FEA Information Inc.
Global News & Technical Information

Dedicated to the Global Engineering Community

LEAP	ALTAIR-ITALY	DYNAMAX	ANSYS-CHINA
DYNALIS	GISSETA	DYNA*more*	FLOTREND
KOSTECH	ERAB	THEME	MFAC
CAD-FEM	Prof Genarro Monacelli	Dr. David Benson	Dr. Alexey I. Borovkov
Dr. Ted Belytschko	Dr. Taylan Altan	Dr. Bhavin V. Mehta	Prof. Ala Tabiei

FEA Information Inc.

Global News & Industry Information

Volume 2 **Issue 4-2002** **April**

Purpose:

The purpose of our publication is to provide technical and industry information

In This Issue:

7th International LS-DYNA Users Conference
FEA Information Inc. Participants
Sponsors and/or Exhibitors
Bringing technical excellence to the engineering community

AMD	**ANSYS**	**CEI**	**ETA**
EASi	**HP**	**JRI**	**MSC.Software**
OASYS	**SGI**		

High Performance Compute Servers (HPC)
Part 1 of 2

Contributors:
Arthur Shapiro – FEA Information
Stan Posey - SGI

The future in automotive, aerospace, electronics, medical, and all industries is to obtain scalability. The predecessor of the HPC servers were the large central computers that eventually turned to distributed computers. Distributed computers needed more power and linking them gave way to today's High Performance Computing. Today's High Performance Computing is distributed and links large amounts of data from various locations in a seamless environment.

Parallel techniques and programming models evolved based mainly on their 'ease' of use and potential for commercial success, both of which are closely related (e.g. shortened time-to-market for MCAE software).

Evolution: There existed roughly 4 choices over time:

1. SMP Fine Grain

Classic Vector choice since H/W offered very few, very fast CPUs, scalability to 4 CPUs typically. Parallelism at the DO-LOOP (matrix/data) level and with compiler options. OpenMP has improved this somewhat today to about 16 CPUs, but not 100's to 1000's that is capable today.

2. SMP Coarse Grain

Evolved with SMP programming model as way to higher levels of parallel within an application. Rather than data level, sought parallelism at the geometry, or domain level. There were not many tools to assist with this effort.

3. DMP (Distributed Memory Parallel)

Began with MPP architectures and cluster of networked W/Ss, and software tools such as PVM (parallel virtual machine) but did not achieve success commercially until current RISC architectures and MPI. Another breakthrough for commercial MCAE software were robust graph partitioning schemes such as METIS that could effectively partition the geometry domain to (1) eliminate the need for user inter-action, (2) evenly distribute computational 'work' between each partition of the domain, and (3) minimize boundary communication, across which messages must be passed (again more on this below on H/W). This is the most popular choice today for MCAE applications.

4. Hybrid SMP/DMP

This is what's now emerging, the potential to use say OpenMP within an SMP 'node' whereby a node is typically as large as 16 CPUs and MPI between the SMP nodes. This method seeks to leverage the programming and performance advantages of both SMP and DMP.

Common terms:

High Performance Computing (HPC), sometimes called High Performance and Technical Computing (HPTC), commonly refers to numerically-intensive computations in scientific and technical fields. These computations are usually performed on servers, supercomputers, or clusters – often, hundreds, or thousands of CPU’s (nodes"), and a job can run for hours, days, and even weeks.

Typical HPC applications include in addition to crash and occupant safety simulations: product structural integrity simulations; computational fluid dynamics (CFD); noise, vibration, and harshness (NVH); aero acoustics; computational biology or bioinformatics (gene sequencing, protein folding, etc.); astrophysics (planetary motion, asteroids, space exploration); computational chemistry (molecular modeling, drug design); weather forecasting; geophysics (plate tectonics, seismic prediction); nuclear weapons design; classified defense/intelligence; oil and gas exploration; high energy physics (e.g., CERN, Argonne, Brookhaven); and computational finance (economic modeling, stock market analysis, risk assessment).

In addition to the number-crunching compute servers, other important elements in a typical HPC environment include: accessibility (perhaps using mobile devices); portals; storage devices; high-performance file systems; networking and high-speed data transfer; interconnects; grid computing (campus grids, cluster grids, and global grids) and load management middleware (e.g., LSF, PBS, Sun Grid Engine, etc.); RAS (reliability, accessibility, and serviceability); and increasingly important, security.

FEA Information Inc. Participant’s bringing Technology in High Performance Computing Servers to the engineering community:

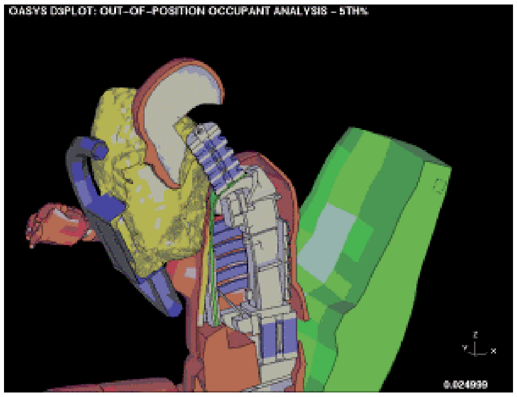
- **Fujitsu**
- **Hewlett-Packard**
- **SGI**

SGI Origin 3000	Scales to 512 processors in a single system image that offers high performance, flexibility and growth for large data-center HPC requirements that support a mix of MCAE applications.
SGI Origin 300	Scales to 32 processors in a single system image and clusters to 1000's of processors with several interconnect options. Ideal solution for departmental mix of MCAE applications, or for a single MCAE application that can scale on large clusters.

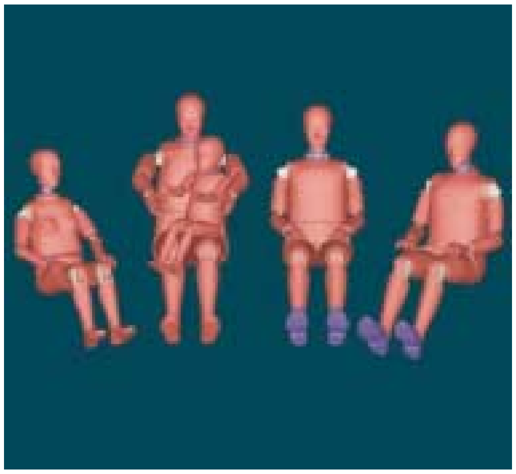
Part II in May edition

LS-DYNA on High Performance Compute Servers

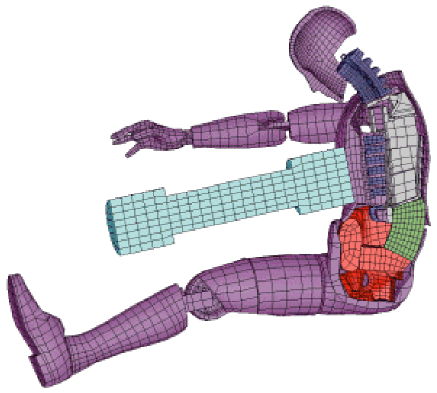
Occupant Protection © Copyright OASYS
Reprinted from the website www.arup.com/dyna



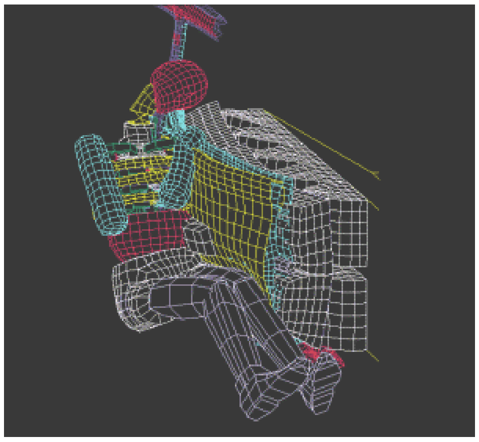
Providing crash protection for the occupants is an integral part of vehicle development. Increasingly sophisticated legislative crash tests, with occupant injury measurements as the pass/fail criterion, demand that detailed modeling be undertaken at an early stage in the design to avoid costly late changes. Arup offers a comprehensive range of modeling skills to predict and optimize occupant protection performance.



The FT-Arup detailed finite element models of the frontal impact crash dummies have been validated against more than 600 laboratory tests. These models offer an unprecedented level of detail and accuracy. The range includes average male (50th%ile) small female (5th%ile), large male (95th%ile) and child dummies.



The small female and child dummy models can be used to evaluate the aggressivity of airbags; small occupants positioned close to the deploying airbag ("out - of - position") can be severely injured.



The new European offset frontal crash test (ECE R94) includes a lower leg injury criterion, which is proving difficult to pass in practice for many vehicles. The sophisticated dummy models are able to predict these injuries and can be used to guide design development, e.g. to assess the effect of reducing toe-board intrusion or adding padding on the floor.

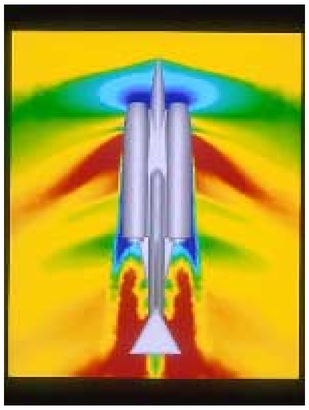
The FT-Arup Free Motion Headform model is available for evaluating vehicle interiors against FMVSS 201 (interior head impact), and to optimzse trim to pass the test.

Previewing the Future At the Speed of Sound
Reprinted with permission from CEI

Computer analysis and visualization show the way as the Thrust SSC team breaks through the sound barrier

by Bob Cramblitt

In mid-October at Nevada's Black Rock Desert, the Thrust SSC became the first land vehicle to break the sound barrier, reaching a peak speed of 763 miles per hour. Although it was a monumental feat, the Thrust SSC team was hardly surprised. The supersonic vehicle's ultimate victory was played out on computer and television screens more than a year ago. Computer-simulated test runs helped verify the vehicle's original design, and paved the way for the Thrust SSC team to accomplish what many thought impossible



A visualization from above the Thrust SSC of air flow at vehicle speed of Mach 1.15 or approximately 874 miles per hour; this is 111 miles faster than the Thrust SSC's actual record speed of 763 miles per hour. Shock waves are visible at the nose of the vehicle and across the length of its body

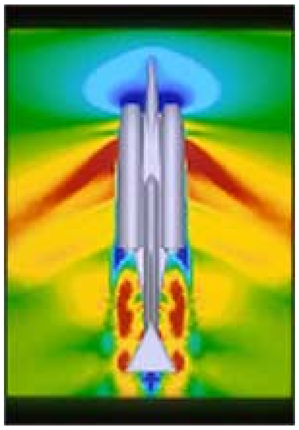
To realize the impact of computer analysis and visualization on the Thrust SSC project, one needs only to flash back less than half a decade. In 1992, Richard Noble, who in 1985 set the world land speed record of 633.5 miles per hour, had an ambition to not only break his own record, but to break the sound barrier as well. He asked Ron Ayers, a retired aerodynamics expert with experience in high-speed vehicles, to give advice on the best shape for a supersonic car

A Leaf in a Gale?

"My immediate reaction was to distance myself from the project," said Ayers. "To drive at supersonic speeds would clearly be extremely dangerous, and indeed, it could well be impossible. The aerodynamic forces would be simply enormous, quite enough to lift the car and throw it around like an autumn leaf in a gale."

According to Ayers, the crux of the problem was how the flow would behave underneath the car at supersonic speeds - and what would happen to the shock waves in that region. Nobody could give Ayers this critical information. One authority on the subject gave the dire warning that the vehicle was "just an expensive way of killing someone."

Fortunately for Noble and his team, Ayers' curiosity got the better of him. He began playing with design ideas, focusing on stability and power. The resulting design was a thin needle-like body framed by two huge Rolls-Royce Spey jet engines. It looked, said Ayers, "like a twin-jet fighter with the wings removed." Despite the radical design, Ayers felt he had a feasible solution. But, there was no way, he thought, to test his design concept.



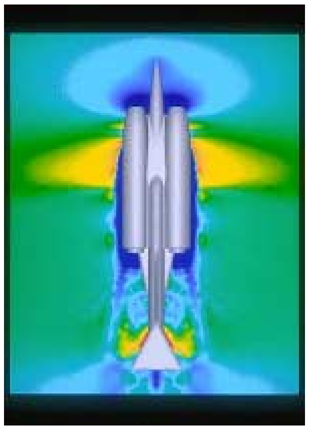
Air flow at vehicle speed of Mach 1.05 or approximately 798 miles per hour; this is 35 miles faster than the Thrust SSC's actual record speed of 763 miles per hour.

From Paper to Computer

The breakthrough came when Ayers found out about the computational fluid dynamics (CFD) analyses being conducted at the University of Wales Swansea. Over the past seven years, the university's department of engineering has been doing aerospace and other engineering analyses using FLITE3D, a program the department developed jointly with Computational Dynamics Research. To display the data in understandable terms for further analysis and presentation, the university loads output from FLITE3D into Computational Engineering International's EnSight, an engineering postprocessing software used by automotive and aerospace firms worldwide

After meeting with engineers at the university, Ayers began providing parts data for sections of the car, working from front to back. The data points were converted into coordinates and a computer model was generated. Eventually, engineers had an entire computer model of the car with which to work. The model was fed into FLITE3D, which allowed engineers to study airflow at various speeds and using slightly different body configurations. The FLITE3D results were loaded into EnSight, where engineers could generate spatial cuts down the car body to show surface pressure at different locations, and to see shock waves forming and disappearing as the virtual car accelerated from a standstill to Mach 1.1, beyond the sound barrier.

To provide the computing power needed to process efficiently the huge amount of analysis and visualization data, University of Wales Swansea engineers relied on a valuable project sponsor, Cray Research. Cray donated several hundred hours of computing time that allowed engineers to do extensive simulations and visualizations of the vehicle's aerodynamics.

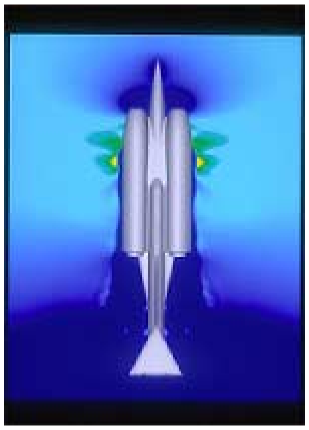

Air flow at vehicle speed of Mach .95 or approximately 722 miles per hour. Airspeed has exceeded supersonic speeds around the wheels and tail of the vehicle.

"They [the Thrust SSC engineering team] were interested in results that showed the pressure the vehicle was undergoing at different speeds," said Kevin Fox, a computer-aided engineering consultant for Cray. "They needed to know what the down forces would be when shock waves hit the vehicle."

Validating the Design

The results of the simulations were analyzed and used by Ayers to understand the effects of exceeding Mach 1 and to refine the vehicle for greater stability and safety. To evaluate and verify the simulation results, a model of the Thrust SSC was built and rocketed down a track at speeds reaching 850 mph. The results of the physical tests matched the data gathered from the computer simulations, thereby validating Ayers' original design concepts

To help present a clearer picture of the results, University of Wales Swansea engineers and Kevin Fox generated animated visualizations within EnSight to display on computer screens and to output onto videotape. "It was a large amount of data, about one-million cels, and it was transient, since we had to display data changes at different speeds," said Fox. "But, EnSight had no problem handling it."


Air flow at vehicle speed of Mach .85 or approximately 646 miles per hour. Airspeed is already near supersonic speeds around the vehicle's wheels

The computer animations clearly showed the forces at work on the vehicle and how the final design would be able to triumph over them. They were a strong tool for understanding the complex issues associated with traveling on land at supersonic speeds - so strong that the animated videos were played on national television in England to educate viewers about the challenge awaiting the Thrust SSC team

"The EnSight animations were good for showing people what was going to happen," said Michael Marchant, one of the project's engineers at the University of Wales Swansea. "We were able to generate images that the general public could understand." There was also a secondary benefit to the university. According to Marchant, the national exposure created greater awareness of the university's renowned engineering program.

Making it Real

The computer analysis and visualization process did not ensure the Thrust SSC's success, of course. It took talented and dedicated engineers, the courage of driver Andy Green, and continuous adjustments during trial runs in the deserts of Jordan and Black Rock to turn concepts into reality. The analyses and visualizations set the stage, however, by showing that a supersonic car was not only possible, but probable.

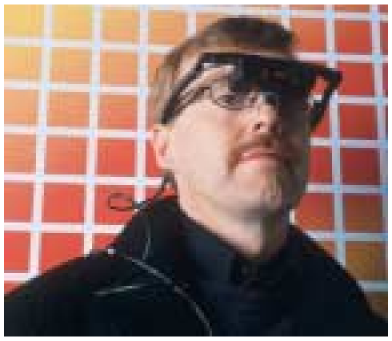
Clinical psychologists say that extraordinary things can be achieved when a person visualizes success before it happens. In the case of Thrust SSC, computer analysis and visualization weren't so much psychological tools as realistic tests that showed what could be done and how it could be achieved. With strong analysis tools and the ability to visualize all types of results for all types of data sets, the future becomes a much more predictable place.

***Bob Cramblitt* is a freelance writer based in Cary, N.C., who specializes in computer graphics, CAD/CAM/CAE and client/server, open systems topics.**

References: Thrust SSC Web Site: http://thrustssc.digital.co.uk - Fox, Kevin, "Supersonic Car Prepares for Race Against Sound Barrier," Cray Channels, Vol. 18, No. 1, 1996, pp. 8-10.

CYBORG CLOTHING

From hardwired jackets to smart sneakers, wearable computers allow you to lace, button and zip up your computer for the ultimate in mobile computing. An interview with hp labs' Phil Stenton.



As developers push at computing components, striving to make them smaller, faster, smarter, more aware, more powerful, more durable and even washable, the borders between the technology and those that use the technology are shifting. Science fiction has long heralded the cyborg -- an individual hardwired with computing components -- part machine, part human. A step in this direction, cyborg clothing -- hardwired jackets, jeans, glasses, sneakers and more -- is rapidly moving out of research labs and preparing for mainstream consumption.

The field of "wearable computing" is being driven by researchers like Phil Stenton from HP Labs Bristol. Having been involved in the prototyping of a number of wearable devices, Stenton offers his views on the future of connected and application-rich devices, opening up the rich and diverse terrain of wearable computers -- a field interwoven with ongoing research in interaction technologies.

What kinds of wearables have you been involved in prototyping?

Our first wearable was the BlazerJet we built with Bristol University. We built simple location-sensitive applications [that used] speech input and the display of a Jornada connected to the BlazerJet for output. It was state of the art at the time but cumbersome by today's technology standards. It did, however, enable us to get out into the streets of Bristol and explore the realities of wearable design in much the same way the pioneers at MIT's media lab were doing in Boston.

Have you spent extensive time wearing any of the prototypes on a day-to-day basis?

Most of the folks I know who wear prototypes almost 24x7 are university researchers. The closest I have come is using head-mounted displays for two weeks as my desktop display. The ergonomics were interesting, but the resolution was awful. Recent attempts to use the Olympus EyeTrek had the same result -- good for DVDs, poor for Windows desktop and Internet browsing.

Wearables like the CyberJacket or BlazerJet make the hardware fairly inconspicuous -- not at all the traditional "cyborg" look. Is this kind of transparency key to widespread acceptance?

Appearance is a big issue for wearables, and the weak answer to this question is "personal choice." We tried to make the BlazerJet unobtrusive because we didn't have the technology to make the circuitry appealing as a fashion statement, and we wanted the wearing of the jacket to be minimally disruptive.

The COMDEX catwalk might display more conservative designs than the fashion houses of London, Paris and New York, but form and function considerations may well have some overlap. COMDEX designs might appear sober, unobtrusive and aligned with services that promote effective business and personal organization in contrast to the fashion industry's wild flamboyant use of services providing intense and flirtatious media experiences.

For wearable computers to be effective seems to entail more than just sewing in, or strapping on, hardware and demands a shift in how we approach both computing and components. How does "wearable computing" force both developers and users to rethink computing expectations and assumptions?

These are important boundaries for mobile computing in general, whether it is notebooks, PDAs, phones or entertainment devices. The importance of interaction technology and the whole area of interaction design are amplified by the explosion in the number of contexts of use and the challenges of a mobile human frame. In addition to these research frontiers, innovations in material science, wireless networking and context sensitivity will move us on from today's bulging jacket pockets and wire-laden seams that provide crude location-awareness and the conscious connectivity of a barcode reader.

What kinds of wearables do you think will first draw attention in the consumer space? How do you think mainstream wearables like Levi's digital jacket and Nike's washable equipment-laden clothing will go over with the general public?

Wearables first appeared in the business community where hands-free and head-free operation provided value for applications like vehicle inspection and inventory management. In a similar way, specialized clothing will take off first. A skiing jacket is a good example. Purchasing decisions have safety and comfort dimensions to them. People already pay a premium for skiwear. Adding instruction and safety services would be a natural extension to the value propositions offered. Children's location-tracking clothes would also strike a basic need for parents. Cyber jackets as trendsetting fashion are more difficult to predict.

What do you envision is the timeline for widespread consumer adoption of wearable computers?

At the launch of the Omnigo100 in 1995, I had a bet with the then Marketing Manager of the Asia-Pacific PC Division that in 10 years -- 2005 -- people would be wearing eyeglass displays in public [tube trains, etc.] like they wear music devices. [Using these displays,] they would be watching movies, playing games and working. I still haven't given up on that one, though it's looking close. Content and communication will continue to provide the major value clusters.

How will this shift to wearables begin?

Specialization will proliferate over the next five years. Specialized clothing, like ski jackets, will provide instruction based on feedback from the pressure- and shape-sensitive material from which they are made. Located audioscapes will be delivered to headsets via WaveLan, and digital audio devices will enrich our experiences of the places we visit.

In the next few years, a proliferation of WaveLan cells in the community will encourage the use and increase the value of ultra-portable devices based on ultra-portable technology. This will be the start of the creation of a fourth digital dimension, which has a one-to-one correspondence with the physical environment and provides the visitor with digital experiences located in the physical world.

What are the biggest research and development challenges for wearable computing?

The biggest challenges are a mixed bunch of science, commerce and design. Technically, creating IT components that are meant to be worn, washed, mixed and matched is a real challenge requiring new materials, new packaging and standards for aggregating sensors, storage and embedded processing.

Network management of heterogeneous, ad hoc wireless networks and massively mobile distribution will stretch current models.

Monitoring performance, [facilitating] payment for services, delivering varying shades of copyright, distributing the payment along the value chain and maintaining privacy for the wearer are all research areas that will affect the degree to which wearables or ultra-portables are accepted.

Interaction design for wearables is an extreme case of appliance design. They are applying their approach beyond wearables, but the notion captures for me the goal of wearable I/O to deliver the right experience at the right time whether it is skiing without getting hot and sweaty, polite and timely instruction as you learn a new skill, being told the name of the person to whom you were introduced yesterday and who is just about to shake your hand or experiencing a remote hug from a close friend in another city.

Why are "smart" features particularly relevant for wearable components?

The idea is that wearables, by design, have the potential to be with you, operating and accessible, in many more situations than devices you have to take out of your pocket and switch on. This opens up many more opportunities for interventions in the wearer's daily activities. In order to make these interventions desirable and effective, it is essential to know as much about the wearer's physical, social and, in some cases, emotional context as possible.

At the surface level, context information can help in choosing the correct interaction mode for the right level of attention -- or to avoid social embarrassment. A mobile phone that knows when you are concentrating on that match-winning putt might hold or take a message rather than put you off your stroke! Access to contextual information takes us beyond the notion of anything, anytime, anywhere to the more useful idea of the right information [and experience] in the right place and at the right time.

Given "storage" realities and/or file size realities -- how portable can computing really be? Take standard MP3 files, for example. At 3-5 MB per song, "wearing" a good collection of music could require gigabytes of storage space. So an MP3 "watch" might not be practical. How does storage come into play in this market -- and does the future hold answers to this problem?

One answer to the storage question you pose is greater density, and the Atomic Resolution Storage research being done in HP Labs may deliver there. The other solution is networking. You wouldn't need to store the whole of your collection on your watch, just the 100 or so [tracks] you want to select from this week. I carry 75 tracks on the CF card in my Jornada.

If you need a track outside the selection on your watch, you can download it from your personal Web store. If you don't own the song, maybe you'll pay 50 cents to download it for the day from the virtual music store behind the band's poster you just passed in the subway.

Is it feasible to think about a wearable computer that could/would replace a user's desktop? Or do we need to think about wearable computing in terms of specific functions rather than general computing?

Many people in business use a notebook as their primary machine now. The driver for this practice was not so much improvements in the form factor or portability of notebooks -- some of these are still quite large and weighty -- but the ease of connection to company networks from different locations. The mobility of many of these devices is limited to the car journey between home and work.

As the technical boundaries are pushed, the possibility of having your computing environment accessible to you wherever you are and whatever you are doing becomes a realistic possibility.

Whether wearables will replace desktops entirely is more of a sociological question than a computer form factor value question. Whether or not desktop computers have a place in the future will depend more on the type of work people do. Today, there are many tasks that are best performed sitting at a desk in an area conducive to concentration. Whether the computing device is a desktop or notebook form factor depends on preference or company policy.

What may happen in the future is that some interaction devices will become located. A study or an office may have a well-designed, substantial keyboard and stylish display. The proximity of your jacket hanging on the back of your chair will be enough to make the display and keyboard available to you as interaction devices. The precursor to this can be found in the way notebooks are used with docking stations -- though in many cases the notebook's keyboard is still used.

What is your favorite wearable from those you've prototyped?

The BlazerJet has a special significance because it was our first working test bed that led to a number of demonstrable applets and was the catalyst in establishing the group at Bristol University.

A few years ago on the way to developing the Jornada camera, Guy Adams and Andrew Hunter put a tiny camera into a wristwatch that sent its pictures via radio to a TV monitor. That was cool.

One wearable demo that amused me immensely was given by one of Roz Picard's group at MIT. Muscle tension sensors were placed above the eyebrows of students listening to a lecture of increasing complexity. Their confusion was shown by connecting the sensors to a laptop display.

I couldn't help thinking it would have been better to have connected the sensors to a spinning bow tie, which would slowly start to turn as confusion set in. As the students became more confused, their ties would spin faster and faster. A sea of variously spinning bow ties would give the presenter instant feedback. Only clarity would slow the bow ties down. The goal would be to finish the presentation with as many stationary bow ties as possible.

How might wearables be used in daily life?

I'm leaving home for work. Just as I am trapped between the third level virtual wizard that hangs around the No. 33 bus stop and the commuter in the pin-striped suit I pass daily and with whom I exchange rounds of digital paintball, a colleague from the real world would interrupt, creating a digital window, to ask how the pilot trial of the new jacket was going.

Input Parameters for Springback Simulation using LS-DYNA

Bradley N. Maker
Xinhai Zhu
Livermore Software Technology Corporation - June, 2001

LS-DYNA has been applied to springback simulation by a large number of users, with generally mixed results. Some results have demonstrated 70% accuracy or better, while others have been entirely misleading. In order to eliminate inconsistent results, this report presents a standard procedure for conducting springback simulations with LS-DYNA. The “seamless” and “dynain” methods for springback are described, followed by a description of general implicit springback problem set-up. Recommendations are given for anticipating and improving springback prediction accuracy.

Wherever possible, LS-DYNA keyword input data is shown to clarify the presentation. Recommended input parameters are identified in **boldface** type and included in boxed keyword input syntax for quick reference. A boldface zero value is entered for required input data which is model specific, such as the termination time **term**.

The Forming Simulation

Results from the forming simulation provide the starting point for the springback simulation. The most important factor in springback accuracy is the accuracy of the forming simulation. This is essential! If trouble occurs during springback, look for the cause in the forming analysis.

In explicit forming simulations, run time can and should be greatly decreased using mass scaling and/or artificially high tool velocity. Both these methods introduce artificial dynamic effects, which must be minimized to reasonable levels in an engineering sense. A single independent parameter describing artificial dynamic effect is the number of explicit time steps (cycles) taken per millimeter of tool motion.

Relatively more cycles per millimeter are required when the forming process allows large unrestrained sheet motion. An example is the crash forming process, which uses no binders. Relatively fewer cycles per millimeter are necessary when the sheet is heavily constrained with binders and punch support. For most simulations, values of between 100 and 1000 cycles per millimeter produce reasonable results. If possible, or when it is otherwise necessary to repeat a simulation, use two different values and compare results to estimate sensitivity to artificial dynamic effects.
For an extensive description of input parameters for the forming simulation, see Maker and Zhu [1].

Springback Methods

LS-DYNA springback simulations can be performed by several methods. A standard explicit dynamic method may not be used since the objective is to obtain a static springback solution free from dynamic oscillations. Explicit dynamic relaxation is a viable method. The preferred approaches to springback employ the static implicit method. The two most common implicit approaches, the “seamless” and “dynain” methods, are described below.

Seamless Springback Method

In the seamless method, LS-DYNA begins by performing an explicit forming simulation. When the termination time is reached, LS-DYNA automatically and seamlessly switches to the implicit method, and continues with the springback simulation. At the time of switching, a user-specified list of parts (the sheet blank) are retained as active, and the remaining parts (the rigid tools) are deleted from the model. All contact interfaces are also automatically deleted. An optional list of nodal constraints are activated to eliminate rigid body motion after the tools are removed for the static springback simulation. (Required constraints are discussed later in this document.)

```
*INTERFACE_SPRINGBACK_SEAMLESS
$      psid
          0
$       nid       tcode      rcode
          0           0          0
          0           0          0
```

After switching seamlessly, LS-DYNA proceeds to perform a static implicit springback simulation. A special set of defaults are used which eliminate all requirements for *CONTROL_IMPLICIT keyword input. These alternate “springback” defaults are identified clearly in the user’s manual, and affect the time step size, artificial stabilization, and automatic time step control parameters. Default parameter values can be overridden by including optional *CONTROL_IMPLICIT keywords into the forming input deck.

Element Formulation Switching

An option is available to automatically switch shell element formulations when using the seamless springback method. When activated, all shell elements which are retained in the model during implicit springback are treated with the S/R Hughes-Liu element formulation #6. This option allows the user to reproduce previous results which were obtained using LS-DYNA3D for forming simulation and LS-NIKE3D for springback simulation, when default element formulations were chosen in both software. For best springback accuracy, however, it is recommended to use the Fast Shell element #16 for both the forming and springback simulations, so the element formulation switching option is not required.

The DYNAIN File Method

At the end of the forming simulation, LS-DYNA can output a keyword-formatted file named “dynain” containing the deformed mesh, stress, and strain state. The dynain file is requested using the keyword *INTERFACE_SPRINGBACK_DYNA3D. Input the id **psid** of a part set containing a list of parts to be included in the output file (usually just the sheet workpiece). An optional list of extra node constraints can be included, which are applied as the dynain file is written. These constraints provide a convenient way to eliminate rigid body motion in springback calculations. (Required constraints are discussed later in this document.)

```
*INTERFACE_SPRINGBACK_DYNA3D
$     psid
         0
$      nid     tcode     rcode
         0         0         0
         0         0         0
```

The dynain file can be used to perform many follow-on simulations such as springback, trimming, or additional forming. It can be included into a new LS-DYNA input deck so that each simulation is performed independently. This procedure avoids cumbersome binary restart databases, and separates multi-stage forming and springback jobs into more manageable pieces. For this reason, the dynain file method is the recommended method for springback simulation. In the dynain file method, the input deck for springback simulation is easily constructed using the part, section, and material information from the original forming model, and the node, element, and initial stress and strain information from the dynain file. A few additional keywords must be added to control the implicit springback process.

Mesh Coarsening

Accurate forming simulation requires a very fine mesh over tool radii – typically at least four elements are needed around a ninety-degree radius. Surprisingly, much of this mesh refinement can be removed prior to springback analysis without significant loss of springback accuracy. Mesh coarsening is the procedure used in LS-DYNA to automatically combine neighboring elements in flat regions of the mesh.

Mesh coarsening can be applied to both uniform and adapted meshes. Mesh coarsening provides three significant benefits for implicit springback analysis: improved convergence behavior during nonlinear equilibrium iteration, due to reduced numerical truncation error; and reduced memory and cpu requirements due to the reduced model size.

The coarsening procedure is performed at the beginning of a simulation. Coarsening is applied to the input mesh, and then the simulation proceeds using the coarsened mesh. If a zero termination time is specified, and the keyword *INTERFACE_SPRINGBACK_DYNA3D is included, a new dynain file will be output containing the coarsened mesh and the simulation will terminate. (Be careful to rename the first dynain file to avoid overwriting it with the second dynain file.) This is the recommended procedure:

- *forming simulation, output dynain file at termination time*
- *mesh coarsening with zero termination time, output second dynain file*
- *springback simulation using coarsened mesh*

Coarsening is activated using the keyword *CONTROL_COARSEN. The only required input parameter is the flatness tolerance **angle**, which limits coarsening to areas of the mesh where the angle between normal vectors of adjacent elements is less than the input value. A recommended value is **angle = 8** degrees, although values of up to 12 degrees have been used successfully. An optional list of **nseed** nodes are used to initialize the search for candidate groups of elements to be coarsened. Seed nodes can be used to assist the automatic searching logic in finding isolated regions of mesh within a part which need to be coarsened. Up to eight nodes may be defined. A seed nodes identifies the *center of a group of four elements* which may be combined into one. To avoid leaving a single row of fine elements around the perimeter in the coarsened mesh, seed nodes should not be chosen on mesh edges or refinement boundaries.

```
*CONTROL_COARSEN
$  icoarse     angle     nseed
         1       8.0         0
$       n1        n2        n3        n4        n5        n6        n7        n8
         0         0         0         0         0         0         0         0
```

An optional number of boxes may also be defined which protect regions of the mesh from coarsening, using the keyword *DEFINE_BOX_COARSEN.   The parameter **iflag** indicates whether elements lying inside or outside the box will be protected.

```
*DEFINE_B0X_COARSEN
$    boxid      xmin      xmax      ymin      ymax      zmin      zmax     iflag
         0         0         0         0         0         0         0         0
```

Implicit Springback

Creating an input deck for implicit springback using a dynain file is simple. Keywords are required to activate the implicit method, and to select the time step size and the termination time. Extra constraints can be added using nodal SPCs to eliminate free rigid body motion of the sheet when the tools are removed. For difficult springback jobs, optional keywords are available to request multi-step springback unloading, to automatically adjust the time step size according to the difficulty of each step, and to control the linear and nonlinear equation solvers. A short template file can be used to save typical values for these keywords. Other necessary input, such as part, material, and section definitions, can be taken directly from the original forming input deck.

Activating The Implicit Method

Since springback is a static process, the implicit solver should be used. This solver is activated using the first parameter **imflag=1** on *CONTROL_IMPLICIT_GENERAL. The time step size is also input here using **dt0**, and can be chosen arbitrarily in most cases since the solution is static. A physically reasonable time step size should be chosen, so use **dt0=0.001** seconds:

```
*CONTROL_IMPLICIT_GENERAL
$   imflag       dt0      iefs   nstepsb      igso
         1     0.001         0         0         0
```

Choosing The Number Of Time Steps

The termination time and time step size determine the total number of springback steps. Springback of most reasonably stiff panels can be performed in a single step, so select the termination time **term=dt0** using *CONTROL_TERMINATION. Some difficult parts require several steps. A reasonable starting point for a difficult, multi-step analysis is four steps, or term=4*dt0.

```
*CONTROL_TERMINATION
$     term
     0.001
```

Required Constraints

All static simulations, including implicit springback analysis, require that rigid body motions be eliminated by defining constraints. These constraints are required since dynamic inertia effects are not included in a static analysis. Without constraints, a tiny applied load would cause the entire workpiece to move rigidly an infinite distance without creating any stresses. Mathematically, this means that without any constraints the global stiffness matrix for the model is singular, and the inverse can not be computed. When constraints are properly chosen this rigid body motion will be eliminated, and the model will deform freely without developing any reaction forces at the constraint points.
Constraints can be applied using the *INTERFACE_SPRINGBACK keywords, or, when the dynain file method is used, constraints can be added to the springback input deck using *BOUNDARY_SPC_NODE. Parameter **nid** indicates the constrained node ID, and a value of one is entered for each degree of freedom (**dx**, **dy**, **dz**) to be constrained:

```
*BOUNDARY_SPC_NODE
$      nid       cid        dx        dy        dz        rx        ry        rz
         0         0         0         0         0         0         0         0
```

Enough constraints must be defined to eliminate six rigid body motions in the model – three translations and three rotations. In theory, this could be accomplished by constraining all six degrees of freedom at a single shell element node point. In practice, numerical truncation error is introduced when rotational degrees of freedom are used to eliminate rigid body motion. The recommended method is therefore to constrain selected translational degrees of freedom at three nodes.

The three constraint nodes should be chosen well separated from each other, and away from edges and flexible areas in the part. The first node “A” receives constraints to all three translational degrees of freedom, and defines the reference point in the model where springback displacements are zero. The second node “B” is located away from node “A” along the global X-direction. Constraints are applied at node “B” to eliminate global Y- and Z-translation. The third node “C” is located away from node “A” along the global Y-direction. Only the global Z-translation is constrained at node “C”. Figure 1 shows a diagram of the location of these nodes on the model.

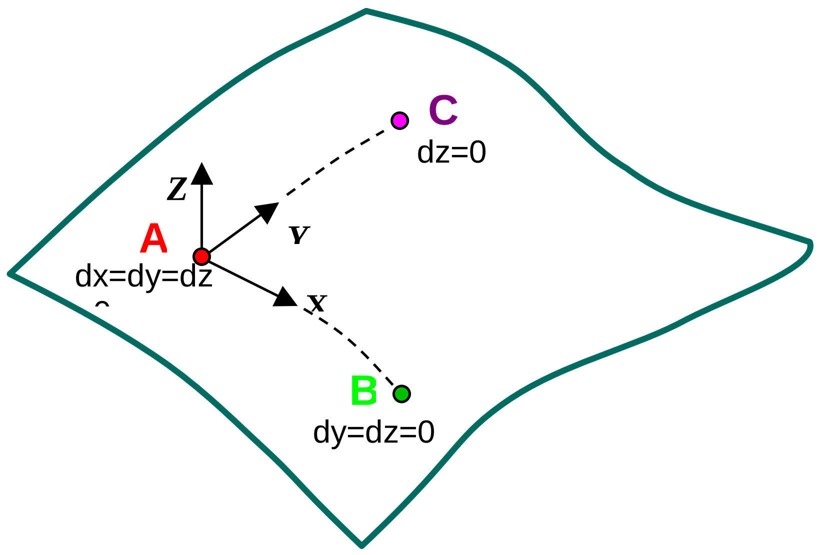


Figure 1 - Diagram showing location of constraint nodes on a typical springback model. Node “A” is the reference node, where all displacements are constrained. This eliminates the three translational rigid body motion of the part. Selected translational degrees of freedom are constrained at nodes “B” and “C” to eliminate the three rigid body rotations of the part about node “A”.

Constraints For Symmetric Models

Some stamping models include only one half of a symmetric panel, such as a hood or deck lid. In these cases, symmetry constraints are applied along one edge of the mesh. To eliminate rigid body motion during springback for these parts, constraints need only be added to two nodes chosen on the symmetry plane: completely constraining all translations for the first node, and eliminating one additional in-plane motion for the second node. Over-constraining a symmetric model by choosing three nodes according to figure 1 can lead to incorrect results. Figure 2 shows an example for the case of symmetry in the X-Z plane.

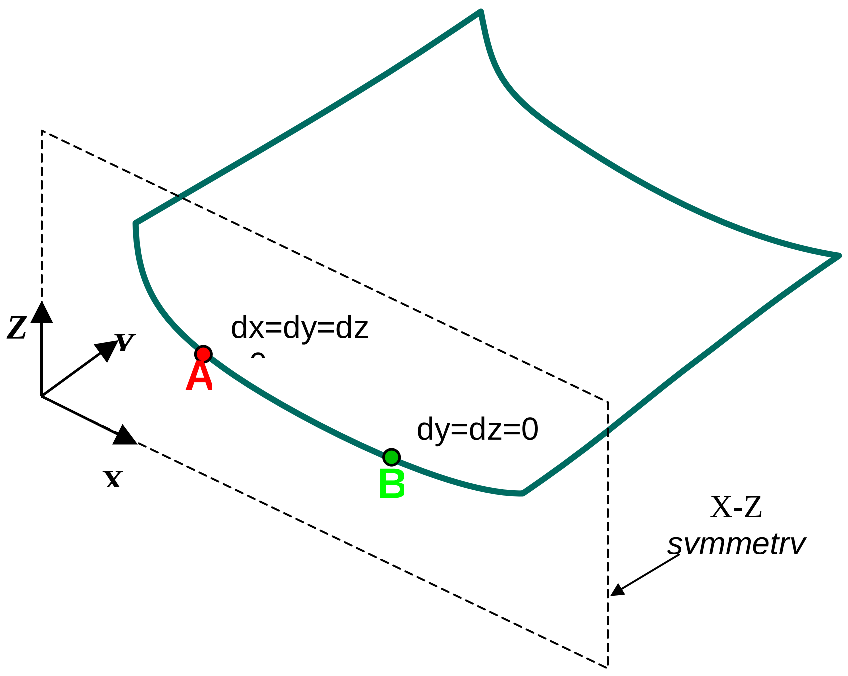


Figure 2 - Diagram showing location of constraint nodes for a symmetric part. The plane of symmetry in this example is the X-Z plane. Constraints must be added to *two nodes on the symmetry plane* to eliminate rigid body motion during static springback analysis. Node "A" is the reference node, where all displacements are constrained. This eliminates the three translational rigid body motion of the part. In addition to the standard symmetry constraints, selected translational degrees of freedom are constrained at node "B" to eliminate the three rigid body rotations of the part about node "A".

Other LS-DYNA Input Parameters

The remaining necessary input parameters can be taken directly from the forming simulation input deck, and should not be modified for springback analysis. These include the *PART, *MAT_…, and *SECTION keywords which describe the workpiece. For recommended values of these parameters, see Maker and Zhu [1].

Running The Nonlinear Implicit Springback Simulation

Unlike explicit simulations where tiny time steps are completed very quickly, a large implicit simulation may take many minutes to complete a single time step. By default, LS-DYNA issues very little screen output information when running in implicit mode. Optional input parameters and interactive controls are available to produce more information about the progress of the simulation, as described below.

Equilibrium Iterations and Convergence

During each time step, the nonlinear solver searches iteratively to find static equilibrium. Activate the nonlinear solver print flag **nlprint=1** using the *CONTROL_IMPLICIT_SOLUTION keyword, or interactively type "<ctrl-c> nlprint" to see the progress of these iterations appear on the screen. The current displacement and energy norms are displayed each iteration, as shown in figure 3. These must both be decreased below their respective tolerances **dctol** and **ectol** before equilibrium is reached. The default values of these tolerances, 0.001 and 0.01 respectively, are generally good and need not be changed.

```
BEGIN implicit time step       3
============================================================
                time =  1.09990E+00
   current step size =  3.67821E-01

 Iteration:   1     *|du|/|u| =  1.0894498E-01     *Ei/E0 =  1.8731172E+00

 DIVERGENCE (increasing residual norm) detected:
     |{Fe}-{Fi}| ( 1.0547507E+07) exceeds |{Fe}| ( 9.1389570E+06)
 automatically REFORMING stiffness matrix...

 Iteration:   2     *|du|/|u| =  3.8969724E-03     *Ei/E0 =  3.3420090E-02
 Iteration:   3     *|du|/|u| =  6.3582980E-03     *Ei/E0 =  3.3460971E-02
 Iteration:   4     *|du|/|u| =  1.3780216E-03     *Ei/E0 =  6.2154527E-03
 Iteration:   5     *|du|/|u| =  6.0081244E-03     *Ei/E0 =  7.7976128E-03
 Iteration:   6     *|du|/|u| =  1.4377093E-03     *Ei/E0 =  8.9132953E-03
 Iteration:   7     *|du|/|u| =  6.4089308E-03     *Ei/E0 =  1.7184228E-02
 Iteration:   8     *|du|/|u| =  1.8267103E-03     *Ei/E0 =  1.9337881E-03
 Iteration:   9     *|du|/|u| =  1.9491626E-03     *Ei/E0 =  2.3472405E-03
 Iteration:  10     *|du|/|u| =  2.2147158E-03     *Ei/E0 =  1.5075735E-03
 Iteration:  11     *|du|/|u| =  1.8921960E-03     *Ei/E0 =  1.9947323E-03
 Iteration:  12     *|du|/|u| =  1.5758326E-03     *Ei/E0 =  7.9428701E-04

 ITERATION LIMIT reached, automatically REFORMING stiffness matrix...

 Iteration:  13     *|du|/|u| =  7.1106170E-04     *Ei/E0 =  3.0991789E-03

 Equilibrium convergence summary for time step       3 at time =  1.0999005E+00
     Number of iterations to converge         =   13
     Number of stiffness reformations         =    2
```

Figure 3 – By selecting nlprint=1 on *CONTROL_IMPLICIT_SOLUTION, or by interactively typing "<ctrl-c> nlprint", the progress of the iterative equilibrium search will be displayed to the screen. Output is shown for a typical implicit step.

The equilibrium search is performed using a Newton-based method. By default, the "BFGS" method is used, where a new stiffness matrix is formed after every 11 iterations. For difficult springback problems (flexible parts with large springback deformation) the "Full Newton" method is better, since this method forms a new stiffness matrix after every iteration. To activate the Full Newton method, set the iteration limit between stiffness reformations to **ilimit=1**, and increase the maximum allowable stiffness reformations per time step to **maxref=100**. In some cases, the full Newton method will perform better if the line search is disabled using **lstol=99999**.

```
*CONTROL_IMPLICIT_SOLUTION
$  nlsolvr    ilimit    maxref     dctol     ectol     rctol     lstol
         0         1       100       0.0       0.0         0    99999.
$    dnorm   divflag   inistif   nlprint
         0         0         0         1
```

Solving The Linear System [K]{x}={f}

The stiffness matrix formed during implicit analysis requires a large amount of memory, and computing its inverse requires most of the CPU time. These operations are performed by the linear equation solver, whose control parameters are found on *CONTROL_IMPLICIT_SOLVER. The default linear solver **lsolvr=0** is generally recommended. A double precision solution to the linear system [K]{x}={f} can be selected using lsolvr=6, however this alone does not often improve results, and does increase memory requirements by 2x. Solver #6 is very efficient at utilizing scratch files on disk to run in "out-of-core" mode, so it is recommended when computer memory resources are limited.

```
*CONTROL_IMPLICIT_SOLVER
$   lsolvr    lprint    negeig
         0         0         0
```

A summary of memory and CPU usage is printed to the screen when the **lprint** flag is activated, either by input using lprint=1, or interactively by typing "<ctrl-c> lprint". The interactive control can be issued a second time to stop printing the memory information. Memory limits can be increased using the execution line argument "**memory=**", where the default is memory=8500000. Note that 1 Mword = 4 Mbytes in single precision, and 1 Mword = 8 Mbytes in double precision.

```
 SPARSE LINEAR EQUATION SOLVER STORAGE data  (Mwords)
           (  225972 degrees of freedom)
               pointer arrays:   initial =   11.523
                                  actual =    6.413
                  stiffness coefficients =    6.187
     Factorization Workspace (estimated)
                                symbolic =   14.015
                                 numeric =   18.335
     Final Storage Requirements (10% for pivoting)
                                            incore  out-of-core
                  symbolic factorization =    5.276     5.276
                   numeric factorization =   69.772     5.292
                        numeric solution =   65.561     3.145
                                   TOTAL =   87.648    23.168
                         TOTAL available =   98.196    98.196

      an INCORE solution will be performed

                      Initialization CPU =  7.220E+00 seconds
              Symbolic Factorization CPU =  1.065E+01 seconds
               Numeric Factorization CPU =  8.539E+02 seconds
                    Forward/Backward CPU =  5.060E+00 seconds
```

Figure 4 – By selecting lprint=1 on *CONTROL_IMPLICIT_SOLVER, or by interactively typing "<ctrl-c> lprint", the memory and CPU requirements for the linear equation solver will be displayed to the screen. Output is shown for a production size springback model. This job will run in core memory since the total memory available (98.196 Mwords) is larger than the total required for incore solution (87.648 Mwords). The option "memory=100m" was used on the command line to request 100,000,000 words of memory.

Difficult Springback Simulations

The following section offers suggestions for solving difficult springback simulations. These typically involve very flexible parts, on which experimental springback measurements are also often difficult. In simulation, these parts usually present convergence trouble for the nonlinear equilibrium iteration process. A method is presented below for using several steps to simulate springback unloading, followed by a troubleshooting checklist with other modeling suggestions.

Multi-Step Springback for Difficult Parts

The applied load in a springback simulation results from the initial stress in the sheet, which is no longer in equilibrium once the tools have been removed. For difficult springback problems, this “load” must be applied slowly over several steps in order to divide the nonlinear springback response into manageable pieces. *Artificial stabilization* is the method used in LS-DYNA to distribute springback response over several steps. In this method, springs are artificially introduced to the model which restrict the motion of the sheet nodes. As the solution proceeds the spring stiffnesses are reduced, allowing more springback. When the termination time is reached the springs are completely removed, allowing completely unrestrained springback. It is important to reach the termination time completely, otherwise some artificial stabilization will remain in the model and the results will not be accurate.

To use multiple steps in a springback solution, the termination time must be extended. A good starting point for difficult jobs is four steps, so if the step size on *CONTROL_IMPLICIT_GENERAL is **dt0=0.001** then the termination time on *CONTROL_TERMINATION should be **term=0.004**.

Artificial stabilization is activated using **ias=1** on *CONTROL_IMPLICIT_STABILIZATION. When active, a message is printed to the screen at the start of each time step showing how much stabilization remains in the model. At the termination time, the message reports that artificial stabilization has been “completely removed”.

```
*CONTROL_IMPLICIT_STABILIZATION
$      ias     scale    tstart       tend
         1     0.001         0          0
```

The initial stiffness of these springs can be scaled using the input parameter **scale**. This parameter must be chosen using some engineering judgement about the flexibility of the panel being studied. Table 1 gives some guidelines on choosing **scale**.

type of panel	example application	scale
stiff, heavy gage	frame crossmember	1.000 (default)
stiff, standard gage	reinforced inner panel	0.100
flexible, curved	fender outer panel	0.010
flexible, flat	hood outer panel	0.001

Table 1 – When artificial stabilization is used for multi-step springback, the stabilization stiffness scale factor must be chosen according to the panel type. Note that stiff panels generally do not require multi-step springback, so the default value scale=1.000 must nearly always be reduced.

A small value for **scale** gives softer springs, allowing more springback in the first few steps of the simulation. If convergence of the first step is difficult, use a larger value for **scale**. If the first few steps converge in very few iterations but the last step is difficult, use a smaller value for **scale**.

If convergence trouble is encountered during the iteration process, automatic time step control is available to repeat a failed step using a smaller step size. Automatic time step control is activated using **iauto=1** on the *CONTROL_IMPLICIT_AUTO keyword. For difficult springback simulations, an aggressive time step control strategy can be used. Increase the optimum number of iterations using **iteopt=200**, and restrict the maximum time step size using **dtmax=0.001**. In this way, the stepsize will always be increased after successfully converging, until the maximum stepsize is reached.

```
*CONTROL_IMPLICIT_AUTO
$    iauto    iteopt    itewin     dtmin     dtmax
         1       200         0       0.0     0.001
```

Troubleshooting Checklist

The following sections offer suggestions for common springback problems.

Poor Accuracy

Most accuracy problems result from errors which were introduced during the forming simulation. By closely examining the forming model and results, it may be possible to identify problems and anticipate poor springback predictions before they are submitted. Follow the guidelines described in Maker and Zhu [1]. In particular, look for:

- Insufficient mesh refinement. At least four elements are needed around ninety-degree tool radii.
- Poor element aspect ratio. Use elements which are as nearly square as possible.
- Artificial explicit dynamic effects. Running the forming simulation too slowly or too quickly can introduce error. Check the number of cycles taken per millimeter of tool motion.
- Changes in element formulation. For best accuracy, the more expensive element #16 must be used in the forming simulation as well as during springback, even though this adds significant cost to the forming simulation.
- Changes in thickness integration points. The number of thickness integration points must never be changed between forming and springback simulation.

Incorrect or Insufficient Material Data.

The effective stress – effective *plastic* strain curve must be carefully checked:

- The first data point must be at zero effective plastic strain and yield stress (0.0 , σy).
- Stress and strain must increase monotonically.
- Slope of each segment must vary smoothly.
- Data must fully include the range of strain seen in the part, including very large strains seen at the outer surface in sharp corners. Do not rely on LS-DYNA to extend your curve.
- Avoid too many data points. Rely on at most four significant digits.

Incomplete Solution

Beware that if convergence fails, LS-DYNA will issue an *error termination* message, and a d3plot state will be generated containing the last trial equilibrium geometry. These results are not accurate! Similarly, if the final step of a multi-step solution is not completed successfully, artificial stabilization will not be completely removed, an *error termination* message will be written, and the d3plot results will not be accurate. Accurate results can only be obtained after a *normal termination*.

Incorrect Constraints

The model must be adequately constrained to remove rigid body motion, but should not be over-constrained. Review the above section “Required Constraints”.

Gravitational Effects

The shape of large, flexible panels can be affected by gravity. Gravity effects can be easily included in springback simulations using *LOAD_BODY and *DEFINE_CURVE keywords. Be careful to employ a consistent system of units when defining gravitational acceleration.

Loose Convergence Tolerance

Nonlinear convergence tolerances can be increased to allow premature convergence, leading to poor accuracy. The default tolerance values for dctol and ectol on *CONTROL_IMPLICIT_SOLUTION are generally adequate, and should not be increased. Decreasing these tolerances to enforce equilibrium more strictly can be beneficial, especially when a double precision executable is used.

Single vs. Double Precision

Use of a double precision version of LS-DYNA improves convergence behavior in many implicit simulations. Merely activating a double precision linear equation solver has marginal benefit in an otherwise single precision executable. Contact LSTC to see if a double precision executable is available for your computer platform.

Mesh Coarsening

Mesh coarsening should be applied to most production size jobs to combine small elements into larger ones, reducing cpu time, memory requirement, and numerical truncation errors. If the number of elements in the formed workpiece exceeds 50,000 consider using mesh coarsening.

Extrusion and Coining

Significant errors result from situations where the workpiece is pinched between upper and lower tools to the extent that it is extruded or coined. This does not include the normal action of binders. Accurate simulation of extruded and coined parts may require layers of 8-node solid elements and advanced friction models which are beyond the scope of standard stamping simulation.

Clearance and Home Gap

Many springback simulations (and experiments!) are very sensitive to sidewall clearance in tools, and the “home gap” left at the bottom of the punch stroke. Carefully verify that your model accurately represents these details by making measurements directly on the model using the post-processor. Errors of less than one millimeter can have substantial effects on accuracy.

FEA Information News Previously Showcased
Archived on the site on the News Page

March 04	**OASYS, LTD**	**Bra Analysis**
	Hewlett-Packard	**HP Pavilion Notebood**
	GissEta	**Distributor in India**
March 11	**EASi**	**Easi-Seal**
	Fujitsu Ltd.	**PRIMERGY**
	Flotrend	**Distributor in Taiwan**
March 18	**LMS Int'l**	**LMS Test.LAB**
	CEI	**EnVideo**
	Strela	**Distributor in Russia**
March 25	**AMD**	**Microprocessors**
	ANSYS	**DesignSpace V6**
	Kostech	**Distributor in Korea**

EVENTS

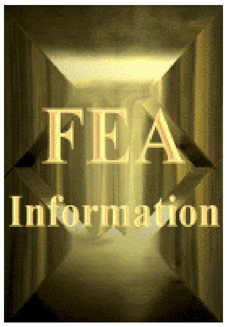
May 14 – 16	**Testing Expo 2002**
May 19 – 21	**7th International LS-DYNA Users Conference**
Sept 16 – 17	**Nordic Users Conference**
Oct 09 - 11	**CAD-FEM Users Meeting**
Oct 24 – 25	**Japanese LS-DYNA & JMAG Users Conference**
May 19 – 21, 2003	**BETECH 2003**

Future Articles Under Consideration:

- **India: The Other Silicon Valley**
- **Mesh Generating**
- **Optimization**

FEA Information Inc. Commercial & Educational Participants

Headquarters	Company	
Australia	Leading Engineering Analysis Providers	www.leapaust.com.au
Belgium	LMS, International	www.lmsintl.com
Canada	Metal Forming Analysis Corp.	www.mfac.com
China	ANSYS Bejing	www.ansys.com (link on international)
France	Dynalis	
Germany	DYNAmore	www.dynamore.de
Germany	CAD-FEM	www.cadfem.de
India	GissEta	www.gisseta.com
Italy	Altair Engineering srl	www.altairtorino.it
Japan	The Japan Research Institute, Ltd	www.jri.co.jp
Japan	Fujitsu Ltd.	www.fujitsu.com
Korea	THEME Engineering	www.lsdyna.co.kr
Korea	Korean Simulation Technologies	www.kostech.co.kr
Russia	State Unitary Enterprise - STRELA	www.ls-dynarussia.com
Sweden	Engineering Research AB	www.erab.se
Taiwan	Flotrend Corporation	www.flotrend.com
UK	OASYS, Ltd	www.arup.com/dyna
USA	Livermore Software Technology	www.lstc.com
USA	Engineering Technology Associates	www.eta.com
USA	ANSYS, Inc	www.ansys.com
USA	Hewlett Packard	www.hp.com
USA	SGI	www.sgi.com
USA	MSC.Software	www.mscsoftware.com
USA	EASi Engineering	www.easiusa.com
USA	DYNAMAX	www.dynamax-inc.com
USA	CEI	www.ceintl.com
USA	AMD	www.amd.com
USA	Dr. T. Belytschko	Northwestern University
USA	Dr. D. Benson	Univ. California – San Diego
USA	Dr. Bhavin V. Mehta	Ohio University
USA	Dr. Taylan Altan	The Ohio State U – ERC/NSM
USA	Prof. Ala Tabiei	University of Cincinnati
Russia	Dr. Alexey I. Borokov	St. Petersburg State Tech. University
Italy	Prof. Genarro Monacelli	Prode – Elasis & Univ. of Napoli, Federico II



University Student Forum

2002 AVI Competition

FEA Information, for Russia and the CIS States, is again offering cash prizes for its LS-DYNA **AVI File**

Contest

Monthly Award:

FEA shall pay student(s) $25.00 for winning AVI files each month and showcase them on the FEA Information website.

Annual Awards:

Each monthly winner will automatically be eligible for the following annual prizes:

First Place:	$200.00
Second Place:	$150.00
Third Place:	$100.00

Contest Rules:

FEA Information's LS-DYNA **AVI File Contest** is open to currently enrolled university students residing in Russia and the CIS States.

- Students may submit more than one AVI file.
- Single file size shall not exceed 4MB.
- Files should be transferred electronically to: feainfo@lstc.com
- A complete LS-DYNA input deck must accompany each submission.
- Payment is contingent on the winner first signing a copyright release.
- Winning AVI files and their respective input decks will be available to LSTC for promotional purposes without restriction.
- Cash prizes will be paid in local funds through Western Union branch offices.
- Files will be judged during the calendar month in which they are received.
- Monthly award winners will be contacted by the 15th of the following month and annual award winners will be contacted by January 31, 2003.
- FEA's determination of monthly and annual winners is final.