

the final expression is:

$$V(x) = (2\pi.a/To)\ \sqrt{[\ (1 + e.x/a) / (1 - e.x/a)\ ]}$$

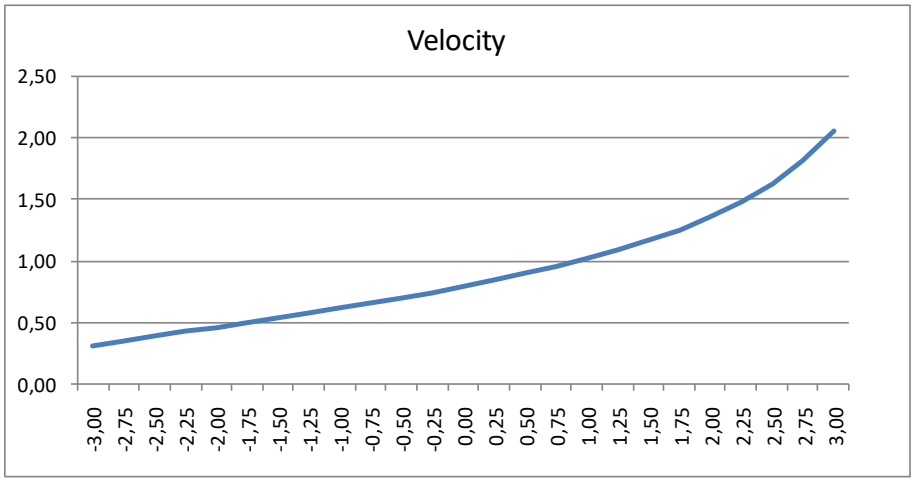
Note that the eccentric anomaly substitution assumes the orbited body is located in the **right** focus.

The function **V(X)** calculates the point velocity; it expects the period in the Y-register and the x-coordinate in the X-register as inputs. Upon completion the result is in X and the coordinate in LastX – but the period remains in Y for convenience in case of repeated executions.

Example: Obtain the table of velocities for the same ellipse at points between perigee (x=3) and apogee (x=-3) at intervals of the abscissa of $\Delta x = 0.25$. Use T0=24 h for the period.

Assuming the ellipse parameters stored in R00-R02, the minimalistic FOCAL program below produces all the results:

```
01  LBL "VX"
02  24
03  ENTER^
04   -3
05  LBL 00
06  V(X)
07  STOP
08  RDN
09  LASTX
10  ,25
11  +
12  GTO 00
13  END
```



x	v
-3,00	0,3000
-2,75	0,3407
-2,50	0,3797
-2,25	0,4177
-2,00	0,4553
-1,75	0,4929
-1,50	0,5309
-1,25	0,5696
-1,00	0,6094
-0,75	0,6504
-0,50	0,6932
-0,25	0,7380
0,00	0,7854
0,25	0,8358
0,50	0,8899
0,75	0,9484
1,00	1,0123
1,25	1,0829
1,50	1,1618
1,75	1,2514
2,00	1,3548
2,25	1,4767
2,50	1,6246
2,75	1,8105
3,00	2,0562

Orbital Velocity using Angular Momentum. {**VR+** , **VR-**}

An alternative approach is based on the conservation of the angular momentum. Because in a two-boy problem no external torque is applied on the set star/planet (or planet/satellite), its angular momentum is constant. In other words, the angular momentum has the same value in all positions on the elliptic orbit.

In particular, on the periapsis and apoapsis the velocity vectors are perpendicular to the elliptic radius, thus the angular momentum at those positions will verify:

r(a). m. Va = r(p) m. Vp ; and therefore: V(a)/V(p) = r(a)/r(p)

At a generic position determined by its focal angle θ, both vectors aren't perpendicular so the angular momentum depends on the angle between the velocity and radius vectors, α.

Va. r(a) = V .r(θ). sin α $\Rightarrow$ V = [Va .a(1+e)] / r(θ) . sin α

where we can apply the Vis Viva equation to obtain either V(a) or V(p).

An expression of r(θ) is easy to get based on the geometric properties of the ellipse. It follows that if we can calculate said angle α, then the velocity will also be known. How to go about it?Simply using the fact that the velocity vector is tangent to the ellipse at the given point, which is the same as saying *the derivative of the ellipse* at that point will give us the slope of the tangent line,

$\beta(\theta)$ = atan(dy/dx)|x=x(θ) ; And from here: $\alpha = \theta + \beta(\theta)$

From the equation of the ellipse it's easy to derive its derivative, say between [-π, π]

y'= dy/dx = - b.x / [a.sqrt(a^2 – x^2)]

Two programs are included to cover both he Left and Right focal angle configurations:

- use right focal angles with **VR+**, i.e. *the sun is at the right focus*
- use left focal angles with **VR-**, i.e. *the sun is at the left focus*

The direction of the movement can be either clockwise or counter-clockwise indistinctly.

Escape Velocity

The formula for escape velocity can be obtained from the Vis-viva equation by taking the limit as "a" approaches infinite∞ :

Vc (r) = 2π.a/P sqrt(2a/r)

We'll remark the fact that the escape velocity depends on the distance between the two bodies.

Note: Vc is also calculated by the **V<R>** program.

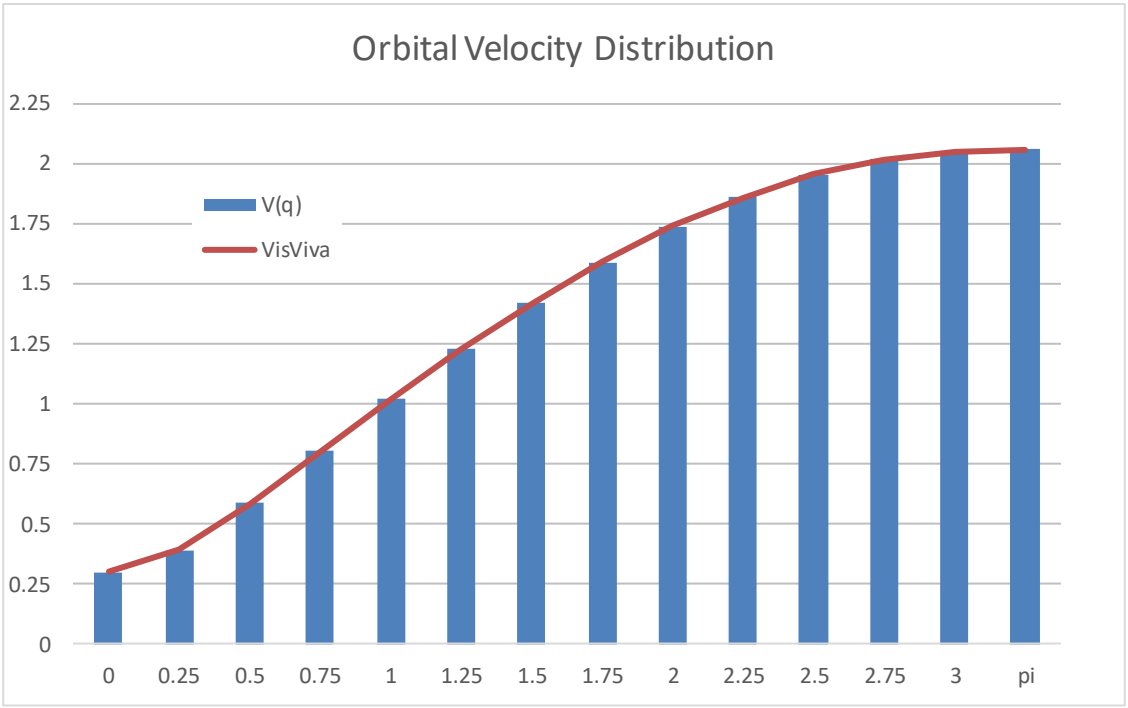
Example. Calculate the velocity distribution for a body orbiting in an ellipse with semi-axis a=3m, b=2m, if the period is 24 hours and the body orbited around is placed at the LEFT focus.

The solutions are provided in the table below. Note that the convention is Left focal angles, i.e. the apoapsis occurs at $\theta=0$, and the periapsis occurs at $\theta=\pi$

Special Position	VisViva	θ	r(θ)	X(θ)	V(θ)
apogee	0.3	0	5.2361	3	0.3
	0.3928	0.25	4.7994	2.4141	0.3928
	0.5859	0.5	3.8548	1.1468	0.5859
x=0	**0.7854**	**θ"**	**3**	**1.00E-09**	**0.7854**
	0.8032	0.75	2.9328	-0.0902	0.8032
	1.0203	1	2.2323	-1.0299	1.0204
	1.2274	1.25			1.2275
	1.4187	1.5	1.4075	-2.1365	1.4187
Left Focus	1.4693	π/2	1.3333	-2.2361	1.4693
	1.7378	2	1.0177	-2.6596	1.7378
	1.9536	2.5	0.8348	-2.9049	1.9537
	2.0512	3	0.7672	-2.9956	2.0511
perigee	2.0562	π	0.7639	-3	2.0562

The position x=0 is a particular point, and its left focal angle can be obtained using the MCODE function **<)F0-**, the counterpart of **<)F0+** for the right-focal angle at the same point.

The velocity distribution is shown in the chart below, using equally spaced angles by an interval of 0.25 rad.



Program listing.

1	**LBL "VR+"**		46	RCL 03	θ
2	CF 04		47	COS	cos θ
3	*GTO 04*		48	*	r.cos θ
4	**LBL "VR-"**		49	RCL 00	*e*
5	SF 04		50	RCL 01	*a*
6	LBL 04		51	*	*ae = OF*
7	**XROM "?ab"**		52	FC? 04	
8	*"P=?"*		53	CHS	*x = r.cos θ -a.e*
9	PROMPT		54	+	*x = r.cos θ +a.e*
10	STO 06		55	ENTER^	*x*
11	1		56	ENTER^	*x*
12	RCL 00		57	RCL 02	*b*
13	**-/+**		58	*	*b.x*
14	SQRT		59	RCL 01	*a*
15	RCL 01		60	/	
16	*		61	X<>Y	*x*
17	PI		62	X^2	*x^2*
18	ST+ X		63	CHS	*-(x^2)*
19	*		64	RCL 01	*a*
20	RCL 06	period	65	X^2	a^2
21	/		66	+	*a^2 - x^2*
22	STO 04		67	SQRT	*sqrt(a^2- x^2)*
23	LBL 00		68	X#0?	
24	*"<)=? ("*		69	/	*b.x./ sqrt(a^2 - x^2)*
25	FC? 04		70	CHS	**y ' (x)**
26	*"/- +)"*		71	ATAN	*α*
27	FS? 04		72	RCL 03	θ
28	*"/- -)"*		73	X<>Y	
29	PROMPT		74	-	θ - atan(Y')
30	STO 03		75	FC? 04	
31	FC? 04		76	*GTO 04*	
32	**<)F-R-**	*left radius*	77	CHS	
33	FS? 04		78	PI	
34	**<)F-R+**	*right radius*	79	+	
35	STO 05	*r*	80	LBL 04	
36	1/X	*1/r*	81	SIN	
37	1	*function derivative*	82	RND	
38	RCL 00	*e*	83	X#0?	
39	+	*1+e*	84	ST/ 05	
40	*	*(1+e)/r*	85	RCL 05	*recall result*
41	RCL 01	*a*	86	*"V(<))="*	
42	*	*a.(1+e)/r*	87	ARCL X	
43	RCL 04	*V(a)*	88	PROMPT	*shos result*
44	*	*V(a).a.(1+e) / r*	89	*GTO 00*	*new position*
45	X<> 05	*r*	90	END	

Appendix. Other Auxiliary functions.

The tables below summarize the auxiliary functions by categories:

1. Input/output utilities

The input/output functions will skip the data entry if you press R/S without entering any numeric value. This is very convenient to use the current values in the data registers.

1.	***"?ab"***	Prompts for semi-axis.	New values or R/S for current	*See (*)*
2.	***"?<)-"***	Prompts for left-angles.	new Values or R/S for current	*See (*)*
3.	***"?<)+"***	Prompts for right-angles.	new Values or R/S for current	*See (*)*
4.	***"?P"***	Prompts for Orbital Period.	new Values or R/S for current	*See (*)*

(*) You can also call one of the Planet Data function at this point to populate the data registers with the orbital data corresponding to that planet. If you do, note that *the semi-axis are given in km*, and *the orbital period in days*.

2. Special Angle Values

Note that all of these functions **use the data registers holding the values of the semi-axis** (R01 and R02), and the eccentricity (R00). They are marked with an asterisk in the table.

(*)	**<)C0+**	Central angle for Focal = 90°	(a b,e) in R00-R02.	*Lifts Stack*
(*)	**<)C-R**	Central angle to Radius	{a,b,e} in R00-R02, α in X	*Saves α in LastX*
(*)	**<)C-XY**	Central angle to coordinates	{a,b,e} in R00-R02, α in X	*Saves α in LastX*
(*)	**<)F0+**	Right focal angle at x=0	(a,b,e) in R00-R02.	*Lifts Stack*
(*)	**<)F0-**	Left Focal angle at x=0	(a,b,e) in R00-R02.	*Lifts Stack*
(*)	**<)F>R+**	Right focal angle to radius	{a,b,e} in R00-R02, α in X	*Saves α in LastX*
(*)	**<)F>R-**	Left Focal angle to radius	{a,b,e} in R00-R02, α in X	*Saves α in LastX*

Warning: remember that the input angles for **<)F-R+** and **<)F>R-** are different; left or right focal angles respectively depending on the function. Also R(+) + R(-) does not equal "2a''

3. Planet Data functions.

Their name says it all: one for each of the planets plus another one for the Moon. They will silently populate the data registers with the appropriate data, not disturbing the stack or ALPHA. Here's the register mapping:

Register	R00	R01	R02	R06
Value	Eccentricity	Semi-major axis	Semi-minor axis	Orbital Period

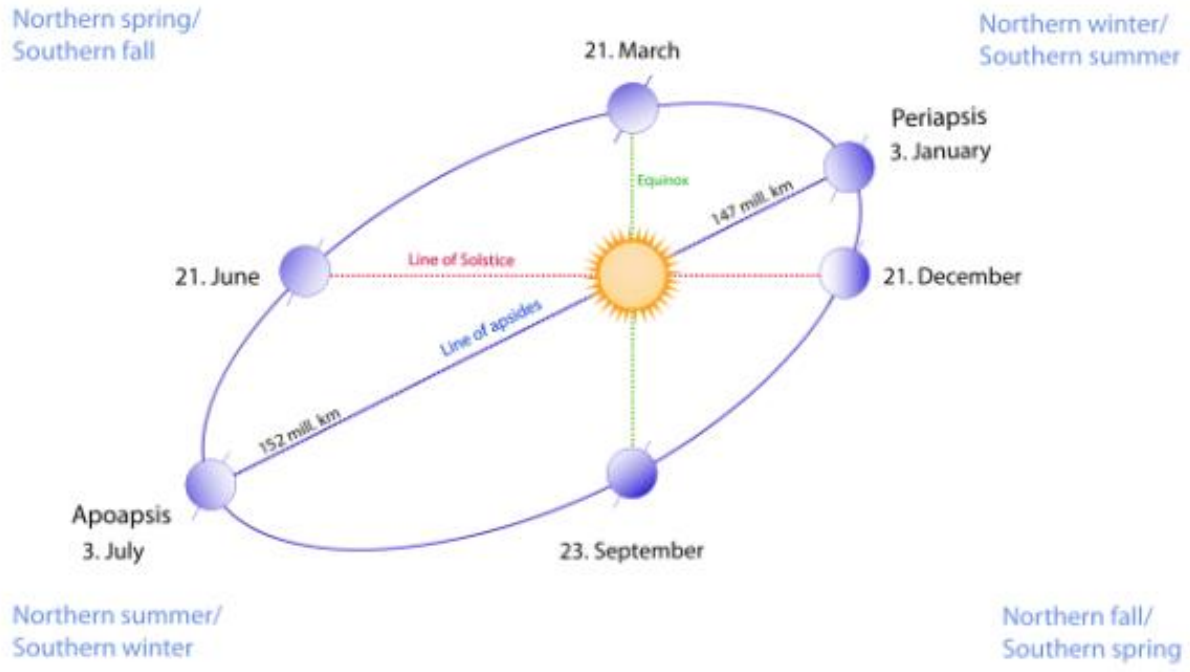
Warning: The values are stored in **km for the semi-axis**, and in **days for the orbital periods**. The semi-minor axis value is derived from the eccentricity and the semi-major axis.

Appendix. Solar System Orbits Data

The table below summarizes the ellipse data for the planets and the moon orbits. The Semi-minor axis is calculated from the semi-major and the eccentricity using: b = a.sqrt(1-e^2)

Name of Planetary Body	Orbital Elements		Orbital Period (years)
	Semimajor Axis (AU)	Eccentricity	
Mercury	0.3870993	0.20564	0.2408467
Venus	0.723336	0.00678	0.61519726
Earth	1.000003	0.01671	1.0000174
Mars	1.52371	0.09339	1.8808158
Jupiter	5.2029	0.0484	11.862615
Saturn	9.537	0.0539	29.447498
Uranus	19.189	0.04726	84.016846
Neptune	30.0699	0.00859	164.79132

The following diagram shows the relation between the line of solstice and the line of apsides of Earth's elliptical orbit. The orbital ellipse goes through each of the six Earth images, which are sequentially the perihelion (periapsis — nearest point to the Sun) on anywhere from January 2 to January 5, the point of March equinox on March 19, 20, or 21, the point of June solstice on June 20, 21, or 22, the aphelion (apoapsis — farthest point from the Sun) on anywhere from July 3 to July 5, the September equinox on September 22, 23, or 24, and the December solstice on December 21, 22, or 23.[7] The diagram shows an exaggerated shape of Earth's orbit; the actual orbit is less eccentric than pictured



Standard Gravitational Parameter and Orbital periods.

Two other functions are related to this chapter; and in fact to all sections as we have already mentioned. The function names are **(T(μ)** and **μ1/2**-yes, tricky to use but that's part of their charm ;-) Tip: use XROM in the OS/X module.

The first one **(T(μ)** calculates the period of an elliptical orbit with major semi-axis given, around a body with mass known (i.e. using its mass as data). The expression used is

$$T = 2\pi . a.\sqrt{(a/\mu)}$$

Example. If the mass of the sun is 1.9885 E30 kg, calculate the orbital period of the Earth around the sun. Make use of **EARTH** to retrieve the major semi-axis value to R01.

We type:

XEQ "EARTH"	=> writes EARTH data to R00-R06	
1.9885 E30, RCL 01, E3, *, XEQ "T(μ)"	=>31,557,670.11	period in seconds
3600, / , 24, /	=>365.2508115	period in days

The second one **μ1/2** derives the standard gravitational parameter μ for a body, when the orbital period and the semi-axis are known, using the expression:

$$\mu = \frac{4\pi^2 a^3}{T^2}$$

Example.- Derive the value of the standard gravitational parameter for the Sun using the Earth's orbit data.

We type:

365.25, ENTER, 3600, *, 24, * =>31,557,600.00
period in seconds

RCL 01, E3, * =>1.4959831 11
major semi-axis in m

XEQ "**μ1/2**" =>1.1520357 10
X^2 =>1.3271864 20

Which is close enough to the value found in the literature, as shown in the table on the right.

Body	μ ($m^3\ s^{-2}$)
Sun	$1.327\ 124\ 400\ 18(9)\times10^{20}$[1]
Mercury	$2.2032(9)\times10^{13}$[2]
Venus	$3.248\ 59(9)\times10^{14}$
Earth	$3.986\ 004\ 418(9)\times10^{14}$
Moon	$4.904\ 8695(9)\times10^{12}$
Mars	$4.282\ 837(2)\times10^{13}$[3]
Ceres	$6.263\ 25\times10^{10}$[4][5][6]
Jupiter	$1.266\ 865\ 34(9)\times10^{17}$
Saturn	$3.793\ 1187(9)\times10^{16}$
Uranus	$5.793\ 939(9)\times10^{15}$[7]
Neptune	$6.836\ 529(9)\times10^{15}$
Pluto	$8.71(9)\times10^{11}$[8]
Eris	$1.108(9)\times10^{12}$[9]

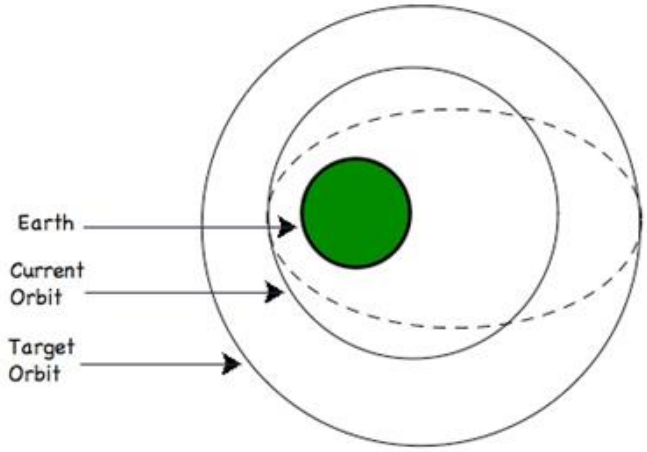
Delta-V Orbit Simulator. {**ORBIT**} *- from HP41 Physics Solutions Book.*

Delta-v (literally "change in velocity"), symbolized as Δv, as used in spacecraft flight dynamics, is a measure of the impulse that is needed to perform a maneuver such as launch from, or landing on a planet or moon, or in-space orbital maneuver. It is a scalar that has the units of speed. As used in this context, it is not the same as the physical change in velocity of the vehicle.

Delta-v is produced by reaction engines, such as rocket engines, and is proportional to the thrust per unit mass and the burn time. It is used to determine the mass of propellant required for the given maneuver through the Tsiolkovsky rocket equation. For multiple maneuvers, delta-v sums linearly.

Program Description

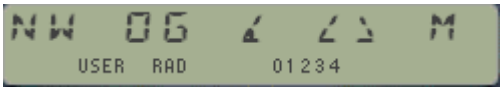
This program calculates orbit parameters from initial position and velocity data both for elliptical and hyperbolic orbits in a plane. It is also possible to move the point of interest to anywhere along the orbit and then recalculate orbit parameters.



The program is taken from the HP-41 Physics Solutions Book without fundamental changes. Only a top-level menu and parameter input prompts have been added for a more convenient use. This main menu is presented after the user has entered the initial orbital data:

Parameter	Position		Velocity	
Mass	Angle	Magnitude	Angle	Magnitude

The options menu offers the following choices:



- LBL A is used to input new initial conditions.
- LBL B is used to show the orbit Geometry
- LBL C is used to move to a new position in the old orbit
- LBL D is used to enter the Delta-V in angle and magnitude
- LBL E is used to present this options menu again.

Each of these options proceeds to prompt for the variables changed or added to the initial set. This approach makes it easier to use without access to the manual, so it's more fool-proof.

Once the calculations are done with the new data, the program enumerates the four new orbital conditions: R, <), V, and V<). You can return to the main menu at any time during this enumeration.

In option "B", the program will tell you the type of orbit created by the initial conditions or after the delta-V changes. The possible choices are Elliptical, Parabolic and Hyperbolic, as determined by the eccentricity value derived from the inputs.

Example. Execute a Hohmann transfer from a low-earth orbit to a high-earth orbit, using a delta-V of 2,300 m/s at the initial point, and another of 1,450 m/s at the transfer orbit. The initial conditions are listed below:

Parameter	Position		Velocity	
5.979 E24 kg	0°	7.1 E3km	90°	7.4 E3 m/s

With the calculator in DEG mode we type:

XEQ "ORBIT"		MASS=?
5.979 E24, R/S		R0=?
7.1 E6, R/S		R0∠)=?
0, R/S		|V0|=?
7.4 E3, R/S		V0∠)=?
90, R/S	->	R=7,100,000.00 ; ∠).=0.000
R/S		|V|=7,400.000 ; V∠)=90.000
R/S		NW 06 ∠) ∠) M

The first step is a burst from the low-earth orbit into a TRANSFER orbit:

XEQ D	V∠)?|V|=?(∠))	- amount and angle of change
90, ENTER^, 2300, R/S	R=7,100,000.00 ; ∠)=0.00	- same position
R/S	|V|=9,700.00 ; V∠)=90.00	- new velocity

After this burst the object is placed in an elliptical orbit that shares the same point with the original one, but with a different velocity due to the boost. The geometry of this transfer orbit can be reviewed using LBL B, as follows:

XEQ B	e=0.6743 ; ∠'=359.9976"	position in the new orbit -
R/S	RMIN=7,100,000.00	same as original orbit -
R/S	ELLIPSE	
R/S	a=21,800.710 ; b=16,098,449.73	
R/S	T=8.8941	

The next step is another burst when the body has reached the opposite position, i.e. determined by the current angle in the new orbit plus 180 degrees. Thus we first let the object travel to the new position, i.e. we "move" its position from the current one (which is why we used LBL B before).

XEQ C	∠) ' = ?
359.9976, ENTER^, 180, +	539.9976 - 180° are added to the first position! -
R/S	R = 36,500 ,420.27 ; ∠) = 179.9976
R/S	|V| = 1,886.7759 ; V∠) = 270.0009

And now we place the object into its FINAL orbit with another burst:

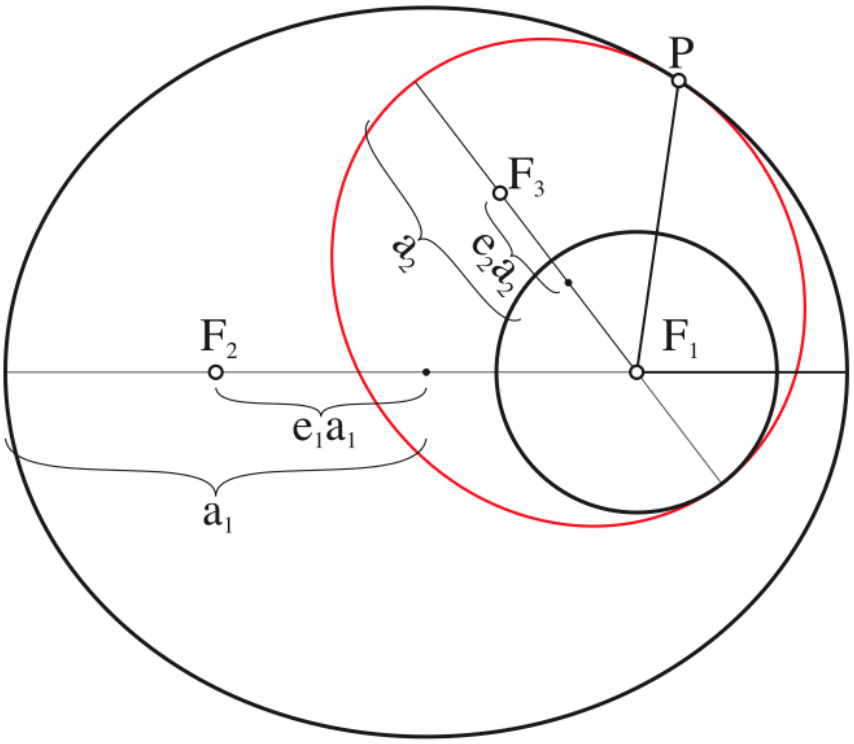
XEQ D	V∠) ⁄ |V| = ? ‹ ∠S ›
270.009, enter^, 1450, R/S	R = 36,500 ,420.27
R/S	∠) = 179.9976 – same position -
R/S	|V| = 3,336.7759
R/S	V∠) = 270.0044 – new velocity –

To see the high-earth orbit geometry we use LBL B as before:

XEQ B	e = 0.0186 ; ∠) ' = 180.3727
R/S	RMIN = 36,500 ,405.9
R/S	ELLIPSE
R/S	a = 37,192,989.60 ; b = 37,186,559.23
R/S	T = 19.8193

Remarks: it's important to note that two bursts are needed, both using the current velocity angle. The second burst occurs at 180 degrees opposite from the position after the first one.

The figure below shows a generic transfer situation, where the velocity bursts don't occur at opposite points but at other determined by the bi-elliptical conditions. This type is faster but requires more fuel.



Velocity Increments forHohmann Transfers.{ **+VH12**}

For most Earth orbits which were serviced by the Space Shuttle, the distancesfrom the earth was small in terms of Earth radii. For such orbits (even out to and beyond geosynchronous distance), the Hohmann transfer is the best transfer to use when transferring between circular coplanar orbits.

For transfers between circular coplanar orbits, the information usually given consists of the radii of the initial and final orbits. The information desired consists of the semi-major axis of the transfer orbit, the velocity increments at the ends of the transfer orbit, and the total velocity increment required for the transfer. The semi-major axis of the transfer orbit is given by:

$$a_T=(r_p+r_a)/2.$$

In order to calculate the required information, it is necessary to calculate the following four velocities:

The velocity in the initial circular orbit, $\sqrt{\frac{\mu}{r_1}}$

The velocity in the transfer orbit at initial orbit height, $\sqrt{\mu\left(\frac{2}{r_1}-\frac{1}{a_T}\right)}$

The velocity in the transfer orbit at final orbit height , $\sqrt{\mu\left(\frac{2}{r_2}-\frac{1}{a_T}\right)}$

The velocity in the final circular orbit , $\sqrt{\frac{\mu}{r_2}}$

The initial and final velocity increments are then given by the equations below:

$$\Delta V_1=\sqrt{\mu\left(\frac{2}{r_1}-\frac{1}{a_T}\right)}-\sqrt{\frac{\mu}{r_1}} \quad ; \quad \Delta V_2=\sqrt{\frac{\mu}{r_2}}-\sqrt{\mu\left(\frac{2}{r_2}-\frac{1}{a_T}\right)}$$

Where:

- r1 is the radius of the initial circular orbit
- r2 is the radius of the final circular orbit
- a(r) is the semi-mayor axis of the transfer orbit: a(r) = (r1 + r2) /2
- m is the gravitational parameter of the central body

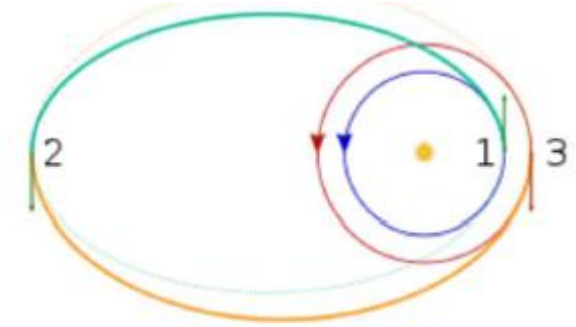
The MCODE function**+VH12** performs both velocity increments calculations. It expects the input parameters in the stack registers, as follows:

Register	Input	Function	Output
Z:	M = Mass of central body	**+VH12**	M = mass of central body
Y:	R1 = Initial orbit radius		ΔV1 = Velocity increment#1
X:	R2 = Final orbit radius		ΔV2 = Velocity Increment# 2
LastX:	-		R2 = Final Orbit radius

Bi-Elliptic tangential Transfers.{ **+VB123** }

A variation of the previous case, the bi-elliptic transfer requires two elliptical orbits that share the same apoapsis (or apogee in case of Earth orbits). This transfer requires three velocity increments, so in principle it'd appear the energy required should be larger. However, depending on the apogee distance used, *the total velocity increment may be smaller than the standard Hohmann transfer* - even if there's one more step involved.

Let r1 and r3 be the radii of the initial and final circular orbits. Let r2 be the common apogee of the two elliptical transfer orbits.



Let's now define:

a1 = (r1 + r2) / 2
a2 = (r3 + r2) / 2

The expressions are shown below. First leaving the initial orbit (prograde burst) and arriving to the final orbit (retrograde burst):

$$\Delta v_1 = \sqrt{\frac{2\mu}{r_1} - \frac{\mu}{a_1}} - \sqrt{\frac{\mu}{r_1}}. \qquad \Delta v_3 = \sqrt{\frac{2\mu}{r_3} - \frac{\mu}{a_2}} - \sqrt{\frac{\mu}{r_3}}.$$

And the intermediate bi-elliptical transfer:

$$\Delta v_2 = \sqrt{\frac{2\mu}{r_2} - \frac{\mu}{a_2}} - \sqrt{\frac{2\mu}{r_2} - \frac{\mu}{a_1}}.$$

Note that if r2 = r3 then ∆V3=0, and it becomes a standard Hohmann transfer.

The function calculates the three velocity increments at the same time. There are four input parameters required for this case, expected to be in the stack as shown in the table below:

Register	Input	Function	Output
T:	M = Central body Mass	**+VB123**	M = Central body Mass
Z:	R1 = Initial orbit radius		∆V1 = Velocity increment#1
Y:	R2 = Final orbit radius		∆V2 = Velocity Increment #2
X:	Rb = Elliptical orbit apogee		∆V3 = Velocity Increment #3
LastX:	-		Rb = Elliptical Orbit apogee

Important remarks: **The input values are expected in SI units,** i.e. meters for the radii and kg for the mass. This is because internally the gravitational constant is used with the SI values:

G = 6.6742800 E-11 N*M2/KG2

Note: The ALPHA register is used for scratch – *its previous contents will be lost.*

Example #1. Calculate the speed increments used in the ORBIT problem example if the data are the two orbit radii, as per the table below:

Parameter	LEO		GEO	
5.979 E24 kg	7.1 E3	km	36.5014059	km

We type: (don't miss the implicit unit requirements!)

5.979 E24, ENTER^, 36.5014059 E6, ENTER^, 7.1 E6
XEQ "+VH12" => 2,203.7880 m/s
X<>Y => 1,419.5187 m/s

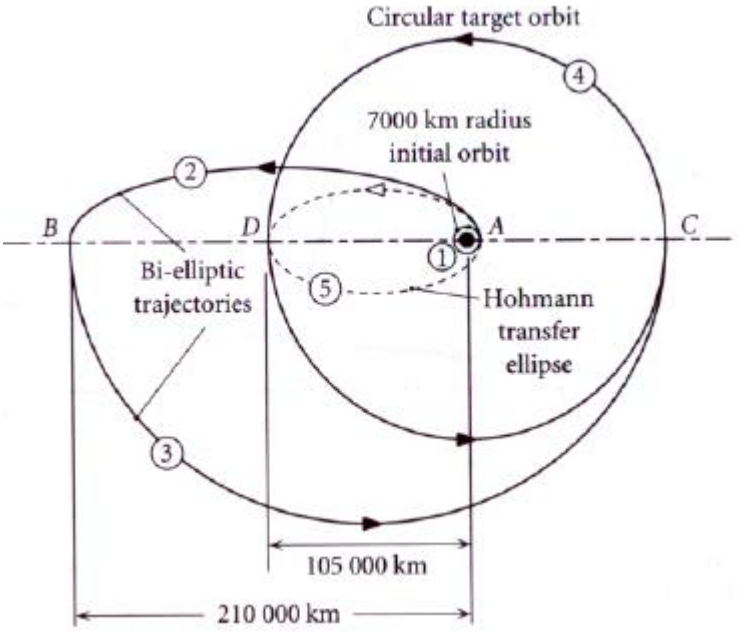
Note that these results are not quite the same exact values as the ones used in the problem, which may be just rounding errors, and the slightly different values for the Earth mass.

Example #2. Find the total delta-v requirement for a bi-elliptic Hohmann transfer from a geocentric circular orbit of 7,000 km radius to one of 105,000 km radius. Let the apogee of the first ellipse de 210,000 km. Compare the delta-v schedule and the total flight time with that for a standard Hohmann transfer ellipse. (Note: Earth's μ = 3.9860044 14).

First for the bi-elliptic transfer.

5.979 E24, ENTER^, 7 E6 , ENTER^,
105 E6, ENTER^, 210 E6

XEQ "+VB123" => 2,953.825526
RDN => 775.4013118
RDN => 301.5877266
RDN, => 5.9790000 24
RDN, +, + => 4,030.814565



Then for the simple transfer:

X<>Y (the mass was still there)
7 E6 , ENTER^, 105 E6

XEQ "+VH12" => 1,259.525314
RDN => 2,786.805728
+ => 4,046.331042

with TOF = T/2 = π. √(105 E6 + 7 E6)^3 / √8μ= √[(112 E6)^3 / 8μ] = 65,942.13822s

The counterpart are of course the times of flight:
In the first ellipse, TOF(1) = T1/2 = π. √[(205 E6 + 7 E6)^3 / 8μ]= π.√ [(212 E6)^3 / 8μ]
In the second one, TOF(2) = T2/2 = π. √[(205 E6 + 105 E3)^3 /8μ]= π. √[(310 E6)^3 / 8μ]

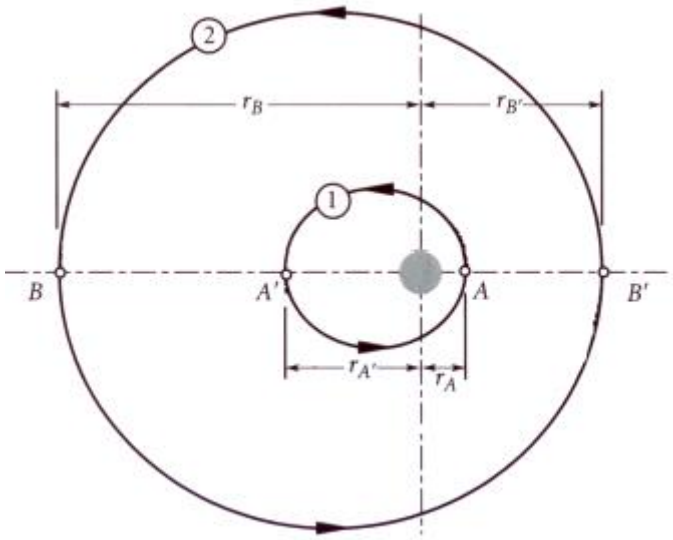
And the total TOF = TOF(1) + TOF(2) = p.(54,662.55282 + 96,655.96462) = 475,381.1427 s

i.e. greater by the factor: 475,381.1428 - 65,942.13822 = 409,439.0046 s = 4.738877367 days

Transfer between Coaxial Elliptical Orbits.{**+VEa12**, **+VEb12**}

We know that planet orbits aren't circular but elliptical – although their eccentricities are not large. Therefore, using this approach would appear to be more appropriate, evenif the accuracy increase isn't that significant in a limited 10-digit mantissa world.

This type of transfers has two possible trajectories, depending on whether it is initiated at the apogee or at the perigee of the initial orbit – which are the two points in the orbit where the velocity is perpendicular to the radius; the required condition for a tangential velocity increment goal.



The perigee and apogee of the two orbits are labeled in the sequence of transfer: From A to B (perigee-1 to apogee-2), or from A' to B' (apogee-1 to perigee-2).

The expressions for the velocity increments for both cases are as follows:

1. Starting at the perigee:

$$\Delta v)_3 = \Delta v_A + \Delta v_B$$

$$= \left(\sqrt{\frac{\mu}{r_A}\frac{2r_B}{r_A + r_B}} - \sqrt{\frac{\mu}{r_A}\frac{2r_{A'}}{r_A + r_{A'}}} \right) + \left(\sqrt{\frac{\mu}{r_B}\frac{2r_{B'}}{r_B + r_{B'}}} - \sqrt{\frac{\mu}{r_B}\frac{2r_A}{r_A + r_B}} \right)$$

2. Starting at the apogee:

$$\Delta v)_{3'} = \Delta v_{A'} + \Delta v_{B'}$$

$$= \left(\sqrt{\frac{\mu}{r_{A'}}\frac{2r_{B'}}{r_{A'} + r_{B'}}} - \sqrt{\frac{\mu}{r_{A'}}\frac{2r_A}{r_A + r_{A'}}} \right) + \left(\sqrt{\frac{\mu}{r_{B'}}\frac{2r_B}{r_B + r_{B'}}} - \sqrt{\frac{\mu}{r_B}\frac{2r_{A'}}{r_{A'} + r_{B'}}} \right)$$

Warning: these formulas are not verified with a second source, so they may contain transcription errors. Also it's not entirely clear they are dimensionally correct ?

There are five input parameters required for this case. The function expects the radii in the stack, and the mass of the central object in the LastX register. Use SF 00 / CF 00 to toggle the cases.

Register	Input	Function	Output
T:	A1 = initial orbit apogee	**+VEa12** **+VEb12**	M = Central body Mass
Z:	B1 = Initial orbit perigee		A1 = initial orbit apogee
Y:	A2 = final orbit apogee		ΔV1 = Velocity increment#1
X:	B2 = Final orbit perigee		ΔV2 = Velocity Increment#2
LastX:	M = Central body Mass		B1 = Final orbit perigee

Example. A spacecraft is in a 480 km by 800 km earth orbit (orbit 1). Determine the most efficient transfer from orbit 1 to another elliptical orbit of 1000 km by 7 000 km, and the required delta-v.

First for the perigee jumps, we type:

5.97219 E24, STO L		- mass of the Earth
480 E3, ENTER^, 8 E5, ENTER,		- perigee and apogee of "lower" ellipse
1 E6, ENTER^, 7 E6		- perigee and apogee of "upper" ellipse
XEQ "+VEa12"	=>	5,269.405 16 m/s
X<>Y	=>	35,376.6630 1 m/s

And the transfers at the apogee:

RDN, RDN, 5.97219 E24, STO L		
CLX, 1 E6, ENTER^, 7 E6		
XEQ "+VEb12"	=>	145,684.9 179 m/s
X<>Y	=>	28 1,982.0250 m/s

Comparing these results, it'd appear initiating the transfer from the perigee is considerably more efficient. This is logical because the velocity at the apogee is always smaller than at the perigee, as determined by the relationship:

$$Va = Vp\,[(1-e)/(1+e)]\ ; \quad \text{thus: } Va/Vp = (1-e)/(1+e) < 1$$

On the other hand, this assumes the numbers correct which as mentioned before, may not be true.

Note that these velocity increments are much larger than those used in circular orbits. This may be yet another reason why elliptical orbits are avoided for satellite placement – besides the many other facts that complicate calculations and aren't well suited for earth communications.

Orbital Rendez-vous. {**CORV** , **CPRV** }

Orbital transfer becomes more complicated when the object is to rendezvous with or intercept another object in space: both the interceptor and the target must arrive at the rendezvous point at the same time. This precision demands a phasing orbit to accomplish the maneuver. A phasing orbit is any orbit that results in the interceptor achieving the desired geometry relative to the target to initiate a Hohmann transfer.

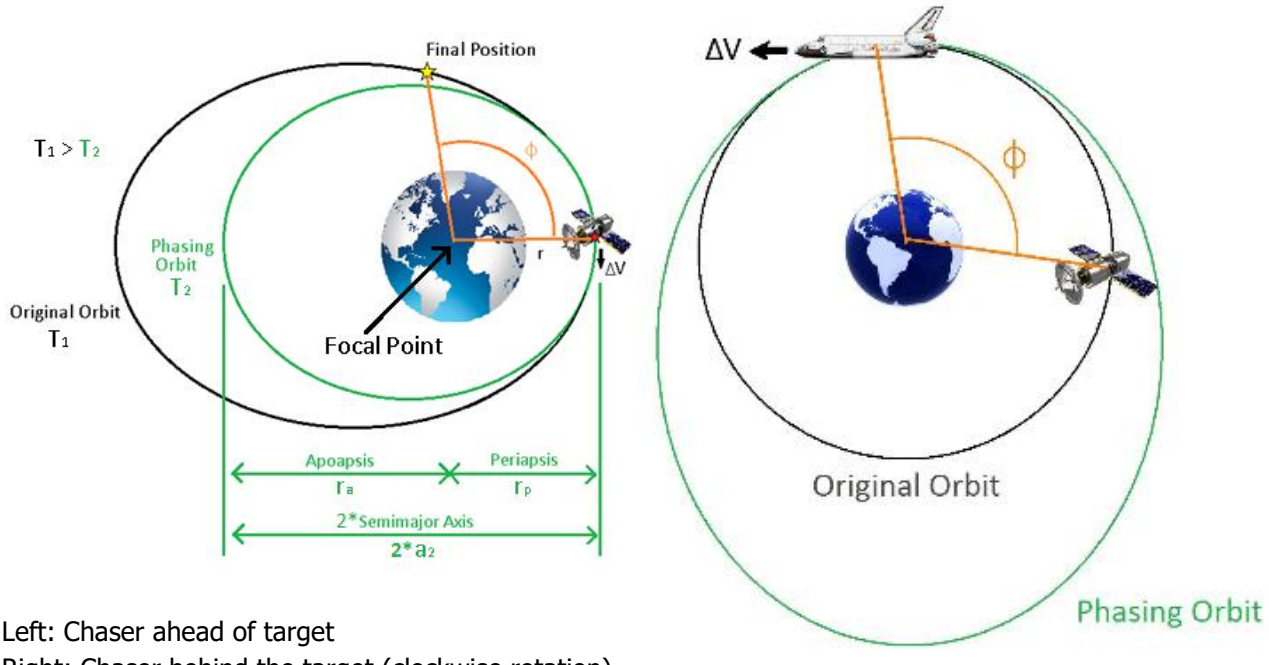
Co-orbital Rendezvous.

This is the easier of the two cases. Both interceptor and target are in the same orbit, at a relative position given by the initial phase angle Φo measured **from the interceptor** *in the direction of rotation*. Therefore, the initial angle will be positive if the interceptor is behind the target, and negative if it is ahead of it. This makes an important difference in the velocity increments to apply.

- If the target is ahead of the chaser, the latter needs to reduce its speed to move into a smaller elliptical orbit (inside the circular one, tangential to the transfer point) – with a smaller period, and thus it'll be able to catch up with the target in the next revolution at the tangent point. The period of such elliptical orbit must equal the time taken by the target to travel to the rendezvous point, i.e. to move an angle:Φt = 2π -Φo .

- If the target is behind the chaser, the latterneeds to increase its speed to move into an outer elliptical orbit (with a positive velocity increment) into an elliptical orbit with greater period than the circular one, to allow the target to move a travel angle Φt = 2π + Φo up to the point of tangency (one revolution later).

- In either case the chaser requires a second velocity increment equal to the first one but in opposite direction, to return to the original circular orbit. The rendezvous can be initiated at any time, there's no need to wait for a "launch window" to occur.



Left: Chaser ahead of target
Right: Chaser behind the target (clockwise rotation)

For circular orbits, the phasing elliptical orbit semi-major axis is easily calculated as:

$$a' = R\,[(2\pi+\Phi o)/2p\,]^{(3/2)}$$

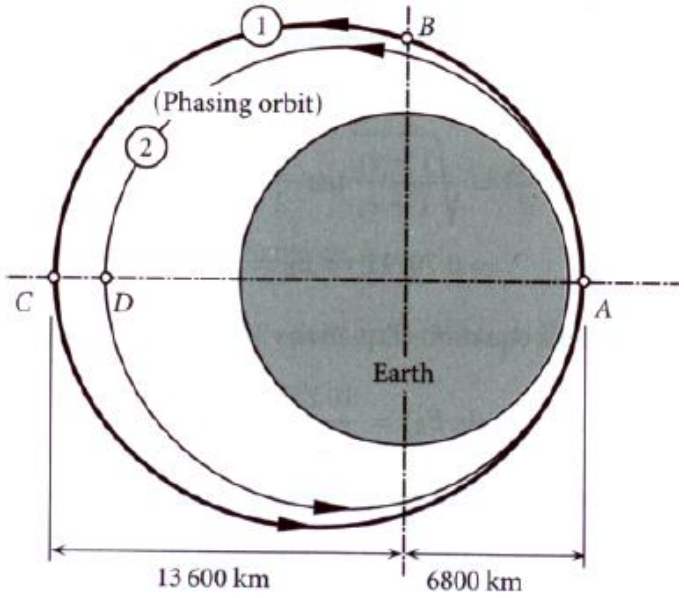
and the TOF is the full period of such orbit:

$$TOF = T' = 2\pi.\sqrt{(a'^3/\mu)}$$

For elliptical orbits one must resort to the Mean anomaly instead – or use an area-based approach. In that case the expression for the semi-major axis of the phasing phase is:

$$a' = R\,[(2\pi+Mo)/2p\,]^{(3/2)},\ \text{with:}\ Mo = Eo - e.\sin(Eo)\ \text{, etc.}$$

Example. Spacecrafts at A and B are in the same elliptical orbit 1. At the instant shown, the chaser vehicle at A executes a phasing maneuver so as to catch the target S/C back at B after just one revolution of the chaser's phasing orbit 2. What is the required total delta-v.



Solution: The driver program **CORV** does all the work. We'll use:2a= (ra + rp)_, and: $b = \sqrt{(ra.rp)}$

XEQ "CORV"	MASS = ?
5.972186 E24, R/S	αTb = ?
13600, ENTER^, 6800, + , 2 , / , ENTER^	10,200,000.00
13600 E3, ENTER^, 6800 E3, * , SQRT	9,616,652.224
R/S	∠)Θ = ? (CST)
PI, 2, / , R/S	α' = 84 19,9 14.484
R/S	dV 1 = -779.77893 1
R/S	TOF = 7,689.046750

Note the very small retrograde increment of velocity in direction contrary to the movement. Note also that it must be repeated as prograde (in the same direction this time) at the rendezvous point to return to the initial orbit.

Co-orbital Rendezvous Program Listing:

1	**LBL "CORV"**		30	Y^X	
2	CF 00	*default: circular*	31	RCL 01	*a*
3	*"MASS=?"*		32	*	
4	PROMPT		33	STO 06	*a'*
5	STO 05	*M*	34	*"a' ="*	
6	**XROM "?ab"**		35	ARCL X	
7	RCL 00	*e*	36	AVIEW	
8	X#0?	*elliptical?*	37	AVIEW	
9	SF 00	*yes, flag case*	38	RCL 05	*M*
10	LBL 00		39	X<>Y	*a'*
11	*"<)=? (C>T)"*	*phase angle*	40	ST+ X (3)	*2a'*
12	PROMPT		41		
13	FC? 43		42	RCL 01	*a*
14	D-R		43	-	*2a' - a*
15	FS? 00	*elliptical?*	44	RCL 01	*a*
16	**T>E**	*yes, Eccectric anomaly*	45	X<>Y	*put in expected order*
17	FS? 00	*elliptical?*	46	**+VH12**	
18	**E>M**	*yes, Mean anomaly*	47	STO 04	
19	STO 03	*Φ o*	48	*"dV1="*	
20	CHS		49	ARCL X	
21	PI		50	AVIEW	
22	ST+ X (3)	*2 π*	51	RCL 05	*M*
23			52	RCL 06	*a'*
24	+	*Φ o + 2 π*	53	**T(μ)**	
25	LASTX	*2 π*	54	*"TOF="*	
26	/	*(Φ o + 2 π)/2π*	55	ARCL X	
27	X^2		56	PROMPT	
28	3		57	*GTO 00*	
29	1/X		58	END	

Note. Using the area-based approach instead of the mean anomaly could also be possible, once we can establish the central angle corresponding to the initial phase angle (which is measured as a true anomaly).

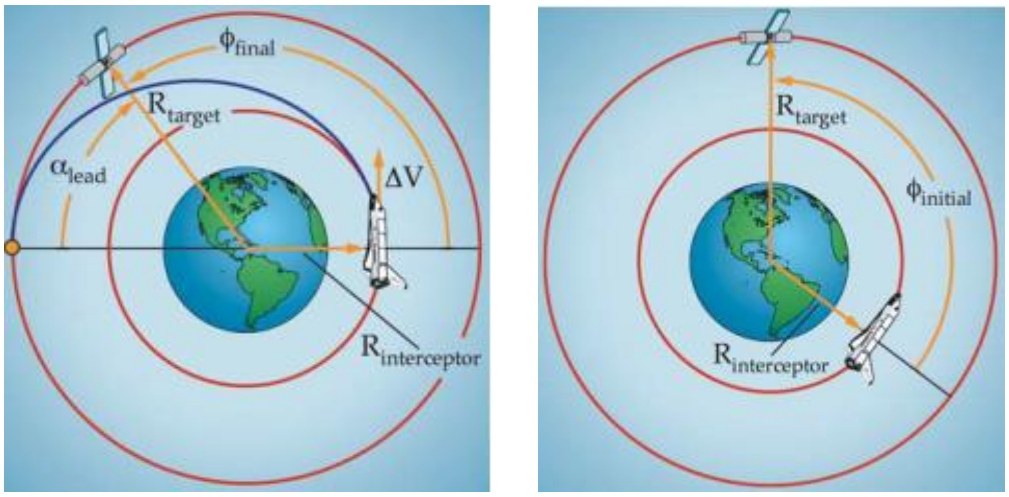
Let A(Φt) be the focal sector area swept by the target. Then the semi-axis of the chaser's phasing orbit would be given by:

$$a' = at * [A(\Phi t) / \pi.a.b\ ]^{(1/3)}$$

Unfortunately no more room was available in the module, so this option is not included.

Co-planar Rendezvous.

If the initial and final orbits are circular, coplanar, and of different sizes, then the phasing orbit is simply the initial interceptor orbit. The interceptor remains in the initial orbit until the relative motion between the interceptor and target results in the desired geometry. At that point, we would inject the interceptor into a Hohmann transfer orbit. Chances are the initial relative positions of target and chaser are not the required ones for the transfer, so a wait time will be required.



Left: Launching condition. Right: Initial condition

The required conditions specify a lead angle α between the target and the rendezvous point, determined by the time taken by the chaser to move along the Hohmann transfer. Only in that circumstance both object will meet at the desired point.

The TOF of the Hohmann section is given by the known geometry (radii for both circular orbits Rc and Rt), as half the period of the elliptical phasing orbit:

$$TOF = \pi . \sqrt{(a` / \mu)} \; ; \text{ with } a` = (Rc + Rt)/2$$

and this time must equal the one taken by the target to travel the lead angle α. If it travels with a known angular velocity ωt, then the condition is: $TT = \alpha / \omega t = TOF$, hence:

$$\alpha = \pi . \omega t . \sqrt{(a` / \mu)} = \pi . \sqrt{(\mu / Rt\text{^}3)} . \sqrt{(a` / \mu)} = \pi . \sqrt{(a`/Rt)\text{^}3}$$

Next, we need to determine the wait time, i.e. how long it'll take for the chaser and target to be in the appropriate position; which is obviously dependent on their initial relative position. Say initially they are situated an angle Φo apart from one another, and let ωt, ωc be the angular velocities of target and chaser in their circular orbits – i.e. their relative velocity is $(\omega c - \omega t)$

The situation we want is when simultaneously the chaser's lead angle is π, and the target's lead angle is α; that's to say the angle between them must be $\pi - \alpha$. Thus, the traveled angle until that point is: $\Phi t = (\pi - \alpha) - \Phi 0$; and the wait time is simply such traveled angle over the relative velocity:

$$WT = [(\pi - \alpha) - \Phi 0 \,] / (\omega c - \omega t) \; ; \quad \text{or } WT = [- (\pi + \alpha) + \Phi 0 \,] / (\omega c - \omega t)$$

Example. Determine the transfer conditions for an automated repair spacecraft in LEO with Ri = 6570 km to rendezvous with a disabled target spacecraft in a geosynchronous orbit with Rt = 42160km, if the initial angle between the two spacecrafts is 180.

Solution - The driver program CPRV does all the work:

XEQ "CPRV"	MASS=?
5.972186 24, R/S	R(C)=?
6570 E3, R/S	R(T)=?
42160 E3 , R/S	∠)Θ=?
180, R/S	TOF= 19,029.72383
R/S	WT= 21 H 42 MIN 32.29 S
R/S	dV 1= 2,405.296636
R/S	dV 2= 1,459.387732

Co-Planar w/ Circular orbits. Program Listing:

1	LBL "CPRV"		44	PI	
2	*"MASS=?"*		45	*	*lead angle*
3	PROMPT		46	CHS	$-\alpha$
4	STO 05	*M*	47	RCL 00	ϕo
5	*"R(C)=?"*		48	-	$-\alpha-\phi o$
6	PROMPT		49	PI	
7	STO 04	*R1*	50	+	$\pi-\alpha-\phi o$
8	*"R(T)=?"*		51	X<=0?	
9	PROMPT		52	*GTO 00*	
10	STO 03	*R2*	53	PI	
11	RCL 04	*R1*	54	ST+ X	
12	+		55	-	
13	2		56	LBL 00	
14	/	*a' = (R1+R2)/2*	57	RCL 03	*Rt*
15	STO 06		58	3	
16	LBL 00		59	Y^X	
17	*"<)0=?"*		60	1/X	
18	PROMPT		61	SQRT	ωt
19	FC? 43		62	RCL 04	
20	D-R		63	3	
21	STO 00	ϕo	64	Y^X	
22	RCL 05	*M*	65	1/X	
23	RCL 04	*R2*	66	SQRT	ωc
24	RCL 03	*R1*	67	-	$\omega t - \omega c$
25	**+VH12**		68	/	$\pi-\alpha-\phi o$ / ($\omega t - \omega c$)
26	STO 02	*dV2*	69	RCL 07	*T*
27	X<>Y		70	RCL 06	*a'*
28	STO 01	*dV1*	71	**μ1/2**	
29	RCL 05	*M*	72	X<>Y	
30	RCL 06	*a' = (R1+R2)/2*	73	RDN	μ^1/2
31	**T(μ)**		74	/	*Wait*
32	STO 07		75	STO 08	
33	2		76	*"WT="*	
34	/		77	ARCL X	
35	"TOF"=		78	AVIEW	
36	ARCLX		79	*"dV1="*	
37	AVIEW		80	ARCL 01	
38	RCL 06	*a'*	81	AVIEW	
39	RCL 03	*Rt*	82	*"dV2="*	
40	/		83	ARCL 02	
41	3		84	PROMPT	
42	Y^X		85	*GTO 00*	
43	SQRT		86	END	

Projectile Trajectory. {**PRJTL**} *- by PoulKaarup.*

This program can be used as an applet to calculate the different variables of a projectile motion. The initial conditions are the position and velocity (both in magnitude and angle) at the point of launch.

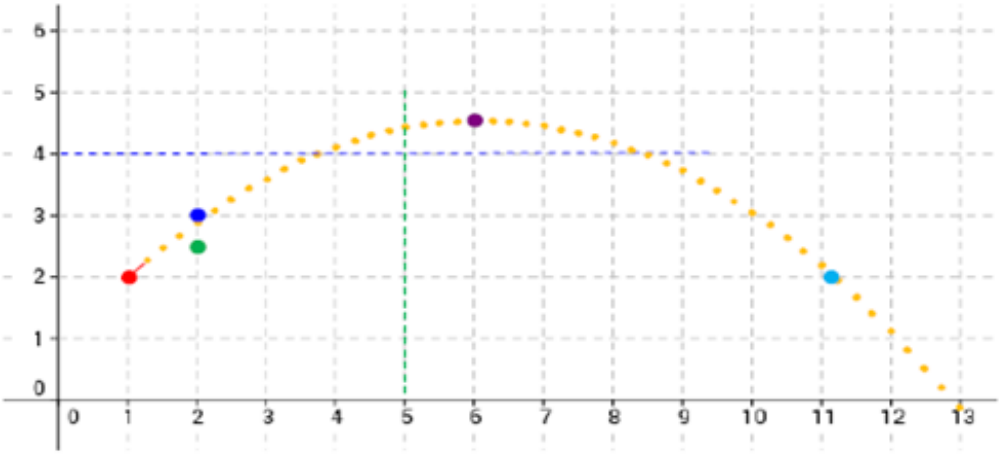
Then a menu of choices is displayed, *where the unknown variablesare shown* - as follows:

- LBL A is used to find the altitude (y) at a given distance (x)
- LBL B calculates the distance and time for a given altitude (y)
- LBL C finds the x,y position for a given time (t)
- LBL D finds the angle required to hit a target x,y and the time to get there
- LBL E finds the velocity required to hit a target x,y and the time to get there.

Note that LBL D and E will prompt for new values for the initial variables

On each of these options the program will prompt for the known variables needed for the calculation of the unknown. Simply answer the prompts and press R/S to proceed.

Example. A projectile is launched with the following initial conditions: from (xo,yo) = (1, 2) – red dot - and with a velocity Vo= 10 m/s, with an angle <(o = 45 deg from the horizon line.



Find the altitude at x=5m; the distance for y=4m, and the position at t=1s.

XEQ A	X = ?
5, R/S	T = 0.5076 ; Y = 4.4288 (green line)
XEQ B	Y = ?
4, R/S	T = 0.3867 ; X = 3.7340 (blue line)
XEQ C	T = ?
1, R/S	X = 8.0711 ; Y = 4.1611 (green dot on the far right)

Find now the angle to hit the "blue" target at (2, 2) and the time to get there:

XEQ D	X?Y=? (blue dot)	
2, ENTER^, 3, R/S	∠=47.9726 ;	T=0.1494
R/S	∠=87.0274 ;	T= 1.9283
XEQ E	X?Y=?	
2, ENTER^, 2.5 , R/S	V0=9.8658;	T= 1.9545 ;

Lastly, find the launch velocity to hit the "green" target located at (x, 2.5), and the time to get there:

XEQ E	X?Y=?	
2, ENTER^, 2,5, R/S	V0=9.8658 ;	T= 1.9545