A Tool for Rapid Vehicle Suspension Design

M. J. Burgess, N. P. Fleming M. Wootton and S. J. Williams
Lotus Engineering, Potash Lane, Hethel, Norfolk England

Reprinted From: Proceedings of the 2004 SAE Motorsports Engineering Conference and Exhibition
ABSTRACT

This paper describes the development of a vehicle suspension analysis tool that is fast enough to be used at the suspension concept design stage. Standard suspension types can be analysed using pre-filled templates that provide easy creation of a kinematic model, in either 2D or 3D modes. The effects of bushes are included and the user can easily modify the rate, position and orientation of any bush. The available pre-built template types include double wishbones, Macpherson struts, trailing arm, semi-trailing arm, pull-rod and push-rod damper actuation, ‘A’ frames and ‘H’ frames. Unique suspension configurations can be constructed for analysis using the user definable template feature. The elasto-kinematic analysis calculates camber angle, castor angle, toe, kingpin angle, roll centre position, damper ratio, spring ratio, track change, wheel-base change, brake steer, lateral force compliance/steer and other quantities used by the designer in optimising a suspension layout.

Results can be displayed either graphically or numerically over specified bump, rebound, roll and steer articulations. The suspension hard points can be picked on screen and simply edited, or dragged on the screen, to review changes to the suspension characteristics – the calculation is virtually instantaneous. The effect of point tolerance on the calculated suspension derivatives can be calculated and displayed through the same interactive environment, by simply selecting the point and setting the tolerance.

INTRODUCTION

As a suspension system moves through its range, the kinematic constraints and compliance of the system define the motion of the wheel. Even suspension systems that are rigidly mounted will have some compliance (possibly structural), and to optimise a system, this must be taken into account.

Analysing the system in purely kinematic terms is simple, since the equations only have to satisfy a set of fixed lengths, as defined by the pin-jointed system. A kinematic program can therefore be written that can operate in ‘real time’, allowing the user to ‘drag’ geometry points and visualise instant graphical results. Lotus has used this type of software, known as SHARK (Suspension Hardpoints And Real-time Kinematics) for many years. The designer is able to see the effects of any geometry change on a wide range of characteristics, and therefore find a good compromise.

The generation of a compliant model is more complex. This is because large displacement compliant models are inherently non-linear. The non-linear build up of forces with suspension displacement are due to:

- GEOMETRY CHANGES
  Variation of geometry with suspension motion changes the way the forces are resolved in each suspension leg.

- STIFFNESS
  Bushes are inherently non-linear, particularly if the bush is small.

- CONTACT
  Changes in load path due to bumpstop contact.

To form one set of equations to represent the complete system becomes complex. Many programs exist that do form such a set of equations (ADAMS, ABAQUS etc.), but the iterations required at each time step make them far too slow for ‘real time’ solutions.

Analysis of suspension systems usually involves:

1. Pure Kinematics
2. Elasto-Kinematics with linear bushes
3. Non-linear bushes and bumpstops
This paper mainly deals with the first two points, where analysis has most influence, but discusses the limitations and inaccuracies that non-linear bushes may generate.

The solution (to generate a program that includes compliance and can work in real time) is to split the problem into two components:

- Kinematic motion, solved using a simple set of equations.
- Subsequent compliance analysis, solved at each position defined by the kinematic motion.

Since only linear bush stiffness (or structural compliance) is being considered, the compliant analysis becomes identical to linear finite element analysis, solved at each suspension position and ‘added on’ to the Kinematic results (superposition). This is valid, since the large motions (non-linear) are solved in an unloaded state – i.e. kinematic solution to re-define the geometry. The subsequent compliance will always be a small perturbation from this geometry and can therefore be considered as linear (no subsequent geometry change).

The solution (at each suspension position), becomes the solution of a very small FE model, which is simply the inverse of a small matrix multiplied by a load array. The load array is calculated at each suspension position defined from the kinematic motion (i.e. compression of the main suspension spring), plus any external force (i.e. brake force). Since this is a direct solve, the time required is very small. Moving geometry points around becomes ‘REAL TIME FE’- and with the non-linearities being calculated separately, the solution is actually ‘REAL TIME non-linear FE’.

This approach allows the programmer to think of the problem as a generic two stage mathematical problem, and solve the problem as two independent sets of equations.

A traditional method would be to consider, for each suspension system, how forces react (with respect to the geometry) and how the suspension systems distort in a link-by-link piecemeal calculation. This is similar to the way it would have been done on a drawing board. This would not be an efficient way and would not allow the program to become flexible (easily adapted to any system). Also, this would only work for simple systems, and if conical rates are also included, each component would become statically indeterminate, and therefore this approach would not work.

This paper outlines the basic principals used to generate sets of equations and relies on ‘off the shelf’ routines to solve these equations. It is not the purpose of this paper to describe how the solvers work.

EXISTING LARGE DISPLACEMENT NON_LINEAR ANALYSIS.

Non-linear solvers (ADAMS, ABAQUS etc.) all use the same basic principals to solve the systems. The load to be applied is divided into fractions of the load (i.e. a small step), and the stiffness matrix $K_a$ is assembled at the starting geometry.

Diagram 1. Typical Iterative Solution of a Non-Linear System – Iteration 1

Using this stiffness matrix, and the applied load for the step ($\Delta P$), a ‘first guess’ at the displacement is generated. This can be seen as $U_a$ in the diagram above. At this new geometry position $U_a$, a new stiffness matrix $K_a$ is formed, and the internal forces $I_a$ are calculated. The force imbalance $R_a$ in the system is then:

$$R_a = P - I_a$$

If $R_a$ is zero (or less than a specified tolerance), then $U_a$ is accepted as the solution for that step, but if $R_a$ is greater than the specified tolerance then the imbalance force $R_a$ is applied to the new stiffness matrix $K_a$, and a new displacement $U_b$ is predicted.

Diagram 2. Typical Iterative Solution of a Non-Linear System – Iteration 2
A new stiffness matrix $K_b$ and a new force imbalance $R_b$ are calculated. If this is less than the tolerance then $U_b$ is taken as the solution for this step. If not a further iteration is performed. When convergence is achieved, the process is then repeated for the next load step.

Summarising, a typical process splits the solution into small load steps and iterates within those steps to find a force balance within the system. A simple example of this is shown below. A rod is pivoted on a bush at end A, and a force is applied to end B. The first iteration in Step 1 moves to 1a (vertically down), but the force imbalance calculated by this new geometry position forces the next iteration to 1b, followed by 1c and finally 1 where a force imbalance is found. Position 1 is then taken as the solution for step 1, and step 2 begins.

### DIAGRAM 3. Typical Iterative Solution of a Non-Linear System – Steps 1, 2 and 3

This is a good approach for a generic non-linear solving strategy i.e. will work for most systems, but is not rapid enough for solving suspension systems in real time. The large geometry changes for a suspension system are generally dictated by the kinematic motion (i.e. ball jointed system), and the compliance component is small and can often be assumed to be linear, as in Diagram 4. Even bushes that may have highly non-linear characteristics will initially be analysed as linear to optimise their initial rates and orientations.

### KINEMATIC ANALYSIS

All kinematic systems can be modeled as a set of equations representing fixed lengths. A double-wishbone, for example, has a set of unknowns for all the outer points.

Diagram 5 shows a schematic of a double-wishbone suspension system. For clarity, all ‘fixed points’ have been represented as capital letters, and all ‘moving points’ are represented as lower case letters. Point ‘i’ is on the stub axle and point ‘j’ is the tyre contact patch.

### DIAGRAM 5. Schematic of a double-wishbone suspension system.

Alternatively, the stub axle can be represented as two points, i and j, as shown in Diagram 5B. The tyre contact patch motion and all kinematic wheel motion characteristics (camber change, toe change etc.) can then be ‘post calculated’ from the motion of the stub axle.
For the system shown in Diagram 5B the variables are: c,f,h,i,j

Each of these has x,y,z as unknowns, so the number of degrees of freedom are:

\[ 5 \times 3 = 15 \quad (c,f,h,i,j \times X,Y,Z) \]

In order to solve the system of unknowns, 15 independent equations must be found.

Since the upright (hub) is rigid, the stub axle vector, i and j can be represented as two ‘wire frame’ three sided pyramids on a base cfh, as shown in Diagram 6. On this base, the first pyramid has a peak at ‘i’, and the second a peak at ‘j’.

This system of ‘lines’ all represent fixed lengths, and therefore form the set of equations.

Fixed lengths are:

- \( Ac, Bc, Df, Ef, hG, cf, ch, hf, ih, ic, if, jh, jc, jf. \)

The lengths of all these are known from the initial ‘starting’ geometry, so the system of equations become:

- \( Ac(x)^2 + Ac(y)^2 + Ac(z)^2 = \text{Constant}1 \) (the length of each line)
- \( Bc(x)^2 + Bc(y)^2 + Bc(z)^2 = \text{Constant}2 \)
- \( Df(x)^2 + Df(y)^2 + Df(z)^2 = \text{Constant}3 \)
- \( Ef \) etc.

This will generate 14 simultaneous equations, however 15 equations are required, and the final equation is the enforced motion of the system (i.e. bump travel). This can be initially set to be the z height of the wheel centre (i), but a correction will have to be used if the enforced displacement is required at the ground (due to the jacking from camber change)

15th equation: \( i(z) = \text{input}(z) \) (i.e. solved at every hub position)

The system of equations are non-linear, and no direct solution is possible – but there are plenty of non-linear equation solvers that can be used directly. SHARK uses Powell’s Hybrid method.

**COMPLIANT ANALYSIS.**

**STIFFNESS MATRIX**

The positions and orientations of all the compliant elements are defined by the kinematic analysis at each suspension position. The stiffness of each bush is represented as a diagonal 6 x 6 matrix (no cross products), since the user will know the principal values in ‘bush’ definition co-ordinates. Using the orientation matrix from the kinematic analysis, the effective stiffness of each bush is determined for the bush orientation at each position. This is a ‘fully populated’ stiffness matrix (i.e. includes cross products).

\[ K \text{ (new)} = DC \times K \text{ (definition)} \times DC^{-1} \]

Where DC is the Direction cosine matrix, defined at each suspension position by the kinematic analysis.

The effective stiffness matrix (for each bush) relative to a reference point is then calculated (again a 6 x 6 matrix). For convenience the contact patch is chosen since this is where the force vector is defined.
The complete system stiffness matrix (at each suspension position) is then assembled from the individual stiffness matrices in the same manner as FE analysis i.e. a square matrix of a size equal to 6 x the number of parts in the system. The rows and columns refer to the 6 degrees of freedom of each part, and the system stiffness matrix is populated to reflect the topology of the system. The resultant is a ‘stack’ of stiffness matrices (defined at the contact patch) that fully defines the system in all the kinematic suspension positions. See Diagram 7.

![Diagram 7. Stack of stiffness matrices to fully define the system.](image)

**FORCE ARRAY**

The external forces acting on the system can be defined as any force acting on any part, since the matrix represents all degrees of freedom of all the parts.

Typically the forces will be external forces acting on the wheel and force derived from the pre load in the spring. The force acting on the wheel (or any part) is a set of three orthogonal forces and three moments, and can therefore represent any force at any position and direction. The forces from the springs are calculated from the displacement of the spring (top and bottom) points obtained from the kinematic analysis, together with the appropriate spring stiffness and directions. In a real suspension system, the actual spring load will be slightly less than the value calculated from the kinematic analysis by an amount from the reduction in spring deflection (due to compliance of any bush). This is allowed for in the model by including the spring as a stiffness element plus a pre-load (treated as an external force). The total spring load is the sum of the two elements.

**DISPLACEMENT OF THE SYSTEM**

The kinematic displacements of all the points are obtained from the simple kinematic model, and the additional compliance components are obtained from:

Compliant Displacement of the system:

\[ S(\text{compliant}) = (K)^{-1} \times (F) \]

where K is the system stiffness matrix and F is the force array.

So the total displacements of all the points is:

\[ S(\text{total}) = S(\text{kinematic}) + S(\text{compliant}) \]

It is also useful that the contribution of the compliance, independent from the kinematics, can be seen directly (without the need to generate two models)

**SUSPENSION CHARACTERISTICS**

All suspension characteristics can be calculated from the motion of the stub axle points. Basic characteristics such as camber, toe and hub motion is simple trigonometry from the results generated for the stub axe, but also anti-dive, anti-squat etc can be derived easily from the hub and contact patch motion without resorting to simplistic geometric methods. Methods using ‘instantaneous’ pivot points do not allow for compliance and therefore have a level of inaccuracy. Using the compliant motion (path) of the stub axle or contact patch gives the correct answer.

**REACTION FORCE RESULTS**

With the displacements of all the parts calculated, the individual bush deflection is the difference in the displacements of the two parts at the connection points. (in the same way as Finite Element analysis calculates internal force). The bush forces can then be found by multiplying this displacement vector with the local stiffness matrix of the bush.

\[ F = K(\text{local}) \times (s2-s1) \]

where \( s1 \) and \( s2 \) are the displacements of each ‘side’ of the bush.

**MODAL RESULTS**

With the stiffness matrix fully defined, at any suspension position, the corresponding modes can be found if the mass matrix is defined.

Forming the mass matrix:

The mass and inertia are initially defined for each part in isolation, and the coordinate system for the component mass matrix is a local (component definition) coordinate system. For each geometry position calculated by the kinematic analysis (including the initial static position), a direction cosine matrix is determined that relates the orientation of the part, with the initial component definition coordinate system. The effective inertia \((3 \times 3)\) of each component at each position is therefore:

\[ I(\text{new}) = DC \times I(\text{definition}) \times DC^{-1} \]
where DC is the Direction cosine matrix

The effective inertia is then re-defined at the reference point (tyre contact patch) by including the effect of each component mass. This is a 6 x 6 matrix in the form:

\[
\begin{bmatrix}
  M_{x} \cdot (y^2 + z^2) & -M_{x} \cdot (y \cdot x) & -M_{x} \cdot (z \cdot x) & 0 & 0 & 0 \\
  -M_{y} \cdot (x \cdot y) & M_{y} \cdot (y^2 + z^2) & -M_{y} \cdot (y \cdot z) & 0 & 0 & 0 \\
  -M_{z} \cdot (x \cdot z) & -M_{z} \cdot (y \cdot z) & M_{z} \cdot (y^2 + z^2) & 0 & 0 & 0 \\
  0 & 0 & 0 & I_{xx} & -I_{xy} & -I_{xz} \\
  0 & 0 & 0 & -I_{yx} & I_{yy} & -I_{yz} \\
  0 & 0 & 0 & -I_{zx} & -I_{zy} & I_{zz}
\end{bmatrix}
\]

where \(x, y, z\) are the coordinates from the tyre contact point to the component centre of gravity.

This assembled into a complete matrix in the same manner as FE analysis i.e. a square matrix of a size equal to 6 x the number of parts in the system. The rows and columns refer to the 6 degrees of freedom of each part, and the matrix is populated to reflect the topology of the system. The modes then become:

\[
\text{Modes} = \text{Sqrt} (\text{Eigenvalues}(M^{-1} \times K))
\]

And therefore an appropriate eigenvalue extraction algorithm can be used.

FORCED RESPONSE

With the stiffness and mass matrices defined, it is a simple task to generate a forced response in the frequency domain. For this a damping matrix is required. The damping matrix is identical in construction to the stiffness matrix, except that damping values are used.

The forced response then becomes:

\[
F = M \cdot a + C \cdot v + K \cdot x
\]

Replacing the differentials with the Laplace operator:

\[
F = M \cdot x \cdot S^2 + C \cdot x \cdot S + K \cdot x
\]

In the frequency domain: \(S = \omega \cdot i\)

Therefore:

\[
F = M \cdot x \cdot \omega^2 \cdot i^2 + C \cdot x \cdot \omega \cdot i + K \cdot x
\]

\[
F = -M \cdot x \cdot \omega^2 + C \cdot x \cdot \omega \cdot i + K \cdot x
\]

\[
F = (-M \cdot \omega^2 + C \cdot \omega \cdot i + K) \cdot x
\]

\[
x = (-M \cdot \omega^2 + C \cdot \omega \cdot i + K)^{-1} \cdot F
\]

so the displacement, at any given frequency, can be found from the matrix multiplication of the matrices above..

COMPARISON TO ADAMS.

The following results are obtained from the analysis of a high performance road vehicle. The suspension is of the double wishbone type, and is mounted on bushes that are highly non-linear.

Comparing Shark With Adams (Non-Linear Bushes), and Adams (Linear Bushes) shows the following results.

For each ADAMS ‘run’, the CPU time for the solve was around 1 second. To use this as a basis of a program to ‘drag’ geometry points around the screen would require 200 seconds (see below). SHARK is instantaneous.

Assuming solving every 0.1 mm (SHARK updates with any changes in geometry that are detectable using a mouse), and a typical drag may be 20 mm in any direction (but is unrestricted). Approximate CPU time required = 20/0.1 sec (200 sec)

Diagram 8 shows a comparison of predicted results obtained using ADAMS with linear bushes characteristics, ADAMS with non-linear bush characteristics, and SHARK suspension analysis. No differences can be seen between any of the traces.
Diagram 9 again shows a comparison of predicted results obtained using ADAMS and SHARK suspension analysis for caster change due to applied braking force. ADAMS with linear bushes and SHARK are almost identical, but non-linear bushes show a small difference at high braking loads. The initial gradients are identical.

Diagram 10 shows a comparison of predicted results obtained for camber change due to lateral acceleration. Again ADAMS with linear bushes and SHARK are almost identical, but non-linear bushes show a small difference at high lateral loads. The initial gradients are identical.

Diagram 11 shows a comparison of predicted results obtained for roll centre height for a variation of vertical wheel travel. It can be seen that all the predictions (ADAMS linear / non-linear, and SHARK) all produce identical results.

The purpose of this work, was to provide a rapid analysis capability that could analyse the kinematic and compliant characteristics of a suspension system and could operate in real time. This would allow the user to ‘drag points around on the screen, and view ‘immediately’ the effect of these changes on the suspension characteristics. The program has been written, see Diagram 12, and the speed of calculation on a typical PC is easily capable of real time analysis with no significant loss of accuracy.
CONTACT

For further information on Lotus Engineering Software please contact:

Lotus Engineering Software,
Potash Lane, Hethel,
Norfolk, NR14 8EZ,
England.

Email: Info@lesoft.co.uk
Web: www.lesoft.co.uk