# [VA] SRC #012f - Then and Now: Angle

1st May, 2023, 18:00 (This post was last modified: 9th May, 2023 06:40 by Valentin Albillo.) **Post: #1**

**Valentin Albillo**
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

**[VA] SRC #012f - Then and Now: Angle**

**Hi, all,**

**Welcome** to the **6th and *last* part** of my ***SRC #012 - Then and Now***, where I'm showing that advanced vintage *HP* calcs which were great problem-solvers back **THEN** in the 80's are **NOW** still perfectly capable of solving recent non-trivial problems intended to be tackled using modern *PC*s, never mind ancient handheld calcs.

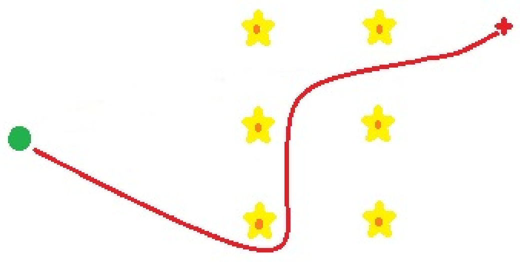
Now, try and solve the last problem using your vintage *HP* calcs while abiding by this one rule, *s'il vous plait*:

Use any **VINTAGE HP CALC** (physical/virtual,) coding in either **RPN**/*Mcode*, **RPL**/*SysRPL* or **HP-71B BASIC**/*FORTH*/*Assembler*. **NO CODE PANELS**.

## Problem 6: Angle

Once ***P1 (Probability)***, ***P2 (Root)***, ***P3 (Sum)***, ***P4 (Area)*** and ***P5 (Roots)*** are over, now's the turn for ***P6 (Angle)***, the last and allegedly the hardest of the lot, which deals with *optimization*, namely:

Six stars of unit mass are ***fixed*** at points *(2, -1), (2, 0), (2, 1), (3, -1), (3, 0)* and *(3, 1)*. A planet of unit mass starts at *(0, 0)* with unit velocity in direction *Angle* (in radians, counterclockwise from the *X* axis,) and travels according to *Newton*'s gravitational law, i.e. under a force of magnitude $(d_i)^{-2}$ from each of the six stars, where $\boldsymbol{d_i}$ is the distance to each star.



Write a program to compute the value of *Angle* which makes the planet reach the point *(**4**, **1**)* in the ***shortest time*** (a fake path is depicted above) and output the *Angle* and corresponding minimum *time*. Assume the gravitational constant *G* is ***1***.

Your program should take no inputs and must produce at least *4-5 correct digits* (give or take a few *ulp*), the faster the better.

In a week I'll post my *original solution* (a straightforward *12-line* **HP-71B** program that does the job) plus comments on my solving strategy and possible enhancements, but first let's see your very own clever solutions. It's your last chance !

**V.**
*Edit: corrected two typos.*

3rd May, 2023, 11:05 **Post: #2**

**Fernando del Rey**
Junior Member

Posts: 19
Joined: Dec 2013

**RE: [VA] SRC #012f - Then and Now: Angle**

Reading Valentin’s original post where he mentions that this last problem of the SRC #12 series is “the last and allegedly the hardest of the lot”, I felt at first discouraged. However, perhaps because I feel more at ease with physics and engineering topics than with advanced mathematics as required in some of his previous problems, I decided to give this one a try.

After some effort to put in paper the equations of motion of the planet in the X-Y plane (position, velocity and acceleration vectors), I realized that they were greatly simplified working with complex numbers. Calculating the gravitational force (i.e., acceleration vector) exerted on the moving planet by the six fixed stars requires just one line of BASIC code in the HP-71B to add the gravitational force of each star (function FNA in line 60 in the program below). Working with complex numbers (which requires the Math ROM), the vectorial operations become extremely simple to implement.

The trajectory is a second-order ordinary differential equation:

p”=f(p) , where f(p)=Σ_(i=1)^6▒(s(i)-p)/|s(i)-p|^3

Which can be reduced to a set of two first-order equations:

p’=v

v’=f(p), with boundary conditions p0=[0,0] , v0=[Cos(α),Sin(α)]

Which can be solved using the Runge-Kutta method (FNP in lines 80 to 150).

The problem is that with the Runge-Kutta method we obtain the trajectory of the planet given the position and velocity vector at one end of the trajectory, the starting point (0,0). The position at the target point (4,1) is unfortunately not a boundary condition.

The approach I have followed is to make a sweep of the velocity angle at the starting point, to obtain an ending point where the trajectory crosses the vertical line X=4 (loop starting at line 190).

If the distance of the crossing point to the target point is small enough (less than 0.5), then the program initiates a fine search of the root angle using the FNROOT function of the Math ROM.

For the calculation of the planet trajectory with the Runge-Kutta method I have tried different values of Delta-T (variable H in the program). A smaller Delta-T gives a more accurate result at a cost of longer execution time. A larger Delta-T gives a much less accurate result, particularly if the trajectory travels very close to one of the stars where the gravitational force if very big. I have finally opted for a Delta-T value of 0.025 for the coarse sweep, and a smaller Delta-T of 0.01 for the fine search of the root with FNROOT.

Also, in order not to consume time searching for trajectories that take too long to reach the target point, the trajectory calculation is stopped after the travel time reaches a value of T9=2.5 time units. I started using a larger value of T9, but finally reduced it to 2.5 time units after I saw that the shortest travel trajectory would take less than 2 time units.

In my sleuthing process I have been able to find six starting angles that produce a trajectory arriving at the target point (4,1). These six angles are in the range from -0.5 radians to +0.75 radians (loop staring at line 190). I have scanned the full range from -1 to +1 radians but could not find any additional roots. There may be roots (i.e., trajectories arriving to the target point) for other angles beyond this range, but the travel time will surely be longer, so they are not of interest to us.

Here’s my code for the 71B:

```
10 DESTROY ALL @ OPTION BASE 1 @ RADIANS
20 COMPLEX P,V,A,S(6),Z,M1,M2,M3,M4,K1,K2,K3,K4
30 INTEGER I,J @ REAL Z0,D,H,L,E(50),U(50)
40 DATA (2,-1),(2,0),(2,1),(3,-1),(3,0),(3,1),(4,1)
50 ! Function to calculate acceleration vector at point P
60 DEF FNA(P) @ A=0 @ FOR I=1 TO 6 @ A=A+(S(I)-P)/ABS(S(I)-P)^3 @ NEXT I @ FNA=A @ END DEF
70 ! Function to calculate trajectory using Runge-Kutta method
80 DEF FNP(D) @ T=0 @ V=(COS(D),SIN(D)) @ P=(0,0)
90 M1=H*V @ K1=H*FNA(P) @ M2=H*(V+K1/2) @ K2=H*FNA(P+M1/2)
100 M3=H*(V+K2/2) @ K3=H*FNA(P+M2/2) @ M4=H*(V+K3) @ K4=H*FNA(P+M3)
110 P=P+(M1+M2+M2+M3+M3+M4)/6 @ V=V+(K1+K2+K2+K3+K3+K4)/6
120 T=T+H @ IF REPT(P)<Z0 AND T<T9 THEN 90
130 IF T>=T9 THEN L=0 @ GOTO 150 ! If time is too long, exit
140 L=IMPT(P)-IMPT(Z)-(REPT(P)-REPT(Z))*(IMPT(V)/REPT(V)) @ T=T-(REPT(P)-Z0)/REPT(V)
150 FNP=L @ END DEF ! L=distance from target vertical line, T=travel time
160 ! Main program SRC #12f
170 T0=TIME @ T5=MAXREAL @ T9=2.5 ! T9 is max trajectory time
180 FOR I=1 TO 6 @ READ S(I) @ NEXT I @ READ Z @ Z0=REPT(Z) ! Read star and target positions
190 FOR D5=-.5 TO .75 STEP .0125 ! Angle search range and step
200 H=.025 @ DISP D5 ! Delta-T for coarse trajectory
210 IF J=0 THEN 230
220 IF D5<E(J) THEN 320 ! If root angle was already found, skip
230 IF ABS(FNP(D5))>.5 THEN 320 ! If too far from target, skip
240 IF T>=T9 THEN 320 ! If time is too long, skip
250 H=.01 @ DISP D5;"*" ! Delta-T for fine trajectory
```

```
260 D0=FNROOT(D5,D5,FNP(FVAR)) ! Fine search of root
270 IF T>=T9 THEN 320 ! If time is too long, skip
280 IF J=0 THEN 300
290 IF ABS(D0-E(J))<1.E-11 THEN 320 ! If root angle was already found, skip
300 J=J+1 @ E(J)=D0 @ U(J)=T ! Save root angle and trajectory time
310 IF T<T5 THEN T5=T @ J5=J ! Look for shortest trajectory time
320 NEXT D5 @ T0=TIME-T0
330 DISP "Angle= ";E(J5);"Time= ";T5;T0
```

The result I obtain with Emu71/Win is:

**Angle= -.27488204751 Time= 1.91608960085 298.11**

The trajectory with the shortest travel time is with a starting angle of -0.27488204751 radians and its travel time is 1.91608960085 time units.

Execution time with Emu71/Win is 298.11 seconds (on a physical 71B this would take around 80 hours!).

The angles and travel times for each of the six trajectories can be found in vectors E and U:

*Angle / Time:*
```
-.461225580816 2.15264790583
-.27488204751 1.91608960085
-.196990737712 1.99997968861
.328560457304 2.02378675724
.55128583185 1.91762537401
.732203465959 2.06532723381
```

EMAIL PM FIND QUOTE REPORT

3rd May, 2023, 21:19 **Post: #3**

**EdS2**
Senior Member

Posts: 517
Joined: Apr 2014

**RE: [VA] SRC #012f - Then and Now: Angle**

Great explanation!

EMAIL PM FIND QUOTE REPORT

4th May, 2023, 20:31 (This post was last modified: 4th May, 2023 21:36 by J-F Garnier.) **Post: #4**

**J-F Garnier**
Senior Member

Posts: 790
Joined: Dec 2013

**RE: [VA] SRC #012f - Then and Now: Angle**

I'm impressed by the performance and quality of the solution from Fernando.
He found all the 6 shortest paths, as far as I can verify them, and some were not easy to catch.
There are probably many more, but much more complex and longer.

At first, I was not sure that integrating the motion equations would work well, because the trajectories are likely to be chaotic. Small changes in the initial angle may produce radically different trajectories, also going too close to a star may produce large acceleration, so integration may not be accurate enough.

So my analysis was looking at other aspects.
One aspect was to consider that the particle (the planet) was free, so the sum of its kinetic energy $v^2/2$ and potential energy (sum of the -1/di quantities for each star at distance di) must be constant.

This gives the value of the final speed:
Initial energy (vi=1 speed) = 1/2 - ( 2/sqr(5) + 1/2 + 2/sqr(10) + 1/3 ) = -1.860
Final energy (vf speed) = 1/2*vf^2 - ( 1/1 + 1/2 + 1/sqr(2) + 2/sqr(5) + 1/sqr(8) )
we get final speed vf = 1.786
Also, since the energy is negative, the planet is bound to the star system and can't escape and may, soon or later, cross the position (4,1) possibly many times.

However, I didn't come to anything really useful for a full solution.

So I was wrong, motion integration was the way to go, and my (small) contribution here will be a few optimizations of Fernando's program.

I noticed that some time consuming fine searches were useless.
I added these 2 tests to avoid them:

- a consistency check of the final speed against the value I calculated above, this protects against some cases where the trajectory goes too close to a star and the integration becomes completely wrong:

**235** IF ABS(V)>2 THEN 320 ! If abnormal final speed, skip

- a better test to avoid testing too close trajectories, that can't be realistic:

220 IF D5<E(J)**+.02** THEN 320 ! If too close from a root angle already found, skip

With these changes, I obtained the same results as Fernando but in 30% less time (223s instead of 326s on my Emu71/Win system).

J-F

EMAIL PM WWW FIND QUOTE REPORT

7th May, 2023, 17:03 **Post: #5**

**J-F Garnier**
Senior Member

Posts: 790
Joined: Dec 2013

**RE: [VA] SRC #012f - Then and Now: Angle**

I was wondering if Fernando's results were fulfilling the requirement to get "at least 5 correct digits (give or take a few ulp)".
So I tested the solution for different values of the integration step H:

H    angle (radians)    time
0.05   -.275351678888   1.91851280944
0.02   -.274895222443   1.91612973771
0.01   -.**27488**2047510   **1.9160**8960085 ***
0.005   -.274881158219   1.91608794633
0.002   -.274881218423   1.91609476789
0.001   -.274881245733   1.91609643372

It appears that H=0.01 is the optimum value to get the required 5 correct digits, and it is the value used by Fernando.

Speaking of *optimization*, there may be some ways to reduce the running time for the solution. At the moment, it would take about 80h of the real HP-71B, can we reduce it to less than a business night (6pm to 9am = 15h) ?

J-F

EMAIL PM WWW FIND QUOTE REPORT

8th May, 2023, 10:40 (This post was last modified: 8th May, 2023 11:17 by Fernando del Rey.) **Post: #6**

**Fernando del Rey**
Junior Member

Posts: 19
Joined: Dec 2013

**RE: [VA] SRC #012f - Then and Now: Angle**

Thanks, J-F, your two improvements in post #4 make a significant difference in terms of execution speed.

I have tried to optimize speed by using an adaptative H for the Runge-Kutta, with larger H when the absolute value of the acceleration vector A is low, and smaller H when acceleration is high. But I did not get any maningful results out of this idea.

I suspect there must be other cases where the fine search for the root does not make sense and can be aborted, but I could not figure them out.

I'm pretty sure that Valentin's solution is going to be much shorter, faster and cleverer than mine.

EMAIL PM FIND QUOTE REPORT

9th May, 2023, 00:08 **Post: #7**

**Valentin Albillo**
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

**RE: [VA] SRC #012f - Then and Now: Angle**

.
Hi, **Fernando**,

> **Fernando del Rey Wrote:** (8th May, 2023 10:40)
> I'm pretty sure that Valentin's solution is going to be much shorter, faster and cleverer than mine.

Thanks for your appreciation but let's not raise unwarranted expectations, please. It usually leads to disappointment.

I'll wait some extra days before posting my original solution (concocted almost a year ago), just in case that someone else wants to post something, you optimize your solution even further and/or **J-F** posts his very own solution.

Talking albout *"expectations"*, this *Problem 6* is going to be the one with the least participation in both *Replies* and *Views*, as I anticipated. Oh, well ...

Best regards.
**V.**

PM WWW FIND EDIT QUOTE REPORT

9th May, 2023, 11:43 (This post was last modified: 9th May, 2023 16:56 by J-F Garnier.) **Post: #8**

**J-F Garnier**
Senior Member

Posts: 790
Joined: Dec 2013

**RE: [VA] SRC #012f - Then and Now: Angle**

> **Fernando del Rey Wrote:** (8th May, 2023 10:40)
> Thanks, J-F, your two improvements in post #4 make a significant difference in terms of execution speed.
>
> I have tried to optimize speed by using an adaptative H for the Runge-Kutta, with larger H when the absolute value of the acceleration vector A is low, and smaller H when acceleration is high. But I did not get any maningful results out of this idea.

There is an easy way to shorten the search for the root, since we are seeking for only 5 significant digits, with this well known trick:

```
145 IF ABS(L)<1.E-5 THEN L=0 ! Close enough from the target !
```

Try it !

> **Valentin Albillo Wrote:** (9th May, 2023 00:08)
> I'll wait some extra days before posting my original solution (concocted almost a year ago), just in case that someone else wants to post something, you optimize your solution even further and/or **J-F** posts his very own solution.

So it's time for me to release the latest optimizations I was able to imagine:
- check each trajectory for both the target (4,1) and the symmetric (4,-1) positions, this let us search for positive angles only,
- rewrite the FNA function without loop and without exponentiation (a pure programming optimization),
- replace INTEGER variables with REAL (no speed gain to use INTEGER in 71B BASIC).

The modified program becomes (added/modified lines in **bold**):

10 DESTROY ALL @ OPTION BASE 1 @ RADIANS
20 COMPLEX P,V,A,S(6),Z,M1,M2,M3,M4,K1,K2,K3,K4
**30** REAL I,J,Z0,D,D5,D6,H,L,T,X,E(50),U(50)
40 DATA (2,-1),(2,0),(2,1),(3,-1),(3,0),(3,1),(4,1)
50 ! Function to calculate acceleration vector at point P
**60** ! DEF FNA(P) @ A=0 @ FOR I=1 TO 6 @ A=A+(S(I)-P)/ABS(S(I)-P)^3 @ NEXT I @ FNA=A @ END DEF
**61** DEF FNA(P) @ X=ABS(S(1)-P) @ A=(S(1)-P)/(X*X*X) @ X=ABS(S(2)-P) @ A=A+(S(2)-P)/(X*X*X)
**62** X=ABS(S(3)-P) @ A=A+(S(3)-P)/(X*X*X) @ X=ABS(S(4)-P) @ A=A+(S(4)-P)/(X*X*X)
**63** X=ABS(S(5)-P) @ A=A+(S(5)-P)/(X*X*X) @ X=ABS(S(6)-P) @ A=A+(S(6)-P)/(X*X*X)
**64** FNA=A @ END DEF
70 ! Function to calculate trajectory using Runge-Kutta method
80 DEF FNP(D) @ T=0 @ V=(COS(D),SIN(D)) @ P=(0,0)
90 M1=H*V @ K1=H*FNA(P) @ M2=H*(V+K1/2) @ K2=H*FNA(P+M1/2)
100 M3=H*(V+K2/2) @ K3=H*FNA(P+M2/2) @ M4=H*(V+K3) @ K4=H*FNA(P+M3)
110 P=P+(M1+M2+M2+M3+M3+M4)/6 @ V=V+(K1+K2+K2+K3+K3+K4)/6
120 T=T+H @ IF REPT(P)<Z0 AND T<T9 THEN 90
130 IF T>=T9 THEN L=0 @ GOTO 150 ! If time is too long, exit
**135** IF IMPT(P)>0 THEN D6=D @ L=IMPT(P)-IMPT(Z) ELSE D6=-D @ L=IMPT(P)+IMPT(Z)
**140** L=L-(REPT(P)-Z0)*(IMPT(V)/REPT(V)) @ T=T-(REPT(P)-Z0)/REPT(V)
**145** IF ABS(L)<1.E-5 THEN L=0 ! Close enough from the target !
150 FNP=L @ END DEF ! L=distance from target vertical line, T=travel time
160 ! Main program SRC #12f - with JFG's modif
170 T0=TIME @ T5=MAXREAL @ T9=2.5 ! T9 is max trajectory time
180 FOR I=1 TO 6 @ READ S(I) @ NEXT I @ READ Z @ Z0=REPT(Z) ! Read star and target positions
**190** FOR D5=0 TO .75 STEP .0125 ! Angle search range and step
200 H=.025 @ DISP D5 ! Delta-T for coarse trajectory
210 IF J=0 THEN 230
**220** IF D5<ABS(E(J))+.02 THEN 320 ! If too close from a root angle already found, skip
230 IF ABS(FNP(D5))>.5 THEN 320 ! If too far from target, skip
**235** IF ABS(V)>2 THEN 320 ! If abnormal final speed, skip
240 IF T>=T9 THEN 320 ! If time is too long, skip
**250** H=.01 @ DISP D6;"*" ! Delta-T for fine trajectory
**260** D0=FNROOT(D6,D6,FNP(FVAR)) ! Fine search of root
270 IF T>=T9 THEN 320 ! If time is too long, skip
280 IF J=0 THEN 300
290 IF ABS(D0-E(J))<1.E-11 THEN 320 ! If root angle was already found, skip

```
300 J=J+1 @ E(J)=D0 @ U(J)=T ! Save root angle and trajectory time
310 IF T<T5 THEN T5=T @ J5=J ! Look for shortest trajectory time
320 NEXT D5 @ T0=TIME-T0
330 DISP "Angle= ";E(J5);"Time= ";T5;T0
```

The result is:

Angle= -.**27488**2047936 Time= **1.9160**8959636 (96.61s)

obtained in about 97s on my Emu71/Win system with ~1400 speed factor,
i.e. about 38 h on a real HP-71B. Not yet a run overnight, but very possible over a week-end.

The 6 trajectories:

```
    Angle             Time
-.196990679676 1.99998010763
-.274882047936 1.91608959636
 .328560456480 2.0237867079
-.461225625588 2.1526452933
 .551285709122 1.91762496347
 .732203223220 2.06532676912
```

J-F
Edit: typos.

EMAIL PM WWW FIND QUOTE REPORT

9th May, 2023, 15:37 **Post: #9**

**EdS2**
Senior Member

Posts: 517
Joined: Apr 2014

**RE: [VA] SRC #012f - Then and Now: Angle**

Although it's not part of Valentin's challenge, I'd love to see what these six trajectories look like!

EMAIL PM FIND QUOTE REPORT

12th May, 2023, 22:04 **Post: #10**

**Valentin Albillo**
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

**RE: [VA] SRC #012f - Then and Now: Angle**

**Hi, all,**

Almost two weeks have elapsed since I posted ***Problem 6*** and, as expected, its *perceived* difficulty has resulted in very few posts, just a single solution (other than my original one, of course) by **Fernando del Rey** and worthwhile refinements to it by the indefatigable **J-F Garnier**. Thank you very much to both of you for your interest and overall outstanding contributions.

Now, this is my detailed ***sleuthing*** *process* and resulting *original* ***solution***, plus *additional* ***comments****:*

## My sleuthing process

First of all, the relevant formula is Newton's equation for universal gravitation, further simplified as all seven masses and the *gravitational constant* ***G*** are to be taken as ***1***. Also, as *force*, *acceleration*, *velocity* and *position* all are two-component values, representing them using *complex numbers* is *ideal* and greatly simplifies the coding while also reducing the running time.

Using the formula we obtain the force exerted by the fixed stars upon the moving planet and thus the *acceleration*, *velocity* and *position* at any given time. I fully knew that those values can be obtained by numerically solving the relevant *2nd-order ordinary differential equation* by using for instance a favorite of mine, the 4th-order *Runge Kutta method (aka RK4)*.

However, though this would surely be the choice method for best efficiency, just for fun I decided to use a much simpler, more *intuitive* procedure, which consist of first computing the *force* (and thus the *acceleration*) by adding up the contributions from each star, then updating *velocity* and *position* by assuming that the acceleration remains constant within the (tiny) time increment.

This procedure is extremely simple to implement and requires a *single* call per step to the function which computes the force/acceleration while *RK4* would require *four* calls per step, which for the current complicated function is quite time-consuming. On the other hand, *RK4* has a local truncation error *(LTE)* of $O(h^5)$ while the simpler procedure's *LTE* is $O(h^2)$ so we'd be trading accuracy/runtime for algorithmic simplicity; however, for the modest accuracy goal it will likely be adequate enough.

After selecting the algorithm to use, I tried out assorted angles in order to decide on a *range* where the optimum angle would surely lie, as angles outside it would obviously result in wasteful, unnecessarily longer or even *"retrograde"* trajectories. Also, though I *did* notice, I didn't take advantage of the initial position's *symmetry*. See **Notes** below.

Now, I'd need to scan the range using a coarse increment (for speed; see **Notes** below for details on the various *heuristics* applied) while keeping track of the *minimum distance* to the destination and, if within a certain tolerance, a *refining* procedure would then be applied to improve the accuracy, followed by linearly *interpolating* both *angle* and *time* so as to exactly hit the destination (within a small tolerance).

Finally, once the scan is completed, the optimum *angle* and resulting *minimum time* among all suitable trajectories would be subjected to a finer refinement to further improve the accuracy. As seen below, this strategy ultimately results in *4-5 correct digits*.

## My original solution

My *original solution* is this ***12**-line, 703-byte* **HP-71B** program:

```
 1  DESTROY ALL @ OPTION BASE 0 @ DIM B,C,D,J,T,U @ COMPLEX X(6),A,P,Q,S,V,Y,Z @ SETTIME 0
 2  DATA .005,.01,.1,.01,1,9,(0,0),(2,-1),(2,0),(2,1),(3,-1),(3,0),(3,1),4,1

 3  FIX 2 @ READ H,F,M,L,V0,G,X,B,C @ Y=(B,C) @ FOR R=-.5 TO .75 STEP F @ IF FND(R)>M THEN 5
 4  W=FNROOT(R,R,FNY) @ DISP W;T;TIME$ @ IF T<G THEN N=W @ G=T
 5  NEXT R @ H=H/10 @ W=FNROOT(N,N,FNY) @ FIX 4 @ DISP "Min T:";T;"for A:";W;"(";TIME$;")" @ END

 6  Q=P @ A=0 @ FOR J=1 TO 6 @ Z=X(J)-P @ A=A+Z/ABS(Z)^3 @ NEXT J @ P=P+H*V+K*A @ V=V+H*A @ RETURN

 7  DEF FND(R) @ P=X(0) @ V=RECT((V0,R)) @ K=H*H/2 @ U=9 @ FOR T=H TO 3 STEP H @ R=REPT(P)
 8  D=U @ GOSUB 6 @ U=ABS(P-Y) @ IF U>D THEN 'N' ELSE IF REPT(P)<R THEN FND=9 @ END
 9  NEXT T @ 'N': T=T-H @ FND=D @ END DEF

10  DEF FNY @ H=H/2 @ P=X(0) @ V=RECT((V0,FVAR)) @ K=H*H/2 @ T=0
11  T=T+H @ GOSUB 6 @ IF REPT(P)<B THEN 11 ELSE S=P-Q @ D=(B-REPT(Q))/REPT(S)
12  T=D*H+T-H @ H=H*2 @ D=D*IMPT(S)+IMPT(Q)-C @ FNY=D*(ABS(D)>L)
```

**Notes:**

- *Lines 1-2* perform the initialization and define the necessary data. There's no need for a **RADIANS** declaration because no trigonometric functions are used but **RECT**, which with a complex argument always uses *radians* regardless of the angular mode.

  Also, I noticed the *symmetry* of the starting position (as ***J-F Garnier*** did as well) but despite the obvious speed up I decided *not* to depend on it so that my program would be able to solve *generalized* cases (including *non-symmetric* starting positions,) by simply defining all relevant parameters (initial coordinates, velocity, increments, limits, tolerances, etc.) in a single **DATA** statement.

- *Lines 3-5* are the main program, which coarsely scans a plausible angle range and once a suitable one is found, it uses **FNROOT** to try and find more precisely the angle which makes the *Y* coordinate of the final position exactly match the *Y* of the destination (within a small tolerance).

  Once the scan is over, both a candidate *optimum angle* and a *minimum time* have been selected among the tentative trajectories, and then a last, finer refinement is performed to further increase the final solution's accuracy (both *angle* and minimum *time*).

- *Line 6* implements a subroutine which is called from both **FND** and **FNY** (defined below) and essentially computes the force (and thus the *acceleration)* upon the moving planet due to the six fixed stars, plus it also updates its current *speed* and *position*. It was initially implemented as another *UDF* but converting it to a subroutine speeds up the process.

- *Lines 7-9* define **FND**, which computes the resulting trajectory for a given angle and returns an approximation to the *minimum distance* to the destination, as well as the nearest position's coordinates and the time to reach it. Besides there being a limit on this time, **FND** further includes *heuristics* to abort the computation if *(a)* the current position's *X* coordinate ever goes *backwards* or if *(b)* the distance to the destination eventually *increases*, to avoid computing a trajectory that can't result in the sought-for minimum time.

- *Lines 10-12* define **FNY**, which is used while refining the minimum distance for a given angle with greater accuracy than when using **FND**, and further it *interpolates* the final coordinates and time so that the *X* coordinate exactly matches the *X* of the destination.

- The **MATH ROM** is used for all *complex-number* variable definitions and operations, plus **FNROOT** is used for the *angle* refinements.

Let's run it:

```
>RUN
         Angle    Time    Runtime
         -----------------------
         -0.46    2.15    00:01:26
         -0.27    1.91    00:04:01    {minimum time}
```

```
-0.19   1.93   00:05:41
 0.33   1.97   00:09:28
 0.55   1.92   00:12:06   {second best}
 0.73   2.05   00:14:43

Min T: 1.9160 for A: -0.2745 (00:24:30) {go71b: 24'30", Emu71/Win: 3'13", HP-71B: ~ 52h}
```

The timing for a *physical* **HP-71B** may seem a bit steep but it's nothing that an *AC* adapter and a weekend can't deal with. Just start the program on *Friday evening*, forget about it and go enjoy the weekend, and by *Monday 4 a.m.* you'll have the solution displayed, waiting for you to wake up.

The correct *10-digit* result is *Angle* = ***-0.2748812540***, *Time* = ***1.916096918***, so we can gauge the accuracy obtained this way:

```
>FIX 9 @ T;W

      1.916021524  -0.274541755
```

and the errors are:

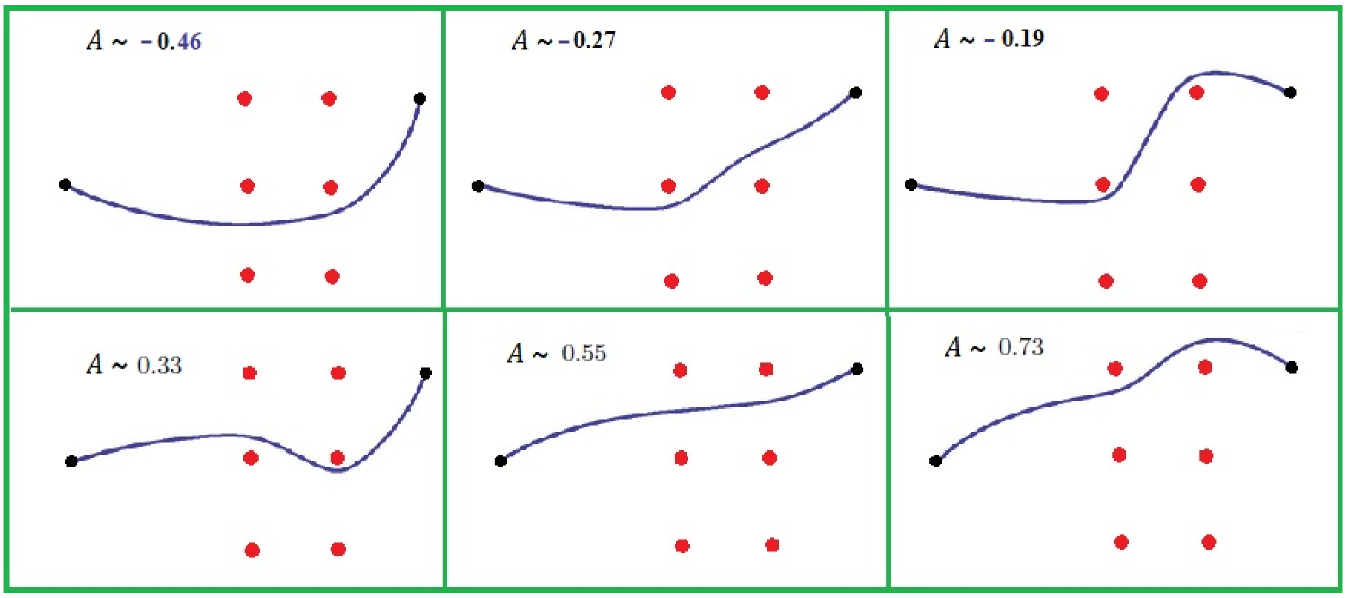
```
(-0.274881254) - (-0.274541755) = 0.000339499    i.e., we got 4 digits (save 3 ulp)
( 1.916096918) - ( 1.916021524) = 0.000075394    i.e., we got 5 digits (save 0.75 ulp)
```

For the record, the correct answer rounded to *33-34 digits* is:

```
Angle = -0.2748812540262767155096292165584 26
Time  =  1.916096918332004910307460694725436
```

## Additional comments

The six angles obtained by my original solution result in the following trajectories, counterclockwise:



- As can be seen in the graphic above, the straightest, most direct paths are the ones for the angles *A* ~ ***-0.27*** and *A* ~ ***0.55*** and matter of fact the latter is clearly the one having the shortest *length* but the former takes the shortest *time*, as asked, namely ***1.9161*** vs. ***1.9176***, probably due to the close encounter with the *(2,0)* star significantly increasing the planet's velocity.

  To better see it, an easy challenge extension would be to compute and display not only the arrival *time* but also the path *length* and *final velocity* at the destination for the various trajectories.

- I hereby *"officially"* declare **J-F Garnier** as the *valedictorian* of this *SRC#12*, as he posted solutions and/or significant contributions to *all* six problems, with **Fernando del Rey** a worthy *salutatorian*, as he participated in most problems and delivered an excellent solution for this last, allegedly most difficult one. ***Congratulations*** and thanks a lot to both of you.

- Last but not least, I *(faintly)* hoped that some users of vintage *HP* ***graphing*** *calcs* would produce *RPL* solutions, obtaining not just the correct numbers but also *graphically* depicting the resulting *trajectories* as shown above, but alas, no such luck. Perhaps it was too difficult, as compared to their usual dealings with this library, or that port, or creating yet another bunch of list utilities and such ...

**Well, that's all, at long last *SRC#12* has been completed *7 months* after I posted *Problem 1*, and the point has been fully made.**

I'm now taking a *sabbatical* from proposing further elaborate challenges. Thanks for your interest and best regards.

**V.**

PM WWW FIND EDIT QUOTE REPORT

14th May, 2023, 10:28 (This post was last modified: 14th May, 2023 17:20 by J-F Garnier.) **Post: #11**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

**RE: [VA] SRC #012f - Then and Now: Angle**

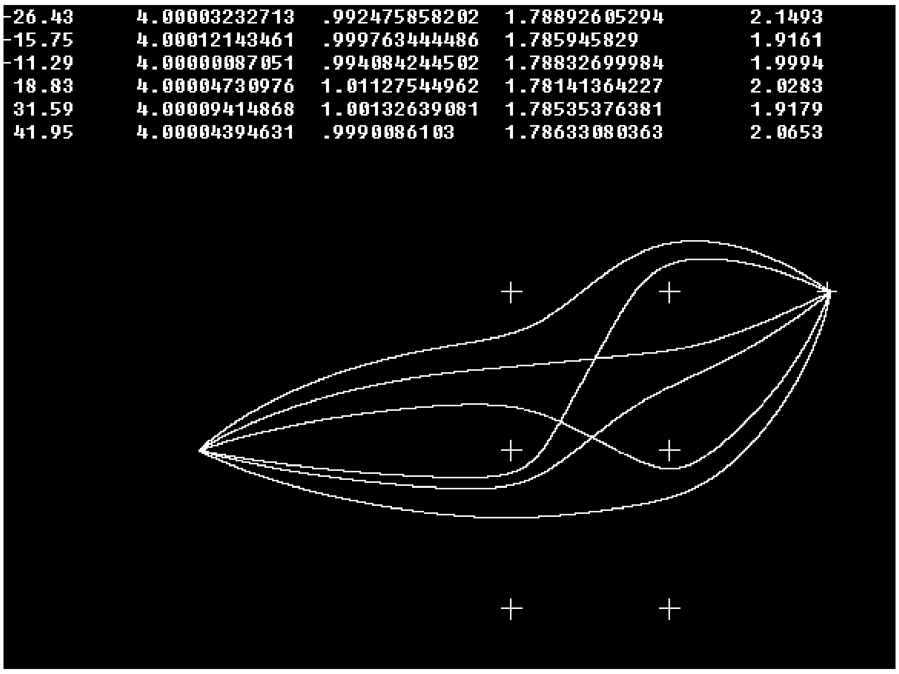
Thanks Valentin, for this last episode. It was particularly interesting, and I'm surprised too by the low participation. Especially, graphic calculators able to display the trajectories are missing.
On the other hand, using a fast emulator was really helpful to devise a solution and improvements. Using a real calculator may not be practical, Prime excepted maybe but this machine is not precisely a vintage machine.

Some more comments:

- as I explained, I was not sure that integrating the motion equations would work well. It actually works for this problem because the trajectories are short , and don't pass too close from a star. But generally, the integration fails quickly, and it's impossible to follow the trajectory in this way, after just a few units of time.

- I didn't post a specific solution of mine, because Fernando's solution already had all the key ingredients: angle coarse sweep, RK4 integration, interpolation and fine search with FNROOT. But there was some room for optimisations...

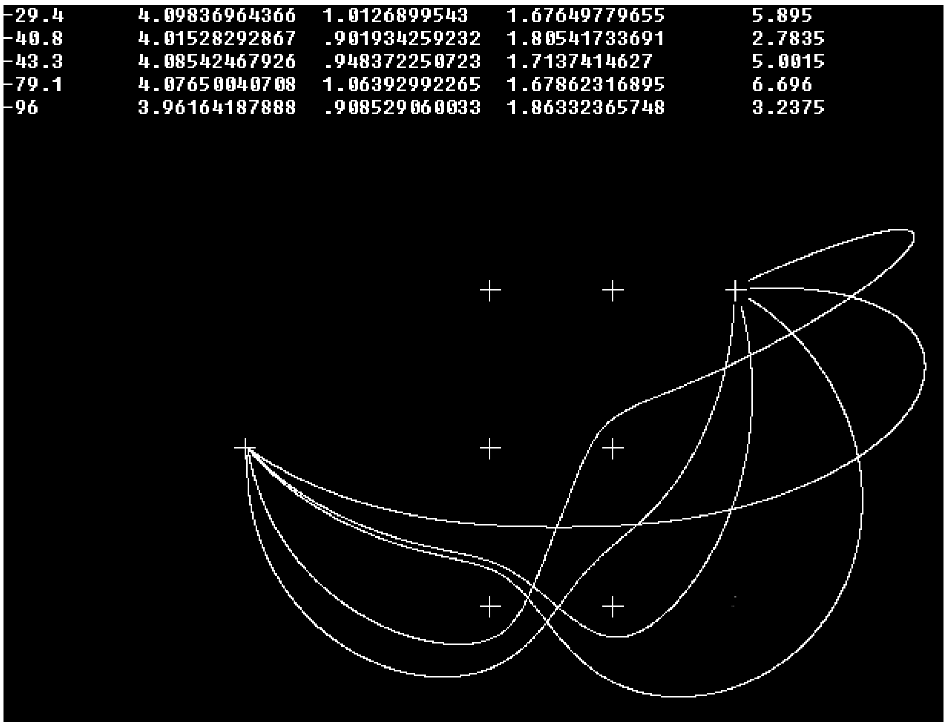
- The HP-71B as no graphic capabilities, so in order to be convinced that the 6 trajectories found by Fernando were the shortest ones, I resorted to my HP BASIC compatible system to display and check the trajectories.
Here is the result:



```
-26.43     4.00003232713  .992475858202  1.78892605294      2.1493
-15.75     4.00012143461  .999763444486  1.785945829        1.9161
-11.29     4.00000087051  .994084244502  1.78832699984      1.9994
 18.83     4.00004730976  1.01127544962  1.78141364227      2.0283
 31.59     4.00009414868  1.00132639081  1.78535376381      1.9179
 41.95     4.00004394631  .9990086103    1.78633080363      2.0653
```

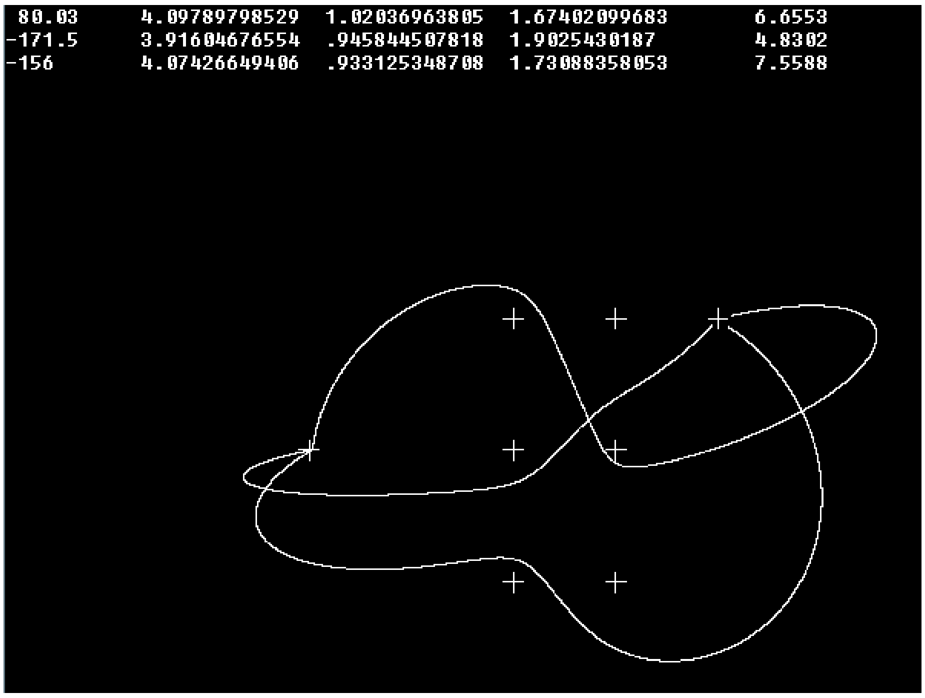
My test program uses the RK4 code from Fernando, but just asks for a starting angle to display the trajectory until crossing the target position.
The lines on top give: starting angle in degrees, final X, Y and speed, and time.
As I demonstrated above, the final speed doesn't depend on the path, the simulation confirms it.

Then, I played further with my code. For most starting angles, it just produces garbage trajectories, but sometime gives funny alternate solutions:



```
-29.4       4.09836964366   1.0126899543    1.67649779655        5.895
-40.8       4.01528292867   .901934259232   1.80541733691        2.7835
-43.3       4.08542467926   .948372250723   1.7137414627         5.0015
-79.1       4.07650040708   1.06392992265   1.67862316895        6.696
-96         3.96164187888   .908529060033   1.86332365748        3.2375
```

See how close starting angles such as 40.8 and 43.3 produce completely different trajectories.



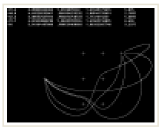
```
 80.03      4.09789798529   1.02036963805   1.67402099683        6.6553
-171.5      3.91604676554   .945844507818   1.9025430187         4.8302
-156        4.07426649406   .933125348708   1.73088358053        7.5588
```

So, thanks again, and hope to see more challenges in the future.

J-F

**Attached File(s)**

**Thumbnail(s)**



EMAIL PM WWW FIND QUOTE REPORT

14th May, 2023, 11:28 **Post: #12**

**EdS2**
Senior Member

Posts: 517
Joined: Apr 2014

**RE: [VA] SRC #012f - Then and Now: Angle**

Excellent results from a promising challenge! Thanks to all. And for the trajectories!

(The final speeds coming out the same is a good consistency check, I think - energy conservation will, I think, demand that the final energy is the same regardless of path.)

EMAIL PM FIND QUOTE REPORT

14th May, 2023, 18:47 **Post: #13**

**pavel nemec cz**
Junior Member

Posts: 11
Joined: May 2014

**RE: [VA] SRC #012f - Then and Now: Angle**

> **Valentin Albillo Wrote:** (12th May, 2023 22:04)
>
> **Well, that's all, at long last *SRC#12* has been completed *7 months* after I posted *Problem 1*, and the point has been fully made.**
>
> I'm now taking a *sabbatical* from proposing further elaborate challenges. Thanks for your interest and best regards.
>
> **V.**

Hi Valentin, participants to all challenges!

First of all thank you Valentin for all your excellent contributions via interesting and fun masterpieces professionally plotted, written and explained here with so special care for all details aiming to our education.
Thanks to all participants for posting solutions which keeps the subject alive. I am a long time follower and enthusiast of those beauties but my skills are just not in par so no active participation from my side. Be sure it does not discourage me to silently follow those threads and studying posts gives me more insight in math, logic, programming - and fun. So thank you again.

Kind regards
Pavel

EMAIL PM FIND QUOTE REPORT

15th May, 2023, 09:11 **Post: #14**

**Gjermund Skailand**
Member

Posts: 52
Joined: Dec 2013

**RE: [VA] SRC #012f - Then and Now: Angle**

Dear Valentin,

Thanks for the challenges.
For me, the last one was very interesting, as I have been curious about "rocket science". Probably the low number of replies to the last problem is simply that summer is coming, and also that integration (R_K) seemed to and turned out to work so well.
Further, thanks to J-F for showing the paths.
Is it possible to calculate longer paths, say circling the star cluster once twice, or would numerical inaccuracies quickly make any such results rubbish?

BR Gjermund

EMAIL PM FIND QUOTE REPORT

15th May, 2023, 10:09 **Post: #15**

**vaklaff**
Member

Posts: 110
Joined: Dec 2019

**RE: [VA] SRC #012f - Then and Now: Angle**

+1 to what Pavel wrote, from another silent observer. Thanks to Valentin and all the contributors!
vaklaff

EMAIL PM FIND QUOTE REPORT

16th May, 2023, 20:55 **Post: #16**

**Valentin Albillo**
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

**RE: [VA] SRC #012f - Then and Now: Angle**

.
Hi, **J-F Garnier**, **pavel nemec cz**, **Gjermund Skailandand** and **vaklaff**,

**J-F Garnier Wrote:**

Thanks Valentin, for this last episode. It was particularly interesting, and **I'm surprised too by the low participation**. Especially, **graphic calculators** able to display the trajectories **are missing**.

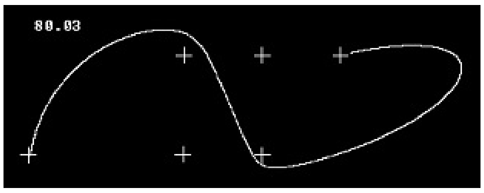
You're welcome, **J-F**, but I'm not surprised in the least, it's just as expected. Some seven months ago I posted a simple *Probability problem* which required no special knowledge and I got ~ 90 replies, but now I post a little *"physics"* problem which requires some very basic knowledge about simple concepts such as force, acceleration, velocity, etc., and I get about 5 replies or less. Oh, well, that's life in this *New Forum*.

As for vintage graphic calculators *missing*, consider that all of them are *RPL* models so no surprise either. 'Nuff said.

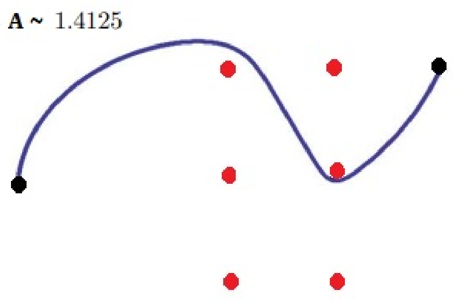
**J-F Garnier Wrote:**

See how **close starting angles** such as 40.8 and 43.3 produce **completely different trajectories**.

Yes, and to further demonstrate it, in that same figure you include the trajectory for ***Angle = 80.03º = 1.3968 rad***, namely:



but you don't include the one for the very close ***Angle = 80.93º = 1.4125 rad*** (less than 1º apart,) which is much shorter as it doesn't backtrack, though (a) still far from the minimum time, (b) outside of my scanning range, and (c) even if it were included it would nevertheless be rejected by my heuristics.



**pavel nemec cz Wrote:**

First of all **thank you Valentin for all your excellent contributions** via interesting and fun masterpieces professionally plotted, written and explained here with so special care for all details aiming to our education. [...] Be sure it does not discourage me to **silently** follow those threads [...]

Thanks for your kind words, **pavel**, I'm glad you like my contributions and even somewhat profit from them. As for *"silently"*, as long as you care to post your appreciation here for me to know, your alleged *"silence"* speaks volumes ! 🙂

**Gjermund Skailand Wrote:**

Thanks for the challenges. [...] **Is it possible to calculate longer paths**, say circling the star cluster once twice, or would numerical inaccuracies quickly make any such results rubbish?

You're welcome, **Gjermund**, thanks for your interest. As for your question, yes, it's quite possible (if difficult,) see *Direct gravitational N-body simulations* in N-body simulation.

To minimize numerical inaccuracies while computing the path for extended time periods, it's usually mandatory to use *multiprecision* with a *high-order* integration method. For instance, the 33-34 digits I posted were obtained using not *RK4* but an ***RK14*** method (which actually produced as many as *50* correct digits that I then rounded down to 33-34 digits for posting here.)

**vaklaff Wrote:**

+1 to what Pavel wrote, from **another silent observer**. Thanks to Valentin and all the contributors!

Thanks for your appreciation, **vaklaff**, but as long as you've posted here to voice it, you're not silent anymore. Much appreciated.

Regards to all of you.
**V**.