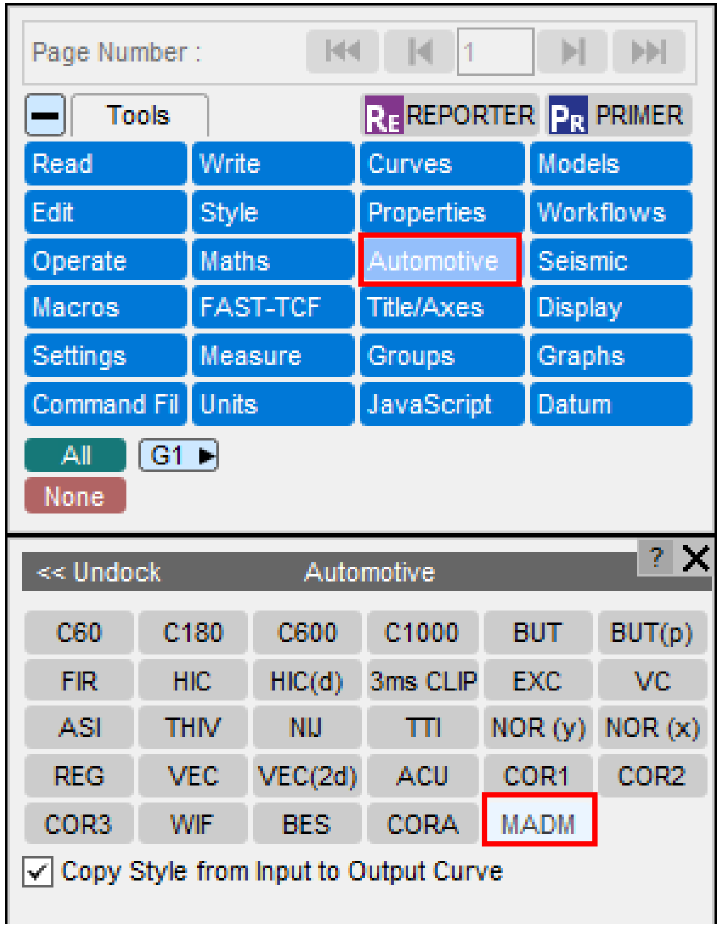
# Calling the MADM correlation tool in a script

# Curve correlation

T/HIS provides a number of ways to determine a measure for the degree to which two curves match i.e. correlation.

Typically, one curve is a reference curve that might come from physical testing, the other is the comparison curve that may be the result of a simulation of the scenario that generated the reference curve.

T/HIS implements a number of curve correlation methods. This tutorial focuses on the MADM correlation tool.



# Introduction

The MADM (Minimum Area Discrepancy Method) correlation tool can be used for correlations between LS-DYNA simulations and physical tests, particularly for force-displacement data. It does not need time-history data. For more information on how to use the tool interactively, watch this short video (YouTube, 2m53s).

This guide focuses on how MADM can be called from another script. This can be useful for automating a process in which you can extract data from multiple models and compare their correlation ratings.

This guide includes snippets of the code, which you can find in your installation directory, under $OA_INSTALL/manuals/tutorials/this/**cora_and_madm_example_files**.

# Overall steps

This tutorial will show how a JavaScript can read in a model, extract certain time-history curves, operate on them as necessary, then combine them together into a force-displacement curve. MADM can then be called, and this combined curve can be used as the “simulation” curve in the MADM script.

Steps:

1. Load experimental test data
2. Extract simulation results comparable to test data
3. Filter + combine test and simulation time histories
4. Perform correlation calculations (automated)
    a. Brief explanation of JSON configuration file

# JSON configuration files

A JSON configuration file can be loaded and saved into the interactive GUI, or used in a script to directly calculate its MADM rating. More details on the configuration file inputs can be found in the T/HIS manual.

The configuration file can specify settings for producing input curves for the MADM rating, and for calculating the MADM rating. It can also specify where output curves, images, or output .json files should be written out to. All parameters configurable in the interactive tool (GUI) have a corresponding parameter.

# Example

# Example file package

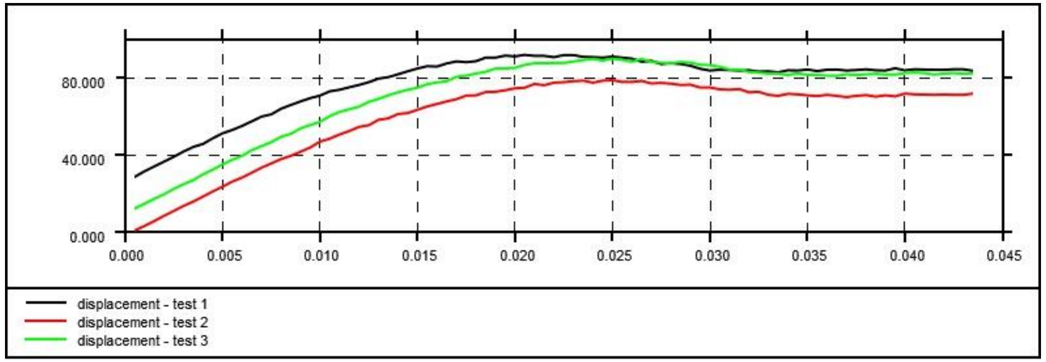
The example file package ($OA_INSTALL/manuals/tutorials/this/**cora_and_madm_example_files**) contains the following folders and files:

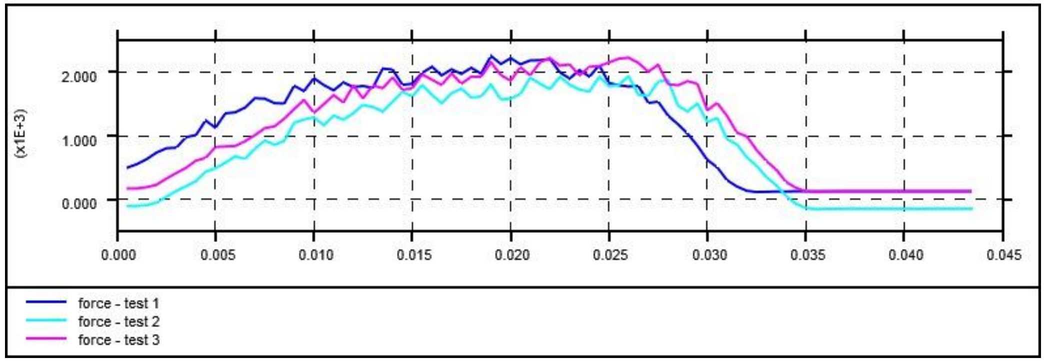
- The **analysis_model** folder contains an example LS-DYNA analysis model with results
- The **test_data** folder contains force and displacement time histories measured during testing
- The files **workflow_test.js** and **workflow_test_input.json** are the JavaScript and configuration file associated with this example.

```
.
├── analysis_model
│     ├── binout0000
│     ├── bouncer.key
│     ├── bouncer.ptf
│     ├── bouncer.ptf01
│     ├── bouncer.ptf02
│     ├── bouncer.thf
│     └── bouncer.ztf
│     └── bouncer.cur
├── test_data
│     └── bouncer.cur
├── workflow_test.js
└── workflow_test_input.json
```

# 1. Load experimental test data

The test data is read into T/HIS from the supplied curve file: ./test_data/bouncer.cur.

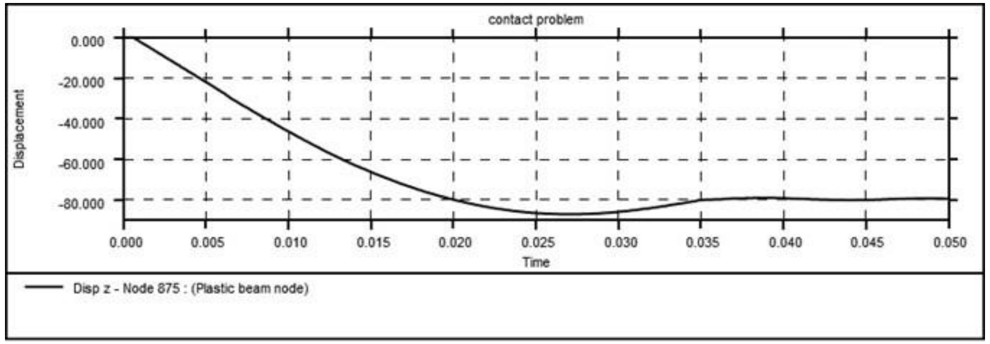


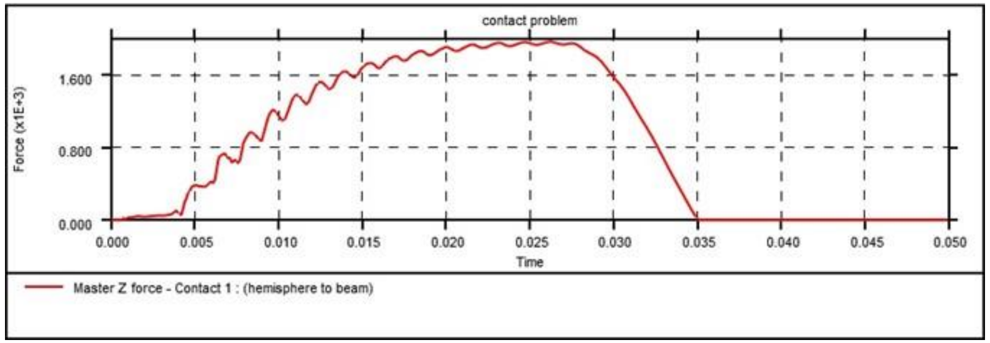




```
// salient file paths as global variables
var install_dir = GetInstallDirectory();
var script_dir = JSPath(arguments[0]);
var module_file_path = install_dir+'/this_library/scripts/CORA/modules/MADM.mjs';
var model_file_path = script_dir + "analysis_model/bouncer.thf";
var test_file_path = script_dir + "test_data/bouncer.cur";
var madm_config_file = script_dir + "workflow_test_input.json";

// // extract curves from test data for reference curves
// read in displacement-time and force-time curves, and assign the curve IDs to arrays
// 3 tests - force and disp i.e. 6 curves total
// note: this assumes you know the contents + order of the .cur file
var start_id = Curve.FirstFreeID();
Read.Cur(test_file_path);
var dt_ids = [start_id, start_id + 1, start_id + 2];
var ft_ids = [start_id + 3, start_id + 4, start_id + 5];
```

# 2. Extract simulation results comparable to test data

We can read in the model from the example model *.thf* file and extract the appropriate time-history data. See Appendix J in the manual for more information on operating on model objects.

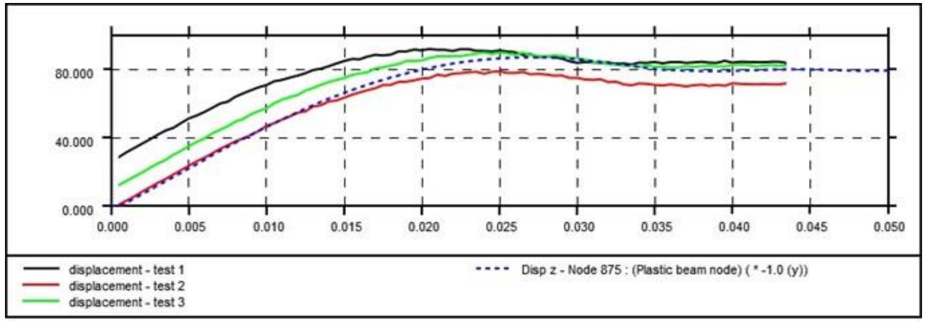


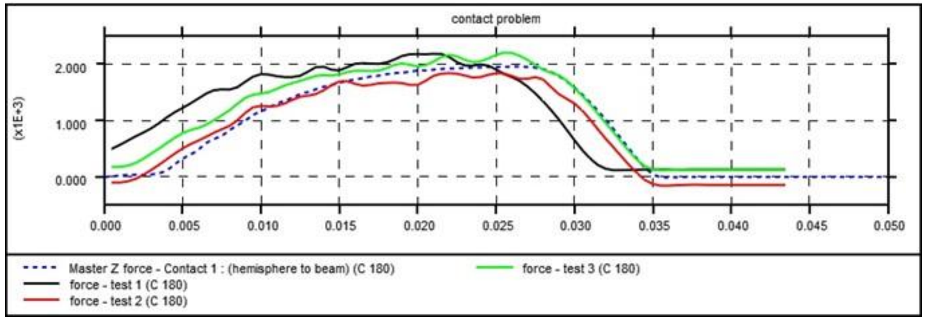




```
// // extract curves from analysis model for simulation curve
// read in model
var m = Model.Read(model_file_path);

// first, extract displacement-time curve
var f_d = AllocateFlag();
m.SetFlag(f_d, Entity.NODE, -875);
var cur_d = m.GetDataFlagged(f_d, Component.DZ)[0];

// then, extract force-time curve
var f_f = AllocateFlag();
m.SetFlag(f_f, Entity.CONTACT, 1);
var cur_f = m.GetDataFlagged(f_f, Component.CFZ)[0];
```

# 3. Filter + combine test and simulation time histories

The simulation displacement direction is reversed to match the test data and both simulation force data is passed through a C180 filter. We can then combine the test force-time and displacement-time curves into a force-displacement curve.




```
// flip displacement time

var cur_d_flipped = Operate.Mul(cur_d, -1);


// filter force-time

var cur_f_filtered = Operate.C180(cur_f)


// then combine displacement-time with force-time + store curve ID

var cur_d_f = Operate.Com(cur_d_flipped, cur_f_filtered).id;
```

# 4. Perform correlation calculations (automated)

T/HIS

After setting the appropriate curve IDs as the curve inputs, the MADM correlation analysis can then be performed.

```
// run the correlation analysis. Note: madm function is defined below.
madm();

async function madm()
{
    // import the required cora functions from the t/his library
    const { calculateMADM, readjson } = await import(module_file_path);

    // set up input object from JSON configuration file
    var MADM_input = readjson(madm_config_file);

    // set input curves as the appropriate curve IDs
    MADM_input.curve_inputs.sim = sim_id;
    MADM_input.curve_inputs.dt = dt_ids;
    MADM_input.curve_inputs.ft = ft_ids;

    // calculate MADM rating!
    calculateMADM(MADM_input);
}
```

**Note:** In the above code snippet and those previous, dynamic module imports are being used to allow flexibility for the installation location variable. If the install location is known, the above could more simply be implemented as:

```
// note: import statement(s) should be the first lines of code in a script
import {calculateMADM, readjson } from ‘<path_to_install_dir>/this_library/scripts/CORA/modules/cora.mjs';
var MADM_input = readjson(madm_config_file);
<… curve ID setting …>
calculateMADM(MADM_input);
```

# 4a. Brief explanation of JSON configuration file

T/HIS

```
{
    "method":"generateAverage",
    "curve_inputs": {
        "avg":null,
        "lower":null,
        "upper":null,
        "sim": null,
        "dt": null,
        "ft": null
    },
    "n": 1,
    "m": 2,
    "offset": null,
    "half_width": 0.06,
    "output_json":"json_out.json",
    "output_curves":"curves_out.cur",
    "output_image":"image_out.png"
}
```

As the method being used generates average/lower/upper curves, they do not need to be provided, as they will be generated with the provided deflection-time + force-time curves, and half-width value.

There are placeholder null values for the “sim”, “dt” and “ft” input curves.

The half-width value of the generated corridor in standard deviations is 0.06.

An output configuration file, output curves, and an output image will all be generated in the same folder as the script.

# Contact us

Oasys
LS-DYNA Environment

## Global / UK

T: +44 121 213 3399
E: dyna.support@arup.com

## India

T: +91 40 69019723 / 98
E: india.support@arup.com

## China

T: +86 21 3118 8875
E: china.support@arup.com

## USA

T: +1 415 940 0959
E: us.support@arup.com

Subscribe to our newsletter:

Follow us on:

@Oasys LS-DYNA Environment

@Oasys LS-DYNA Environment

@Oasys

@Oasys

www.oasys-software.com/dyna/