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 which are programmed are those used for reentry. The results are to be compared with
those obtained by using the potential function for a non-homogeneous spheroid using " recuper " 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 1 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
2. **Symbols and Definitions**

<table>
<thead>
<tr>
<th>Flow Chart Symbol</th>
<th>Program Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{x} )</td>
<td>( \mathbf{x} )</td>
<td>inertial x-axis</td>
</tr>
<tr>
<td>( \mathbf{y} )</td>
<td>( \mathbf{y} )</td>
<td>&quot;y&quot;</td>
</tr>
<tr>
<td>( \mathbf{z} )</td>
<td>( \mathbf{z} )</td>
<td>&quot;z&quot;</td>
</tr>
<tr>
<td>( \dot{x} )</td>
<td>( \dot{x} )</td>
<td>inertial x velocity</td>
</tr>
<tr>
<td>( \dot{y} )</td>
<td>( \dot{y} )</td>
<td>&quot;y&quot;</td>
</tr>
<tr>
<td>( \dot{z} )</td>
<td>( \dot{z} )</td>
<td>&quot;z&quot;</td>
</tr>
<tr>
<td>( \mathbf{r}_s )</td>
<td>( \mathbf{r}_s )</td>
<td>Radius to spacecraft ((\sqrt{x^2+y^2+z^2}))</td>
</tr>
<tr>
<td>( \Delta t )</td>
<td>( \Delta t )</td>
<td>step size or comp cycle length</td>
</tr>
<tr>
<td>( T_p )</td>
<td>( T_p )</td>
<td>Orbital period</td>
</tr>
<tr>
<td>( LC1 )</td>
<td>( LC1 )</td>
<td>A control word to determine the first pass thru the program</td>
</tr>
<tr>
<td>( LC4 )</td>
<td>( LC4 )</td>
<td>A control word used to indicate whether 26 bit (LC4=0) or 36 bit (LC4=222) computation is to be done, accumulated time</td>
</tr>
<tr>
<td>( T )</td>
<td>( T )</td>
<td>Accumulated time up to 10sec, which indicates when to print</td>
</tr>
<tr>
<td>( TPRNT )</td>
<td>( TPRNT )</td>
<td></td>
</tr>
</tbody>
</table>
Fixed point subroutine name.

\[ -\frac{Gy_3}{Y_5} \]

\[ \left(\frac{Ye}{Y_5}\right)^2 \]

\[ \frac{2}{Y_5} \]

\[ x \text{ gravitational acceleration} \]

\[ y \]

\[ z \]

A control record used in modified Euler integration.

\[ \frac{\Delta T^2}{2} \]
\(X_{l-1}\) \(X_{D0TML}\)
\(Y_{l-1}\) \(Y_{D0TML}\)
\(Z_{l-1}\) \(Z_{D0TML}\)

\(SQR\) \(SQR\)
\(K_SQR\) \(K_{SQR}\)

\(LC_3\) \(LC_3\)
\(ATR\) \(ATR\)

\(LC_2\) \(LC_2\)

Result of square root iteration
First guess of the square root iteration.
A control word used to sequence the square root routine for \(Y_S\) and \(R\).
Part of the argument for square root. Either \(x\) or \(\frac{x}{y}\) depending on the size of \(x\) compared to \(y\).
A control word used to determine the final calculation of \(Y_S\) or \(R\).

Total argument of the square root routine \((1 + ATR^2)\)
Intermediate values used to compute \( r_s \)
Floating point value of \( T \)

Mean radius of Earth
Cube root of \( G \)

\( R' \)
\( TFLT \)
\( TFLT \)

\underline{CONSTANTS}

\( Y_e \)
\( G' \frac{3}{2} \)
\( J \)

\( K_3 \)
\( K_5 \)
\( K_6 \)
\( K_{1\phi} \)
\( K_8 \)
\( K_9 \)

\( RE \)
\( G_{1.3} \)
\( J \)

\( K_3 \)
\( K_5 \)
\( K_6 \)
\( K_{1\phi} \)
\( K_8 \)
\( K_9 \)

\( \text{Used to calculate } F_{xg} \text{ and } F_{yg} \)
\( \ldots \quad \ldots \quad \ldots \quad F_{xg} \)

Part of the square root argument \((\equiv 1)\)

\( \text{Used to determine when accuracy of square root iteration is sufficient.} \)
3. **Program Flow**:

a.) Read in the data cards
   
   \[ x, y, z, x, y, z, y_s, \Delta T, T_p, L_{C1}, L_{C4} \]

b.) Set \( T = 0 \) and \( T_{PRINT} = 0 \) to initialize

c.) Convert the inputs to fixed-point using Joe Constable's `FIXER-FLTR`

d.) Enter the fixed-point subroutine

e.) Test \( L_{C4} \) 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
   
   \[ x, y, z, x, y, z, \Delta T, y_s, G^{1/3}, y_e, l \]

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

Since \( G \) is such a large number (20,925,738 x 10^6) it would be very inaccurate with only 26 bits. Therefore the cube root is taken and this is divided by \( y_s \).

There, in \( F_x, F_y \) and \( F_z \) \( Q^{3} = \frac{G}{y_s^{3}} \)
L) In Fig. 1, if $K\theta$ is added to the quantity inside the brackets to obtain a 3 as required by the equations.

Sheet 2

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

The logic flow is straightforward for trapezoidal. LC5 will be + and SSW $\omega$ up. LC1 is initially plus 3 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:

$$\begin{align*}
\text{EQ} & \quad (1) \quad \ddot{x}_v = f(x) \\
& \quad (2) \quad x = x + \Delta T \ddot{x} \\
& \quad (3) \quad \ddot{x}_i = f(x) \\
& \quad (4) \quad x = x + \frac{\Delta T}{2} [\ddot{x} \Delta t - \dot{x}] = x + \Delta T \ddot{x} + \frac{\Delta T^2}{2} \dddot{x} \\
& \quad (5) \quad \ddot{x} = \ddot{x} + \frac{\Delta T}{2} [\dddot{x} + \dddot{x}] 
\end{align*}$$
The program flow is as follows:

LC5 is initially + and SSW2 is down and Fxg, Fyg, Fzg, are set up. This gives EQ 1. LC5 is set negative and EQ 2 is computed using part of the rectangular integration equations. SSW2 is down so the program goes back and computes Fxg, Fyg, and Fzg with the new x, y, z to give EQ 3.

LC5 is again tested and is now negative. D17 is computed and used to compute EQ 4. Then EQ5 is tested computed using the trapezoidal velocity integration. SSW2 is down and the program is back in the normal flow.

j) In 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 \( X3 = \sqrt{x + y^2 + z^2} \)

Since these values are so large and scaled by 825, squaring them and adding
would loose to many bits. So different method is used to get $R$. 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.

\[ R = \sqrt{x^2 + y^2} \]

a. IF $y > x$:

1. Compute $\tan \theta = \frac{x}{y}$
2. " $\sec \theta = \sqrt{1 + \tan^2 \theta} = \sqrt{1 + \frac{x^2}{y^2}}$
3. then $R = \frac{y}{\cos \theta} = y \sec \theta$

b. IF $x > y$

1. Compute $\cot \theta = \frac{y}{x}$
2. " $\csc \theta = \sqrt{1 + \cot^2 \theta} = \sqrt{1 + \frac{y^2}{x^2}}$
3. then $R = \frac{x}{\sin \theta} = x \csc \theta$
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 \( 7/8 \) to setup the first guess on the square root of \( 1 + ATR^2 \).

LC 3 \( \phi \) set + which means this is the first pass thru the square root routine to obtain \( R' \) (\( [1, x, y] \)) 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 \( x > y \) \( L2 \) is set negative to indicate that the case \( \text{"b"} \) described above is used to compute \( R' \). If \( y > x \) then \( L2 \) is set + to indicate case \( \text{"a"} \) is used. Then the quantity \( 1 + \tan^2 \theta \) or \( 1 + \cot^2 \theta \) is computed and used as the argument for the square root. The square root is then computed using Newton's method.

The next box is to determine whether the square root is accurate to 23 bits.

\( L2 \) is then tested to determine the equation to be used for \( R' \).

\( L3 \) is tested. It is now + (set + in Col 5) so \( x \) and \( y \) are replaced by \( z \) and \( R' \); \( L3 \) is set negative to indicate this is the computation of \( Y3 \) and \( \text{SQRT} \) is reinitialized to all \( 7's \).
The same tests and logic settings are performed again in Col 5 and a new square root (now $\sqrt{1 + \tan^2 \Phi}$ or $\sqrt{1 + \cot^2 \Phi}$) is computed and the same equation used to compute $R'$ which will now be $Y_5$.

If $3.$ is now negative so the flow goes to Col 7 where $X$ and $Y$ are restored and $Y_5 = R'$.

This updated by FT 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

<table>
<thead>
<tr>
<th>METHOD</th>
<th>ERROR IN POSITION RADIUS AFTER 5370 SEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>26 BIT TRAPEZOIDAL $\Delta T = 10$ SEC</td>
<td>3,162,525 ft</td>
</tr>
<tr>
<td>26 BIT TRAPEZOIDAL $\Delta T = 1$ SEC</td>
<td>1,316,289 ft</td>
</tr>
<tr>
<td>26 BIT TRAPEZOIDAL $\Delta T = 0.3$ SEC</td>
<td>83,969 ft</td>
</tr>
<tr>
<td>26 BIT TRAPEZOIDAL $\Delta T = 0.1$ SEC</td>
<td>2,112 ft</td>
</tr>
<tr>
<td>26 BIT MODIFIED EULER $\Delta T = 1$ SEC</td>
<td>8,165 ft</td>
</tr>
<tr>
<td>36 BIT MODIFIED EULER $\Delta T = 1$ SEC</td>
<td>197 ft</td>
</tr>
</tbody>
</table>
SUGGESTED CHANGES AND APPROACHES

1.) The scaling of $\Delta T$ is now 84 to allow a step size of 10 sec. I think this should be lowered to about 12. This will affect the computations of $\chi, \gamma, z, x, y, z$ 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 computations in the integrations.

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

Item 3 is the one to try first.