# Basic Principles for ReadRes

ReadRes is a MatLab based function (m-file) that will read data from the restart file created by Simulate 3 and Simulate 5.

The basic idea is to do one first read to create a “table of contents” of the file, which is called an info structure (will be called resinfo from now on) and contains the basic information of the file, the core and the positions of the data in the file.  
Resinfo is then used in the second read, where the data wanted is read.

The restart file always has the same basic structure and is based on records (one label, the size of the data and the data). These are all the same if additional exposure points are saved on one restart file.

First read - FindLabels

The first read in ReadRes has two different options, ‘FULL’ or ‘NODATA’, where ‘FULL’ is the default. ‘NODATA’ is an option where the positions of the fuel data are not read. The first read is done by the function FindLabels.

The first part of FindLabels is a “read and locate” loop that will locate the labels and the positions of the labels, when the loop finds the same label as the first one (‘PARAMETERS’) it stops.  
The next part is to find out how many exposure points that are saved on the file, this is done by calculating the size of the file based on the total size of the file and the size in between to ‘PARAMETERS’ labels, this is tested as well to have conformation that the correct number is obtained.

If ‘NODATA’ is the option the loop will stop when the labels starts to repeat themselves (fuel data with the same amount of records of each labels as assemblies in the core). FindLabels will then estimate the size of the fuel data and continue after the fuel data to find the last labels. As with the estimation of the number of exposure points this is tested to get the correct positions.

The next part of FindLabels builds resinfo. The main parts is to reads some data from the different labels needed (see GetResData in second read), find out what distributions are available and find out if the file is a Simulate 3 or Simulate 5 restart file.

Second read - GetResData

With the second read the user can get the data wanted from the restart file. The second read is done by the function GetResData and the main thing needed is resinfo from the first read where all positions are stored. There are three additional inputs next to resinfo used in GetResData. The “distlab” which tells the GetResData what data is wanted (not optional), the “xpo” which tells which state point is wanted (optional, default the first), and the last input is the “asspos” or assembly position which tells which assembly is wanted (optional, default all assemblies). GetResData can only read one “distlab” and one “xpo” but any number of “asspos”. ReadRes can handle both multiple “distlab” and multiple “xpo”, but this is handled in ReadRes.

GetResData is basically a wrapper that will look at two key parameters, simulate version (3 or 5) and if Simulate has been run with 2x2 or not. GetResData will then direct the information needed one of the four different GetResData (GetResDataS3\_1x1, GetResDataS3\_2x2, GetResDataS5\_1x1 and GetResDataS5\_2x2) this division is done since the data is quite different from each case and have a more easy code (without all extra if statements).

In the different GetResData there are two different ways of reading data. The most used is GetNextRecord (used in the subfunction GetDist), which is a program needing the label, the “formats” (‘int’, ‘char’ and ‘float’) and “number” of entities for each format, and reads the data from that label. All labels (for both Simulate 3 and 5) have their formats predefined in GetFormatNrS3 and GetFormatNrS5. GetNextRecord is used to get all data except for the fuel data.

The different fuel data is read directly in the different GetResData. Since there can be different distributions on the file the relative positions within the fuel data is found with RelativePosInData.

**Important!** When a new “distlab” is added be sure to add it to the variable ‘nondists’ if it is not core distribution. And if “switch case” is used (other than for distlab) add the cases to exceptions, this is so that resinfo will be correct when changing GetResData.

ReadRes

ReadRes is the program controlling whether FindLabels or GetResData and controls the input and output from them.

FindLabels needs the filename to create resinfo, and the additional 'FULL' or 'NODATA' will be used if given. If the second input to ReadRes is not 'FULL' or 'NODATA' ReadRes will assume that it is a "distlab" and a distribution is wanted. Then ReadRes will send the filename to FindLabels to create resinfo and then send resinfo to GetResData, this will generate a warning that always create resinfo first to save time.

Since GetResData only handles one "distlab" and one “xpo” (in number format) ReadRes needs to take care of additional "distlab" and a number of different inputs for "xpo". The “asspos” is handled in GetResData so ReadRes does not need to take care of them. When the "distlab" and "xpo" is sorted out GetResData is used to read the data.

When the data is read it in cell arrays (row "distlab" columns "xpo"), this is handled in the last part of ReadRes where the data is converted to the correct format, according to input.

Formats:

One "distlab" and one "xpo” returns the distribution in matrix format (exept for S5 subnode formats will be in cell array).  
 One "distlab" and multiple "xpo" returns a cell array with each state point per cell.   
Multiple "distlab" will return a structure with each field with each different distribution. The same principle is used for the "xpo" as with one "distlab".

Resinfo layout

The basic layout of the resinfo is like any info file from cms\_read, or ReadSum with .core, .fileinfo, .distlist and .Xpo. In addition to .distlist resinfo has a miscellaneous list called misclist. The “distlabs” in .distlist will always give a distribution of some kind (fuel data, control rod data, pin data). The “distlabs” in .misclist will return other data and different structures will data (eg. Thermal hydraulics data, Library data).

For GetResData the .data is the most important field. The resinfo.data contains

resinfo.data

* Fuel\_data
  + readopt – ‘FULL’ or ‘NODATA’ options from input
  + abs\_pos – positions of the fuel data labels
  + Label – name of the fuel data labels
* Label – all record labels present on file, can be used as input to ReadRes
* abs\_pos – the positions of the Labels (only in first state point)
* statepoints – number of state points
* st\_pts\_pos – the starting position of each state point (used as offset for abs\_pos)
* Parameters – a label in the restart file that is often used to read
* Dimensions – a label in the restart file that is often used to read

ReadCore

ReadCore has the same basic principles as ReadRes but works for .sum, .out, .cms, .res, .pinfile. ReadCore is used to call the different functions that read from each file.

ReadCore checks the extension of the file to see how to use it. If the extention is not known or if there is not one (binary files) ReadCore uses the function "identfile" to try to identify the kind of file.

ReadCore will (if not already on that format) do the same conversion as ReadRes to the data.

identfile

identfile is a function that tries to identify what kind of file it is. It was mostly create to take care of binary files which do not have a specific extension to the file.

The idea is to read 100 characters and then compare to some predefined variable known from the different files.

What identfile is looking for in different files:

* Restart file: “PARAMETERS” is on the 9:18 positions on the file
* Cms file: “XVIS0003” is the first 8 characters on the file
* Pin file: looks for the date stamp in a certain position in the beginning (not the best way)
* Sum file: the characters “Title” is the first in the file
* Out file: if “using executable” is present in the beginning of the file

TODOS

GetResData

2x2 is based on only PWR, no thermal hydraulics data programmed (except histories in fuel data), (both S3 and S5)

1x1 is based on only BWR,

In S5 the S5 based hydraulics are not programmed. The required Labels are programmed so the data is accessible.

In GetResDataS5.m the following labels are not programmed

TH-S5-MEC  
PWR STEAM  
PIN PCI  
ADAPTLPRM-MODEL  
EXCORE\_DET  
CRDMODULE  
S3KBYPASS