5.3 The Planet

Author: Michele Moro

5.3.1 Teacher's guide

- Title: the Planet
- Introduction: Simulation of the motion of a planet (or another object) around the Sun based on the gravitational force; study of the conic curves.
- Goals:
  - To improve the knowledge of some basic physics concepts, such as space, speed, acceleration, time;
  - To study the theory of the planet motion (Kepler's and Newton's laws, conics);
  - To study some aspects of the analytical representation of conics;
  - To get some experience of what a simulation is and to what extension it can give a measurable representation of a real phenomenon;
  - To combine appropriately integer operation with loss of precision (division and square root) in an expression in order to minimize the total error.
- Age group: 16-19 years old.
- Rationale of the teaching approach: physics is better taught and learned when theory is presented together with some experimental activities. The notion of gravitational field is one of the most fascinating and difficult ones at the same time. The essence of the Kepler's and Newton's laws is not immediately intuitive and, therefore, the robotic simulation makes them more acceptable. The simulation, made with a relatively simple robot, is also the occasion to make some accessory reasoning about the interesting properties of conics and other geometrical aspects.

5.3.2 The problem

When Isaac Newton tried to determine what force justified the planetary motion in accordance with the Kepler's laws, he reached the conclusion that this force must be a mutual attraction, proportional to the product of the masses involved (the sun and the planet) and inversely proportional to the square of the distance from their centers of mass. In formula, the form of this force, the gravitational force (fig. 5.3.1), is given by:

$$F_G = G\ M\ m\ /\ r^2 \qquad (5.3.1)$$

with G the universal constant equal to:

$$G = 6.670\ 10^{-11}\ Nm^2kg^{-2} \qquad (5.3.2)$$

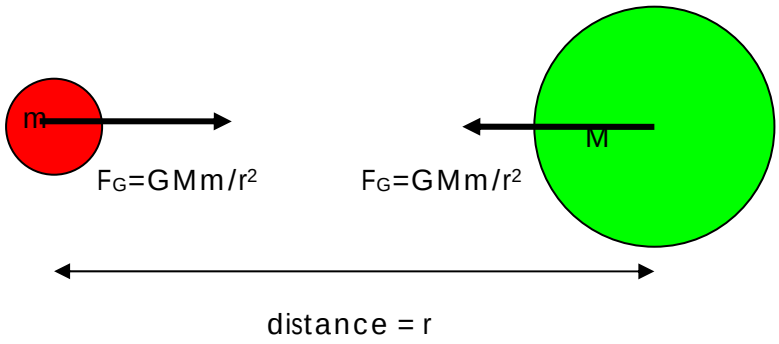

***Fig. 5.3.1** – The gravitational force*

Limiting the analysis to the motion of a single body around the sun, making the usual simplifying assumption that *M* (mass of the sun) >> *m* (mass of the body), it is possible to consider that the gravitational effect is limited to the only moving body, forced in its orbit by a centripetal acceleration in the form of:

$$a_G = F_G / m = G\ M / r^2 = \mu / r^2 \qquad (5.3.3)$$

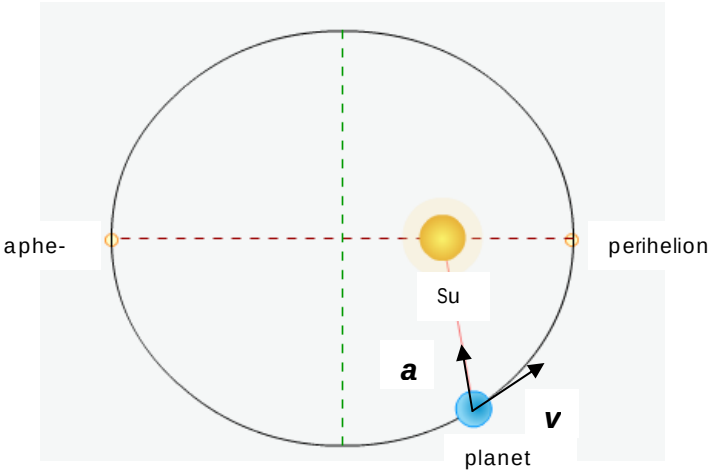

***Fig. 5.3.2** – The orbit of a body around the sun*

Under these conditions, the orbit is a conic which has the sun in one of its focus (fig. 5.4.2). The speed vector is always tangent to the orbit, oriented in the same direction of the motion, whereas the acceleration with modulus a_G is always directed from the body to the sun and, therefore, has an effect to accelerate (positive sign of the projection of the vector acceleration onto the speed vector) on the speed modulus, when the body is approaching and to decelerate (negative sign of the projection of the vector acceleration onto the speed vector), when the body is distanc-

ing. At the point closest to the sun (called *perihelion*), the acceleration is orthogonal to the speed and it is the greatest. In the case of the elliptical orbit, in the farthest point (the *aphelion*), the acceleration is still orthogonal but the speed is the smallest.

Unfortunately, the analytical study of a gravitational field based on the force of (5.3.1) is very hard and out of competence of a normal secondary level student. Therefore, a practical experience, which can give at least a qualitative evidence of the kind of orbit a body is forced to follow in a gravitational field, is of great interest.

Conics can be analytically defined as the locus of points *p*, for which the ratio between the distance of *p* from a point *F* (focus) and a line *D* (Directrix) is constant. This ratio is called *eccentricity* (*e*): when $0 \leq e < 1$ the locus is an ellipse, with $e = 1$ a parabola, with $e > 1$ a hyperbola. In the cases of hyperbola and ellipse, these properties remain substantially the same for the second focus. Said *a* the semi-major axis of the ellipse (or the distance between the center of the focuses and the cusp of the hyperbola), *d* and *f* the distance between the center and respectively D and F, it also holds:

$$f = e/a \qquad e = f/a \qquad (5.3.4)$$

$$d = a/e \qquad (5.3.5)$$

Thus, for $e = 0$ the two focuses coincide, while the directrix is at infinity, and you get the special case of ellipse of the circumference.

The equation of an ellipse with the main focus placed on the origin and the other on its left on the x-axis is given by:

$$(x+f)^2/a^2 + y^2/b^2 = 1 \qquad (5.3.6)$$

where *a* and *f* are the semi-major axis and the focal distance already defined above and *b* the semi-minor axis. An alternative characterization is given in polar coordinates (fig. 5.3.3), placing the ellipses in the same position as in (5.4.6):

$$r = l / (1 + e \cos \theta) \qquad (5.3.7)$$

where *l* is the distance of the main focus from its vertical projection on the ellipse, *r* e θ are the polar coordinates of a generic point on the ellipse. For the perihelion and aphelion it holds:

$$r_{PER} = r_{MIN} = a - f = [\theta = 0] = l / (1+e) \qquad (5.3.8)$$

$$r_{APH} = r_{MAX} = a + f = [\theta = \pi] = l / (1-e) \qquad (5.3.9)$$

$$r_{PER} + r_{APH} = (l / (1+e)) + (l / (1-e)) = 2a \; a=l/1-e^2 l=a\,(1-e^2) \qquad (5.3.10)$$

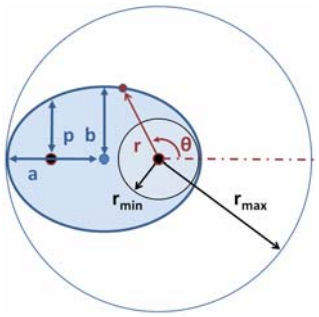

Fig. 5.3.3 – *Ellipse in polar coordinates*

Finally one can easily verify that:

$b = a\ \mathrm{sqrt}(1-e^2)$ $b^2 = a^2 - a^2e^2 = a^2 - f^2$ $b^2 + f^2 = a^2$ (5.3.11)

from which it derives that the distance of focus from the topmost point of the ellipse coincides with the semi-major axis.

Generally speaking, there is a known relation between speed and distance from the sun that, in the cases shown above, is given by:

Elliptical trajectory: $v = \mathrm{sqrt}(\mu\ ((2/r)-(1/a)))$ (5.3.12)

Parabolic trajectory : $v = \mathrm{sqrt}(\mu\ (2/r))$ (5.3.13)

Hyperbolic trajectory: $v = \mathrm{sqrt}(\mu\ ((2/r)+(1/a)))$ (5.3.14)

The speed of (5.3.13) is also known as the 'escape speed' (or 'escape velocity') because it is necessary that the initial speed of the body is greater than or equal to that of the escape speed for the body to escape the close orbit.

In the elliptical case, the maximum (at perihelion) and minimum (at aphelion) speeds are given by:

$$v_{PER} = \mathrm{sqrt}(\mu\ ((2/(a-f))-(1/a))) \quad (5.3.15)$$

$$v_{APH} = \mathrm{sqrt}(\mu\ ((2/(a+f))-(1/a))) \quad (5.3.16)$$

whereas for the period T we have:

$T = (2\pi/\mathrm{sqrt}(\mu))\ \mathrm{sqrt}(a^3)$ $T^2 = (4\pi^2/\mu)\ a^3$ (5.3.17)

in harmony with the Kepler's third law.

5.3.3 Our simulation with NXT

In preparing the experience, one needs to be aware of the limitations, both physical and programming, of the NXT robot: inaccuracies in the control of motors and motion, integer arithmetic only (in the standard firmware, there is no floating point

support), basic operations (lacking in particular of trigonometric functions). The results of this experience have a rather qualitative than quantitative value.

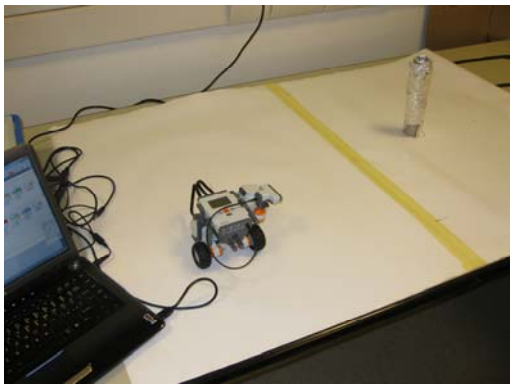

Fig. 5.3.4 – *The 'orbiting body'*

The robot, which is the orbiting body, is a *tribot* (fig. 5.3.4), with two independently driven wheels and a third one free, and equipped with a sonar head mounted on a third motor in order to adjust the angle of its 'vision' during the motion. The sun is represented by a fixed object 'visible' to the sonar so that the distance r is measured by this sensor.

The basic idea is to choose two reasonable values for μ and a in order to obtain practical values for the other parameters, specifically speed and acceleration. You have to choose also an appropriate (small) time interval Δt on which the calculation of each motion step will be based. During the simulation, acceleration and speed will always be calculated using formulas (5.3.3) and (5.3.12) (assuming you have installed the square root block in your NXT-G environment).

To simplify the simulation, we also assume that, initially, the robot is put on the aphelion, at a distance of $a+f$ from the focus, so that its axis can be orthogonal as to the major axis of the expected elliptical trajectory. The simulation does not maintain any information about the position and the orientation of the robot: we always assume that, at any motion step, the starting position and orientation are correct as the cumulative effect of the previous steps.

In the simulation, a step at time t is formed by small straight-line motion, followed by a rotation around the robot's axis middle point. The first motion corresponds to the contribution of the tangential speed vector, and, therefore, it is calculated as $v*\Delta t$; the rotation corresponds to the contribution of the acceleration vector that rotates the speed vector: fig. 5.3.5 shows that the vector composition of speed v at time t and its variation given by the vector $\Delta v = a_G * \Delta t$.

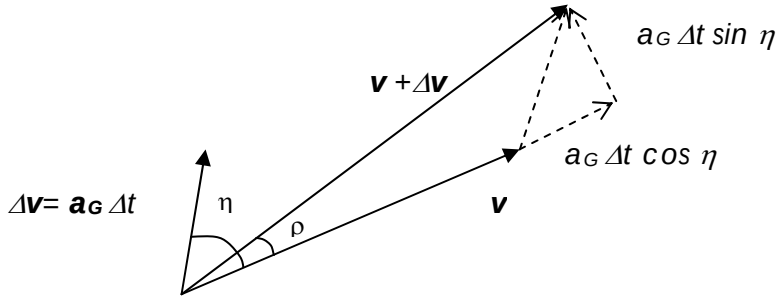

Fig. 5.3.5 – *Vector composition*

To ensure that the orientation of the robot at the instant $t+\Delta t$ is the vector $v+\Delta v$, as shown by the figure, we need to rotate the robot of an angle ρ equal to:

$$\rho = \arctan((a_G \Delta t \sin \eta) / (v + a_G \Delta t \cos \eta)) \qquad (5.3.18)$$

where η is the angle between vectors v and a_G.

A first rough approximation is to consider the acceleration vector always substantially orthogonal to the speed vector ($\eta = \pi/2$). With this simplification it holds:

$$\rho_{APPROX} = \arctan(a_G \Delta t / v) \approx a_G \Delta t / v \qquad (5.3.19)$$

Further (5.3.19) approximation of considering the angle so small as to have the arctan value and the angle (in radiants) coincident has been applied. This produces a first approximated calculation of the rotation to be applied to the robot very simple.

Plotting the theoretical ellipse and that of the simulation, obtained by applying the approximate method described above, by using the following values of characteristic parameters (lengths scaled in centimeters)

a=40 f=20 e=0.5 b = a sqrt(1-e^2) = 20 √3

μ=6000 Δt=0.5 (5.3.20)

and starting, as already assumed, with the robot placed on the hypothetical aphelion and oriented parallel to the minor axis of the ellipse, we obtained the plotting values of fig. 5.3.6. As you can see the approximation is not good after the first quadrant. Thus, we decided to add a correction to this first approximation trying to maintain the calculation simple.

From an analysis of the variation of the angle η, while the point moves along the bottom semi-ellipse, you observe that, both at aphelion ($r_{AF} = a+f$) and at perihelion ($r_{PER} = a-f$) $\eta=\pi/2$, but along the path, the angle decreases until it reaches a minimum, which depends on the value of e, when $r = a$. With the data given above, the minimum is equal to $\eta_{MIN} = \pi/3$ with $\sin(\eta_{MIN}) = \sqrt{3}/2 \approx 0.87$, $\cos(\eta_{MIN}) = 0.5$, while $\eta_{MAX} = \pi/2$ with $\sin(\eta_{MAX}) = 1$ and $\cos(\eta_{MAX}) = 0$. Therefore, we decided to

apply a correction to the angle in the range where sin(η) is more different from 1, which has been empirically determined as $a\text{-}f/2 \le r \le a\text{+}f/2$: in this interval, the rotation to be applied is calculated as:

$$\rho_{APPROX} \approx ((a_G\,\Delta t \sin \eta_m) / (v + a_G\,\Delta t \cos \eta_m)) = ((a_G\,\Delta t\ 0.9) / (v + a_G\,\Delta t\ 0.5)) \quad (5.3.21)$$

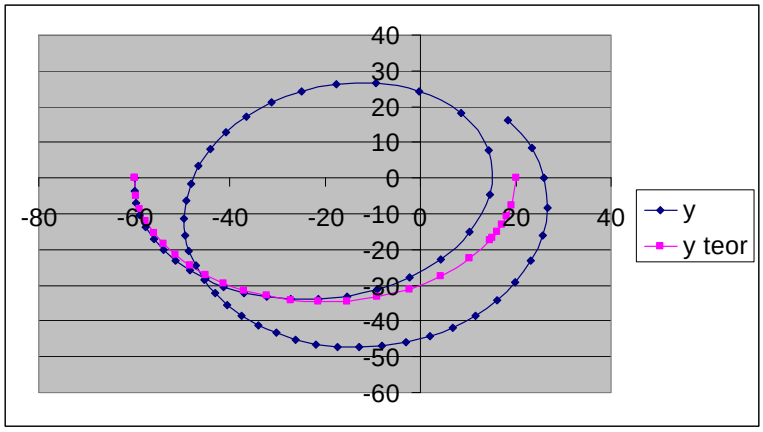

Fig. 5.3.6 – *A first approximation*

With this correction, the simulated evolution is plotted in Fig. 5.3.7 and the improvement is evident.

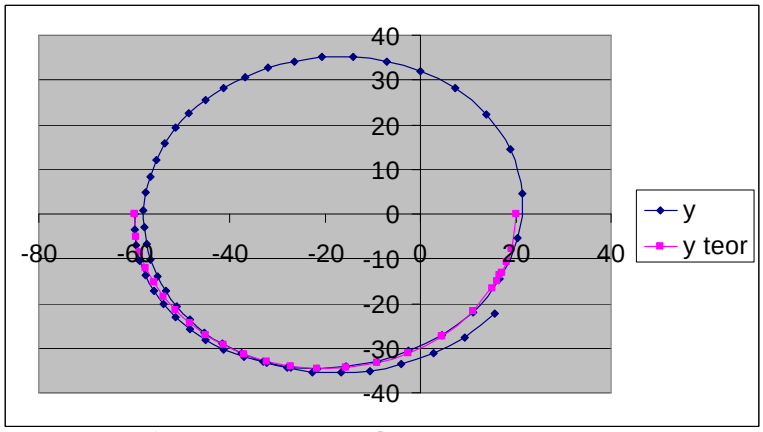

Fig. 5.3.7 – *A second approximation*

Another straightforward correction is necessary in order to compensate the measure of the distance *r* performed by the sonar: in fact, the measure is less than the actual distance of the focus from the rotation point of the robot, which we consider as the application point of the speed and acceleration vectors, both because the object used as the attracting body has a non-null radius and because the sonar is at a cer-

tain distance from the axis of the robot, where the rotation point lies. Thus, to obtain the correct value of *r*, the measure must be incremented of the sum of these two distances (the sun's radius and the offset of the sonar).

In the practical realization with NXT, we must also take into account the limitations of the integer calculation: besides delaying as much as possible all the divisions in just one division as the final operation, in our case we must also deal with the imprecision introduced by the square root. It results more precise to execute the integer square root as the very last operation. But you must also take into account the relative error of the square root in cases of small and large numbers. This could suggest to move inside the square root external multiplicative factors (elevating them to their square), whereas divisional factors should be moved inside the square root only if the inner division maintains a quotient not too small (in order to maintain small the error of the integer square root). In doing these passages, you must not generate overflow of the 32 bit integer capacity ($+2^{31} \approx 2\ 10^9$).

Therefore, assuming that

$$\Delta t = I / j \qquad R_w = p / q \qquad D_w = w / z \tag{5.3.22}$$

with R_w the radius of the wheels, D_w the distance between the wheels and *i*, *j*, *p*, *q*, *w*, *z* integer values, for the straight-line motion step it follows:

$$v \cdot \Delta t = \theta_w R_w = \theta_{wd} (2\pi/360) \cdot R_w \tag{5.3.23}$$

where θ_w and θ_{wd} are the angles which the wheels must rotate of, respectively measured in radiants and degrees. From (5.3.23) you obtain the angle to be set as a parameter of the motor control block, expressed in degrees:

$$\theta_{wd} = 360 \cdot v \cdot \Delta t / (2\pi \cdot R_w) \tag{5.3.24}$$

To perform this calculation accurately, we now apply the recommendations suggested above:

$$\begin{aligned} \theta_{wd} &= 360 \cdot \mathrm{sqrt}(\mu((2/r)-(1/a))) \cdot (i/j) / (2(314/100) \cdot (p/q)) = \\ &= ((9000 \cdot i \cdot q) / (157 \cdot p \cdot j))\ \mathrm{sqrt}((2 \cdot \mu \cdot a - \mu \cdot r) / (a \cdot r)) \end{aligned} \tag{5.3.25}$$

Assuming that the first fraction (the terms are all constant) is reduced to the lowest terms as *pp / qq* , we could finally obtain:

$$(9000 \cdot i \cdot q) / (157 \cdot p \cdot j) = pp/qq \qquad \theta_{wd} = \mathrm{sqrt}(pp^2 \cdot (2 \cdot \mu \cdot a - \mu \cdot r) / (qq^2 \cdot a \cdot r)) \tag{5.3.26}$$

The convenience to move pp/qq under square root depends on their size. In the case of the values given by (5.3.20) and with standard wheels with a diameter of 56 mm, it holds (linear measures in centimetres):

$$\Delta t = 1/2 \qquad R_w = 28/10 \qquad (9000 \cdot 1 \cdot 10) / (157 \cdot 2 \cdot 28) = 11250 / 1099 \tag{5.3.27}$$

pp and qq are too big to be moved under square root. Let us try to find a simpler approximation:

$$11250 / 1099 \approx 10.2365 \approx 1024/100 = 256/25 \qquad (5.3.28)$$

Say isqrt and idiv respectively our available integer version of the square root and division, we obtain:

$$\theta_{wd} = isqrt(idiv(pp^2 \cdot (2 \cdot \mu \cdot a - \mu \cdot r), (qq^2 \cdot a \cdot r))) =$$
$$= isqrt(idiv(256^2 \cdot (2 \cdot 6000 \cdot 40 - 6000 \cdot r), (25^2 \cdot 40 \cdot r))) =$$
$$= isqrt(idiv(65536 \cdot (480 - 6 \cdot r), (25 \cdot r))) = \qquad (5.3.29)$$

The minimum value of θ_{wd} is reached at the aphelion when $r = a+f = 60$:

$$\theta_{wdMIN} = isqrt(idiv(65536 \cdot (480 - 6 \cdot 60), (25 \cdot 60))) =$$
$$= isqrt(idiv(7864320, 1500)) = isqrt(5242) = 72° \qquad (5.3.30)$$

corresponding to approximately 3.52 cm. The result is good because the precise value is 72.34. The maximum appears at the perihelion when r = a-f = 20:

$$\theta_{wdMAX} = isqrt(idiv(65536 \cdot (480 - 6 \cdot 20), (25 \cdot 20))) =$$
$$= isqrt(idiv(23592960, 500)) = isqrt(47185) = 217° \qquad (5.3.31)$$

Even here the result, which corresponds to a move of about 10.6 cm, is good because its precise value is 217.04.

Now, we consider the elementary rotation of the robot: to turn the robot of an angle ρ, we need to set to its maximum the steering parameter of a *move* block (100% with the correct direction) and make the motors to rotate of an angle equal to:

$$\theta_{wd} = 360 \cdot D_w \rho / (2\pi \cdot 2 \cdot R_w) = (90 \cdot D_w / (\pi \cdot R_w)) \rho \qquad (5.3.32)$$

The approximated values of ρ are given by (5.3.19) and (5.3.21), respectively for each one of the two identified parts of the orbit. Using the first formula:

$$\rho = ((\mu/r^2) \cdot \Delta t) / (sqrt(\mu \cdot ((2/r)-(1/a)))) = sqrt(\mu \cdot a \cdot \Delta t^2 / (r^3 \cdot (2a-r))) =$$
$$= sqrt(\mu \cdot a \cdot i^2 / (r^3 \cdot (2a-r) \cdot j^2)) \qquad (5.3.33)$$

$$\theta_{wd} = (90 \cdot (w/z) / ((314/100) \cdot (p/q))) sqrt(\mu \cdot a \cdot i^2 / (r^3 \cdot (2a-r) \cdot j^2)) =$$
$$= (4500 \cdot w \cdot q / (157 \cdot z \cdot p)) sqrt(\mu \cdot a \cdot i^2 / (r^3 \cdot (2a-r) \cdot j^2)) \qquad (5.3.34)$$

Using the already used data and with $D_w = w/z = 10$, we obtain:

$$\theta_{wd} = (450000 / (157 \cdot 28)) sqrt(6000 \cdot 40 / (r^3 \cdot (80-r) \cdot 4)) =$$
$$= (112500 / 1099) sqrt(60000 / (r^3 \cdot (80-r))) \qquad (5.3.35)$$

A 100 factor can be moved under square root, and we obtain:

$$\theta_{wd} = (1125 / (1099 \cdot r)) sqrt(600000000 / (r \cdot (80-r))) \qquad (5.3.36)$$

For the acceleration increases faster than the speed when r varies, the maximum angle occurs when the acceleration is at its maximum and, thus, having the minimum r (=20):

$$\begin{aligned}\theta_{wdMAX} = {} & \text{idiv((1125·isqrt(idiv(600000000, (20·(80-20))))),} \\ & \text{(1099·20)) =} \\ & \text{idiv((1125·isqrt(idiv(600000000, 1200))),} \\ & \text{(21980)) =} \\ & \text{idiv((1125·isqrt(500000)), 21980) =} \\ & \text{idiv((1125·707), 21980) = idiv(795375, 21980) = 36°} \quad (5.3.37)\end{aligned}$$

The result is good because the correct value is 36.17.

Now considering the second approximation of the formula (5.4.21), in the range of interest $30 \le r \le 50$, it approximately follows $10 \le v \le 15$, $1.25 \le a_G \cdot \Delta t \le 3.2$. With these values it is advisable to scale up of 100 in order to make some decimals significative for the integer calculations:

$$\rho = ((a_G\, \Delta t\, 90) / (100 \cdot v + a_G\, \Delta t\, 50) \qquad (5.3.38)$$

$$\begin{aligned}\theta_{wd} &= (90 \cdot (w/z) / ((314/100) \cdot (p/q))) \cdot (a_G \cdot (i/j) \cdot 90 / (100 \cdot v + a_G \cdot (i/j) \cdot 50)) = \\ &= (8100 \cdot 10 \cdot 6000 \cdot 1000) / (314 \cdot 28 \cdot 2 \cdot r^2 \cdot (\text{sqrt}(1500000 \cdot (80-r)/r) + (6000 \cdot 25/r^2)\,)) = \\ &= 27638762 / (r \cdot 10 \cdot \text{sqrt}(15000 \cdot r \cdot (80-r)) + 6000 \cdot 25) \qquad (5.3.39)\end{aligned}$$

With an intermediate value (r=a=40) and the usual truncations you obtain 13 degrees versus a precise 12.22.

What about the power to be applied to the motors during the two types of motions? As known, in the absence of excessive load, the power control is actually a speed control. Considering that the speed of the second motion, the rotation, is not so important to have the feeling of the simulated speed, which is originally continuous, in every straight-line motion step we would impose a speed able to make the robot move exactly in the step time Δt of the simulation. The angular speed to be set follows from (5.3.24):

$$\omega_{wd} = \theta_{wd} / \Delta t = 360 \cdot v \cdot / (2\pi \cdot R_w) \text{ degrees/s} \qquad (5.3.40)$$

with $\Delta t = 0.5$, $\omega_{wd} = 2 \cdot \theta_{wd}$. For with the given experimental data we have estimated $72 \le \theta_{wd} \le 217$, it would follow $144 \le \omega_{wd} \le 434$. Assuming true the already estimated relation:

$$P(\text{ower}) = (1/8.15) \cdot \omega \qquad (5.3.41)$$

we obtain $17.66 \le P \le 53.25$. Thus, if this range of powers is problematic, we could scale up or down the power of a given factor to maintain the same feeling of the motion, especially the increase of speed from aphelion to perihelion.

Finally, as regards the position of the sonar, we decided not to estimate step by step the viewing angle of the sonar in respect of the mutual orbiting body-sun position, because we didn't want to maintain a state variable describing the position of the robot. Instead, a more robust solution is to initially put the sonar with its axis orthogonal to the robot, and at the beginning of every step make the sonar sweep an angle of sufficient amplitude in the range of $\pm\alpha$, performing a certain number of readings and taking the minimum as a measure of the distance.

5.3.4 The program

We are describing the core of the program, which is an infinite loop every execution of which corresponding to a single step. One step is formed by three stages: the measurement of the distance, the straight-line motion, the rotation.

The first stage is represented by the code of fig. 5.3.8. Assuming that the sonar is mounted on a motor connected to port B, the scanning of the head is made as follows: an initial rotation of 60 degrees in one direction; 5 steps with one reading at the beginning of every step, an updating of the variable d to represent the minimum distance measured, a rotation of 30 degrees in the opposite direction at the end of the step. After this evaluation, the head is repositioned to its original direction with a final rotation of 90 degrees.

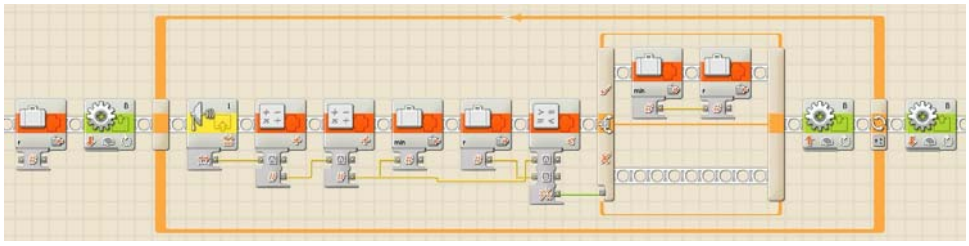

Fig. 5.3.8 – *Part I: measuring the distance*

```
VarDecl(Name=delta, Type=NUM) -- the angle to be performed
VarDecl(Name=min, Type=NUM) -- the minimum distance
VarDecl(Name=r, Type=NUM) -- the distance for the calculations
Var(Name=r.NUM, Act=WR, Val=255)  -- d=initial minimum
Motor(Port=B, Dir=BK, Act=CONST, Pwr=20, PwrCtrl=ON,
  Dur=60.DEG, Wait=ON, Next=BRK)
Loop1: Loop(Ctrl=FOREVER, Dis=OFF) [
        Loop2: Loop(Ctrl=COUNT, Until=5, ShowCnt=OFF) [
```

So: SonarSens(Port=1, Cmp=??, Show=CM)

Ad1:MathOp(Type=ADD, A=So.Dist, B=<SonarOffset>)

Ad2:MathOp(Type=ADD, A=Ad1.Res, B=<SunRadius>)

Var(Name=min.NUM, Act=WR, Val=Ad2.Res)

Vd1: Var(Name=r.NUM, Act=RD)

Cm1: CmpOp(Type=LT, A=Ad2.Res, B=Vd1.Val)

Sw1: Switch(Ctrl=VAL, Type=LOGIC, Dis=ON,
CondUp=TRUE, Val=Cm1.Res)

[Sw1.IF

Vm1: Var(Name=min.NUM, Act=RD)

Var(Name=r.NUM, Act=WR, Val=Vm1.Val)

Sw1.IF]

[Sw1.ELSE

Sw1.ELSE]

Motor(Port=B, Dir=FD, Act=CONST, Pwr=20,
PwrCtrl=ON, Dur=30.DEG, Wait=ON, Next=BRK)

Loop2]

Motor(Port=B, Dir=BK, Act=CONST, Pwr=20, PwrCtrl=ON,
Dur=90.DEG, Wait=ON, Next=BRK)

The second stage (see fig. 5.3.9) is the calculation of the angle to be performed for the straight motion based on the (5.3.29) formula. This piece of code is rather straightforward.

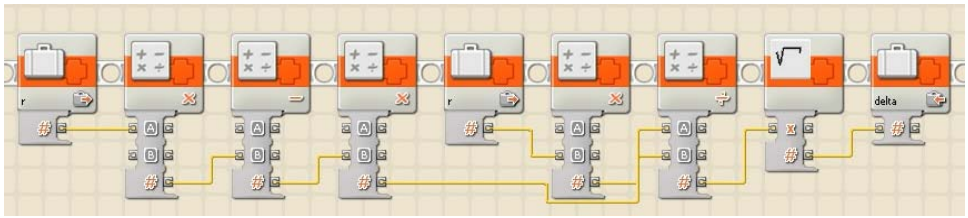

***Fig. 5.3.9** – Part II: calculating the angle for straight motion*

Vd2:Var(Name=r.NUM, Act=RD)

Mu1:MathOp(Type=MUL, A=Vd2.Val, B=6)

Su1:MathOp(Type=SUB, A=480, B=Mu1.Res)

Mu2:MathOp(Type=MUL, A=65536, B=Su1.Res)

Vd3:Var(Name=r.NUM, Act=RD)

Mu3:MathOp(Type=MUL, A=25, B=Vd3.Val)

Di1:MathOp(Type=DIV, A=Mu2, B=Mu3.Res)

Sq1:Sqrt(x1=Di1.Res)

Var(Name=delta.NUM, Act=WR, Val=Sq1.Res)

In order to give the impression of the variation of speed during the orbit, we apply a varying motor power in the range of 30÷60 (30 when the angle is minimum, i.e. 72°, 60 when it is maximum, i.e. 217°), linearly scaling the angle value (fig. 5.3.10):

$30 = \alpha \cdot 72 + \beta$ $60 = \alpha \cdot 217 + \beta$ $\Rightarrow$
(subtracting the first from the second) $30 = \alpha \cdot 145$ $\Rightarrow$ $\alpha = 30/145$
$\beta = 30 - 30 \cdot 72/145$
$pot = (30/145) \cdot \theta + 30 - 30 \cdot 72/145 = (4350 + 30 \cdot (\theta - 72)) / 145$ (5.4.42)

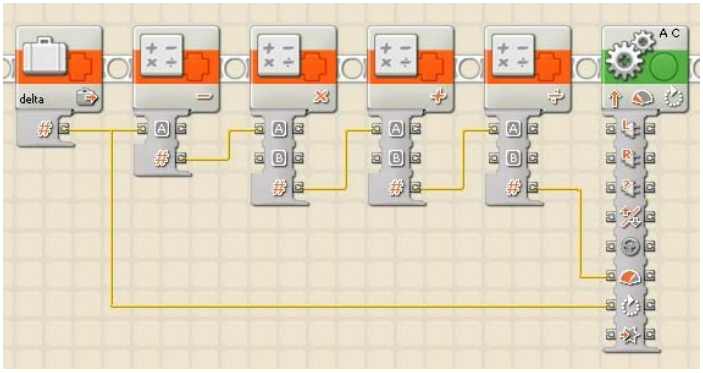

***Fig. 5.3.10** – Part III: simulating the variation of speed*

Vd4:Var(Name=delta.NUM, Act=RD)

Su2:MathOp(Type=SUB, A=Vd4.Val, B=72)

Mu4:MathOp(Type=MUL, A=Su2.Res, B=30)

Ad3:MathOp(Type=ADD, A=Mu4.Res, B=4350)

Di2:MathOp(Type=DIV, A=Ad3.Res, B=145)

After the straight motion, a short rotation is applied. We must distinguish the section in which we apply the simpler approximation of (5.3.19) from the section where we apply the more complex one (5.3.21). The choice is based on the value of

the measured distance from the focus: the simpler approximation is the case when such distance is out of the range of 30÷50 (fig. 5.3.11).

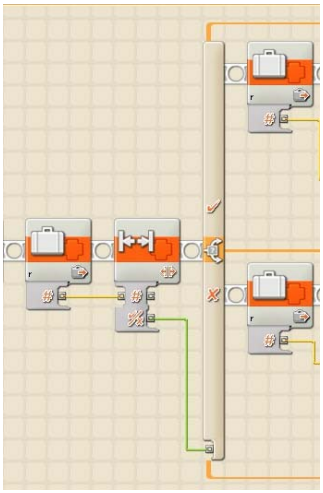

Fig. 5.3.11 *– Part IV: to distinguish the two approximations*

Vd5: Var(Name=r.NUM, Act=RD)

Ra1:Range(Type=OUT, A=30, B=50, Val=Vd5.Val)

Now, for the 'then part' we must apply (5.3.36) (fig. 5.3.12):

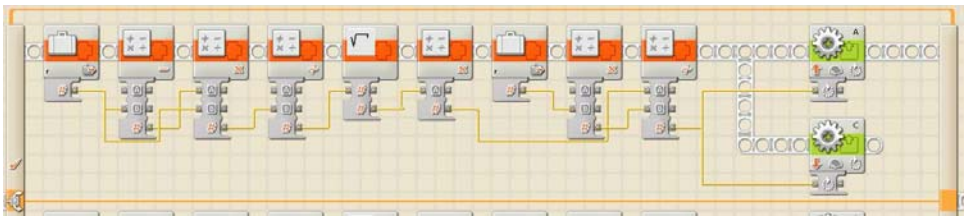

Fig. 5.3.12 – Part V: simpler approximation

Sw2: Switch(Ctrl=VAL, Type=LOGIC, Dis=ON,
CondUp=TRUE, Val=Ra1.Res)

[Sw2.IF

Vd6: Var(Name=r.NUM, Act=RD)

Su4:MathOp(Type=SUB, A=80, B=Vd6.Val)

Mu5:MathOp(Type=MUL, A=Su4.Res, B=Vd6.Val)

Di3:MathOp(Type=DIV, A=600000000, B=Mu5.Res)

Sq2:Sqrt(x1=Di3.Res)

```
        Mu6:MathOp(Type=MUL, A=Sq2.Res, B=1125)
        Vd7: Var(Name=r.NUM, Act=RD)
        Mu7:MathOp(Type=MUL, A=2156, B=Vd7.Val)
        Di4:MathOp(Type=DIV, A=Di3.Res, B=Mu7.Res)
        {1
            Motor(Port=A, Dir=FD, Act=CONST, Pwr=15,
              PwrCtrl=ON, Dur=Di4.Res.DEG, Wait=ON,
              Next=BRK)
        1}
        {2
            Motor(Port=C, Dir=BK, Act=CONST, Pwr=15,
              PwrCtrl=ON, Dur=Di4.Res.DEG, Wait=ON,
              Next=BRK)
        2}
    Sw2.IF]
```

The more complex approximation follows (fig. 5.3.13).

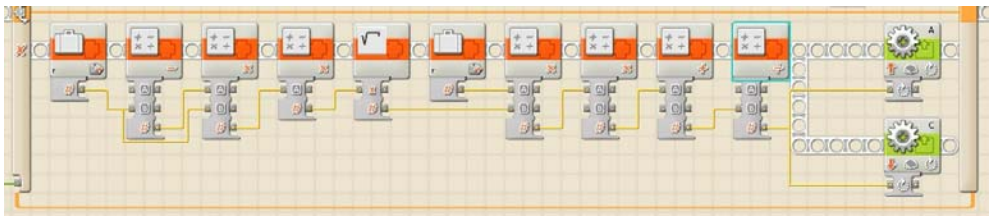

Fig. 5.3.13 – Part VI: more complex approximation

```
    [Sw2.ELSE
        Vd7: Var(Name=r.NUM, Act=RD)
        Su5:MathOp(Type=SUB, A=80, B=Vd7.Val)
        Mu6:MathOp(Type=MUL, A=Su5.Res, B=Vd7.Val)
        Mu7:MathOp(Type=MUL, A=Mu6.Res, B=15000)
        Sq3:Sqrt(x1=Mu7.Res)
        Vd8: Var(Name=r.NUM, Act=RD)
        Mu8:MathOp(Type=MUL, A=Vd8.Val, B=Sq3.Res)
```

```
            Mu9:MathOp(Type=MUL, A=Mu8.Res, B=10)
            Ad4:MathOp(Type=ADD, A=Mu9.Res, B=150000)
            Di5:MathOp(Type=DIV, A=27638762, B=Ad4.Res)
            {1
                  Motor(Port=A, Dir=FD, Act=CONST, Pwr=15,
                    PwrCtrl=ON, Dur=Di5.Res.DEG, Wait=ON,
                    Next=BRK)
            1}
            {2
                  Motor(Port=C, Dir=BK, Act=CONST, Pwr=15,
                    PwrCtrl=ON, Dur=Di5.Res.DEG, Wait=ON,
                    Next=BRK)
            2}
      Sw2.ELSE]
Loop1]
```