CERN

Management of Material Properties for Superconducting Magnet

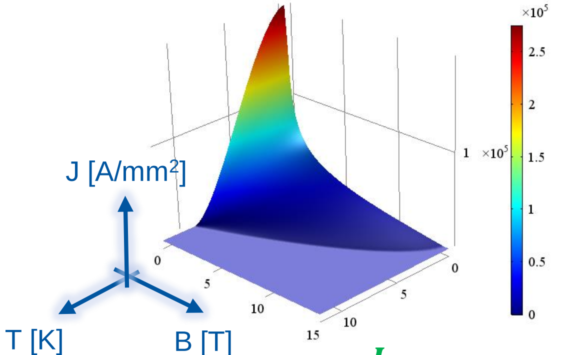
Youssef Raouyane

Introduction

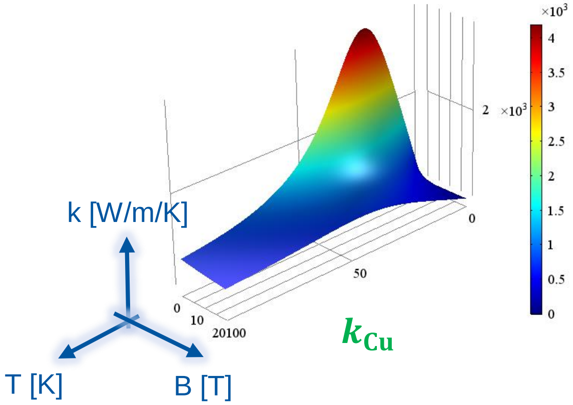
1. There are numerous material properties used in superconducting magnet modelling
2. The goal of this work is to summarize existing material properties in C
3. The material properties in C are then compared with equivalent in MATLAB and available references
4. To this end, an automated testing routine was developed in Matlab
5. This would allow to link model with material property (very important for co-simulation)

Material Properties in C

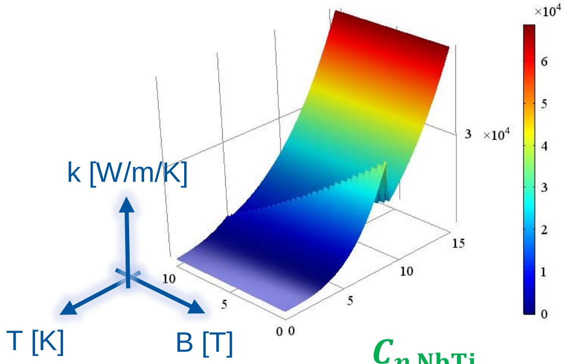
- Materials C library (versioning, storage)
- C compiler: Visual C++ Build Tools
- Dynamic Link Library (immutability)
- COMSOL external function



$J_{c\ \mathrm{NbTi}}$



k_{Cu}



$C_{p\ \mathrm{NbTi}}$

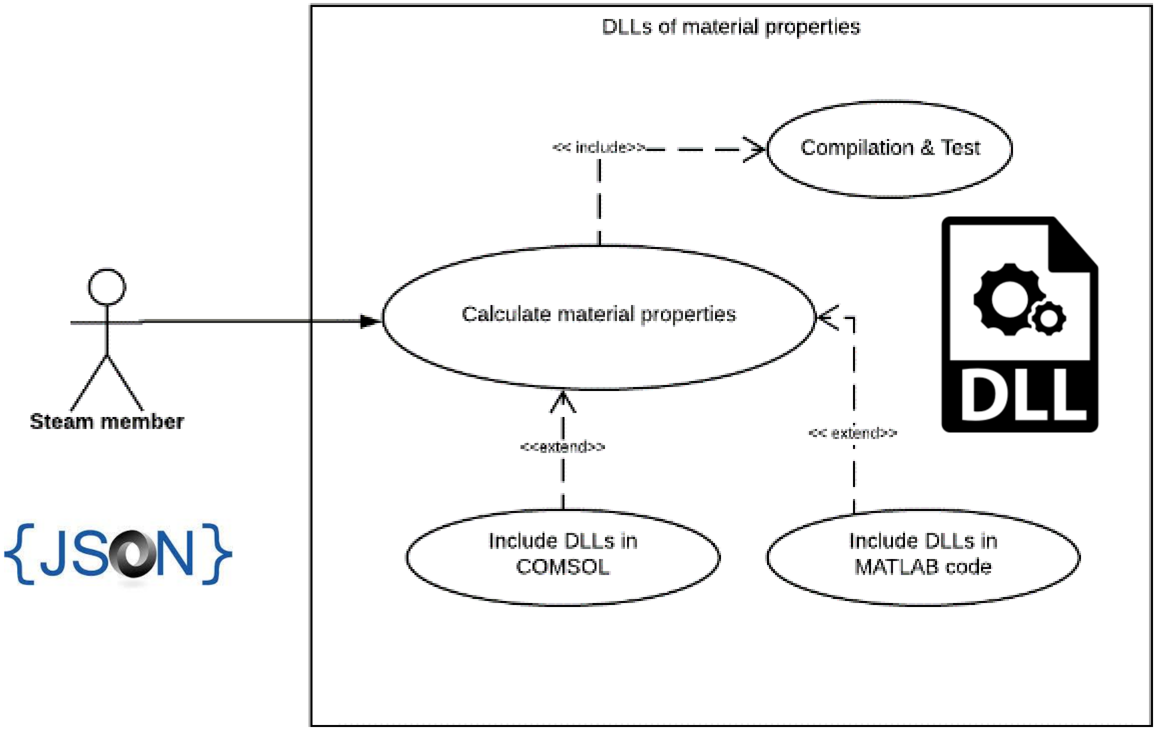
Definition

A DLL is a library that contain compiled code and data that can be used by more than one program at the same time.

Advantages :

- DLLs can be used by multiple programs at the same time
- DLLs are much smaller (one copy of the file in the physical memory)
- Faster compilation time

Use case diagram



{JSON}

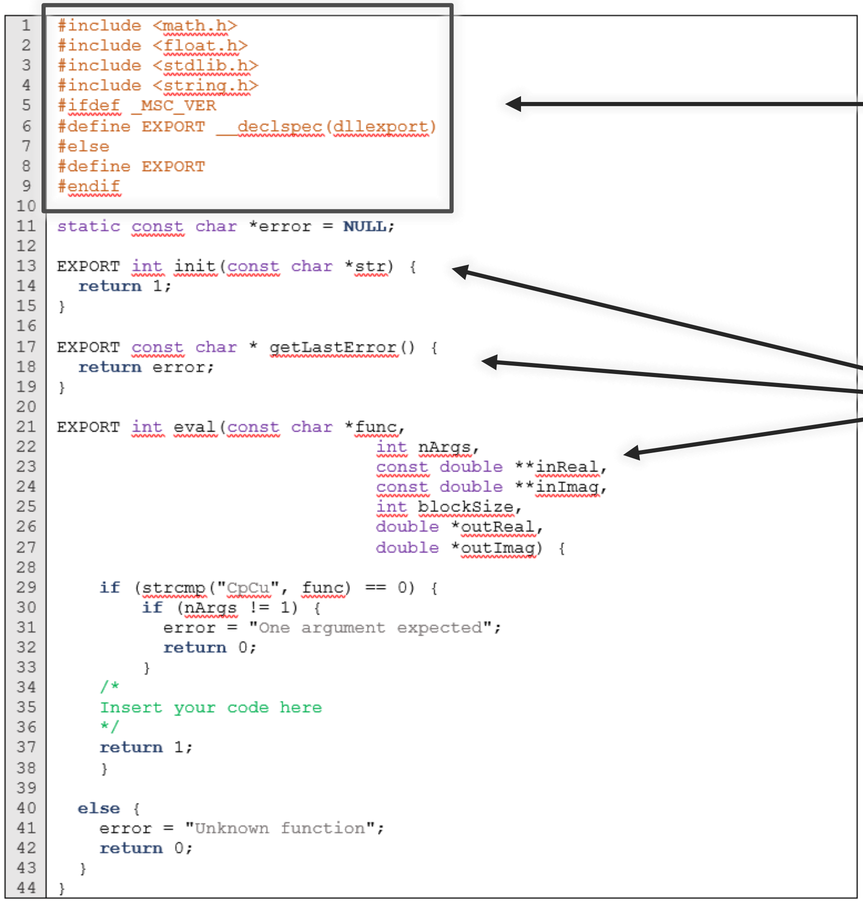
GitLab

MATLAB

COMSOL

Include – user has to take an action
Extend – user may take an action

C-function structure required by COMSOL



```c
#include <math.h>
#include <float.h>
#include <stdlib.h>
#include <string.h>
#ifdef _MSC_VER
#define EXPORT __declspec(dllexport)
#else
#define EXPORT
#endif

static const char *error = NULL;

EXPORT int init(const char *str) {
  return 1;
}

EXPORT const char * getLastError() {
  return error;
}

EXPORT int eval(const char *func,
                              int nArgs,
                              const double **inReal,
                              const double **inImag,
                              int blockSize,
                              double *outReal,
                              double *outImag) {

    if (strcmp("CpCu", func) == 0) {
        if (nArgs != 1) {
          error = "One argument expected";
          return 0;
        }
    /*
    Insert your code here
    */
    return 1;
    }

  else {
    error = "Unknown function";
    return 0;
  }
}
```

- C-header with library includes

- Init : Initialize the DLL in COMSOL

- getLastError : return the last error

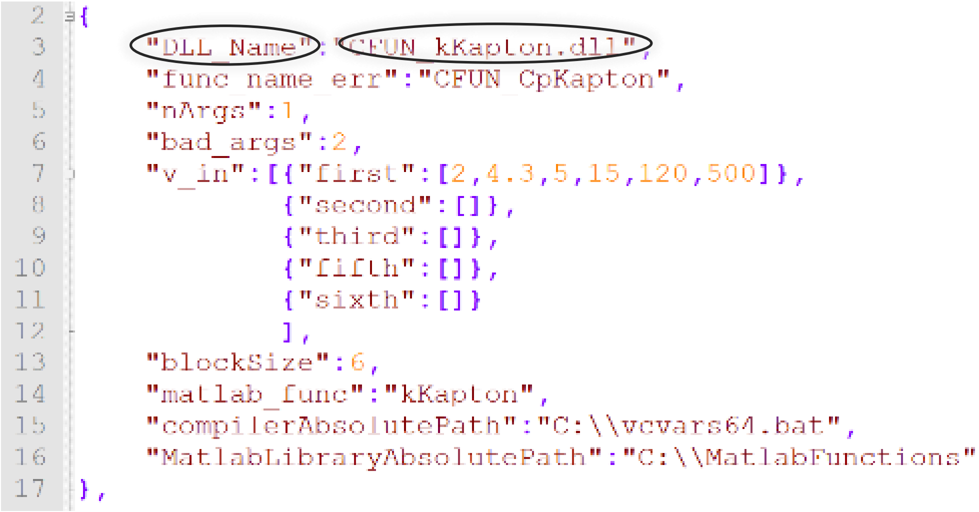
- Eval : Main function

More information :

L. Bortot, USE OF EXTERNAL C FUNCTIONS IN COMSOL MULTIPHYSICS, p. 11

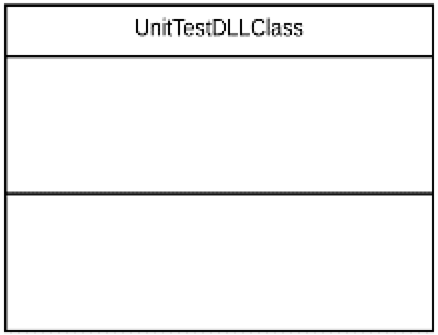
JavaScript Object Notation

- Structured and stored data
- Text based files and human readable



```
{
    "DLL_Name":"CFUN_kKapton.dll",
    "func_name_err":"CFUN_CpKapton",
    "nArgs":1,
    "bad_args":2,
    "v_in":[{"first":[2,4.3,5,15,120,500]},
            {"second":[]},
            {"third":[]},
            {"fifth":[]},
            {"sixth":[]}
            ],
    "blockSize":6,
    "matlab_func":"kKapton",
    "compilerAbsolutePath":"C:\\vcvars64.bat",
    "MatlabLibraryAbsolutePath":"C:\\MatlabFunctions"
},
```

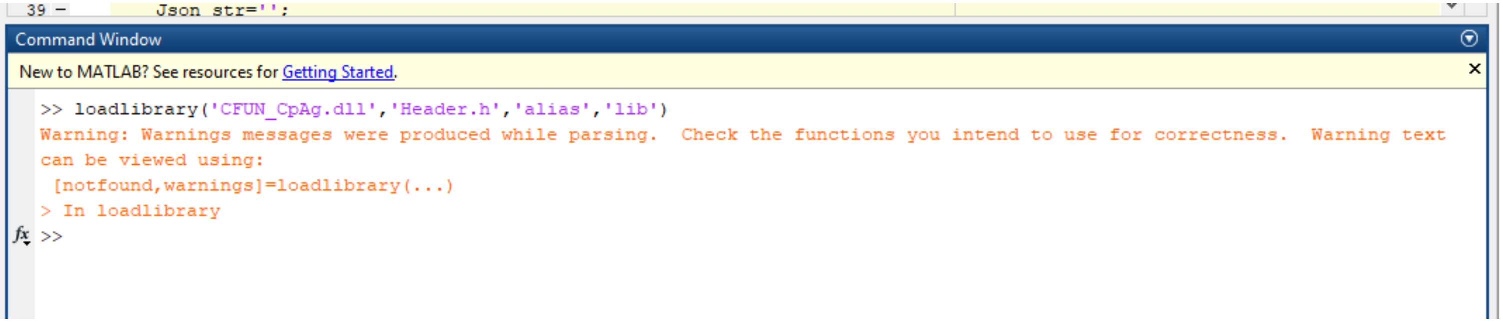
Testing code



UnitTestDLLClass

- Read the block corresponding to the DLL from the JSON file
- Load the DLL
- Check if it contains all the functions
- Call the library using invalid inputs to check if the error messages are caught
- Compare the output values with the output of the equivalent MATLAB function

How to load the DLL ?



```
Json_str='';
```

Command Window

New to MATLAB? See resources for Getting Started.

```
>> loadlibrary('CFUN_CpAg.dll','Header.h','alias','lib')
Warning: Warnings messages were produced while parsing.  Check the functions you intend to use for correctness.  Warning text
can be viewed using:
 [notfound,warnings]=loadlibrary(...)
> In loadlibrary
>>
```

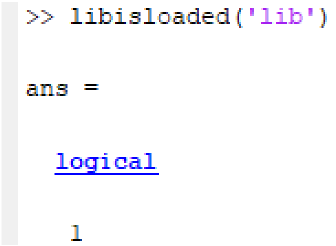
```
Loadlibrary( DLLNAME, HEADER)
```

DLLNAME as char
Header name as char

full path or the name if we are in the same directory

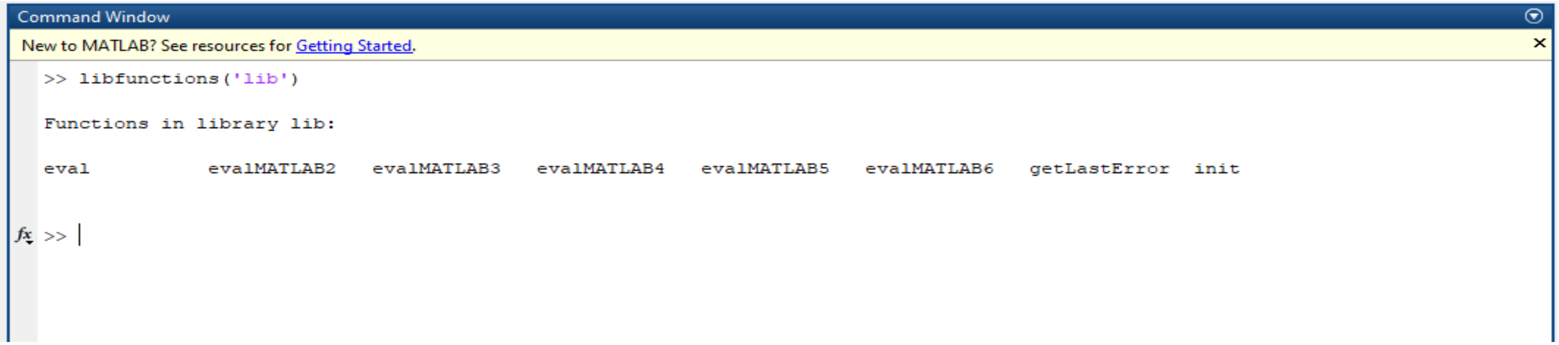
Useful functions

- **Libisloaded : verify if the DLL is loaded**



```
>> libisloaded('lib')

ans =

  logical

   1
```

- **Libfunctions: display the DLL functions**



New to MATLAB? See resources for Getting Started.

```
>> libfunctions('lib')

Functions in library lib:

eval           evalMATLAB2   evalMATLAB3   evalMATLAB4   evalMATLAB5   evalMATLAB6   getLastError  init

>> 
```

Use the library (1/3)

Main function :

```
int eval(const char *func,
                    int nArgs,
                    const double **inReal,
                    const double **inImag,
                    int blockSize,
                    double *outReal,
                    double *outImag)
```

- Func : The name of the function
- nArgs : The number of arguments required

See: https://espace.cern.ch/steam/_layouts/15/start.aspx#/SitePages/Material%20Properties.aspx

Use the library (2/3)

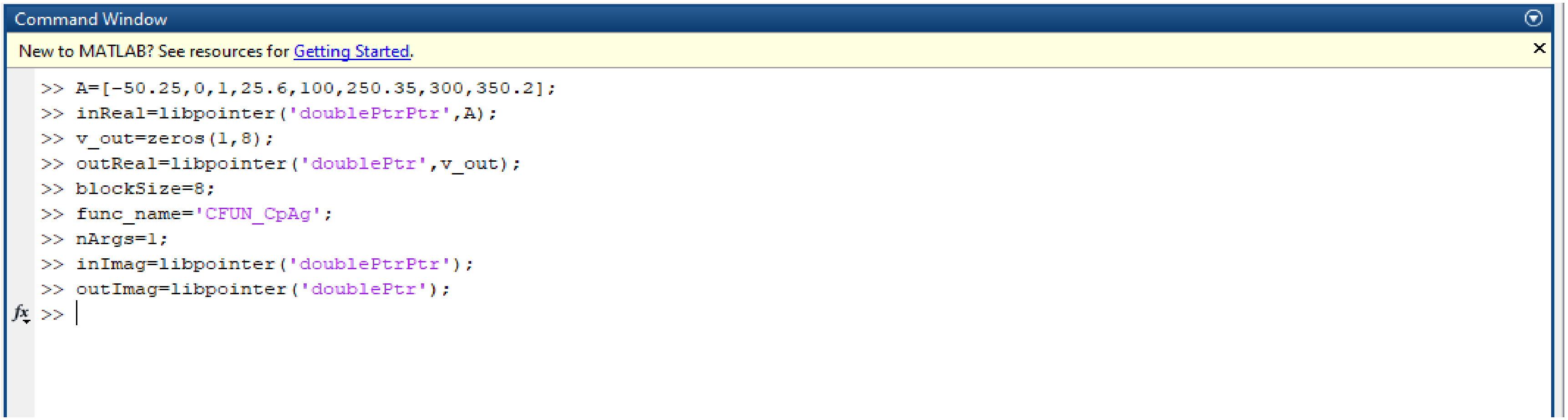
Main function :

```
int eval(const char *func,
         int nArgs,
         const double **inReal,
         const double **inImag,
         int blockSize,
         double *outReal,
         double *outImag)
```

- InReal: Pointer of array that contain the evaluation points
- InImag : as Inreal for imaginary inputs
- blockSize : dim(InReal)
- outReal : Pointer of array for output values
- outImag: as outReal for imaginary values

See: https://espace.cern.ch/steam/_layouts/15/start.aspx#/SitePages/Material%20Properties.aspx

Use the library (3/3)



```
>> A=[-50.25,0,1,25.6,100,250.35,300,350.2];
>> inReal=libpointer('doublePtrPtr',A);
>> v_out=zeros(1,8);
>> outReal=libpointer('doublePtr',v_out);
>> blockSize=8;
>> func_name='CFUN_CpAg';
>> nArgs=1;
>> inImag=libpointer('doublePtrPtr');
>> outImag=libpointer('doublePtr');
>> 
```

Libpointer: Create a pointer of arrays

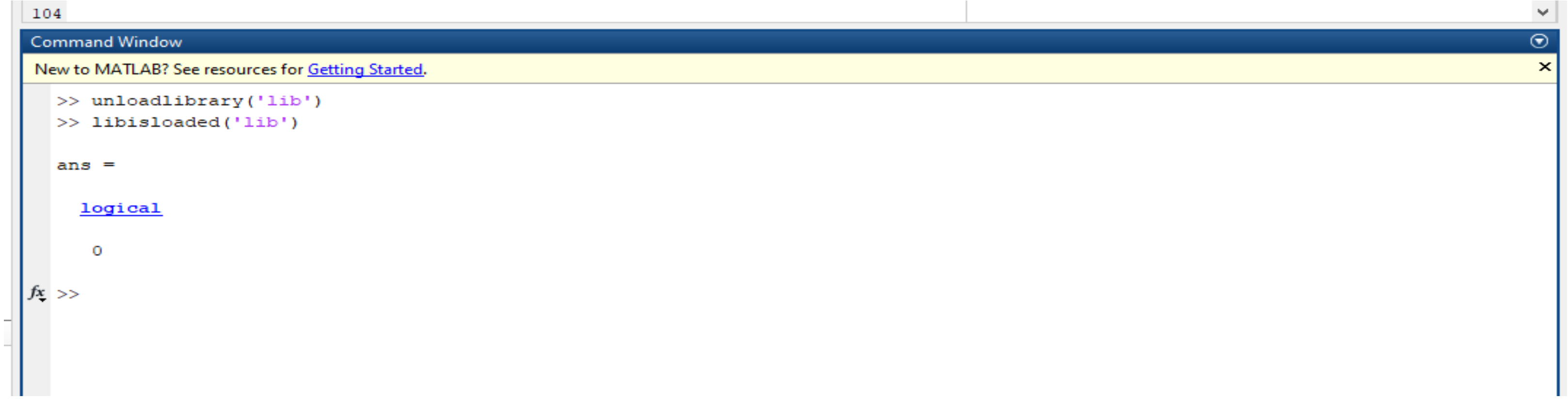
Libpointer(‘datatype’, initialValue);

Example :

```
>> A=[-50.25,0,1,25.6,100,250.35,300,350.2];
>> inReal=libpointer('doublePtr',A);
```

inReal is a pointer to A, its type is doublePtr

Unload the library

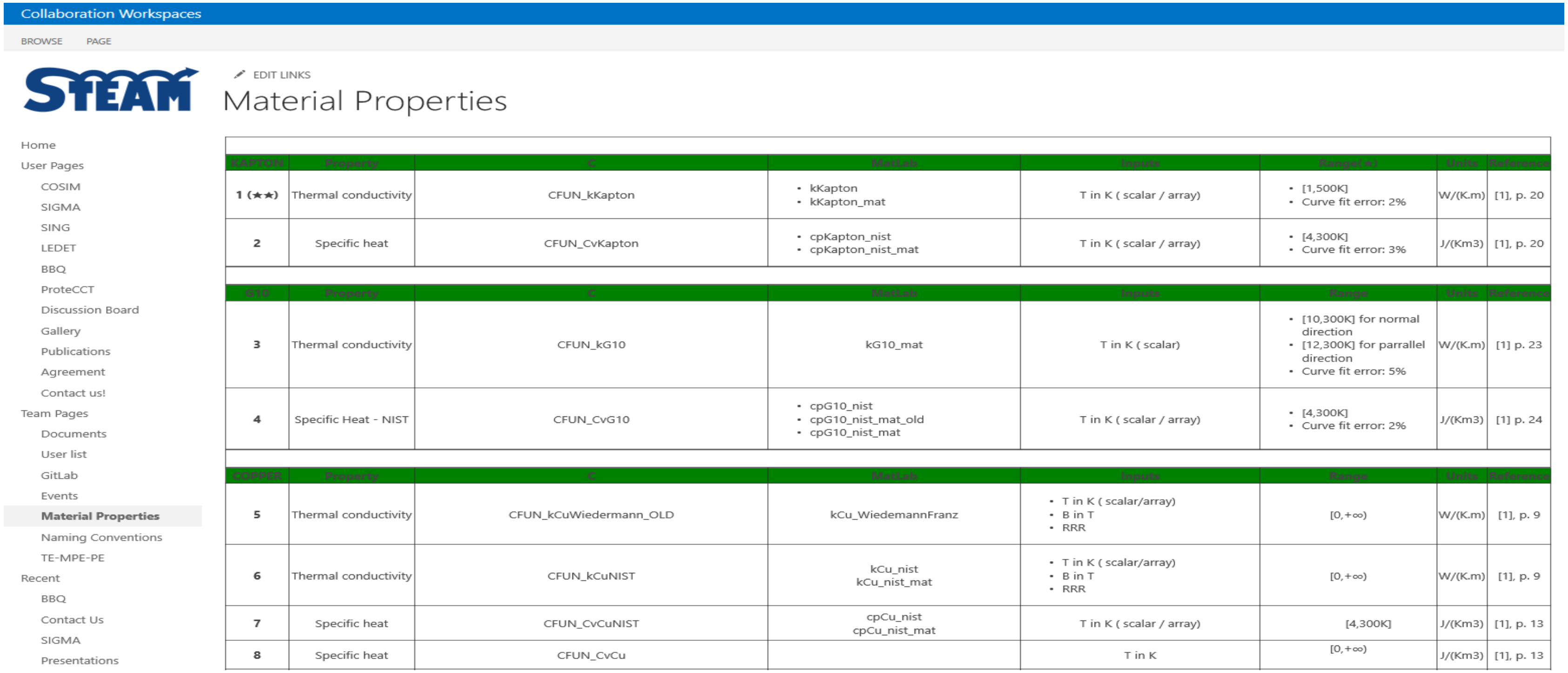


New to MATLAB? See resources for Getting Started.

```
>> unloadlibrary('lib')
>> libisloaded('lib')

ans =

  logical

   0

>>
```

`Unloadlibrary(DLLNAME)`

Documentation (1/2)



Collaboration Workspaces

BROWSE PAGE

EDIT LINKS

Material Properties

- Home
- User Pages
  - COSIM
  - SIGMA
  - SING
  - LEDET
  - BBQ
  - ProteCCT
  - Discussion Board
  - Gallery
  - Publications
  - Agreement
  - Contact us!
- Team Pages
  - Documents
  - User list
  - GitLab
  - Events
  - **Material Properties**
  - Naming Conventions
  - TE-MPE-PE
- Recent
  - BBQ
  - Contact Us
  - SIGMA
  - Presentations

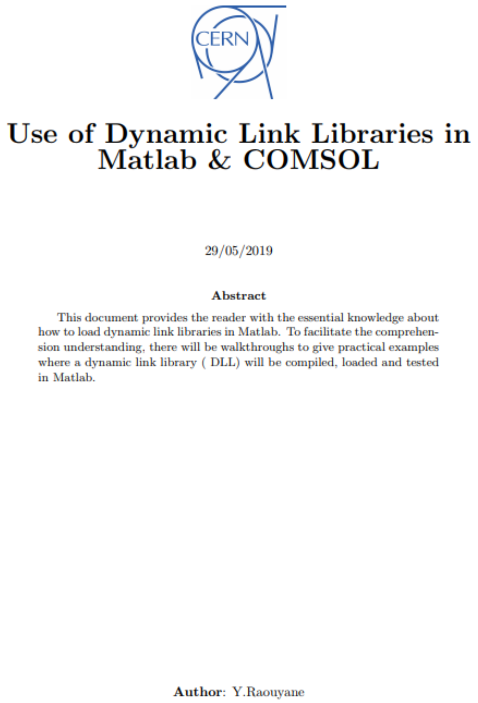
KAPTON	Property	C	MatLab	Inputs	Range(★)	Units	Reference
1 (★★)	Thermal conductivity	CFUN_kKapton	• kKapton • kKapton_mat	T in K (scalar / array)	• [1,500K] • Curve fit error: 2%	W/(K.m)	[1], p. 20
2	Specific heat	CFUN_CvKapton	• cpKapton_nist • cpKapton_nist_mat	T in K (scalar / array)	• [4,300K] • Curve fit error: 3%	J/(Km3)	[1], p. 20

G10	Property	C	MatLab	Inputs	Range	Units	Reference
3	Thermal conductivity	CFUN_kG10	kG10_mat	T in K (scalar)	• [10,300K] for normal direction • [12,300K] for parrallel direction • Curve fit error: 5%	W/(K.m)	[1] p. 23
4	Specific Heat - NIST	CFUN_CvG10	• cpG10_nist • cpG10_nist_mat_old • cpG10_nist_mat	T in K (scalar / array)	• [4,300K] • Curve fit error: 2%	J/(Km3)	[1] p. 24

COPPER	Property	C	MatLab	Inputs	Range	Units	Reference
5	Thermal conductivity	CFUN_kCuWiedermann_OLD	kCu_WiedemannFranz	• T in K (scalar/array) • B in T • RRR	[0,+∞)	W/(K.m)	[1], p. 9
6	Thermal conductivity	CFUN_kCuNIST	kCu_nist kCu_nist_mat	• T in K (scalar/array) • B in T • RRR	[0,+∞)	W/(K.m)	[1], p. 9
7	Specific heat	CFUN_CvCuNIST	cpCu_nist cpCu_nist_mat	T in K (scalar / array)	[4,300K]	J/(Km3)	[1], p. 13
8	Specific heat	CFUN_CvCu		T in K	[0,+∞)	J/(Km3)	[1], p. 13

- Web page in The STEAM Website: **https://espace.cern.ch/steam/SitePages/Material%20Properties.aspx**
- A Table that contain name of the functions in C & MATLAB, inputs required and the range of validity

Documentation (2/2)



Use of Dynamic Link Libraries in Matlab & COMSOL

29/05/2019

Abstract

This document provides the reader with the essential knowledge about how to load dynamic link libraries in Matlab. To facilitate the comprehension understanding, there will be walkthroughs to give practical examples where a dynamic link library (DLL) will be compiled, loaded and tested in Matlab.

Author: Y.Raouyane

- Detailed description of how to compile, test & use the DLLs in both MATLAB and COMSOL

Conclusion

- All available material properties were summarized and cross-checked
- Some differences between C and Matlab implementations were identified and will be clarified
- The automated testing framework is completed and can be extended by adding new material properties
- You can find a summary here: **https://espace.cern.ch/steam/_layouts/15/start.aspx#/SitePages/Material%20Properties.aspx**

CERN