利用 LS_DYNA 和 LoCo 进行汽车模型组装和仿真

汪亚东[1]，彭冰元[1]

（1. 磐翼信息科技（上海）有限公司，上海 201100）

摘要：本文以汽车仿真计算为主要研究方向，以公开汽车模型 Toyota Yaris 为例，建立仿真数据库，介绍了仿真模型创建、模型组装的相关流程，并用 LS-DYNA 进行汽车碰撞安全性能的仿真计算。基于此，可大大简化仿真建模流程，实现流程自动化，减少仿真工程师的工作量，并且模型数据不丢失、可回溯，流程便于管理和监控，系统易于维护。

关键词：LS-DYNA；模型组装；管理；流程自动化

现今，车辆工程在对不同工况下车身安全性的仿真需求日益增加，这就要求各大汽车厂商在模型组装方面高标准化和自动化。求解器（例如：LS-DYNA）的输入文件，一个完整的 FE 模型都是根据工况由不同的 key 文件组装而成，包括：安全气囊、座椅、车门、假人、壁障等。模型的组装通常都是由工程师根据不同的工况多次编辑完成，但这样的工作耗时耗力而且易出错，所以需要新的解决方案。

LoCo（Loadcase Composer）正在被许多不同的 CAE 部门使用，目前 AUDI 有超过 700 名用户，德国大众超过 200 名用户都在使用 LoCo。LoCO 可以为用户提供一个便利的方式来管理 CAE 仿真模型，并且会根据用户选择的工况（loadcase，由一系列属性值来定义）来自动从模型数据库中选择相应的模型（key 文件）来组成整车模型（如图 1）。这样，可以大量减少甚至避免 key 文件的冗余使用和存储。这样的设计，使得不同的工况下，建模简单化，整车模型中相同的部分可以直接被调用，而仅仅需要修改相关工况下不同的部分。通过这样的方式可以实现模型组装的自动化。本文以公开模型 Toyota Yaris 为例，介绍如何使用 LoCo 来进行自动化的模型组装，来解释 LoCo 和 LS-DYNA 的相互集成关系。

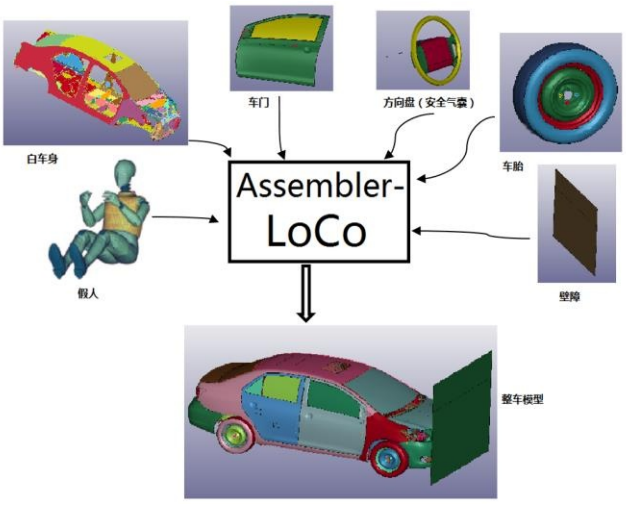

图 1 利用一系列 include 文件组装整车模型

1 LS-DYNA 和 LoCo 的内在基理

1.1 *INCLUDE 和 key 文件

LS-DYNA 进行碰撞仿真的输入文件由多个子 key 文件组成，这些 key 文件通过*INCLUDE 关键字链接到主文件（master）中。除了材料卡片等一般定义之外，key 文件只包含了整车模型的一部分。在模型组装的过程中，LoCo 会自动读取每个子文件，并为每个 key 文件生成*INCLUDE 关键字写入到 MASTER 文件中，组装成一个整车的模型，并且在子文件修改或者更新后，LoCo 会同时保存新旧两个子文件，在重新组装时会根据新版本的子文件生成新的关键字链接到主文件中。图 2 展示了 LoCo 中如何进行模型修改、替换等操作的逻辑。一个主文件、三个子 key 文件以及材料定义组成版本 1 的整车模型。当 sub2 子文件有过改动之后，LoCo 会生成一个新的版本，并且在组装时自动生成新的*INCLUDE 关键字链接到主文件中。

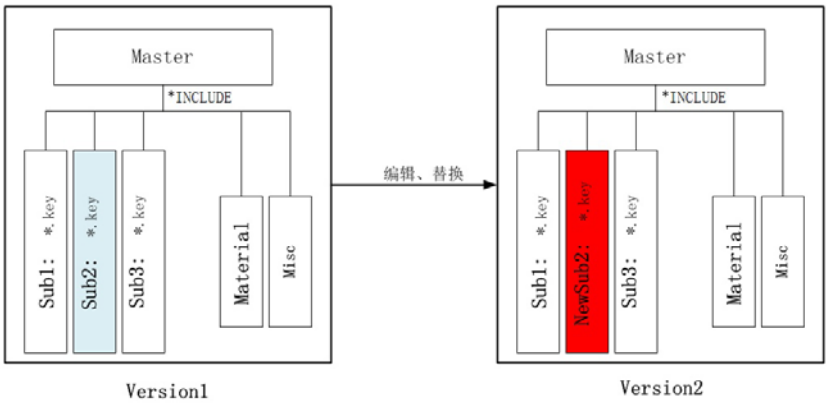

图 2 LoCo 修改、替换 key 文件

1.2 坐标转换

在 LoCo 中，内置了一个脚本，在导入的的过程中，会运行该脚本，会读取每个 key 文件的坐标信息，转化后并通过定义*INCLUDE_TRANSFORMATION 和*DEFINE_TRANSFORMATION 卡片，导入每个组件的 key 文件。该脚本还会创建会话文件（session files），以便在后处理器（LS-PrePost、Animator、META 等）中进行动画演示时修正各组件颜色。

```
*INCLUDE_TRANSFORM
axle_____rear_axl0783_67d347d5.key
$    IDNOFF    IDEOFF    IDPOFF    IDMOFF    IDSOFF    IDFOFF    IDDOFF
   3400000   3400000   3400000         0   3400000   3400000   3400000
$    IDROFF
   3400000
$    FCTMAS    FCTTIM    FCTLEN    FCTTEM   INCOUT1
         1         1         1         1         0
$    TRANID
         0
$
*DEFINE_TRANSFORMATION
      1001
$  Rotations
$    ROTATE       0.0       0.0       0.0       0.0       0.0       0.0  0.0
     ROTATE       1.0       0.0       0.0       0.0       0.0       0.0        0
     ROTATE       0.0       1.0       0.0       0.0       0.0       0.0        0
     ROTATE       0.0       0.0       1.0       0.0       0.0       0.0     30.0
$  Translations
     TRANSL   245.3753       0.0       0.0
     TRANSL       0.0         0       0.0
     TRANSL       0.0       0.0         0
$
```

图 3 定义坐标转换

1.3 接触

由于 LS-DYNA 出色的接触算法，对于汽车碰撞仿真工程师来说，定义部件间的接触就十分简单了。在 LoCo 中，模型组装时的接触有不同的方式。通常单面接触就可以完全定义整车模型的接触方式，包括偏置变形障碍等。*SET_PART_COLLECT 和 *CONTACT_AUTOMATIC_SINGLE_SURFACE 卡片提供了一种处理不相交

单元之间的相互作用的方法，可以防止部件之间或者同一部件内部元素之间的穿透。卡片*CONTACT_TIED_SHELL_EDGE_TO_SURFACE_BEAM_OFFSET 用于定义零件间的连接。卡片*CONTACT_NODES_TO_SURFACE 用于定义安全气囊的接触。卡片*CONTACT_TIED_SHELL_EDGE_TO_SURFACE 用于定义模型中的焊点。

```
*CONTACT_AUTOMATIC_SINGLE_SURFACE
$ Globalkontakt
$#     ssid      msid     sstyp     mstyp    sboxid    mboxid       spr       mpr
      1001         0         2         0         0         0         0         0
$#       fs        fd        dc        vc       vdc    penchk        bt        dt
       0.2       0.1       1.0       0.0       0.0         0       0.0       0.0
$#      sfs       sfm       sst       mst      sfst      sfmt       fsf       vsf
       0.0       0.0       0.0       0.0       0.0       0.0       0.0       0.0
$#     soft    sofscl    lcidab    maxpar     sbopt     depth     bsort    frcfrq
          2       0.0         0       0.0       0.0         0         0         0
$#   penmax    thkopt    shlthk     snlog      isym     i2d3d    sldthk    sldstf
       0.0         0         0         0         0         0       0.0       0.0
$#     igap    ignore    dprfac    dtstif    unused    unused    flangl
          0         1       0.0       0.0                         0.0         0
```

```
*CONTACT_TIED_SHELL_EDGE_TO_SURFACE_BEAM_OFFSET
$ Global Tied Kontakt fuer Anbindungen
$#     ssid      msid     sstyp     mstyp    sboxid    mboxid       spr       mpr
      2002      2001         2         2
$#       fs        fd        dc        vc       vdc    penchk        bt        dt
     0.000     0.000     0.000     0.000     0.000         0     0.000     0.000
$#      sfs       sfm       sst       mst      sfst      sfmt       fsf       vsf
     0.000     0.000     0.000     2.000     0.000     0.000     0.000     0.000
```

```
*CONTACT_TIED_SHELL_EDGE_TO_SURFACE
$ Global Tied Kontakt fuer Schweisspunkte
$:     ssid      msid     sstyp     mstyp    sboxid    mboxid       spr       mpr
      3002      3001         2         2         0         0         0         0
$:       fs        fd        dc        vc       vdc    penchk        bt        dt
       0.0       0.0       0.0       0.0       0.0         0       0.0       0.0
$:      sfs       sfm       sst       mst      sfst      sfmt       fsf       vsf
       0.0       0.0       0.0       0.0       0.0       0.0       0.0       0.0
$
*SECTION_SHELL
$#    secid    elform      shrf       nip     propt   qr/irid     icomp     setyp
         1         2       0.0         3       0.0       0.0         0         0
$#      t1        t2        t3        t4      nloc     marea      idof   edgset
      2.41      2.41      2.41      2.41       0.0       0.0       0.0         0
*PART
$emptySlaveTied
EmptySlave
         1         1   2000000         0         0         0         0         0
*SET_PART_COLLECT
      2001       0.0       0.0       0.0       0.0
         1
```

```
$ Contact Airbag to rest
*CONTACT_NODES_TO_SURFACE
$#      cid                                                                 title
$#     ssid      msid     sstyp     mstyp    sboxid    mboxid       spr       mpr
      1001   5000002         2         2         0         0         0         0
$#       fs        fd        dc        vc       vdc    penchk        bt        dt
       0.0       0.0       0.0       0.0       0.0         0       0.0       0.0
$#      sfs       sfm       sst       mst      sfst      sfmt       fsf       vsf
       0.0       0.0       0.0       0.0       0.0       0.0       0.0       0.0
```

图 4 利用多个关键字定义接触和连接

每个零件中都存在相同的关键字。每个零件在导入的时候都会记录下零件 ID 对应的偏移量（offset），因此，在组装的过程中，LoCo 会根据实际选择的属性值自动收集所有零件。这种方法的一大优点是：在实际仿真过程中，不需要去定义每个零件的接触面或者连接方式，而只需要去定义哪些零件需要被引用。因此，更换、删除、添加零件到模型中也变得很容易。

1.4 车轮的旋转

整车模型的初始速度可以通过 LoCo 的中参数进行定义，在*INITIAL_VERLECITY_GENERATION 卡片中可以引用这个参数。在这个卡片中，初始速度会被应用到一个给定的点集。该点集包含了模型的所有 node 点。这是通过在每一个 key 文件中定义*SET_NODE_LIST_GENERATE_COLLECT_TITLE 卡片来实现的。

为了实现模型中轮胎的旋转，需要为轮子定义不同的初始速度。因此，对于每一个车轮的 key 文件中，需要另外添加*INITIAL_VERLECITY_GENERATION 卡片。该卡片包含平移速度和角速度两个值。角速度的值也是通过 LoCo 中的参数来定义，角速度的表达式取决于车轮的半径以及车辆的平移速度。

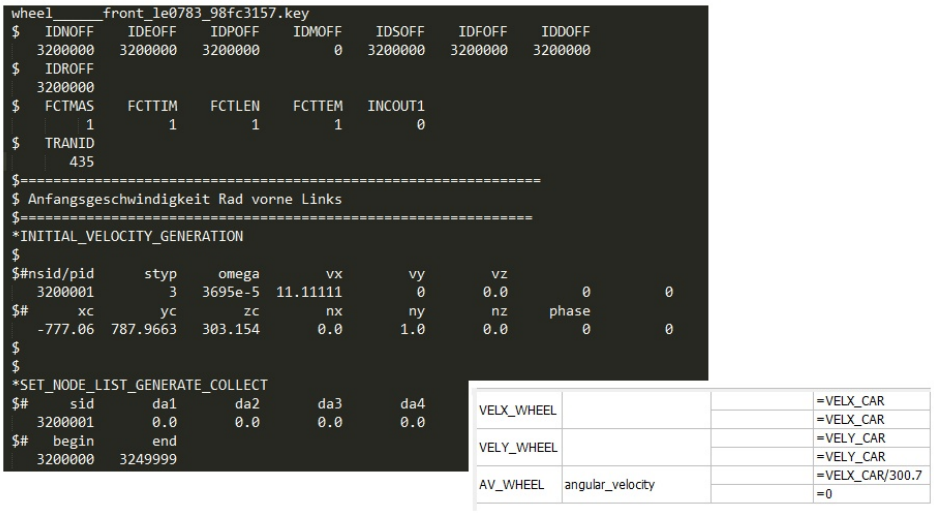

图 5 利用关键字和参数定义车轮速度

2 使用 LoCo 进行建模和仿真

2.1 模型整理和存储

网上有公开的 Toyota Yaris 公开有限元模型，用户可以自由检索资源并免费下载。该模型是整车模型，模型还包含壁障等组件。用户需要自己做一些修改，将整车模型剥离成各个组件，并且根据自己的需要修改网格，提高模型质量。

将剥离好的组件一个个导入到 LoCo 中，LoCo 会里的脚本会根据配置进行分类并存储各个组件到数据库。LoCo 允许多个用户同时编辑、修改同一个零件，并且跟踪处理组件的用户和时间，并同时保存所有用户的修改记录，有利于回溯。

LoCo 还可以集成其他外部前后处理程序（Hypermesh、ANSA、LS-PrePost 等。）。一旦所有组件被保存到 LoCo 中，用户可以直接在 LoCo 中直接调用前后处理器对组件进行编辑。

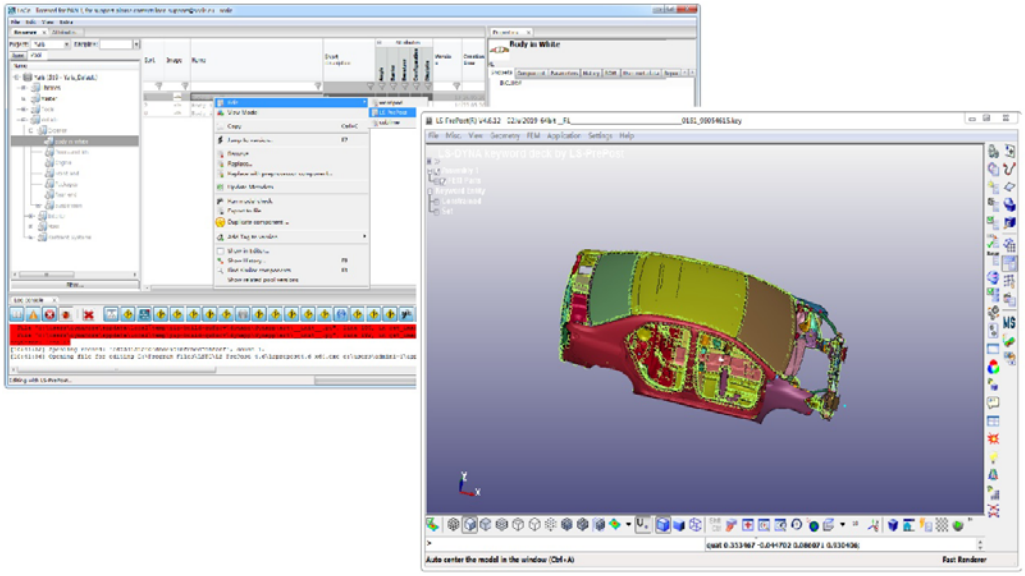

图 6 在 LoCo 中集成外部程序（LS-PrePost）对组件进行编辑

在 LoCo 中，材料库、壁障等公共件，是存储在一个在各个子共享库中，不同的项目可以调用这些共享库中。

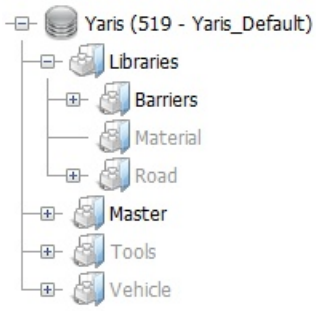

图 7 在 LoCo 中定义材料和壁障库

2.2 模型组装

除了整车的零件模型，模型组装还需要一些其他文件，包括 control cards、material cards 等。这些卡片也可以导入到 LoCo 中并保存。

在组装的过程中，LoCo 会通过预定义工况的属性值选择关联的组件并将其组装成整车模型。这些组件中包括：汽车零部件 key 文件、壁障 key 文件、材料卡、参数等。这些属性值都是用户定义的，用以区分同一类零件的不同功用。例如，定义汽车左舵和右舵的属性值： “Hand Drive”。当用户在定义工况时，选择的是左舵，那么 LoCo 在组装时就只会选择“lhd”对应的舵机进行组装。LoCo 也会根据用户定义的参数值修改模型。例如对于壁障，可以定义参数 Angle=30，则 LoCo 在组装的过程中，会自动修改壁障的角度。

Image	Name	Short description	Attributes	
			Angle	Hand Drive
	Suspension	steering_gear_for_RHD		rhd
	Suspension	steering_gear_for_LHD		lhd

图 8 通过属性“Hand Drive”来定义舵机

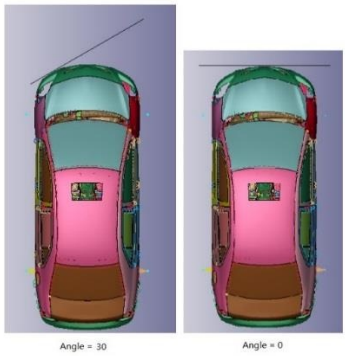

图 9 利用参数值“Angle”定义壁障角度

在使用 LoCo 进行组装的时候，会同时运行 Python 脚本。这些脚本可以控制前后处理，包括前处理的模型检查，送到服务器计算，后处理的数据分析、报告生成等一系列流程。在这里，LoCo 中导入了一个运行 LS-DYNA 进行仿真计算的脚本，所以在组装完模型后，会自动将模型送到服务器进行计算。仿真结果如下。

图 10 Yaris 正碰仿真结果（56km/h）

3 结果与分析

根据 LS-DYNA 的特点和独特算法，LoCo 可以很方便的进行模型组装。LoCo 作为仿真数据管理系统（SDM），不仅可以对 CAE 仿真数据进行管理，而且可以根据 CAE 部门的流程将前后处理都囊括到系统中，所以 LoCo 不只是一个仿真数据管理系统（SDM，Simulation Data Management），也是一个流程管理系统（PDM，Process Management），对包括 LS-DYNA、Pamcrash 等在内的求解器都有很好的支持。