LS-DYNA 的二次开发环境及应用

Zhidong Han, Brian Wainscott
Livermore Software Technology Corp.

摘要

本文介绍了 LS-DYNA 新一代二次开发环境，编译连接过程和新增功能。新的开发环境完全兼容原有的开发环境，包括所有的材料模型，状态方程，单元类型，和求解器控制等各种用户子程序。新开发环境简化用户子程序的编译连接过程，直接生成动态连接库，与 LS-DYNA 主执行程序完全脱离。LS-DYNA 主程序支持多个用户子程序的动态连接库同时加载，按用户规则同时调用。新增功能包括支持用户自定义关键字，模型参数化及自动生成等。本文介绍了新开发环境的编译环境设置和编译连接过程，以及多个动态连接库的同时加载和调用方法。

引言

LS-DYNA 是一个大型的通用有限元程序，秉承一个执行程序，一个模型文件，执行多类型多物理分析的开发宗旨，致力简化用户建模过程并提高模型的重复利用率。LS-DYNA 内置的显式和隐式高效求解器及两者之间的动态互换，对解决多重非线性的大规模问题具有独特的优势，在实际工程中也得到非常广泛的应用。考虑到实际物理问题的复杂性和多样性，LS-DYNA 在开发初期就开放程序内核，让用户根据实际问题开发相应的用户模块来增强主程序的功能。现有的用户子程序大体上包括以下几类：

1）材料模型 UMAT
2）热材料模型 TUMAT
3）状态方程 UEOS
4）单元 UELEM
5）求解控制模块
6）输入输出模块

LS-DYNA 为每个模块都提供现成的模板程序，用户根据需要修改相应模块的模就可以实现二次开发。因此，对于有一定编程经验的有限元开发人员来说，LS-DYNA 的用户模块开发是相对比较简单的，尤其是全套的模板程序提供了很好的示例和开发基础，演示了在大变形大转动及各种非线性下的高效编程。这么多年来，有大批用户成功地根据自己的需要开发出高质量的用户子程序，实现各种复杂问题的计算。

从目前的一些用户的使用情况来看，二次开发比较容易出错的一个环节是编译和连接过程。目前 LS-DYNA 提供的一种开发方法是把所有主程序的 OBJ 文件打包成库文件提供给用户，而这些 OBJ 文件是在 LS-DYNA 标准编译环境下编译出来的半成品二进制文件。然后用户在自己的开发环境下编译其用户子程序，与主程序的 OBJ 库文件连接生成含有用户子程序的 LS-DYNA 执行程序。该方法的好处是生成的 LS-DYNA 执行程序内含用户子程序，方便执行。容易出错的地方是用户的编译环境往往 LS-DYNA 的标准编译环境不一样，可能会导致连接后的 LS-DYNA 执行程序不能正常工作。两个编译环境之间的差异可能会存在于各个方面，比如操

作系统类别和版本，FORTRAN 编译器的主版本及修正版本，C/C++编译器的版本及其所带的标准库文件等等。这些差异导致的错误有时还很难发现，对二次开发造成一定的困扰。

另外，LS-DYNA 得到越来越广泛的应用，在有些工业领域逐渐被认为是行业的标准分析软件。该行业的原材料供应商针对自己的材料等开发专门的材料模型及配套参数，提供给客户对用其材料的产品利用 LS-DYNA 进行分析。近几年来这种开发模式逐渐形成了一个发展趋势。从另一面看，制造商在一次分析中可能要用到多个供应商的不同材料模型，而如何保证所有供应商的子程序 OBJ 版本都与 LS-DYNA 一致并正确地连接在一起，难度往往较大。LS-DYNA 预分配的用户材料号从 41 号到 50 号，总共只有 10 个，如何协调众多供应商的材料号避免冲突，又增加协调的难度。因此，这些需求都对 LS-DYNA 的开发环境提出了更高的要求。

为此，在完全兼容现有用户子程序的基础上，LS-DYNA 推出另一种新的开发环境，在方便性，兼容性和灵活性等方面有很大的提高。首先，LS-DYNA 的主程序是一个可以进行独立分析的标准版执行程序，与用户子程序完全分离，也不依赖于任何用户子程序；LS-DYNA 的主程序可以单独升级，同时保持对用户子程序的兼容性，用户子程序无需重新编译和连接。其次，用户子程序是在用户的开发环境下的独立编译连接并生成的动态连接库，其所用的系统库函数不影响 LS-DYNA 主程序；动态连接库也保证了用户子程序的版本独立性和兼容性，无需和 LS-DYNA 主程序同时升级；有些情况下，动态连接库可以允许不同的 FORTRAN 编译器来编译和连接。最后，用户可以根据模型需要，在模型文件里面指定加载一个或多个多动态连接库，并与模型中的相应部件关联，实现动态调用。此外，若将来用户子程序的接口有一定的变化时，LS-DYNA 的高版本将考虑对以前版本的用户子程序的兼容性，可以直接加载以前版本的用户子程序的动态连接库，而用户无需重新编译和连接。

本文先对 LS-DYNA 的用户子程序做一个概述，介绍新开发环境下的开发过程及对源程序进行跟踪和调试过程，最后演示多个动态连接库的加载和调用过程。

LS-DYNA 用户子程序

1）材料模型 UMAT

用户材料模型是用户子程序中应用最广泛的，也是最实用的模块。LS-DYNA 中的用户材料号是从 41 号到 50 号，受关键字*MAT_USER_DEFINED_MATERIAL_MODELS 控制。所有用户材料子程序的统一入口子程序是 dyn21.f 中的

```
      subroutine usrmat (lft,llt,cm,bqs,capa,eltype,mt,ipt,
     . npc,plc,crv,nnpcrv,rcoor,scoor,tcoor,nnm1,nip,ipt_thk)
```

进入这个子程序后，再根据不同的单元类型选择不同的材料子程序

- urmathn: 体单元的三维材料模型
- urmats: 壳单元的二维平面应力材料模型
- urmatb, urmatd, urmatt: 三种不同的梁单元模型

这三个不同子程序根据各自的单元特点对应力应变进行相应的处理，再进入具体的用户子程序 umat41, umat42, … , umat50。这 10 个子程序是标准的串行版本模板，演示不同类型的材料模型，用户可以从这 10 个子程序模板中选一个较为贴近的开始。如果对计算效率要求较高，用户可以选与其对应的矢量版的模板，umat41v, umat42v, … , umat50v。矢量版子程序的特点是利用现代 128 位或更多位 CPU 的宽度，一次对多个操作数同时进行运算，比如一个 128 位 CPU 一次对 4 个 32 位的单精度实数进行运算，而一个 512 位 CPU 则一次对 16 个单精度实数

进行运算。用户开发出这个 umat 子程序就可以进行显式分析。如果要进行隐式分析，LS-DYNA 还需要该材料的切线刚度阵子程序。不同单元的切线刚度阵入口子程序分别是：

- urtanh: 体单元的三维材料模型
- urtans: 壳单元的二维平面应力材料模型
- urtanb: 三种不同的梁单元模型

这三个入口子程序也是根据各自的单元特点处理后，进入具体的切线刚度阵子程序 utan41, utan42, … , utan50， 或者其相应的矢量版子程序 utan41v, utan42v, … , utan50v。用户需要开发对应的 utan 子程序，就可以进行隐式分析了。

如果该材料需要支持 LS-DYNA 的界面单元，则用户还要开发对应的用户界面材料子程序 (Cohesive Materials)。界面材料子程序的统一入口子程序是 dyn21b.f 中的

```
      subroutine umat41c(idpart,cm,lft,llt,fc,dx,dxdt,aux,ek,
     & ifail,dt1siz,crv,nnpcrv,nhxbwp,cma,maketan,dsave,ctmp,elsiz,
     & reject,ip,nip)
```

进入后转入相应的具体界面材料子程序 umat41c, umat42c, … , umat50c。

2）热材料模型 TUMAT

热材料模型的材料号是从 11 号到 15 号，由关键字*MAT_THERMAL_USER_DEFINED 控制。其统一入口子程序是 dyn21b.f 中的

```
      subroutine thusrmat(mt,c1,c2,c3,cvl,dcvdtl,hsrcl,dhsrcdtl,
     1 hsv,iphsv,r_matp,crv,nnpcrv,npc,plc,nel,nep,iep,eltype,dt,atime,
     2 ihsrcl,hsvm,nmecon,temp,hsv2,hstored)
```

相应的用户热材料模型子程序是 thumat11, thumat12, … , thumat15，而没有单独的矢量版子程序。

3）状态方程 UEOS

状态方程在 LS-DYNA 的显式分析中非常重要，是冲击力学的基础。用户状态方程的号码是从 21 号到 30 号，由关键字*EOS_USER_DEFINED 控制。其入口子程序是 dyn21b.f 中的

```
      subroutine ueoslib(lft,llt,nes,mte,eosp,pnew,v0,dvol,
     & crv,nnpcrv,ivect,ihistp,iflag,nh)
```

相应的用户状态方程的子程序是 ueos21s, ueos22s, … , ueos30s, 及其对应的矢量版分别是 ueos21v, ueos22v, … , ueos30v。

4）单元 UELEM

用户单元开发分两类，壳单元和体单元。用户壳单元的号码是从 101 号到 105 号，由关键字*SECTION_SHELL 控制，统一入口子程序是 dyn21b.f 中的

```
      subroutine usrshl(rule,ixp,x,rhs,rhr,vt,vr,strain,yhatn,fibl,
     1 auxvec,mtype,ro,cm,csprop,nsubgv,mtnum,nfegp,ihgq,hgq,ies,ener,
     2 mpusr,lav,nmel,nnm1,mxe,ibqshl,iqtype,bkqs,gmi,ihgenf,hgener,
     3 lft,llt,rhssav,eig,eign,qextra,nmtcon,ithxpid,ietyp,cmusr,
     4 lenvec8,xipn,drlstr,rhsl,loceps,epsint,eosp,isdrill,rots)
```

进入到壳单元程序后，所有的变量都是在壳单元的单元坐标系中完成。壳单元的额外控制参数见*CONTROL_ACCURACY，*CONTROL_SHELL 等。用户单元开发的工作量及复杂度

要远超用户材料模型的开发，涉及到单元的形函数，B 矩阵，沙漏控制，单元内力集成等等。另外还需提供用户壳单元的质量阵，见 dyn21b.f 中的

```
subroutine ushlmass(iop,w,nxdof,x,rho,cm,lmc)
```

而用户体单元的号码也是从 101 号到 105 号，由关键字*SECTION_SOLID 控制，统一入口子程序是 dyn21b.f 中的

```
subroutine usrsld(mtype,cm,u,v,fv,x,auxvec,eosp,tnew,fval,ener,
. npc,pld,hgforc,ies,bqs,nhxbwp,hgener,mte,nmtcon,lav,ihg,
. nnm2,lft,llt,ibq,nes,idmp,kp,nnm1,mxe,iehgfg,strains,rhssav,
. volfrc,cmaux,eig,eign,idam,damag,lpwphv,rots,hges,lochvh,ithxpid,
. ietyp,cmusr,xipn)
```

质量阵的入口子程序是在 dyn21b.f 中

```
subroutine usldmass(iop,w,nxdof,x,rho,cm,lmc)
```

用户体单元的变量都是在整体坐标系中进行，对各向异性材料需要转动。开发的复杂度比较高，其模板中需要提供很多子程序。详细情况参阅 dyn21b.f 的用户单元模板。

5）求解控制模块及输入输出模块

这个部分的子程序很多，多数都在 dyn21.f 中，还有几个在 couple2other_user.f 和 dynrfn_user.f 中。这些子程序是分散在手册的不同章节，没有一个统一的说明。在关键字*MODULE_USE 一节中，这些子程序都被简单地分类处理。另外，在 LS-DYNA 手册第一卷关键字手册的附录 A-H 中对二次开发有非常详细的介绍。

LS-DYNA 用户子程序的编译和连接

在新的用户子程序开发环境中，LS-DYNA 的主程序与用户子程序完全分开，二次开发包中也不包含 LS-DYNA 主程序的 OBJ 文件。因此，新的二次开发包的文件很小，全部打包压缩后只有 165KB，极大地提高了用户子程序的编译和连接速度，使得二次开发更加方便。

二次开发包中包括以下三部分内容：

1） 各个用户子程序的模板，是 FORTRAN 的源程序，包括 dyn21.f 和 dyn21b.f 等；

2） 头文件，也是源程序，包含 LS-DYNA 中各个 COMMON BLOCK 参数，供二次开发使用；

3） 编译脚本文件，Makefile，用于编译和连接。

前两部分的源程序与用户子程序的具体功能相关。脚本文件 Makefile 是一个纯文本文件，可以用普通的文本编辑器修改，主要内容包括以下几个变量的设置：

```
MY_FLAG = -fPIC -O2 -safe_cray_ptr -xSSE2 -align array16byte ......
FC = /opt/platform_mpi/bin/mpif90
LD = /opt/platform_mpi/bin/mpif90 -shared -nofor_main
export MPI_F77 := /opt/intel/composer_xe_2013.5.192/bin/intel64/ifort
MY_TARGET = libusermat_105657.so
MY_OBJS = dyn21.o dyn21b.o init_dyn21.o ......
MY_INC = nlqparm define.inc define2.inc ......
```

其中：

- MY_FLAG 是 FORTRAN 编译器的标准选项。如果用户的 FORTRAN 编译器和 LS-DYNA 主程序的编译器的版本一样，不建议更改这些标准选项。
- FC 指定 MPP 的 FORTRAN 编译器，此例中给出的是 platform_mpi 的编译器。
- LC 指定 MPP 的连接器，此例中给出的是 platform_mpi 的连接器。
- Export MPI_F77 是用来指定真正的 FORTRAN 编译器，MPP 编译器会调用这个编译器来编译 FORTRAN 源程序。此例中指定了 Intel FORTRAN 编译器的的版本及其安装路径。FC，LC，MPI_F77 中指定的程序都包含有安装路径。如果用户机器上安装路径或版本与此不同，则需要修改相应的变量，否则不能正确编译连接。

- MY_TARGET 是指定动态连接库的名称，在 LINUX 系统下一般以.so 作为后缀，而 Windows 系统以.dll 作为后缀
- MY_OBJS 包含 LS-DYNA 的所有模板子程序的 FORTRAN 源码。有些模版源码可能没有用上，手工去掉或保留都可以，不影响真正开发部分的源码的执行。用户可以在这个变量里加入自己的源程序文件。
- MY_INC 包含 LS-DYNA 用户开发包的所有的头文件，用户可以添加自己的头文件，但不建议删除已有的头文件。

当这些变量都设置好后，在当前目录下运行 LINUX 系统的命令“make”来执行这个编译脚本文件，自动完成编译和连接过程，并产生 MY_TARGET 所指定的动态连接库。如果源程序有错误，则打印相应的错误信息，并终止编译连接过程。用户在修改相应的源程序后，可以再次执行“make”命令来重试编译和连接。另外，新的开发环境仅支持 LINUX 的单机或集群系统，对 Windows 系统暂时还不能支持。

在用户子程序开发过程中，经常需要对源程序进行跟踪和调试。用户只需将 Makefile 中的 MY_FLAG 变量里的优化选项“-O2”改为“-g”，就可以关掉编译器的优化功能并在动态连接库中加入源程序信息，方便对源码调试。调试 MPP 版本的 LS-DYNA，用户避免 MPIRUN 启动多进程，而是直接启用 gdb（或者其它的跟踪程序，如 idb，ddd 等）加载主程序，并在用户子程序中设置断点：

```
set breakpoint pending on
break <source file name>:<line number>
```

再用 r 命令启动 LS-DYNA 进入单进程模式运行。LS-DYNA 主程序加载带有源程序信息的动态连接库后就设置相应的断点，并在进入该用户子程序后就在该断点处停下来等待调试。

LS-DYNA 用户子程序的动态连接库的调用

在一般情况下，LS-DYNA 主程序进行普通分析是不加载任何用户动态连接库，也没有必要。只有当模型需要用到某个用户动态连接库时，则在原来的关键字文件中加入一个新的关键字 *MODULE_LOAD 来实现加载。该关键字的格式如图一所示：

Card Sets. Repeat as many sets data cards as desired (cards 1 and 2) to load multiple libraries. This input ends at the next keyword ("*") card.

Card 1	1	2	3	4	5	6	7	8
Variable	MDLID		TITLE					
Type	A20		A60					
Default	none		none					

Card 2	1	2	3	4	5	6	7	8
Variable	FILENAME							
Type	C							
Default	none							

图一　关键字*MODULE_LOAD 卡片

情形一：只有一个动态连接库

当一个模型只用到一个动态连接库的时候，只需要*MODULE_LOAD 就可以：

```
*MODULE_LOAD
my_mod
libusermat_105657.so
```

第一张卡片是给这个动态连接库在这个模型中定义一个标识名，不能重名。第二张卡片是动态连接库的具体文件名，可以包含绝对路径或者相对路径。文件名及其路径的长度限制为 80 个字符。如果不够的话，则需要用到另外一个关键字*MODULE_PATH 来指定动态连接库的路径。LS-DYNA 则会搜索这个路径并加载动态连接库。

只有一个动态连接库的情形是最简单的，也和以前的开发模式完全兼容。此情形下，LS-DYNA 主程序会自动把所有对用户子程序的需求都转到这个动态连接库。

情形二：调用多个动态连接库

若模型需要用到多个动态连接库，则可使用关键字*MODULE_LOAD 来单独加载每个动态连接库：

```
*MODULE_LOAD
my_mod
libusermat_105657.so
mod_a
/ext/libusermat_moda.so
mod_b
/ext/libusermat_modb.so
```

此例演示了同时加载三个动态连接库，并定义了相应的三个独立标识名："my_mode"，"mod_a" 和 "mod_b"。LS-DYNA 把这些动态连接库加载后，还需要另外一个关键字

*MODULE_USE 定义各种调用规则，把对用户子程序的调用转到相应的动态连接库。关键字*MODULE_USE 需要两张或更多的卡片来定义一个动态连接库的一个或多个调用规则。每个动态连接库都需要至少一个单独的*MODULE_USE 关键字来定义其调用规则。

The rules defined in *MODULE_USE are applied to only one dynamic library.

Card 1	1	2	3	4	5	6	7	8
Variable	MDLID							
Type	A20							
Default	none							

Rule Cards. Card 2 defines rules for the module specified in Card 1. Repeat as many cards as needed if defining multiple rules. New rules override existing rules, in case of a conflict. Input ends at the next keyword ("*") card.

Card 2	1	2	3	4	5	6	7	8
Variable	TYPE		PARAM1		PARAM2			
Type	A20		A20		A20			
Default	none		blank		blank			

图二　　关键字*MODULE_USE 卡片

*MODULE_USE 的第一张卡片输入动态连接库的标识名，后续的调用规则只适用于该动态连接库。第二张卡片定义规则，一张卡片定义一个规则。若需要定义多个规则，则可以重复这张卡片。当多个规则有冲突时，后输入的规则为准，因此定义规则的时候要注意顺序。另外也可利用顺序，把普通的规则定义在先，再定义一些特殊的规则。

在多数情况下，调用规则都很简单。借用上面的例子，假设模型中用到 my_mod 中 UMAT41，mod_a 中的 UMAT42，以及 mod_b 中的 UMAT45 和 UMAT46，则定义以下四个规则就可以了：

```
*MODULE_USE
my_mod
UMAT, 41, 41
*MODULE_USE
mod_a
UMAT, 42, 42
*MODULE_USE
mod_b
UMAT, 45, 45
UMAT, 46, 46
```

这样 LS-DYNA 就会把所有用到 UMAT41 的材料转到 my_mod，而其它的 UMAT 转到相应的动态连接库 mod_a 或 mod_b。假若模型里还用到了 UMAT48，但没有相应的规则指定如何调用，LS-DYNA 主程序就会报告错误并终止执行，指明 UMAT48 没有找到。

情形三：调用材料号有冲突的多个动态连接库

假若情形二中用户子程序有冲突，比如上例模型需要同时用到三个动态连接库 my_mod，mod_a，mod_b 中的 UMAT41 子程序，则需要更详细的规则来定义调用关系。上例的规则是针对真实的用户子程序名字来定义的，而此例中真实子程序名字有了冲突，就需要定义一个虚拟的子程序名称来。在 LS-DYNA 中的材料号从 1001 到 2000 被指定为用户材料模型，也就是说关键字*MAT_USER_DEFINED_MATERIAL_MODELS 的材料号 MT 既可以是 41 到 50，也可以是 1001 到 2000。这些虚拟的材料号并没有真实的用户子程序来对应的，必须通过规则来定义调用关系。有了这些虚拟材料号后，有冲突的材料号就可以重新定义：

- 所有用到 my_mod 中 UMAT41 的材料都定义为 1001
- 所有用到 mod_a 中 UMAT41 的材料都定义为 1002
- 所有用到 mod_b 中 UMAT41 的材料都定义为 1003

然后定义下面三个规则：

```
*MODULE_USE
my_mod
UMAT, 1001, 41
*MODULE_USE
mod_a
UMAT, 1002, 41
*MODULE_USE
mod_b
UMAT, 1003, 41
```

用虚拟的材料号来定义规则比较简单，只是需要对原来的模型文件中材料号做一点修改。除此之外，LS-DYNA 还允许对材料的标识号（MATID）定义调用规则，不过 LS-DYNA 中的用户材料模型限制同一个材料号(MT)的用户子程序必须要有相同的控制参数，参阅关键字*MAT_USER_DEFINED_MATERIAL_MODELS 中对 MT 的说明。因此，在实际使用上，虚拟材料号的方法比较适用，也不容易出错。

另外，针对材料号的规则不是仅仅对 UMAT 子程序定义的，LS-DYNA 会自动把这些规则应用到与 UMAT 的配套子程序上，如切线刚度阵子程序 URTANH，URTANS，URTANB，及界面材料子程序 UMATC 等。切线刚度子程序的调用还会自动根据单元类型来进入正确的入口，无需用户做更多的输入。

本文针对材料号举例演示了不同动态连接库的调用规则，而关键字*MODULE_USE 还可以对用户开发包中的所有子程序都可以定义调用规则，包括用户热材料，用户单元，用户控制模块。详细的规则定义参阅关键字手册中*MODULE 一节。

LS-DYNA 二次开发的其它新增功能

用户参数*USER_PARAMETER

为支持用户二次开发，LS-DYNA 的主程序还新增一些辅助功能，使得用户子程序的功能更完善更强大。用户参数*USER_PARAMETER 支持用户定义自己的标识字，并输入相应的模型参数，可以同时输入多个整数(I)，实数(F)，字符串(A)，或者多行文本(L)。用户子程序在运行时可以调用系统程序来获取这些参数：

```
subroutine get_usparam(key,nparam,itype,mptr)
```

其中 key 就是用户自定义的标识字，其长度可以多到 80 个字符。

用户模型自动化*USER_KEYWORD

这个功能是让 LS-DYNA 的主程序在读取模型文件是，调用用户子程序 rdusrkwd，让用户子程序根据自己的参数来直接生成模型，或者模型中的部分部件。这个功能应用在在很多方面，比如：

- 企业可以将标准化部件的不同密度的网格集中存放在中央数据库，用户子程序可以根据部件的标识号，分析的类型和要求，动态调入相应的网格和计算参数，并加入到当前模型中。
- 用户子程序在读取模型中材料的供应商和标识号等信息后，直接从本地数据库或供应商的远程数据库读取相应的材料模型设置及参数，使得模型本身更加自动化和智能化。
- 用户子程序可以为模块化产品自动建模，生成 LS-DYNA 的模型，简化建模过程。

用户应用程序开发

目前的用户二次开发是在 LS-DYNA 主程序的基础上，利用子程序来增强 LS-DYNA 的功能，是限制在 LS-DYNA 主程序的框架内。LS-DYNA 的最新开发环境将支持用户的独立应用程序开发，而不仅仅是动态连接库。用户自己的主程序在 MPP 的框架下与 LS-DYNA 进行实时交换数据，实现更加宽松的多物理场耦合分析。目前很多 LS-DYNA 的用户都有自己独特的独立应用主程序，若要改造为 LS-DYNA 的动态连接库运行，开发工作量较大，难度也较大。而新的开发环境将配备并行开发模板，该模板可以直接将用户的应用主程序加入到 LS-DYNA 的 MPP 并行环境中，自动实现 MPP 初始化对接，并在运行时与 LS-DYNA 进行数据交换。因此，新的开发环境将极大地方便用户进行不同层次的开发，与 LS-DYNA 实现耦合分析。

结束语

LS-DYNA 新的二次开发环境在完全兼容原来的用户子程序的基础上，简化了用户开发过程，提供了支持多用户模块的无冲突加载解决方案，并实现用户子程序的独立模块化。LS-DYNA 新的二次开发环境还为用户子程序提供自定义关键字，自动模型生成，用户应用程序 MPP 耦合等支持，使得用户开发的模块能全面地融入到 LS-DYNA 的开放架构中，解决更多的实际工程问题。

作者简介

*韩志东/Zhidong Han 博士 1998 年毕业于清华大学计算固体力学专业，于 2011 年加入 LSTC。他目前从事材料损伤断裂分析及厚壳单元等方面研发。