MFree: An Explicit Meshfree Code for Non-Linear Solid Mechanics release 0.1

Stephen Smith
Queen's University Belfast

2018
October

Contents

Chapter 1

Introduction

1.1 MFree

The Mfree libary is a framework developed in go-lang for mesh-free modelling in a research environment. It was developing during my PhD for application to stretch blow moulding a manufacturing technique used to produce polymer bottles. The main features of the code are the ability to simulate non-linear solid mechanics problems using an explicit solver. The use of the MFree library is intended to simplify the use of mesh-free methods within a research context, and further provide a learning resource for meshfree methods.

To facilitate this goal, this manuscript has been created in order to describe the main features of this code. The code has been intended to be designed around the idea of the interaction between objects, similar to the object oriented style of programming, without the complexities of inheritance and polymorphism present in other langauges such as C++ and Java. Hence, it is the aim of this code is to provide a balance between a user friendly black-box (similar to Abaqus) and a tool for researchers within computational mechanics.

The go-lang langauge has been chosen as it provides inbuilt conncurncy along with a readability, and usability often not found in other high performance langauges (C, Fortran, C++).

1.2 Overview

Chapter 2

Theory

In order to understand how to use the library it is necessary to have an appreciation of the theory behind the application of meshfree methods to non-linear solid mechanics. This chapter explains the fundamental ideas of meshfree methods, and discretization of a continous problem in a discrete, and numerically solvable one.

2.1 Meshfree methods

Meshfree methods were developed in the middle of the 1990's to overcome difficulties associated with the finite element method. The fundamental problem in meshfree or finite elements is to provide a set of approximation function (subject to a set of constraints) that can be used as an approximation space for the trial and test functions in the weak form of a partial different equation. To illustrate this problem, consider the balance of linear momentum, cast in a Lagrangian(reference) form Eq. (2.1). The intention in this manuscript is to use capital symbols to describe material coordinates, however often irregularities will exist between this intention and the result.

$$GRAD\ P_{iJ} = \rho_0 \ddot{u}_i \tag{2.1}$$

where P is the first Piola-Kirchoff stress, ρ_0 the material density in the reference frame (typically $t = 0$) and $\ddot{u}$ the acceleration. To complete the balance of linear moemntum boundary conditions must be specified. A boundary is called a displacemnt boundary, denoted Γ_u, if a displacement u is prescribed on that boundary, likewise a similar defintion is present for the traction

boundary, denoted Γ_t The boundary conditions for Eq. (2.1) are typically given as:

$$u = \bar{u} \quad on \quad \Gamma_u \tag{2.2}$$

$$P_{ij}N_j = \bar{t_i} \quad on \quad \Gamma_t \tag{2.3}$$

Equation 2, subject to the boundary conditions (2.2,2.3) is often termed the strong form

Strong form of the momentum balance

$GRAD\ P_{iJ} = \rho_0 \ddot{u}_i$	Momentum Balance
$u = \bar{u} \quad on \quad \Gamma_u$	Displacement condition
$P_{ij}N_j = \bar{t_i} \quad on \quad \Gamma_t$	Traction condition

Strong form to weak form

Chapter 3

Design

3.1 Domain

The domain package is intended to hold together the geometric details of the problem, i.e the nodes, the intergration cells and the degrees of freedom attached to the domain. From this other packages will refrence this domain, such as the meshfree packaage, which builds the meshfree domain, and the integration package. At present there are two ways to build a domain, manually by adding nodes to a domain object, or more simply by providing a planar-straight line graph (PLSG) to the domain constructor, given by:

```
domain := domain.DomainNew(fileName string, options string, dim
    int, global_coordinate coordinateSystem)
```

Where $fileName$ is the name of the PLSG, *options* provides a set of rules the mesh, and voronoi generator, see `https://www.cs.cmu.edu/~quake/triangle.html` for a description. The variable *dim* ensures that correct number of degrees of freedom(DOFs) will be set. The coordinate system can be generated using the following function (for Cartesian coordinates),

```
globalCS := coordinatesystem.CreateCartesian()
```

axisymmetric (cylindrical coordinates) are also supported. This function will construct the nodes, the degrees of freedom (assuming they are all free to start with) and the Voronoi diagram, which is used in the stabalised nodal integration scheme (SCNI).

Chapter 4

Usage

4.1 Domain

Chapter 5

Furture Plans

Chapter 6

Area of Improvements