Chapter 4

Solution Technique

4.1 Vehicle Model Format

The model is a lumped mass representation of an automotive vehicle. As seen in Figure 4-1 the model is made up of five masses: the large sprung mass, and four smaller unsprung masses. The sprung mass has six rigid body degrees-of-freedom (dof) and each unsprung mass has an essentially vertical dof. The last dof is the steering angle. This adds up to a total of eleven dof's

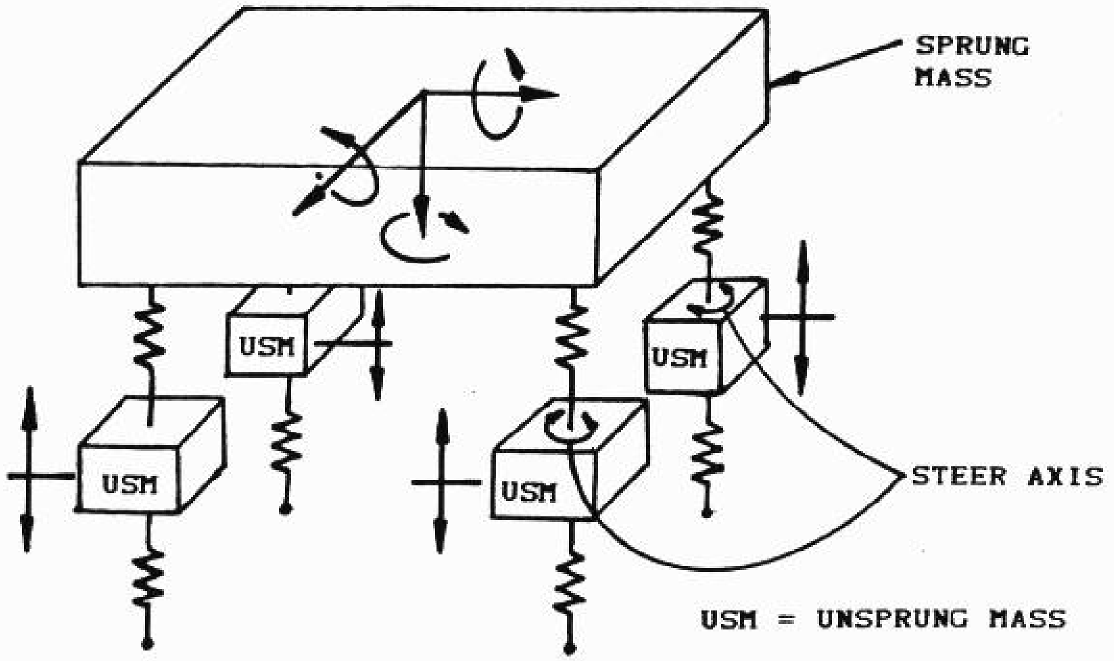

Figure 4-1: Ten degree-of-freedom vehicle model

but in chapter three, fourteen are described. The difference is the model developed uses spindle forces as the unknowns (three at each spindle) instead of the dof's described above which are displacements. To further explain, the sprung mass is supported by four reaction forces at the wheels which makes it a statically indeterminate system. Typically, the constitutive relations of the suspension springs are used to eliminate the force unknowns in favor of rigid body displacements unknowns. This would reduce the number of unknowns and eliminate the indeterminacy. However, in this case the forces are favored as unknowns for the following reasons:

- implementation of complex suspension models that influence the anti-features (anti-dive and anti-squat) is easier;
- the drive train constraints are easier to describe in terms of forces;
- the lateral tire forces are functions of longitudinal and normal spindle forces not displacements; and
- the unsprung mass equations-of-motion are in a simpler form if left as a function of forces.

As a result three more unknown forces are required which brings the total number of unknowns to fourteen.

4.2 Newton Raphson Iteration

For the fourteen unknowns described above the fourteen equations derived in chapter three must be solved. The fourteen equations are of the form $\underline{K}\ \underline{x} = \underline{r}$ where $\underline{K}$ is the system matrix, $\underline{x}$ is the vector of unknowns, and $\underline{r}$ is the vector of externally applied inertia loads. Because the equations are non-linear $\underline{K}$ is a function of $\underline{x}$. This implies that an iterative technique must be implemented to solve the equations. Newton-Raphson is such a technique as described by Bathe [20].

The first step in the Newton-Raphson procedure is to rewrite the equations as general functions in the form, $\underline{K}\ \underline{x} - \underline{r} = 0$. Theses functions are noted as $\underline{f}$ and are evaluated at an initial guess of $\underline{x}_o$. This calculation creates a vector of out-of-balance forces (or error) because this initial guess will not exactly satisfy the $\underline{f}$ functions. The next step involves the use of the Jacobian matrix $[\delta\underline{f}/\delta\underline{x}]$ which is an n x n matrix of partial derivatives (derivatives of the functions, $\underline{f}$, with respect to the unknowns, $\underline{x}$). The Jacobian is used to calculate the corrections to the initial guess by noting that $[\delta\underline{f}/\delta\underline{x}]\ d\underline{x} = -\underline{f}$. Then by inversion and multiplying, the correction factors are $d\underline{x} = [\delta\underline{f}/\delta\underline{x}]^{-1}(-\underline{f})$. So, the vector of initial guesses are updated by the correction factors, $\underline{x}_1 = \underline{x}_o + d\underline{x}_o$ and in general after each iteration the correction is $\underline{x}_{n+1} = \underline{x}_n + d\underline{x}_n$. To summarize, we have three relations to use in the Newton Raphson iteration:

$$\underline{K}\ \underline{x} - \underline{r} = \underline{f} = \text{functions;} \qquad (4-1)$$

$$d\underline{x} = [\delta\underline{f}/\delta\underline{x}]^{-1}(-\underline{f}) = \text{correction factors; and} \qquad (4-2)$$

$$\underline{x}_{n+1} = \underline{x}_n + d\underline{x}_n = \text{corrected unknowns.} \qquad (4-3)$$

The iteration is continued until the function values, $\underline{f}$, and the correction factors, $d\underline{x}$, go to zero.

The computation necessary to solve equation (4-2) is done by a subroutine called SIMQ. SIMQ is an IBM scientific subroutine that was converted from fortran to basic. SIMQ does Gauss elimination and back substitution as described in the program documentation:

PURPOSE

OBTAIN A SOLUTION OF A SET OF SIMULTANEOUS LINEAR EQUATIONS AX=B. A IS NxN

METHOD

METHOD OF SOLUTION IS BY ELIMINATION USING LARGEST PIVOTAL DIVISOR. EACH STAGE OF ELIMINATION CONSISTS OF INTERCHANGING ROWS WHEN NECESSARY TO AVOID DIVISION BY ZERO OR SMALL ELEM.

THE FORWARD SOLUTION TO OBTAIN THE VARIABLE N IS DONE IN N STAGES. THE BACK SOLUTION FOR THE OTHER VARIABLES IS CALCULATED BY SUCCESSIVE SUBSTITUTIONS. FINAL SOLUTION VALUES ARE DEVELOPED IN VECTOR B, WITH VARIABLE 1 IN B(1), VARIABLE 2 IN B(2) ..... VARIABLE N IN B(N). IF NO PIVOT CAN BE FOUND EXCEEDING A TOLERANCE OF 1.0E-20 THE MATRIX IS CONSIDERED SINGULAR AND KS IS SET TO 1. THIS TOLERANCE CAN BE MODIFIED BY REPLACING THE FIRST STATEMENT.

This procedure is more efficient than actually doing the matrix inversion and multiplication. SIMQ is called at each iteration step and is the core of the iteration routine.

4.3 Enhancement of Computational Efficiency by Variable Elimination

Several of the fourteen system equations are rather simple, therefore, eight of the unknowns can be solved for closed form. This will increase the computational efficiency of the computer model by only having to deal with a six-by-six Jacobian matrix. Eight of the fourteen unknowns are solved for in terms of the other six as outlined below.

The six unknowns that will be assumed, for the initial guess

$\underline{X}_O$, are: F_{YTFR}, F_{YTFL}, F_{YBL}, F_{ZBL}, BETA, and DEL. So the first equation can only be a function of these six. Then subsequent equations can be a function of the variable previously solved for and the six assumed variables.

First, the drive train constraint equations allow all the longitudinal spindle forces to be solved for in terms of F_{YTFR} and F_{YTFL} which are guessed values. The equations are (3-43), (3-44), or (3-45) depending on the drivetrain configuration as shown below. The equations are listed in the order in which they would be solved in the computer model. The equations for rear wheel drive are:

$$F_{XTFR} = 0.0 \qquad (3\text{-}43a)$$

$$F_{XTFL} = 0.0 \qquad (3\text{-}43b)$$

$$F_{XBR} = 1/2\ (\ SFX + (\ F_{YTFR} + F_{YTFL})\ \sin(DELTA)) \qquad (3\text{-}43d)$$

$$F_{XBL} = F_{XBR} \qquad (3\text{-}43c)$$

The equations for front wheel drive are:

$$F_{XBR} = 0.0 \qquad (3\text{-}44a)$$

$$F_{XBL} = 0.0 \qquad (3\text{-}44b)$$

$$F_{XTFR} = \frac{(SFX + (\ F_{YTFR} + F_{YTFL})\ \sin(DELTA)\ )}{2\ \cos(DELTA)} \qquad (3\text{-}44d)$$

$$F_{XTFL} = F_{XTFR} \qquad (3\text{-}44c)$$

The equations for braking are:

$$F_{XBR} = \frac{(SFX + (F_{YTFL} + F_{YTFR})\ \sin(DELTA))}{\dfrac{2\ \ (1 + (BP\ \cos(DELTA)))}{(1 - BP)}} \qquad (3\text{-}45d)$$

$$F_{XTFR} = \frac{F_{XBR}\ \ BP}{(1 - BP)} \qquad (3\text{-}45c)$$

$$F_{XTFL} = F_{XTFR} \qquad (3\text{-}45a)$$

$$F_{XBL} = F_{XBR} \qquad (3\text{-}45b)$$

Next, equation (3-22) was solved for F_{ZFR} and substituted into equation (3-24) to evaluate F_{ZBR} in terms of known quantities.

$$F_{ZBR} = [ST_Y + SF_Z\ L1 - F_{ZBL}\ (L2 + L1) - F_{XBR}\ H_{BR}$$
$$- (F_{XTFR}\ H_{FR} + F_{XTFL}\ H_{FL}\ ) \cos(DELTA)$$
$$+ (F_{YTFR}\ H_{FR} + F_{YTFL}\ H_{FL}\ ) \sin(DELTA)$$
$$- F_{XBL}\ H_{BL}\ ] \ /\ (L1 + L2) \qquad (4\text{-}4)$$

Equation (3-41) was solved for F_{ZFR} and substituted into equation (3-21) to evaluate F_{ZFL} in terms of known quantities.

$$F_{ZFL} = [\ (SF_Z - F_{ZBR}\ (1 + C) - F_{ZBL}\ (1 - C)\ ] \qquad (4\text{-}5)$$

Equation (3-22) is used to evaluate F_{ZFR}.

$$F_{ZFR} = SF_Z - F_{ZBR} - F_{ZBL} - F_{ZFL} \qquad (4\text{-}6)$$

Finally, equation (3-21) is solved to get F_{YBR}.

$$F_{YBR} = SF_Y - F_{YBL} - (F_{XTFR} + F_{XTFL}) \sin(DELTA)$$
$$- (F_{YTFR} + F_{YTFL}) \cos(DELTA) \qquad (4\text{-}7)$$

To summarize, the above equations are used to solve for, F_{XTFR}, F_{XTFL}, F_{XBR}, F_{XBL}, F_{ZBR}, F_{ZFL}, F_{ZFR}, and F_{YBR} respectively in closed form by a specific progression through the equations. By doing this only six variables are involved in the iteration routine. The variables F_{YTFR}, F_{YTFL}, F_{YBL}, F_{ZBL}, BETA, and DEL are iterated by using the six remaining equations in the Newton Raphson routine discussed in section 4.2

4.4 Program Flow Chart

Shown below is a flow chart of the combined maneuver program called, COMMAN.

```
                 DEFINE VARIABLE PRECISION
                     DIMENSION ARRAYS
              OPEN INPUT FILE READ INPUT AND
                     SAVE INITIAL GUESS
                              |
                              |
                              |
                              |
                              |
         --->START VEHICLE STEADY STATE VELOCITY LOOP
         |                    |
         |    OPEN OUTPUT FILES ON INITIAL LOOP
********>|                    |
*        |------->START ACCELERATION LOOP
*        ----------->START THETA LOOP
*                             |
*                             |
*              CALCULATE CONSTANT VALUES AND
*          ECHO INPUT TO THE SCREEN AND TO THE OUTPUT FILES
*                             |
*       >>>>>>>>>START NEWTON RAPHSON LOOP
*       ^                     |
*       ^                     |
*       ^       CALCULATE INITIAL
*       ^        FUNCTION VALUES BY
*       ^     CALLING THE SYSTEM EQUATIONS
^       ^      |    ^ |   ^ |    ^ |
*       ^      |    | |   | |    |  -------> CALL DRIVETRAIN
*       ^      |    | |   | |     --<------<   EQUATIONS
*       ^      |    | |   ^ |
*       ^      |    ^ |   |  ---->----->CALL SUSPENSION
*       ^      |    | |  ------<-----<   EQUATIONS
*       ^      |    | |
*       ^      |    |  ----->---------->CALL TIRE EQUATIONS
*       ^      |      ------<----------<    FOUR TIMES
*       ^      |
*       ^      |
*       ^     STORE INITIAL FUNCTION VALUES IN B(__) AND FFx
*       ^                  |
*       ^                  |
*       ^ ++>+++>START LOOP FOR JACOBIAN
*       ^ +   (MATRIX OF PARTIAL DERIVATIVES)
*       ^ +                |
*       ^ +    INCREMENT UNKNOWNS FOR DERIVATIVE CALCULATION
*       ^ +                |    (ONE AT A TIME)
*       ^ +                |
*       ^ +    CALCULATE THE F(x+h) PART
*       ^ +    OF THE DERIVATIVES BY
*       ^ +   CALLING THE SYSTEM EQUATIONS
*       ^ +    |    ^ |   ^ |    ^ |
^       ^ +    |    | |   | |    |  -------> CALL DRIVETRAIN
*       ^ +    |    | |   | |     --<------<   EQUATIONS
*       ^ +    |    | |   ^ |
*       ^ +    |    ^ |   |  ---->----->CALL SUSPENSION
*       ^ +    |    | |  ------<-----<   EQUATIONS
```

```
    *       ^  +   |    | |
    *       ^  +   |    |  ----->----------->CALL TIRE EQUATIONS
    *       ^  +   |      ------<----------<    FOUR TIMES
    *       ^  +   |
    *       ^  +   WITH THE FF_'S AND F(x+h) CALCULATE THE
    *       ^  +   DERIVATIVES FOR ONE COLUMN OF THE JACOBIAN
    *       ^  +   AND STORE THEM IN A(__)
    *       ^  +                 |
    *       ^  +       DECREMENT THE UNKNOWN
    *       ^  +                 |
    *       ^  ++++++< NEXT JACOBIAN COLUMN
    *       ^                    |
    *       ^                    |
    *       ^                    |
    *       ^    CALL SIMQ TO FIND NEWTON RAPHSON --->--MATRIX===>
    *       ^          CORRECTION FACTORS               SINGULAR  |
    *       ^                    |                                |
    *       ^    UPDATE THE UNKNOWNS WITH CORRECTION FACTORS      |
    *       ^                    |                                |
    *       ^                    |                                |
    *       ^         CHECK FOR CONVERGENCE                       |
    ^       ^           NO           YES                          |
    *       ^            |             |                          |
    *       ^-O--<STEP ITERATION       |                          |
    *         |                        |                          |
    *         |==>TO MANY= = = > SEND OUTPUT TO SCREEN<-----<-----
    *            ITERATIONS      AND OUTPUT FILES
    *                            |
    *                            |
    *   -------<-------<NEXT THETA VALUE
    *   |                        |
    ****|---<------<NEXT ACCELERATION MAGNITUDE
        |                        |
        -<---<NEXT STEADY STATE VELOCITY MAGNITUDE
                                 |
                         CLOSE ALL FILES
                                 |
                               END
```

4.5 Computer Requirements

The computer model was written in Microsoft^{tm} BASIC on an IBM PC-XT computer, then compiled for use with an 8087 math co-processor for additional speed. The model should run on most any home computer with minor changes to the location of allocated files. The question is how fast and how much storage is needed?

The computer program can be used effectively with the following minimum requirements; Microsoft interpreter basic or compatible substitute, 64k to 128k RAM, and a disc drive. Optional, but highly recommended components are; a graphics printer or plotter, plotting software, and the necessary drivers.

The computer used during this project has the following attributes; coming close to the optimum use of the program:

1. IBM PC-XT with a color/graphics monitor, 10.5 mega-byte hard disk, and necessary interfacing hardware;
2. 640k of RAM in which 320k is used as a RAM disk;
3. an 8087 math co-processor;
4. 8087 support software for the basic compiler;
5. an EPSON RX-80 graphics printer;
6. a HEWLETT PACKARD 7470A plotter; and
7. LOTUS-123 spread sheet analysis software for data manipulation and plotting.

The run time of the program depends on the computer hardware, the number of iterations needed to solve the non-linear equations, and the complexity of the tire and suspension routines. The Newton Raphson iteration technique usually converges within five to nine steps; if it converges at all. So, below is run time information based on one iteration for a given condition and a linear tire routine.

1. Running the program under interpreter BASIC each iteration takes 40 seconds.
2. Compiling the basic program and running the executable version increases the iteration step time to four seconds.
3. Compiling with 8087 support libraries further increases the iteration step time to one second.

Necessary storage for the program is small but significant. The program is thoroughly documented which takes up a significant portion of the required storage. A copy of the program listing is given in Appendix A and a sample input file is given in Appendix B. The source code takes approximately 35k to 40k of storage depending on the size of the tire and suspension routines. The compiled version (for use with no BASIC support libraries) takes about 40k to 45k of storage. The total storage necessary can grow very rapidly if comprehensive tire or suspension models are employed.