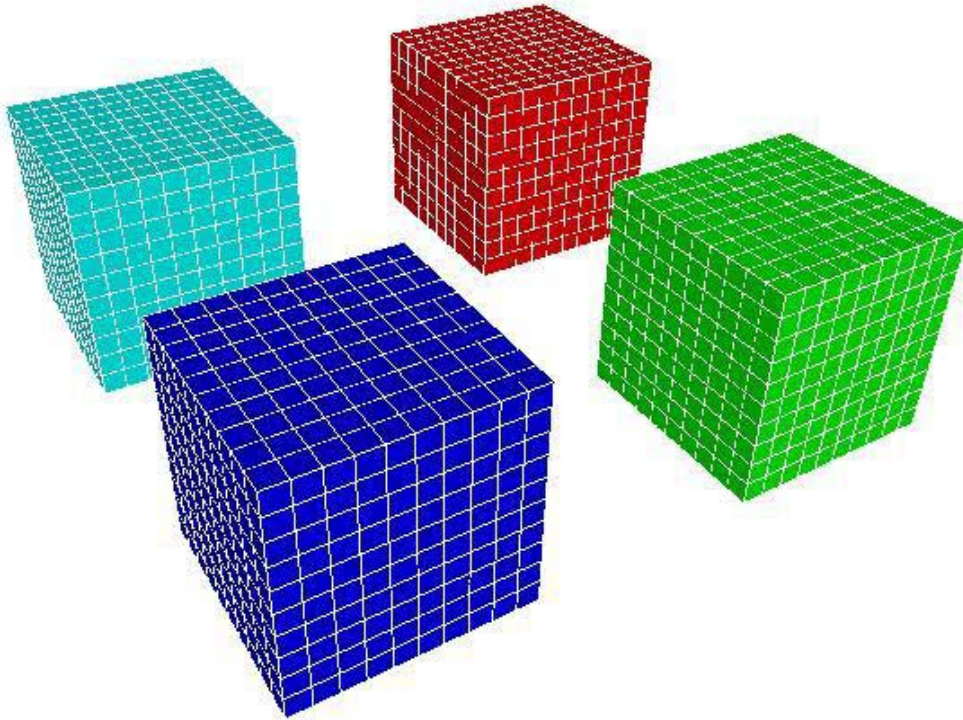


AUTODYN[®]

Explicit Software for Nonlinear Dynamics



Parallel Processing Tutorial

Version 6.1



CENTURY DYNAMICS
Solutions through Software

www.century-dynamics.com

AUTODYN is a trademark of Century Dynamics, Inc.

© Copyright 2005 Century Dynamics Inc. All Rights Reserved

Century Dynamics is a subsidiary of ANSYS Inc, www.ansys.com



Century Dynamics Incorporated
1001 Galaxy Way
Suite 325
Concord
CA 94520
U.S.A.
Tel: +1 925 771 2300
Fax: +1 925 771 2309
E-mail: customer@centdyn.com

Century Dynamics Limited
Dynamics House
Hurst Road
Horsham
West Sussex, RH12 2DT
England
Tel: +44 (0) 1403 270066
Fax: +44 (0) 1403 270099
E-mail: all@centdyn.demon.co.uk

Century Dynamics
16350 Park Ten Place
Houston
TX 77084
USA
Tel: +1 281 398 6113
Fax: +1 281 398 6061
E-mail: cdhouston@centdyn.com

Century Dynamics
Röntgenlaan 15
2719 DX Zoetermeer
The Netherlands
Tel: +31 79 36 20400
Fax: +31 79 36 30705
E-mail: sales@centurydynamics.nl

autodyn.support@century-dynamics.com

www.century-dynamics.com

www.ansys.com

PREFACE	1
Chapter 1. Introduction	3
Chapter 2. Parallelization of Grid and SPH Calculations and Contact Interactions	5
Chapter 3. Establishing a Parallel Processing Environment	9
Chapter 4. Creating Host Configuration Sets	19
Chapter 5. Creating Domain Decomposition Sets	27
Chapter 6. Benchmarking Calculations	61

Preface

AUTODYN Tutorial Manuals

AUTODYN tutorial manuals provide detailed tuition on particular features available in the program. The manuals assume that you are proficient in setting up, reviewing, executing, and post processing data for simple problems such as those presented in the AUTODYN-2D or AUTODYN-3D demonstration manuals. If you have worked through the problems in the demonstration manual, you should have no difficulty following the tutorials.

Most tutorials are interactive and you are expected to have access to AUTODYN while working through the tutorial. Some tutorials have associated files, which contain sample calculations used in the tutorial.

Existing manuals are continuously updated and new manuals are created as the need arises, so you should contact Century Dynamics if you cannot find the information that you need.

Chapter 1. Introduction

AUTODYN is a general-purpose computer code that has been developed specifically for analyzing non-linear, dynamic events such as impacts and blast loading of structures and components. The program offers users a variety of numerical techniques with which to solve their problems. These include Lagrange, Shell, Euler, ALE (Arbitrary Lagrange Euler) and SPH (Smooth Particle Hydrodynamics) solvers. As reliance on computational simulations becomes accepted, the complexity of the problems to be solved increases in size and resolution. However, the practical computation of these very large simulations has been restrained by the lack of performance of available computers. Problems requiring millions of elements and run-times that can run many weeks are not uncommon. Even the fastest single CPUs cannot easily cope with these larger problems. One approach to overcoming these limitations is to utilize parallel systems. Parallel algorithms have been implemented in AUTODYN to take advantage of parallel systems that allow simultaneous use of multiple CPUs either on a single multi-processor machine or over a distributed network of computers. This tutorial describes the method used by AUTODYN to process problems in parallel and explains how the user sets up and runs a calculation using parallel processing.

Parallelization work on AUTODYN has focused on two specific areas at present.

- Parallelization of the grid calculations for different solvers (including joins)
- Parallelization of contact interaction used to compute impact and sliding between two bodies (e.g. a projectile penetrating a target) or self-impact of a single body (e.g. a crushing cylinder).

I/O and plotting have not yet been parallelized. Users set up problems for parallel processing in exactly the same way as they do for serial processing, and the processing of results (plotting, saving, etc.) are also performed in the usual way.

At the current time, the grid calculations of the following solvers have been parallelized:

- Lagrange (with joins)
- ALE (with joins)
- Shell (with joins)
- SPH (with joins)
- Euler-Godunov

Interaction calculations have been parallelized between Lagrange, ALE, Shell and SPH solvers.

Parallelization of the grid calculations for the remaining solvers is currently an active area of development.

The basic approach used is described in the following chapter.

Chapter 2. Parallelization of Grid, SPH and Unstructured Calculations and Contact Interactions

§1. Parallelization of Grid Calculations

Domain Decomposition is used to parallelize the grid computations performed by all solvers which utilize a grid or mesh. Using this method, each subgrid is divided along index planes in the I, J and K directions (J and K directions for Shell subgrids) to form smaller grids called sub-domains. At the present time, users must define the domain decomposition to be used for each problem they wish to run in parallel.

These sub-domains are distributed amongst the CPUs of the parallel machine using an algorithm that attempts to minimize inter-CPU communications and balance the computational load on each machine. Each sub-domain is processed in parallel as if it were a standard subgrid in serial processing. This importantly allows most of the source code for serial processing to be used without modification.

Because the grid structure does not normally change during simulations, a static decomposition of the entire index space is usually sufficient to achieve good parallel performance.

Efficient parallel processing of the grid calculations requires not only good load-balancing of the computation, but also the efficient exchange of data at sub-domain boundaries. This is best achieved by users clearly understanding the process involved and choosing their domain decompositions to suit the specific host configuration they intend to use.

§2. Parallelization of SPH calculations

One of the main benefits of the SPH solver is its ability to evaluate the governing variables of a nonlinear dynamic analysis without utilizing a mesh or grid structure, thus allowing for very high levels of deformation. This benefit means that the domain cannot be decomposed by choosing grid lines and intersections as occurs with grid calculations.

In order to decompose the SPH calculation we utilize the virtual work units which are set up to facilitate the searching algorithm which is integral to the SPH solver. Each work unit is a cube and together they encompass the entire computational domain with a grid structure. The number of SPH nodes within a work unit is evaluated and a similar algorithm to that used to decompose the grid calculations is used to decompose the work units and therefore SPH nodes in order to minimize inter-processor communication. At present this is a static decomposition; each SPH node stays on the same processor for the length of the calculation.

§3. Parallelization of Unstructured calculations

As the numbering of nodes/elements of the unstructured solvers do not adhere to the same restrictions as the structured solvers, the parts cannot be simply decomposed by I, J, K index lines. The decomposition is done in a similar manner to SPH calculations. Virtual work

units are applied to the model and the work done in each work unit is evaluated. A recursive bi-section method is then applied in order to effectively load balance the model. The amount of communication between processors is minimized but this is a secondary consideration compared to the load balancing. This method is completely automatic and requires a minimal amount of information from the user.

§4. Parallelization of Contact Interactions

To understand how contact interactions are parallelized, we must first understand how the algorithm works in serial calculations.

Contact logic is used when one surface element of a subgrid attempts to penetrate another surface element of the same or a different subgrid. This requires a global search of Cartesian space to find possible contacts. Using the same sub-domain decomposition used for the grid calculation is not an efficient way to parallelize contact interactions. This is because the contact surfaces that need to be processed come from elements that lie on the surface of a subgrid and thus comprise only a subset of the total elements in the subgrid. It is easy to see that some sub-domains might contain many surface elements, while others none at all. Moreover, if a calculation allows the erosion or removal of elements as penetration occurs, the actual surface elements of a subgrid will change as the calculation proceeds. We are therefore forced to use a second, dynamic domain decomposition of Cartesian space for the contact calculations.

Generally, any two surface elements anywhere in the simulation can come in contact with each other during some time step, even those that belong to the same object (self-interaction). Checking for all such contacts requires a global search in Cartesian space that in practice can take up to 50% of the overall CPU time. For efficiency, the contact nodes and faces are spatially sorted to speed this computation and to avoid unnecessary tests of distant elements. Thus, the contact algorithm used in AUTODYN can be considered in two parts. Firstly, a calculation is performed to identify neighboring nodes/faces that require to be checked for interaction. Secondly, a detailed interaction calculation is performed for all these identified nodes/faces.

Determining which nodes/faces require to be checked for interactions is achieved with a bucket-sort. A grid of virtual work units is defined in Cartesian space. Each work unit is a cube, with sides twice the smallest face dimension of all interacting faces. In tests, this cube size was found to not only yield the most efficient computing times (due to the fine sort), but also to generate sufficient work units to allow efficient load-balancing for the parallelization. These work units are virtual because storage for a particular work unit is only allocated when it is determined that the work unit contains nodes/faces that are to be tested for interaction.

The bucket-sort loops over all the surface nodes and faces of a problem, and constructs a list of the actual work units required for a particular analysis. The sort is performed in two passes, in which all the nodes are sorted, and then the faces are sorted. First, each surface node is added to the work unit, which contains it. Next, each surface face is added to all work units, which contain nodes that might interact with the face. This is determined by checking each node of the face to see if it is contained within, or is in close proximity to, a work unit's domain. At this stage, only work units that already contain surface nodes are considered. The proximity test is based on the size of the contact detection zone used for the interaction logic and the amount of "slack" allowed to enable the calculations described here to be performed less frequently than every cycle.

Finally, the node and face tables built for each work unit in the list are examined to determine the total number of node/face interactions that will be required to be computed for the work unit (this number is used to facilitate load-balancing in the parallel version). In general, this will equal the total number of nodes in the work unit times the total number of faces. However, this can be reduced if, for example, self-interaction of a subgrid with itself is not permitted, or two subgrids have been specified not to interact with each other. If the total number of interactions required to be computed for a work unit is found to be zero, then the work unit is removed from the list.

At the end of this procedure a compact group of work units has been generated, each containing a list of surface nodes and faces that require testing for interaction. Each node has been uniquely assigned to a particular work unit. Faces have been assigned to multiple work units, as required. These lists may be valid for a number of computational cycles, depending on the proximity test used to determine potential node-face interactions and on changes in surface definitions (if an element is eroded or removed, surfaces need to be redefined).

Within each work unit, detailed interaction calculations are performed between the nodes and faces in each work unit list. The calculation is very robust in that every impact is detected and dealt with correctly regardless of the deformation and relative movement of bodies or changes in surface definitions

Parallelization of the contact algorithm described above is fairly straightforward. Once the work units containing the node and face lists to be tested for impact have been generated, a load-balancing algorithm efficiently distributes them amongst the available CPUs, assuming each CPU has either the same speed or a pre-determined relative speed provided by the user. The data decomposition used for the contact interaction calculation is different from the one used for the grid calculation, so the load-balancing algorithm attempts to minimize the inter-CPU communications required between these two decompositions. Although a static decomposition is used for the grid calculation, a dynamic decomposition has to be used for the contact calculation. Consequently, load balancing of the newly formed work units is performed for each cycle on which a sort is carried out. This allows contact calculations to remain well load-balanced even when surfaces are reconfigured as the simulation progresses, or during the erosion (removal) of elements.

Results have shown that the contact algorithm generates sufficient work units during the sort phase to allow efficient load balancing for parallel processing. Furthermore, the scheme uses simpler communication patterns than those that rely on recursive coordinate bisection (RCB) to assign equal amounts of nodes to all CPUs, and adapts well to heterogeneous systems where CPUs may have different speeds and workloads that may vary with time.

Chapter 3. Establishing a Parallel Processing Environment

AUTODYN has been designed for parallel processing on a variety of systems ranging from a Massively Parallel Processor (MPP) using shared memory to heterogeneous distributed networks of computers. In particular, AUTODYN has been ported and tested on the following configurations:

- Multi-processor unix workstations (Sun/Compaq/SGI/HP).
- Multi-processor Pentium PC's running Windows NT/2000/XP
- A network of up to four unix workstations (DEC Alpha), using 10 Mbps Ethernet
- A network of up to eight Pentium PC's running Windows 2000, using 100Mbps Ethernet
- A network of up to sixteen PC's running Red Hat Linux, using a 100 Mbps Ethernet
- A network of 4 AMD PCs running Windows XP using 1Gbps Ethernet

In this chapter, we outline the procedure for establishing a parallel processing environment for AUTODYN V6. The procedure should be similar for most platforms. Platform specific procedures for configuring the parallel processing environment are given in referenced appendices.

When using AUTODYN V6 for parallel processing, data must be exchanged between cooperating tasks, and some message passing protocol has to be used to achieve this. We currently use PVM (Parallel Virtual Machine) or MPI(Message Passing Interface) to allow a heterogeneous collection of computers networked together to be viewed by AUTODYN as a single parallel computer. The component computers can be single or multiple processor machines, including MPPs (Massively Parallel Processors).

The PVM system is currently the default option for running under UNIX/Linux and was the system originally used in the development of the parallel processing capability in AUTODYN V6.

The MPI system is currently the default option for running under the Windows NT/2000/XP environment.

It is likely that in future we will only use one message passing tool on all platforms.

If you wish to use the parallel processing options described in this tutorial, you must install the message passing protocol on all the machines you intend to use. PVM is downloadable from the web for most Unix platforms. Details on how to obtain and install MPI for Windows NT/2000/XP are given in the following chapter, please note that MPI must be obtained from a specific supplier in order for it to work with AUTODYN.

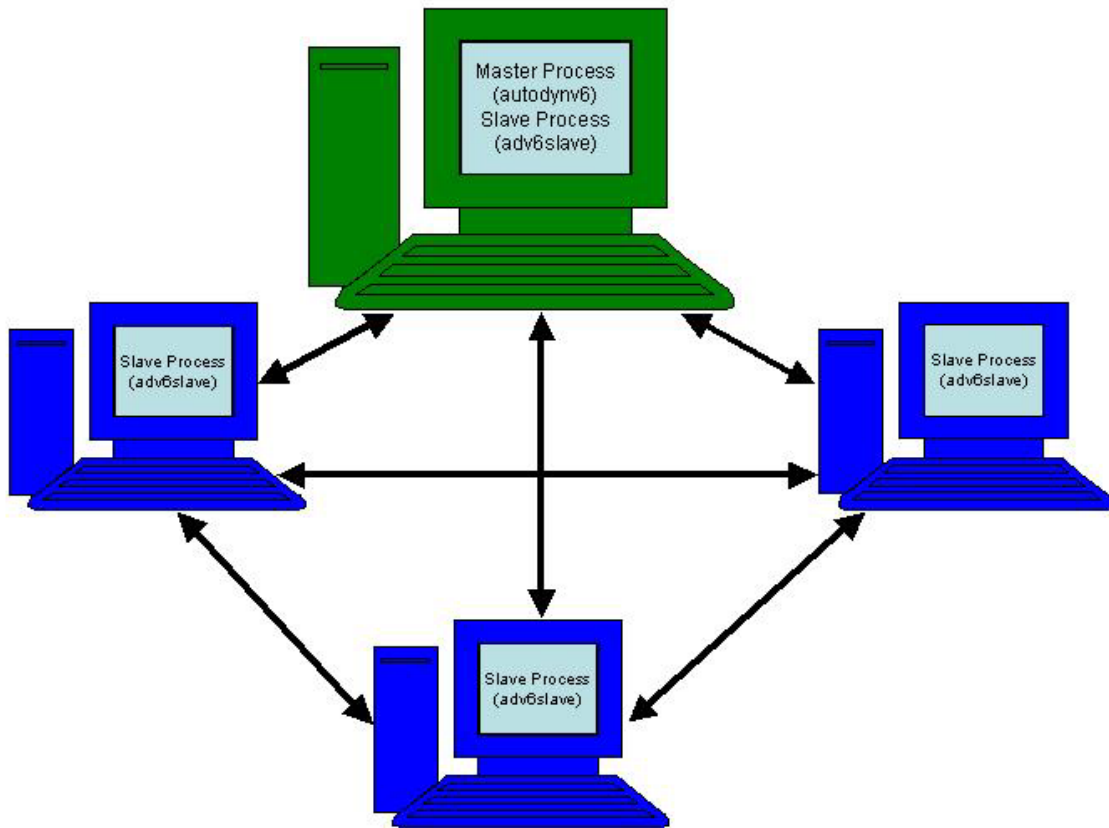


Figure 1: Schematic of the Parallel Processing Environment

Whether you intend to run a calculation in serial mode or parallel mode, AUTODYN V6 is always started by activating a single process on one machine. This is done by executing the file **autodyn6.exe**.

If a serial calculation is being performed, this is the only process that is run. It performs all aspects of the problem (setup, execution, I/O, post processing etc.)

If a parallel calculation is being performed, this process is considered to be the **master process** and additional **slave processes** are started on each CPU used by the parallel system (including the CPU running the master process). These slave processes are spawned automatically by the master process, which issues the required system commands to execute the file **adv6slave.exe** once on each CPU. This parallel processing environment is shown schematically in figure 1.

In what follows, we assume that the configuration we are implementing comprises of **n** hosts, each having **m** CPU's, and that the hostname of the nth host is **hostn**. **Host1** is special, in that it runs the master process (**autodyn6**) and is usually the system on which the program files are stored.

When running AUTODYN V6 in parallel on a system of networked computers, the program executables and libraries must be accessible on each of the computers. They can either be

installed on all computers in the system or installed on one computer (usually the one running the master process) and shared with the others using NFS mounting or Windows Sharing. We recommend sharing as it simplifies the handling of the files. Setup is further simplified if the shared mount points/folders are the same on each host (including the master).

§1. Windows NT/2000/XP Systems using MPI

Before running parallel calculations with AUTODYN under Windows NT/2000/XP you must obtain a license for MPI. Further, AUTODYN is only currently supported for the WMPI message passing libraries from Critical Software.

In order to obtain a license please email: wmpi-sales@criticalsoftware.com stating that you require WMPI version 1.6 for use with the AUTODYN software from Century Dynamics.

In what follows, we assume that AUTODYN is installed in the directory `c:\autodyn\adv61` and WMPI is installed on all host machines in a directory similar to the default i.e. `c:\Program Files\WMPI`.

The first step is to share the executable directory (`c:\autodyn\adv61`) with all other hosts. Using Windows explorer, highlight this folder and right click, select *Sharing*. Now enter a share name for this folder (for example `adv61`).

Now that we have shared this folder, we can tell MPI to use the executable [\\host1\adv61\adv6slave.exe](#) irrespective of which host(s) we are asking MPI to run a slave task on.

The environment settings outlined in the next four sub-sections must be defined before any parallel computations can be performed. To parallel process a problem within this environment, information must be supplied to define :

1. The host machines to be used for the calculation
2. The Domain decomposition to be applied to each subgrid in the problem

Input of these data is described in the next two chapters.

§1. -1 SYSTEM ENVIRONMENT VARIABLES

In order for each host to communicate using MPI we must set a number of environment variables on all host machines in the parallel cluster:

Firstly, go to *Start, Settings, Control Panel* and select *System, Advanced*.

Select *Environment Variables* and add the following to the *System Variables* panel

```
WMPI_MASTER_ERROR_OUTPUT = filename1  
WMPI_SLAVE_ERROR_OUTPUT = filename2
```

WMPI_MASTER_ERROR_OUTPUT specifies a location into which WMPI errors are reported. *Filename1* and *filename2* should therefore denote a file into which the WMPI errors

Establishing a Parallel Processing Environment



will be written, for example C:/wmpi_master_error.txt and C:/wmpi_slave_error.txt. If a parallel calculation fails to proceed and causes AUTODYN v6 to prematurely close, this file should be viewed and the errors acted upon.

If you are working on a computer with Windows NT/2000 installed, go to *Control Panel, Administrative Tools, Local Security Policy*. Select the folder *Local Policies* and then *User Rights Assignment*.

The account that is executing the WMPI Daemon or service must have the following user rights:

- Act as part of the operating system
- Increase Quotas
- Replace a process level token
- Bypass traverse checking

Alternatively if the computer has Windows XP installed, go to *Control Panel, Administrative Tools, Local Security Policy*. Select the folder *Computer Configuration, Windows Settings, Security Settings, Local Policies, User Assignment Rights*.

The account that is executing the WMPI Daemon or service must have the following user rights:

- Act as part of the operating system
- Adjust memory quotas for a process
- Replace a process level token
- Bypass traverse checking

Note that you will have to restart your computer for any changes you make to take effect.

§1. -2 WMPI SERVICES/DAEMON

On each host in the parallel cluster, you must ensure that the WMPI services have the appropriate status. Go to *Control Panel, Administrative Tools, Services*

If using WMPI 1.5.8 there will be two services running WMPI Service and WMPI TCP Service. If you wish to run the TCP daemon instead of the TCP service then stop the WMPI TCP Service and start the WMPI TCP daemon in C:\Program Files\Critical Software\WMP\bin.

Note you can use the Hide button to make the Daemon into an icon in the toolbar at the bottom right of the screen.

If using the daemon instead of the service you may find it useful to paste a shortcut to the wmpitcpdaemon.exe file in

C:\Documents and Settings\All Users\Start Menu\Programs\Startup.

This will automatically start-up the daemon whenever the system is restarted.

If using WMPI 1.6 only a WMPI Service will be running. If you wish to use the daemon instead of the service then stop the WMPI Service and run the wmpi daemon in C:\Program Files\WMP\bin. The main advantage of the service is that there is not a necessity for a user to be logged into the cluster machines in order to run the parallel computations.

§1. -3 MPI CLUSTER CONFIGURATION

MPI needs to know which machines (hosts) may be used in a given computation, the communication devices that can be used to exchange data between processes and the security context of processes for each machine.

A file named wmpi.clusterconf needs to be placed in the directory of the master process (host1) and contains information about all the machines in the cluster, though each computation might use just a subset of them. Detailed information about the format of this file can be found in the WMPI electronic documentation. For use with AUTODYN, the following example should be sufficient to enable you to get started.

On installing AUTODYN, a general wmpi.clusterconf template file will be placed in the /bin directory. To avoid overwriting existing files, this will be named wmpi.clusterconf_temp.

The contents of this file will be

```
/Machines
.
startup address .
shmem .
tcp .
# Enter remote machine details below
# <Windows Machine Name>
# startup address <Windows Machine Name>
# tcp <Windows Machine Name>
/Connections
internal device shmem
external device tcp
# Enter remote machine details below
# <Windows Machine Name> <Windows Machine Name> tcp
/Security
default user .
default domain .
# Enter local and remote login details. Note that the Domain
# can be the same as the Windows Machine Name
# <Windows machine Name> <User Name> <Domain>
```

You will need to customize this file for your specific parallel cluster and rename it `wmpi.clusterconf`. The example below is for a cluster of three machines with network names `host1`, `host2` and `host3`. The master process is running on machine `host1`.

```
/Machines
.
startup address .
shmem .
tcp .
# Enter remote machine details below
# <windows Machine Name>
# startup address <windows Machine Name>
# tcp <windows Machine Name>
host2
startup address host2
tcp host2
host3
startup address host3
tcp host3
/Connections
internal device shmem
external device tcp
# Enter remote machine details below
# <windows Machine Name> <windows Machine Name> tcp
host2 host2 tcp
host3 host3 tcp
/Security
default user .
default domain .
# Enter local and remote login details. Note that the Domain
# can be the same as the windows Machine Name
# <windows machine Name> <User Name> <Domain>
host1 administrator host1
host2 administrator host2
host3 administrator host3
```

If all of the available computers are part of the same network domain, then this domain can be entered within the `wmpi.clusterconf` file.

```
/Machines
.
startup address .
shmem .
tcp .
# Enter remote machine details below
# <windows Machine Name>
# startup address <windows Machine Name>
# tcp <windows Machine Name>
host2
startup address host2
tcp host2
host3
startup address host3
tcp host3
/Connections
internal device shmem
external device tcp
# Enter remote machine details below
# <windows Machine Name> <windows Machine Name> tcp
host2 host2 tcp
host3 host3 tcp
/Security
default user .
default domain .
# Enter local and remote login details. Note that the Domain
# can be the same as the windows Machine Name
# <windows machine Name> <User Name> <Domain>
host1 administrator office
host2 administrator office
host3 administrator office |
```

If this setup is applied, Security information is required to identify the Windows login information that WMPI will need to use. In the example above, we are logging on as administrator into a network domain called "office". The remote hosts that you will be using for the parallel computation must know the password for the user name that you are running the AUTODYN tasks under. You must therefore register each user in the WMPI software following the procedure outlined in the next section.

§1. -4 MPI USER REGISTRATION

To avoid entering passwords every time you run AUTODYN in parallel, you can register each user name and domain by executing the program UserRegister. To execute this program go to the *Start Menu* and navigate to *Programs, WMPI* and select UserRegister. The following would register the security information for administrator in the example above.

User name	administrator
User domain	office
User password	*****
Verify password	*****

§2. Unix/Linux Systems using PVM

In what follows, we assume that AUTODYN and PVM files are installed on host1, in a directory named **/home/host1**, and that this directory is NFS mounted on all other hosts (with the same mount point). To NFS mount this directory, add the following line to the file **/etc/fstab** on each host (except **host1**):

```
host1:/home/host1 /home/host1 nfs defaults 0 0
```

/etc/fstab may have a different name for other Unix systems (on Solaris this file is **/etc/vfstab**).

Following the normal installation procedures for AUTODYN products, all AUTODYN V6, PVM (if distributed) files will be installed in **/home/host1/autodyn**. In particular, the AUTODYN V6 master and slave executables (**autodynv6.exe** and **adv6slave.exe**) will reside in the directory **/home/host1/autodyn/adv61**

After installing AUTODYN, the following environment variables must be set for GKS and PVM, by adding the following lines to the user's **.cshrc** file (c-shell) :

```
setenv PVM_ROOT /home/host1/pvm3  
setenv PVM_ARCH LINUX  
setenv PVM_DPATH $PVM_ROOT/lib/pvmd  
setenv PATH "$PATH:$PVM_ROOT/bin/LINUX:$PVM_ROOT/lib"  
setenv MANPATH "/usr/man:/usr/local/man:/usr/X11R6/man:$PVM_ROOT/man"
```

To parallel process a problem within this environment, information must be supplied to define :-

1. The host machines to be used for the calculation
2. The Domain decomposition to be applied to each subgrid in the problem

Input of these data is described in the next two chapters

Chapter 4. Creating Host Configuration Sets

To run AUTODYN V6 in parallel, you must define the host configuration you wish to use. AUTODYN V6 allows you to define up to ten host configuration sets and select one of these sets as the active configuration.

Host configuration set data is saved in an external file called “**parallel.cfg**”, located in the executable directory of the master process (**autodyn6**). This is a text file that can be edited directly or created within AUTODYN V6.

Here is an example of the typical contents of this file.

1. **#@EPDEF=\\host1\adv61**
2. **#@PPDEF office**
3. **#@PPCFG office**
4. **host1 sp=1000**
5. **#@ mem=128 cpu=1 task=1**
6. **host2 sp=500**
7. **#@ mem=256 cpu=2 task=2**

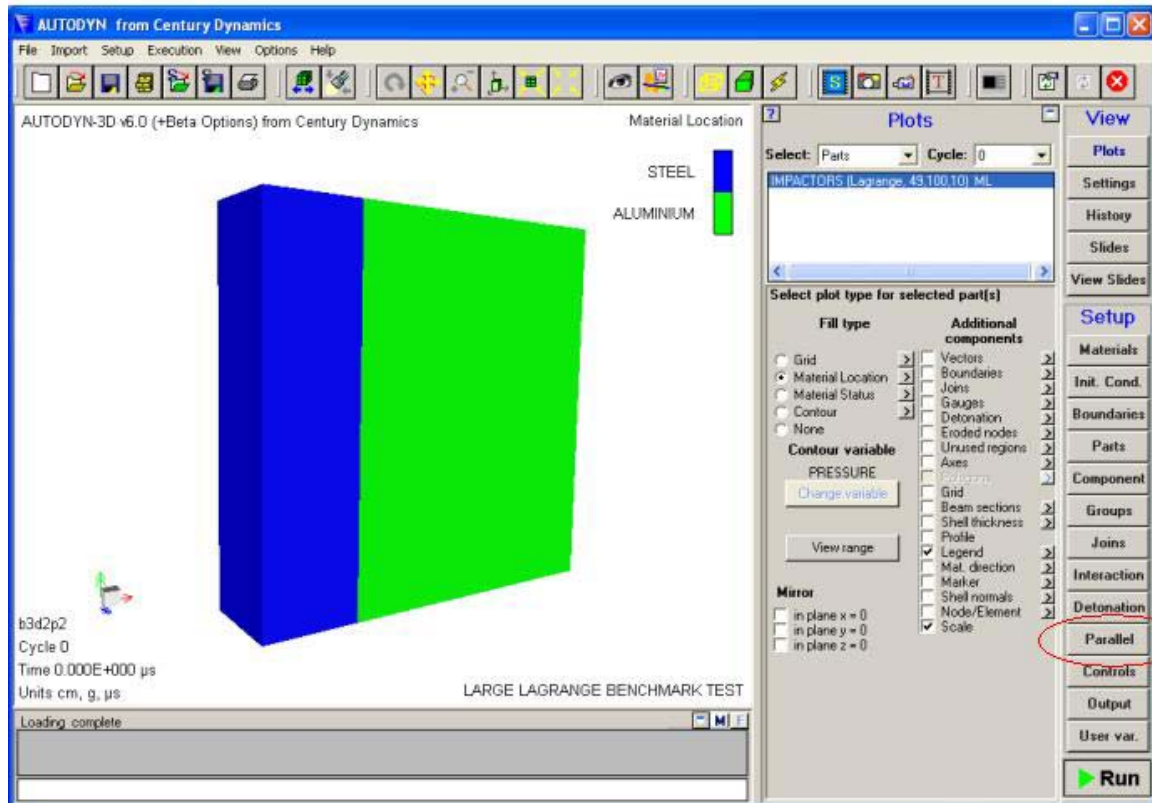
- Line 1: Defines the default path for the AUTODYN V6 executable to be **\\host1\adv61**. Where **adv61** is a shared directory on **host1**.
- Line 2: Defines the configuration set name that is to be used (**office**)
- Line 3: Indicates that the following lines, until the next **#@PPCFG** statement define configuration set **office**. (There is only one configuration set defined in this example)
- Line 4: Adds **host1** to configuration set “**office**”. It has a relative speed of **1000** and uses the default path for the AUTODYN V6 executable (since no **ep** parameter is defined)
- Line 5: **host1** has **128** Mb of memory, **1** CPU, and is to have **1** task (slave process) started on it.
- Line 6: Adds **host2** to configuration set “**office**”. It has a relative speed of **500** and **/autodyn6** as the path for the AUTODYN V6 executable.
- Line 7: **Host2** has **256** Mb of memory, **2** CPUs, and is to have **2** tasks (slave processes) started on it. (The operating system will automatically allocate 1 task to each CPU).

Whereas you can construct “**parallel.cfg**” in an editor, a more convenient approach is to define configuration sets within AUTODYN V6 and let the program write this file for you.

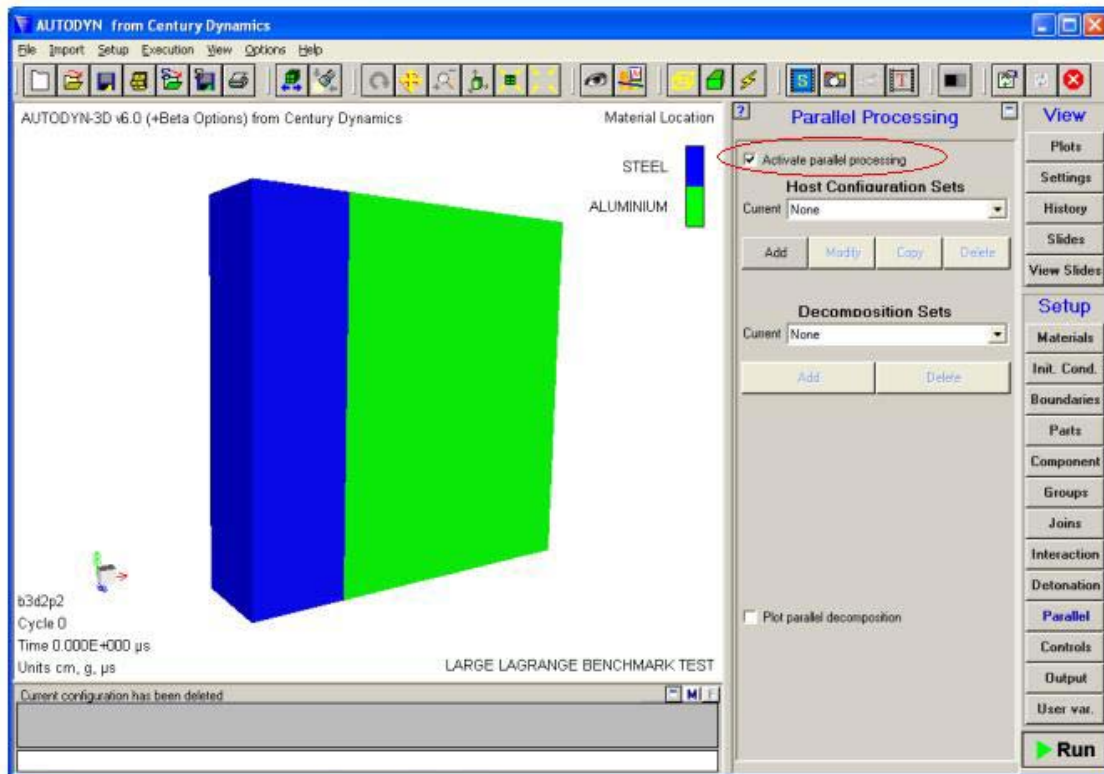
Creating Domain Decomposition Sets

First open an AUTODYN model. Load in the model or ident **B3D2P2** in the Samples3d directory of the AUTODYN installation. This model is a Lagrange impact benchmark testcase.

Input host configuration set data is written by selecting **Parallel** in the **Setup** toolbar:

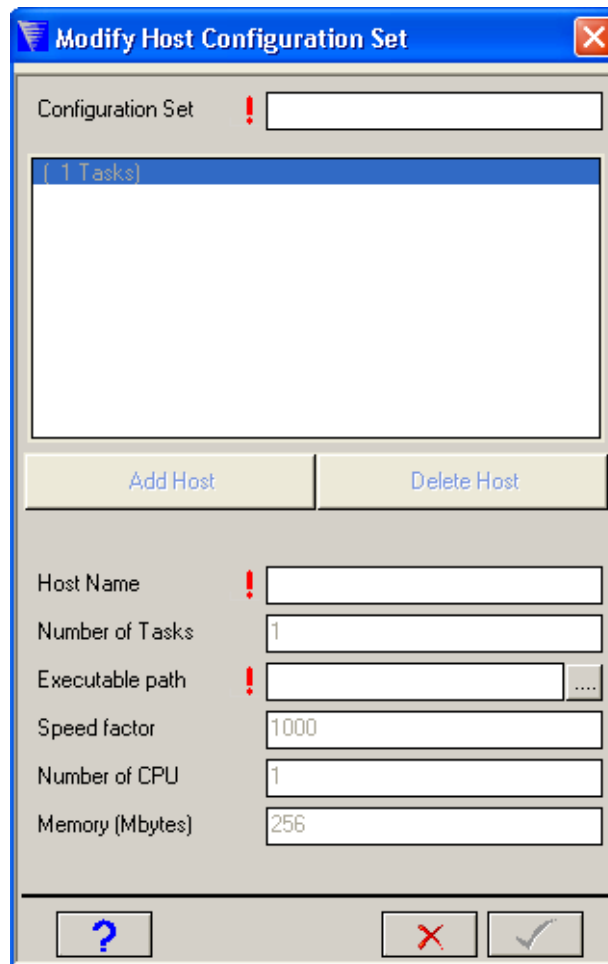


This action brings you to the following screen:



To activate parallel processing, tick the checkbox highlighted in the figure above. This will trigger the buttons within this panel that are used to define host configuration and decomposition sets.

Select **Add**, which will activate the popup window below:



When you transfer to this menu, any previously saved configuration sets will be automatically loaded and the currently selected host configuration set name will be displayed in the top right corner of the screen. You have not defined any configuration sets yet, so nothing is displayed.

Therefore, type a name for the host configuration set that you are going to define and press return.

Then complete the requested information in the dialogue boxes:

Host name: Network name of the computer that will process an AUTODYN task

Number of tasks: The number of slave processes that are to be run on the machine. This number will usually be equal to the number of CPUs the machine has. If it is set to zero, the machine will not be used. This offers you the option to specify a configuration containing all possible hosts and then turn on only those you wish to use for a particular session. On dedicated multiprocessor machines, if the number of tasks is set equal to the number of CPUs, the operating system will automatically assign one task to each CPU.

Executable path: Directory in which the AUTODYN executables are stored

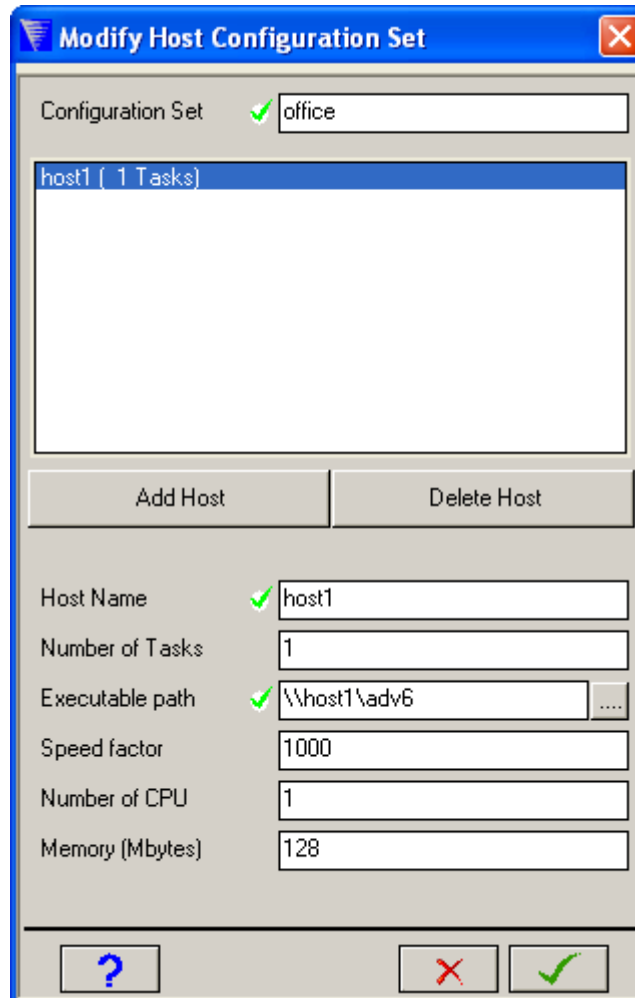
Speed Factor: This value relates to the clock speed of the computer's CPU

Number of CPUs: The number of CPUs contained within the computer

Memory (Mbytes): The amount of RAM accessed by the computer.

In the example below two hosts have been added, on which the AUTODYN tasks will be processed.

Enter the information for the first host:



The screenshot shows a dialog box titled "Modify Host Configuration Set". At the top, the "Configuration Set" is set to "office". Below this, a list contains "host1 (1 Tasks)". There are "Add Host" and "Delete Host" buttons. The host configuration details are as follows:

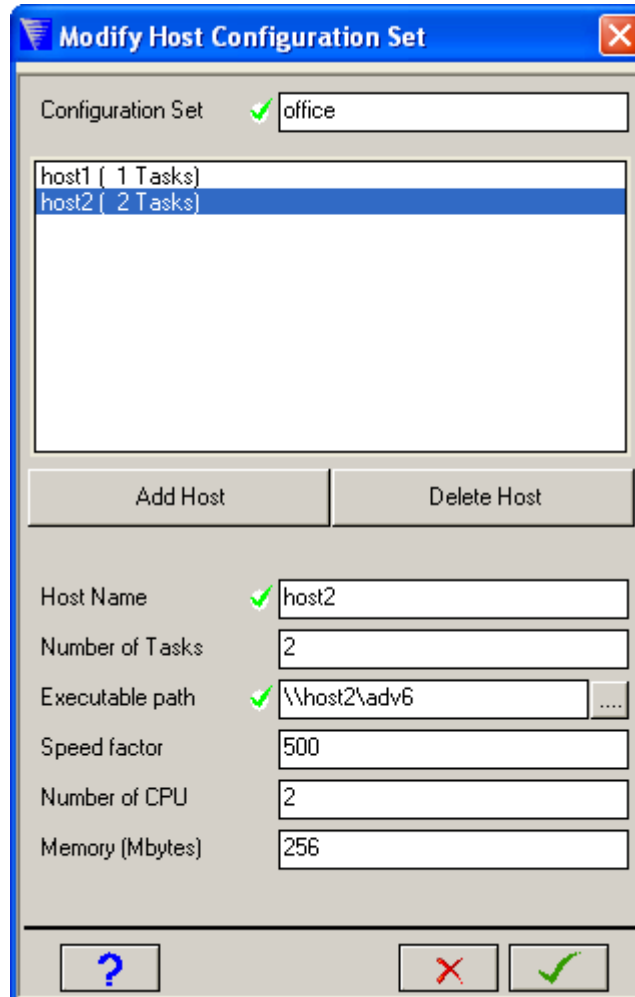
Host Name	✓	host1
Number of Tasks		1
Executable path	✓	\\host1\adv6
Speed factor		1000
Number of CPU		1
Memory (Mbytes)		128

At the bottom of the dialog are three buttons: a help button (question mark), a cancel button (red X), and an OK button (green checkmark).

The name of the host or computer is **host1**. This computer has 1 CPU and 128 Megabytes of RAM. As the computer has one CPU it is sensible to run one task on this computer. As the AUTODYN V6 executables are stored in a shared directory called **adv61** on **host1**, we can specify the executable path to be [\\host1\adv61](#). This executable path can be used for all machines within the configuration providing they have access to **host1**.

Creating Domain Decomposition Sets

After completing the information for the first host, press “**Add Host**” in order to clear the input values and enter new information regarding host number 2:



Modify Host Configuration Set

Configuration Set office

host1 (1 Tasks)
host2 (2 Tasks)

Add Host Delete Host

Host Name host2

Number of Tasks 2

Executable path \\host2\adv6

Speed factor 500

Number of CPU 2

Memory (Mbytes) 256

? X ✓

The second host, **host2**, is a dual processor machine and therefore 2 has been entered as the number of tasks to be run on this machine and as the number of available CPUs. The executable path is the same as that used for host1 as the directory **adv61** is shared. The speed of the CPUs on this machine is half as fast as the CPU on **host1**, correspondingly the **Speed factor** has been set to 500.

After the required information regarding hosts has been entered press the tick button. A parallel.cfg file will be created wherever the AUTODYN v6 executable resides. This file will contain all of the information that has been entered regarding configuration sets:

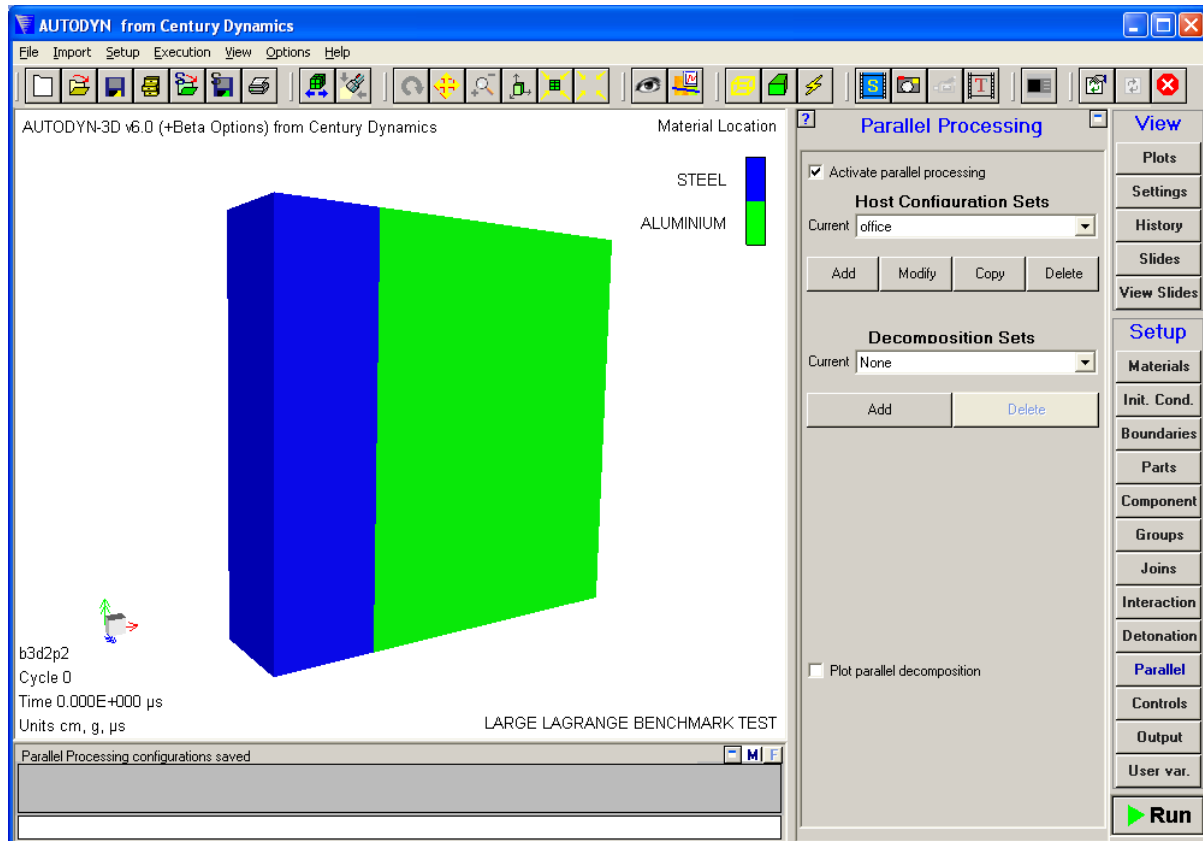
```
#@PPDEF office
#@PPCFG office
host1 sp=1000 ep=\\host1\adv6
#@ mem=128 cpu=1 task=1
host2 sp=500 ep=\\host2\adv6
#@ mem=256 cpu=2 task=2
```

Here there is no global definition of the executable path i.e. #@EPDEF. The executable paths are locally defined for each machine e.g ep = [\\host1\adv61](#).

parallel.cfg is an external file that is read whenever a problem is executed in AUTODYN V6. Host configuration sets are therefore not problem dependent (unlike the domain decomposition sets described in the next chapter). When you create or modify a host configuration set while working on a particular problem, that set becomes available for all problems.

Creating Domain Decomposition Sets

After specifying the host configuration, the name of the host configuration that we have specified will appear in the **Host Configuration Set** instead of the previously stated **None**:



If more than one configuration set had been defined we could scroll through the various sets using a drop down list accessed by pressing the list button to the right of the **Host Configuration Set** dialogue box.

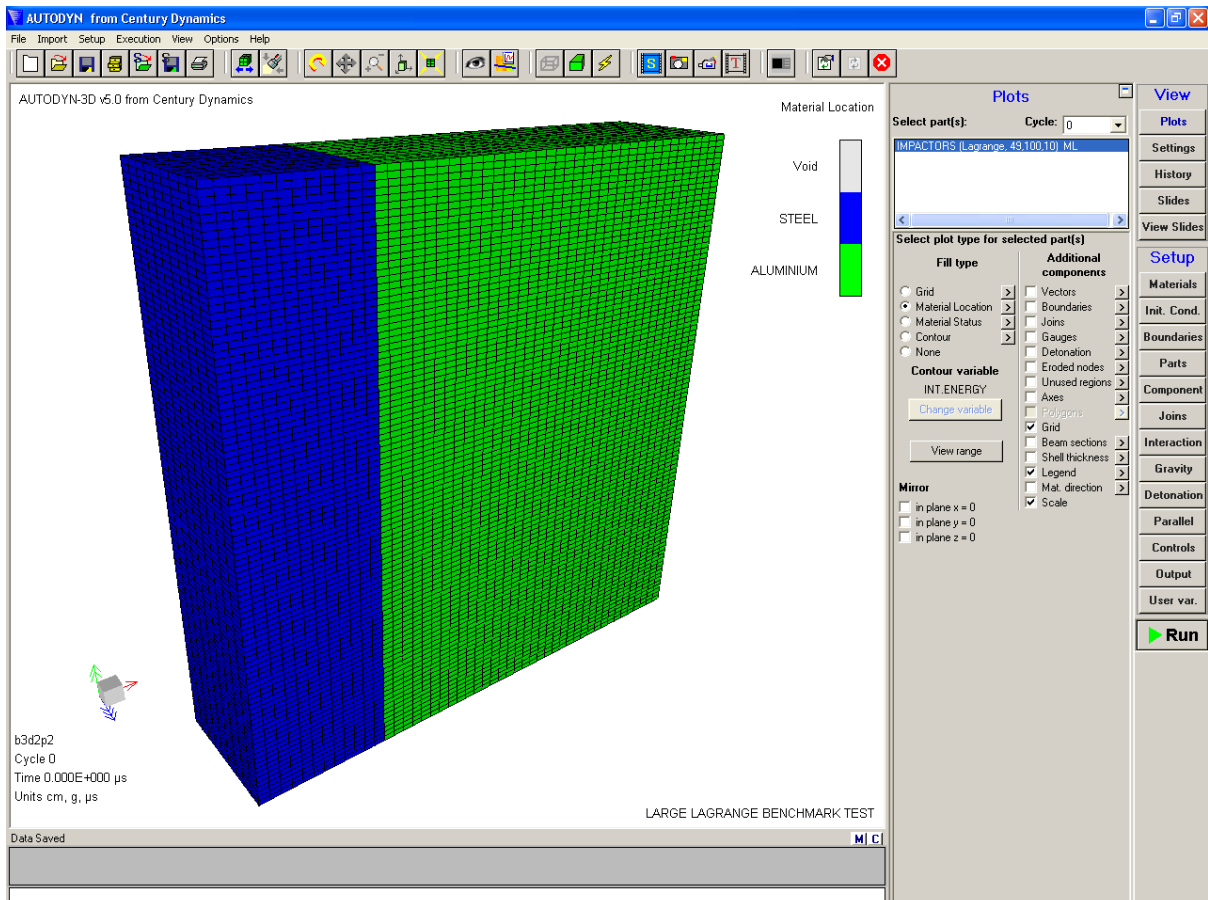
Chapter 5. Creating Domain Decomposition Sets

To enable parallel processing we must decompose the problem and assign the various sub-domains to the available processors.

Domain decompositions sets are associated with a particular model. When they are created, they are stored along with all other problem data whenever the model is saved.

§1. Decomposing Structured Calculations

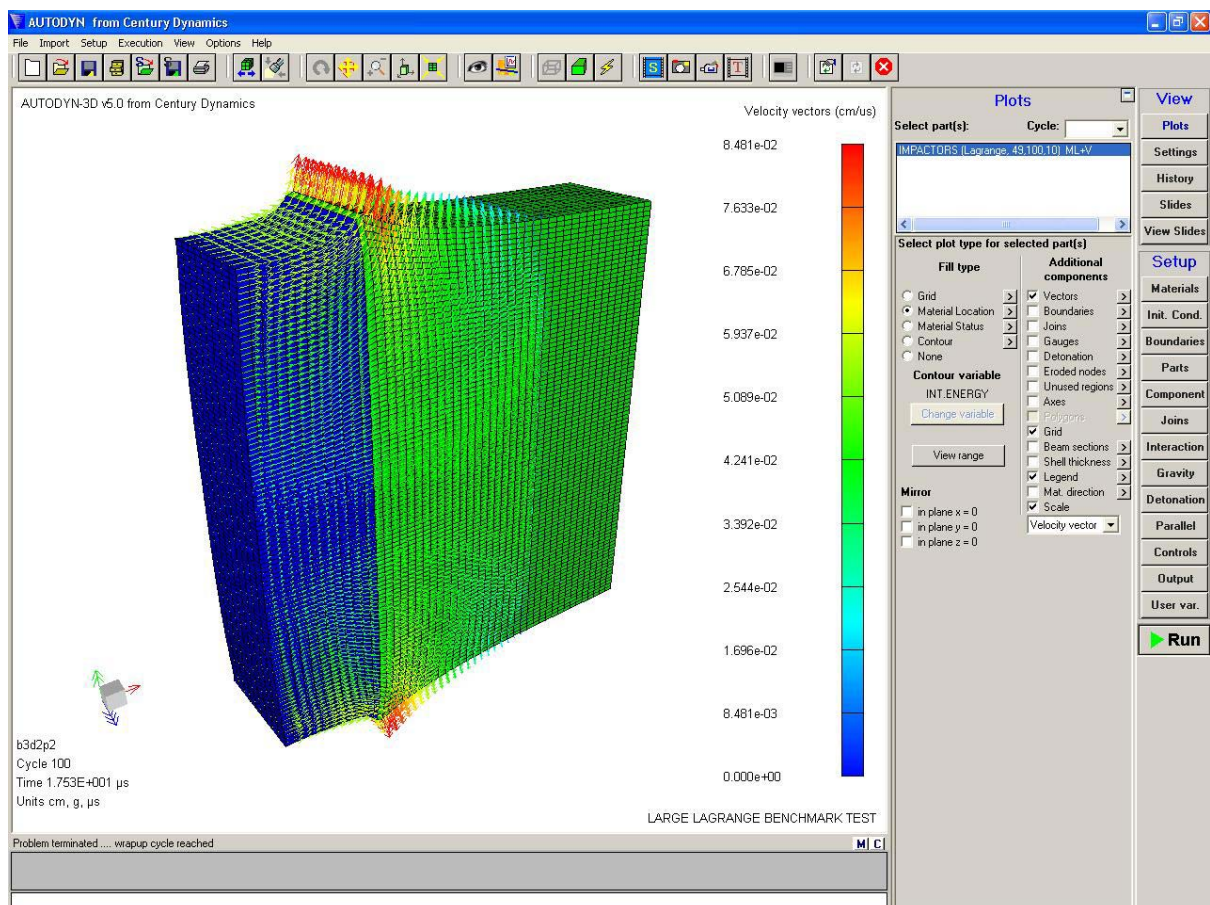
In the following example, you will create a domain decomposition set for the Lagrange impact benchmark calculation that you have already opened. Though this example specifically applies to a Lagrange process, the method shown is appropriate to decompose any grid calculation, including Euler processes.



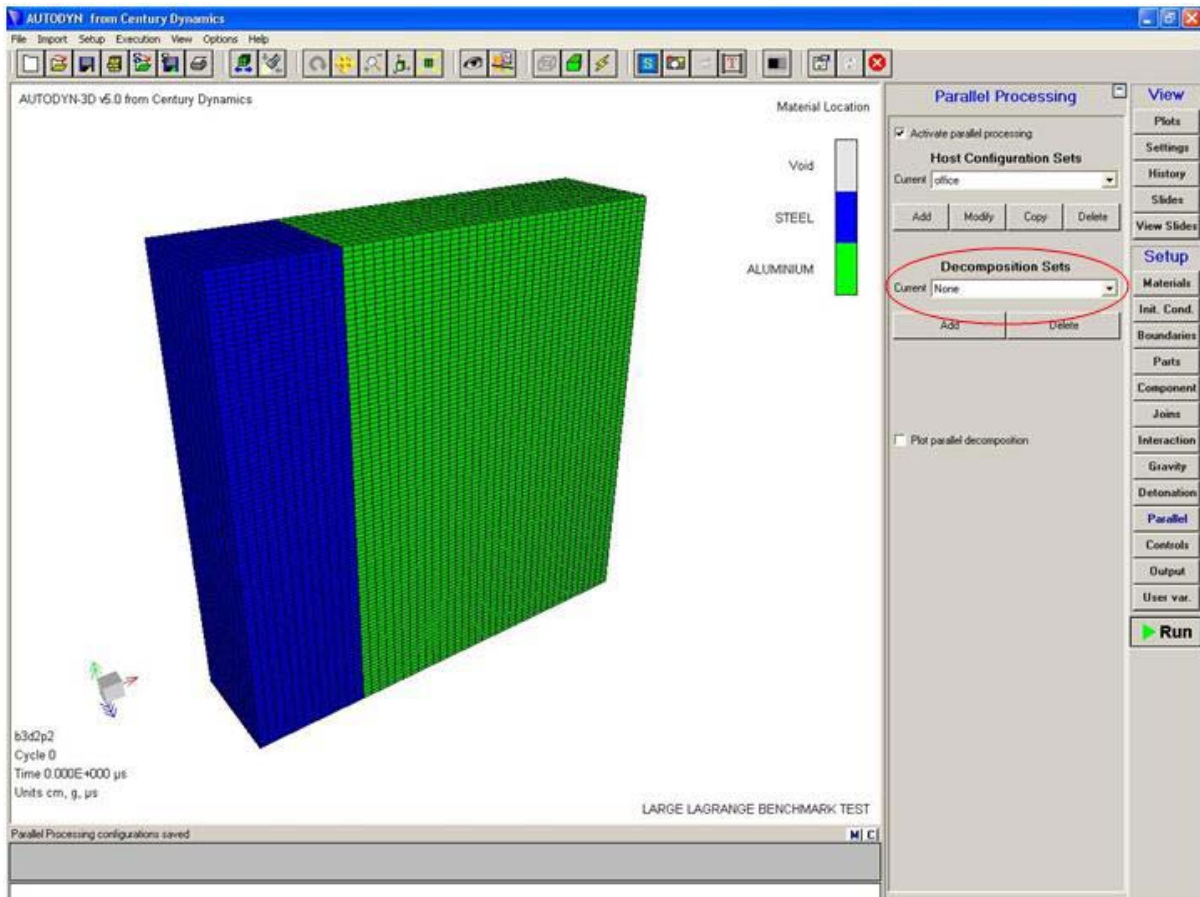
Creating Domain Decomposition Sets

The problem consists of a single Lagrange subgrid (49x100x10 cells) filled with two materials. The steel has an initial z-velocity of 0.06 cm / microsecond, while the aluminum is initially at rest.

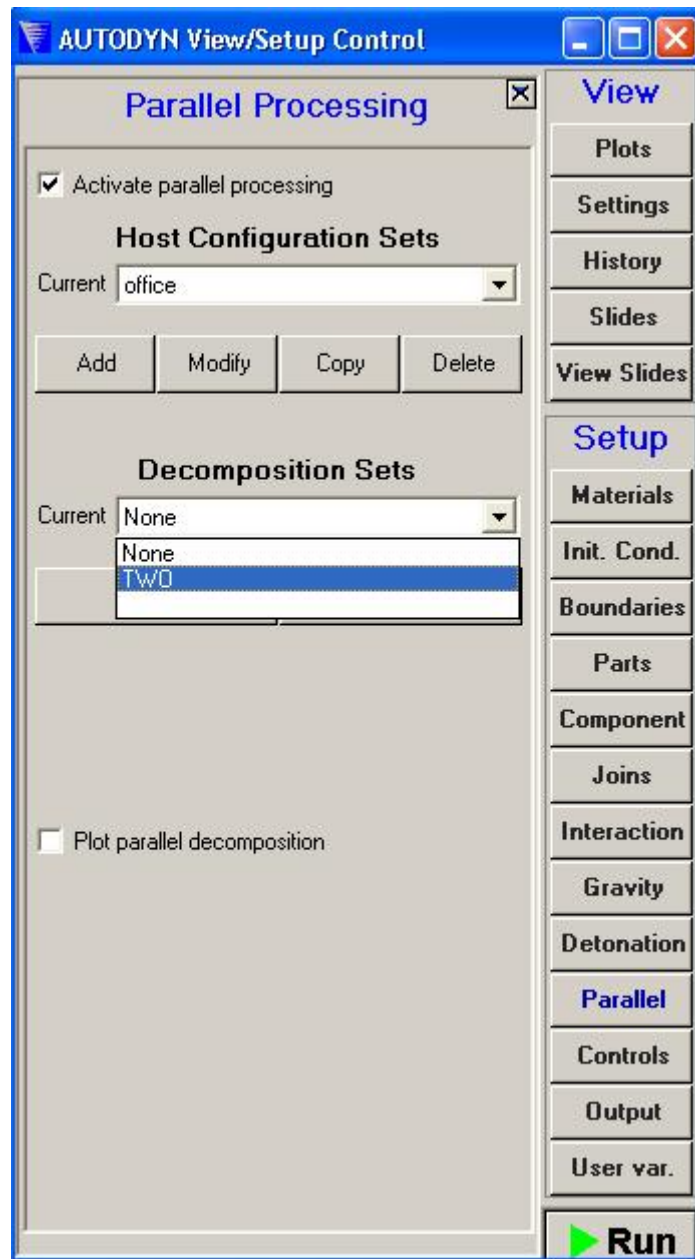
If you run this problem for 100 cycles and then view a velocity vector plot, you will see the following screen, showing the impact of the steel on the aluminum.



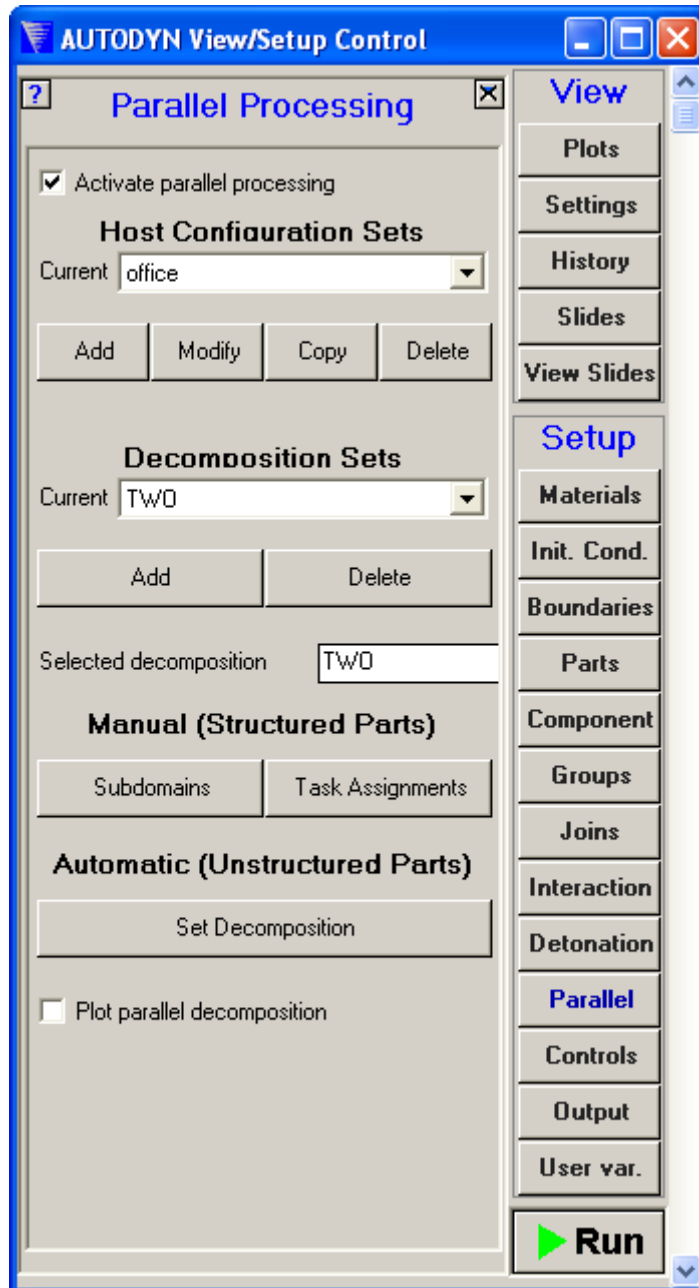
This model already contains **Decomposition Set** information. The current decomposition is shown in the **Decomposition Set** text box.



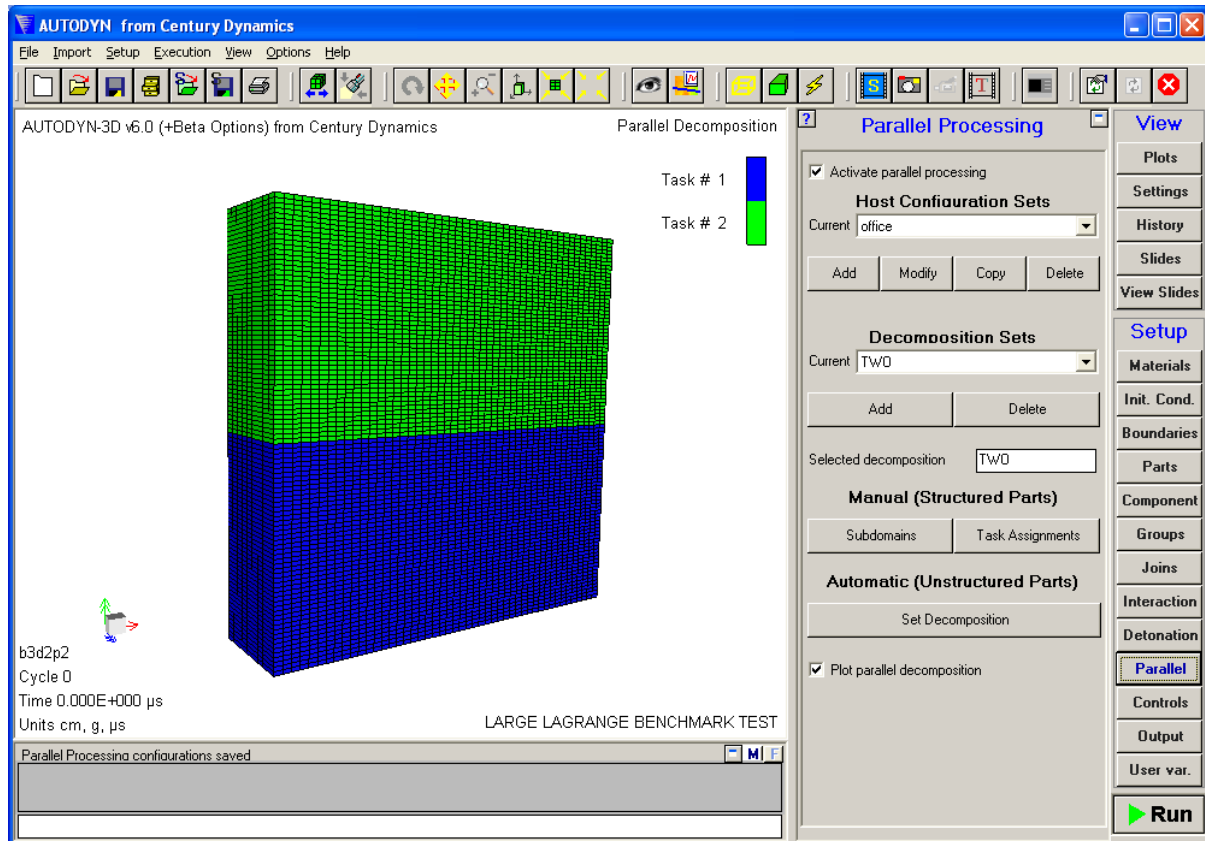
Presently the current decomposition set is set to “**None**”. The current decomposition set can be changed by using the drop down list activated by pressing the list button on the right of the text box. The image below shows a close up of the **Parallel** panel.



Choose the decomposition set denoted by “**TWO**”. Further buttons allowing the modification of the decomposition set will appear.



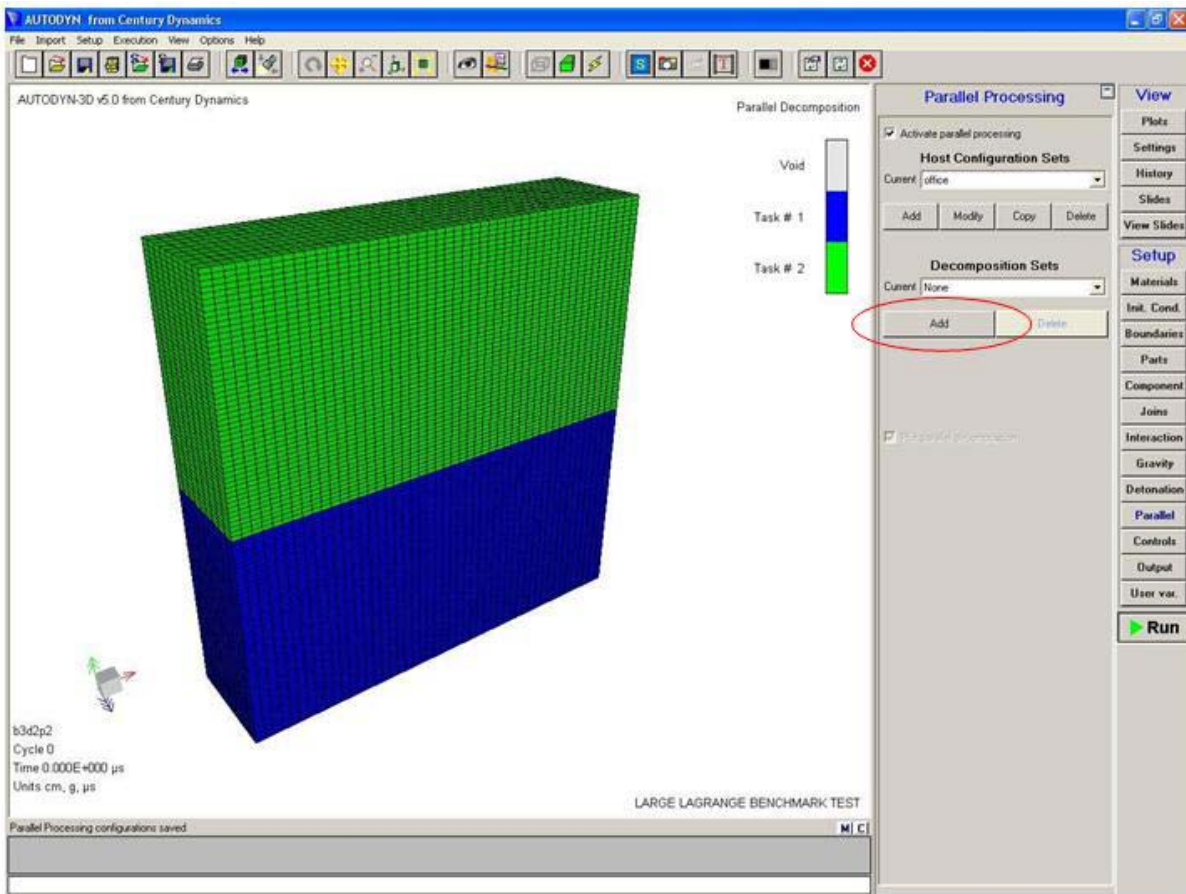
Check the tickbox marked **Plot parallel decomposition**. The decomposition of the model will now be displayed (you may need to manually refresh the screen if automatic refresh is not activated):



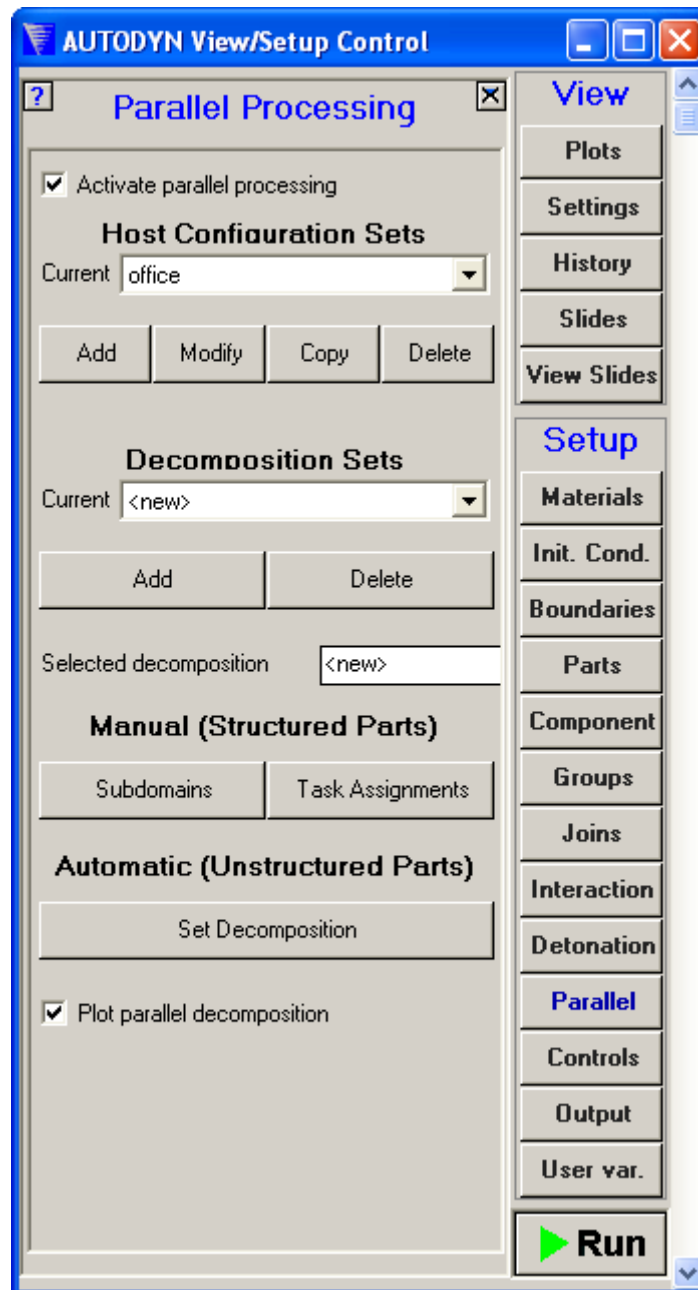
This image shows the decomposition of the model into its constitutive sub-domains. The scale illustrates which sub-domain is assigned to which task. Here the bottom sub-domain is assigned to task 1 and the top sub-domain assigned to task 2. This is the optimum decomposition for efficient parallel computing over two tasks. The model has been decomposed in the J direction. As the J direction contains the most elements, this decomposition will result in the smallest amount of information being passed between tasks. There are 490 elements on each sub-domain boundary; 49 in the I direction and 10 in the K direction. If we had decomposed the subgrid in the I direction, there would have been 1000 elements on each sub-domain boundary; 100 in the J direction and 10 in the K direction. Obviously there is a computational overhead associated with the amount of information passed between tasks and therefore each element on the sub-domain boundaries increases the amount of communicated information and therefore computational expense.

Before proceeding press the **Delete** button to discard the “**TWO**” decomposition. The instructions to enable you to replace it are detailed below.

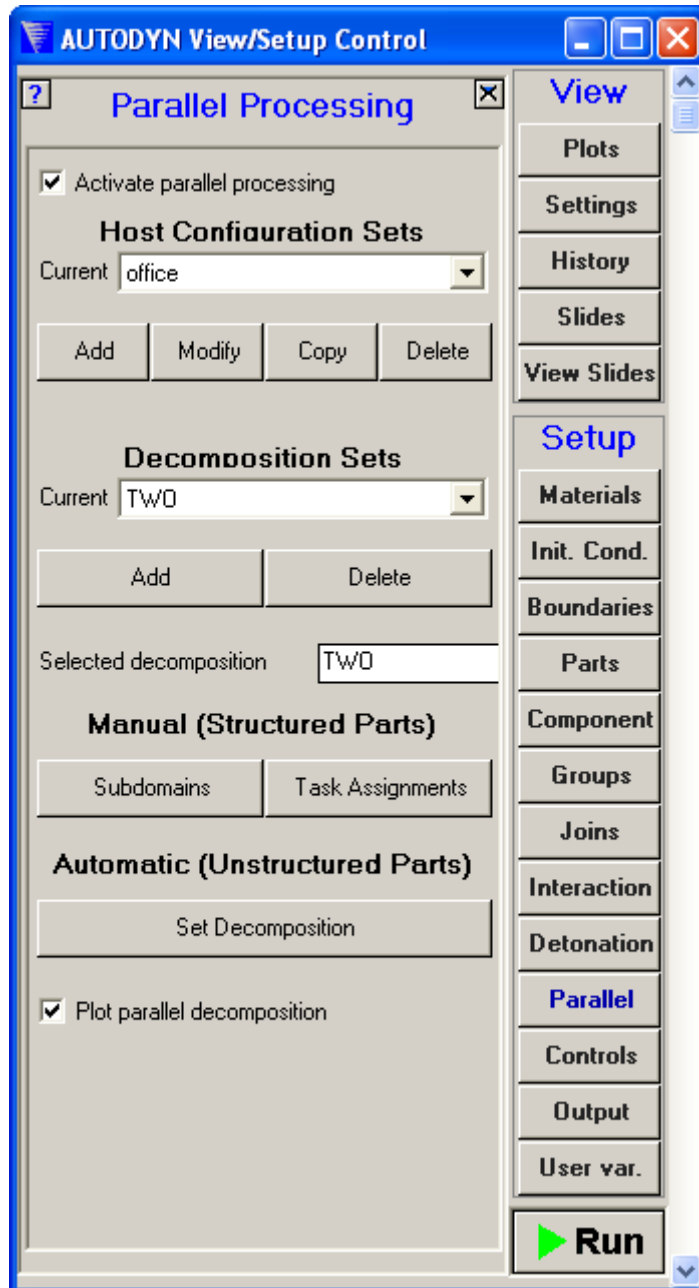
Press the **Add** button in the **Decomposition Set** section of the **Parallel** panel.



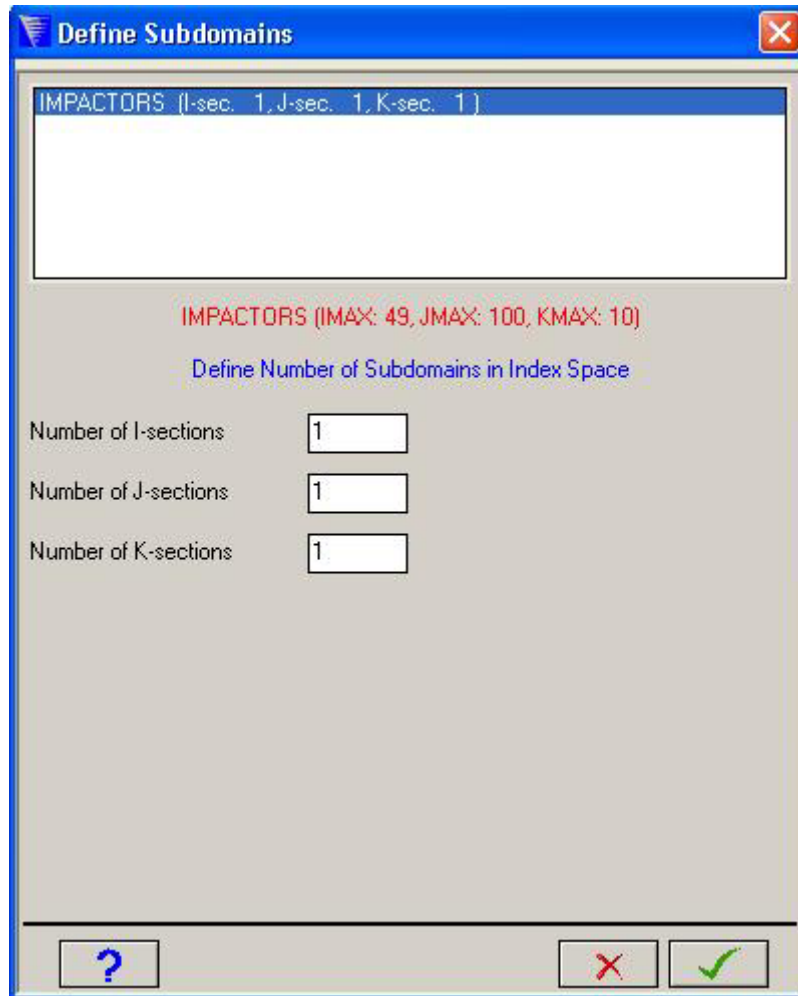
Additional buttons and information will then appear in the **Parallel** panel:



Type **"TWO"** in the **Selected decomposition** text box. This will be the name of the decomposition set that will be defined. This name is also displayed in the **Current** text box:

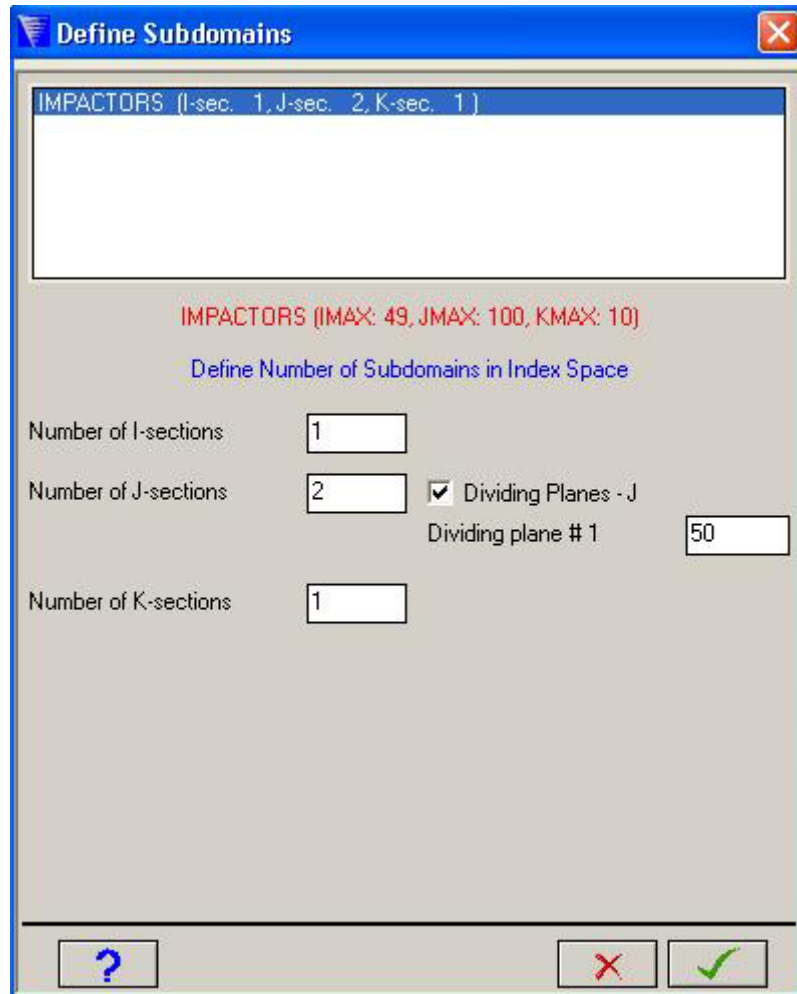


Press the **Subdomains** button to activate a popup window in which the decomposition information can be entered. The following window will appear:



The image shows a software dialog box titled "Define Subdomains". At the top, it displays "IMFACTORS (I-sec. 1, J-sec. 1, K-sec. 1)". Below this, a red text label reads "IMFACTORS (IMAX: 49, JMAX: 100, KMAX: 10)". Underneath, the instruction "Define Number of Subdomains in Index Space" is shown. There are three input fields: "Number of I-sections" with the value "1", "Number of J-sections" with the value "1", and "Number of K-sections" with the value "1". At the bottom of the dialog, there are three buttons: a question mark icon, a red 'X' icon, and a green checkmark icon.

As we wish to decompose the subgrid into two sub-domains in the J direction, enter “2” in the **Number of J-sections** text box. AUTODYN V6 will automatically evaluate the optimum dividing plane for the decomposition and enter it in the **Dividing plane #1** text box:



You may alter the J value of the dividing plane by changing the number within the text box. There may be situations where this would be advantageous, but this would involve more complicated models.

Press the OK button to store this information.

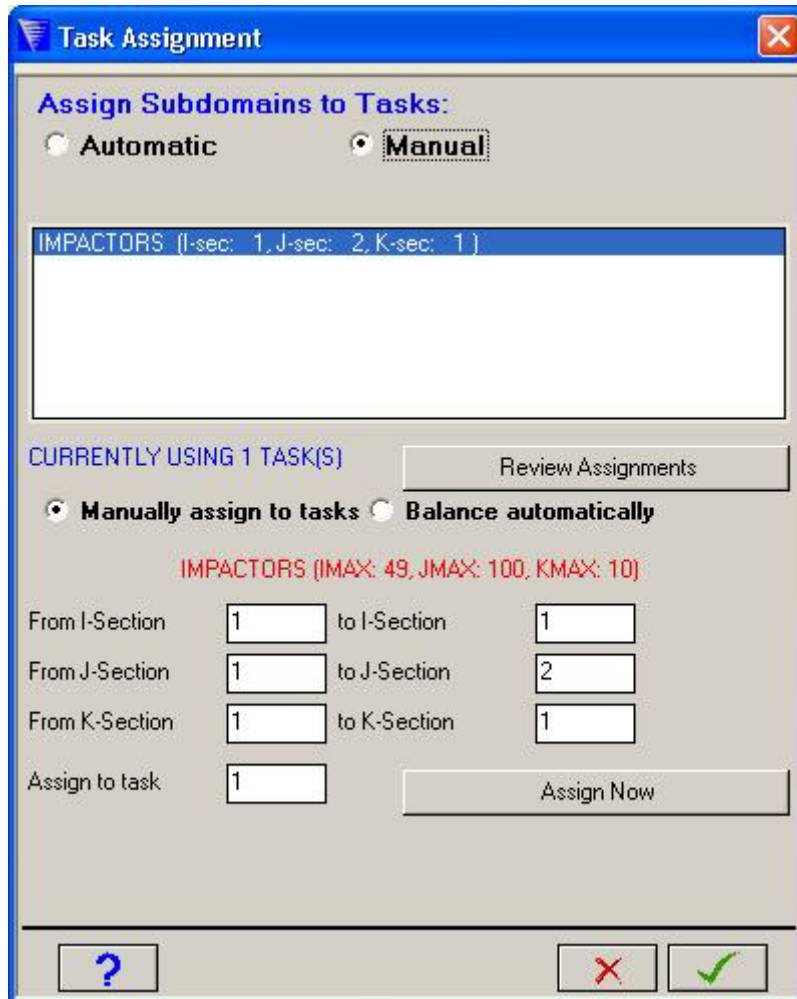
In this case the divisions can be chosen such that the resulting two sub-domains are identical in size. This is the optimum way to set up a parallel processing calculation for excellent load-balancing and most efficient execution.

This may not always be possible. You may have multiple subgrids, each of different dimensions, making it more difficult to decompose each subgrid so that one sub-domain of equal size is created for each processor.

- For the current host configuration, if you have two subgrids of differing size, it may be possible to subdivide each subgrid into two equal size sub-domains, so that you have a total of four sub-domains. These sub-domains could then be allocated one to each processor (one from each subgrid) to allow perfect load balancing. There would be a small amount of overhead, due to processing two sub-domains instead of one on each processor, but this is likely to be a small price to pay for perfect load-balancing.

- Taking this principal a step further, for multiple subgrids that do not offer very uniform decomposition, you can decompose the subgrids into many more sub-domains than there are processors and let AUTODYN automatically load-balance the sub-domains over all processors (a discussion of automatic load-balancing follows). The more sub-domains AUTODYN has to distribute, the easier it will be to effectively load-balance a problem. However, the more sub-domains created, the more overhead will be involved to exchange data at sub-domain boundaries. You may have to experiment to find a good balance, but our experience so far indicates that parallel calculation are generally computation intensive, so it is likely that good load-balancing is more important than minimizing data communication between sub-domains. Most of our benchmarking has been on well load-balanced problems with minimal communication overhead, so we cannot offer more detailed guidelines at present.

The subgrid has now been decomposed into sub-domains. Though, if you view the parallel decomposition you will see that both of the sub-domains have been automatically allocated to one task. Therefore, we need to assign these sub-domains to the available tasks in order to produce an efficient calculation. This process is initiated by pressing the **Task Assignments** button. The following popup window will appear:

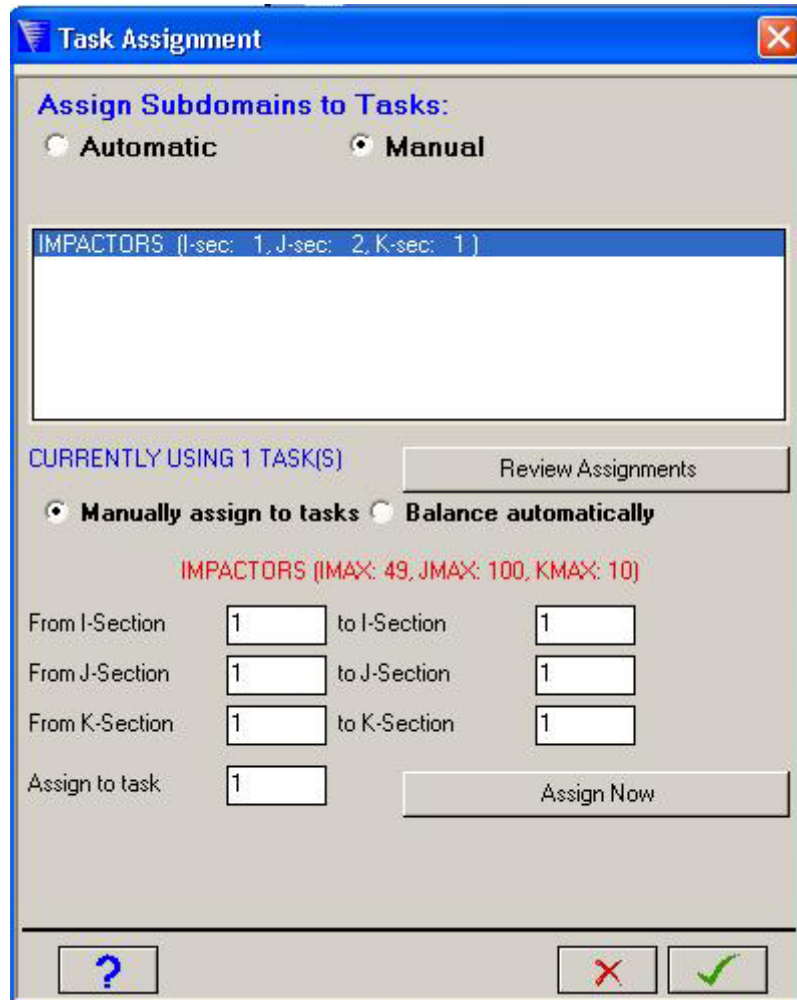


The image shows a 'Task Assignment' dialog box with the following elements:

- Assign Subdomains to Tasks:**
 - Automatic
 - Manual**
- IMPACTORS (I-sec: 1, J-sec: 2, K-sec: 1)
- CURRENTLY USING 1 TASK(S) [Review Assignments]
- Manually assign to tasks** **Balance automatically**
- IMPACTORS (IMAX: 49, JMAX: 100, KMAX: 10)
- From I-Section: 1 to I-Section: 1
- From J-Section: 1 to J-Section: 2
- From K-Section: 1 to K-Section: 1
- Assign to task: 1 [Assign Now]
- Buttons: ? (Help), X (Close), and a green checkmark (OK).

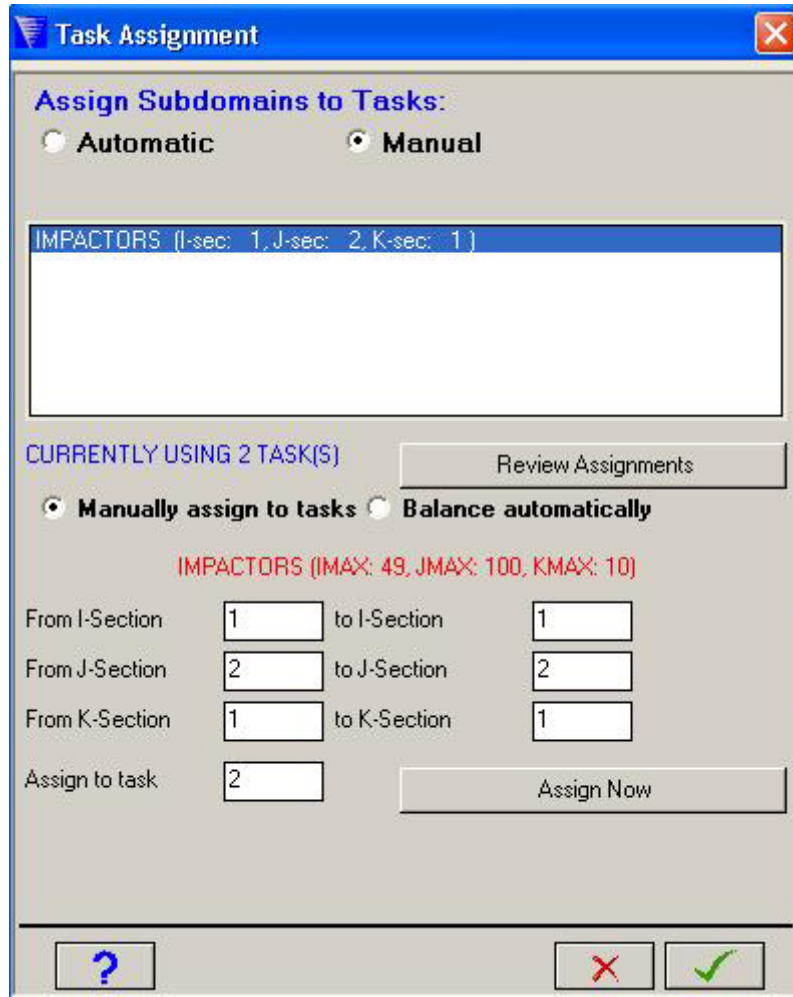
The sub-domains can be assigned manually or automatically. If the **Automatic** radio box is checked then the sub-domains will be assigned at run time. If the model is a very simple one then this course of action could be taken. Here we will manually assign the sub-domains to the available tasks in order to become familiar with the process.

We wish to assign the two J-sections of the model to two different tasks. To do this, enter 1 into the **to J-Section** text box and press the **Assign Now** button:



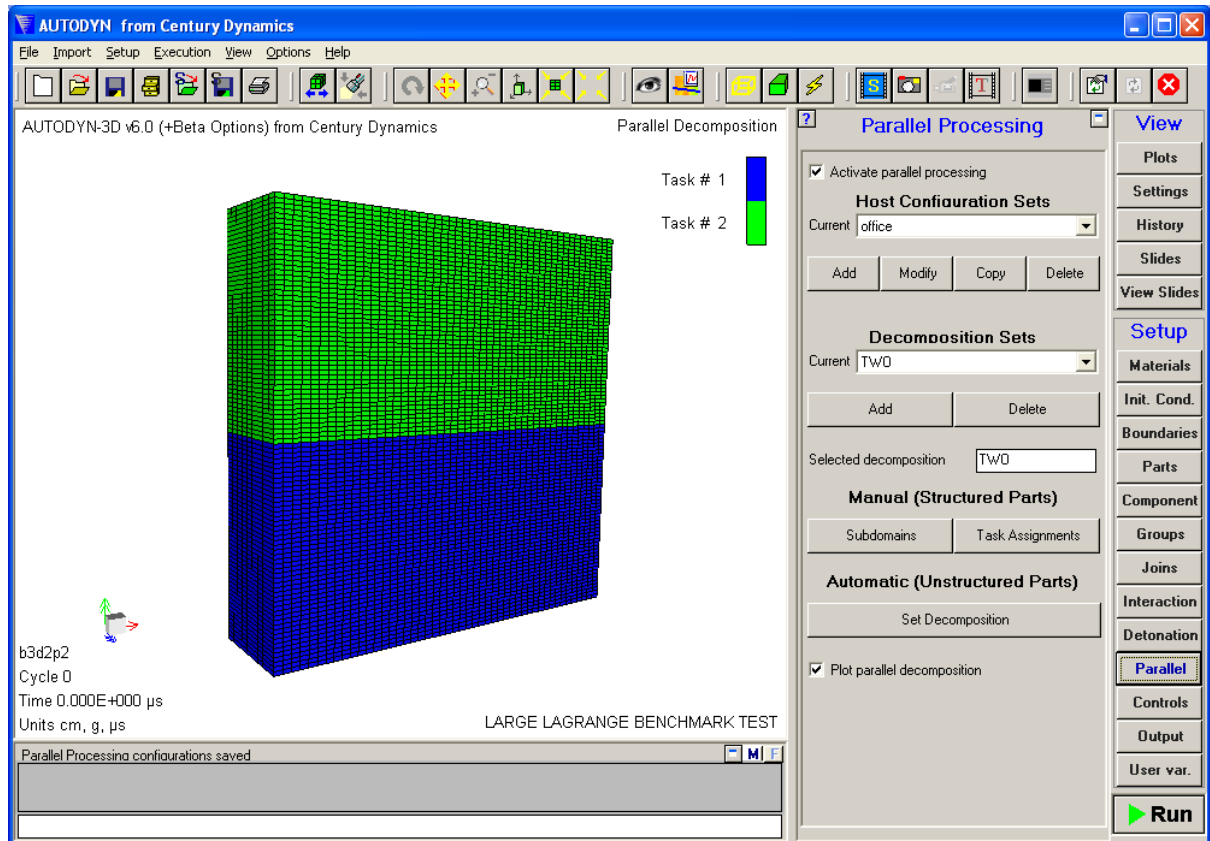
These actions have assigned the first J-section sub-domain to the first task.

To assign the second J-section sub-domain to the second task enter “2” into the **From J-section, to J-section** and **Assign to task** text boxes and press the **Assign Now** button.

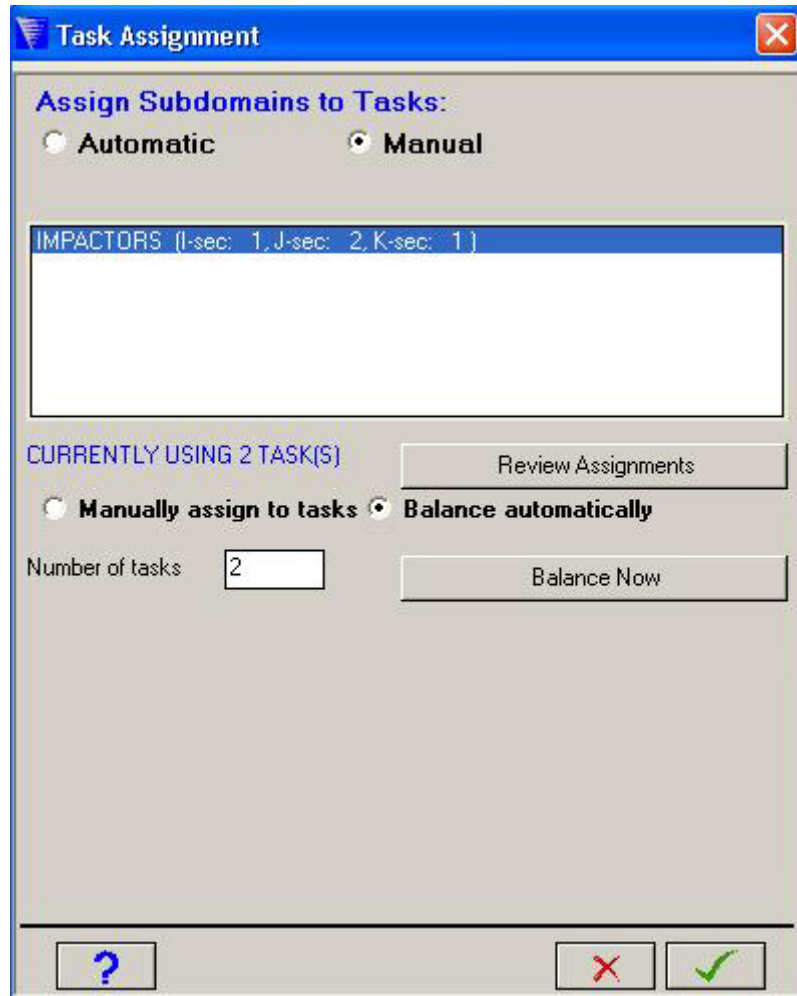


If you have automatic refresh activated the image of the model on screen should have been updated to show the two sub-domains on two different tasks:

Creating Domain Decomposition Sets



We could have balanced the task assignments automatically and viewed the results instantly by checking the **Balance automatically** radio box in the **Task Assignment** window. This results in a change to the configuration of the window:



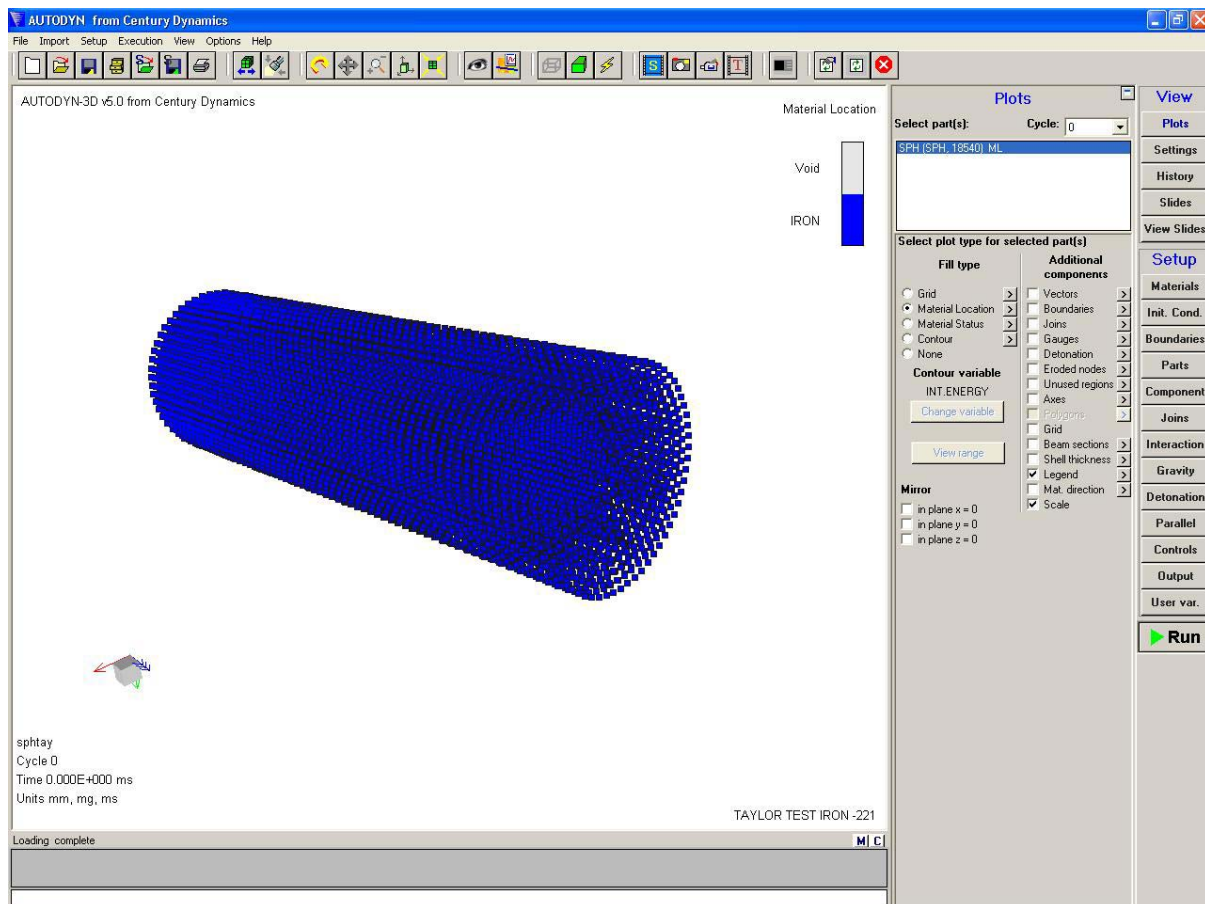
By pressing the **Balance Now** button the task assignments will be immediately processed automatically. If you view the domain decomposition image then you will see that the decomposition is identical to that which we defined manually.

The model is now ready to be run in parallel. Press the **RUN** button to execute the calculation.

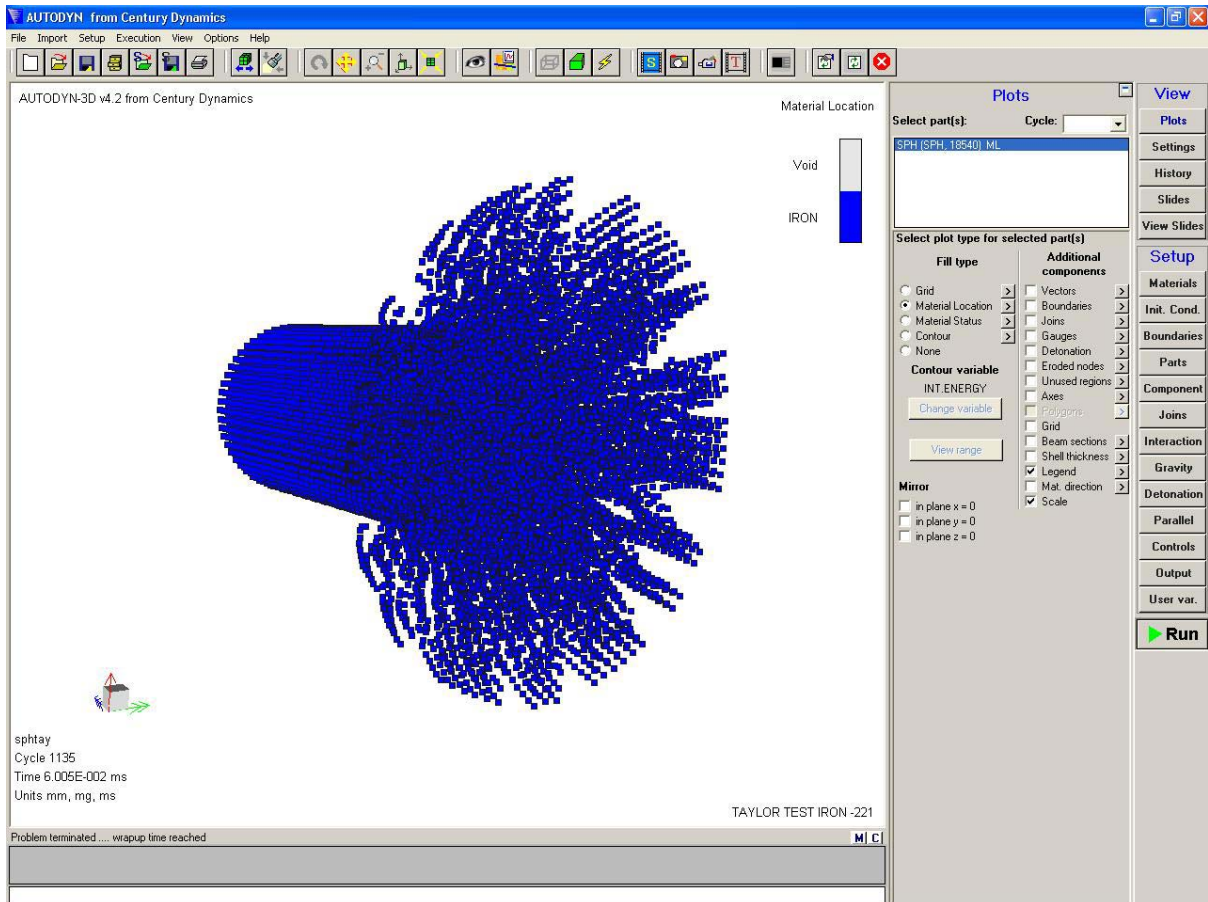
§2. Decomposing SPH calculations

Decomposing SPH subgrids adopts a similar approach to that applied for grid calculations, though there is no manual option to decompose the problem.

Load in the SPH problem **SPHTAY** from the Sample3d directory within the AUTODYN distribution.



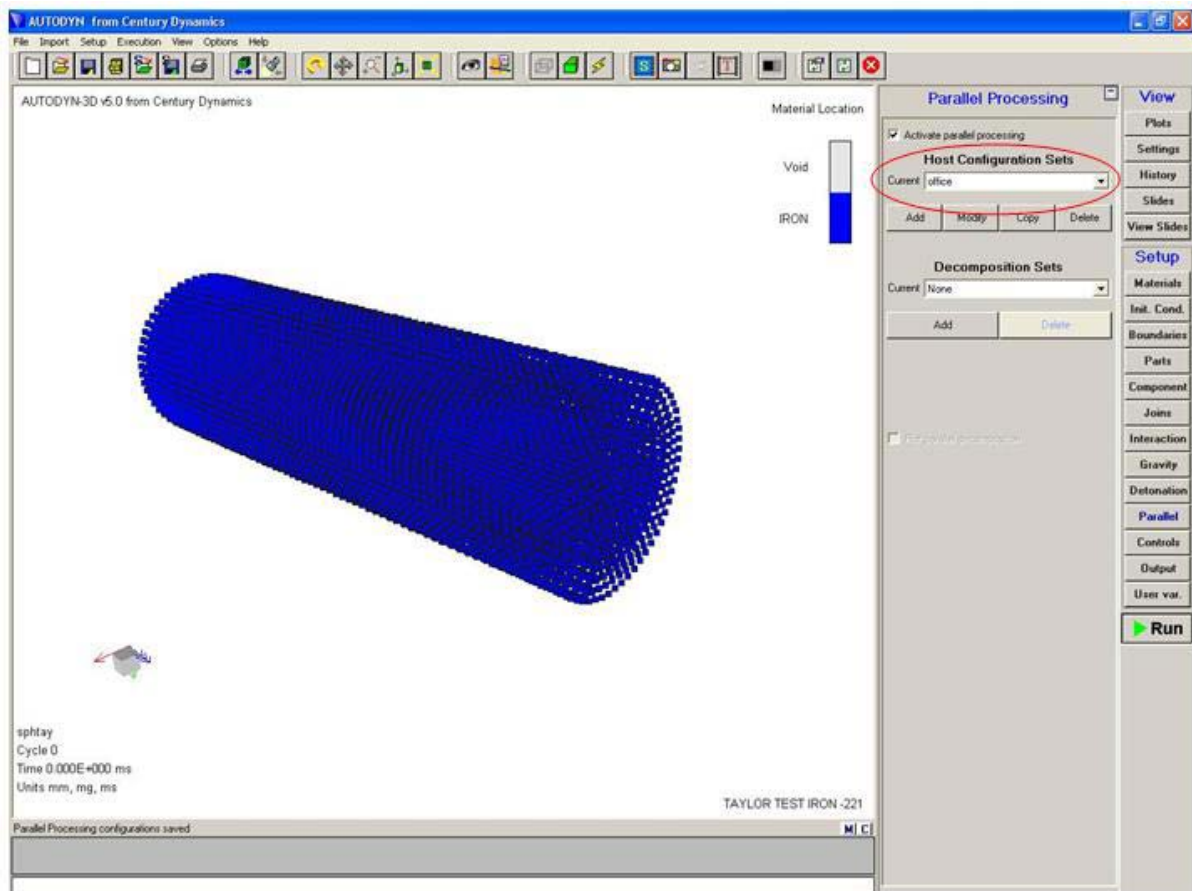
This is an iron Taylor Test problem where the deformation of a cylinder is studied after impacting a wall at 221 ms^{-1} . The model consists of a single SPH object made up of 18540 SPH nodes. Running the model to 0.06 ms will produce the following results:



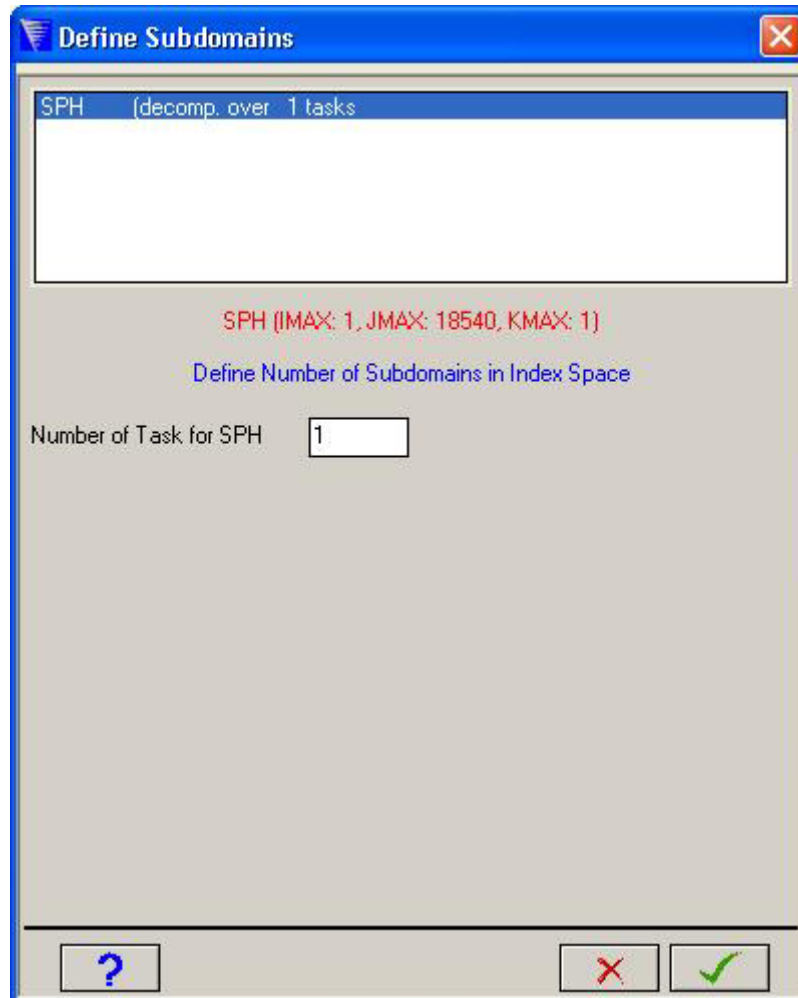
To decompose the model, press the **Parallel** button in the **Setup** toolbar and activate parallel processing by ticking the appropriate check box.

As the host configuration information is not specific to an AUTODYN model, this information will have been read by AUTODYN and the current configuration will be displayed in the **Host Configuration Sets** text box.

Creating Domain Decomposition Sets

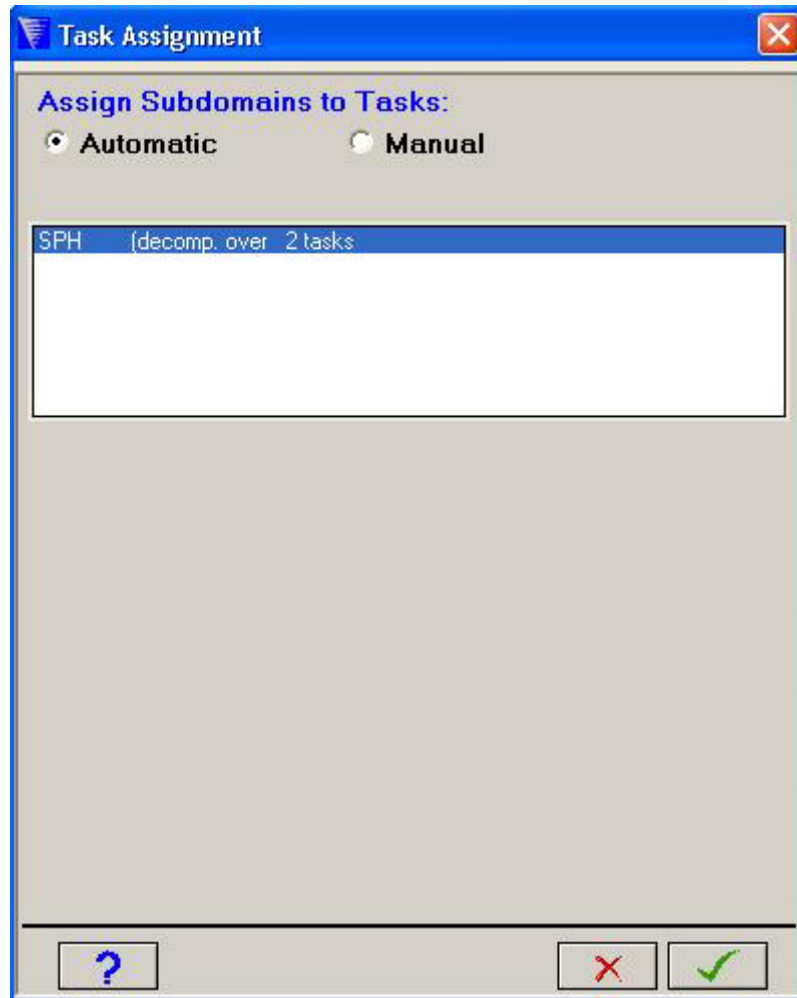


Following the same method as when decomposing the Lagrange calculation, press the **Add** button in the **Decomposition Sets** section of the **Parallel Processing** panel. Again enter "TWO" as the name of the decomposition and press the **Sub-domains** button, which will result in the following window being displayed.

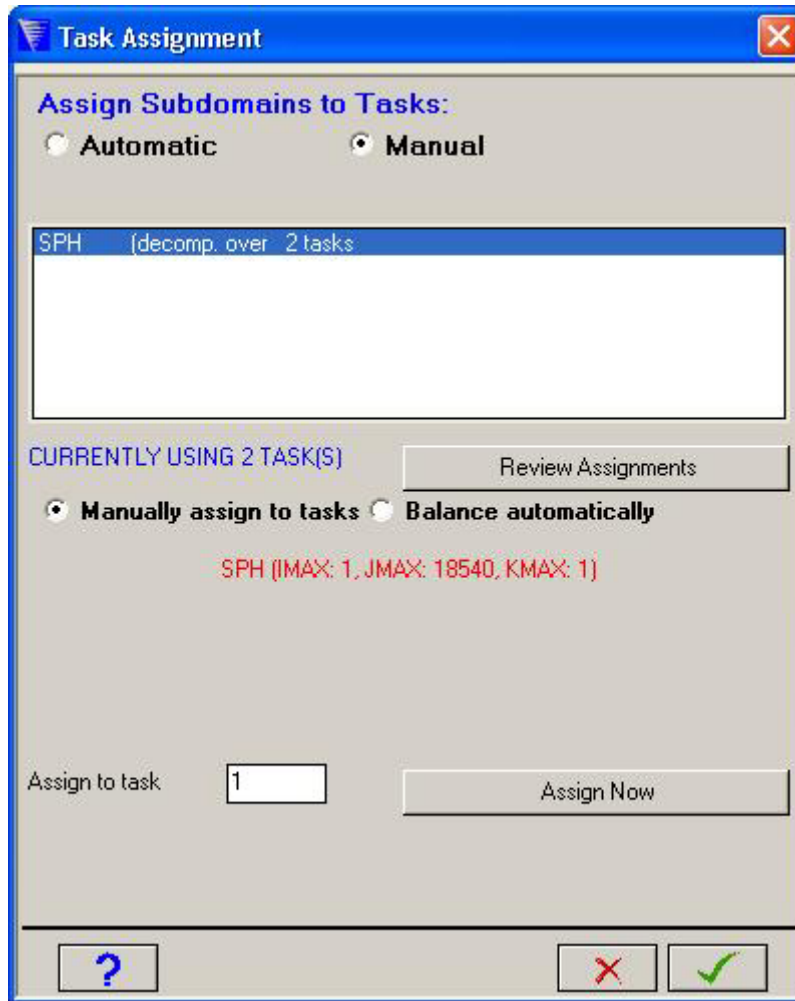


Enter “2” in the **Number of Task for SPH** text box and press the tick button.

Now, press the **Task Assignments** button on the main **Parallel Processing** panel. The following window will be displayed.

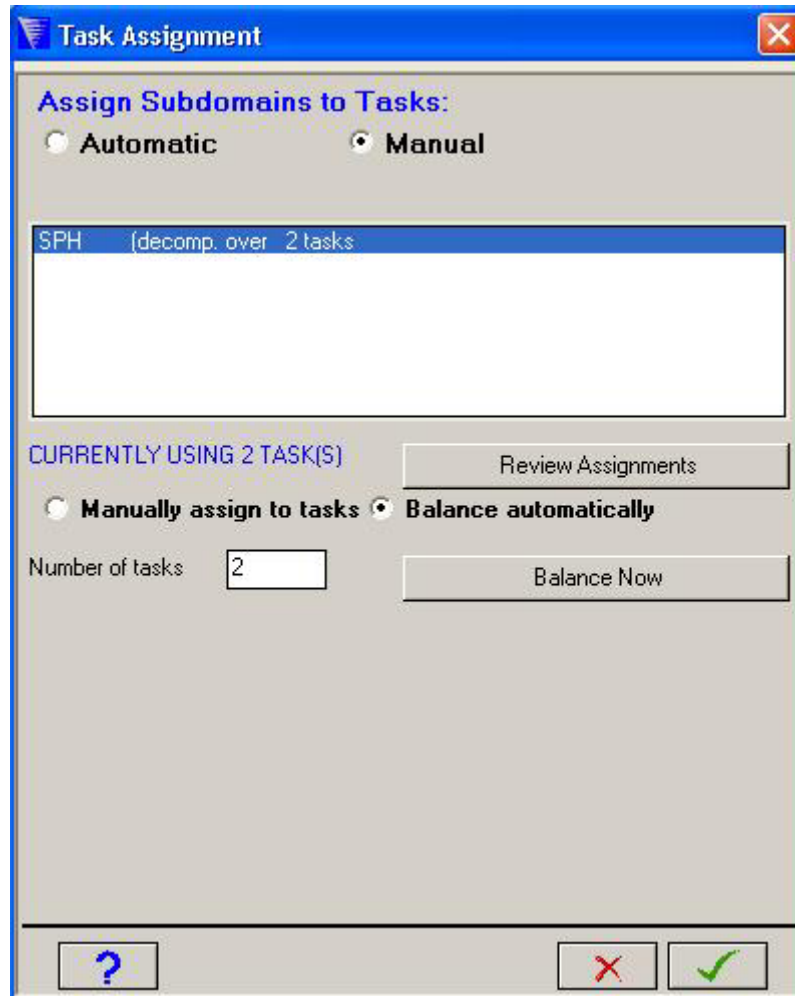


By default the “**Automatic**” radio box will be chosen. By leaving the default setting, the subdomains would be assigned to tasks at runtime. As we wish to view the decomposition, check the **Manual** radio box, which will result in further options being displayed.

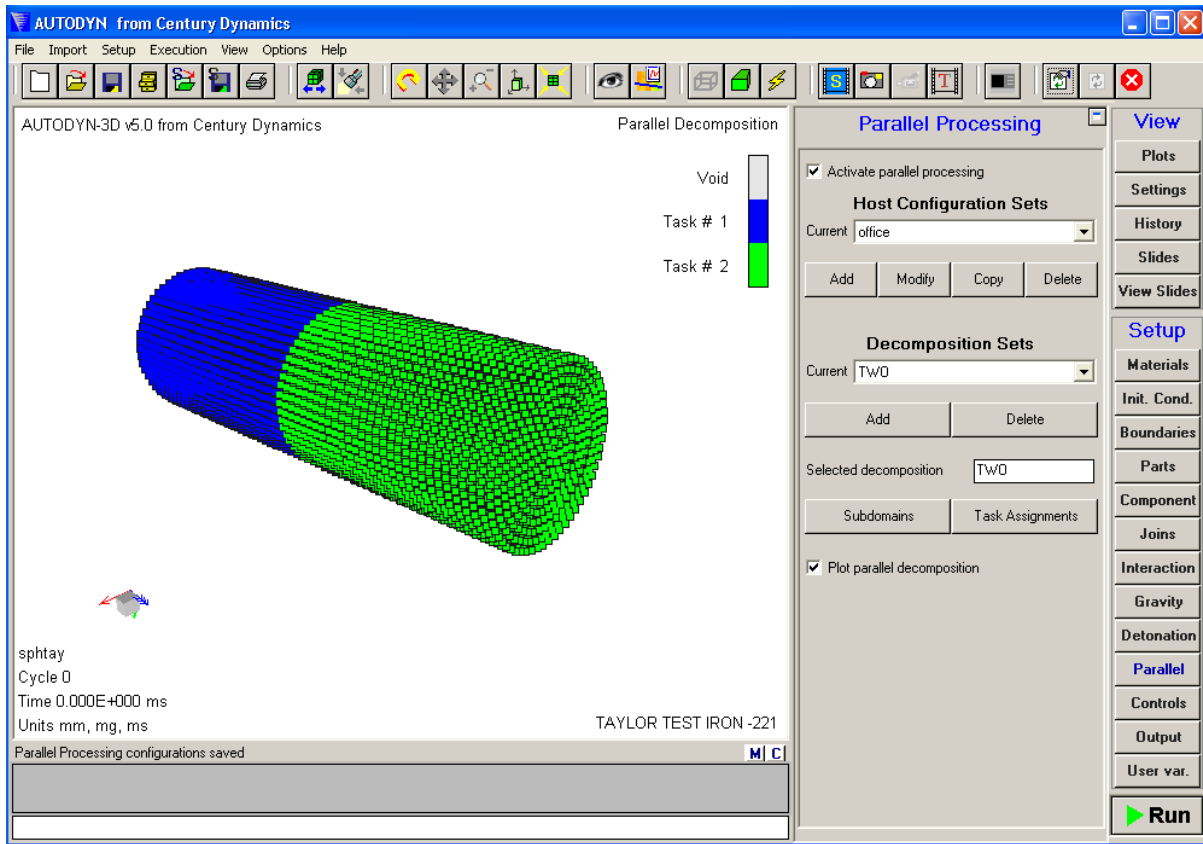


If we wished to assign the entire SPH object to one task, which may be a viable option if the model contained multiple subgrids, we would enter the task number in the **Assign to task** text box and press the **Assign Now** button. As the model consists of a single SPH object we need to decompose the object onto the two available processors in order to efficiently compute the results.

Check the **Balance automatically** radio box and the available options will be modified.



The number of tasks will automatically be placed in the **Number of tasks** text box and therefore we only have to press the **Balance Now** button in order to assign the SPH nodes to the different tasks. AUTODYN V6 will attempt to assign the nodes in the most efficient configuration possible for parallel processing. After doing this, press the tick button to change the focus back to the main AUTODYN V6 window. Check the **Plot parallel decomposition** tickbox and the decomposition will be displayed.



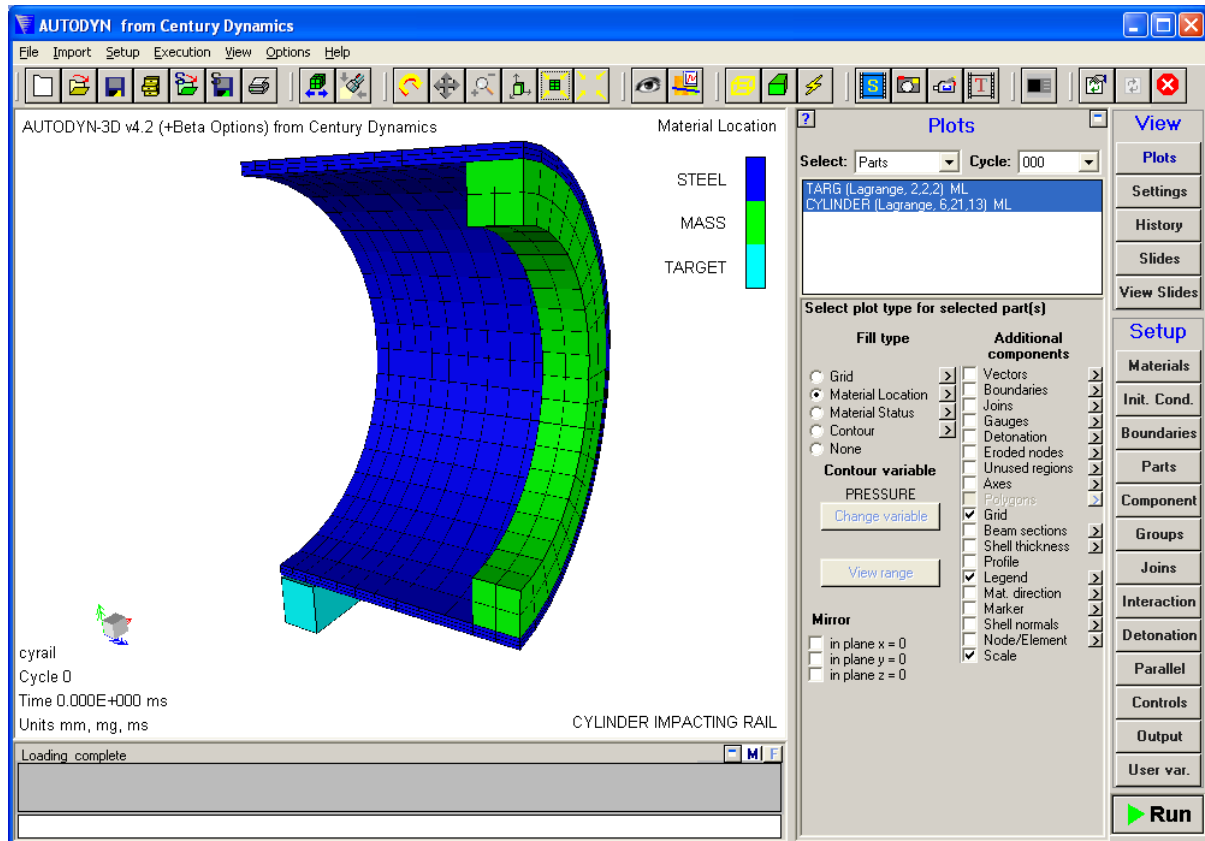
The model can be now efficiently run as a parallel computation.

§3. Decomposing Unstructured calculations

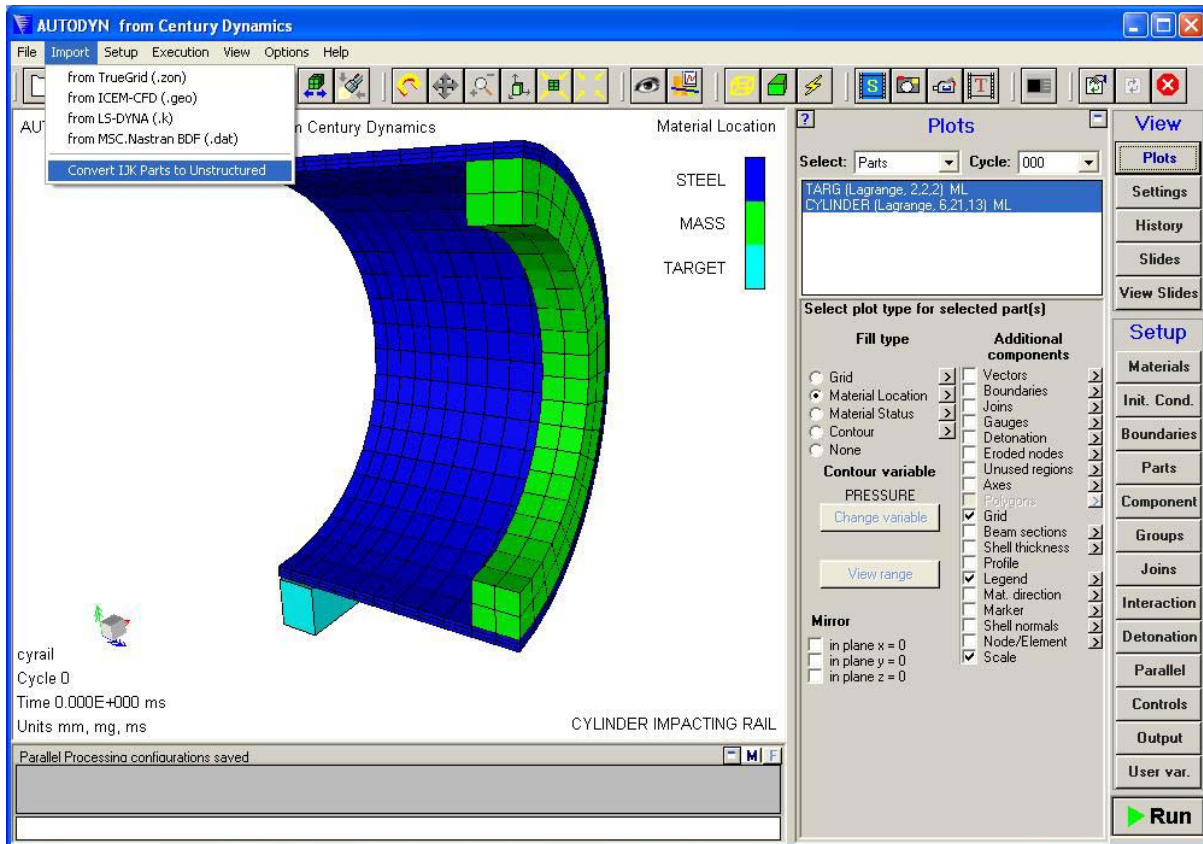
The decomposing of unstructured parts is applied automatically in a similar method to the decomposition of SPH grids.

Load in CYRAIL from the samples directory within the AUTODYN distribution:

Creating Domain Decomposition Sets

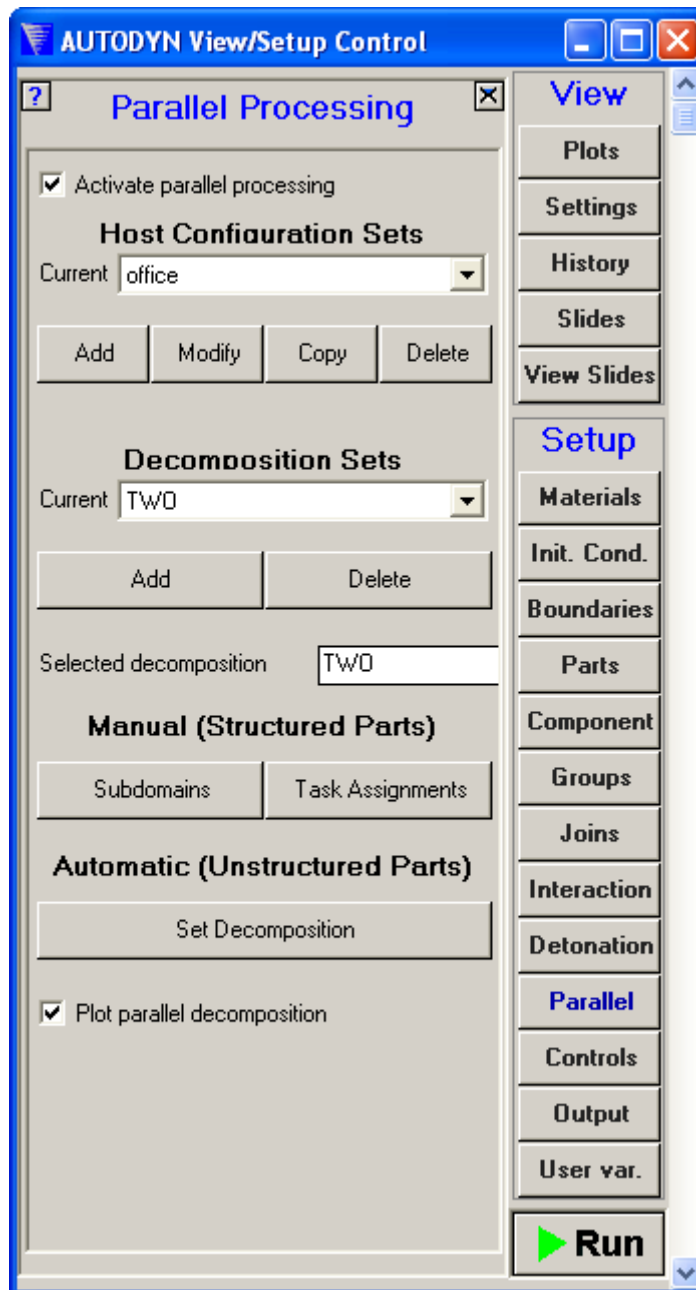


Convert this structured model into unstructured parts by using the “Import->Convert IJK Parts to Unstructured” option and ticking all of the available options:

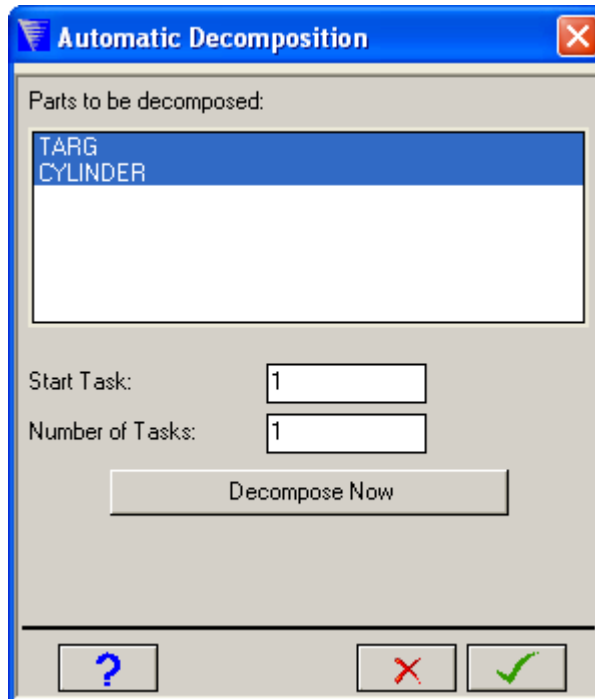


This makes two unstructured parts called VOLUME 1 and VOLUME 2. Rename these parts to TARG and CYLINDER in the parts menu.

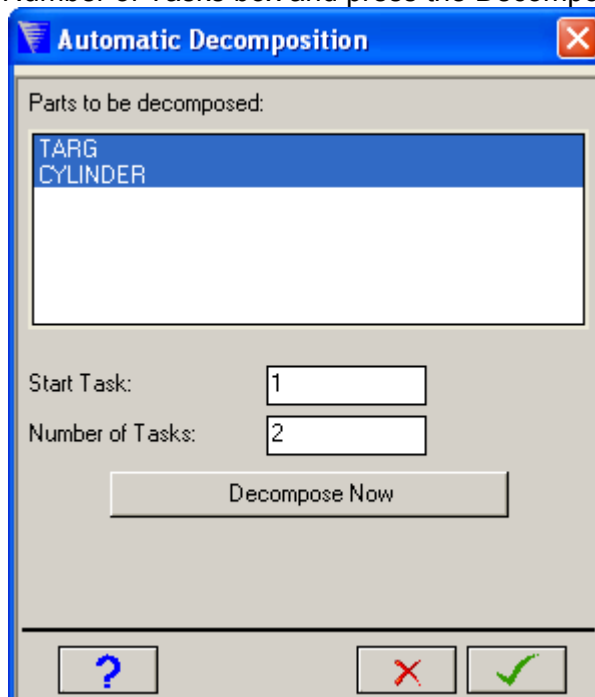
Go to the Parallel panel and tick the box to activate parallel processing. Add a decomposition set and call it TWO:



You will see that there are two sections within the Decomposition Sets area of the menu. For unstructured parts the second button marked Set Decomposition should be used. Pressing this button activates a window in which the information regarding the decomposition can be entered:

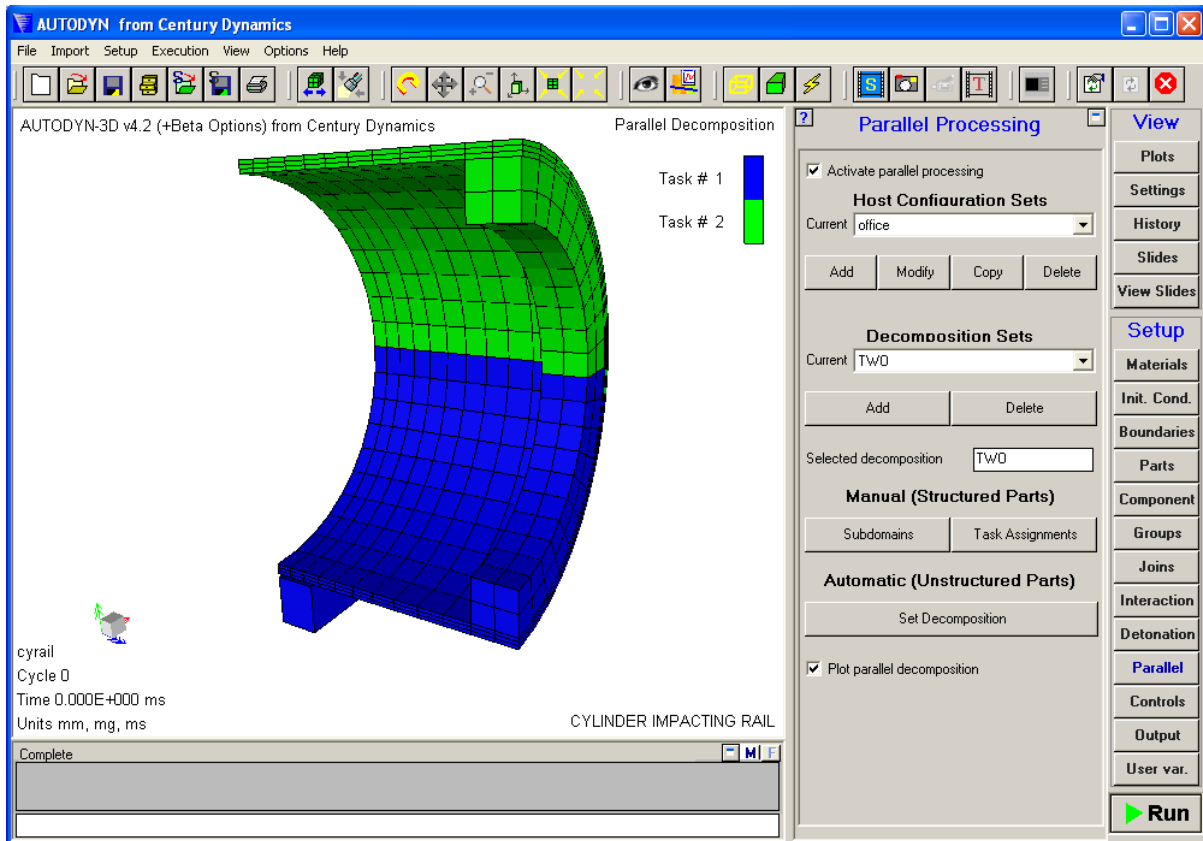


All the unstructured parts within the model will be in the Parts to be decomposed window and all are highlighted by default. To decompose the model over two tasks (task 1 and task 2) simply enter 2 into the Number of Tasks box and press the Decompose Now button:

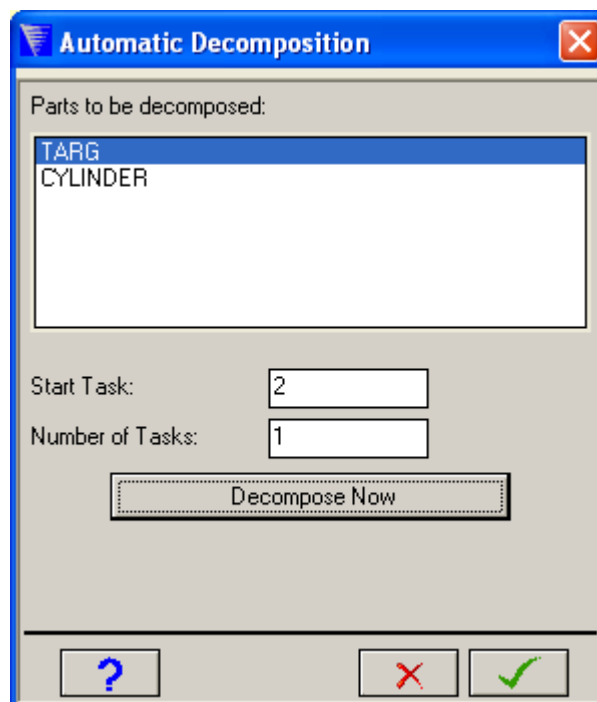


Press the tick button and by choosing to plot the parallel decomposition, the following decomposition is seen:

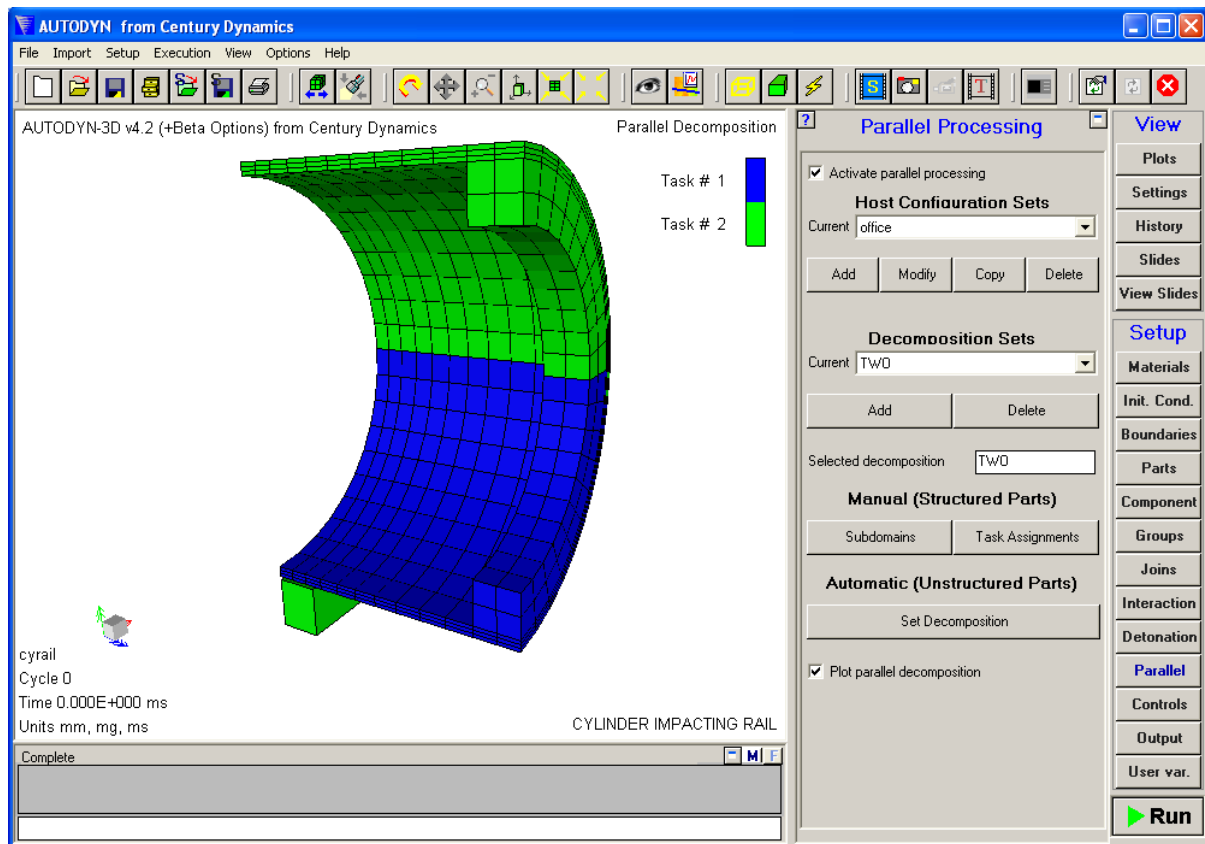
Creating Domain Decomposition Sets



If you wish to have the part TARG on Task 2, press the Set Decomposition again and highlight part TARG only, enter two into the Start Task box and press the Decompose Now button:

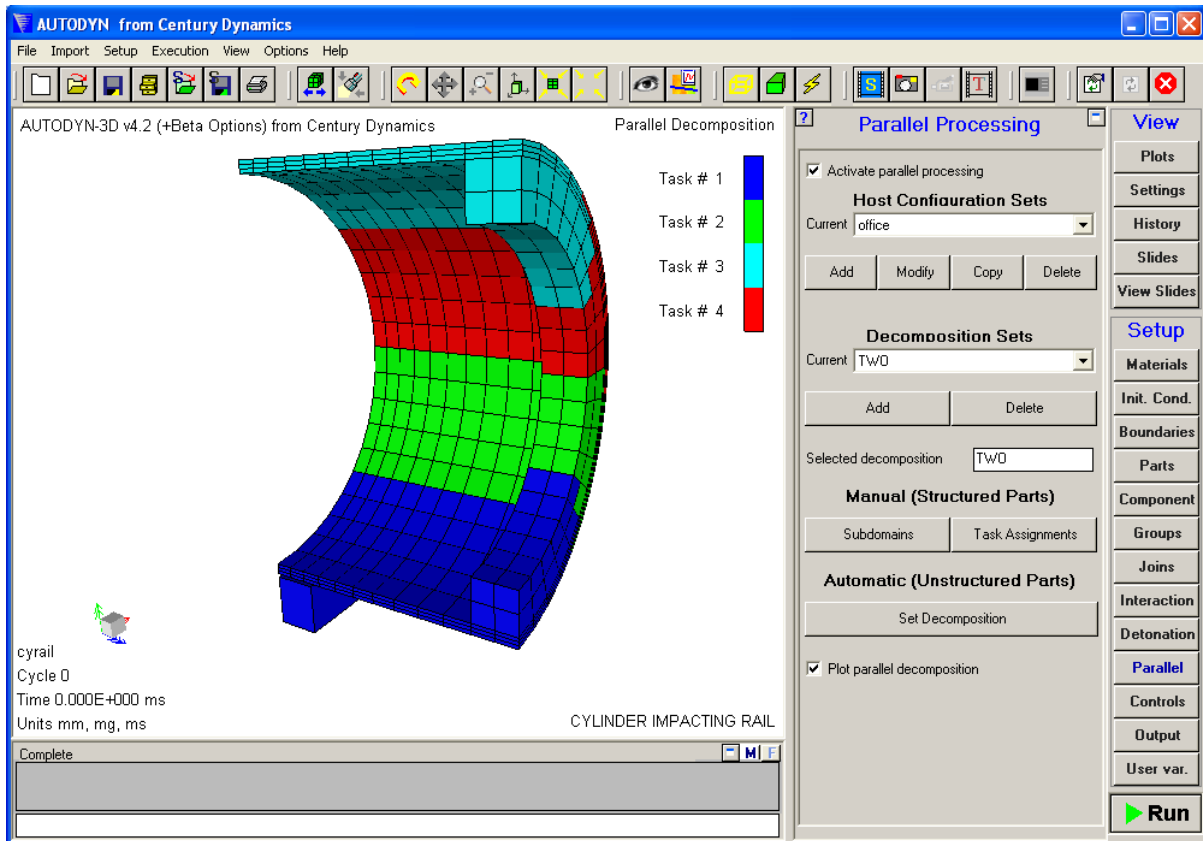


The part TARG will now be assigned to task 2:

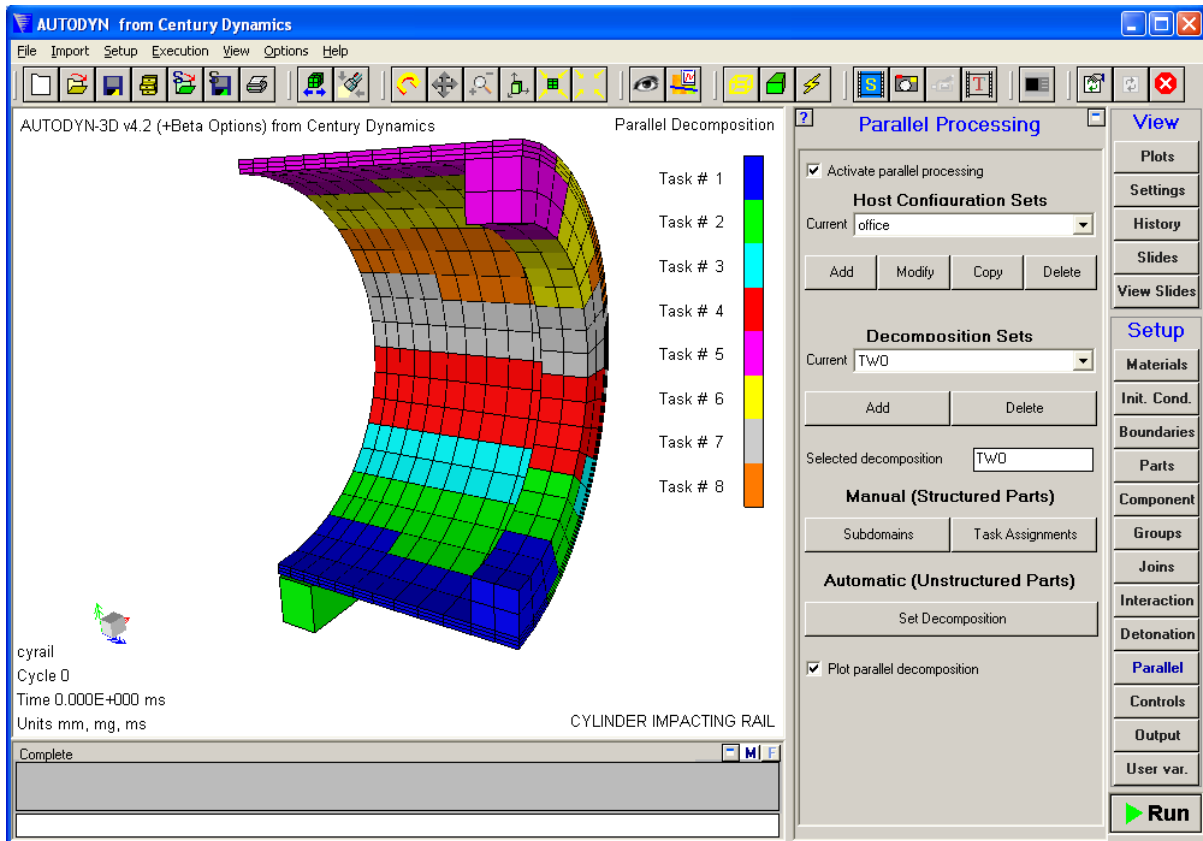


In a similar manner the model can be decomposed over 4 slaves:

Creating Domain Decomposition Sets



and over 8 slaves:



An Important Note on Host Configuration Set Names and Decomposition Set Names.

In general you can choose any names you like for Host Configuration sets and Decomposition sets. However, if you select a Decomposition set and a Host Configuration set exists with the same name, that Host Configuration set will be automatically used for the calculation, overriding whatever Host Configuration set was currently active (including "None").

Chapter 6. Benchmarking Calculations

The primary reason for using parallel processing is to execute problems faster. It is therefore important to be able to assess how efficiently a problem is being processed in parallel. The simplest and best way to get this information is to monitor the time it takes to complete each computational cycle. Comparing cycle times from a problem run in scalar mode with those from the same calculation run in parallel mode allows you to accurately assess the efficiency of the parallelization.

AUTODYN V6 allows you to print this information to the **Performance Profiler** window if you wish. To take advantage of this option, you must create an empty file, **profile.cfg**, in the directory where the AUTODYN executable (**autodynv6**) resides.

The **Performance Profiler** can be accessed through the **View** menu.

If you do this, the following type of output will be written to the **Performance Profiler** window:

```

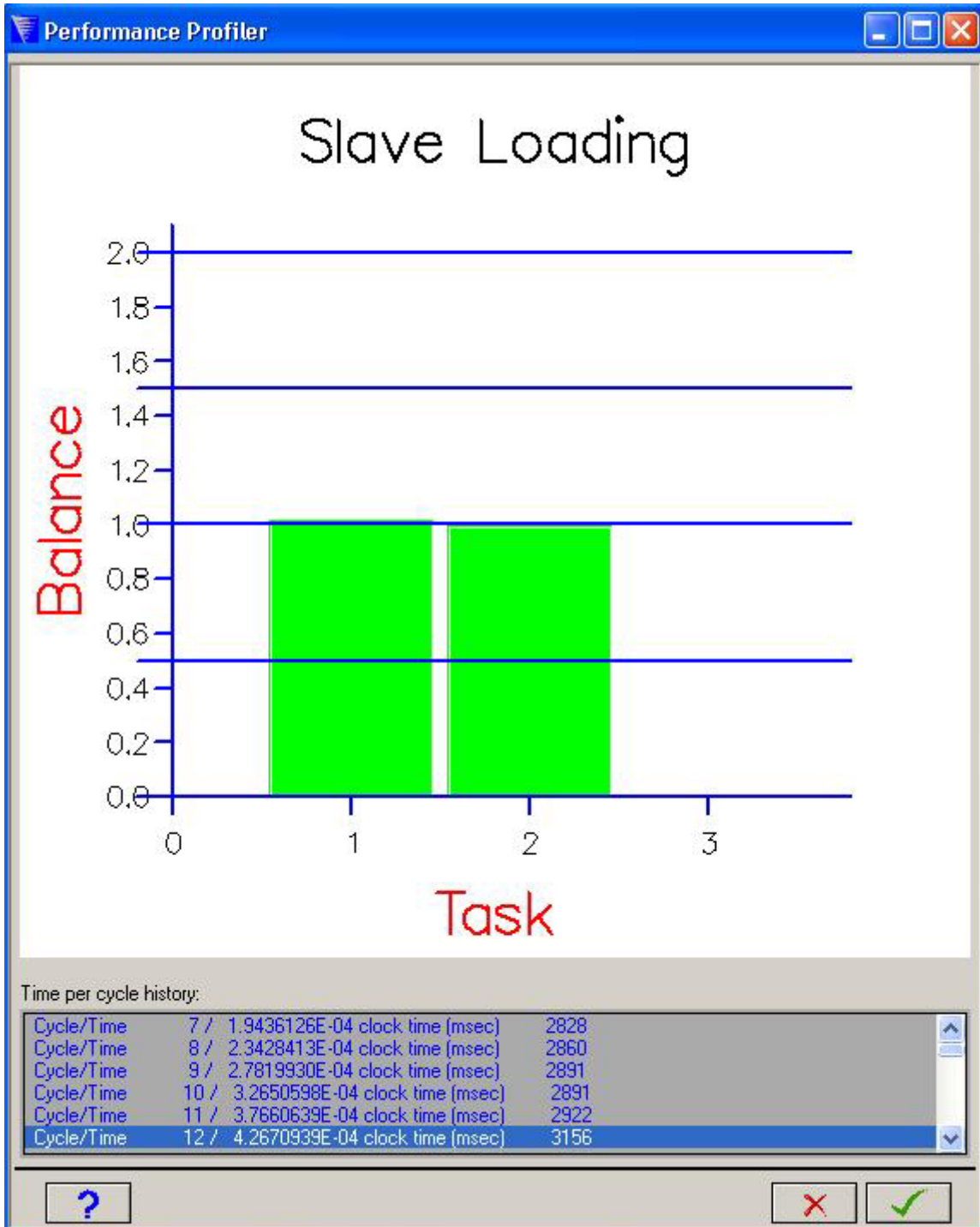
Cycle/Time  1 / 1.637809225157053E-4 clock time (msec) 32856 (update)
Cycle/Time  2 / 3.439399411878227E-4 clock time (msec)  time 17030
Cycle/Time  3 / 5.421148660224775E-4 clock time (msec)  time 17056
Cycle/Time  4 / 7.601072880654561E-4 clock time (msec)  time 17029
Cycle/Time  5 / 9.998989575100770E-4 clock time (msec)  time 17054
Cycle/Time  6 / 1.263669799616239E-3 clock time (msec)  time 17016
Cycle/Time  7 / 1.553817732221804E-3 clock time (msec)  time 17081
Cycle/Time  8 / 1.872980465005591E-3 clock time (msec)  time 16994
Cycle/Time  9 / 2.224059478677190E-3 clock time (msec)  time 17116
Cycle/Time 10 / 2.610246402086324E-3 clock time (msec)  time 17024
Cycle/Time 11 / 3.035052027043786E-3 clock time (msec)  time 17075
Cycle/Time 12 / 3.502338224625149E-3 clock time (msec)  time 17070
Cycle/Time 13 / 4.016353053105620E-3 clock time (msec)  time 17053
Cycle/Time 14 / 4.581769376689206E-3 clock time (msec)  time 17019
Cycle/Time 15 / 5.203727346111727E-3 clock time (msec)  time 17070
Cycle/Time 16 / 5.887881127305133E-3 clock time (msec)  time 17011
Cycle/Time 17 / 6.640450302929377E-3 clock time (msec)  time 17085
Cycle/Time 18 / 7.468276414058693E-3 clock time (msec)  time 34853 (update)
Cycle/Time 19 / 8.378885156037852E-3 clock time (msec)  time 17017
Cycle/Time 20 / 9.380554793925532E-3 clock time (msec)  time 17098

```

Each line gives the cycle number, the cycle time, and the wall clock time that the calculation used to compute the cycle (in milliseconds). If the line ends with “**(update)**”, this indicates

that work unit information for the interaction logic was updated on that cycle (see chapter 2). Such cycles usually take longer to compute.

The **Performance Profiler** also illustrates the computational load on each slave.



The load balance shown above illustrates an optimum configuration. The load on each slave is approximately the same. If the load balance shown is predominantly biased towards one

slave then the parallel decomposition should be analyzed and preferably altered to produce a more balanced parallel decomposition configuration.

A number of benchmark calculations have been performed to determine the efficiency of AUTODYN when performing Lagrange, ALE, Shell and Euler calculations. The results of these calculations have been presented in the following papers:

1. "A Parallel Algorithm For The Deformation And Interaction Of Structures Modeled With Lagrange Meshes In Autodyn-3D", M. S. Cowler, T. Wilson, O. La'adan International Symposium on Impact Engineering, Singapore, December 1998
2. "High performance computing using AUTODYN-3D", M. S. Cowler, O. La'adan, T. Ohta, 6th International Conference On Application of High Performance Computing in Engineering", Maui, Hawaii, January 2000.

In general we have obtained efficiencies of close to 100% for Lagrange and Shell calculations (including contact interactions). Efficiencies dropped to around 75% for ALE calculations and approximately 60% for Euler calculations .

The current load-balancing algorithm needs improvement and does not work perfectly in all cases. Because of this we offer the following guidelines if you wish to set up problems that will process efficiently in parallel.

- Use a homogeneous system
- Use a dedicated cluster of CPU's
- Check the automatic load-balancing assignments before and during the execution of a problem. If they don't look good, do your own manual assignments

If you are using a combination of solvers, it is advisable to undertake your own manual sub-domain to processor assignments. Combinations of Lagrange and Shell subgrids may balance well, but weighting factors for other solvers have not been determined at this time.