An Overview of User-Defined Interfaces in LS-DYNA

Tobias Erhart

Dynamore GmbH, Stuttgart, Germany
tobias.erhart@dynamore.de

Abstract:

The user-defined features in LS-DYNA are powerful tools that allow users in academia or industry to verify research results in the context of general and complicated finite element applications. Implementation work concerns only the special field of interest, and there is no need for the comprehensive task of developing and maintaining the complete finite element software. The most popular user interface is for material modeling. But there also exist user interfaces for structural elements, airbag sensors, solution control, friction, interface control, weld failure, loads, output control, adaptivity, thermal contact, and others.

An overview of current user-defined interfaces in LS-DYNA version 971 R5.0 will be presented. The aim of this contribution is to bring together the possibilities to add own numerical models and algorithms to the code. Therefore, each interface is described in its functionality. It will be explained, how to activate the particular interface in the input, where to find the corresponding subroutine, and which input/output arguments can be used.

Keywords:

User-defined interfaces

1 Introduction

A sophisticated multi-physics finite element code consists of a huge amount of different algorithms for the numerical simulation of large-scale real-world problems. In order to provide enough freedom to choose solution methods for the problem at hand, LS-DYNA already offers many options for all imaginable tasks such as element types, materials, contacts, connections, loads, boundary conditions, etc.. But sometimes, the user still wishes to implement her or his own algorithm at a particular point of the solution procedure. Therefore, LS-DYNA also provides user-defined interfaces, i.e. the source code is partly open for customized modifications. Several user interfaces for the implementation of own numerical models or algorithms exist. Before the available user interfaces will be described, a short introduction into the general handling of the special "usermat" version will be given.

First of all, a so-called "usermat package" is needed, which is available via local LS-DYNA distributors. In general, this package is a compressed archive which contains several files such as library files (`*.a`), object files (`*.o`), include files (`*.inc`), Fortran files (`*.f`) and a `Makefile`. Most important files for the user are the Fortran files `dyn21.f` and `dyn21b.f` as well as the `Makefile`. The last one specifies how to derive the target program, in this case the LS-DYNA executable. It also gives information about the specific Fortran compiler that should be used. If this compiler is installed, then the `make` command will start the generation of her or his own executable, called "ls971" (SMP) or "mpp971" (MPP). This program can then be operated in the same way as a standard version of LS-DYNA. If the translation and linking procedure was successful, the next step is to implement own algorithms for materials, elements, friction, or other features into the user subroutines of Fortran files `dyn21.f` and/or `dyn21b.f`. An overview of available user interfaces and associated keywords is given in table form below (Table 1). The listed subroutines represent the highest level of entrance of the particular interface. In the following, the individual user interfaces will be presented in more detail.

2 User-defined materials

The most applied and probably most sophisticated user interface of LS-DYNA serves the implementation of self-made material models, especially standard constitutive models (computation of stresses from strains) for shell and solid elements. However, other material laws such as Equations of State, cohesive zone models, thermal material properties or failure criteria for standard materials can be incorporated. The different possibilities will be presented in the following.

2.1 Structural materials

A key ingredient of every finite element analysis is the material model, where the strains or deformations are connected to the stresses or forces by a constitutive relation. In LS-DYNA, various material models exist ranging from isotropic elastic to anisotropic elasto-viscoplastic with damage and many other specialized constitutive laws. They are able to describe the behavior of materials such as metals, plastics, rubber, foam, concrete, soil, composites, wood, and many others. But since this is a broad subject and also still an active area of research, individual users try to find the best solution for their material at hand. Therefore, the possibility to implement own material models is crucial.

With the help of the keyword `*MAT_USER_DEFINED_MATERIAL_MODELS` one can define the input for the user material interface. If this keyword is used, the main program calls subroutine usrmat in `dyn21.f`, and from there, different subroutines are called depending on the element type in use: `urmathn` for solid elements, `urmats` for shell elements, `urmatb` for beam elements, `urmatd` for discrete elements and `urmatt` for truss beam elements. Those subroutines in turn contain the calls to the actual user subroutines `umat`*XX* , where it is the objective to compute stresses from strains (see Figure 1).

The input arguments of those subroutines are material constants from the input, strain increments, old stresses, old history variables, current time and step size, and others. The output variables are the new (Cauchy) stresses and the updated history values. If a user-defined constitutive subroutine is to be used in implicit analysis, the user has also to define the material tangent modulus. In this case LS-DYNA calls the subroutines `urtans` or `urtanh` from within different material tangent operator subroutines `utan`*XX* are called. Within this subroutine the element stiffness matrix has to be filled with consistent data to achieve a quadratic rate of convergence in the global equilibrium iteration.

User interface	Keyword input	Subroutine(s)	File
Structural Materials	`*MAT_USER_DEFINED_...`	`usrmat, utans`	`dyn21.f`
Equations of State	`*EOS_USER_DEFINED`	`ueoslib`	`dyn21b.f`
Cohesive Materials	`*MAT_USER_DEFINED_...`	`umatXXc`	`dyn21b.f`
Thermal Materials	`*MAT_THERMAL_USER_...`	`thusrmat`	`dyn21b.f`
Material Failure	e.g. `*MAT_024` with `FAIL<0`	`matusr_24/103`	`dyn21.f`
Structural Elements	`*SECTION_SHELL/SOLID` with `ELFORM=101...105`	`usrshl, usrsld`	`dyn21b.f`
Interface Friction	`*USER_INTERFACE_FRICTION`	`usrfrc`	`dyn21.f`
Solution Control	-	`uctrl1`	`dyn21.f`
Interface Control	`*USER_INTERFACE_CONTROL`	`uctrl2`	`dyn21.f`
Airbag Sensors	`*AIRBAG:` parameter `RBID`	`airusr`	`dyn21.f`
Thermal Contact	`*USER_INTERFACE_CONDUC...`	`usrhcon`	`dyn21.f`
Boundary Flux	`*BOUNDARY_FLUX: NHISV>0`	`usrflux`	`dyn21.f`
Adaptivity	`*CONTROL_ADAPTIVE` with `ADPOPT=9`	`useradap`	`dyn21.f`
Loads	`*USER_LOADING`	`loadud`	`dyn21.f`
Weld Failure	`*MAT_100` with `OPT=2, 12, 22`	`uweldfail…`	`dyn21.f`
Joint Forces	`*CONSTRAINED_JOINT_USER…`	`ujntfdrv`	`dyn21.f`

Table 1: User-Defined Features and corresponding keyword input, subroutines and files

Since the whole user material interface is described in Appendix A of the User's Manual [1] in great detail, the most important features are only listed at this point:

- Curve/table arrays for accessing defined load curves/tables in the keyword input file are available.
- Local coordinate system can be used to model anisotropic materials.
- Temperatures are made available if needed.
- Failure (element deletion) can be included.
- Individual strain measures can be used with the help of the deformation gradient.
- The user can choose between scalar or vectorized implementation.
- History variables can be post-processed.
- Element IDs, Node IDs, Integration point numbers, coordinates, and many others are available.
- Unlimited number of history variables can be used starting with the R5 release of Version 971.

Examples for isotropic hypoelastic and hyperelastic (Neo-Hooke) materials can be found in `dyn21.f`.

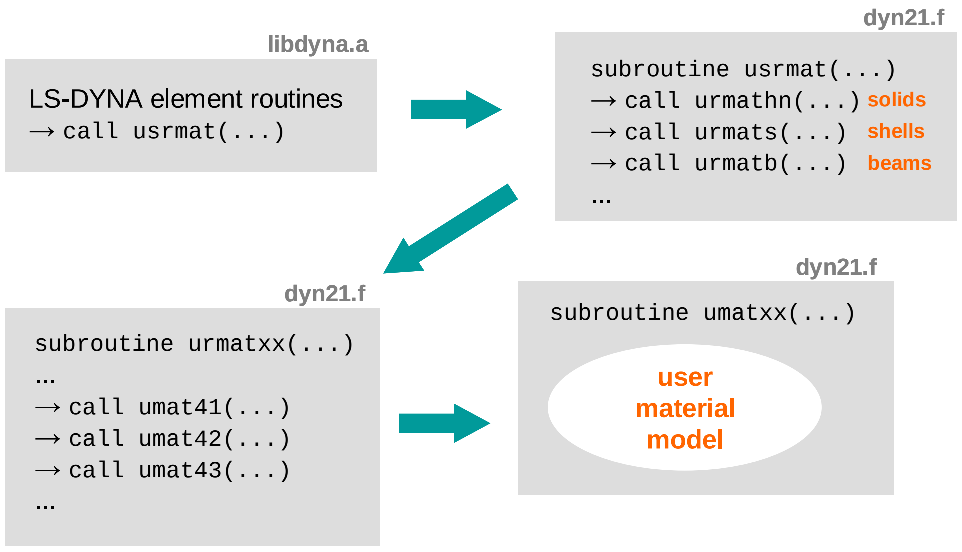

Figure 1: User material interface scheme

2.2 Equations of state

In some situations, an Equation of State (EOS) is required in order to accurately simulate material behavior. An EOS determines the hydrostatic, or bulk, behavior of the material by calculating pressure as a function of density and perhaps, energy and/or temperature. Situations that call for an EOS are characterized by very high strain rates, material pressures far in excess of yield stress, and propagation of shock waves.

To define an own Equation of State in LS-DYNA, keyword `*EOS_USER_DEFINED` has to be used. With that, subroutine `ueoslib` in `dyn21b.f` is called and from there, the actual user subroutines `ueos`*XX*`s` (scalar) or `ueos`*XX*`v` (vectorized) are called. The chosen subroutine is called twice for each integration point in each element. The first call requires the calculation of the bulk modulus, and the second updates the pressure and internal energy. Parameters that are passed to those routines are pressure, reference density, history variables, initial and relative volume, internal energy, and a pressure cut-off value. An example is given in the Fortran file `dyn21b.f`. More informations about that user-defined interface can be found in Appendix B of the User's Manual [1].

2.3 Cohesive materials

Cohesive connections are characterized in that two solid bodies hold together by atomic or molecular forces, which usually takes place within a thin layer. Typical applications are adhesives, weld connections, laminates or composites, interfaces between matrix and inclusions, etc. The mechanical behavior of such cohesive interfaces inside a structure can have a big influence on the overall load bearing behavior, especially when it comes to material separation, i.e. failure, delamination or debonding of those cohesive connections.

For those applications, LS-DYNA provides so-called cohesive elements, which can be placed between solid elements or shell elements. Such a cohesive element formulation includes a material law for the relation between separations and traction stresses. At the moment, about 5 different material laws exist in LS-DYNA to model e.g. elastic-viscoplastic behavior with damage, but the user is also free to implement her or his own model.

Therefore, keyword `*MAT_USER_DEFINED_MATERIAL_MODELS` should be used in combination with cohesive elements (`ELFORM=19` or 20 on `*SECTION_SOLID`). This invokes the call of subroutines `umat`*XX*`c` in `dyn21b.f`. The objective for the user is then to calculate the traction stresses on the mid-surface of the element as a function of the displacements and velocities of the upper and lower surfaces. The arguments that are passed to the subroutines are material constants, displacements, velocities, history variables, time step size, curves, failure flag and the traction stresses. A further description of this user interface is given in Appendix R of the User's Manual [1].

2.4 Thermal materials

With LS-DYNA, structural and thermal analysis can be combined through coupled constitutive models. The user may select thermal elastic and viscoelastic materials for such an analysis. Therefore, the `*MAT_THERMAL_...` cards allow the input of thermal properties such as specific heat capacity, thermal conductivity, and others.

If the standard thermal materials in LS-DYNA are not sufficient to describe the specific material at hand, the `*MAT_THERMAL_USER_DEFINED` keyword could be used to implement her/his own model. This keyword invokes the call of user subroutine `thusrmat` in `dyn21b.f`. From there, user subroutines `thumat`*`XX`* are called. Arguments of these subroutines are material parameters, current temperature, history variables, curve arrray, element number, element type, and others. It is possible to share history variables with mechanical user materials, i.e. quite advanced coupled phase transition models can be implemented. Detailed informations are given in comments in `dyn21b.f` and in Appendix H of the User's manual [1].

2.5 Material failure for standard materials

For the commonly used LS-DYNA (metal) materials 24, 103, 114, 123, 124, 155, and 195 the user has the possibility to use his own failure criteria. This option is activated by setting `FAIL`<0 on the corresponding `*MAT_...` card. This invokes the call of subroutines `matusr_24` or `matusr_103` in `dyn21.f`. Arguments of these subroutines are stress and strain rate components, time step size, plastic strain, material constants and curves. With these informations, the user can construct her or his own failure criteria. If this criteria is met, a flag can be set to trigger failure of an integration point and element deletion.

3 User-defined elements

Element technology is still an active area of research, where the following topics are important: robustness, speed, (coarse mesh) accuracy, prevention of locking, prevention of hourglassing, etc. Exemplary deformations of an elasto-plastic block for different element formulations are shown in Figure 2. The idea with the interface in LS-DYNA is to raise the level of abstraction to a point where the user has to provide a minimum of input and still be able to define useful elements. Therefore the interface is split into one for continuum elements with numerical integration and one for resultant elements. In this way, the integrated elements take advantage of the extensive LS-DYNA material library and the user does not have to write a separate material routine for the element in question - unless he wants to.

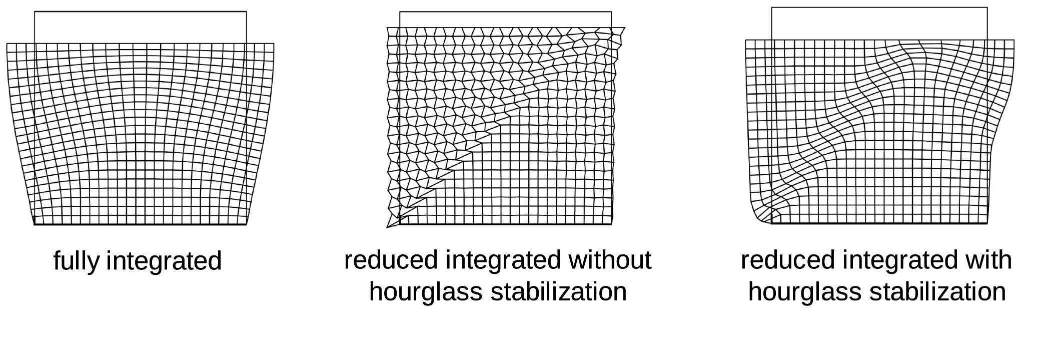
fully integrated
reduced integrated without hourglass stabilization
reduced integrated with hourglass stabilization

Figure 2: Deformations for different element formulations

User-defined elements in LS-DYNA can be implemented for hexahedral solids and quadrilateral shells. Up to a total of ten elements can be implemented in a single executable for both explicit and implicit analyses. The feature is invoked on the *SECTION_SOLID or *SECTION_SHELL card, where the element formulation ELFORM should be specified to a number between 101 and 105. The corresponding user subroutine `usrshl` (shells) or `usrsld` (solids) is called for computing the internal force and tangent stiffness. As already mentioned, a distinction is made between integrated elements and resultant elements.

For the integrated elements the user only needs to implement the gradient displacement matrix and Jacobian, leaving LS-DYNA to take care of stress updates and force and stiffness assembly. This allows the user to take advantage of the rich supply of standard materials in LS-DYNA and also relieves him/her from the tedious task of assembling the global finite element matereices.

On the other hand, the global force vector and stiffness matrix must be implemented in a single routine for the resultant elements. While this puts little or no limits in terms what type of elements can be defined, extra amount of work may be needed to define the element in question.

The arguments that are passed to the user subroutines are property parameters, history variables, nodal coordinates, isoparametric coordinates, shape functions and derivatives, etc. Another interesting feature is the optional specification of extra degrees of freedom per node. These extra degrees of freedom could for instance represent shell thickness, hydrostatic pressure, or additional degrees of freedom from a multi-scale formulation.

A very good desription of the user-defined element interface is given by Borrvall [2].

4 User-defined friction

The friction between two contact partners is based on a Coulomb formulation in LS-DYNA. Its behavior can be controlled with the parameters on the `*CONTACT_...` card 2: static and dynamic friction coefficients, exponential decay coefficient, and coefficient for viscous friction. The underlying frictional algorithm updates the interface force to a trial value first, then computes the tangential part, computes the coefficient of friction and the yield force, and finally determines the frictional force. This is equivalent to an elastic-plastic spring model.

Now, this standard algorithm can be modified by the user via the keyword `*USER_INTERFACE_FRICTION` which invokes the call of subroutine `usrfrc` in `dyn21.f`. In fact, two separate subroutines `usrfrc` exist in the Fortran file for MPP and SMP for reasons that have to do with how the contacts are implemented in general.

In the SMP version of LS-DYNA, the user is required not only to define the frictional coefficients but also to assemble and store contact forces and history. However, this gives the opportunity to make quite general changes to the frictional algorithm. The standard algorithm is already given in subroutine `usrfrc` as a starting point. In MPP-DYNA, only the frictional coefficients have to be defined.

However, the main idea of this interface is the user-defined modification of friction coefficients, e.g. static and dynamic friction coefficients as functions of temperature, contact pressure, sliding velocity, and other input values such as penalty stiffness, sliding displacement, own history variables, averaged plastic strain from adjacent elements etc. Also, the curve array for accessing defined load curves in the keyword input file is available. An example for pressure and velocity dependent rubber friction is given in Figure 3.

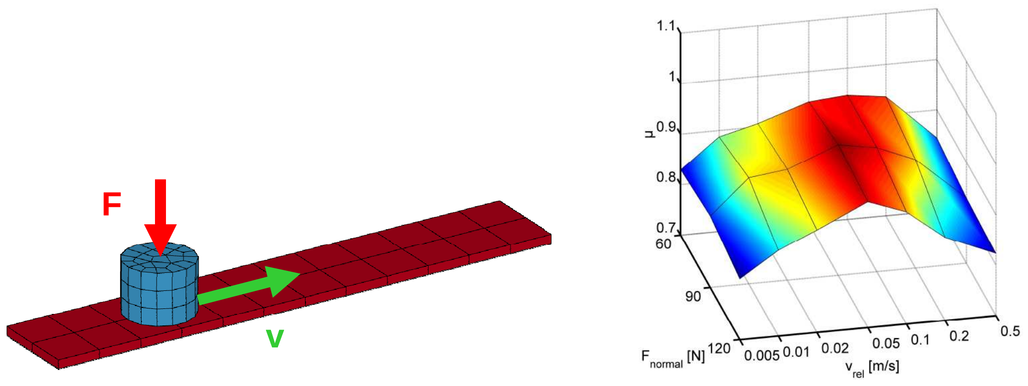

Figure 3: Rubber friction coefficient as function of pressure and velocity [3]

At the moment, the user friction interface is not supported by all available contacts in LS-DYNA, but should cover the most interesting ones. Further informations about this user interface can be extracted from the comments in `dyn21.f` and can be found in Appendix G of the User's Manual [1].

Remark: If only the dependence of the friction coefficient from relative velocity and pressure is needed, the option `FS=2` of the `*CONTACT_...` keyword should be used, where a corresponding table can be defined in the input.

5 Other user-defined features

Besides the three main topics - materials, elements and contact friction - other user-defined interfaces exist, which will be presented in this chapter.

5.1 Solution control

This option may be provided by the user to control the I/O, monitor the energies and other solution norms of interest, and to shut down the problem whenever he pleases. The corresponding subroutine `uctrl1` in `dyn21.f` is called once each time step and does not need any control card to operate. Arguments of this subroutine are nodal displacements, velocities, accelerations, nodal masses, energies, time step size, cycle number, ASCII file units, and others.

5.2 Interface control

This feature may be provided by the user to turn contact interfaces on and off. It is activated by the keyword `*USER_INTERFACE_CONTROL`, which invokes the call of subroutine `uctrl2` in `dyn21.f`. Input arguments of this subroutine are interface number, interface type, current solution time, cycle number, slave nodes, master nodes, thickness, velocity, time step size, etc. Based on these quantities, a criterion can be defined to decide if the current contact is active or inactive. An example for the user-defined interface is given in `dyn21.f`. A description of that feature is given in Appendix F of the User's Manual [1].

5.3 Airbag sensor

To trigger the deployment of an airbag in LS-DYNA, the keyword `*SENSOR_...` can be used. Different sensor types and locations, switch criterions and math computations on sensor results can be defined to govern the airbag ignition (see Figure 4).

As an alternative, a user sensor interface can be used. The sensor is mounted on a rigid body which is attached to the structure. The motion of the sensor is provided in the local coordinate system defined for the rigid body in the definition of `*MAT_RIGID`. When the user-defined criterion is met for the deployment of the airbag, a flag is set and the deployment begins. All load curves relating to the mass flow rate versus time are then shifted by the initiation time. This option is activated by parameter `RBID` on the `*AIRBAG` keyword and the corresponding subroutine is `airusr` in `dyn21.f`. Parameters that are passed to the subroutine are rigid body displacements, velocities, accelerations, input parameters, history variables, time, step size, airbag ID, etc. An example for airbag activation based on a resultant acceleration is given in `dyn21.f`.

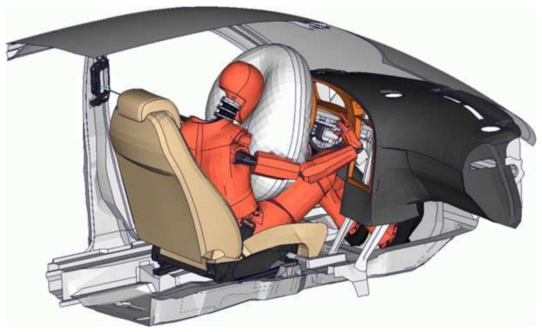

Figure 4: Airbag deployment (Courtesy of Saab Automobile AB)

5.4 Thermal contact

When two solid bodies come into contact, heat flows from the hotter body to the colder body (see Figure 5). The coefficient describing that heat conduction is called the thermal contact conductance. In LS-DYNA, it is possible to define thermal contact conductance as a function of temperature and pressure using the keyword `*CONTACT_..._THERMAL_FRICTION_...`. Different formulas for that behavior can be chosen by setting parameter `FORMULA`.

If an own relationship between heat conductance and pressure, temperature, velocity, etc. ought to be implemented, the keyword `*USER_INTERFACE_CONDUCTIVITY` should be used. The corresponding subroutine in `dyn21.f` is named `usrhcon`. Input arguments of this subroutine are user parameters, interface pressure, temperature, displacements, velocities, distance between contact interfaces, and the curve array for accessing defined load curves in the keyword input file. The output of this routine is the thermal contact conductance. More informations about the general topic of hot stamping with LS-DYNA can be found in [4].

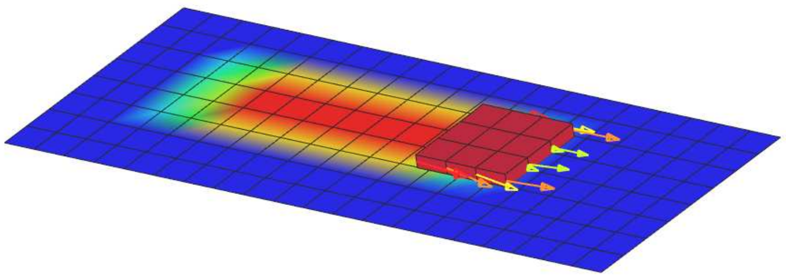

Figure 5: Thermal Contact

5.5 Boundary flux

Surface heat flux is a fundamental thermal boundary condition along with convection and temperature. In a thermal or coupled thermal/structural analysis with LS-DYNA, it is possible to define corresponding flux boundary conditions via `*BOUNDARY_FLUX_`*`OPTION`*.

If the number of history variables is chosen greater than zero (`NHISV>0`), then the user-defined subroutine `usrflux` in `dyn21.f` is called. With that interface, it is possible to define the boundary flux in or out of a surface segment as an arbitrary function of temperature and history. Parameters that are passed to the subroutine include the heat (energy) per time and per surface area, the segment nodal temperatures at previous and current time, the segment nodal coordinates and the time integration parameter. Also, the current thermal simulation time, the time step, and average segment temperature is provided together with the curve array. An example for user-defined boundary flux is given in `dyn21.f`. More details about this user-defined interface can be found in Appendix S of the User's Manual [1].

Remark: The `*DEFINE_FUNCTION` keyword can be used instead of thermal user subroutines [5]. This keyword allows in-line functions to be defined in the input file. Such as for thermal contact one can write h = function (time, temperature_slave_seg, temperature_master_seg, temperature_average, pressure, gap). For flux this is flux = function (x, y, z, Vx, Vy, Vz, time, temperature).

5.6 Mesh refinement for 3D-shell adaptivity

In a finite element analysis, it is sometimes helpful or even necessary to refine the mesh locally or to remesh a whole part. The reason could be a better resolution of high gradients, the minimization of discretization errors or a better approximation of geometrical details as it is the case for sheet metal curvatures in forming simulations (see Figure 6). For those applications the keyword `*CONTROL_ADAPTIVE` is used to govern the refinement procedure. The adaptive options of that feature include angle change criterions, shell thinning, or a stress-based error norm.

If the user wants to trigger adaptive refinement for shells with her or his own criteria (e.g. an error indicator), this can be done by setting `ADPOPT=9`. The associated subroutine for this option is `useradap` in `dyn21.f`. Arguments of this subroutine are shell element connectivities, stresses and plastic strain at each integration point. The user's own refinement indicator is then used to decide which elements should be refined.

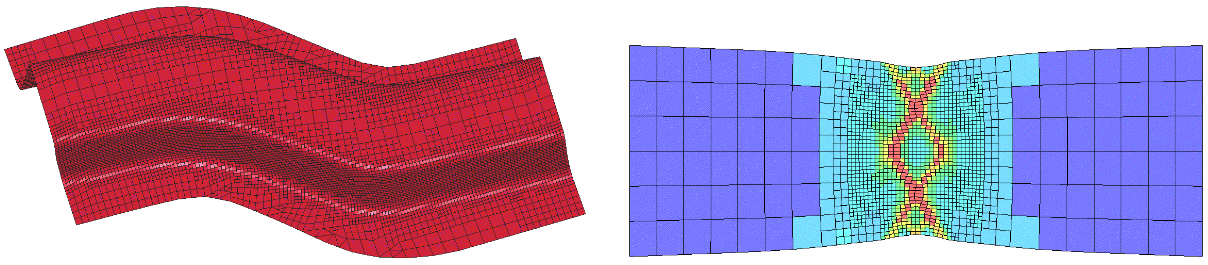

Figure 6: Examples for adaptive refinement - metal forming, localization.

5.7 Loads

This feature may be provided to apply pressure and force boundary conditions in a user-defined way. The keyword `*USER_LOADING` activates this option. The corresponding subroutine is `loadud` in `dyn21.f`. There, the user has access to several variables like nodal displacements, velocities and accelerations, element connectivities as well as to his own parameters. Pressure can then be defined as a function of these variables. More informations about that topic can be found in a paper from the 4th European LS-DYNA Users Conference [6].

5.8 Weld failure and damage

The standard modeling technique for spotwelds is a combination of tied contacts and deformable structural elements such as beams or one or more solid elements (see Figure 7). For the elasto-plastic material behavior of those spotwelds `*MAT_SPOTWELD` is often used, where different failure and damage options exist.

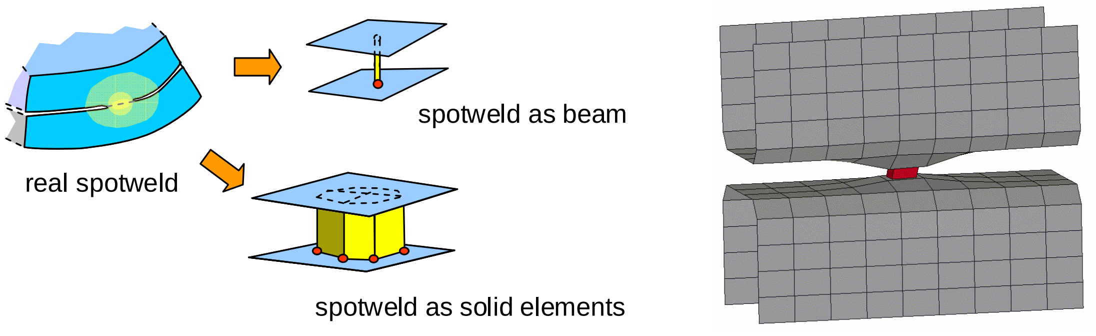

Figure 7: Spotweld modeling techniques

With `OPT=2`, `12`, or `22` user-written subroutines `uweldfail`, `uweldfail12`, or `uweldfail22` in `dyn21.f` are invoked. The appropriate routine is called every time step and a failure flag can be set based on a criterion arranged by oneself. The main differences between those three user options is that `uweldfail` handles sudden failure with up to 6 user parameters, `uweldfail22` deals sudden failure with up to 22 user parameters, and `uweldfail12` provides the opportunity to define a damage model with failure and up to 22 user parameters. These different options will be described in the next two sections. Currently, user-defined weld failure is available for beam welds and hex assembly welds. More informations about that topic can be found in Appendix Q of the User's Manual [1] or comments in `dyn21.f`.

5.8.1 *Failure only:* `OPT=2,22`

Arguments of subroutine `uweldfail` (`OPT=2`) are axial and shear forces, bending moments, torsional resultant, weld diameter, 6 user parameters, and the current simulation time. With these input values, the user can define her/his own failure criterion and he can set the failure flag to 1 ("failed") if the criterion is fulfilled. In this case, the corresponding element gets deleted immediately afterwards. A demonstration failure model is given in `dyn21.f`. Option `OPT=22` (subroutine `uweldfail22`) is identical with the just described one with the minor difference that 22 user parameters can be defined and used instead of only 6. Therefore, two more lines are read on the `*MAT_SPOTWELD` keyword.

5.8.2 *Damage and failure:* `OPT=12`

The newest option provides the opportunity to incorporate not only a failure criterion but also a damage model to describe the post-peak or softening behavior of the spotweld. In this subroutine `uweldfail12`, the user has access to several input variables such as forces, moments, diameter of weld, effective plastic strain, effective strain rate, number of intergration points in the weld, current simulation time and cycle number. Input and output values are current damage, weld history variables, and others. The damage takes values between 0 (undamaged) and 1 (totally damaged) and is used to derive the true stresses from the nominal stresses.

5.9 Joint forces

Joint connections between rigid bodies are controlled by the keyword `*CONSTRAINED_JOINT_....` There, the geometric location of the joint is defined. Afterwards, the resulting forces acting on the rigid bodies are obtained by the penalty method depending on the joint motion. The associated penalty stiffness can be defined with that keyword, too. For the definition of optional rotational and translational joint stiffnesses the keyword `*CONSTRAINED_JOINT_STIFFNESS_...` can be used.

As an alternative, keyword `*CONSTRAINED_JOINT_USER_FORCE` can be used which invokes the call of subroutine `ujntfdrv` in `dyn21.f`. With that interface, force resultants can be generated as a function of time and joint motion. At the moment, only the revolute joint type is fully supported. An example for a spring and damper system for a revolute joint can be found in `dyn21.f`.

6 Conclusions

An overview of all user-defined interfaces, that are currently available in LS-DYNA, was given. After a general description of the handling of usermat packages, the individual interfaces were presented in a way, that the user might get an idea, if one of them is useful for her or his needs. Each of them is explained in more detail in the Appendices of the manual, in other papers or in code comments. The main idea of this contribution was to merge the scattered informations about user-defined interfaces in one study. It is unquestionable that this overview is only a snapshot, since the possibilities for further customized interfaces will grow in the future, where new ideas from users are very welcome.

7 References

[1] Hallquist, J.O.: "LS-DYNA Keyword User's Manual".
[2] Borrvall, T.: "A User-Defined Element Interface in LS-DYNA v971", Proc. 9th International LS-DYNA Users Conference, Dearborn, MI, USA, 2006, 18-11ff.
[3] Lindner, M., Kröger, M., Popp, K., Blume, H.: "Experimental and Analytical Investigation of Rubber Friction", Proc. of the ICTAM 2004, Warsaw, August 15-21, 2004.
[4] Shapiro, A.B., "Using LS-DYNA for Hot Stamping", Proc. 7th European LS-DYNA Users Conference, Salzburg, Austria, 2009.
[5] Shapiro, A.B., "Using LS-DYNA for Hot Forming", Proc. 6th German LS-DYNA Forum, Frankenthal, Germany, 2007, C-II-11ff.
[6] Adoum, M., Lapoujade, V.: "Examples' manual for *USER_LOADING option", Proc. 4th European LS-DYNA Users Conference, Ulm, Germany, 2003, G-I-29 ff.