8/2/62

TO: R. E. Ekstrom

SUBJECT: Writeup of Problem 6056 - 26 BIT ORBIT CALCULATION.

1. PROBLEM DESCRIPTION:

The purpose of the program is to determine the accuracy which the Gemini Computer can achieve in computing spacecraft position for one orbit in advance. It is not known at this time whether it must be done in real time or whether an arbitrary step size can be used. If it is not, there is still a limitation on the smallness of the step size due to the slow cycle time of the Gemini Computer.

The heart of the program lies in doing the computations on the 7090 using fixed point, 26-bit arithmetic.

The equations of motion which are programmed are those used for Reentry. The results are to compared with

those obtained by using the potential function for a non-homogenous spheroid using six terms.

At present, the method of attack was to try trapezoidal integration with various step sizes. This proved to be no good immediately. The next approach was to incorporate, under sense switch 2 control, a Modified Euler method. This also is not too good. I ran the problem in 36 bits, Modified Euler, and found it to be pretty good. About 150 feet error in the final radius. This was with a one second step size.

It appears that the next thing to do is incorporate some form of double ~~precision~~ register computation in the integration routines and see if it can eliminate the round-off error which is evidently

Creeping in.

2. SYMBOLS AND DEFINITIONS

FLOW CHART SYMBOL	PROGRAM NAME	DEFINITION
X	X	inertial X-axis
Y	Y	" Y- "
Z	Z	Z- "
$\dot{X}$	XDOT	inertial X velocity
$\dot{Y}$	Y DOT	" Y "
$\dot{Z}$	Z DOT	" Z "
R_s	RS	Radius to spacecraft $(\sqrt{X^2+Y^2+Z^2})$
ΔT	DT	step size or comp cycle length
T_P	TP	orbital period
LC1	LC1	A control word to determine the first pass thru the program
LC4	LC4	A control word used to indicate whether 26 bit (LC4=0) or 36 bit (LC4=222) computation is to be done.
T	T	accumulated time
TPRNT	TPRNT	accumulated time up to 10 sec which indicates when to print.

FIXPT	FIXPT	7094 point subroutine name.
Q	Q	$-\frac{G^{1/3}}{r_S}$
R^2	RSQ	$\left(\frac{r_E}{r_S}\right)^2$
S S^2	S SSQ	$\frac{z}{r_S}$
F_{xg}	FXG	X gravitational acceleration
F_{yg}	FYG	Y " "
F_{zg}	FZG	Z " "
Q^3	QCUBE	
LG-5	LC5	A control word used in modified Euler integration.
D_{TT}	DTT	$\frac{\Delta T^2}{2}$
Fxg_{L-1}	FXGM1	
Fyg_{L-1}	FYGM1	
Fzg_{L-1}	FZGM1	
X_{L-1}	XIMO	
Y_{L-1}	YIMO	
Z_{L-1}	ZIMO	

$\dot{X}_{L-1}$	XDOTM1	
$\dot{Y}_{L-1}$	YDOTM1	
$\dot{Z}_{L-1}$	ZDOTM1	
SQR	SQR	result of square root iteration
KSQR	KSQR	First guess of the square root iteration.
LC 3	LC3	a control word used to sequence thru the square root routine for Y_S and R'
ATR	ATR	Part of the argument for square root. Either $\frac{Y}{X}$ or $\frac{X}{Y}$ depending on the size of X compared to Y
LC2	LC2	A control word used to determine the final calculation of Y_S or R'
SQRA		Total argument of the square root routine $(1 + ATR^2)$

R'	RPRIME	Intermediate value used to compute Y_S
TFLT	TFLT	Floating point value of T

CONSTANTS

Y_E	RE	Mean Radius of earth
$G^{1/3}$	G13	Cube root of G
J	J	
K3	K3	Used to calculate F_{xg} and F_{yg}
K5	K5	" " " "
K6	K6	" " " "
K10	K10	" " F_{zg} "
K8	K8	Part of the square root argument (=1)
K9	K9	Used to determine when accuracy of square root iteration is sufficient.

3. PROGRAM FLOW:

a.) Read in the data cards

$x, y, z, \dot{x}, \dot{y}, \dot{z}, r_s, \Delta T, T_P, LC1, LC4$

b.) Set $T=0$ and $TPRNT=0$ to initialize

c.) Convert the inputs to fixpoint using Joe Constable's FIXER-FLTR

d.) Enter the fix point subroutine

e.) Test LC4 to determine whether the mask is to be for 26 bits or 36 bits.

f.) Mask all of the variables and constants to 26 or 36 bits

$x, y, z, \dot{x}, \dot{y}, \dot{z}, \Delta T, r_s, G^{1/3}, r_E, J$

g.) Calculate $Q = -\frac{(G^{1/3})}{r_s}$

Since G is such a large number ($20.925\,738 \times 10^6$) it would be very inaccurate with only 26 bits. Therefore the cube root is taken and this is divide by r_s.

Then, in F_{xg}, F_{yg} and F_{zg} $Q^3 = -\frac{G}{r_s^3}$

h.) In Eq K_{10} is added to the quantity inside the brackets to obtain a 3 as required by the equations.

Sheet (2)

i.) Sheet 2 contains the integrations and either trapezoidal or Modified Euler ~~can~~ can be used depending on sense switch 2. (up = trapezoidal; down - Mod Euler)

The logic flow is straight forward for trapezoidal. LC5 will be + and SSW 2 up. LC1 is initially plus so that the first step is rectangular. Then LC1 is set negative and Trapezoidal is used from there on.

For Modified Euler, the logic is set up to compute the following:

EQ (1) $\ddot{X}_{i-1} = f(X)$

" (2) $X = X + \Delta T\, \dot{X}$

" (3) $\ddot{X}_i = f(X)$

" (4) $X = X + \frac{\Delta T^2}{2}\left[\ddot{X}_{i-1}\right] = X_{(i-1)} + \Delta T \cdot \dot{X} + \frac{\Delta T^2}{2}\cdot\left[\ddot{X}_{i-1}\right]$

" (5) $\dot{X} = \dot{X} + \frac{\Delta T}{2}\left[\ddot{X}_{i-1} + \ddot{X}_i\right]$

$\ddot{X}_{(t)} = f(X)$

$X^p_{(t+h)} = x(t) + h\dot{X}(t)$

$\ddot{X}^p_{(t+h)} = f\left(X^p_{(t+h)}\right)$

$X^c_{(t+h)} = x(t)$

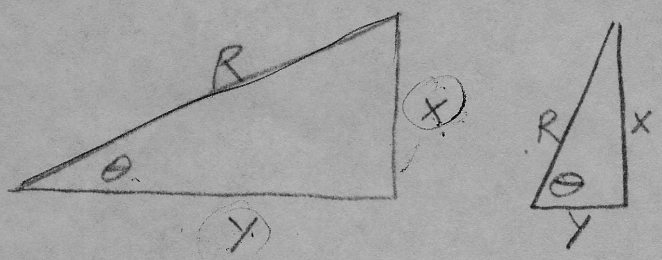
the program flow is as follows:
LC5 is initially + and SSW 2 is down and $F_{xg_{i-1}}$, $F_{yg_{i-1}}$, $F_{zg_{i-1}}$ are set up. This gives EQ 1. LC5 is set negative and EQ 2 is computed using part of the rectangular integration equations. SSW 2 is down so the program goes back and computes F_{xg}, F_{yg} and F_{zg} with the new X, Y, Z to give EQ 3.

LC5 is again tested and is now negative. D_{TT} is computed and used to compute EQ 4. Then EQ 5 is ~~used~~ computed using the trapezoidal velocity integration. SSW 2 is down and the program is back in the normal flow.

j.) On the next column on sheet 2 the first box is setting all the "i" values to "i-1" for trapezoidal integration. This is by passed for Modified Euler.

k. The next step is to compute $r_3 = \sqrt{X^2+Y^2+Z^2}$ Since these values are so large and scaled B25, squaring them ~~and adding~~

would loose to many bits. So, different method is used to get r_s. Taking for example, a 2 dimensional case where it is desired to compute $R = \sqrt{x^2+y^2}$ and x & y are of the magnitude that squaring them would loose accuracy, the following can be done.



a. IF $Y > X$:

(1) Compute $\text{Tan}\theta = \frac{X}{Y}$.

(2) " $\text{SEC}\theta = \sqrt{1+\text{Tan}^2\theta} = \sqrt{1+\frac{X^2}{Y^2}}$

(3) then $R = \frac{Y}{\cos\theta} = Y \text{ SEC } \theta$

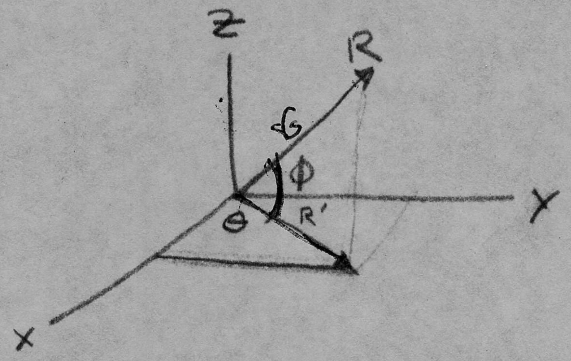
b. IF $X > Y$

(1) Compute $\text{CTN}\,\theta = \frac{Y}{X}$

(2) " $\text{CSC}\,\theta = \sqrt{1+\text{CTN}^2\theta} = \sqrt{1+\frac{Y^2}{X^2}}$

(3) then $R = \frac{X}{\sin\theta} = X \text{ CSC } \theta$

For three dimensions then, it is a matter of doing these computations twice.



First compute R' using x and y. Then Compute R using R' and Z in a similar fashion.

Therefore, in COL 5 SQR is set equal to KSQR which is all 7's to set up the first guess on the square root of $1+ATR^2$. LC 3 is set + which means this is the first pass thru the square root routine to attain R'. |X|, |Y| are compared to determine which is larger and make the larger one the denominator to prevent overflow in the division. The right shift of one is in case they are equal.

If X > Y LC 2 is set negative to indicate that the case "b" described above is used to compute R'. If Y > X then LC2 is set + to indicate case "a" is used. Then the quantity $1 + \text{Tan}^2\theta$ or $1 + \text{CTN}^2\theta$ is computed and used as the argument for the square root. The square root is then computed using Newtons method. The next box is to determine whether the square root is accurate to 23 bits.

LC 2 is then tested to determine the equation to be used for R'.

LC 3 is tested. It is now + (set + in Col 5) so X and Y are replaced by Z and R'; LC 3 is set negative to indicate this is the computation of Y_S and SQR is reinitialized to all 7's.

8.

The same tests and logic settings are performed again in Col 5 and a new square root (now $\sqrt{1+\tan^2\Phi}$ or $\sqrt{1+CTN^2\Phi}$) is computed and the same equation used to compute R' which will now be Y_S.

LC 3 is now negative so the flow goes to COL 7 where X and Y are restored and $Y_S = R'$.

T is updated by ΔT and the program then returns to the Fortran section.

The necessary variables are converted to floating point for printing and computation.

The rest is just the determination of the end of the period and when to print.

RESULTS TO DATE:

METHOD	ERROR IN POSITION RADIUS AFTER 5370 SEC. (5370 SEC).
26 BIT TRAPEZOIDAL ΔT = 10 SEC	3,162,525 ft.
26 BIT TRAPEZOIDAL ΔT = 1 SEC	1,316,289 ft
26 BIT TRAPEZOIDAL ΔT = .3 SEC	83,969 ft
26 BIT TRAPEZOIDAL ΔT = .1 SEC	2,112 ft
26 BIT MODIFIED EULER ΔT = 1 SEC	8,165 ft
36 BIT MODIFIED EULER ΔT = 1 SEC	197 ft

SUGGESTED CHANGES AND APPROACHES

1.) The scaling of ΔT is now B4 to allow a step size of 10 sec. I think this should be lowered to about B2. This will effect the computations of $x, y, z, \dot{x}, \dot{y}, \dot{z}$ and T. Also, the scaling arguments in the FIXER-FLTR routines must be changed.

2.) Try a larger step size than 1 sec with Modified Euler.

3.) Install double register computation in the integrations.

4.) Try Runge-Kutta. (I have a flow diagram of the necessary computations).

Item 3 is the one to try first.