Mechanics of Defect Evolution Library

Generated by Doxygen 1.8.9.1

Sat Jan 24 2015 12:34:45

Contents

Chapter 1

The MODEL library

1.1 Introduction

MODEL is the The Mechanics of Defect Evolution Library, a collection of free software for the study of defects in materials.

MODEL contains the following modules:

- Discrete Dislocation Dynamics (DDD)
- Particle Interaction Library (PIL)

1.2 Installation

MODEL is a header-only library, therefore you just need to have MODEL in your include path.

MODEL is written in C++ and is heavily object-oriented. In particular, some of the latest features of the C++11 standard are used. Therefore MODEL can only be compiled with C++11-capable compilers. The following compilers should do the job:

- GNU g++ version 4.8 or above (available at `http://gcc.gnu.org`)
- INTEL icc version 13.0.1 or above

Are you testing MODEL on `Mac OsX`? Then make sure you read the dedicated Mac osX page.

1.2.1 External libraries

The Discrete Dislocation Dynamics module of MODEL requires the following external libraries:

- the c++ standard library
- the boost library (header only, available at `http://www.boost.org/`)
- the Eigen library (header only, available at `http://eigen.tuxfamily.org`)

In order to run MODEL-DDD in parallel mode, additional libraries are needed:

- the open-MPI protocol
- the METIS partitioner

1.2.2 Other useful packages

Some bash scripts used for post-processing of pictures and videos are:

- `ImageMagick`
- `FFmpeg`

These are standard packages on most Unix/Linux distributions.

Meshes are generated using the following library:

- `TetGen`

See the dedicated Generating Meshes with Tetgen page for sample use.

Chapter 2

Class Documentation

2.1 model::PlanarSplineImplicitization< polyDegree > Class Template Reference

Class template that determines the implicit equation det $(\mathbf{M}_x X + \mathbf{M}_y Y + \mathbf{M}_c) = 0$ of a planar spline of arbitrary degree and computes intersections with other splines of also arbitrary degree.

`#include <PlanarSplineImplicitization.h>`

Public Member Functions

- PlanarSplineImplicitization (const Eigen::Matrix< double, 2, polyCoeff > &coeffs_in)
- template<short unsigned int otherPolyDegree>
  std::set< std::pair< double, double > > intersectWith (const Eigen::Matrix< double, 2, otherPolyDegree+1 > &otherCoeffs) const

Public Attributes

- EIGEN_MAKE_ALIGNED_OPERATOR_NEW const Eigen::Matrix< double, 2, polyCoeff > coeffs
- const MatrixPolyDeg Mx
- const MatrixPolyDeg My
- const MatrixPolyDeg Mc

Static Public Attributes

- static double compTol =FLT_EPSILON
- static double physTol =DBL_MAX

2.1.1 Detailed Description

template<short unsigned int polyDegree>class model::PlanarSplineImplicitization< polyDegree >

Class template that determines the implicit equation det $(\mathbf{M}_x X + \mathbf{M}_y Y + \mathbf{M}_c) = 0$ of a planar spline of arbitrary degree and computes intersections with other splines of also arbitrary degree.

Adapted from "Algorithms for Intersecting Parametric and Algebraic Curves", Manocha, 1994.

Definition at line 34 of file PlanarSplineImplicitization.h.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 template<short unsigned int polyDegree> model::PlanarSplineImplicitization< polyDegree >::PlanarSplineImplicitization (const Eigen::Matrix< double, 2, polyCoeff > & *coeffs_in*) [inline]

Consider a planar spline in explicit form:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \sum_{i=0}^{P} \begin{bmatrix} a_i \\ b_i \end{bmatrix} t^i$$

where P is the degree of the spline. In order to derive the corresponding implicit equation of the spline, co

$$F(t) = a_i t^i - X$$
$$G(s) = b_i s^i - Y$$

and observe that each point (X, Y) on the spline makes $F(t_0)$ and $G(s_0)$ vanish for some common root $t_0 = s_0$. Now consider the bivariate polynomial:

$$P(t,s)\frac{F(t)G(s) - F(s)G(t)}{t-s} = s^i \left(M_{xij}X + M_{yij}Y + M_{0ij}\right) t^j$$

If t_0 is a common root then $P(t_0, s) = 0\ \forall s$. This is possible only if $\left(M_{xij}X + M_{yij}Y + M_{0ij}\right) t_0^j = 0$. Therefore for non-trivial solutions we need:

$$det\left(\mathbf{M}_x X + \mathbf{M}_y Y + \mathbf{M}_c\right) = 0$$

which is the implicit equation of the spline.

We show that $P(t,s)\frac{F(t)G(s)-F(s)G(t)}{t-s} = s^i \left(M_{xij}X + M_{yij}Y + M_{0ij}\right) t^j$ and give expressions for M_{xij}, M_{yij} and M_{cij}. Substituting the expressions for F_t and G_s:

$$P(t,s) = X\sum_{j=0}^{P} b_j \frac{t^j - s^j}{t-s} + Y\sum_{i=0}^{P} a_i \frac{s^i - t^i}{t-s} + \sum_{i=0}^{P}\sum_{j=0}^{P} \frac{a_i b_j t^i s^j - a_i b_j s^i t^j}{t-s}$$

Noticing that some terms vanish identically:

$$P(t,s) = X\sum_{j=1}^{P} b_j \frac{t^j - s^j}{t-s} + Y\sum_{i=1}^{P} a_i \frac{s^i - t^i}{t-s} + a_0\sum_{j=1}^{P} \frac{b_j s^j - b_j t^j}{t-s} + b_0\sum_{i=1}^{P} \frac{a_i t^i - a_i s^i}{t-s} + \sum_{i=1}^{P}\sum_{j=1}^{P} \frac{a_i b_j t^i s^j - a_i b_j s^i t^j}{t-s}$$

Now, recalling that $t^j - s^j = (t-s)\sum_{k=0}^{j-1} t^{j-1-k}s^k$, we obtain, for the first term:

$$X\sum_{j=1}^{P} b_j \frac{t^j - s^j}{t-s} = X\sum_{j=1}^{P} b_j \sum_{k=0}^{j-1} t^{j-1-k}s^k = X\sum_{l=0}^{P-1} b_{l+1}\sum_{k=0}^{l} t^{l-k}s^k = X\sum_{k=0}^{P-1}\sum_{l=k}^{P-1} b_{l+1}t^{l-k}s^k = X\sum_{k=0}^{P-1}\sum_{m=0}^{P-1-k} b_{m+k+1}t^m s^k = X\sum_{k=0}^{P-1}$$

where in the last step we used $b_k = 0$ for $k > N$. Analogously:

$$a_0\sum_{j=1}^{P} \frac{b_j s^j - b_j t^j}{t-s} = -a_0\sum_{k=0}^{P-1}\sum_{m=0}^{P-1} b_{m+k+1}t^m s^k$$

$$b_0\sum_{i=1}^{P} \frac{a_i t^i - a_i s^i}{t-s} = b_0\sum_{k=0}^{P-1}\sum_{m=0}^{P-1} a_{m+k+1}t^m s^k$$

$$P(t,s) = s^i \left(M_{xij}X + M_{yij}Y + M_{0ij}\right) t^j$$

We now give explicit expression for the matrices

$$\mathbf{M}_x = \begin{bmatrix} b_1 & b_2 & \dots & b_n \\ b_2 & \ddots & b_n & 0 \\ \vdots & b_n & 0 & 0 \\ b_n & 0 & 0 & 0 \end{bmatrix} \qquad M_{xij} = \begin{cases} b_{i+j+1} & i+j+1 \leq N \\ 0 & i+j+1 > N \end{cases}$$

$$\mathbf{M}_y = -\begin{bmatrix} a_1 & a_2 & \dots & a_n \\ a_2 & \ddots & a_n & 0 \\ \vdots & a_n & 0 & 0 \\ a_n & 0 & 0 & 0 \end{bmatrix} \qquad M_{yij} = \begin{cases} -a_{i+j+1} & i+j+1 \leq N \\ 0 & i+j+1 > N \end{cases}$$

$$\mathbf{M}_c = -a_0 \begin{bmatrix} b_1 & b_2 & \dots & b_n \\ b_2 & \ddots & b_n & 0 \\ \vdots & b_n & 0 & 0 \\ b_n & 0 & 0 & 0 \end{bmatrix} + b_0 \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ a_2 & \ddots & a_n & 0 \\ \vdots & a_n & 0 & 0 \\ a_n & 0 & 0 & 0 \end{bmatrix}$$

PlanarSplineImplicitization: HERE WE SHOULD ASSER THAT THIS SPLINE IS NOT DEGENERATE !!!!!!

Definition at line 81 of file PlanarSplineImplicitization.h.

2.1.3 Member Function Documentation

2.1.3.1 template<short unsigned int polyDegree> template<short unsigned int otherPolyDegree> std::set<std::pair<double,double> > model::PlanarSplineImplicitization< polyDegree >::intersectWith (const Eigen::Matrix< double, 2, otherPolyDegree+1 > & *otherCoeffs*) const [inline]

Computes the intersection points between the implicitized spline and another spline defined by the coefficients otherCoeffs.

Parameters

in	*otherCoeffs*	the matrix of coefficients of the other spline

{Theoretical Background}. Assume that the other spline has degree Q. The 2xQ coefficient matrix of the other spline defines the curve:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \sum_{i=0}^{Q} \begin{bmatrix} c_i \\ d_i \end{bmatrix} s^i$$

Substituting in the implicit form of the original spline we obtain:

$$det\left(\mathbf{M}_x \sum_{i=0}^{Q} c_i s^i + \mathbf{M}_y \sum_{i=0}^{Q} d_i s^i + \mathbf{M}_c\right) = 0$$

or, grouping powers of s:

$$det\left[(c_0\mathbf{M}_x + d_0\mathbf{M}_y + \mathbf{M}_c) + \sum_{i=1}^{Q} (c_i\mathbf{M}_x + d_i\mathbf{M}_y)\, s^i\right] = det\left[\mathbf{M}_0 + \sum_{i=1}^{Q} \mathbf{M}_i s^i\right] = 0$$

where we defined $\mathbf{M}_0 = c_0\mathbf{M}_x + d_0\mathbf{M}_y + \mathbf{M}_c$ and $\mathbf{M}_i = c_i\mathbf{M}_x + d_i\mathbf{M}_y$. Intersection points are therefore the roots of the above equation. We now transform the root finding problem into and eigenvalue problem. For this observe that if

$det\left[\mathbf{M}_0 + \sum_{i=1}^{Q} \mathbf{M}_i s^i\right] = 0$ then

$$\left[\mathbf{M}_0 + \sum_{i=1}^{Q} \mathbf{M}_i s^i\right] \mathbf{v}_0 = \mathbf{0}$$

has non-trivial solution. We now introduce the vectors $\mathbf{v}_{i-1} = s^{i-1}\mathbf{v}_0$ and obtain:

$$-\mathbf{M}_0 \mathbf{v}_0 = s \sum_{i=1}^{Q} \mathbf{M}_i \mathbf{v}_{i-1}$$

which, together with the recursive relation for the $\mathbf{v}_i$'s, leads to the generalized eigenvalue problem:

$$\begin{bmatrix} -\mathbf{M}_0 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{N-1} \end{bmatrix} = s \begin{bmatrix} \mathbf{M}_1 & \mathbf{M}_2 & \dots & \mathbf{M}_N \\ \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{N-1} \end{bmatrix}$$

So we obtained the generalized eigenvalue problem:

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{B}\mathbf{x}$$

where:

$$\mathbf{A} = \begin{bmatrix} -\mathbf{M}_0 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} \end{bmatrix}$$

is a symmetric matrix (since it's a linear combination of symmetric matrices) and

$$\mathbf{B} = \begin{bmatrix} \mathbf{M}_1 & \mathbf{M}_2 & \dots & \mathbf{M}_N \\ \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix}$$

Note that $det(\mathbf{B}) = det(\mathbf{M}_N)$ and that

$$\mathbf{B}^{-1} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{M}_N^{-1} & -\mathbf{M}_N^{-1}\mathbf{M}_1 & \dots & -\mathbf{M}_N^{-1}\mathbf{M}_{N-1} \end{bmatrix}$$

Therefore if ${M}_N$ one can also solve:

$$\mathbf{C}\mathbf{x} = \lambda \mathbf{x}$$

where:

$$\mathbf{C} = \mathbf{B}^{-1}\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \\ -\mathbf{M}_N^{-1}\mathbf{M}_0 & -\mathbf{M}_N^{-1}\mathbf{M}_1 & \dots & -\mathbf{M}_N^{-1}\mathbf{M}_{N-1} \end{bmatrix}$$

Algorithm is: 1- Assemble the generalized eigenvalue problem At=sBt

2- Solve generalized eigenvalue problem At=sBt

4- Return the intersections

Definition at line 217 of file PlanarSplineImplicitization.h.

2.1.4 Member Data Documentation

2.1.4.1 template<short unsigned int polyDegree> EIGEN_MAKE_ALIGNED_OPERATOR_NEW const Eigen::Matrix<double,2,polyCoeff> model::PlanarSplineImplicitization< polyDegree >::coeffs

Definition at line 71 of file PlanarSplineImplicitization.h.

2.1.4.2 template<short unsigned int polyDegree> double model::PlanarSplineImplicitization< polyDegree >::compTol =FLT_EPSILON [static]

Definition at line 76 of file PlanarSplineImplicitization.h.

2.1.4.3 template<short unsigned int polyDegree> const MatrixPolyDeg model::PlanarSplineImplicitization< polyDegree >::Mc

Definition at line 74 of file PlanarSplineImplicitization.h.

2.1.4.4 template<short unsigned int polyDegree> const MatrixPolyDeg model::PlanarSplineImplicitization< polyDegree >::Mx

Definition at line 72 of file PlanarSplineImplicitization.h.

2.1.4.5 template<short unsigned int polyDegree> const MatrixPolyDeg model::PlanarSplineImplicitization< polyDegree >::My

Definition at line 73 of file PlanarSplineImplicitization.h.

2.1.4.6 template<short unsigned int polyDegree> double model::PlanarSplineImplicitization< polyDegree >::physTol =DBL_MAX [static]

Definition at line 77 of file PlanarSplineImplicitization.h.

The documentation for this class was generated from the following file:

- PlanarSplineImplicitization.h