Towards a Material Point Method with Powell-Sabin spline basis functions

MPM with higher-order C^1-continuous basis functions on triangulations

Pascal B.J. de Koster

Delft University of Technology

TU Delft Delft University of Technology

Deltares Enabling Delta Life

Challenge the future

Towards a Material Point Method with Powell-Sabin spline basis functions

MPM with higher-order C^1-continuous basis functions on triangulations

Pascal B.J. de Koster
4302508

The Netherlands, Delft, July 2018

Thesis for the Master degree
Master of Science in Applied Mathematics

Delft University of Technology
Delft Institute of Applied Mathematics,
Section of Numerical Analysis

To be defended at 14 August, 2018

Supervisors TU Delft:	Dr. M. Möller
	Ir. R.P.W.M. Tielen
Supervisor Deltares:	Dr. V. Galavi
Thesis Committee:	Dr. M. Möller
	Prof.dr.ir. C. Vuik
	Dr. V. Galavi
	Dr.ir. W.T. van Horssen

TU Delft Delft University of Technology

Deltares Enabling Delta Life

ABSTRACT

The material point method (MPM) is a numerical method for simulating ground deformations, that combines the benefits of a fixed grid with moving material points, which carry the history dependent information of the continuum. During each time step, this information is projected to the grid, on which the equations of motion are solved and a corresponding acceleration field is constructed. Then, the material points undergo motion corresponding to the acceleration field and the particle properties are updated accordingly. MPM uses a finite element method to solve the equations of motion over a triangular grid, but in current versions, the basis functions used are only piece-wise linear, which results in low-order spatial convergence for function reconstruction. Furthermore, the piece-wise linear basis functions have discontinuous first derivatives, which cause so called grid-crossing errors when a material point crosses between elements, where these discontinuities are occur. To solve these problems for 2D MPM on a triangular grid, this study investigates the possibility to use higher-order basis functions with the properties of C^1-continuity and non-negativity. B-spline basis functions would be very suitable for this purpose and earlier studies for 1D have already shown the potential of B-spline MPM. This thesis shows that it is also possible to use B-splines on a 2D triangular grid, which has led to the use of quadratic Powell-Sabin (PS) spline basis functions, which are piece-wise quadratic, C^1-continuous, non-negative basis functions on triangulations. In this thesis, we have implemented these basis functions in MPM and investigated the performance of PS-spline MPM. Although Constructing and evaluating PS-splines has proven more cumbersome than the piece-wise linear basis functions classically used in MPM, but PS-spline MPM shows higher-order spatial convergence for function reconstruction and completely erases grid crossing errors. For these reasons, PS-spline MPM would be very suitable for application in engineering in which large numbers of basis functions are necessary: due to the higher-order spatial convergence of PS-spline MPM, the number of required basis functions could be significantly reduced.

Before implementation in engineering application can take place, however, two issues still appear in PS-spline MPM, which should be further researched. The first issue is that PS-spline MPM shows instabilities when elements become nearly empty, as the mass matrix becomes ill-conditioned when this occurs. A way to mitigate this problems is by lumping the mass matrix, however, this leads to the second issue. When the mass matrix is lumped, the accuracy of the solution drops significantly, far more than when classical MPM with piece-wise linear basis functions is lumped. If these issues are resolved in further research, PS-spline MPM may prove very suitable for engineering applications.

CONTENTS

1

INTRODUCTION

In geomechanics, predicting the behaviour of large soil deformations can be very difficult because the resulting geometry of the solid can be drastically different from the original geometry. Typical examples include landslides, tunnel collapses and pile driving [1].

In order to investigate these problems, it is possible to simulate such scenarios numerically. One of the most common techniques for these simulations is the updated Lagrangian finite element method, which uses basis functions over a grid to reconstruct the material properties over the domain of the continuum. However, this technique has great difficulty dealing with large deformations. In this method, the grid moves along with the material, which makes it easy to track its properties. However, large deformations will often lead to grid distortion, resulting in large numerical errors or non-physical solutions. On the other hand, when using a fixed grid and having the continuum flow through it, it becomes hard to track properties of the material due to non-linear convective terms that come along with such a description [2].

In order to overcome these difficulties, the material point method (MPM) was invented by Sulsky et al. around 1994 [3]. This method combines a discretisation of the continuum using material points with a fixed background grid. Since a fixed grid is used, it will not get distorted, while material points move along with the continuum and keep track of its properties. At each time step, the information of the material points is projected onto the grid, where the equations of motion are solved and the acceleration field is reconstructed over the grid. Finally, the acceleration is projected back onto the material points, which undergo the corresponding motion and the particle properties are updated accordingly. This way, the method combines the advantages of having a fixed grid, but is still able to track the history dependent properties of the material. To project the continuum properties from the particle points to the grid, the property distributions are reconstructed as a linear combination of basis functions, which are defined in correspondence with the grid.

In order to find accurate solutions for two-dimensional problems with arbitrary geometries, it is desirable to use a triangular grid. Such a grid can be easily refined at places of interest and can approximate any arbitrary geometry. Therefore, this thesis will focus on MPM on a triangular grid. The typical choice for basis functions over triangulations is the set of piece-wise linear Lagrangian basis functions. However, this basis is only second order spatial convergent, which means that a fine grid is required to get a reliable solution. This again results in long computation times. Since problems in MPM can have thousands up to millions of degrees of freedom, computation time and accuracy become a serious problem.

Another drawback of the piece-wise linear Lagrangian basis functions is the occurrence of grid-crossing errors. Grid crossing errors appear due to the discontinuity in the gradient of the Lagrangian basis functions over the edges of the elements. In most versions of MPM, this error can be mitigated, for example as described by Al-Kafaji [1], but to completely erase this error, C^1 continuous basis functions should be used.

In this thesis, we investigate if the use of higher-order B-spline functions on triangular grids in MPM can mitigate these problems. In previous research, instead of higher-order B-splines, the use of higher-order Lagrangian basis functions in MPM has been researched, but this basis turns out to be unsuitable due to negativity of the basis functions. It is preferable to use non-negative basis functions within the material point method, because else the mass matrix used for calculating the solution will have negative entries. In several cases, this

may already lead to numerical instabilities and non-physical results [4], as negative entries represent a negative mass. Furthermore, negative entries prevent the mass matrix from being lumped, whereas lumping is very important for quickly solving large systems of equations. Therefore, using non-negative basis functions is of great importance. However, all higher-order Lagrangian basis function contain negative parts. Finally, higher-order Lagrangian basis function also have discontinuous gradients, so grid crossing errors still pose a problem. For these reasons, higher-order Lagrangian basis functions are not suitable for MPM.

As an alternative to a Lagrangian basis, this thesis studies the use of a basis of a Powell-Sabin (PS) splines, which are C^1-continuous non-negative B-splines on arbitrary triangulations. Earlier studies on the use of a B-spline basis for one-dimensional MPM showed great potential, see [5, 6]: higher-order spatial convergence was achieved and grid crossing errors no longer occurred. PS-spline basis functions are also expected to have the same benefits for two-dimensional MPM.

The goal of this thesis is to implement the Powell-Sabin spline basis functions into MPM, and investigate if these basis function lead to the expected higher-order convergence and solve the grid crossing error. This research is executed in cooperation with Deltares, a research institute that investigates geomechanical problems with MPM. The main application of interest for Deltares is pile driving for off-shore platforms for wind-turbines. The current MPM version of Deltares uses piece-wise linear Lagrangian basis functions on triangular grids, in which a mitigation for grid-crossing errors is included. A B-spline based MPM could lead to a more accurate and faster method and completely solve the problem of grid crossing as well. To research this, Powell-Sabin spline MPM is validated by solving benchmarks problems, and comparing these to their analytic solution. From these benchmarks, the convergence and stability of PS-spline MPM is studied.

The outline of this study is as follows. First the equations of motion are presented and the material point method is introduced for solving these equations in Chapter 2. In Chapter 3, the weaknesses of Lagrange basis functions are discussed and to replace these functions, higher-order Powell-Sabin spline basis function are introduced and constructed. In Chapter 4 Lagrangian basis MPM and PS-spline MPM are compared in terms of spatial convergence and general accuracy. Finally, in Chapter 5, the most pressing open issues of PS-spline MPM is discussed: PS-spline MPM is prone to instabilities when handling near-empty elements. Some mitigation techniques are proposed and investigated to improve stability of PS-spline MPM.

2

MATERIAL POINT METHOD FOR SOIL DEFORMATION PROBLEMS

In this chapter, the material point method is discussed in detail. There are several versions of MPM, but in this thesis the formulation as originally described by Sulsky et al. [7] is used, although we follow the comprehensive step-by-step implementation by Al-Kafaji [1]. This is also the method used by Tielen [5], on whose work this thesis is building, and Deltares follows this procedure as well.

First the equations of motion for a continuum undergoing deformations are presented. Next, the basic concept of MPM will be explained in Section 2.2. Then in Section 2.3 the spatial discretisation is discussed. In Section 2.4, the full algorithm for a single time step in MPM is presented. Finally, we also discuss the grid crossing error in Section 2.7, as this error is an important incentive to use spline basis functions in MPM.

For the reader who is already familiar with the material point method and the associated grid crossing error, we recommend immediately moving on to Chapter 3, in which the construction of the higher-order, C^1-continuous, Powell-Sabin spline basis functions is explained.

2.1. GOVERNING EQUATIONS

In this section, we will present the equations of motion that we will solve using the material point method. The equations will not be derived in this thesis, but a full derivation can be found in [1].

2.1.1. LAGRANGIAN FRAME OF REFERENCE

In continuum mechanics, different frames of reference can be used to formulate the equations of motion. The two most common ones are the Lagrangian and the Eulerian frame of reference. In both frames, a control volume is considered and within this control volume, equations can be derived for the velocity, density, internal force and other properties. Within the Eulerian frame of reference, the control volume is kept stationary, and thus the continuum moves into and out of the control volume. The mass contained within the control volume can change over time. However, within the Lagrangian frame of reference, the control volume typically contains the same part of the continuum all the time, and moves along with it. Therefore, its mass is constant over time, but the control volume will change shape and its volume may change as well. An illustration of the Eulerian and Lagrangian frames of reference can be found in Figure 2.1. In MPM, particles are used to represent the continuum, and each particle contains the same piece of the continuum at all time steps and tracks the properties of it over time. This corresponds with a Lagrangian frame of reference.

A large advantage of this frame of reference is that the equations of motion simplify much. In general, when a quantity f is considered in the a control volume, its change over time is described by the material derivative:

$$\frac{\mathrm{d}f(\boldsymbol{x},t)}{\mathrm{d}t} = \frac{\partial f}{\partial t} + \boldsymbol{v}\cdot\nabla f, \tag{2.1}$$

in which t denotes the time, $\boldsymbol{x}$ the position, $\boldsymbol{v}$ the velocity and ∇f the gradient of f. The change in time in the control volume is due to two parts: a partial derivative with respect to time, and an advection term. For example, we may consider the temperature inside a control volume. The temperature can change due to heating of the particles inside the volume. This corresponds to the partial derivative with respect to time in

Equation 2.1. Furthermore, the temperature may also change because particles flow into the control volume on one side and out on the opposite side. The material flowing in may have a different temperature than the material flowing out. This change is equal to the flow speed multiplied with the gradient of the temperature in the direction of the flow, $\boldsymbol{v} \cdot \nabla f$. This yields the advection term in Equation 2.1. Now, consider the fact that in MPM the Lagrangian frame of reference is used. That is, the control volumes moves along with the material. Therefore, these control volumes will contain the same material independent of time and no particles flow into or out of the control volume. Therefore, the advective term can be dropped and the material derivative reduces to the partial derivative for all purposes in MPM [8]:

$$\frac{\mathrm{d}}{\mathrm{d}t} = \frac{\partial}{\partial t}. \tag{2.2}$$

This greatly simplifies the equations of motion and is one of the main reasons to use MPM over other finite element methods.

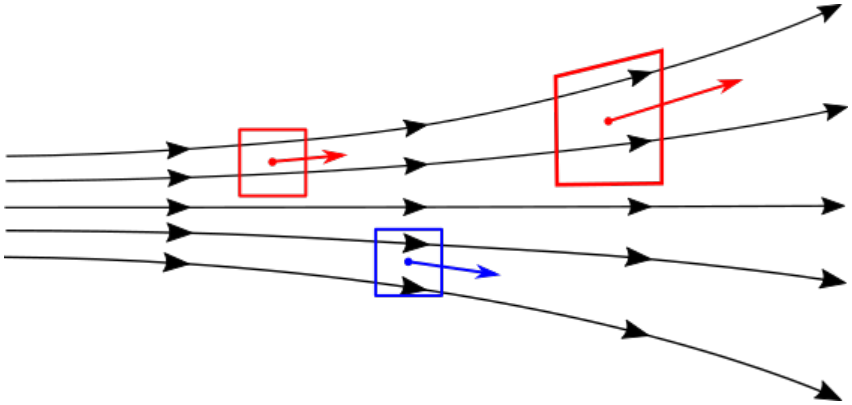


Figure 2.1: Illustration of the Lagrangian (upper, red) and Eulerian (lower, blue) frame of reference in 2D. The control volumes in the Eulerian frame of reference are stationary and material can go in and out. The Lagrangian frame of reference always contains the same material and moves along with it.

2.1.2. Equations of motion

For describing the equations of motion of a continuum, it is common that both conservation of mass and conservation of momentum are considered. However, conservation of mass is already guaranteed for MPM, as particles are used that represent the grid. These particles contain the some part of the continuum at all times, so the mass of each particle does not change and therefore conservation of mass is guaranteed. The equations of motion for use in MPM will be based only on conservation of momentum:

$$\rho \frac{\partial \boldsymbol{v}}{\partial t} = \nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{g}. \tag{2.3}$$

In this equation, $\boldsymbol{v} = \frac{\partial \boldsymbol{x}}{\partial t}$ denotes the velocity vector, $\nabla \cdot \boldsymbol{\sigma}$ the divergence of the stress tensor, ρ the density, and g the acceleration vector, which is -9.81 in the y-direction for the 2D benchmarks in this thesis, unless mentioned otherwise. Note the fact that in the above equation, only partial derivatives to the time are used; this is due to the use of a Lagrangian frame of reference, reducing all the material derivatives to partial derivatives as in Equation 2.2.

The stress tensor is a measure for the internal force in a material. A differential equation for the stress tensor $\boldsymbol{\sigma}$ is given below, relating the stress tensor to the strain tensor $\boldsymbol{\varepsilon}$ through a constitutive relation [1, 9]. The strain is a measure for the deformation of the continuum, indicating compression, stretching and shear deformation. More general information about the stress and strain tensors, and the corresponding motion can be found in Appendix A.1. The Einstein summation convention is used for repeated indices:

$$\frac{\partial \sigma_{ij}}{\partial t} = D_{ijkl} \frac{\partial \epsilon_{kl}}{\partial t} - \frac{\partial \epsilon_{kk}}{\partial t} \sigma_{ij} + \omega_{ik} \sigma_{kj} - \sigma_{ik} \omega_{kj}, \tag{2.4}$$

The term D_{ijkl} and the spin tensor ω_{ij} need to be further defined. The spin tensor ω_{ij} accounts for the rotational acceleration of the continuum, and is given by

$$\omega_{ij} = \frac{1}{2} \left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i} \right). \tag{2.5}$$

The term D_{ijkl} gives a linearised relation between stress and strain in Equation 2.4, derived from Hooke's law. The term is given by

$$D_{ijkl} = \left(K - \frac{2}{3}G\right)\delta_{ij}\delta_{kl} + G\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right). \tag{2.6}$$

The δ_{ij} denotes the Kronecker delta, which is 1 if $i = j$ and zero otherwise. K and G represent the bulk modulus and shear modulus respectively; the bulk modulus is the ratio between pressure and the corresponding decrease of volume, and the shear modulus is the ratio between the shear strain and the resulting shear stress. The bulk modulus and shear modulus are both determined by the Poisson ratio ν and the Young's modulus E, which are material properties that are assumed to be known:

$$K = \frac{E}{3(1-2\nu)} \quad \text{and} \quad G = \frac{E}{2(1+\nu)}. \tag{2.7}$$

The Poisson ratio ν describes the ratio between transversal expansion to axial compression. That is, when a volume gets compressed in one direction, it will generally expand in directions orthogonal to the compression. The rate at which this happens is the Poisson ratio. The Young's modulus is the ratio between axial stress and the corresponding axial strain.

Finally, a differential equation for the incremental strain is given by

$$\frac{\partial \epsilon_{ij}}{\partial t} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right). \tag{2.8}$$

The differential equations for the velocity, stress and strain in Equations 2.3, 2.4-2.7 and 2.8 respectively together form a system of differential equations that describe the movement of a continuum. With initial and boundary conditions, the problem will be fully described.

2.1.3. Initial and Boundary Conditions

In order to fully define a problem, the initial and boundary conditions are required. Let the domain of interest be Ω, and its boundary $\partial\Omega$, see Figure 2.2. Depending on the problem of interest, it is well possible that the geometry of the continuum changes. Therefore the boundary and the domain may also be dependent of time. The boundary is split in two parts, $\partial\Omega_u$ and $\partial\Omega_\tau$. On $\partial\Omega_u$, the displacement is prescribed,

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{U}(t) \quad \text{on} \quad \partial\Omega_u(t). \tag{2.9}$$

This corresponds to a Dirichlet boundary condition. On $\partial\Omega_\tau$ the surface traction is prescribed,

$$\sigma_{ij}(\boldsymbol{x}, t)n_j = \tau_i(t) \quad \text{on} \quad \partial\Omega_\tau(t), \tag{2.10}$$

where n_j denotes the j^{th} component of the unit vector, normal to the boundary pointing outwards. Prescribed traction corresponds to a Neumann boundary condition, although in this thesis, only homogeneous traction boundary condition are considered in the benchmarks. Together $\partial\Omega_u$ and $\partial\Omega_\tau$ are assumed to make up the entire boundary. These parts are not allowed to overlap, only one of the two boundary conditions can be active at a time.

Initial conditions are required for each of the equations described in differential form. The displacement and velocity require initial conditions, and we also require an initial condition for the stress, as the magnitude of the stress is used in Equation 2.4. The strain is only required in differential from, and therefore, we do not need an initial condition for this. The initial conditions are thus given by

$$\boldsymbol{u}(\boldsymbol{x}, t_0) = \boldsymbol{U_0}(\boldsymbol{x}), \quad \boldsymbol{v}(\boldsymbol{x}, t_0) = \boldsymbol{V_0}(\boldsymbol{x}), \quad \text{and} \quad \sigma_{ij}(\boldsymbol{x}, t_0) = \sigma^0_{ij}(\boldsymbol{x}). \tag{2.11}$$

2.1.4. Weak Form

For numerically solving the momentum conservation equation every discrete time step, MPM uses a finite element approach, solving the momentum equation in its weak form. The weak form the momentum equation is easiest evaluated component wise: first evaluate the acceleration in the x-direction, second in the y-direction.

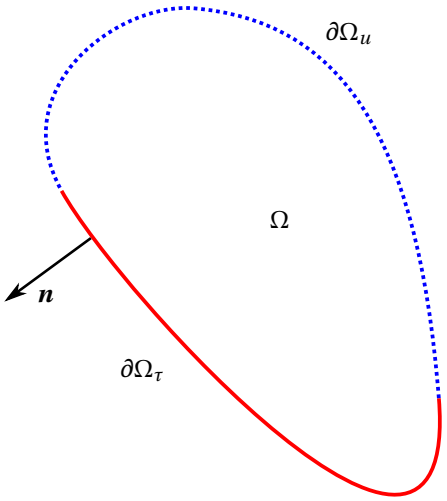


Figure 2.2: A domain Ω with its boundary $\partial\Omega$. The boundary is divided in two parts. On the dotted part $\partial\Omega_u$, the displacement is prescribed, and on the solid part $\partial\Omega_\tau$, the traction is prescribed. $\boldsymbol{n}$ marks an outwards pointing normal vector.

Let a_k denote the acceleration in the x_k direction. The momentum equation is multiplied by a test function ϕ from a test space Φ, and then integrated over the domain.

$$\begin{aligned} \rho a &= \nabla\cdot\boldsymbol{\sigma} + \rho\boldsymbol{g} \\ \Rightarrow \int_\Omega \phi\rho\boldsymbol{a}\,\mathrm{d}\Omega &= \int_\Omega \phi\nabla\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega + \int_\Omega \phi\rho\boldsymbol{g}\,\mathrm{d}\Omega. \end{aligned} \tag{2.12}$$

Let Φ be the space of sufficiently smooth functions, which are zero on the boundary parts where essential boundary conditions are imposed in the original equation. Then both the weak form and the original equation will have the same solution [10]. In case that ϕ meets these conditions, then Equation 2.12 can be further evaluated. First consider the stress term on the right hand side. Apply integration by parts and then the Gauss integration theorem. Finally split the boundary integral into its Dirichlet and its Neumann part:

$$\begin{aligned} \text{(Integration by parts)} \quad \int_\Omega \phi\nabla\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega &= \int_\Omega \nabla\cdot(\phi\boldsymbol{\sigma})\,\mathrm{d}\Omega && - \int_\Omega \nabla\phi\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega \\ \text{(Gauss theorem)} \quad &= \int_{\partial\Omega} \phi\boldsymbol{n}\cdot\boldsymbol{\sigma}\,\mathrm{d}\Gamma && - \int_\Omega \nabla\phi\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega \\ \text{(Split boundary)} \quad &= \int_{\partial\Omega_u} \overset{0}{\phi}\,\boldsymbol{n}\cdot\boldsymbol{\sigma}\,\mathrm{d}\Gamma + \int_{\partial\Omega_\tau} \phi\boldsymbol{n}\cdot\boldsymbol{\sigma}\,\mathrm{d}\Gamma && - \int_\Omega \nabla\phi\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega \\ \Rightarrow \int_\Omega \phi\nabla\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega &= \int_{\partial\Omega_\tau} \phi\boldsymbol{n}\cdot\boldsymbol{\sigma}\,\mathrm{d}\Gamma && - \int_\Omega \nabla\phi\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega. \end{aligned} \tag{2.13}$$

When splitting the boundary integral into its part over the Dirichlet and over the Neumann boundary parts, note that the test function is zero on the Dirichlet part. Therefore, this part can be crossed out and Equation 2.13 remains. Plug this back into the weak form in Equation 2.12 and this yields the final form of the weak equation. The full problem formulation then becomes

Find $\boldsymbol{a} \in \mathscr{A}$ such that

$$\int_\Omega \phi\rho\boldsymbol{a}\,\mathrm{d}\Omega = \int_{\partial\Omega_\tau} \phi\boldsymbol{n}\cdot\boldsymbol{\sigma}\,\mathrm{d}\Gamma - \int_\Omega \nabla\phi\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega + \int_\Omega \phi\rho\boldsymbol{g}\,\mathrm{d}\Omega \tag{2.14}$$

for all $\phi \in \Phi$.

The space $\mathscr{A}$ represents the space of sufficiently smooth functions for the acceleration that respect the essential boundary conditions. This space will be specified for MPM later. In MPM, Equation 2.14 will be used to solve the acceleration $\boldsymbol{a}$ at each time step.

2.2. CONCEPT OF MPM: PARTICLE IN GRID METHOD

The material point method is a finite element method to solve the equations of motion for a deforming continuum. MPM can be considered as a particle in grid method: a stationary background grid is combined with moving material points to discretise the continuum. The equations of motion, as previously described, are discretised over time and space and then solved. In order to keep track of the material data, MPM uses material points, which are also referred to as *particles*. These store all the properties of the material such as stress, velocity and density. These particles can move freely through the domain and always contain the same material. Therefore, the mass of each material point does not change, but its position, volume and other properties can. By doing so, it is easy to track the history of the material, which is necessary to solve the equations of motion.

The material points contain the information about the continuum, but in order to update these properties, a stationary background grid is used. The properties at the material points are projected onto the background grid, where the equations of motion are solved using a finite element approach. The finite element method uses the weak form of the momentum equation (Equation 2.14) to calculate the acceleration on the grid level. Then, the acceleration is projected back to the particles, where the velocities, positions and other properties of the material points are updated. Figure 2.3 illustrates the concept of the particle in grid method of MPM. Because the equations of motion use a Lagrangian frame of reference, no non-linear convective terms appear in the equations of motion, which would have appeared when the equations without the use of material points. Combining material points with a stationary background grid gives the advantage that the non-linear convective terms are avoided without the risk of grid distortion.

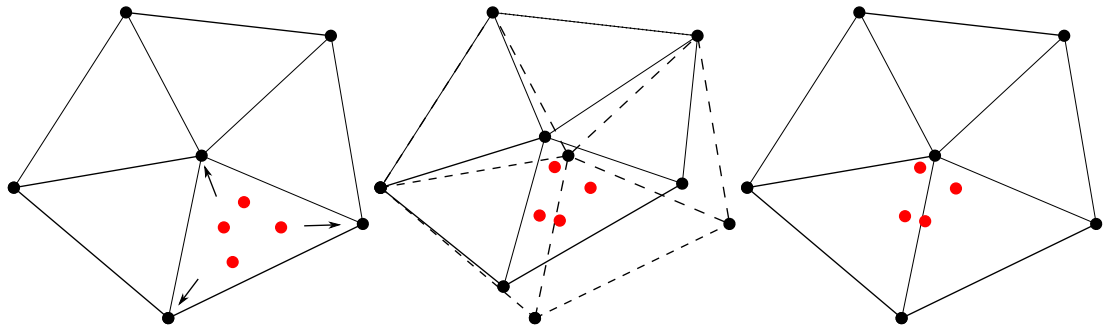


Figure 2.3: A part of a triangular MPM grid with particles inside. The concept of MPM is shown. (1): The particles properties are projected onto the grid. (2): The equations of motion are solved on the grid level. (3): The particle properties are updated and the grid is reset.

2.3. SPACE DISCRETISATION

In order to solve the momentum equation, it must be discretised on the background grid. The grids considered in this thesis are triangular. Triangular grids have several advantages over quadrilateral tensor grids. Triangular grids can be constructed by a Delaunay triangulation from an arbitrary set of points, whereas tensor grids are typically mapped from a unit square to the geometry. The tensor product grids have the advantage that the basis functions over the grid can be tensor products of one-dimensional basis functions, and are therefore easy to construct and evaluate. However, the mapping can give difficulties when trying to approximate complex geometries. Furthermore, it may be quite difficult to locally refine the grid. These problems do not appear when using a triangular grid. It is very easy to approximate an arbitrary geometry with triangles, also the local refinement process can simply be done by adding additional vertices locally and running the Delaunay triangulation process again. All together, a triangular grid is preferred for deformation problems with arbitrary geometries and thus this thesis studies MPM specifically with a triangular background grid.

Over the triangular background grid, a set of n_{bf} basis functions that span Φ is defined. For each of the basis functions ϕ_i, the weak form for the momentum conservation equation of Equation 2.14 will be solved, which is repeated here component-wise,

$$\int_\Omega \phi \rho a_k \, \mathrm{d}\Omega = \int_{\partial\Omega_\tau} \phi \sigma_{mk} n_m \cdot \mathrm{d}\Gamma - \int_\Omega \frac{\partial \phi}{\partial x_m} \sigma_{mk} \, \mathrm{d}\Omega + \int_\Omega \phi \rho g_k \, \mathrm{d}\Omega. \tag{2.15}$$

Note the Einstein summation over repeated index m in the above notation. To solve the equation, the acceler-

ation for each direction x_k is approximated by a linear combination of basis functions in Φ,

$$a_k(\boldsymbol{x}) \approx a_{k,h}(\boldsymbol{x}) = \sum_{j=1}^{n_{bf}} \hat{a}_{k,j}\phi_j(\boldsymbol{x}), \tag{2.16}$$

in which $a_{k,h}$ denotes the approximation of the acceleration in the x_k direction and $\hat{\boldsymbol{a}}_{k,j}$ the coefficients of its corresponding basis function. For each direction, the acceleration has its own coefficients.

Substituting the approximation $a_{k,h}$ into the weak form of the momentum equation results in the following equation for each of the basis functions ϕ_i,

$$\int_\Omega \phi_i \rho \sum_{j=1}^{n_{bf}} \hat{a}_{k,j}\phi_j \,\mathrm{d}\Omega = \int_{\partial\Omega_\tau} \phi_i \sigma_{mk} n_m \,\mathrm{d}\Gamma - \int_\Omega \frac{\partial \phi_i}{\partial x_m}\sigma_{mk}\,\mathrm{d}\Omega + \int_\Omega \phi_i \rho g_k \,\mathrm{d}\Omega, \quad \text{for } i = 1\dots n_{bf}.$$

By exchanging summation and integration, this can be rewritten to

$$\sum_{j=1}^{n_{bf}} \left(\int_\Omega \phi_i \rho \phi_j \,\mathrm{d}\Omega\right) \hat{a}_{k,j} = \int_{\partial\Omega_\tau} \phi_i \sigma_{mk} n_m \,\mathrm{d}\Gamma - \int_\Omega \frac{\partial \phi_i}{\partial x_m}\sigma_{mk}\,\mathrm{d}\Omega + \int_\Omega \phi_i \rho g_k \,\mathrm{d}\Omega, \quad \text{for } i = 1\dots n_{bf}. \tag{2.17}$$

Now assume that an acceleration function in the x_k-direction $a_{k,h}$ with coefficients $\hat{a}_{k,j}$ is found, such that Equations 2.17 hold for each of the basis functions ϕ_i. Then this acceleration function will also solve the equation when ϕ is any linear combination of the basis functions. This is true because the equation is linear in ϕ_i and therefore, $a_{k,h}$ will be a solution for any arbitrary function $\phi \in \Phi$.

Next, note that Equations 2.17 are a linear system of equations, and can be rewritten in the form

$$\boldsymbol{M}\hat{\boldsymbol{a}}_{\boldsymbol{k}} = \boldsymbol{F}_k^{trac} + \boldsymbol{F}_k^{int} + \boldsymbol{F}_k^{grav}. \tag{2.18}$$

In this equation, $\boldsymbol{M}$ denotes the mass matrix, and $\boldsymbol{F}_k^{trac}$, $\boldsymbol{F}_k^{int}$ and $\boldsymbol{F_k}^{grav}$ respectively denote the traction force, the internal force and the gravitational force in the x_k direction. The mass matrix is defined as a n by n matrix, with n the number of basis functions. Its entries are defined as

$$\boldsymbol{M}_{ij} = \int_\Omega \phi_i \rho \phi_j \,\mathrm{d}\Omega. \tag{2.19}$$

The entries i of the force vector are defined as

$$F_{k,i}^{trac} = \int_{\partial\Omega_\tau} \phi_i \sigma_{mk} n_m \,\mathrm{d}\Gamma, \tag{2.20}$$

$$F_{k,i}^{int} = \int_\Omega \frac{\partial \phi_i}{\partial x_m}\sigma_{mk}\,\mathrm{d}\Omega, \tag{2.21}$$

$$F_{k,i}^{grav} = \int_\Omega \phi_i \rho g_k \,\mathrm{d}\Omega. \tag{2.22}$$

For evaluating these integrals, the values of σ_{ml} and ρ are only known in the material points. Therefore, the material points uses the integration points for quadrature integration, and thus the information of the particles is mapped to the grid. The volumes V_p of the material points are used as integration weights, as this is the part of the domain occupied by the particle. The quadrature integration rule then becomes

$$\int_\Omega f(\boldsymbol{x})\,\mathrm{d}\Omega \approx \sum_{p=1}^{n_p} V_p f(\boldsymbol{x}_p), \tag{2.23}$$

where f is an arbitrary function and n_p the number of particles. When using this quadrature rule, the mass matrix and forces in Equation 2.18 are evaluated as

$$\boldsymbol{M}_{ij} = \sum_{p=1}^{n_p} V_p \phi_i(\boldsymbol{x}_p)\rho_p\phi_j(\boldsymbol{x}_p) = \sum_{p=1}^{n_p} m_p \phi_i(\boldsymbol{x}_p)\phi_j(\boldsymbol{x}_p), \tag{2.24}$$

$$F_{k,i}^{int} = \sum_{p=1}^{n_p} V_p \frac{\partial \phi_i}{\partial x_m}(\boldsymbol{x}_p)\,\sigma_{mk,p}, \tag{2.25}$$

$$F_{k,i}^{grav} = \sum_{p=1}^{n_p} V_p \phi_i(\boldsymbol{x}_p)\rho_p g_k \qquad = \sum_{p=1}^{n_p} m_p \phi_i(\boldsymbol{x}_p) g_k. \tag{2.26}$$

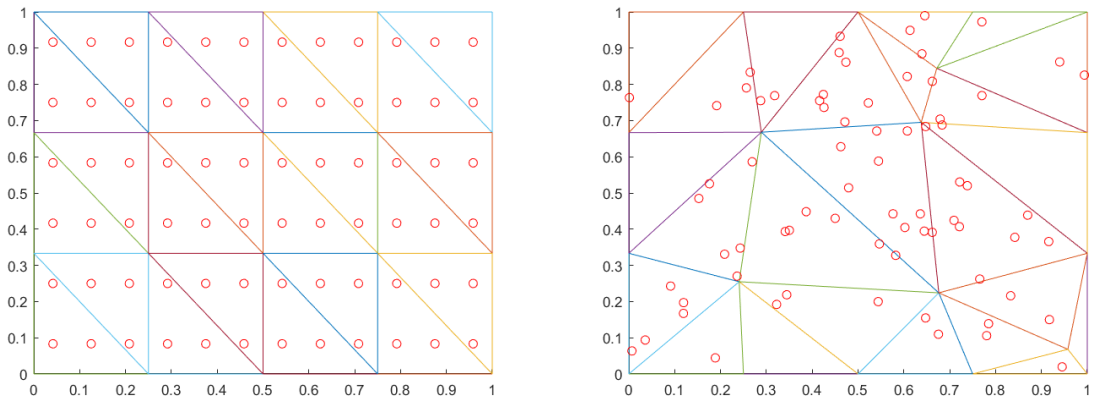
Here ρ_p, V_p and m_p respectively denote the density, volume and mass in each material point.

This is the spatial discretisation that will be used to solve the momentum equation in each time step. In the next section the time discretisation and the time stepping procedure will be explained in detail.

2.4. SOLUTION PROCEDURE

In order to solve the equations of motion of a continuum over time, a time stepping procedure is required. The time stepping procedure presented in this section was recovered from [7], and is used by Deltares [1] as well. The basic principle of the method is to discretise the time and update all particle properties in each time step using the momentum conservation equation.

For the time integration, first the material should be initialised on the domain of interest Ω^0. The number of material points n_p must be chosen and the grid should be defined. To get a more reliable quadrature integration more particles can be added per element. To reduce the global quadrature integration error as much as possible for a fixed number of particles, we suggest distributing the particles evenly over all elements, such that each element starts with the same number of material points. Depending on the situation, different initial particle distribution may prove to be better, however. For now, assume that the triangular grid is defined and the material points are initially evenly distributed over the triangles. At time $t = 0$, each material point is assigned its initial position $\boldsymbol{x}_p^0$, its displacement from its stationary position $\boldsymbol{u}_p^0$, velocity $\boldsymbol{v}_p^0$, mass m_p^0, volume V_p^0, density ρ_p^0 and stress $\boldsymbol{\sigma}_p^0$, where p denotes the particle and the superscript 0 the time step. Figure 2.4 illustrates two possible initialisations of the particles in the grid, a regular one and an irregular one. MPM can work with either type, though in most cases, a structured grid is preferable.



(a) Regular

(b) Irregular

Figure 2.4: A possible discretisation of a unity square domain. The domain is divided in triangular elements, with particles inside. On the left a fully regular grid and particle distribution are shown. Both the grid and particles can also be generated in a more arbitrary way, as shown on the right. Both examples have the same number of triangles and particles.

Having initialised the grid and the particles, the time stepping procedure can start, which passes the following phases during each time step.

1. Assemble the mass matrix and force vectors of Equation 2.18 on the background grid, using the particles as integration points as shown in Equations 2.24-2.26.
2. Solve the coefficient vector $\hat{\boldsymbol{a}}_k$ from the system of Equations 2.18. From the coefficient vector, the global acceleration field can be determined.
3. Using the acceleration over the grid, update the velocity and position in the particles. Then from the updated positions, determine the new strain and stress in the particles using the stress and strain relations shown in Section 2.1.

After phase 3, a new time step can be taken and the procedure starts at phase 1 again. In the next subsections, each of these three phases will be elaborately described.

2.4.1. ASSEMBLING MOMENTUM CONSERVATION EQUATION

In the first phase, Equation 2.18 needs to be assembled. The mass matrix and the force vectors are assembled using quadrature integration in the material points, as explained in Section 2.3. Only the assembly of the

traction force has not been explained yet. The traction force is determined by integrating over the boundary where a traction force is applied. This boundary is not prescribed and may move over time, so boundary particles are assigned at the start of the simulation and these are assumed to be boundary particles for the rest of the simulation. Each of the boundary particles will be assigned a part of the traction force corresponding to its location and its initial boundary surface. Then, each of those boundary particles has a traction force prescribed at each moment in time. The assumption made here is that the traction force originates from a solid object pressing on the surface, for example a slab of concrete on top, compressing the continuum, and that the resulting force vector on the material is fully known. The traction force in the x_k-direction in each of the boundary particles p is then prescribed as: $V_p \sigma_{mk,p} n_{k,p} = f^{trac}_{k,p}$. The traction force in the remaining particles is equal to 0. The traction force vector $\boldsymbol{F}^{trac}$ can then be assembled in a similar fashion as the other force vectors. The full assembly at time t for the x_k direction then becomes

$$\boldsymbol{M}^t_{ij} = \sum_{p=1}^{n_p} m_p \phi_i(\boldsymbol{x}_p)\phi_j(\boldsymbol{x}_p), \tag{2.27}$$

$$\boldsymbol{F}^{trac,t}_{k,i} = \sum_{p=1}^{n_p} \phi_i(\boldsymbol{x}_p)\, f^{trac}_{k,p}, \tag{2.28}$$

$$\boldsymbol{F}^{int,t}_{k,i} = \sum_{p=1}^{n_p} V_p \frac{\partial \phi_i}{\partial x_m}(\boldsymbol{x}_p)\, \sigma_{mk,p}, \tag{2.29}$$

$$\boldsymbol{F}^{grav,t}_{k,i} = \sum_{p=1}^{n_p} m_p \phi_i(\boldsymbol{x}_p) g_k. \tag{2.30}$$

2.4.2. Solving the momentum equation

After assembling the momentum equation on the grid level, the system of equations should be solved for the acceleration in each direction,

$$\hat{\boldsymbol{a}}^t_k = \left(\boldsymbol{M}^t\right)^{-1} \boldsymbol{F}^t_k, \tag{2.31}$$

in which $\left(\boldsymbol{M}^t\right)^{-1}$ denotes the inverse of the mass matrix at time step t and $\boldsymbol{F}^t_k$ denotes the force vector in x_k-direction at time step t,

$$\boldsymbol{F}^t_k = \boldsymbol{F}^{trac,t}_k - \boldsymbol{F}^{int,t}_k + \boldsymbol{F}^{grav,t}_k. \tag{2.32}$$

It is computationally very expensive to calculate an inverse, and most choices of basis functions yield a sparse mass matrix $\boldsymbol{M}$, so a numerical technique can be used to solve this system of equations instead. For example, Richardson iteration can be used to solve the system. It is also common practice in MPM to solve this equation by lumping the mass matrix, see Appendix A.2 for more details on lumping in MPM. However, in this thesis, we will solve the consistent system of equations unless stated otherwise.

After solving the system of equations and determining the coefficients $\hat{\boldsymbol{a}}^t_k$, recall that the full acceleration field can be reconstructed from the basis functions as

$$a^t_{k,h}(\boldsymbol{x}) = \sum_{i=1}^{n_{bf}} \hat{a}^t_{k,i} \phi_i(\boldsymbol{x}), \tag{2.33}$$

with n_{bf} the number of basis functions ϕ_i. This is the solution for the acceleration field in any direction over the full domain. Using this field, first the particle velocities can be updated, then the particle positions, and finally the other properties as well.

2.4.3. Updating particle properties

Various different methods within MPM exist for updating the particle properties. The method followed in this thesis was proposed by Sulsky et al. [7], and will be referred to as the *modified update algorithm* in this thesis. Instead of immediately updating the both the particle velocity and particle position from the acceleration field, the modified algorithm first only updates the velocity in the particles, and then reconstructs the total velocity field over the grid. After the projection of the velocity on the grid, the displacement in each particle is updated by using this velocity field. Afterwards, the strain and stress are determined locally in the particles again.

Given the acceleration coefficient vector $\hat{\boldsymbol{a}}$, the algorithm first updates the velocity in the particles. This is done by first calculating the acceleration in each particle point using Equation 2.33. Then the velocity is updated by adding the acceleration times the time step size to the old velocity:

$$v_{k,p}^{t+\Delta t} = v_{k,p}^{t} + \Delta t \sum_{i=1}^{n_{bf}} \hat{a}_{k,i}^{t} \phi_i(\boldsymbol{x_p}). \tag{2.34}$$

After determining the new velocity in each of the particles, the modified update algorithm projects the velocity from the particles on to the grid. This is done by using a density weighted L_2-projection onto the space Φ spanned by the basis functions ϕ_i. For this, the projected velocity field in the x_k direction $v_{k,h}^{t}$ is assumed to be a linear combination of the basis functions ϕ_i,

$$v_{k,h}^{t+\Delta t}(\boldsymbol{x}) = \sum_{j=1}^{n_{bf}} \hat{v}_{k,j}^{t+\Delta t} \phi_j(\boldsymbol{x}), \tag{2.35}$$

in which $\hat{v}_{k,j}^{t+\Delta t}$ denotes the velocity coefficients in the x_k direction at time $t+\Delta t$. For the density weighted L_2-projection on Φ the following system of equation must hold,

$$\int_{\Omega^t} \rho(\boldsymbol{x})\, v_{k,h}^{t+\Delta t}(\boldsymbol{x}) \phi_i \,\mathrm{d}\Omega = \int_{\Omega^t} \rho(\boldsymbol{x})\, v_{k}^{t+\Delta t}(\boldsymbol{x}) \phi_i \,\mathrm{d}\Omega, \quad \text{for } i = 1 \dots n_{bf}. \tag{2.36}$$

Note the difference between the projected velocity $v_{k,h}$ on the left versus the velocity field v_k on the right. The velocity field v_k is only known in the particles, and therefore, the integral on the right hand side will again be approximated by using a quadrature rule. The velocity projection on the left hand side can be substituted with Equation 2.35. The L_2-projection is expanded next,

$$\begin{aligned}
&\int_{\Omega^t} \rho(\boldsymbol{x})\, v_{k,h}^{t+\Delta t}(\boldsymbol{x}) \phi_i \,\mathrm{d}\Omega = \int_{\Omega^t} \rho(\boldsymbol{x})\, v_{k}^{t+\Delta t}(\boldsymbol{x}) \phi_i \,\mathrm{d}\Omega, \\
\Rightarrow &\int_{\Omega^t} \rho(\boldsymbol{x}) \left(\sum_{j=1}^{n_{bf}} \hat{v}_{k,j}^{t+\Delta t} \phi_j(\boldsymbol{x}) \right) \phi_i(\boldsymbol{x}) \,\mathrm{d}\Omega = \sum_{p=1}^{n_p} V_p \rho_p\, v_{k,p}^{t+\Delta t} \phi_i(\boldsymbol{x}_p), \\
\Rightarrow &\sum_{j=1}^{n_{bf}} \left(\int_{\Omega^t} \phi_i(\boldsymbol{x}) \rho(\boldsymbol{x}) \phi_j(\boldsymbol{x}) \,\mathrm{d}\Omega \right) \hat{v}_{k,j}^{t+\Delta t} = \sum_{p=1}^{n_p} m_p\, v_{k,p}^{t+\Delta t} \phi_i(\boldsymbol{x}_p)
\end{aligned}$$

The last equation must hold for each of the basis functions ϕ_i, $i = 1 \dots n_{bf}$. This results in a system of equations for the coefficient vector $\hat{\boldsymbol{v}}_k^{t+\Delta t}$ in which the mass matrix $\boldsymbol{M}$ from Equation 2.27 is used again. The system can be written as

$$\boldsymbol{M} \hat{\boldsymbol{v}}_k^{t+\Delta t} = \boldsymbol{P}_k^{t+\Delta t}. \tag{2.37}$$

Here, $\boldsymbol{P}$ represents the right hand side of the equation, which is also the linear momentum of the system integrated over ϕ, hence the symbol $\boldsymbol{P}$,

$$P_{k,i}^{t+\Delta t} = \sum_{p=1}^{n_p} m_p\, v_{k,p}^{t+\Delta t} \phi_i(\boldsymbol{x}_p). \tag{2.38}$$

Equation 2.37 can be solved in the same manner as the acceleration coefficient vector was solved in Equation 2.18. Again, it is possible to apply lumping, but in this thesis this will not be done unless stated explicitly.

Having determined the projection of the velocity field on the grid level, first displacement increment Δu_k in the x_k-direction of the particles is calculated

$$\Delta u_{k,p}^{t+\Delta t} = \Delta t \sum_{i=1}^{n_{bf}} v_{k,i}^{t+\Delta t} \phi_i(\boldsymbol{x}_p). \tag{2.39}$$

We do not update the particle position until the all other properties are updated though, and for determining the value of $\phi_i(\boldsymbol{x}_p)$ in the next equations, the old particle positions are still used.

Next, the strain is updated using the incremental particle displacement using Equation 2.8; note that i and j temporarily represent the x_i- and x_j-direction,

$$\begin{aligned}
\frac{\partial \boldsymbol{\epsilon}_{ij,p}^{t+\Delta t}}{\partial t} &= \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \\
\Rightarrow \Delta\boldsymbol{\epsilon}_{ij,p}^{t+\Delta t} &= \Delta t \frac{1}{2}\left(\frac{\partial\left(\sum_{l=1}^{n_{bf}} \phi_l \hat{v}_{i,l}^{t+\Delta t}\right)}{\partial x_j} + \frac{\partial\left(\sum_{l=1}^{n_{bf}} \phi_l \hat{v}_{j,l}^{t+\Delta t}\right)}{\partial x_i}\right) \\
\Rightarrow \Delta\boldsymbol{\epsilon}_{ij,p}^{t+\Delta t} &= \Delta t \frac{1}{2}\sum_{l=1}^{n_{bf}}\left(\hat{v}_{i,l}^{t+\Delta t}\frac{\partial \phi_l}{\partial x_j}(\boldsymbol{x}_p) + \hat{v}_{j,l}^{t+\Delta t}\frac{\partial \phi_l}{\partial x_i}(\boldsymbol{x}_p)\right)
\end{aligned} \tag{2.40}$$

Next, the strain increment in Equation 2.40 will be used to update the stress. The constitutive relation from Equation 2.4 is discretised for this, note the summation of repeated indices k and l,

$$\begin{aligned}
\frac{\partial \sigma_{ij}}{\partial t} &= D_{ijkl}\frac{\partial \epsilon_{kl}}{\partial t} - \frac{\partial \epsilon_{kk}}{\partial t}\sigma_{ij} + \omega_{ik}\sigma_{kj} - \sigma_{ik}\omega_{kj}, \\
\Rightarrow \Delta\sigma_{ij,p}^{t+\Delta t} &= D_{ijkl}\Delta\epsilon_{kl,p}^{t+\Delta t} - \Delta\epsilon_{kk,p}^{t+\Delta t}\sigma_{ij,p}^{t} + \Delta\omega_{ik,p}^{t+\Delta t}\sigma_{kj,p}^{t} - \sigma_{ik,p}^{t}\Delta\omega_{kj,p}^{t+\Delta t},
\end{aligned} \tag{2.41}$$

in which $\omega_{ij,p}^{t}$ is the discretised spin tensor in $\boldsymbol{x}_p$,

$$\begin{aligned}
\omega_{ij} &= \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i}\right), \\
\Rightarrow \Delta\omega_{ij,p}^{t+\Delta t} &= \frac{1}{2}\Delta t\left(\frac{\partial\left(\sum_{l=1}^{n_{bf}} \hat{v}_{i,l}^{t+\Delta t}\phi_l(\boldsymbol{x}_p)\right)}{\partial x_j} - \frac{\partial\left(\sum_{l=1}^{n_{bf}} \hat{v}_{j,l}^{t+\Delta t}\phi_l(\boldsymbol{x}_p)\right)}{\partial x_i}\right), \\
\Rightarrow \Delta\omega_{ij,p}^{t+\Delta t} &= \frac{1}{2}\Delta t\sum_{l=1}^{n_{bf}}\left(\hat{v}_{i,l}^{t+\Delta t}\frac{\partial \phi_l}{\partial x_j}(\boldsymbol{x}_p) - \hat{v}_{j,l}^{t+\Delta t}\frac{\partial \phi_l}{\partial x_i}(\boldsymbol{x}_p)\right).
\end{aligned} \tag{2.42}$$

Note that the basis functions are fully known, and therefore, the derivatives are also known. Therefore, the full right hand sides in Equations 2.40 and 2.42 are known. After determining the strain increments and the spin tensor, the stress increment can be calculated using Equation 2.41. Finally the total stress is updated as

$$\sigma_{ij,p}^{t+\Delta t} = \sigma_{ij,p}^{t} + \Delta\sigma_{ij,p}^{t+\Delta t}. \tag{2.43}$$

Next, the volume and density of each particle must also be updated. This happens based on the normal strain increment, as this determines the stretching or compressing of the volume element. The factor by which a volume shrinks or expands would then be $\Delta\epsilon_{vol} = (\Delta\epsilon_{11} + \Delta\epsilon_{22}) = \Delta\epsilon_{kk}$. The full volume update then becomes

$$V_p^{t+\Delta t} = \left(1 + \Delta\epsilon_{vol,p}^{t+\Delta t}\right)V_p^{t+\Delta t} = \left(1 + \Delta\epsilon_{kk,p}^{t+\Delta t}\right)V_p^{t}. \tag{2.44}$$

The density update then follows directly from the volume update and the fact that the mass of each particle is constant through time,

$$\begin{aligned}
V_p^{t+\Delta t}\rho_p^{t+\Delta t} &= m_p = V_p\rho_p^{t}, \\
\Rightarrow \rho_p^{t+\Delta t} &= \frac{\rho_p^{t}}{\left(1 + \Delta\epsilon_{kk,p}^{t+\Delta t}\right)}.
\end{aligned} \tag{2.45}$$

Lastly, the total displacement and the particle positions are updated.

$$x_{k,p}^{t+\Delta t} = x_{k,p}^{t} + \Delta u_{k,p}^{t+\Delta t} \tag{2.46}$$

$$u_{k,p}^{t+\Delta t} = u_{k,p}^{t} + \Delta u_{k,p}^{t+\Delta t}. \tag{2.47}$$

As the particle position are updated last, all the function evaluations happened in the old particle positions.

This is the whole algorithm that happens each time step. Figure 2.5 shows an overview of all the steps that are taken during each time step.

1. Assemble the mass matrix $\boldsymbol{M}$ and the force vectors $\boldsymbol{F}^{trac}$, $\boldsymbol{F}^{int}$ and $\boldsymbol{F}^{grav}$ from the particle data, as shown in Equations 2.27-2.30.
2. Solve the acceleration coefficient vector $\hat{\boldsymbol{a}}_k$ from the system of equations $\boldsymbol{M}\hat{\boldsymbol{a}}_k = \boldsymbol{F}_k$ (Equation 2.18) for each direction x_k. This represents the acceleration field on the grid level.
3. Evaluate the acceleration field in the particles and update the particle velocity, as shown in Equation 2.34.
4. Reconstruct the new velocity field by projecting the particle data onto the grid. This is done using an L_2-projection onto the basis functions. Assemble Equation 2.35 and solve for the coefficient vectors of the velocity on the grid level.
5. Using the velocity field on the grid level, find the displacement increment at each of the particles using Equation 2.39.
6. Find the strain increment at each of the particles using the displacement increment at the particles as shown in Equation 2.42.
7. From the strain increment, the stress, volume and density can be updated using Equations 2.43, 2.44 and 2.45 respectively.
8. Finally, the total displacement and the positions of each of the particles are updated, as in Equations 2.46 and 2.47

Figure 2.5: The time stepping algorithm.

2.5. BOUNDARY CONDITIONS

In this section, the implementation of the boundary conditions is discussed. In Section 2.1.3, it was stated that the boundary was divided into a part with prescribed displacement and a part with prescribed traction. First the boundary with the prescribed traction is considered.

The boundary part $\partial\Omega_\tau$ on which the traction is described, is represented by a set of boundary particles. Each boundary particle represents a part of the boundary, and is assigned a traction force in correspondence with the boundary part it represents. The particle is considered a boundary particle throughout the simulation. More advanced techniques can also be used to represent the boundary traction force, but for the benchmarks considered in this thesis, this boundary condition is always homogeneous, so this is not necessary.

The boundary conditions on the displacement boundary $\partial\Omega_u$ are implemented by imposing that the acceleration and velocity field at this boundary should be in accordance with the boundary condition. The particles at the boundary do not have to exactly follow this condition, as they represent a piece of mass, of which the centers lie just next to the physical boundary. The boundary conditions are imposed on the reconstructed velocity and acceleration fields by setting the degrees of freedom at the boundary to the corresponding value instead. This is done by changing a number of equations in the system of Equations 2.18. Consider an example in which piece-wise linear basis functions are used. A basis function ϕ_i is 1 at the boundary, and on the boundary, this basis function is the only one that contributes to the value of the reconstructed velocity on that location. In case the boundary condition imposes that at this point, the x-velocity should be v_x, then the coefficient $\hat{v}_i^x$ of this basis function should be immediately set to the value of v_x. This can be done by replacing the original equation for ϕ_i in Equations 2.18 by an equation setting its value to v_x. In this thesis however, the Dirichlet boundaries considered will be homogeneous. Therefore, the equations for these degrees of freedom at the boundary are replaced with equations setting the corresponding coefficients to zero for both the acceleration and velocity field.

2.6. Critical time step

The time integration method used in the solution procedure is called the semi-implicit Euler-Cromer scheme [11]. First, the acceleration is determined at time t, and this acceleration is used to explicitly update the velocity at the next time step $t+\Delta t$. Then, this new velocity is used to update the position of each particle. Because the particle position is updated with the velocity at time $t+\Delta t$, this update part is implicit.

If the time steps are taken too large for this method, the solution of the simulation becomes unstable and diverges from the real solution. The critical time step at which this happens depends on the spatial discretisation and the material. In general, a bound for the critical time step using this scheme is given by the CFL condition (Courant, Friedrichs, Lewy) [1, 12]. The condition is given by

$$\Delta t \leq \frac{h}{\sqrt{E/\rho}}. \tag{2.48}$$

Here, h stands for the typical length of the smallest element used in the grid, and $\sqrt{E/\rho}$ is a measure for the wave speed in the material. The condition imposes that a wave in the continuum should not travel more than the typical length of an element during one time step. Throughout this thesis, the time step is always chosen sufficiently small such that this condition is respected. In general, a time step size in the range $\Delta t \approx 10^{-3} - 10^{-4}$ is used, whereas $\frac{h}{\sqrt{E/\rho}}$ is never larger than 10^{-3}, guaranteeing a stable time integration.

2.7. Grid Crossing Error

One of the larger issues of MPM is the appearance of grid crossing errors. These errors occur when a particle crosses over from one element to a neighbouring element and thus crosses the grid. If basis functions are used with a discontinuous derivative over element interfaces, the derivative of the basis function in the particle will make a jump when crossing this interface. The internal force in the MPM procedure is evaluated by integrating the velocity field multiplied with the gradient of the basis functions. The integral is then approximated by using a quadrature rule, summing over the values in the material points. Recall Equation 2.29,

$$F_{k,i}^{int,t} = \sum_{p=1}^{n_p} V_p \frac{\partial \phi_i}{\partial x_j}(\boldsymbol{x}_p)\sigma_{jk,p}.$$

Next consider a 1D example with a piece-wise linear basis function and particles as shown in Figure 2.6. There are 4 particles, two in the left element and two in the right. Next time step, the blue particle just left of x_2 crosses over to the right element. The internal force before and after the grid crossing are compared. From the illustration, it can be seen that $\frac{\partial \phi}{\partial x}$ is positive constant left of x_2 and negative constant right of x_2. Assume that $\frac{\partial \phi}{\partial x}$ is 1 on the left element and -1 over the right element. Also assume that stress is a constant scalar σ over the entire domain and that the volume of each particle is constant V. Then before the crossing, the total internal force is

$$F_{1,i}^{int,t} = \sum_{p=1}^{n_p} V_p \frac{\partial \phi_i}{\partial x}(\boldsymbol{x}_p)\sigma = 2V\sigma - 2V\sigma = 0,$$

whereas after the crossing, the internal force is

$$F_{1,i}^{int,t} = \sum_{p=1}^{n_p} V_p \frac{\partial \phi_i}{\partial x}(\boldsymbol{x}_p)\sigma = V\sigma - 3V\sigma = -2V\sigma.$$

A sudden jump in the internal force is the result of the discontinuity in the basis function.

This problem can be mitigated in several ways, for example by first reconstructing a global stress function and then using quadrature integration in fixed Gauss points. As the used Gauss points are fixed, grid crossing errors do not occur anymore. A more structural solution is to use basis function with C^1-continuity over element interfaces. This is the reason to investigate C^1-continuous basis functions in this thesis, such that grid crossing errors are will not occur. In the next chapter, a higher-order C^1-continuous basis is researched, which leads to Powell-Sabin spline basis functions.

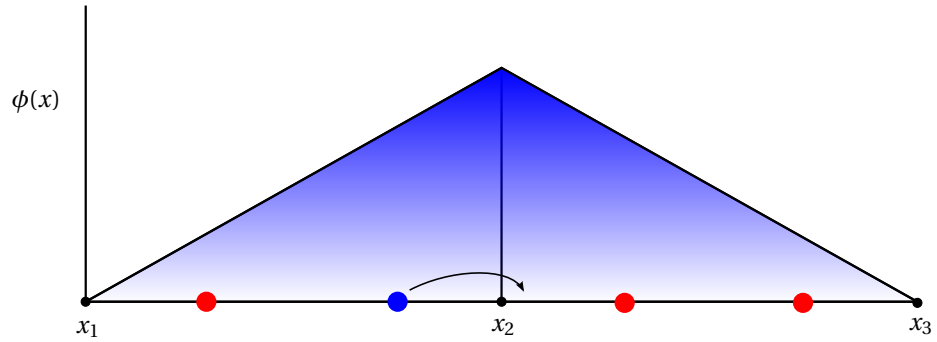


Figure 2.6: Illustration of grid crossing on a 1D grid, the green particle just left of x_2 moves to the position of the blue particle just right of x_2 during one time step. The red particles do not move.

3

POWELL-SABIN SPLINE BASIS FUNCTIONS

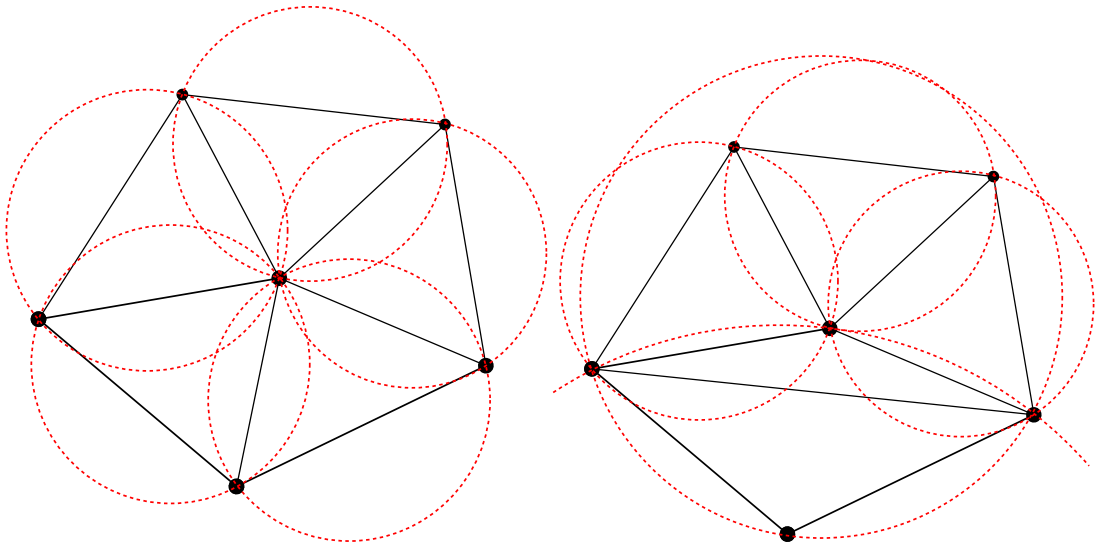
In the last chapter, the numerical algorithm has been described for solving a continuum deformation problem with MPM. In this chapter, will discuss a higher-order basis for implementation in MPM, which will finally lead us to adopt Powell-Sabin splines, which are non-negative, C^1-continuous splines on triangulations. This chapter structurally explain the choice for the Powell-Sabin basis functions, by first investigating the logical alternatives and marking their weaknesses. To do so, we first provide a short introduction into the construction of the grid and barycentric coordinates. Then the choice of basis function over the grid is considered. The basis function used in classical MPM are the piece-wise linear Lagrangian basis functions, which are discussed in Section 3.3, and we explain the shortcomings of higher-order Lagrangian basis functions. The goal of this thesis is to replace the Lagrangian basis by one that has continuous derivatives over the full domain and has the property of non-negativity. As a Lagrangian basis will not possess these properties, this leads us to investigating spline basis functions instead. First spline basis functions are introduced for 1D in Section 3.4, which will prove to have suitable properties. It is also shown that a tensor product of spline basis function is not possible for triangulations. Finally a 2D basis of Powell-Sabin (PS) spline functions for triangulation is considered in Section 3.5, which are also implemented in MPM in this thesis. This section is the most important part of this chapter, and we suggest skipping Sections 3.3 and 3.4 in case the reader is already familiar with Lagrange basis functions and splines and is mainly interested in the construction of the final Powell-Sabin spline basis functions. Finally, in Section 3.6, the higher-order spatial convergence for L_2-projections is proved for PS-splines. In Chapter 4, the resulting PS-spline MPM will be compared to classical Lagrange MPM.

3.1. TRIANGULAR GRID

In the version of MPM considered in this thesis, a triangular grid with material points inside is used to describe a continuum. The grid and the material points are initialised before the simulation. When the simulation starts, the material points move, but the grid is stationary. The grid must be made large enough such that the particles will never leave the grid, otherwise the method cannot be applied. It may very well be possible that the particles leave the element they started in and therefore may also leave the grid entirely. Therefore, the grid should be initialised large enough to include the largest possible deformation. This may lead to empty elements, i.e. triangles without any particles inside, but this is allowed in MPM. The grid must be large enough such that the particles never leave the grid, but preferably not any larger, as more elements imply more required memory and extra computation time.

First, we consider the construction of the grid. Triangular grids are easy to construct and refine. The triangular grid construction method we use is the Delaunay triangulation procedure [13]. For this procedure, first a set of vertices is defined. Then, these vertices are connected such that the domain is divided into triangles and the circumcircle of each triangle contains only its own vertices. A 2D example of a Delaunay triangulation is shown in Figure 3.1. The procedure avoids as much as possible sliver triangles, which are very long stretched triangles with small angles. Sliver triangles can cause instabilities and should therefore be avoided [14]. As sliver triangles are avoided as best as possible in the Delaunay triangulation procedure, this is is very suitable method for creating a grid. Furthermore, it is very easy to locally refine a grid with this procedure. If a certain part of the domain is of interest, then the grid can be refined there by adding more vertices at that part ini-

tially and the Delaunay triangulation procedure produces smaller triangles at that area. Finally, this procedure works for any set of initial vertices, and can therefore easily generate unstructured grids and approximate arbitrary shapes.



(a) A Delaunay triangulation (b) Not a Delaunay triangulation

Figure 3.1: The condition for a Delaunay triangulation of a set of vertices. The circumcircle of every triangle must contain no other vertices than its own. On the left, the Delaunay triangulation for a set of points is shown. On the right, a possible other triangulation is shown, which is not Delaunay, as the circumcircles of the lower two triangles contain other points.

When initialising the grid, the particles are also initialised in MPM. After defining the triangulation, it is advised to initialise each triangle that lies within the initial domain with a comparable number of particles. This is because the error in the quadrature rule for the mass matrix and force vectors in Equations 2.27-2.30 depend on the number of integration points. When dividing the particles evenly over the elements, the total integration error is mitigated well.

Once the particles have been distributed over the domain, each should be assigned an initial volume. In this thesis, the volume of each material point is determined from a Voronoi diagram. This diagram divides the domain into volumes around the particles, such that every position in volume V_p is closer to particle p than to any other particle. An example of the division of a 2D rectangular domain using a Voronoi diagram is shown in Figure 3.2.

3.2. Barycentric coordinates

Before defining the basis functions over the triangulation, we require a barycentric coordinate system. Barycentric coordinates are commonly used when describing functions over triangles. The barycentric coordinates of a point in a triangle describe the position of the point with respect to the triangle. The barycentric coordinates are defined as follows: given a triangle in 2D with vertices (x_1, y_1), (x_2, y_2) and (x_3, y_3) in Cartesian coordinates, the Cartesian coordinates of a point (x, y) can be converted to barycentric coordinates with respect to this triangle by solving

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{3.1}$$

Here, $(\zeta_1, \zeta_2, \zeta_3)$ are the barycentric coordinates of the point with respect to the triangle. Note that there are three coordinates, whereas the physical space is only $\mathbb{R}^2$. This is because the sum of the barycentric coordinates is defined to be $\zeta_1 + \zeta_2 + \zeta_3 = 1$. This is also visible in the lower row of Equation 3.1. The barycentric coordinates of a point denote its location as a linear combination of the three vertices. Therefore, the barycentric coordinates $(1, 0, 0)$ denote the location of vertex 1, $(1/2, 1/2, 0)$ is the point in the middle between vertex 1 and 2, and $(1/3, 1/3, 1/3)$ is the center of mass of the triangle. Figure 3.3 shows an example of a triangle and

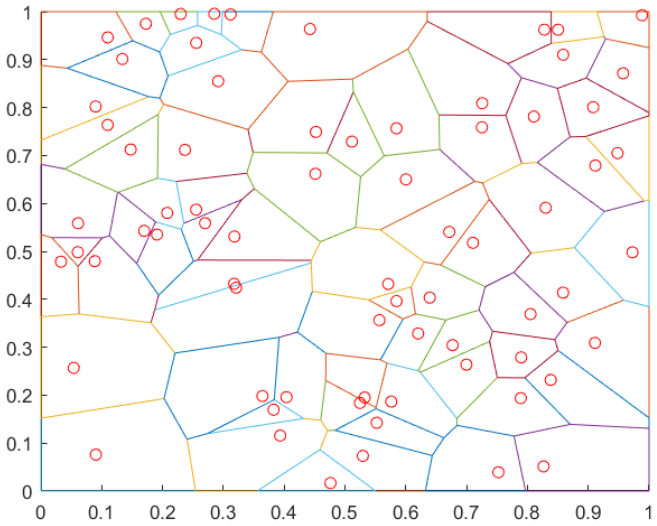


Figure 3.2: A Voronoi diagram of randomly distributed particles, marked as circles. Not to be confused with the background grid, which has nothing to do with it. The Voronoi diagram is used to initialise the volume of each particle.

its barycentric coordinates. In the next sections, the barycentric coordinates will be used to define the basis functions over triangulations.

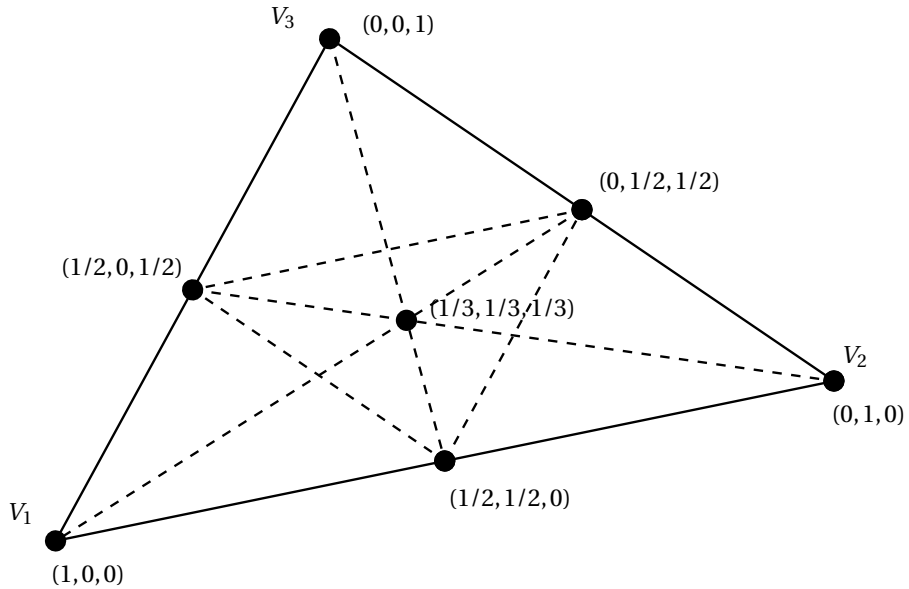


Figure 3.3: A number of barycentric coordinates of an arbitrary triangle with respect to its vertices V_1, V_2, V_3.

3.3. LAGRANGIAN BASIS FUNCTIONS AND THEIR SHORTCOMINGS

In this section, the Lagrangian basis functions over a triangulation are discussed. The basis functions will be expressed in barycentric coordinates. First however, the one-dimensional Lagrangian basis functions are considered. For the one-dimensional basis functions, the grid is defined by a set of nodes on a line. The elements in this grid are the spaces between two neighbouring nodes. Figure 3.4 illustrates a possible division of a grid into nodes and elements. The 1D Lagrangian basis functions are defined locally over each element. Basis function i is associated with node i, and takes the value 1 at x_i, and 0 at every other node. This is the

interpolatory property of the Lagrangian basis functions,

$$\phi_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}, \tag{3.2}$$

In this equation, δ_{ij} is the Kronecker delta function, x_j the location of node j and ϕ_i is the i^{th} basis function. Furthermore, the Lagrangian basis functions also posses the property of locality. Each basis function is only nonzero on at most two elements. On all the other elements, the function is equal to zero. Finally, the functions are piece-wise polynomials. Over each element, the basis functions are smooth polynomials, and over function interfaces, the piece-wise polynomials are continuously connected. The derivatives however, are not.

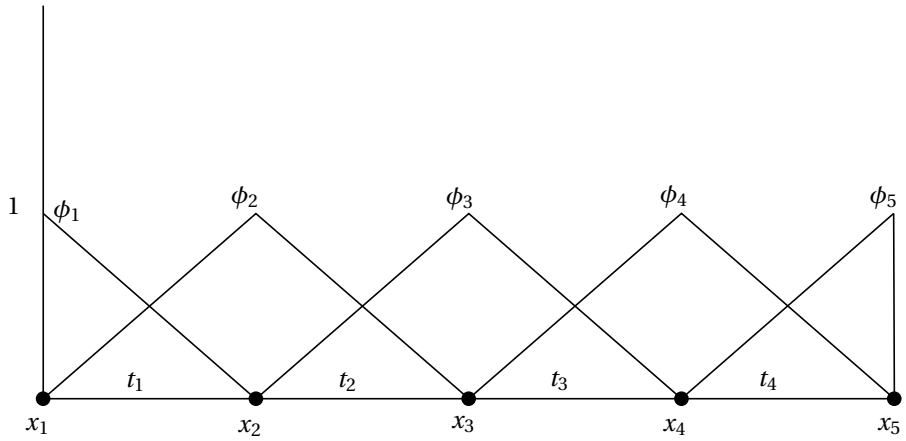


Figure 3.4: A 1D equidistant grid with 5 vertices V and 4 elements t. The piece-wise linear Lagrangian basis functions are shown over this grid.

First consider the piece-wise linear Lagrangian basis functions. Over the element left and right of x_i, the basis function is piece-wise linear, and over all the other elements the basis function is 0. ϕ_i is defined as

$$\phi_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & \text{if } x_{i-1} \leq x < x_i \\ \frac{x-x_{i+1}}{x_i-x_{i+1}} & \text{if } x_i \leq x \leq x_{i+1} \\ 0 & \text{else.} \end{cases} \tag{3.3}$$

These basis functions are illustrated in Figure 3.4.

Next, consider the piece-wise quadratic Lagrangian basis function. For this choise of basis functions, the grid has to be refined. In the middle of each element, an additional node is added, as shown in Figure 3.5. The elements are the same as in the unrefined grid; however, each element has two points at its borders and an additional one in the middle. Consider element i, over this element, three basis functions are non-zero, these are given by

$$\phi_i = \frac{(x - x_{i+1/2})(x - x_{i+1})}{(x_i - x_{i+1/2})(x_i - x_{i+1})}, \tag{3.4}$$

$$\phi_{i+1/2} = \frac{(x - x_i)(x - x_{i+1})}{(x_{i+1/2} - x_i)(x_{i+1/2} - x_{i+1})}, \tag{3.5}$$

$$\phi_{i+1} = \frac{(x - x_i)(x - x_{i+1/2})}{(x_{i+1} - x_i)(x_{i+1} - x_{i+1/2})}. \tag{3.6}$$

Note that this formulation ensures that each of these three basis functions defines a parabola that is 1 in exactly one of the points x_i, $x_{i+1/2}$ or x_{i+1}, and zero in the other two. The basis functions are depicted in Figure 3.6. The mid point basis functions $\phi_{i+1/2}$ are zero outside element i, and are therefore continuous. The mid point basis functions are only nonzero on a single element. The vertex basis functions however are nonzero over the two elements around its corresponding vertex.

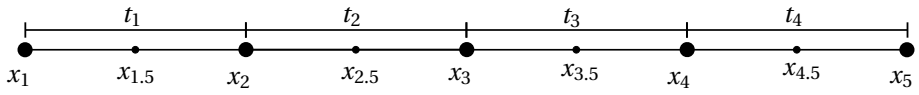


Figure 3.5: A 1D equidistant grid with 5 vertices V and 4 elements t. The grid has been refined with one extra vertex in each element t.

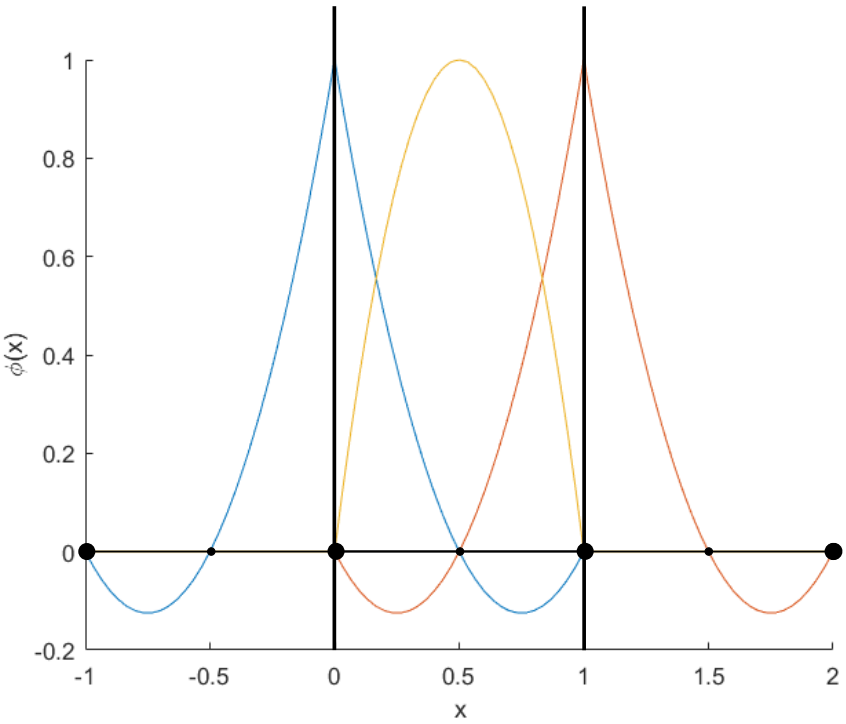


Figure 3.6: The quadratic Lagrangian basis functions corresponding to a reference element $[0,1]$.

For a higher-order Lagrangian basis, the basis functions are defined in a similar fashion. For basis functions of order p, first refine each element with $p-1$ equidistant nodes. Then the basis functions over this element are defined as p^{th}-order polynomials that are 1 in exactly one of the points, and 0 in all the other points. The general way of constructing such a polynomial is given by

$$\phi_{i+k/(p-1)} = \prod_{j=0, j\neq k}^{p} \frac{x - x_{i+j/(p-1)}}{x_{i+k/(p-1)} - x_{i+j/(p-1)}}. \tag{3.7}$$

When x is equal to one of the other nodes $x_{i+j/(p-1)}$, then the basis function is clearly zero, and when $x_{i+k/(p-1)}$ is filled in for x, then numerator and denominator are always the same, so then the function will be 1. Basis functions for several polynomial degrees are depicted in Figure 3.7. Note that all the higher-order basis functions contain negative parts, which makes lumping impossible for use in MPM. Furthermore, the functions are only C^0-continuous over the edges, which causes grid-crossing problem in MPM as well. Next the Lagrangian basis functions will be considered in 2D, which are constructed similarly, but also have the same drawbacks as the 1D basis functions.

3.3.1. 2D Lagrangian basis functions

For two-dimensional MPM, a triangular is used. On this grid, the Lagrangian basis functions are still piece-wise polynomials. In each element, nodes are inserted for higher-order basis functions similar to the 1D functions. Again the basis functions are defined to be smooth polynomials that are 1 in one of the nodes and 0 in all the other nodes in the element. First, consider the piece-wise linear case. The elements do not have to be refined for this. Each 2D triangular element has three nodes, which are the corners of the triangle. Therefore, there are three basis functions over each triangle. The basis functions are easiest described using the barycentric coordinates with respect to the triangle. For a triangle, the basis functions over the element in barycentric

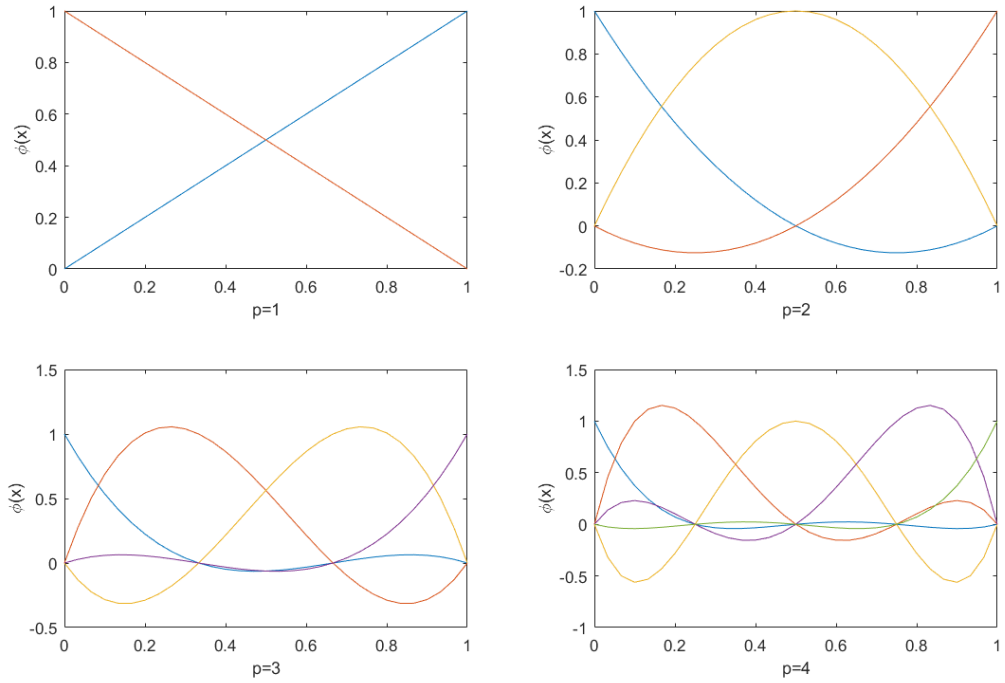


Figure 3.7: The Lagrange basis functions over a single element for various polynomial orders p.

coordinates $(\zeta_1, \zeta_2, \zeta_3)$ are

$$\phi_1(\boldsymbol{\zeta}) = \zeta_1, \tag{3.8}$$
$$\phi_2(\boldsymbol{\zeta}) = \zeta_2, \tag{3.9}$$
$$\phi_3(\boldsymbol{\zeta}) = \zeta_3. \tag{3.10}$$

Recall that the barycentric coordinates $(\zeta_1, \zeta_2, \zeta_3)$ of the vertices are $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$, so again for each of the three basis functions the property δ_{ij} holds. Figure 3.8 shows one of these basis functions over a single element.

In order to get higher-order basis functions over the triangulation, first the grid has to be refined. If a p^{th}-order basis function is required, then first each element is refined with $p-1$ nodes along each edge and in the domain in correspondence with the edges. The nodes corresponding with this refinement are defined as by their barycentric coordinates $\left(\frac{k}{p}, \frac{l}{p}, \frac{m}{p}\right)$, in which $k, l, m \in \mathbb{N}$ are integers. Furthermore, the nodes must be inside the triangle, so $0 \leq k, l, m \leq p$ and $k + l + m = p$. Let n_{klm} be the node with barycentric coordinates $\frac{1}{p}(k, l, m)$. An example of this refinement is shown in Figure 3.9.

Next, the basis functions are defined over the refined grid. The basis functions are of order p, and should be 1 in exactly one of the nodes in the grid and 0 everywhere else. Consider basis function ϕ_{klm}, which is 1 in n_{klm} and 0 in all the other nodes in the considered element. Then ϕ_{klm} is defined as

$$\phi_{klm} = \left(\prod_{q=0}^{k-1} \frac{\zeta_1 - \frac{q}{p}}{\frac{k}{p} - \frac{q}{p}}\right) \cdot \left(\prod_{r=0}^{l-1} \frac{\zeta_2 - \frac{r}{p}}{\frac{k}{p} - \frac{r}{p}}\right) \cdot \left(\prod_{s=0}^{m-1} \frac{\zeta_3 - \frac{s}{p}}{\frac{k}{p} - \frac{s}{p}}\right). \tag{3.11}$$

Though this equation may seem complicated, it is designed such that each term causes a whole row of nodes to be zero. The term $\zeta_1 - \frac{q}{p}$ causes all nodes with $\zeta_1 = \frac{q}{p}$ to be zero. In the end, only the node where $(\zeta_1, \zeta_2, \zeta_3) = \frac{1}{p}(k, l, m)$ will not be zero. An example of this procedure is shown in Figure 3.10. In this example, the node with coordinates $(1/2, 1/4, 1/4)$ is considered, marked as the large dot. The function must be zero in all other nodes. The required function corresponding to this grid is of order $p = 4$, so the basis function is a product of 4 linear terms. First the term (η_1) is added, such that the function is zero in all nodes with $\eta_1 = 0$. This line is marked

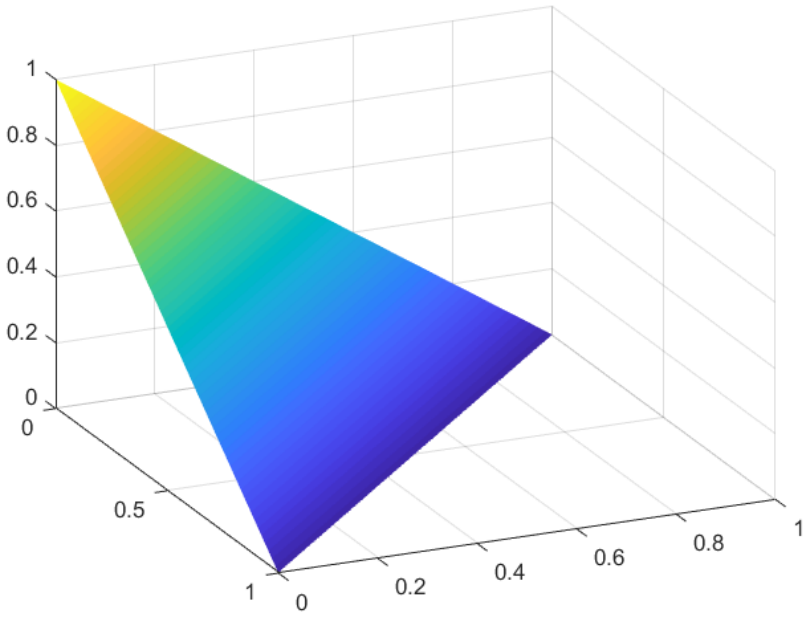


Figure 3.8: A linear basis function over a single triangular element.

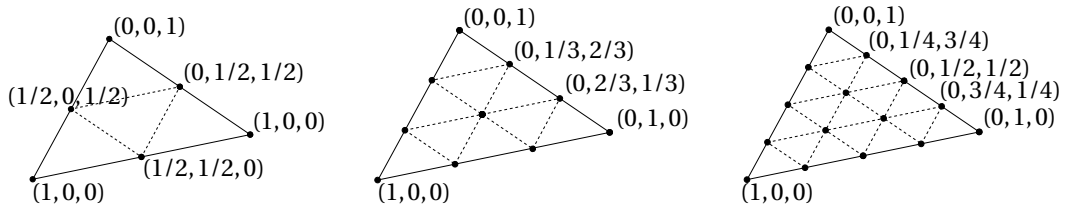


Figure 3.9: Three refinements of an element for order p Lagrange basis functions. The barycentric coordinates are $\frac{1}{p}(k,l,m)$, some of which are shown in the figure.

as the $\eta_1 = 0$ line. Likewise, this procedure continues with $\eta_1 = 1/4$, $\eta_2 = 0$ and $\eta_3 = 0$. The only non-zero node left is the marked one. The raw resulting formula is then $\phi_{211} = \frac{1}{c}(\eta_1 - 0)(\eta_1 - 1/4)(\eta_2 - 0)(\eta_3 - 0)$, with c a normalisation constant.

The normalisation constant is used to guarantee that ϕ_{klm} is equal to 1 in n_{klm}. This is done by ensuring that each term in Equation 3.11 is equal to 1 in this point. This is done by dividing the numerator of each term with its value in $n_{klm} = \left(\frac{k}{p}, \frac{l}{p}, \frac{m}{p}\right)$. The result of this procedure for quadratic basis functions is shown in Figure 3.11. Note that each basis function that is not zero on all the edges should be mirrored over the edge on which it is not zero towards a neighbouring element. If the basis function is nonzero in a node of the element, then the full basis function should be extended over all the elements that include that node.

This concludes the definition of Lagrangian basis functions over triangular grids. Having defined and shown the basis functions, two properties appear that are undesirable for implementation in MPM. First, similar to 1D basis function, the 2D functions have discontinuous derivatives over triangle edges. The basis functions do not smoothly transition to zero over element edges. Also when the basis functions are extended over multiple elements by mirroring, a kink appears in the basis functions similar to Figure 3.6. These discontinuities in the gradient cause grid crossing errors, as explained in Section 2.7. Second, all basis functions that are piece-wise polynomials of second-order and higher contain negative parts, as can be seen from the figures in this section. This causes large problems, such as loss of mass conservation and the impossibility to lump the matrix, as mentioned before. For extending MPM with higher-order basis functions, the Lagrangian basis functions are most unsuitable. This is the main incentive to investigate B-spline basis functions. These are expected to overcome these drawbacks.

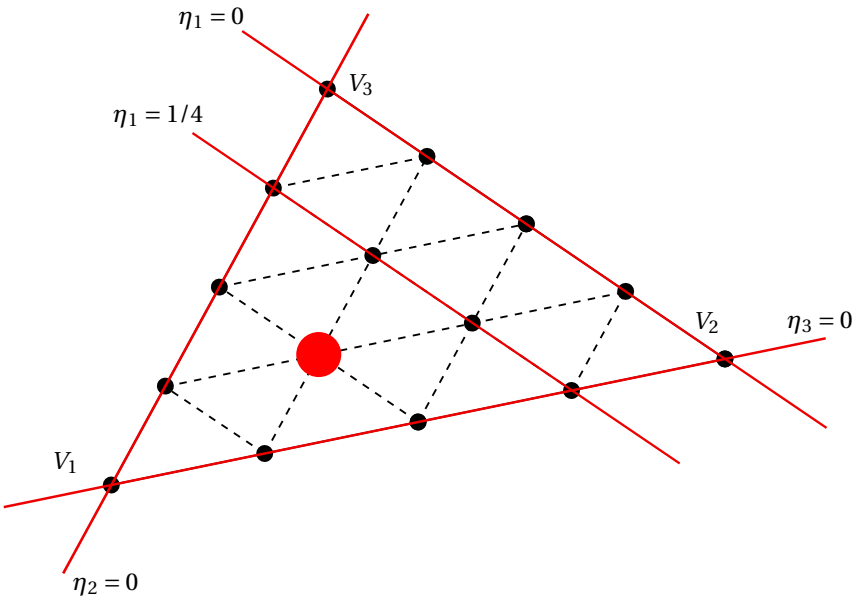
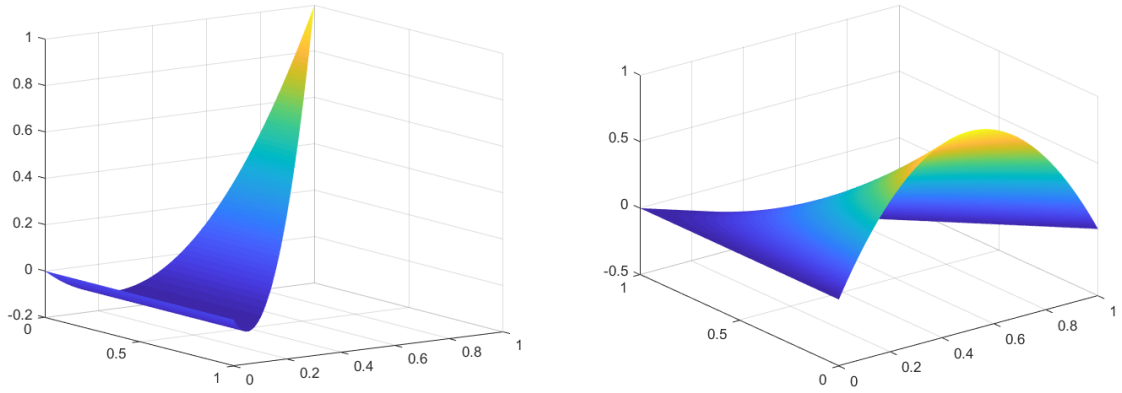


Figure 3.10: The procedure for finding a basis function that is zero in all nodes except for one.



(a) Vertex basis function

(b) Edge middle basis function

Figure 3.11: The quadratic basis functions over a triangular element. A vertex basis function is shown on the left, and a edge middle basis function is shown on the right.

3.4. SPLINE BASIS FUNCTIONS IN 1D

In this section, a set of B-spline functions will be considered in 1D. Tielen et al. [5] used 1D B-spline basis function, which were constructed with the Cox-De Boor recursion formula. The concept of these basis functions is explained in this section and the properties of these B-spline basis functions are discussed. Although these basis functions are not directly extendable to triangulations, the concept and properties may be used to construct and understand a B-spline basis on triangulations in 2D.

The desired spline basis functions for MPM should satisfy a number of properties. First, the basis functions should have a continuous gradient over the entire domain. This will prevent any grid crossing errors. Second, the functions must be non-negative. This will ensure that the mass matrix used in MPM is positive and can be lumped. Third, the basis should be a partition of unity; the basis functions should add up to 1 over the entire domain. Together with the property of non-negativity, this ensures conservation of mass in the mass-matrix.

The functions used for MPM by Tielen et al. [5] possess all of the mentioned properties. For these basis functions, first the domain should be divided in elements. As these basis functions are considered on 1D, the domain is a line, and the elements are pieces of this line separated by nodes. Over this grid, the basis functions will be defined. Next, the order p of the basis functions should be decided. The B-spline basis functions are piece wise polynomials of order p, where the polynomials are defined over each element. Let $\phi_{i,p}$ denote basis function i of order p. The B-spline basis functions are then fully defined by their *knot vector*. A knot vector is an ordered set of positions. These positions correspond with the positions of the nodes of the grid. The knot vector contains all the positions of the grid, but some nodes can be repeated multiple times. In particular, the first and last node of the domain are repeated $p+1$ times. Other nodes can be repeated as well, but first consider a knot vector without repeated inner nodes. Let

$$\Xi = \left[\xi_1, \xi_2, \ldots \xi_{n+p+1}\right] \tag{3.12}$$

be a knot vector, where n is the number of basis functions and p the order of the basis functions. For example, consider a grid with 6 equidistant vertices at the positions $x = 0,1,2,3,4,5$, and $p = 2$, resulting in with piece-wise parabolic basis functions, with C^1-continuity everywhere. Then the corresponding knot vector is $\Xi = [0,0,0,1,2,3,4,5,5,5]$.

Given a knot vector Ξ, the p^{th}-order basis functions are defined recursively using the Cox-de Boor recursion formula [15]. This recursion formula first defines piece-wise constant functions using the knot vector. From the piece-wise constants, piece-wise linear B-splines $\phi_{i,0}$ are generated, and so on. In each layer, partition of unity is maintained. From B-splines of order d, $\phi_{i,d}$, order $d+1$ B-splines $\phi_{i,d+1}$ are generated, up to the desired order p, $\phi_{i,p}$. First define piece-wise the constant functions as

$$\phi_{i,0}(\xi) = \begin{cases} 1 & \text{for } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{else.} \end{cases} \tag{3.13}$$

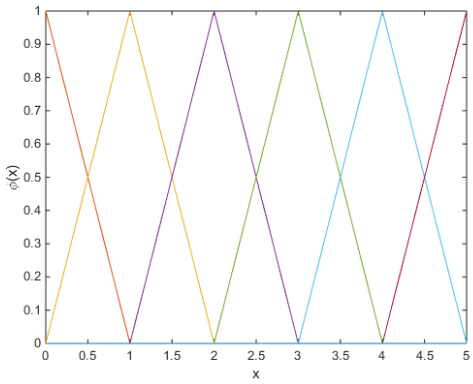
Note that it is possible that nodes in the knot vector are repeated, $\xi_i = \xi_{i+1}$, and therefore some basis functions $\phi_{i,0}$ may be zero on the entire domain. This does not pose a problem in the process though.

Next, higher-order basis functions are constructed recursively from the basis functions of one order lower,

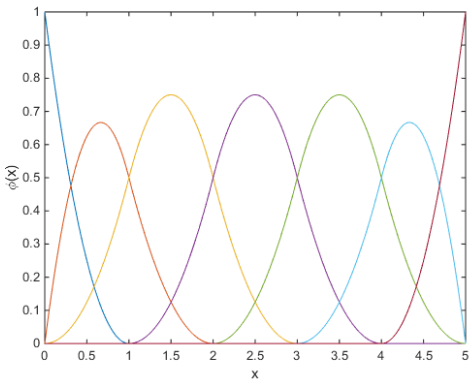
$$\phi_{i,d}(\xi) = \frac{\xi - \xi_i}{\xi_{i+d} - \xi_i}\phi_{i,d-1}(\xi) + \frac{\xi_{i+d+1} - \xi}{\xi_{i+d+1} - \xi_{i+1}}\phi_{i+1,d-1}(\xi). \tag{3.14}$$

Each higher-order basis function is constructed from two lower-order basis functions. Note that for basis function $\phi_{i,d}$ the basis function $\phi_{i+1,d-1}$ is required, so with each increase in order, there is one less basis functions than the lower level. It was assumed that ξ has $n+p+1$ elements, so there are $n+p$ basis functions of order 0, $n+p-1$ or order 1 and so on. Finally, there are exactly n basis functions of order p. Consider again the example knot vector $\Xi = [0,0,0,1,2,3,4,5,5,5]$, its corresponding basis functions of order $p = 2$ are shown in Figure 3.12 along with the lower order basis functions used to construct them. Note that the basis functions are polynomials of order p over each element, and are connected over the nodes. In each of the nodes, the p^{th}-order basis has C^{p-1}-continuity.

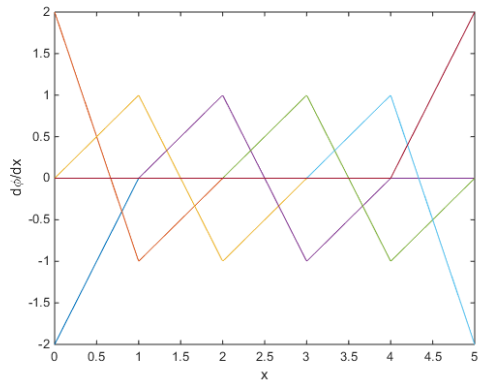
Now consider the effect of repeating inner nodes in the knot vector. For example, consider the knot vector $\Xi = [0,0,0,1,2,3,4,4,5,5,5]$. The corresponding basis functions are shown in Figure 3.13. Repeating an inner node lowers the continuity at that node of the basis function. The continuity at a node is equal to C^{p-r}, where



Linear B-splines basis functions.



Quadratic B-spline basis functions.



The derivatives of the quadratic B-spline basis functions.

Figure 3.12: The piece-wise linear and piece-wise quadratic basis functions, corresponding to the knot vector $\Xi = [0,0,0,1,2,3,4,5,5,5]$. The derivatives of the piece-wise quadratic basis functions are also shown, which illustrates the C^1-continuity of the order 2 basis functions.

r denotes the number of times the node of interest appears in the knot vector. Lowering the continuity at a point may be of interest when considering geometries with imposed discontinuities in derivatives or sudden jumps.

All together, these B-splines are non-negative, can be constructed C^1-continuity and are partition of unity. Therefore, these basis functions are very suitable for application in MPM. The basis functions are easy to construct and very flexible. The knots in the knot vector can easily be spatially refined by adding more nodes closer to each other at places of interest. High-order basis function can be generated easily as well. The basis functions have high continuity over element edges, which can be lowered in accordance with the geometry. Finally, the basis functions have the properties of locality, partition of unity and non-negativity. B-spline basis functions possess very suitable properties for implementation in 1D MPM.

For 2D, it is possible to use a tensor product of 1D knot vectors:

$$\phi_{ij,p}(x,y,z) = \phi^x_{i,p}(x) \cdot \phi^y_{j,p}(y). \tag{3.15}$$

The 2D basis function $\phi_{ij,p}$ is a tensor product of two 1D basis functions, which are individually defined using 1D knot vectors and the Cox-de Boor algorithm. Therefore, this method has the same advantages as the 1D basis functions. However, it is hard to approximate any arbitrary geometry using tensor products. The tensor grid would have to be mapped curvilinearly to the real geometry. Consider the example mapping of a 2D grid in Figure 3.14. The geometry in this example is hard to represent as a mapping Ψ from a square, parametric domain to the physical domain; finding a good division and mapping of the domain may prove very hard. Furthermore, it is hard to locally refine the grid. If the grid has to be refined in a certain part of the domain, this is done by adding additional nodes on both the x and y positions of the parametric domain, but this causes

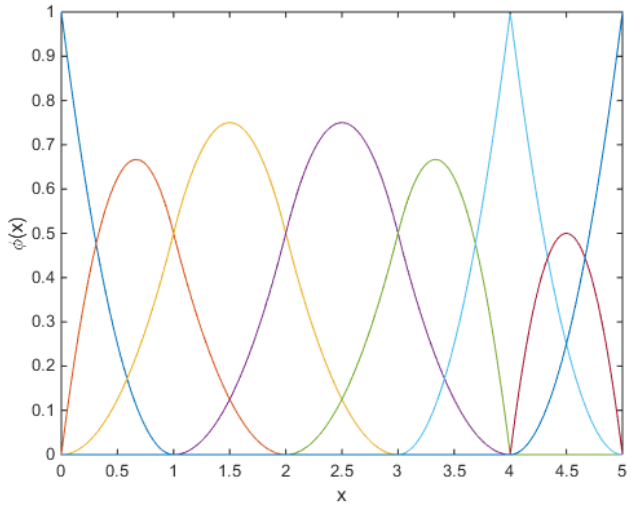


Figure 3.13: The quadratic B-spline basis functions corresponding to the knot vector $\Xi = [0,0,0,1,2,3,4,4,5,5,5]$. Note the C^0 continuity at 4.

unnecessary local refinement in other parts of the domain as well, as can also be seen in Figure 3.14. This costs more computation time when solving the solution over the grid. For these reasons, a tensor product grid is unsuitable for arbitrary geometries, whereas a triangular grid would be much more suitable for this.

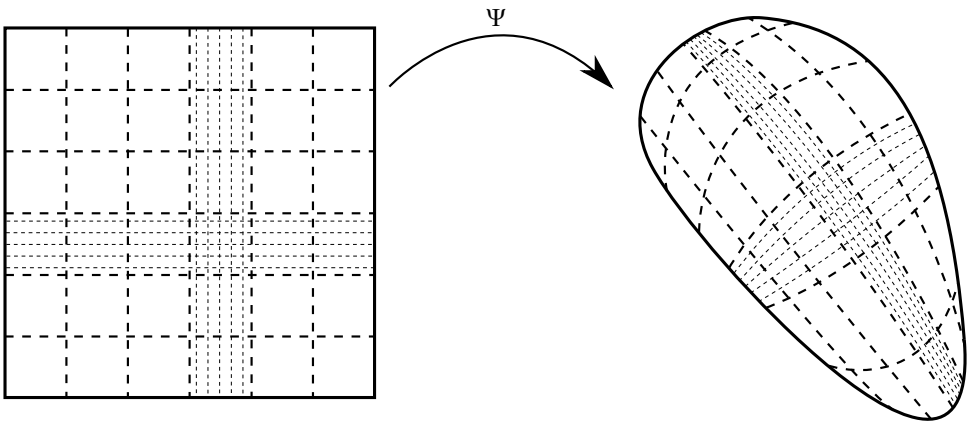


Figure 3.14: A possible mapping Ψ of a square, parametric domain to the physical domain. In the middle, the grid has been refined. The quality of this mapping is unsure, also due to the fact that the square domain has discontinuities at the corners, whereas the physical domain has a smooth boundary.

When using a triangular grid however, it is very hard to construct B-spline basis functions over arbitrary triangulations. The Cox-de Boor algorithm does not have a known equivalent on triangulations, and therefore, it is not possible yet to construct B-splines over triangles of any order with a straightforward algorithm. A tensor product approach such as for the rectangular grid is not possible over an arbitrary triangulation, so the 1D B-splines cannot be used in the current form. In the next section, it will be shown that despite these setbacks, it is possible to construct B-splines basis functions over an arbitrary triangulation, but the process will have to be derived separately for each order of basis functions. Once the basic functions have been constructed, however, the basis functions will have similar properties to the 1D spline basis functions.

3.5. Powell-Sabin splines: quadratic B-splines on arbitrary triangulations

For the construction of B-splines on arbitrary triangulations, there is not yet a general algorithm to generate these for an arbitrary order p. Therefore, the focus of this section will be on the construction of quadratic B-splines for a 2D triangulation. The construction of these B-splines are not unique; for a certain triangulation there are different possibilities to generate B-splines over the grid. The B-splines are still required to possess the properties of locality, non-negativity, partition of unity and be piece-wise polynomials over each of the (sub-)elements, with C^1-continuity over the edges. A full research by Dierckx [16] was dedicated to constructing and researching quadratic B-splines over a 2D quadratic grid, which resulted in so called *normalised Powell-Sabin B-splines*. This thesis follows the approach by Dierckx for the construction of these B-splines. These Powell-Sabin B-splines will be referred to as PS splines throughout this thesis. The potential of quadratic PS splines for function reconstruction has already been proved by Speleers et al. [14] for 2D advection-diffusion-reaction problems, and therefore PS splines are expected to improve function reconstruction in MPM as well.

Before defining the PS splines, the purpose, properties and general construction process are first summarised. The purpose of a basis of PS splines is to represent a function as a linear combination of PS spline basis functions, which are defined on an arbitrary triangulation. For each vertex i in the triangulation, three basis functions ϕ_i^q, $q \in \{1,2,3\}$ with local support will be defined associated to that vertex. The functions will be nonzero on all triangular elements directly around this vertex, and zero on all other elements, in Figure 3.23, a PS-spline basis function is shown. Any continuous function can be approximated by a linear combination of these basis functions,

$$f(x,y) \approx \hat{f}(x,y) = \sum_{i=1}^{n_v} \sum_{q=1}^{3} c_i^q \phi_i^q(x,y), \tag{3.16}$$

in which $\hat{f}$ is the approximation of f, n_v is the total number of vertices in the grid and c_i^q the coefficient corresponding to the PS spline basis function ϕ_i^q. Note that there are $3n_v$ basis functions over the grid in total. The basis functions should be non-negative and partition of unity, which are formally defined as

$$\phi_i^q(x,y) \geq 0, \quad \text{for all } (x,y) \in \Omega, \tag{3.17}$$

$$\sum_{i=1}^{n_v} \sum_{q=1}^{3} \phi_i^q(x,y) = 1, \quad \text{for all } (x,y) \in \Omega. \tag{3.18}$$

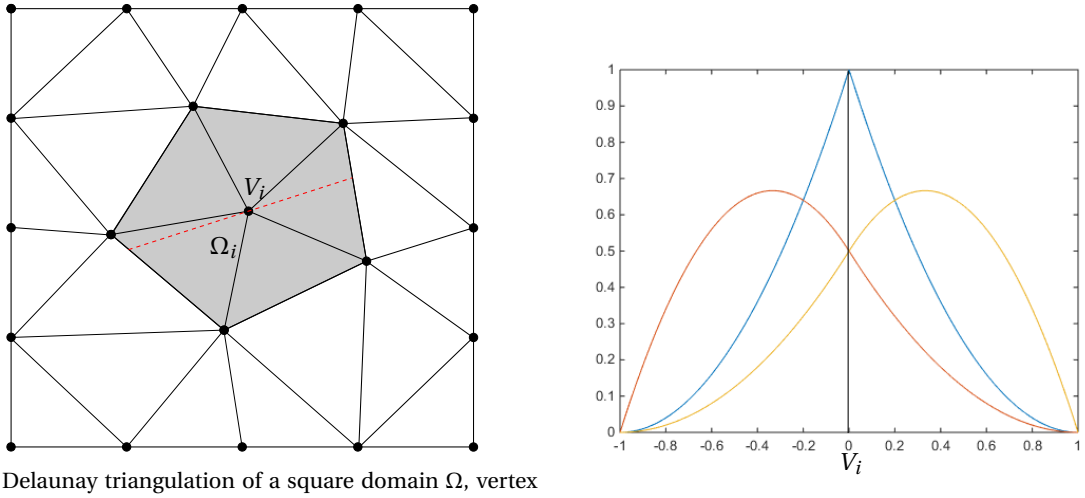
Furthermore, the functions have to be piece-wise quadratic and C^1-continuous over (sub-)element edges, so their gradient should be continuous.

In order to achieve this, first the triangular grid must be refined, where each triangular element is divided into smaller sub-elements. After the refinement of the grid, three control points will be chosen for each vertex, which form a so-called control triangle. The control triangle at each of the vertices will fully define the three basis functions $\phi_i^1, \phi_i^2, \phi_i^3$ at each vertex i, and therefore, the choice of the control triangles should guarantee suitable basis functions. First, the refinement of the grid will be discussed, then the choice of the control triangles, and finally the construction of the PS-spline basis functions from the control triangles.

3.5.1. Grid refinement

Given a domain Ω, let Δ be a triangulation of Ω. In this section the refinement process of Δ will be explained. First we will discuss the need for a refinement. Consider a vertex i and one of its three basis functions ϕ_i^q. This basis function should be locally defined on the triangles directly around node i, let this local domain be called molecule Ω_i. An example of Ω_i is considered in Figure 3.15a. The basis function ϕ_i^q should be a piece-wise parabola, in which over each element a parabola is defined with smooth transitions over the boundary of each element. At the boundary of Ω_i, the value of ϕ_i^q should be zero as well as its gradient, because the function should connect smoothly to the 0-function outside the domain Ω_i. Next consider the piece-wise parabola over the dotted line in Figure 3.15a. This line passed through two elements, and over this line, the basis function should be a C^1-continuous 1D piece-wise parabola. The piece-wise parabola should smoothly connect to the 0-function at the boundaries of the domain Ω_i. Figure 3.15b shows that it is not possible to construct such a piece-wise parabola with just two elements other than the trivial solution $\phi_i^q = 0$. Either is has a discontinuous derivative at the boundary of the domain or at the connection between the elements.

Any straight line through node i passes only two elements, and therefore the basis function must be zero over such lines. It is possible to draw such a line through any point in the domain Ω_i, and therefore, the basis function must be zero everywhere in the domain. With the unrefined triangulation Δ, the only possible basis function is the trivial function 0. This illustrates the need for a refinement of the triangulation Δ.



(a) A Delaunay triangulation of a square domain Ω, vertex V_i and its molecule Ω_i are marked.

(b) piece-wise parabolas consisting of two parts.

Figure 3.15: A triangulation (a) of a square domain. A vertex V_i and its corresponding molecule Ω_i are marked. Over this molecule, a 2D piece-wise parabola has to be defined, which must be C^1-continuous and smoothly goes to zero at the boundary. Some possibilities for the 2D piece-wise parabola are investigated over the dotted line in (a). The two elements the dotted line goes through in (a) are separated by the vertical line in (b). Some possible piece-wise parabolas are shown.

The grid Δ will be refined with a so-called Powell-Sabin refinement, in which each triangle element will be divided in six sub-triangles. Let Δ^* be the Powell-Sabin refinement of the original triangulation Δ. A possible refinement Δ^* of Δ is shown in Figure 3.16, which can be constructed in the following way:

1. Choose an interior point Z_j in each triangle t_j, such that if two triangles t_j and t_m have a common edge, the line between Z_j and Z_m intersects the edge. The intersection point is called Z_{jm}. A way to guarantee the existence of the point Z_{jm} is by choosing all Z_j as the incenter of each triangle t_j, the intersection point of the angle bisectors. Different choices are also possible, but throughout this thesis, the incenters are used as interior points.

2. Connect all Z_j to the vertices of its triangle t_j with straight lines.

3. Finally connect the Z_j to all point $Z_{j,m}$ on the edges of triangle t_j with straight lines. In case t_j is a boundary element, t_j will have at least one edge without a neighbouring element, and therefore no point $Z_{j,m}$ on this edge to connect Z_j with. In that case, join Z_j with an arbitrary point on that edge. Throughout this thesis, Z_j is connected to the middle of the edge.

3.5.2. CONTROL TRIANGLES

After refining the grid, the control triangles are created, from which finally the basis functions will be constructed. Consider the refined triangulation Δ^*. Define the Powell-Sabin (PS) points as the vertices V_i and the midpoints of all edges in the PS refinement with V_i at one end, as shown in Figure 3.17. Let PS_i be the set of PS points associated with vertex i, then the control triangle of vertex V_i must contain all the PS points in PS_i. If the control triangle of vertex V_i contains all PS points of PS_i, then non-negativity of the basis functions associated the control triangle is assured. The control triangle is not uniquely defined, Figure 3.17 shows a number of possible control triangles that all contain the PS points in PS_i. A possible choice for the control triangle is a triangle with minimal area containing PS_i, as this produces PS-splines with good stability properties [14]. Finding an optimal PS-triangle with minimal area corresponds to solving a quadratic programming problem, which can be computationally very expensive. Instead, a good PS-triangle could also be constructed using a simple algorithm, which is also used throughout this thesis. Note that an optimal PS-triangle often shares

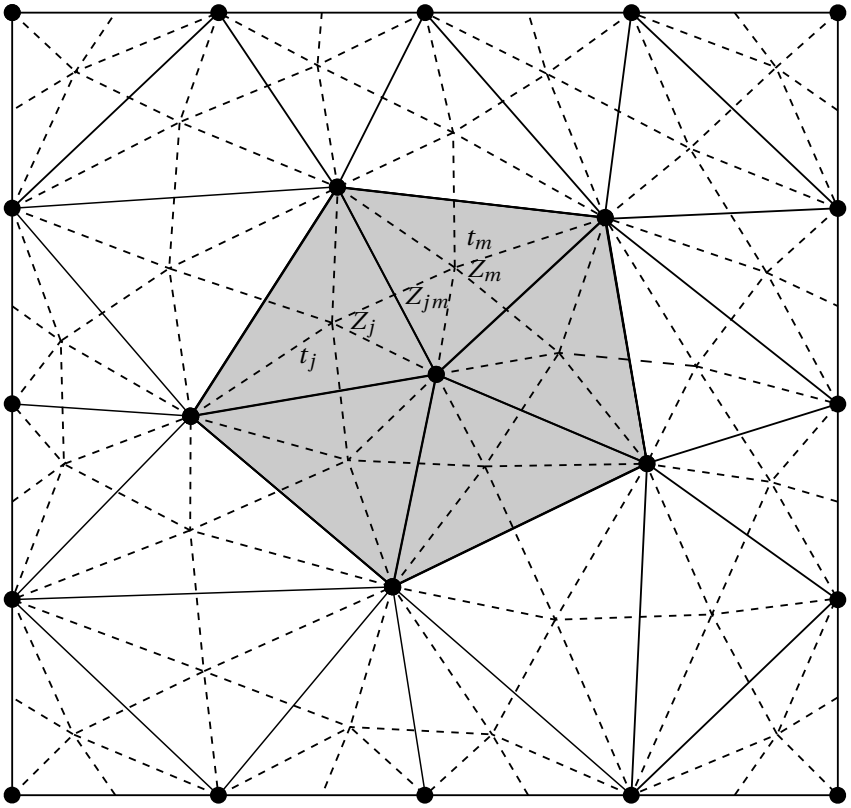


Figure 3.16: A PS refinement Δ^* of the triangulation Δ used earlier.

two or three edges with the convex hull of PS_i. Therefore, it is also possible to consider all such triangles and determine the one with the smallest area. In case of three common edges, the corresponding triangle is fully defined. In case of two common edges, take the third edge orthogonal to the bisector of the lines through the two common edges. Then let this third edge go through the PS-point such that all points in PS_i are contained in the resulting triangle.

Some technical details will be provided for implementing the algorithm for finding a near-optimal PS-triangle. In case this is not of interest to the reader, we recommend moving on to the next subsection, which explains how the PS-splines are constructed from the control triangles.

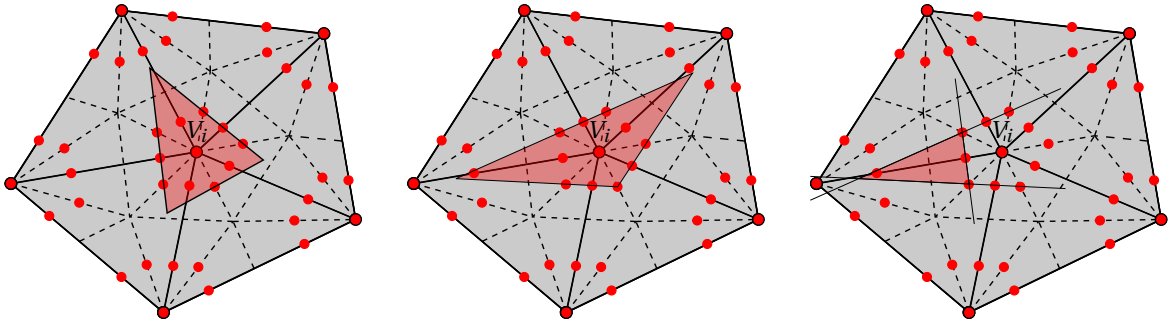
Given a set of PS points, the edges should be constructed, the intersection points between edges need to be found and from the points of the triangle, the surface must be determined. From two neighbouring PS points $\boldsymbol{x} = (x_i, y_i)$, $i = 1,2$, the formula for an edge is determined in vector form as

$$e_{12}(t) = \boldsymbol{x_1} + t\boldsymbol{v_{12}} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \zeta \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}. \tag{3.19}$$

Then the intersection point of two edges e_1 and e_2 is determined by

$$e_1(\zeta_1) = e_2(\zeta_2) \tag{3.20}$$
$$\Rightarrow \boldsymbol{x_1} + \zeta_1 \boldsymbol{v_1} = \boldsymbol{x_2} + \zeta_2 \boldsymbol{v_2} \tag{3.21}$$
$$\Rightarrow \zeta_1 \boldsymbol{v_1} - \zeta_2 \boldsymbol{v_2} = (\boldsymbol{x_2} - \boldsymbol{x_1}) \tag{3.22}$$
$$[\boldsymbol{v}_1 \boldsymbol{v}_2] \begin{bmatrix} \zeta_1 \\ \zeta_2 \end{bmatrix} = (\boldsymbol{x_2} - \boldsymbol{x_1}) \tag{3.23}$$
$$A\boldsymbol{\zeta} = \boldsymbol{b}. \tag{3.24}$$

Solving this system of equations yields a value for ζ_1 and ζ_2. The intersection point q is then determined by $\boldsymbol{q} = \boldsymbol{x_1} + \zeta_1 \boldsymbol{v_1}$.



(a) 2 common edges. (b) 3 common edges. (c) 3 common edges, but invalid.

Figure 3.17: The PS points and some possible PS-triangles of the marked molecule from the example triangulation of Figure 3.16. The PS points and the PS-triangles are marked in red. The PS-triangle of the node V_i must contain the convex hull of the PS-points directly around V_i. In (a), a PS-triangle is shown with two edges in common with the convex hull and in (b) a PS-triangle with three edges in common with the convex hull. (c) shows a triangle which also has three edges in common with the convex hull, but this one does not contain al PS point of vertex v_i, so it is invalid.

Given a control triangle with vertices $\boldsymbol{q}_i = (x_i, y_i), i = 1,2,3$, its surface S can be determined by

$$S(\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3) = \frac{1}{2}\left|x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)\right|. \tag{3.25}$$

Note that when considering a triangle with three edges of the convex hull, it may happen that the center point is not contained, see Figure 3.17c. Special attention should be paid to ensure this case is excluded.

For the triangles with two edges in common with the hull, the bisector of two edges $e_i = \boldsymbol{x_i} + t\boldsymbol{v_i}$, $i = 1,2$ should also be determined. The bisector $\boldsymbol{b}_{12}$ is given by

$$\boldsymbol{b_{12}}(\zeta) = \boldsymbol{q}_{12} + \zeta \boldsymbol{v_{12}} = \boldsymbol{q_{12}} + \zeta\left(\frac{\boldsymbol{v}_1}{|\boldsymbol{v}_1|} + \frac{\boldsymbol{v}_2}{|\boldsymbol{v}_2|}\right), \tag{3.26}$$

in which $\frac{\boldsymbol{v}_i}{|\boldsymbol{v}_i|}$ is the normalised direction vector of e_i, and $\boldsymbol{q}_{12}$ is the intersection point of e_1 and e_2. Next the orthogonal line to the bisector should be moved far enough away from $\boldsymbol{q}_{12}$, such that all PS points are contained in the corresponding triangle, see Figure 3.18. This corresponds to finding the PS point with the greatest distance to point q_{12} in the semi-norm $\mathrm{d}(\boldsymbol{x}, \boldsymbol{q}_{12}) = |(\boldsymbol{x} - \boldsymbol{q}_{12}) \cdot \hat{\boldsymbol{b}}_{12}|$. This semi-norm only takes into account the distance from $\boldsymbol{x}$ to $\boldsymbol{q}_{12}$ in the direction of the normalised bisector $\hat{\boldsymbol{b}}_{12}$. When the PS point with the greatest distance is found, a line orthogonal to $\boldsymbol{b}_{12}$ is constructed through this PS point and intersected with the other two edges. The triangle with the smallest surface is stored and used for the PS triangle. This concludes the construction details of the control triangle. In the next subsection, the construction of the basis function from the control triangle is discussed.

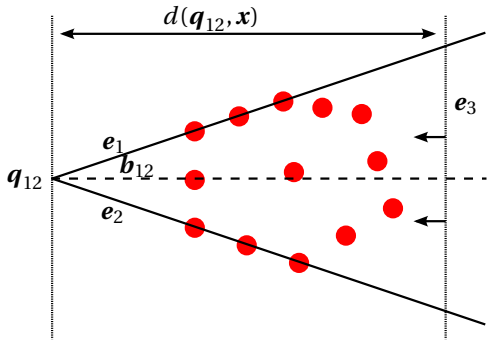


Figure 3.18: Illustration for construction of the last edge for a PS triangle with two shared edges with the convex hull of PS points. The last edge is a line orthogonal to the bisector $\boldsymbol{b}_{12}$ and goes through the PS-point with the greatest distance $d(\boldsymbol{q}_{12}, \boldsymbol{x})$. This corresponds to sliding the $\boldsymbol{e}_3$ to the left until it collides with a PS-point.

3.5.3. PS SPLINE BASIS FUNCTION CONSTRUCTION

In this section, the construction of piece-wise quadratic basis function from the PS-triangles is explained. During this section, the choice for the control triangles will also become clear. For a full derivation, see the papers of Dierckx [16, 17].

First, consider the construction of a parabola over a triangle. In Cartesian coordinates, this 2D parabola can be expressed as

$$p(x,y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2. \tag{3.27}$$

Any parabola can be described using 6 degrees of freedom $c_1, \ldots, c_6$. Instead of Cartesian coordinates, the parabola can also be described in barycentric coordinates, Equation 3.1. Using the barycentric coordinates $\boldsymbol{\zeta}$ of triangle t, it is possible to describe any parabola over a triangle using the so called Bernstein polynomials of degree 2, $B^2_{i,j,k}$, which also form a partition of unity. The Bernstein polynomials are defined as

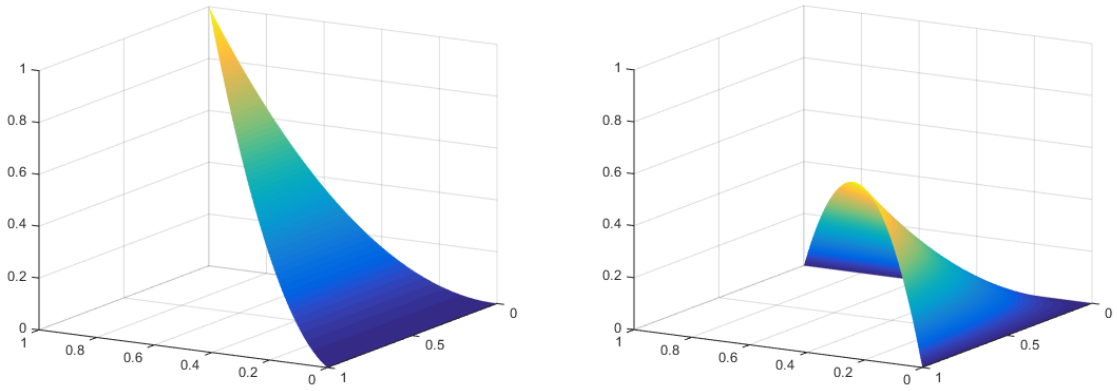
$$B_{i,j,k}(\boldsymbol{\zeta}) = \frac{2!}{i!j!k!}\zeta_1^i \zeta_2^j \zeta_3^k, \tag{3.28}$$

with the properties of non-negativity and partition of unity:

$$B^2_{i,j,k}(\boldsymbol{\zeta}) \geq 0 \quad \forall (x,y) \in t, \tag{3.29}$$

$$\sum_{\substack{i+j+k=2,\\ i,j,k\geq 0}} B^2_{i,j,k}(\boldsymbol{\zeta}) = 1 \quad \forall (x,y) \in t. \tag{3.30}$$

The Bernstein polynomials of degree 2 can be divided in two sets, the set with indices $(2,0,0), (0,2,0), (0,0,2)$, and the set with indices $(1,1,0), (0,1,1), (1,0,1)$. The first set contains the parabolas that attain their maximum in one corner and have both value and gradient 0 at the opposite edge in the triangle. The second group attain their maximum halfway one of the edges and are zero in the other edges. Figure 3.19 shows the shape of these types of functions.



(a) Corner parabola

(b) Edge middle parabola

Figure 3.19: General shape of the quadratic Bernstein polynomials over a triangle (in Cartesian coordinates).

These six Bernstein polynomials are linearly independent quadratic polynomials, and as an arbitrary quadratic polynomial has exactly 6 degrees of freedom as in Equation 3.27, the Bernstein polynomials form a basis for all possible 2D quadratic polynomials. That is, it is possible to construct any required 2D parabola p as a linear combination of the six Bernstein parabolas,

$$p(x,y) := b(\boldsymbol{\zeta}) = \sum_{\substack{i+j+k=2,\\ i,j,k\geq 0}} b_{i,j,k} B^2_{i,j,k}(\boldsymbol{\zeta}). \tag{3.31}$$

The coefficients $b_{i,j,k}$ are called the Bézier ordinates of the polynomial $b(\boldsymbol{\zeta})$. This quadratic polynomial can be schematically represented by filling in all its Bézier ordinates $b_{i,j,k}$ at the coordinates $(i/2, j/2, k/2)$ in triangle t. These barycentric coordinates correspond with the three vertices of t and to the three midpoints of

the edges, see Figure 3.20. Each of these positions $(i/2, j/2, k/2)$ also represent the point in the triangle where $B_{i,j,k}$ is maximal, compare Figure 3.19 with Figure 3.20.

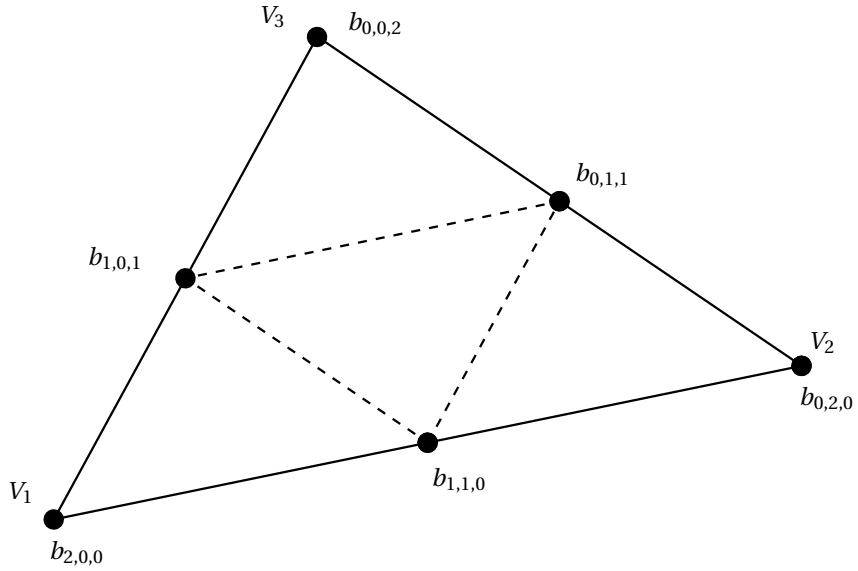


Figure 3.20: The Bézier ordinates of a parabola over a triangle. Each Bézier ordinate is positioned at the location where its corresponding Bernstein polynomial has its maximum (see Figure 3.19).

Adopting the schematic notation using Bézier ordinates, it is very straightforward to define a parabola over a triangle. This notation turns out to be very convenient for multiple reasons. From Equation 3.28 and corresponding Figure 3.19, it can be seen that only Bernstein polynomials with Bézier ordinates on edge e are non-zero on this edge, and all other Bernstein polynomials are zero there. Therefore, the value at an edge e is fully determined by the Bézier ordinates that are indicated on that edge. If two triangles are connected by an edge e, and the parabolas over each of the elements are required to connect continuously, it is only required that they have the same Bézier ordinates on that edge. In order to have a smooth gradient as well, an additional condition can be imposed on the Bézier ordinates one step away from that edge. This condition is more complicated, and we refer to [16] for this. Finally, note from the corner parabola in Equation 3.19 that both its values and gradients at its far edge are zero, so this value can be chosen freely without losing C^1-continuity over the far edge. A full basis function over a triangular element can then be described by filling in the Bézier ordinates on the corresponding points in its sub-triangles, as shown in Figure 3.21.

It was shown that an arbitrary piece-wise quadratic basis function $\phi_i^q, q = 1,2,3$ can be fully expressed in its Bézier ordinates over a triangular element. As the PS-triangle around vertex V_i should fully define the three basis functions ϕ_i^q associated with vertex V_i, the Bézier ordinates must be derived from the PS-triangle. To do so, first note that ϕ_i^q, $q = 1,2,3$ are only nonzero in the molecule Ω_i, and zero on its boundary. This implies that the values and gradients of the basis function ϕ_i^q are zero in all grid vertices V_j, with $j \neq i$,

$$\phi_i^q(V_j) = 0, \quad \text{if } i \neq j, \tag{3.32}$$

$$\nabla\phi_i^q(V_j) = 0, \quad \text{if } i \neq j. \tag{3.33}$$

This also implies that only the three basis function ϕ_i^q associated with this node V_i are nonzero at the position of V_i. For each of the three basis function ϕ_i^q at V_i, define the *triplet*,

$$\left(\alpha_i^q, \beta_i^q, \gamma_i^q\right) = \left(\phi_i^q(V_i), \frac{\partial\phi_i^q}{\partial x}(V_i), \frac{\partial\phi_i^q}{\partial y}(V_i)\right), \tag{3.34}$$

with the basis function value, its partial derivative to x and its partial derivative to y at V_i. Note that this

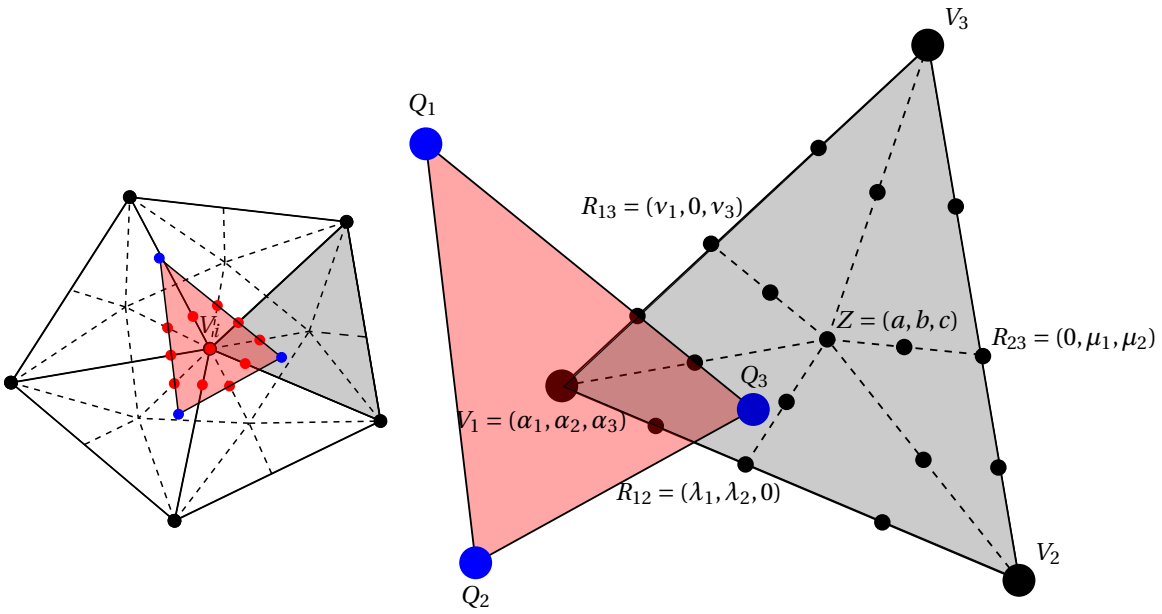


Figure 3.21: Considering a node V_i of the triangulation Δ, the molecule Ω_i is shown on the left, along with a suitable PS-triangle, marked red. Only the triangular element marked in grey and the PS-triangle on the right are considered here. The locations of the Bézier ordinates are marked to construct basis function ϕ_i^q, corresponding to node V_i and PS-triangle vertex Q_q. The location of V_1 is given in barycentric coordinates with respect to the PS-triangle. The locations of R_{12}, R_{23}, R_{13} and Z are given in barycentric coordinates with respect to the triangular element V_1, V_2, V_3. All other Bézier ordinate locations are on the edge middles of the mentioned vertices.

procedure takes place locally in each molecule Ω_i, so the index i will be dropped whenever possible. The triplet of the three basis functions are then $(\alpha^1, \beta^1, \gamma^1)$, $(\alpha^2, \beta^2, \gamma^2)$ and $(\alpha^3, \beta^3, \gamma^3)$.

Next recall the partition of unity property in Equation 3.18. From this it follows that $\phi^1(V)+\phi^2(V)+\phi^3(V) = 1$, and $\nabla(\phi^1(V) + \phi^2(V) + \phi^3(V)) = \mathbf{0}$ and therefore,

$$\alpha^1 + \alpha^2 + \alpha^3 = 1, \tag{3.35}$$
$$\beta^1 + \beta^2 + \beta^3 = 0, \tag{3.36}$$
$$\gamma^1 + \gamma^2 + \gamma^3 = 0. \tag{3.37}$$

Next consider the reconstruction of a function at the center node $V = (x, y)$, $\hat{f}(V) = c_1\phi^1(V) + c_2\phi^2(V) + c_3\phi^3(V)$, then the following system holds for the coefficients c:

$$c^1\alpha^1 + c^2\alpha^2 + c^3\alpha^3 = \hat{f}(x, y), \tag{3.38}$$
$$c^1\beta^1 + c^2\beta^2 + c^3\beta^3 = \hat{f}_x(x, y), \tag{3.39}$$
$$c^1\gamma^1 + c^2\gamma^2 + c^3\gamma^3 = \hat{f}_y(x, y). \tag{3.40}$$

Next, let the x-coordinates of the three vertices $Q^q = (X^q, Y^q)$, $q = 1, 2, 3$ of the PS triangle represent the coefficients for the reconstruction of the function $\hat{f} = x$. Likewise, let the y-coordinates be the coefficients for the reconstruction of the function $\hat{f} = y$. This results in the following systems of equations:

$$X^1\alpha^1 + X^2\alpha^2 + X^3\alpha^3 = x, \tag{3.41}$$
$$X^1\beta^1 + X^2\beta^2 + X^3\beta^3 = 1, \tag{3.42}$$
$$X^1\gamma^1 + X^2\gamma^2 + X^3\gamma^3 = 0, \tag{3.43}$$

and

$$Y^1\alpha^1 + Y^2\alpha^2 + Y^3\alpha^3 = y, \tag{3.44}$$
$$Y^1\beta^1 + Y^2\beta^2 + Y^3\beta^3 = 0, \tag{3.45}$$
$$Y^1\gamma^1 + Y^2\gamma^2 + Y^3\gamma^3 = 1. \tag{3.46}$$

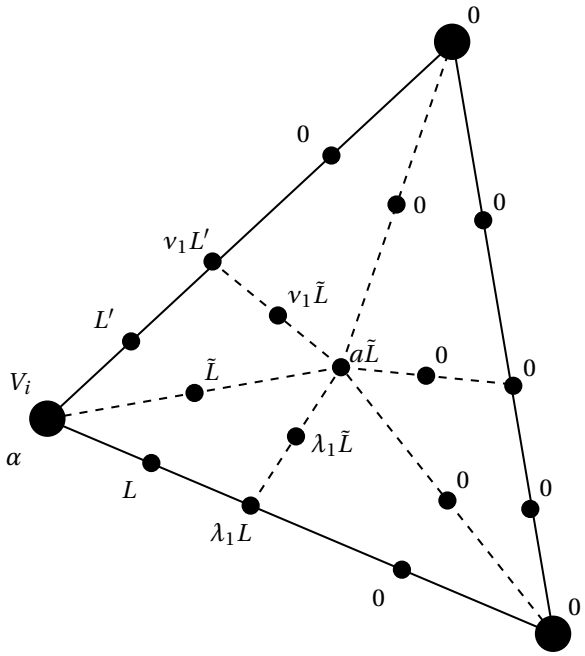


Figure 3.22: The Bézier ordinates for basis function ϕ_i^1, corresponding to the vertex V_i considered in Figure 3.21. There are three different basis functions, note that for each of these basis functions, α, β and γ should correspond with the triplet $(\alpha^q, \beta^q, \gamma^q)$ corresponding to PS-triangle vertex Q_q.

Clarifying the function of the PS-triangle, the location of its vertices represent the coefficients to reconstruct the functions $\hat{f} = x$ and $\hat{f} = y$. From these two function along with the function $\hat{f} = 1$, any triplet (α, β, γ) can be constructed as a linear combination from these functions. The next thing to do is to determine the triplets $(\alpha^q, \beta^q, \gamma^q)$ of each of the basis functions ϕ^q, $q = 1,2,3$. The values of α can be determined by considering Equations 3.35, 3.41 and 3.44. The values of β and γ can be determined likewise with Equations 3.36, 3.42 and 3.45 for β and with Equations 3.37, 3.43 and 3.46 for γ. This results in the following three systems of equations:

$$\begin{bmatrix} X^1 & X^2 & X^3 \\ Y^1 & Y^2 & Y^3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha^1 \\ \alpha^2 \\ \alpha^3 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{3.47}$$

$$\begin{bmatrix} X^1 & X^2 & X^3 \\ Y^1 & Y^2 & Y^3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \beta^1 \\ \beta^2 \\ \beta^3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{3.48}$$

$$\begin{bmatrix} X^1 & X^2 & X^3 \\ Y^1 & Y^2 & Y^3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \gamma^1 \\ \gamma^2 \\ \gamma^3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}. \tag{3.49}$$

Note that the values for α^q, $(q = 1,2,3)$ correspond to the barycentric coordinates of $V = (x, y)$ with respect to the PS-triangle, see Equation 3.1. Furthermore, let A denote the matrix used in the systems of equations. Then the values of β^q and γ^q can be determined by considering the inverse of the matrix A. A co-factor expansion approach can be used to find the inverse, which results in an explicit formula for β and γ,

$$A^{-1} = \begin{bmatrix} X^1 & X^2 & X^3 \\ Y^1 & Y^2 & Y^3 \\ 1 & 1 & 1 \end{bmatrix}^{-1} = \frac{\begin{bmatrix} X^2 - Y^3 & X^2 - X^3 & X^2 Y^3 - X^3 Y^2 \\ Y^3 - Y^1 & X^3 - X^1 & X^3 Y^1 - X^1 Y^3 \\ X^1 - Y^2 & X^1 - X^2 & X^1 Y^2 - X^2 Y^1 \end{bmatrix}}{\begin{vmatrix} X^1 & X^2 & X^3 \\ Y^1 & Y^2 & Y^3 \\ 1 & 1 & 1 \end{vmatrix}}. \tag{3.50}$$

By Equations 3.48 and 3.49, the values of β are given by the first column of the inverse of A and γ by the second

column of A:

$$\begin{bmatrix}\beta^1\\ \beta^2\\ \beta^3\end{bmatrix} = A^{-1}\begin{bmatrix}1\\0\\0\end{bmatrix} = \frac{1}{d}\begin{bmatrix}Y^2-Y^3\\ Y^3-Y^1\\ Y^1-Y^3\end{bmatrix}, \tag{3.51}$$

$$\begin{bmatrix}\gamma^1\\ \gamma^2\\ \gamma^3\end{bmatrix} = A^{-1}\begin{bmatrix}0\\1\\0\end{bmatrix} = \frac{1}{d}\begin{bmatrix}X^2-X^3\\ X^3-X^1\\ X^1-X^3\end{bmatrix}, \tag{3.52}$$

in which d is the determinant of the matrix A

$$d = \begin{vmatrix}X^1 & X^2 & X^3\\ Y^1 & Y^2 & Y^3\\ 1 & 1 & 1\end{vmatrix}. \tag{3.53}$$

From the PS-triangle, three triplets $\left(\phi^q(V), \frac{\partial\phi^q}{\partial x}(V), \frac{\partial\phi^q}{\partial y}(V)\right) = (\alpha^q, \beta^q, \gamma^q)$, $q \in \{1,2,3\}$ have been determined. Each triplet defines the value, and x- and y-derivative of its corresponding basis functions ϕ^q in the node V. From these triplets, the full piece-wise quadratic basis functions are uniquely defined with the properties of partition unity, non-negativity and C^1-continuity by an algorithm provided by earlier work of Dierckx [17]. The resulting basis functions take the prescribed values and derivatives of their triplets. The algorithm provides the Bézier ordinates over a full element (consisting of 6 sub-elements) depending on the corresponding triplets. Consider an element t which has $V = V_1 = (x_1, y_2)$ as one of its vertices and let the other vertices be $V_2 = (x_2, y_2)$ and $V_3 = (x_3, y_3)$. Let the center point of the refinement be Z, with barycentric coordinates (a, b, c) with respect to the vertices V_1, V_2 and V_3. Finally, let the refinement points on the edges have barycentric coordinates $R_{1,2} = (\lambda_1, \lambda_2, 0)$, $R_{2,3} = (0, \mu_2, \mu_3)$ and $R_{1,3} = (\nu_1, 0, \nu_3)$, as shown in Figure 3.21. Then the Bézier ordinates of the corresponding basis function ϕ_i^q are shown in Figure 3.22. This fully defines the basis function.

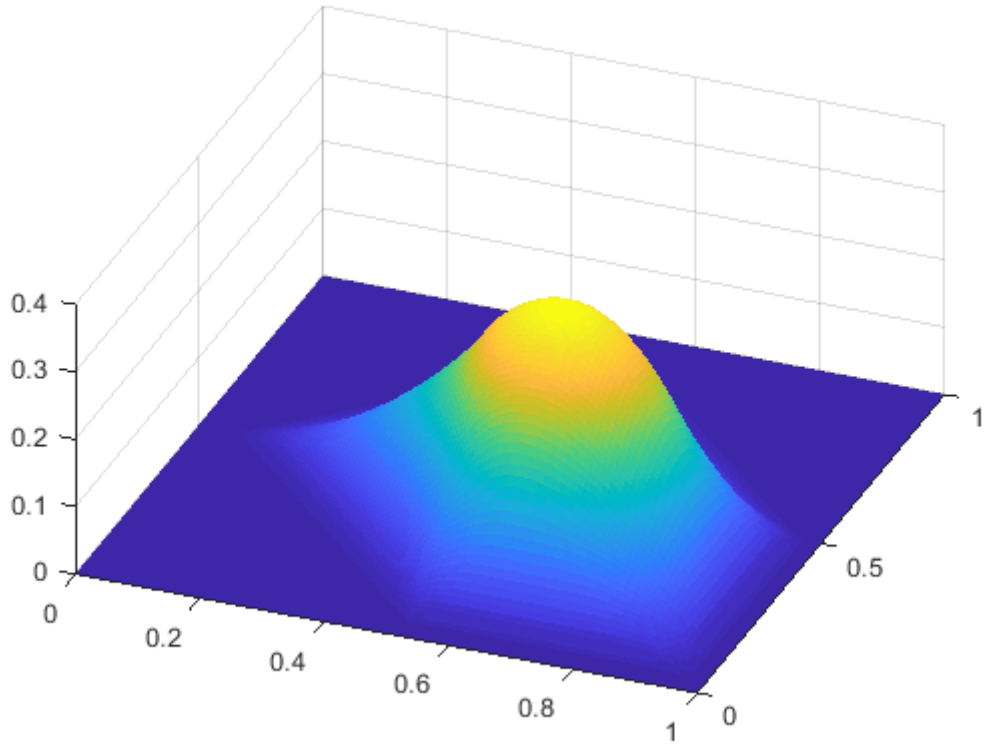


Figure 3.23: A C^1 continuous PS-spline basis function over a triangular grid.

The coefficients in Figure 3.22 are then given in terms of the triplet $(\alpha^q, \beta^q, \gamma^q) = (\alpha, \beta, \gamma)$

$$L = \alpha + \frac{1-\lambda_1}{2}\overline{\beta}, \tag{3.54}$$

$$L' = \alpha + \frac{1-\nu_1}{2}\overline{\gamma}, \tag{3.55}$$

$$\tilde{L} = \alpha + \frac{b}{2}\overline{\beta} + \frac{c}{2}\overline{\gamma}, \tag{3.56}$$

with

$$\overline{\beta} = \beta(x^2 - x^1) + \gamma(y^2 - y^1), \tag{3.57}$$

$$\overline{\gamma} = \beta(x^3 - x^1) + \gamma(y^3 - y^1). \tag{3.58}$$

Figure 3.23 shows an example of a resulting piece-wise quadratic 2D basis function constructed from the Bézier ordinates determined in this way.

3.5.4. Imposing boundary conditions with PS-splines basis functions

In this section we will describe how boundary conditions can be imposed on a function field, that is reconstructed with PS-splines. In MPM, we are mainly interested in homogeneous Dirichlet boundary conditions, and therefore, this will be the focus of this section.

The PS-spline basis functions will be used to reconstruct a function field f in MPM as a linear combination of PS-spline basis function ϕ_i^q,

$$f(\boldsymbol{x}) \approx \hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n_v} \sum_{q=1}^{3} c_i^q \phi_i^q(\boldsymbol{x}),$$

in which c_i^q is the coefficient of basis function ϕ_i^q, and n_v is the number of vertices V_i of the triangulation, and for each triangulation vertex V_i, there are three associated PS-spline basis functions ϕ_i^q, $q \in \{1,2,3\}$. For imposing a Dirichlet boundary condition at V_i, recall that its three associated basis function ϕ_i^q are the only ones that can be non-negative in V_i. Each basis function is constructed from a triplet $(\alpha_i^q, \beta_i^q, \gamma_i^q)$, which denotes its value and partial derivatives in its associated vertex V_i. We will focus again on vertex V_i and drop the index i in the rest of the section. Equation 3.34 describes the triplet, and is repeated here for convenience:

$$\left(\alpha^q, \beta^q, \gamma^q\right) = \left(\phi^q(V), \frac{\partial \phi^q}{\partial x}(V), \frac{\partial \phi^q}{\partial y}(V)\right).$$

Only the three basis function ϕ^q, $q \in \{1,2,3\}$ associated with V can be non-zero in V, which can be described as follows:

$$\hat{f}(V) = c^1\phi^1(V) + c^2\phi^2(V) + c^3\phi^3(V). \tag{3.59}$$

Next, the information of the triplets can be substituted in the above equation for the function value and for function the partial derivatives as well (see also Equations 3.38-3.40):

$$\hat{f}(V) = c^1\alpha^1 + c^2\alpha^2 + c^3\alpha^3, \tag{3.60}$$

$$\hat{f}_x(V) = c^1\beta^1 + c^2\beta^2 + c^3\beta^3, \tag{3.61}$$

$$\hat{f}_y(V) = c^1\gamma^1 + c^2\gamma^2 + c^3\gamma^3. \tag{3.62}$$

Using this information, it is straightforward to impose a homogeneous Dirichlet boundary condition on the boundary of the domain. First, identify the vertices V that are on the boundary. Second, impose an equation

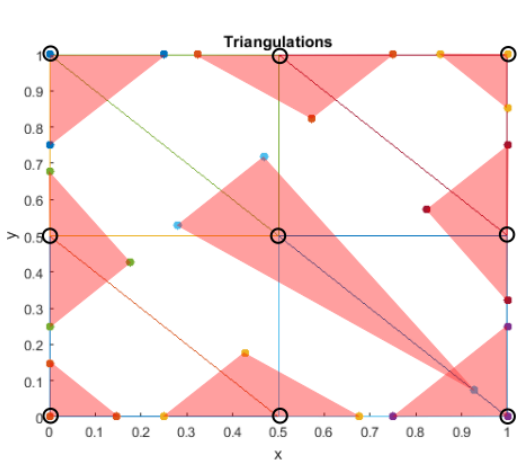
$$\hat{f}(V) = c^1\alpha^1 + c^2\alpha^2 + c^3\alpha^3 = 0, \tag{3.63}$$

to ensure that the value of the reconstructed field $\hat{f}$ is equal to 0 in boundary vertex V. To ensure that the boundary is also equal to 0 directly around V, it can also be imposed that the partial derivative along the boundary is also equal to zero:

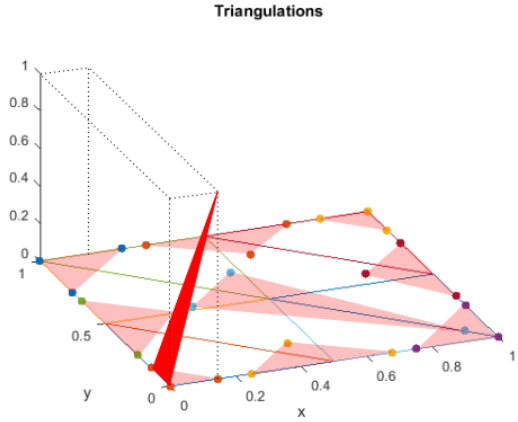
$$\begin{aligned} \nabla\hat{f}(V)\cdot\boldsymbol{n} &= 0, \\ \Rightarrow \left(c^1\beta^1 + c^2\beta^2 + c^3\beta^3\right)n_x + \left(c^1\gamma^1 + c^2\gamma^2 + c^3\gamma^3\right)n_y &= 0, \\ \Rightarrow c^1\left(\beta^1 n^x + \gamma^1 n_y\right) + c^2\left(\beta^2 n^x + \gamma^2 n_y\right) + c^3\left(\beta^3 n^x + \gamma^3 n_y\right) &= 0. \end{aligned} \tag{3.64}$$

In this equation, $\boldsymbol{n}$ is the unity direction vector tangent to the boundary in V. It may be possible that the boundary is kinked in V, in which case this approach should be further developed. In this thesis, we only investigate problems on rectangular domains, so this will not pose a problem in this thesis. By imposing Equations 3.63 and 3.64, it is guaranteed that the value of the reconstructed field is 0 in V and on the boundary near V. When solving a system of equations for a projection, these two equations should replace two of the three projection equations associated with the three basis functions of V.

If a non-homogeneous boundary condition is imposed, it is very likely that it can not be reconstructed exactly by the PS-basis functions, as it is most likely not in the space Φ spanned by all ϕ_i^q. However, it can be approximated quite well by imposing the boundary function values and derivatives in the vertices V at the boundary. In this case, the boundary condition will be exact in all boundary vertices V and be approximate around V. For doing so, Equations 3.63 and 3.64 should not be set equal to 0, but to the imposed corresponding value and partial derivative.



(a) The control triangles of each triangulation vertex.



(b) The lower left control triangle is lifted at one of its control triangle vertices. The value and gradient of this function in triangulation vertex $(0,0)$ is its triplet.

Figure 3.24: A triangulation consisting of 9 vertices and 8 triangles, the triangulation vertices are marked with black circles. The red control triangles and control triangle vertices of each grid vertex are shows as well, the control triangle vertices are marked with differing colours.

Finally, we will show another way to impose a homogeneous Dirichlet boundary condition in case a structured rectangular grid is used. Consider Figure 3.24, for this domain, the control triangle vertices are structured at the boundary. For a boundary vertex on a corner of the domain, there is one control triangle vertex right on top of it, and two on the boundary around it, see for example the lower left control triangle. For non-corner boundary vertices of the triangulation, two of their control triangle vertices are on the boundary, see for example the middle left control triangle. Each control triangle vertices is associated with a basis function, and therefore is also associated with a triplet $(\alpha^q, \beta^q, \gamma^q)$. In case the control triangles are structured in this way, it is very straightforward to implement a homogeneous Dirichlet boundary condition. Suppose a homogeneous Dirichlet boundary condition is imposed on the entire left boundary in Figure 3.24, at $x = 0$. In this discretisation there are three triangulation vertices on that boundary marked with black circles, at $(x, y) = (0,0); (0,0.5); (0,1)$. Each of those three triangulation vertices has exactly two control triangle vertices on the left boundary, at $x = 0$: two orange, two green and two blue. Setting the coefficient of the basis functions of exactly these control triangle vertices guarantees a reconstruction that is exactly 0 on the left boundary.

In general, if there is a triangulation vertex with two of its control triangle vertices on the boundary, then setting the basis functions of these two control vertices to zero guarantees that the value in V and the derivative in the direction over that boundary in V is equal to 0. The third control triangle vertex, which is not on the boundary, already has value 0 on that entire boundary, and therefore, its coefficient does not have to be set to 0. These properties can be derived from the construction of the triplets from the locations of the control triangles, as shown in Equations 3.47-3.49. In short, the triplet of a PS-basis function is generated by making a function g by "lifting" the control triangle to 1 in the control triangle vertex of that PS-function and keeping it zero in the other two. The triplet is then the value and gradient of this lifted triangle in vertex V_i. For example, consider Figure 3.24, the lower left control triangle and the control triangle vertex at $(x, y) \approx (0.15, 0)$. "Lift" this

control triangle at that control vertex to a value of 1 (in the z-direction), but leaving it at zero in the other two control triangle vertices, see Figure 3.24b. The result is a linear function g, that is 0 at the left boundary, and 1 in $(x, y) \approx (0.15, 0)$. The value of this function in $V = (0, 0)$ is zero, and also $g_y(V) = 0$, and $g_x(V) \approx 1/0.15$. Therefore, its triplet becomes $(0, 0, 1/0.15)$. More importantly, note that the function is entirely 0 over the left boundary, and therefore, its associated function ϕ_i^q is also 0 in V and has y-derivative 0 in V. Therefore, this function does not have to be set to zero to ensure a homogeneous Dirichlet boundary condition at the left boundary.

3.6. Validation of L_2-projection on B-spline basis.

In this section, we validate the projection of a function onto a basis of the quadratic B-spline basis functions. In this validation, an arbitrary analytic function is projected onto a base of quadratic B-spline basis functions defined over a triangular grid. Then the rate of convergence is measured by calculating the L_2-error for different refinements of the grid.

To investigate the rate of convergence of the L_2-projection, the unit square $\Omega = [0, 1]^2$ is considered. This domain is discretised with n equidistant vertices both in the x-direction and in the y-direction. The spacing h is therefore $1/(n-1)$, which we define for this case as the typical element length. In total, the domain is discretised using n^2 vertices, ordered in equally large squares. From these vertices, a Delaunay triangulation forms the triangular grid. A discretisation with 5 by 5 vertices is shown in Figure 3.25.

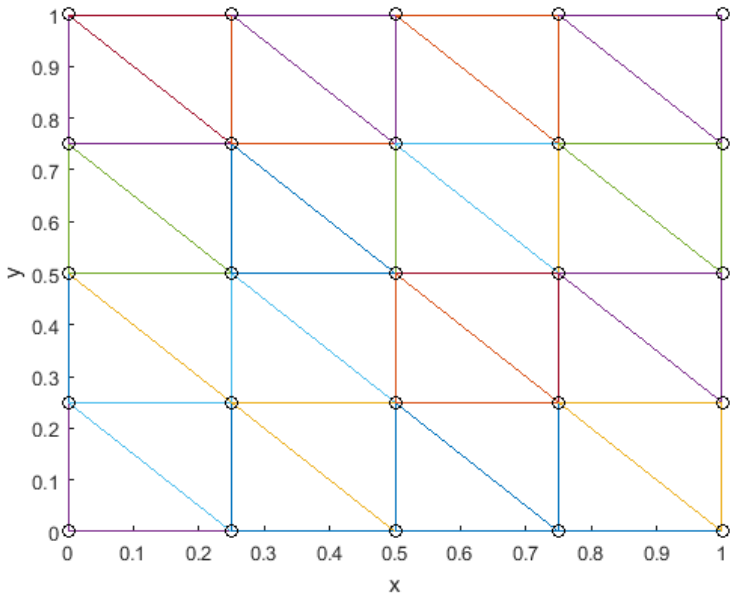


Figure 3.25: A discretisation of the unit square domain Ω, discretised using $n = 5$ vertices in each direction, which results in the shown triangular discretisation Δ.

Over the domain Ω, an analytic function will be defined. In this case, the function

$$f(x, y) = \sin(\pi x) \sin(\pi y) \tag{3.65}$$

will be projected onto the basis of B-splines. A different L_2-projection was also used earlier in Equation 2.35. For convenience, the process is repeated here without a weight-function. Let

$$\hat{f}(x, y) = \sum_{j=1}^{n^2} \sum_{r=1}^{3} c_j^r \phi_j^r(x, y) \tag{3.66}$$

be the L_2-projection of $f(x, y)$ onto the basis of B-splines ϕ_i^q. As there are n^2 vertices in the discretisation, and 3 basis functions per node, so $3n^2$ basis functions in total. For the L_2-projection $\hat{f}(x, y)$, the function f and its

approximation $\hat{f}$ must be equal in the weak form, giving a system of equations:

$$\int_\Omega \hat{f}(x,y)\phi_i^q\,\Omega = \int_\Omega f(x,y)\phi_i^q\,\mathrm{d}\Omega, \quad \text{for } i=1,\dots,n^2,\ q=1,2,3 \tag{3.67}$$

$$\text{(Substitute } \hat{f}) \quad \Rightarrow \int_\Omega \sum_{j=1}^{n}\sum_{r=1}^{3} c_j^r\phi_j^r\phi_i^q\,\mathrm{d}\Omega = \int_\Omega f(x,y)\phi_i^q\,\mathrm{d}\Omega, \quad \text{for } i=1,\dots,n^2,\ q=1,2,3 \tag{3.68}$$

$$\Rightarrow c_j^r \sum_{j=1}^{n}\sum_{r=1}^{3} \int_\Omega \phi_j^r\phi_i^q\,\mathrm{d}\Omega = \int_\Omega f(x,y)\phi_i^q\,\mathrm{d}\Omega, \quad \text{for } i=1,\dots,n^2,\ q=1,2,3 \tag{3.69}$$

$$M\boldsymbol{c} = \boldsymbol{f}, \tag{3.70}$$

in which $\boldsymbol{M}_{iq,jr} = \int_\Omega \phi_j^r\phi_i^q\,\mathrm{d}\Omega$ and $\boldsymbol{f}_{iq} = \int_\Omega f(x,y)\phi_i^q\,\mathrm{d}\Omega$. Note that for entry $M_{iq,jr}$, it may be ambiguous which column and row are meant. In this thesis, basis function ϕ_i^q is given global basis function number $3(i-1)+q$, first counting over all basis function of one vertex V_i, and then over the vertices. It is then assigned the corresponding row in the matrix.

Solving this system for $\boldsymbol{c}$ yields the coefficients for the projection $\hat{f}(x,y)$. The integration of the integrals is done with Gauss integration with 7 Gauss points per element, such that the quadrature error is insignificant. The only error of interest in this test is the projection error, which will be measured in the L_2-norm:

$$E(h) = \sqrt{\int_\Omega \left(\hat{f}-f\right)^2\,\mathrm{d}\Omega}. \tag{3.71}$$

Note that the error $E(h)$ is only a function of the spacing h of the vertices, which also represents the typical length of the triangles.

For various values of h, the projection error has been calculated. In Table 3.1, the results are presented. The table shows a reduction in error of about a factor $8 \approx 2^3$ when the spacing h is halved. This implies that $E(h) = \mathcal{O}(h^3)$, so the B-spline basis shows third-order convergence as expected. The sin-function, its projection and the error are also shown in Figure 3.26

Table 3.1: The L_2-error for the projection of the function $f(x,y) = \sin(\pi x)\sin(\pi y)$ on a basis of quadratic B-splines for different discretisations of the domain.

n	n^2	Degrees of freedom	h	$E(h)$	$^2\log(E(h)/E(2h))$
3	9	27	1/2	1.18575e-02	-
5	25	75	1/4	1.14331e-03	3.37
9	81	243	1/8	1.31622e-04	3.12
17	289	867	1/16	1.61689e-05	3.03

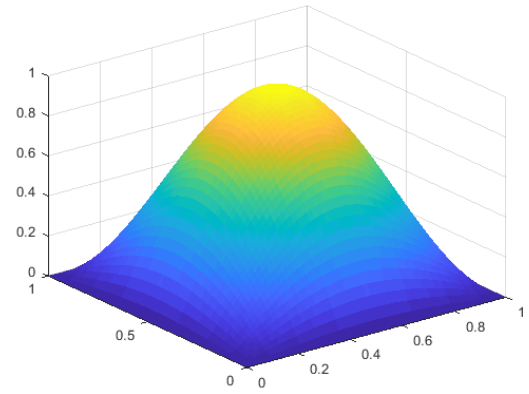
For the sake of comparison, an L_2-projection on a basis of piece-wise linear Lagrange basis functions has also been validated, using the same structure of grid as in Figure 3.25. The error of the projection on this basis is shown in Table 3.2. The table shows that the error is indeed of second-order with respect to the typical element length, $E(h) = \mathcal{O}(h^2)$.

Table 3.2: The L_2-error for the projection of the function $f(x,y) = \sin(\pi x)\sin(\pi y)$ on a basis of piece-wise linear basis functions for different discretisations of the domain.

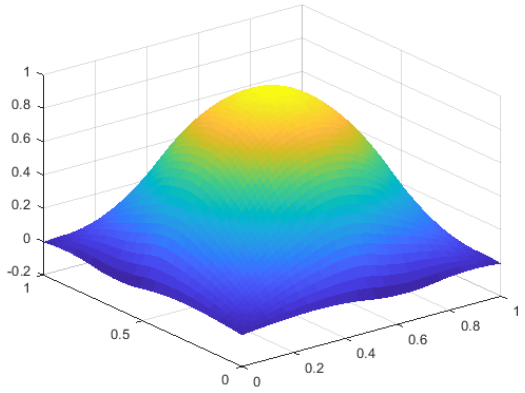
n	n^2	Degrees of freedom	h	$E(h)$	$^2\log(E(h)/E(2h))$
5	25	25	1/4	1.71200e-03	-
9	81	81	1/8	4.13089e-04	2.05
17	289	289	1/16	1.02202e-04	2.02
33	1089	1089	1/32	2.54938e-05	2.00

3.7. Cubic Powell-Sabin splines

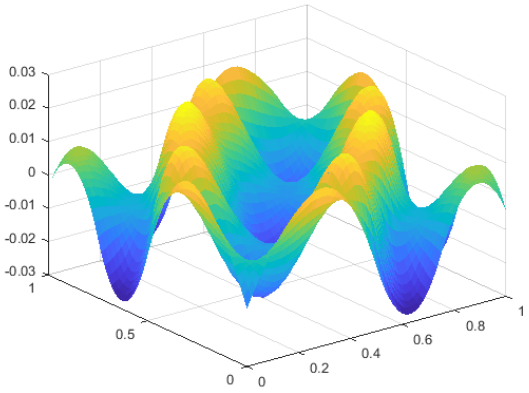
In the last section, we have seen that a basis of quadratic Powell-Sabin splines leads to third-order spatial convergence for function reconstruction. It is also possible to use cubic PS-spline instead, for even higher-order spatial convergence. Grošelj and Speleers [18] have described a construction of these cubic Powell-Sabin



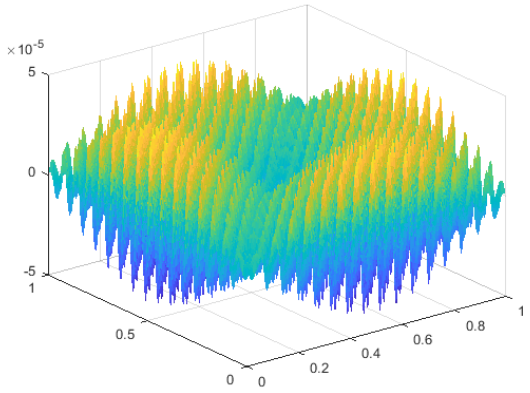
(a) Function $f(x, y) = \sin(\pi x)\sin(\pi y)$.



(b) Projection $\hat{f}$ with $n = 3$, $h = 1/2$.



(c) Error $E = \hat{f} - f$ with $n = 3$, $h = 1/2$.



(d) Error $E = \hat{f} - f$ with $n = 17$, $h = 1/16$.

Figure 3.26: The double sin function, and its reconstruction for $n = 3$ are shown in the upper two figures. For more precise reconstructions, the difference is not visible from the functions. Instead the error $\hat{f} - f$ is shown in the lower two figures for $n = 3$ and $n = 17$.

B-splines, which is shortly summarised in Appendix A.3. These basis functions are piece-wise third-order polynomials, with the properties of non-negativity, locality, partition of unity and C^1-continuity. However, the construction process and analysis is even more elaborate than for quadratic PS-splines, and therefore. Furthermore, the result in the next chapter will show that spatial convergence will no longer be the limiting factor for total error in MPM, and therefore, a cubic-spline basis is not expected to further reduce the error. Therefore, in this thesis only quadratic PS-spline basis functions are considered.

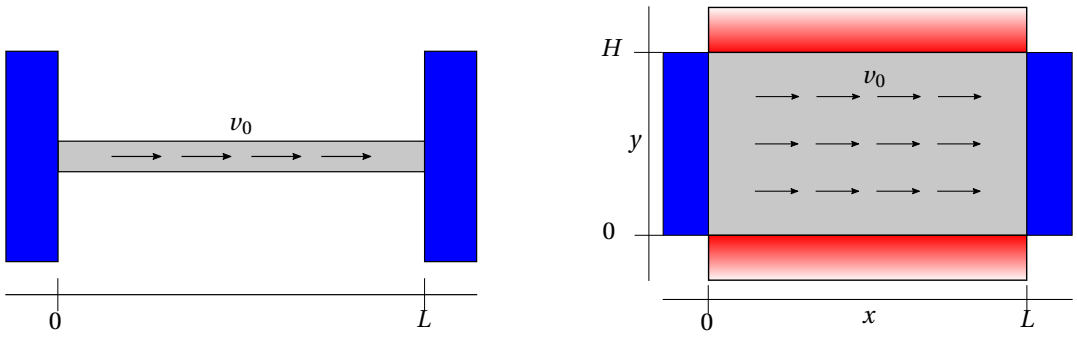
4

RESULTS

In this chapter, PS-spline MPM will be compared to standard Lagrange basis MPM on a triangular grid. We expect that MPM with quadratic B-spline basis functions does not show any grid crossing errors. Also, the quadratic PS-splines are expected to show third-order spatial convergence, whereas the piece-wise linear Lagrange basis functions should result in only second-order convergence. This difference in convergence should occur in the projection of the material properties to the grid. In Section 3.6, it was shown that the PS-spline basis functions indeed show third-order convergence for general function reconstruction. In this section, we will investigate if this also holds when applied in MPM. In order to compare PS-spline MPM with standard Lagrange MPM two benchmark problems are considered: a vibrating bar and a soil column under self weight. Both benchmarks are considered for both small and large deformations.

4.1. BENCHMARK TESTING: VIBRATING BAR, SMALL DEFORMATIONS

In this section, a vibrating bar with both ends fixed is considered as benchmark to compare PS-spline MPM with classical Lagrange MPM in terms of spatial convergence and general accuracy. The vibrating bar problem can be described as a one-dimensional problem, in which the particles in the bar are only allowed to move in the x-direction. The particles start with an initial x-velocity and as a result, the bar will be stretched and compressed in the domain. Figure 4.1 shows a schematic illustration of the situation. Furthermore, the figure shows a way to represent this one-dimensional benchmark as a two-dimensional problem. By adding a second dimension, the problem may be solved with MPM with the quadratic 2D Powell-Sabin splines on a triangular grid.



(a) 1D (b) 2D

Figure 4.1: The benchmark problem of a vibrating bar, described as a one-dimensional problem on the left, and as a two-dimensional problem on the right. On the left and right of the bars there are homogeneous Dirichlet conditions for both x- and y-displacement. The top and bottom boundary conditions are homogeneous Dirichlet for only the y-displacement. An initial velocity field is imposed over the bar with the maximum speed v_0 in the middle.

The one-dimensional problem has homogeneous Dirichlet boundary conditions at $x = 0$ and at $x = L$. For the two-dimensional problem, homogeneous Dirichlet boundary conditions for the x- and y-velocity are also imposed at $x = 0$ and $x = L$. At the upper and lower edge, where $y = 0$ and $y = H$ respectively, only

a homogeneous Dirichlet boundary condition is imposed for the y-velocity. This is to prevent the bar from deforming in the y-direction, which does not happen in the one-dimensional problem. For the x-velocity at $y = 0$ and $y = H$, no conditions are imposed, as the bar must be free to move to the left and right within the domain. With these conditions, it is expected that the two-dimensional solution will be only dependent on x, and be stationary in the y-direction. The solution should be the same as for the one-dimensional problem.

This problem has also been described and researched extensively in 1D by Tielen [19]. The same parameters will be used in this thesis, and the 2D solution will be compared to the 1D solution of Tielen. The specific parameters used for the problems are shown in Table 4.1. The parameters are chosen such that the deformations in the bar will be small, i.e. the normal strain in the x-direction $\epsilon_{11} \leq 1\%$. Deformations in the y-direction do not occur in the analytical solution, as there will only be an initial velocity in the x-direction. Also, as the Poisson ratio is zero, deformation in the x-direction should not lead to deformation in the y-direction. Finally, the width of the bar is chosen sufficiently large such that the elements will have about the same size in both dimensions, in order to prevent sliver elements.

Table 4.1: The parameters for a bar undergoing small deformation, modelled as a $2D$ object. When modelled as a $1D$ object, the width is dropped. Retrieved from [19].

Quantity	Symbol	Value	Unit
Density	ρ	1	$[\mathrm{kg/m^3}]$
Young's modulus	E	100	[Pa]
Poisson Ratio	ν	0	[−]
Length	L	25	[m]
Width	H	2	[m]
Velocity	v_0	0.1	[m/s]

For small deformations, the differential equation derived from the conservation of momentum can be reduced to the wave equation [20],

$$\frac{\partial^2 u_x}{\partial t^2} = \frac{E}{\rho}\frac{\partial^2 u_x}{\partial x^2}. \tag{4.1}$$

The boundary conditions for the displacement $\boldsymbol{u} = (u_x, u_y)$ are described as

$$\begin{aligned}
u_x(0,y,t) = 0, \quad & u_y(0,y,t) = 0, && \text{(left end fixed)},\\
u_x(L,y,t) = 0, \quad & u_y(L,y,t) = 0, && \text{(right end fixed)},\\
& u_y(x,0,t) = 0, && \text{(bottom fixed in } y\text{-direction, free in } x\text{-direction)},\\
& u_y(x,H,t) = 0, && \text{(top fixed in } y\text{-direction, free in } x\text{-direction)}.
\end{aligned}$$

The initial conditions are given by

$$\begin{aligned}
u_x(x,y,0) &= 0,\\
u_y(x,y,0) &= 0,\\
\frac{\partial u_x}{\partial t}(x,y,0) &= v_0 \sin\left(\tfrac{\pi}{L}x\right),\\
\frac{\partial u_y}{\partial t}(x,y,0) &= 0.
\end{aligned}$$

The solution for the vibrating bar problem under these conditions is given by

$$u_x(x,y,t) = \frac{v_0}{\omega}\sin(\omega t)\sin\left(\tfrac{\pi}{L}x\right), \tag{4.2}$$

with ω denoting the wave speed, defined as

$$\omega = \frac{\pi\sqrt{\frac{E}{\rho}}}{L}. \tag{4.3}$$

Having defined the problem and the analytic solution, the numerical solution using PS-spline MPM can be compared to classical MPM solutions and to the analytic one. For PS-spline MPM, the domain is divided into

rows and columns of rectangles or blocks, and each block is divided into four triangles. For the first simulation, the grid is divided in 12×1 blocks, see Figure 4.2, yielding 38 vertices. The particles are also initialised in a rectangular fashion: 6 rows and 96 columns (8 columns per rectangle), see Figure 4.2 again for the distribution of the particles in the grid. In general, the grid can be fully described by the number of rows and columns of rectangles, and the number of rows and columns of particles in each rectangle.

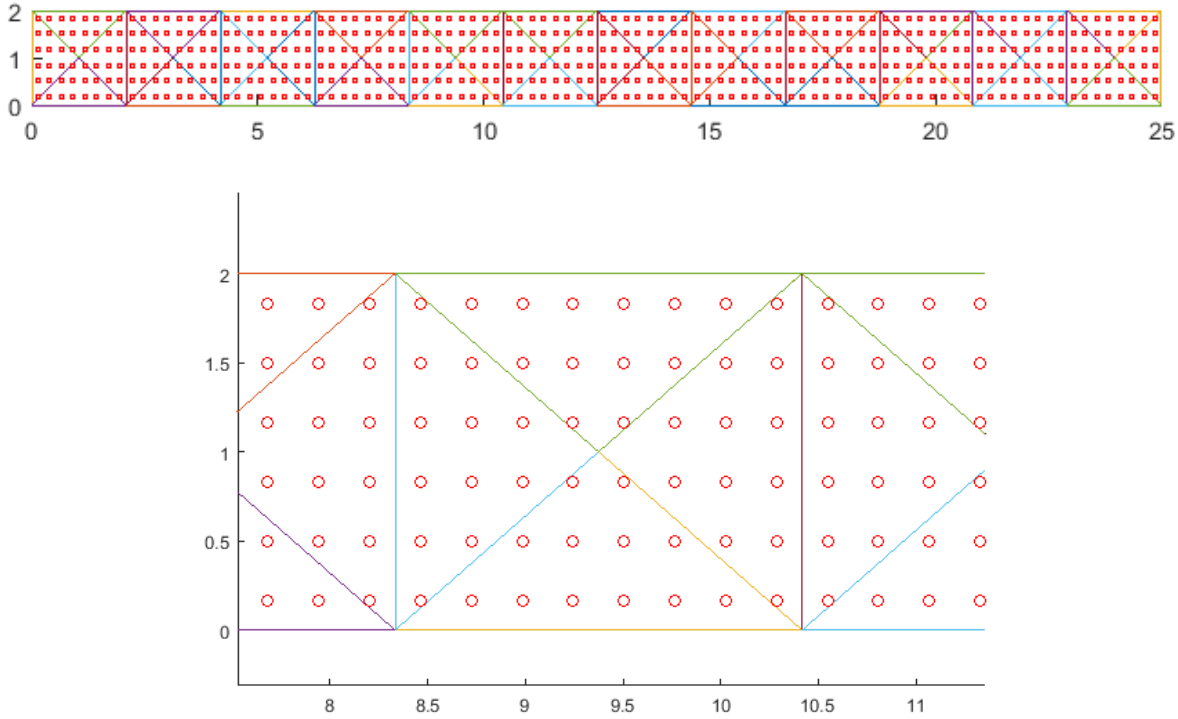


Figure 4.2: The grid for classical Lagrange basis MPM modelling of a vibrating bar. The triangulation of the domain is shown above, four elements in a rectangle are shown below, along with the initial particle positions, of which there are 12 in each element.

Using this grid, a solution can be determined using B-spline MPM and with classical Lagrange MPM. First, consider classical Lagrange MPM. The numerical solution for the velocity of a particle in the middle of the bar is shown in Figure 4.3. In the figures on the left, the results are shown for a the vibrating bar, with an initial velocity of 0.1m/s in the center, whereas on the right, the initial velocity was only 0.025 m/s. Although in the displacement the difference is hardly visible, a difference in the velocity is clear: the case with the higher initial velocity shows oscillations. The reason for this is the grid crossing error. In the upper left figure, the maximum displacement of the considered particle is about 0.08 m, which is enough for several particle to cross from one element to a neighbouring one, as can be seen from Figure 4.2. Lowering the initial velocity also lowers the maximum displacement, and for an initial velocity of 0.025 m/s, grid crossing no longer happens. Despite the fact that grid crossing errors appear in the case with $v_0 = 0.1$ m/s, the movement of the vibrating bar is captured quite well, and the oscillations in the velocity are mitigated for the displacement through the integration over time.

Next consider the same grid, but with PS-spline MPM. PS-spline MPM has three basis function per vertex, whereas classical Lagrange MPM has only one basis function per vertex. Therefore, the same grid yields three times as many basis functions with PS-splines compared to a Lagrange basis, and therefore these two cases cannot be compared fairly on the same grid. For a better comparison, a coarser grid can be used for PS-spline MPM to get a comparable amount of basis functions. The discretisation of Figure 4.2 has 38 vertices, which corresponds to 38 piece-wise linear Lagrange basis functions. To get a comparable number of PS-spline basis functions, a grid of 1×4 block could be constructed, as shown in Figure 4.4, resulting in 42 PS-basis functions (14 vertices, 3 PS-basis function per vertex). Note that the elements for this configuration are more sliver like, which may give problems if the y-velocity gets too big. For the vibrating bar, however, only a velocity in the x-direction is expected, and the displacement in the y-direction should be insignificant due to the formulation of the problem.

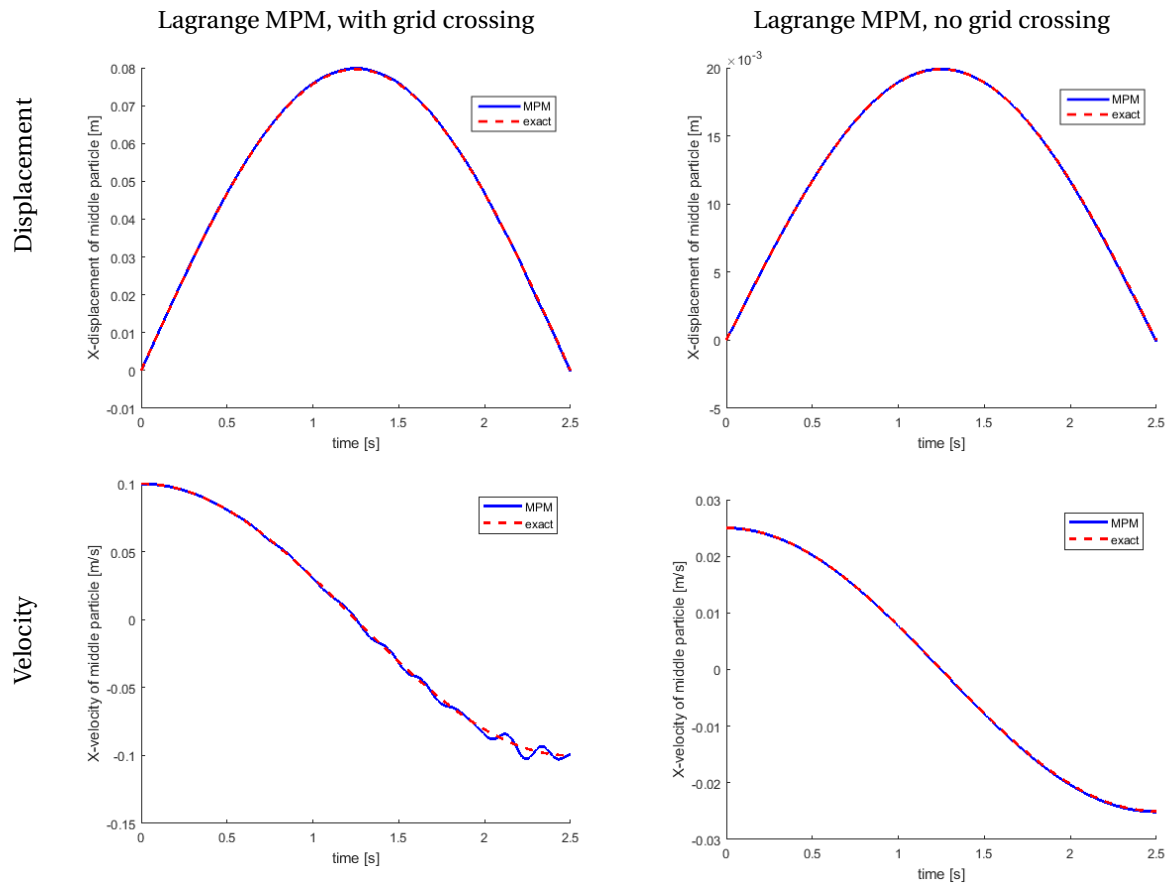


Figure 4.3: The solution for the movement of a 25 m by 2 m vibrating bar with small deformations, using classical Lagrange MPM. 38 basis functions were used, resulting in a typical element length of 2.1 m as shown in Figure 4.2. For the left case, the initial maximum velocity was 0.1 m/s, which resulted in grid crossing. On the right, the initial maximum velocity was 0.025 m/s, for which grid crossing did not occur.

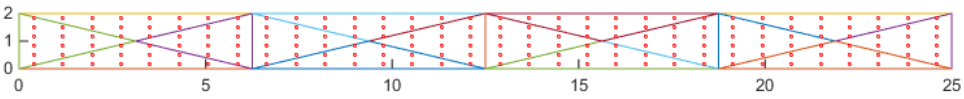


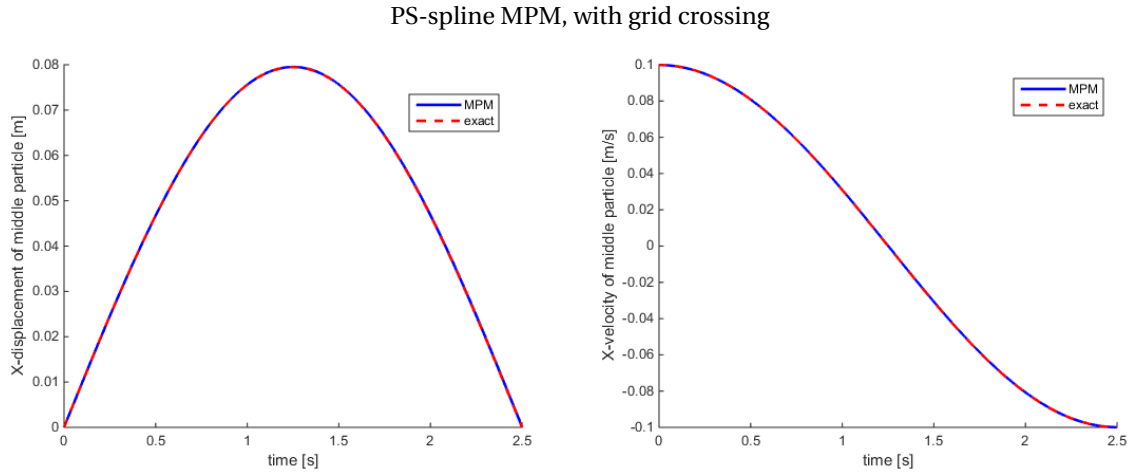
Figure 4.4: The grid and initial particle positions for 2D PS-spline MPM modelling of a vibrating bar.

Using the grid of Figure 4.4, the solution for the displacement and the velocity over time are calculated and shown in Figure 4.5. The figure shows that both the displacement and the velocity are reconstructed in accordance with the analytical solution; the error is hardly visible. For this test case, grid crossing did occur, but this does not seem to influence the solution at all, no oscillations occur in the velocity as in the bottom right in Figure 4.3. This is in accordance with the theory on grid crossing in Section 2.7: the C^1-continuity of the PS-basis functions eliminates the grid crossing error.

Next, we will investigate the spatial convergence for both classical Lagrange MPM and PS-spline MPM. For this, the L_2-error will be considered at time $t = 0.02$, and be investigated as a function of the typical element length. The L_2-error of the displacement E_u is calculated by comparing the exact and numerical solutions in the particles, and applying quadrature based integration:

$$E_u = \sqrt{\sum_{p=1}^{n_p} V_p \cdot (\hat{u}_p - u(\boldsymbol{x}_p))^2}. \tag{4.4}$$

Here, V_p is the volume of particle p, $u(\boldsymbol{x}_p)$ is the exact x-displacement of a particle with initial position $\boldsymbol{x}_p$ and $\hat{u}_p$ is the x-displacement in particle p determined in the simulation. Note that the error in the numerical solution is caused by many factors, most importantly the time stepping error, the spatial reconstruction error,



(a) x-displacement with PS-spline MPM.

(b) x-velocity with PS-spline MPM.

Figure 4.5: The solution for the movement of a 25m by 2m vibrating bar with small deformations, using PS-spline MPM on the grid shown in Figure 4.2; 42 basis functions were used.

the quadrature integration error and the grid crossing error. The time $t = 0.02$ is chosen very small, such that no grid crossing happens before this time. By taking small time steps and many integration points, it can also be ensured that the time integration error and quadrature error are small, and thus the error is mainly dependent on the number of basis functions used for the spatial reconstruction. For these simulations, the time step size will be chosen as $\Delta t = 10^{-5}$, the number of particles per element is set to 56 (16×14 respectively in x- and y-direction per block). The discretisation of the grid is similar to Figure 4.2 and Figure 4.4: the grid is divided into $n \times 1$ blocks, with n the number of blocks in the x-direction. Then each rectangle is divided into four triangles by adding a vertex in the middle to ensure a symmetric grid. The number of columns of blocks in the x-direction is varied and changes the typical element length. The results for the L_2-error using Equation 4.4 are shown for Lagrange MPM in Table 4.2 and in Table 4.3 for PS-spline MPM respectively. Both results are also plotted in Figure 4.6.

Table 4.2: The L_2-error of a vibrating bar with small deformations at $t = 0.02s$ with Lagrange MPM.

n_x	h_x	$E_u(h)$	$^2\log(E_u(h)/E_u(2h))$
5	$25 \cdot 1/4$	1.80956e-04	-
9	$25 \cdot 1/8$	4.44076e-05	2.03
17	$25 \cdot 1/16$	1.07531e-05	2.05
33	$25 \cdot 1/32$	2.68352e-06	2.00

Table 4.3: The L_2-error of a vibrating bar with small deformations at $t = 0.02s$ with PS-spline MPM.

particles per element	n_x	h_x	$E_u(h)$	$^2\log(E_u(h)/E_u(2h))$
56	2	$25 \cdot 1/1$	3.70985e-04	-
56	3	$25 \cdot 1/2$	3.13639e-05	3.56
56	5	$25 \cdot 1/4$	3.56528e-06	3.14
56	9	$25 \cdot 1/8$	4.90397e-07	2.86
56	17	$25 \cdot 1/16$	1.76596e-07	1.47
90	17	$25 \cdot 1/16$	1.14682e-07	-

The results for the L_2-error show that Lagrange MPM converges spatially with second order, as expected. For PS-spline MPM, third-order convergence can be observed in part of the graph in Figure 4.6, but as the error becomes smaller, finally the third-order convergence is lost. A possible explanation for this is that the spatial reconstruction error is of the same magnitude as other errors, which will then also contribute to the total L_2-error. It is quite possible that despite the many integration points, the spatial reconstruction error is so small that the quadrature integration error may be significant. To test this, two almost identical test cases

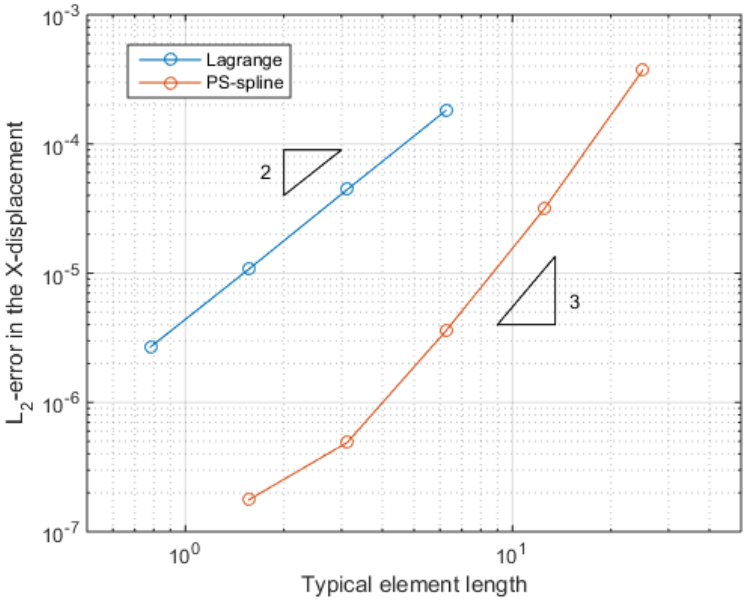


Figure 4.6: The L_2-error of the X-displacement in the vibrating bar with small deformations for simulations with Lagrange MPM and PS-spline MPM.

have been run, which are depicted in the lower two rows of Table 4.3. In both cases, the typical element length h_x in x-direction is equal and very small. The same grid was used, only the number of particles per element was increased in the second case. The error in the case with more particles is significantly lower than in the first case, which shows that the integration error is of importance for this typical element length and the error on display is no longer only dependent of the spatial reconstruction.

Another observation is that far less elements are required for a comparable L_2-error. For the case with typical element length $h = 25/33$ for Lagrange MPM, the error is of comparable magnitude as with $h = 25/5$ with Lagrange MPM. For these cases, the number of basis functions were 98 for Lagrange MPM, and 42 for PS-spline MPM. Therefore, we can conclude that for a fixed spatial reconstruction accuracy, far less basis functions are required using PS-spline MPM compared to Lagrange MPM.

4.1.1. VIBRATING BAR, SMALL DEFORMATIONS: SUMMARY

Summarising, for the test case of a vibrating bar with small deformations, Lagrange MPM shows grid crossing errors, but still manages to capture the movement quite well. PS-spline MPM does not show the grid crossing errors, and performs well even with very few elements. Lagrange MPM shows second-order spatial convergence, whereas PS-spline MPM shows third-order convergence when few elements are used. For more refined grids, the spatial convergence of PS-spline MPM is lost, as other errors become dominant, such as the quadrature integration error.

4.2. BENCHMARK TESTING: VIBRATING BAR, LARGE DEFORMATIONS

In this section, the vibrating bar of the last section is considered again, but this time the bar undergoes large deformations. As MPM is a method to simulate large deformations, it is important to check the performance of PS-spline MPM in such scenarios. When large deformations occur, the analytical solution in Equation 4.2 no longer holds; instead, solutions of Lagrange and PS-spline MPM will be compared to the numerical solution on a 1D simulation of the vibrating bar using the Updated Lagrangian Finite Element Method (ULFEM). This method is not a material point method, but a standard finite element method in which the elements move along with the material. In general, this method will not work for simulations in which grid distortion takes place, but due to the fact that the vibrating bar is only compressed and stretched in the x-direction, grid distortion will not happen and ULFEM will give reliable results. The reference solution for the vibrating bar will be simulated using a 1D ULFEM simulation, with many elements and small time steps, to ensure a very accurate solution. The ULFEM code was retrieved from Tielen [19].

For simulating a vibrating bar undergoing large deformations, we consider the same situation as in Figure 4.1, except that some quantities are different. In Table 4.4, the quantities for the vibrating bar with large deformations are shown. For the chosen parameters, the maximum normal strain in the x-direction is about $\varepsilon_{11} \approx 7\%$

Table 4.4: The parameters for a bar undergoing large deformation, modelled as a $2D$ object. When modelled as a $1D$ object, the width is dropped. Retrieved from [19].

Quantity	Symbol	Value	Unit
Density	ρ	25	$[\mathrm{kg/m^3}]$
Young's modulus	E	50	[Pa]
Poisson ratio	ν	0	[−]
Length	L	25	[m]
Width	H	2	[m]
Velocity	v_0	0.1	[m/s]

The vibrating bar undergoing large deformations has first been simulated using Lagrange MPM. Again, the grid was discretised similar to Figure 4.2. More specifically, the grid was divided into 16 blocks in the x-direction, and 1 in the y-direction, and each block was divided again in 4 triangles. The results are shown in Figure 4.7. On the left, 6 particles per element were used. Although the displacement still captures the general movement of the bar quite well, a slight error in the x-displacement is apparent. Especially in the velocity in the bottom left of Figure 4.7 the error becomes well visible; the oscillations are due to grid crossing errors.

In order to mitigate this problem, we can run the same simulation again with more particles. The grid crossing error causes trouble in the quadrature integration, as explained in Section 2.7, so adding more particles should mitigate this problem. If few particles are used, each particle contains a relatively large momentum. Therefore, when one particle crosses an element boundary, the element to which it went suddenly contains a much larger total momentum. If more particles are used, each particle will contain a much smaller portion of the total momentum, so the jump in total momentum in an element will become much smaller and therefore the grid crossing error becomes smaller. This is also visible from plots on the right in Figure 4.7, in which the same vibrating bar was simulated, but with 56 particles per element instead of 6. The error in the displacement is much smaller and the oscillations in the velocity are much smaller as well. However, the oscillations in the velocity are still significant; adding more particles does not solve the grid-crossing error, but only reduces it. In theory, doubling the number of particles per cell halves the momentum per particle. As the grid crossing error upon occurrence is proportional to the momentum of the particle crossing over, the grid crossing error is inversely proportional to the number of particles per elements. Due to the fact that there is only first-order convergence with respect to the particles per element, many particles must be used, which will lead to slow computations.

Next, the vibrating bar undergoing large deformation is simulated using PS-spline MPM. The grid was divided in 8 columns of blocks in x-direction and 1 row of blocks in the y-direction, and each block was divided in 4 triangles. This led to a total of 26 vertices, so $26 \cdot 3 = 78$ basis functions. The typical element length in x-direction was $h = 0.125\,\mathrm{m}$. The results are shown in the left side of Figure 4.8. The solution using MPM is compared to the ULFEM solution. For the displacement, there is hardly any visible difference. For the velocity

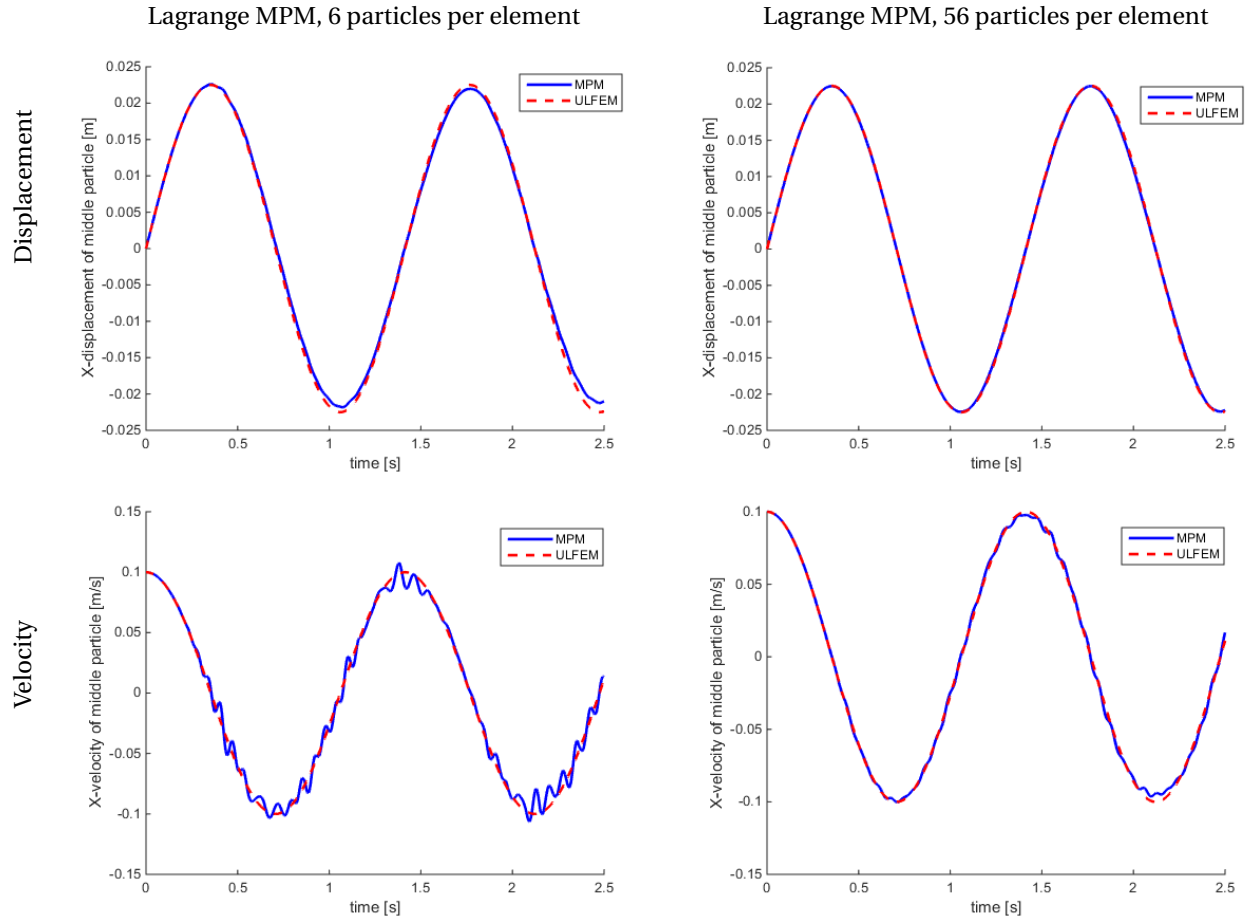


Figure 4.7: The solution for the movement of a 25 m by 2 m vibrating bar undergoing large deformations, simulated using classical Lagrange MPM. 50 basis functions were used, the typical element length was 1/16 m. The results on the left were determined with 6 particles, whereas on the right 56 particles were used.

however, a small oscillation can be seen, although its amplitude is very small. We investigate if refining the grid and adding more particles yields a more accurate result for the velocity. The grid spacing is halved for both the x- and y-direction, and the number of particles is raised from 20 to 90 particles per element. The results are shown on the right in Figure 4.8. Again, the x-displacement is not distinguishable from the ULFEM solution, but no significant improvement of the velocity error is visible. Upon inspection, the oscillations are still present, although with a slightly lower amplitude than in the less refined case on the left. Furthermore, the frequency of the oscillation has risen.

4.2.1. VIBRATING BAR, LARGE DEFORMATIONS: SUMMARY

Altogether, PS-spline MPM shows good result for simulating large deformations in a vibrating bar. Even when few elements are used, the displacement is well reconstructed, leaving only small vibrations in the velocity. Lagrange MPM suffers from grid crossing errors and is outperformed by PS-spline MPM. In case more particles are used for Lagrange MPM, the effect of grid crossing is reduced, but the errors are remain visible.

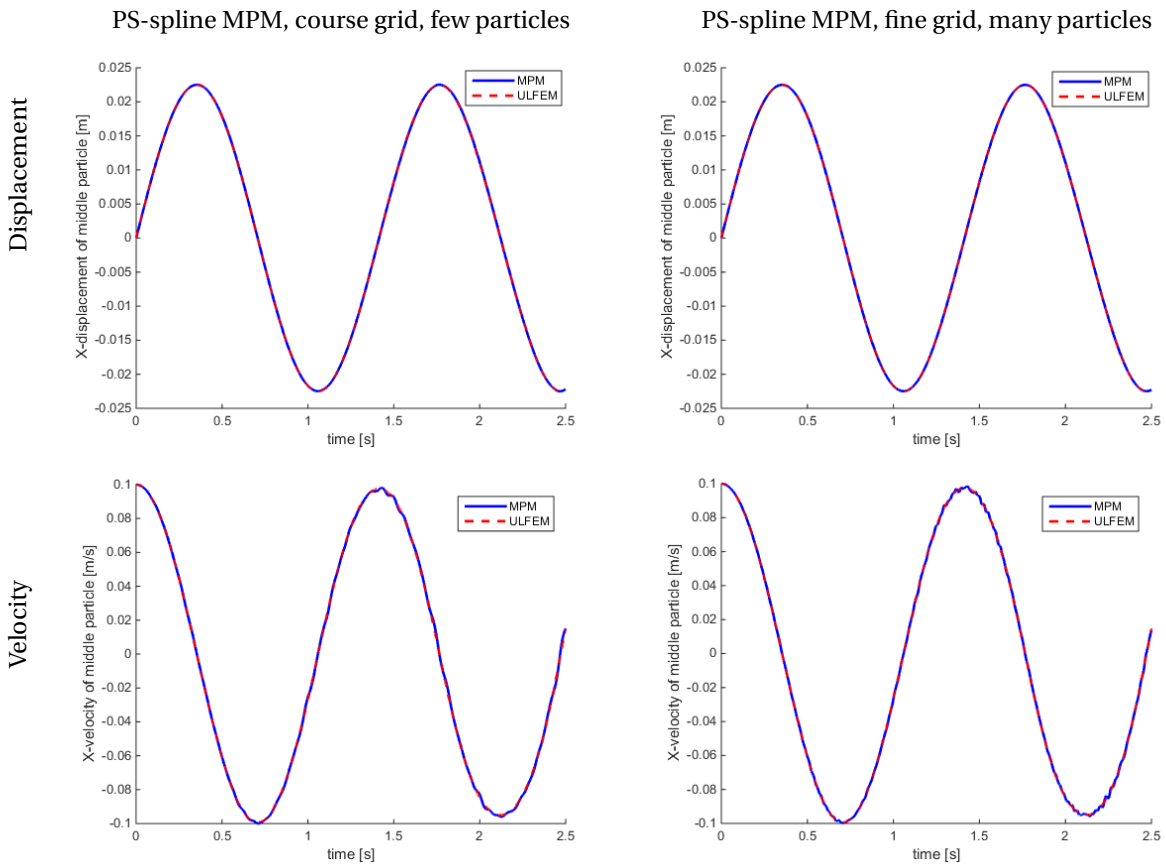
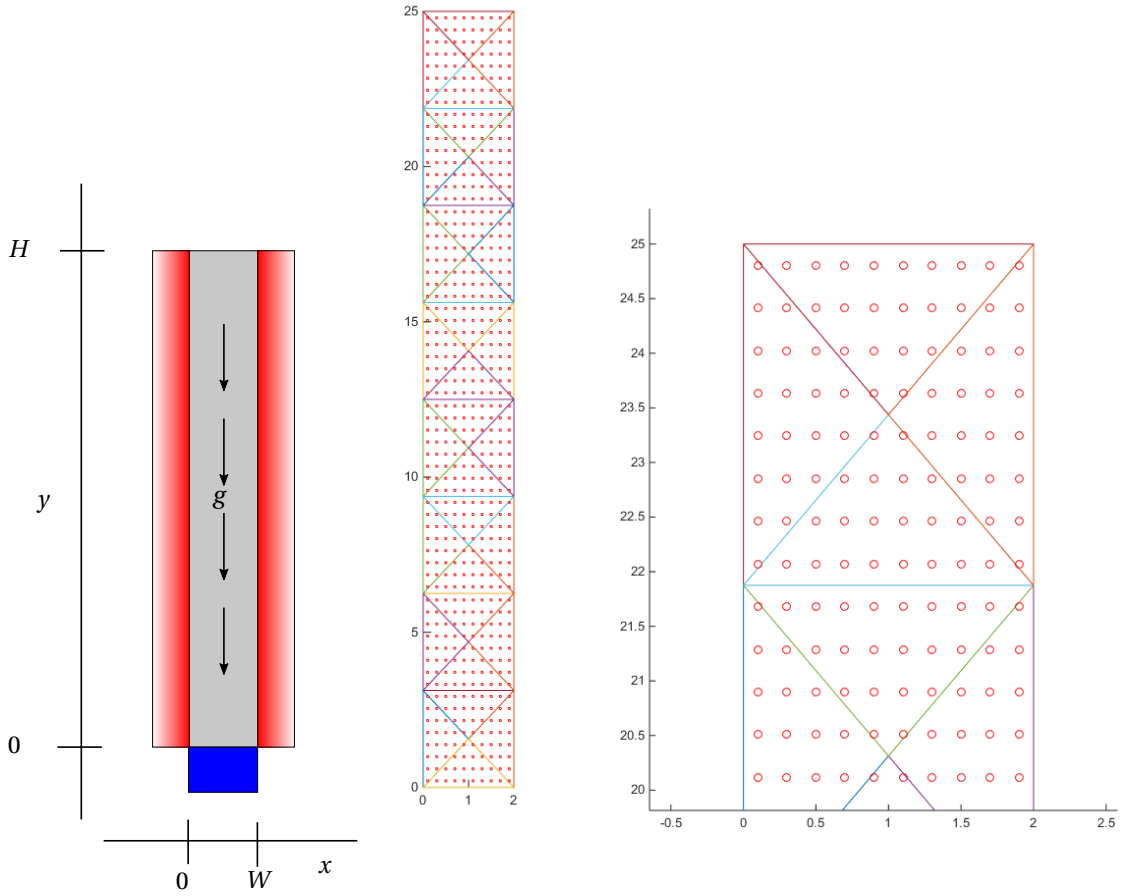


Figure 4.8: The solution for the movement of a 25m by 2m vibrating bar undergoing large deformations, simulated using PS-spline MPM. For the left cases, 78 basis functions were used, the typical element length was 1/8m and 20 particles per element were used. The results on the right were determined with 249 basis functions and 90 particles per element.

4.3. SOIL COLUMN UNDER SELF-WEIGHT, SMALL DEFORMATIONS

In the previous sections, the benchmark of a vibrating bar was considered. For this benchmark, the particles can undergo large deformation, but the geometry of the domain will not essentially change due to the Dirichlet boundary conditions at the left and right ends of the bar. In this benchmark, we will consider a soil column under self-weight, which is fixed at the bottom, but not at the top. Therefore, the soil column can be compressed, and the geometry of the column changes over time. First, we will investigate a soil column undergoing small deformation, in Section 4.4, the same benchmark is considered with large deformations. Figure 4.9a shows a schematic representation of the soil column benchmark.



(a) Geometry of a soil column with boundary conditions.

(b) 2D soil column discretisation.

(c) Soil column discretisation, zoomed in at the top.

Figure 4.9: The model of a soil column and the grid used in MPM. On the left, the soil column is modelled as a 2D object in which the boundary conditions are marked red and blue: at the bottom, x- and y-displacements are fixed, left and right only the x-displacement is fixed. In the right two figures, the domain is divided into blocks, and each block is divided into four elements. In each block, particles are distributed initially in rows and columns. On the right, the top of the column is depicted in more detail.

The soil column benchmark considered here undergoes deformation due to self-weight. The initial soil column has uniform pressure and density throughout the domain. Due to gravity the column will be compressed until the soil at the bottom is so dense that it is compressed no further and a shock wave will go from the bottom to the top. Then the column will start expanding again top first, and the shock wave mentioned before is reflected from the top. Note that the problem can also be described as a 1D problem in which the displacement of the soil is only a function of the height; this benchmark has been researched extensively with 1D B-spline MPM by Tielen et al. [5, 19]. In order to investigate the 2D PS-spline MPM, the same benchmark is considered, but with an extra dimension to be able to use Powell-Sabin splines on triangulations. For the 2D case, the expected result is the same as for the 1D case, and for small deformation, the analytical solution can

be derived from the 1D wave equation [20]:

$$\frac{\partial^2 u}{\partial t^2} = \frac{E}{\rho}\frac{\partial u}{\partial y} - g, \tag{4.5}$$

in which u is the y-displacement, t the time, E the Young's modulus and g the gravitational acceleration. The boundary conditions are described as

$$\begin{aligned} u_x(x,0,t) = 0, \quad u_y(0,y,t) &= 0, \quad \text{(bottom fixed)}, \\ u_y(0,y,t) &= 0, \quad \text{(left side fixed in } x\text{-direction, free in } y\text{-direction)}, \\ u_y(0,W,t) &= 0, \quad \text{(right side fixed in } x\text{-direction, free in } y\text{-direction)}, \end{aligned}$$

in which u_x and u_y are the displacement in the x- and y-direction respectively. For initial conditions, the system is at rest with initial velocity and displacement:

$$\begin{aligned} u_x(x,y,0) &= 0, \qquad u_y(x,y,0) = 0, \\ \frac{\partial u_x}{\partial t}(x,y,0) &= 0, \qquad \frac{\partial u_y}{\partial t}(x,y,0) = 0. \end{aligned}$$

The analytical solution for this problem is given by

$$u(y,t) = \frac{\rho g y^2}{2E} + \frac{\rho g H y}{E} + \sum_{n=1}^{\infty} u_n \cos\left(\frac{\sqrt{\frac{E}{\rho}}(2n-1)\pi t}{2H}\right)\sin\left(\frac{(2n-1)\pi y}{2H}\right), \tag{4.6}$$

with

$$u_n = \frac{16\rho g H^2}{(4n^2-4n+1)(2n-1)\pi^3 E}.$$

For the simulation of a soil column with small deformations, the parameters in Table 4.5 were used. Adopting these parameters yields a maximum stress of about 1%.

Table 4.5: The parameters for a soil column under self-weight for small deformations. The column is modelled as a $2D$ object with a height and width; when modelled as a $1D$ object, the width is dropped. Retrieved from [19].

Quantity	Symbol	Value	Unit
Density	ρ	1	$[\mathrm{kg/m^3}]$
Young's modulus	E	$5 \cdot 10^4$	[Pa]
Poisson Ratio	ν	0	[−]
Gravitational acceleration	g	-9.81	$[\mathrm{m/s^2}]$
Height	H	25	[m]
Width	W	2	[m]

The described soil column problem has been solved numerically with Lagrange MPM and with PS-spline MPM. For Lagrange MPM, the grid was divided into 2×16 blocks, which resulted in 83 basis functions. For the PS-spline MPM, the grid was divided into 1×8 blocks, which resulted in 78 basis functions. For both cases, 12 particles per element were used. The grid for the PS-spline MPM is shown in Figure 4.9. The grid for Lagrange MPM had a similar structure, but the grid was divided in more elements to get approximately the same number of basis functions. The grid was constructed to prevent sliver triangles as much as possible by taking the typical element length in x- and y-directions of similar size. This was also the reason for the choice of the width of the soil column.

The results for the Lagrange MPM and PS-spline MPM simulations are shown in Figure 4.10. On the left of the figure, the results for Lagrange MPM are shown and on the right, the results of PS-spline MPM. Both methods seem to simulate the displacement very well. Both methods develop a very small phase difference from the analytical solution over time, but this is due to the fact that the analytical solution is based on infinitesimal deformations, which is not entirely the case, and is therefore not entirely accurate. Simulations have also been run with extremely small deformation and in this case, the displacement is indistinguishable from the exact solution for both cases.

Furthermore, a large difference in the velocity results can be observed between Lagrange MPM and PS-spline MPM. The velocity results of PS-spline MPM are hardly distinguishable from the exact solution, whereas the results of Lagrange MPM oscillate very much. This result was also reported by Tielen [19] and is caused by grid crossing. When the Lagrange MPM simulation is run again with a lower gravity for smaller displacements, the oscillations disappear mostly. Again, PS-spline MPM does not have these problems at all.

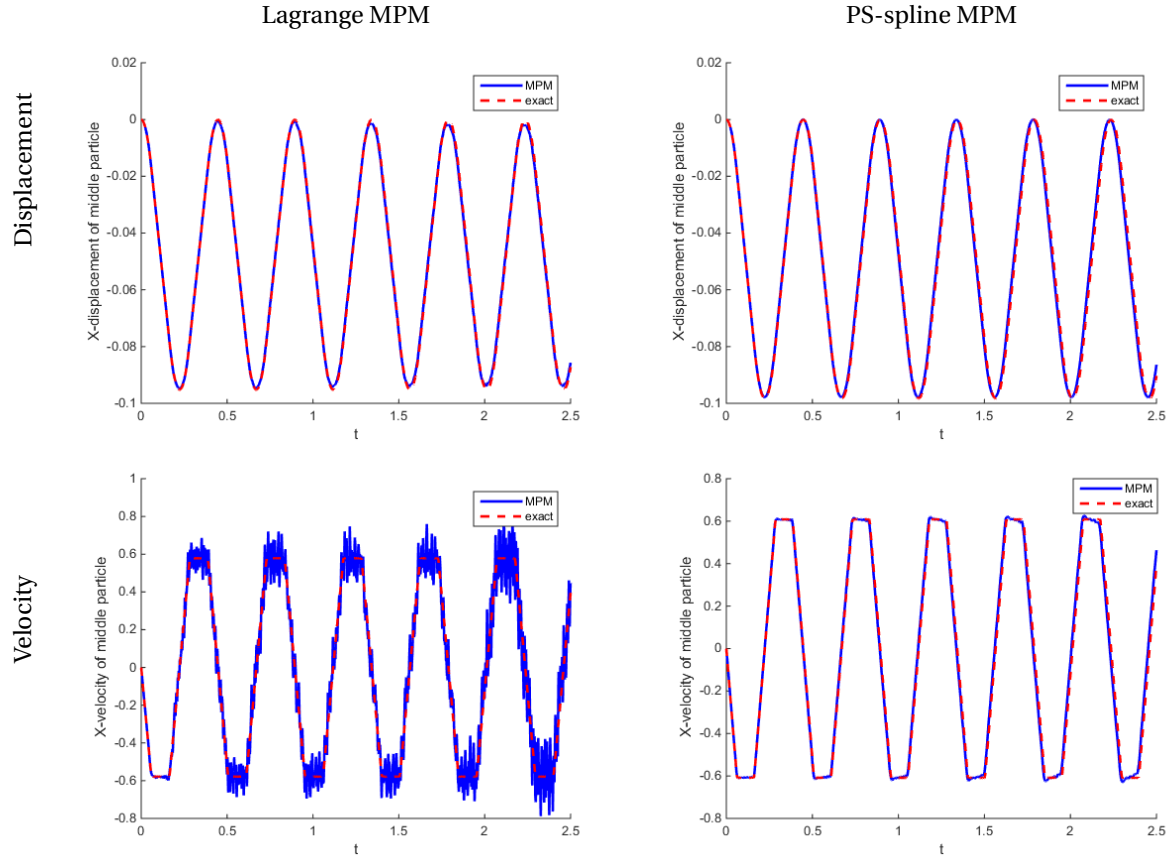
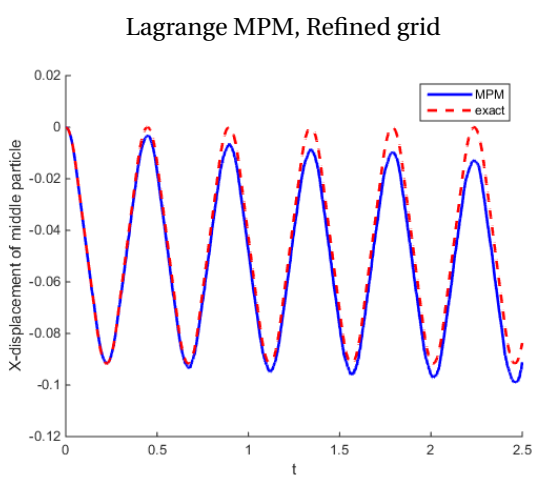


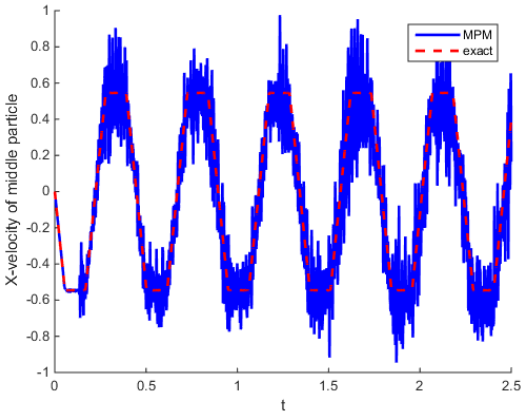
Figure 4.10: MPM simulation of a soil column undergoing small deformation. The left case was simulated with classical Lagrange MPM; the grid was divided into 2 columns and 16 rows of blocks, resulting in 83 basis functions. The right case was simulated with PS-spline MPM, and divided into 1 column and 8 row of blocks, resulting in 78 basis functions. For both cases, 12 particles per element were used.

Upon further refinement of the grid, a new phenomenon appears in the results for both Lagrange MPM and PS-spline MPM. When the grid for Lagrange MPM is refined from 2×16 blocks from the previous simulation to 2×24 blocks, the instabilities in the velocity becomes larger, as can be seen from Figure 4.11. This behaviour is expected, as grid crossing will occur more often on a refined grid than on a coarse grid. The difference now is that the instabilities are also visible in the displacement, in contrast to earlier results, the displacement is no longer in correspondence with the analytical solution. Similar results have also been reported by Tielen et al. for 1D, this is a large drawback of classical Lagrange MPM.

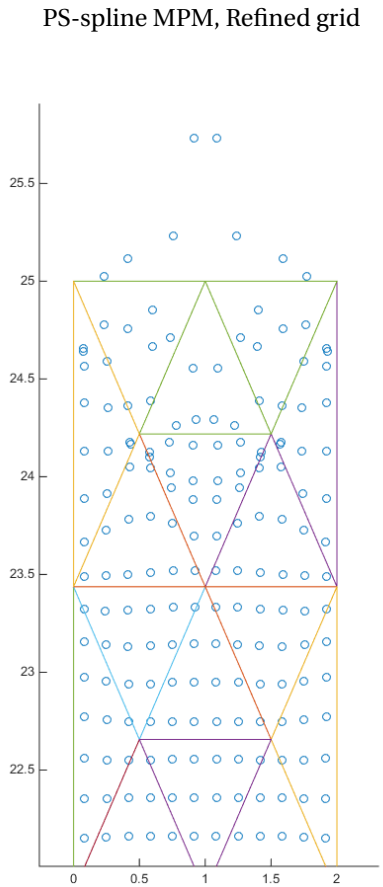
Although Lagrange MPM does poorly for a more refined grid, PS-spline MPM does a lot worse if no special integration technique is used. The simulations using PS-spline MPM show large instabilities when the grid is refined. A simulation with PS-spline MPM was run with a grid refined into 2×16 blocks, compared to the 1×8 blocks used in Figure 4.10. The result was that particles went out of bounds at the top side of the domain shortly after the simulation was started, as can be seen in Figure 4.11c. Upon investigation in the code, it turns out that problems occur in the assembly of the mass matrix when elements become partially empty. This is currently one of the largest issues we are facing with our implementation of PS-spline MPM and several methods to mitigate this problem are proposed in Chapter 5, such as matrix lumping, partial lumping, moving grids and interpolation. For now, the conclusions on PS-spline MPM will be based on the results on coarse grids, at shown in Figure 4.10.



(a) Displacement



(b) Velocity



(c) Particle position with PS-spline MPM at t=0.031.

Figure 4.11: Lagrange MPM simulation of a soil column undergoing small deformation. The grid was divided in 2 columns and 24 rows, resulting in a typical element in x-direction of 0.96. For the right figure, PS-spline MPM was used with 2 columns and 16 rows. For both cases, 12 particles per element were used.

4.3.1. SOIL COLUMN, SMALL DEFORMATIONS: SUMMARY

From the simulation of a soil column under self-weight undergoing small deformations, we have observed that Lagrange MPM and PS-spline MPM both yield fine results when no grid-crossing occurs. When grid crossing does occur on a small scale, Lagrange MPM can still produce good results for the displacement, as the oscillations in the velocity may be mitigated due to integration over time. When grid crossing occurs on a large scale, the solution for the displacement can show entirely wrong results. PS-spline MPM does not face the problem of grid crossing error, and shows very good results for both the displacement and the velocity on coarse grids. However, when the grid is refined, PS-spline MPM faces new problems: the elements at the top of the grid become partially empty, as the soil column is compressed, and this yield problems in the assembly of the mass matrix, causing instabilities. This problem will be further addressed in Chapter 5.

4.4. SOIL COLUMN UNDER SELF-WEIGHT, LARGE DEFORMATIONS

In this section, we study the soil column under self-weight again, but this time the parameters are chosen such that large deformations occur. This problem is closest to applications in geomechanical engineering, in which large soil deformations problems appear very often. The soil column will be compressed from the top, and its height will be lowered, comparable to the soil column with small deformations. This time, however, the deformation are significantly larger and at the largest deformation, about 10% of the original domain will be empty. In the last section we have also observed that PS-spline MPM shows problems in case elements are partially empty, and this problem will become only larger when large deformations appear. In order for PS-spline MPM to work properly, the elements will be chosen very large, such that the empty fraction of the top most elements will be smaller. More structural fixes are considered in Chapter 5, but in this section, PS-spline MPM is first considered without any further adaptations.

For simulating the soil column with large deformation, the same setup was used as in the previous section. Only the height, density and Young's modulus were changed; the choices for the parameters are shown in Table 4.6. Adopting these parameters leads to a maximum strain of 18%. As large deformations occur in the benchmark, the analytical solution presented in the last section for a soil column will no longer hold. Instead, an ULFEM solution will be used as reference, as was also done in the case of a vibrating bar with large deformations in Section 4.2.

Table 4.6: The parameters for a soil column under self-weight for large deformations. The column is modelled as a $2D$ object with a height and width; when modelled as a $1D$ object, the width is dropped. Retrieved from [19].

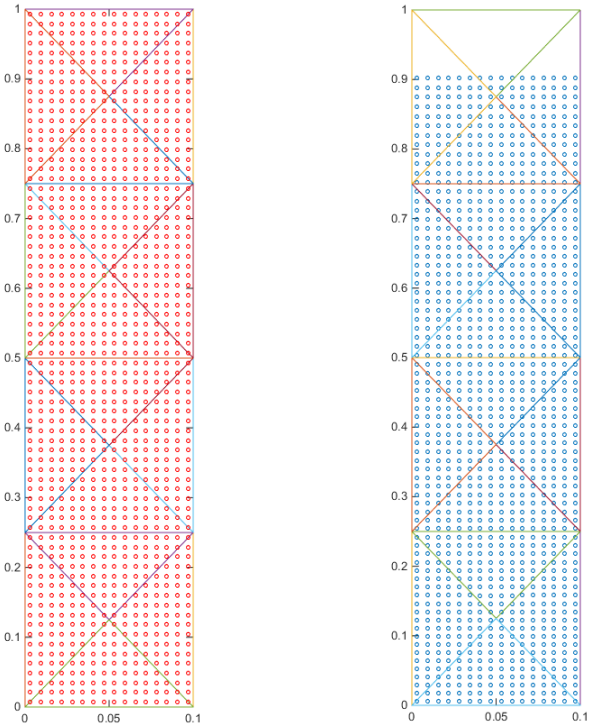
Quantity	Symbol	Value	Unit
Density	ρ	$1 \cdot 10^3$	$[\mathrm{kg/m^3}]$
Young's modulus	E	$1 \cdot 10^5$	[Pa]
Poisson Ratio	ν	0	[−]
Gravitational acceleration	g	-9.81	$[\mathrm{m/s^2}]$
Height	H	1	[m]
Width	W	0.1	[m]

For the simulations, a grid consisting of 1×4 blocks was used, with 16×18 particles in each block. This results in a total of 14 Lagrange basis functions or 42 PS-spline basis functions, with 72 particles per element. The grid is shown in Figure 4.12. It can be seen that refinement in more than 4 blocks would lead to an empty element at the top of the grid. In these cases, both Lagrange MPM and PS-spline MPM no longer converge to a solution. Also, if less particles are chosen with this specific grid, PS-spline MPM will also diverge, most probably because the integration error in the top most particle becomes too large and causes instabilities as well. All together, partially empty elements lead to stability problems for PS-spline MPM. Chapter 5 considers possible ways of improving the stability and will look into partially empty elements more thoroughly.

First, we will consider the results of both Lagrange MPM and PS-spline MPM on this coarse grid. Figure 4.13 shows the results for Lagrange MPM on the left and PS-spline MPM on the right. Note that Lagrange MPM has large instabilities in the velocity, but still captures the movement of the bar quite well. This is all due to the fact that a great number of particles was used, mitigating the grid crossing error. When the number of particles was halved, the errors would get significantly larger and the particle would not return to its initial position, but deform permanently, as was also observed in Figure 4.11 of the last section. On the left side of Figure 4.13 the results for PS-spline MPM are shown. For the shown grid, the computation was stable, as can be seen from the results. The displacement is captured very well, there is no visible difference between the solutions of PS-spline MPM and ULFEM. The PS-spline MPM solution for the velocity also follows the ULFEM solution quite well, although some oscillations can be observed. Nonetheless, with only a limited number of basis functions, the reconstructed solution is promising for a higher-order material point method.

4.4.1. SOIL COLUMN, LARGE DEFORMATIONS: SUMMARY

For large deformations, the PS-spline MPM will show instabilities when partially empty elements appear. However, when a coarse grid is chosen, such that all elements are occupied with enough elements at all times, the results are very promising, even when few basis functions are used, and show better solutions than Lagrange



(a) The grid with initial particle positions.

(b) The grid with particle positions at largest deformation.

Figure 4.12: The grid used to simulate large deformations in a soil column under self-weight. There are initially 16×18 particles per block, so 72 particles per elements initially.

MPM, which shows large oscillations in the velocity. The next step for PS-spline MPM is to improve the robustness of the scheme, in order to cope with partially empty elements. In the next chapter, this issue will be addressed, although a full solution has not been found yet for the stability issues.

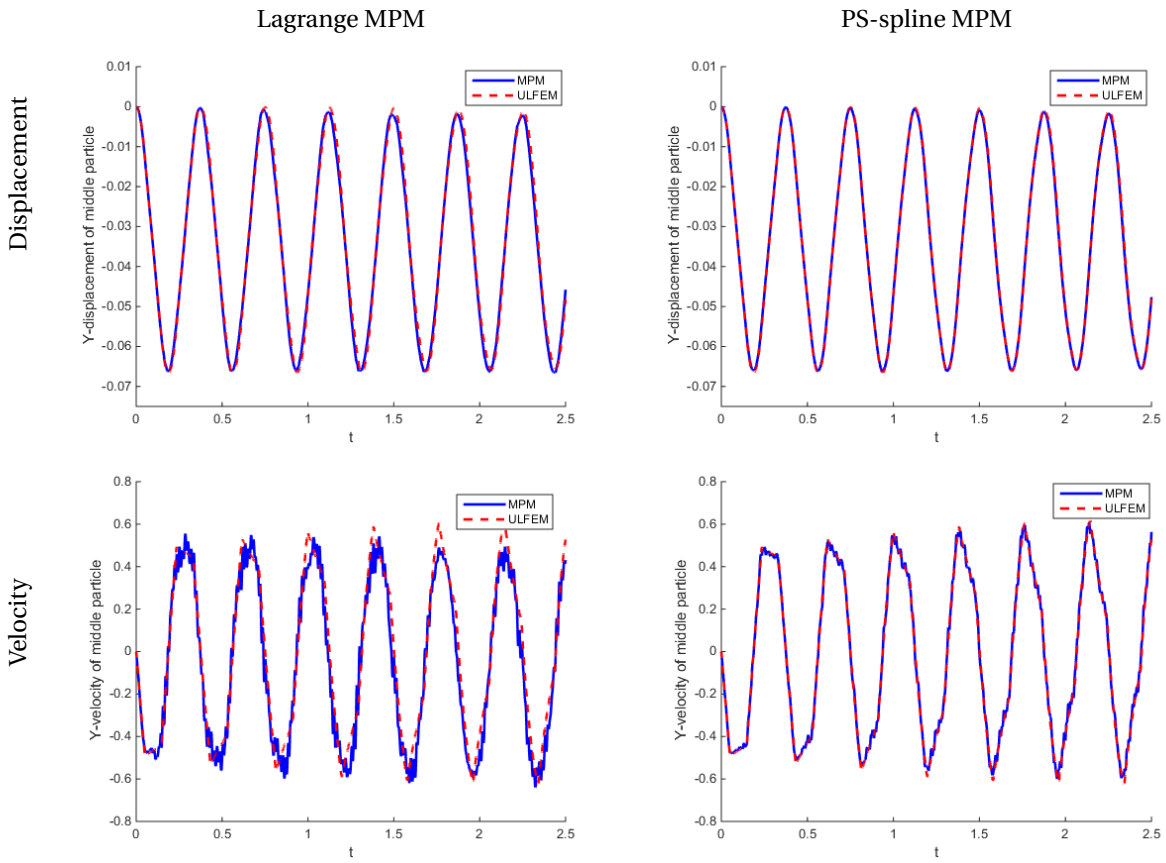


Figure 4.13: MPM simulation of a soil column undergoing large deformation. The grid was divided in 1 columns and 4 rows of blocks, and 72 particles per element. The left case was simulated with classical Lagrange MPM and the right case with PS-spline MPM. As the same grid was used for both Lagrange MPM and PS-spline MPM, the latter case had three times as many basis functions: 14 basis functions for Lagrange MPM and 42 for PS-spline MPM.

5

OPEN ISSUE: ALMOST EMPTY ELEMENTS

In the last chapter, we have seen that PS-spline MPM has shown accurate results and eliminates grid-crossing errors, but also showed stability issues when elements become partially empty, which will happen frequently in geomechanical problems usually solved with MPM. In this chapter, we will first explain why the instabilities appear, and second, we have proposed and investigated several methods to fix this issue. The methods will prove to yield better stability, although sometimes at the cost of the quality of the solution.

5.1. PROBLEM: INSTABILITIES

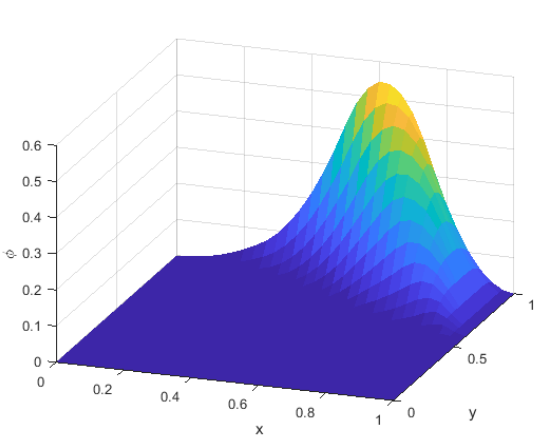
The stability problem in PS-spline MPM is caused by partially empty elements, which we have also seen in the soil column problem and Figure 4.11c. Figure 5.1 illustrates the problem: a Powell-Sabin spline basis function is depicted along with the molecule on which it is defined (the grey area). Recall that each PS-spline basis function is associated with a vertex, and the basis function is locally defined on the elements directly around that vertex. The area consisting of the elements directly around the vertex is called the molecule. In Figure 5.1b, a contour plot is shown, and from this we can see that the basis function is defined on a molecule consisting of three elements, but at the lower edges of these elements, it is very close to zero. Recall from the construction of PS-splines with Bézier ordinates in Figure 3.22, that the value at the edge of a molecule is only determined by the Bézier ordinate at the center point at the vertex. If this Bézier ordinate is either zero or very small, the value at the edge of the domain will also be zero or very small. Also, at the boundary of the domain, the basis function will go to zero quadratically, which also causes very small values at the molecule edges. A particle at the boundary of the molecule is indicated with a red dot in the contour plot, and the value of the basis function in this particle is indeed very small; its value is around 10^{-16}, the same order as the machine precision.

Next consider the situation that the molecule of the shown basis function is almost empty, only a couple of particles are at the lower edges of its molecule, with positions similar to the red dot. First of all, note that in the MPM algorithms, the considered basis function is still considered active, as its molecule is not empty. For the construction of the mass matrix M, recall that each element M_{ij} is calculated as in Equation 2.19, which is repeated here:

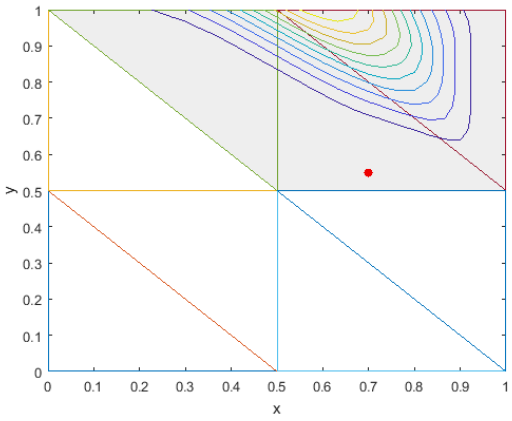
$$M_{ij} = \int_\Omega \phi_i \rho \phi_j \, \mathrm{d}\Omega \approx \sum_{p=1}^{n_p} V(\boldsymbol{x}_p)\rho(\boldsymbol{x}_p)\phi_i(\boldsymbol{x}_p)\phi_j(\boldsymbol{x}_p).$$

However, the value of the considered basis function ϕ_i is close to zero in all the material points $\boldsymbol{x_p}$. Therefore, the mass matrix entries on row i are extremely small as well, in particular the diagonal element M_{ii}, as the small values $\phi_i(\boldsymbol{x_p})$ are squared here as well. These small entries M_{ij} cause the matrix to have a very small condition number, typically smaller than 10^{-16}, but also values of 10^{-200} have been observed just before particles left the domain. The fact that the mass matrix is ill-conditioned can cause severe instabilities when $M\boldsymbol{a} = F$ is solved consistently for the acceleration $\boldsymbol{a}$. Also note from Figure 5.1 that as particles are moving out of the element, at first this is not a problem, as the basis function still assumes sufficiently large values in the particles that are still halfway the element. However, as particles move further to the edges, the mass matrix entries will be smaller and smaller, up to the point that the matrix is very ill-conditioned.

The problem of small matrix elements and ill-conditioned matrices is much less of a problem for classical Lagrange MPM, because piece-wise linear basis functions are used: due to the fact that the basis functions



(a) A Powell-Sabin spline basis function.



(b) A contour plot of the PS-spline basis function and a particle (red) in the molecule of the basis functions at the edge of the element.

Figure 5.1: A PS-spline basis function at the boundary of a square domain, and a contour plot of this basis function on the triangular grid it was constructed on. The basis function belongs to the upper middle vertex and the corresponding molecule on which the basis function is defined is marked grey. Finally, a particle at the lower boundary of the grey molecule is also indicated with a red dot.

are linear between the middle vertex in the molecule and the edges, the basis functions are nowhere 0 in the molecule and even at the edges, the value is still significantly large. In comparison, in PS-spline MPM it can occur that a basis function is exactly 0 near the edge of its molecule, and even if it is not exactly zero, it still approaches zero quadratically instead of linearly near the molecule boundary and therefore, the values there are much smaller compared to Lagrange basis functions.

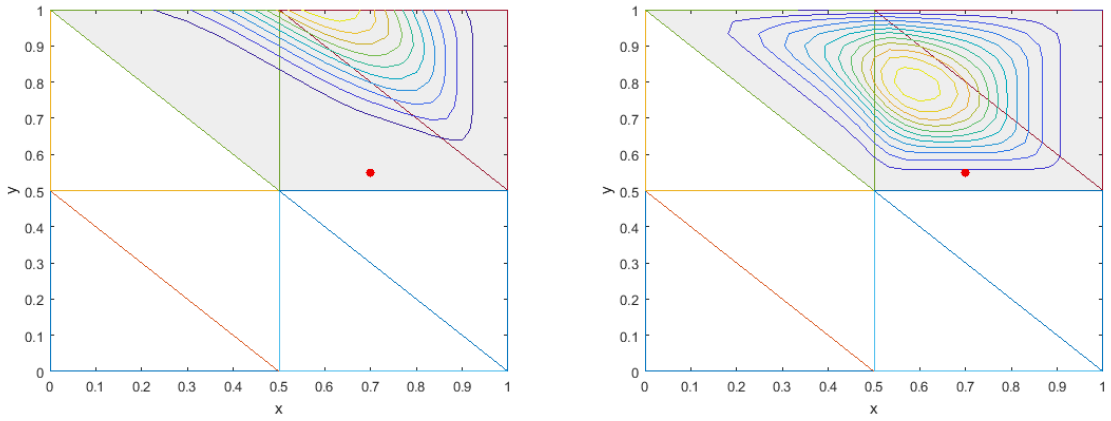
Altogether, PS-spline basis functions face a problem which is hardly an issue in classical Lagrange MPM: the PS-spline basis functions can be arbitrarily small or even zero near the edges of its molecule. When there are particles situated only in these areas, tiny values will appear in the mass matrix, which will become ill-posed, causing instabilities when solving for the acceleration $M\boldsymbol{a} = \boldsymbol{F}$ and during the velocity projection $M\boldsymbol{v} = \boldsymbol{P}$.

In the next sections, we will address this problem with a number of stability enhancing solutions, the most important of which are deactivating basis function, lumping, and partial lumping. All of these solutions will prove to be stability enhancing, but either not fully solve the problem, or influence the quality of the solution. Guaranteeing full PS-spline stability without loss of accuracy for arbitrary refinements remains an open issue.

5.2. Possible solution: deactivating basis functions

In this section, we investigate the first possible method to solve the instabilities caused by partially empty elements: deactivating the basis functions with insufficient support. The idea of this solution is very simple: in case a basis function is considered active, as there are particles in its molecule, we first check if its entries in the mass matrix are sufficiently large. In case the entries are not large enough, then the particles are all near the edges of its molecule and are close to zero in the considered basis function. Therefore, the considered basis function can be assumed to be inactive. Note that in PS-spline MPM, each vertex is associated with three basis functions, which share the same molecule. However, the basis functions all assume their maximum in different places of the molecule and can be very small or even zero in part of the molecule, see Figure 5.2. The right contour plot in this figure shows that everywhere in the grey molecule the basis function is significantly larger than zero, whereas the left function is insignificant near the bottom of the three elements of its molecule. In case there are only particles at the bottom of the molecule, it can be argued that the left basis function should be deactivated and set to zero before the system of equations is solved.

The proposed method is quite straightforward, but does contain a large issue: it is unclear how the bound should be chosen for deactivating a basis function. In order to get a better insight in this, the soil column with large deformation investigated in Section 4.4 is considered again. Recall that for a division of the grid into 1×4 blocks gave a stable computation, as for this case all elements contained sufficient particles to prevent



(a) A basis function with very small values near the lower boundary of its support.

(b) A basis function with significant values near the boundary of its support.

Figure 5.2: Two basis function associated with the upper middle vertex, sharing the same molecule. The molecule on which the basis function is defined is marked with grey. A material points is again marked with a red dot.

instabilities, see Figure 4.12. If we refine the grid to 1×8 blocks, the uppermost element will become empty, so there will also be a phase in which it is almost empty. When using the unadapted PS-spline MPM on the 8×1 block grid, the computation becomes unstable at around $t = 0.1$ and shortly thereafter particles leave the domain due to the instabilities. The situation at $t = 0.1$ is shown in Figure 5.3, we can observe that the topmost element is almost empty at that time, which corresponds to earlier observations that instabilities are caused by near-empty elements.

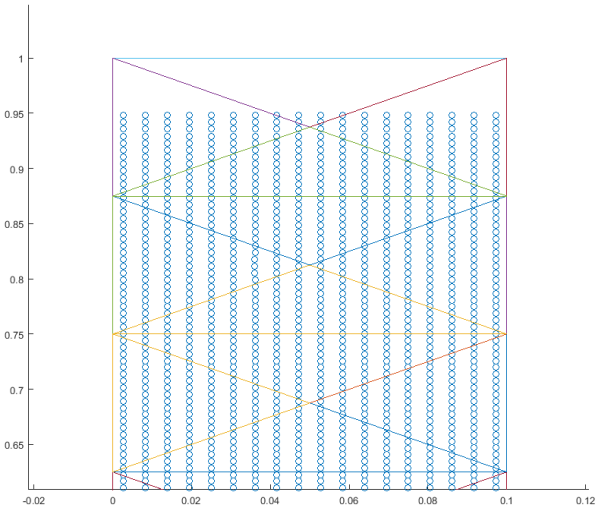
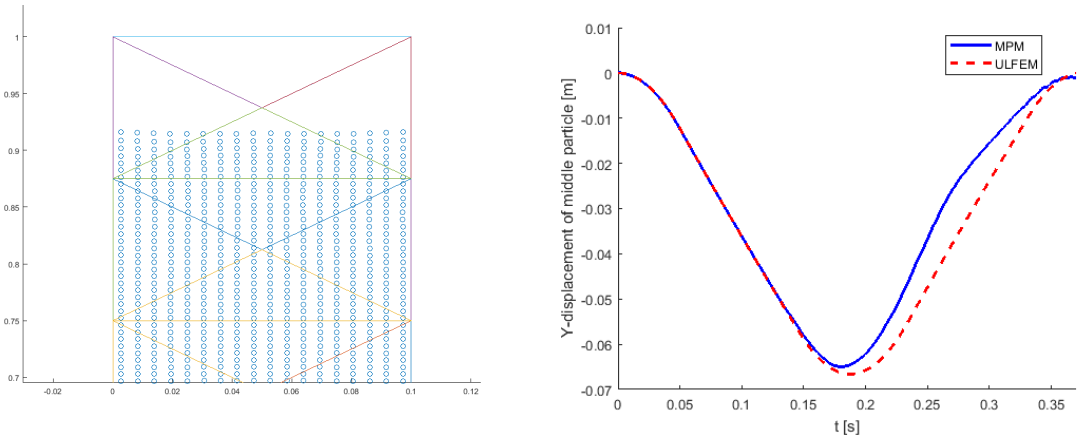


Figure 5.3: The particle positions at $t = 0.1$s in the soil column, simulated with unadapted PS-spline MPM. At this moment, instabilities start to occur.

If we run the same simulation again with a 1×8 block grid, but deactivate the nodes with $M_{ii} < 10^{-2}$ in the mass matrix, then the computation becomes stable. The result for the displacement is shown in Figure 5.4. Note that the simulation managed to finish, and thus no great instabilities have occurred, despite that an entire element has become empty during the simulation. The choice of $M_{ii} < 10^{-2}$ has purely been chosen based on trial and error. For a bound of 10^{-3}, the simulation still became unstable, again due to the fact that the mass matrix M was already ill-posed before the bound was reached and basis functions were deactivated. In theory, the bound should be chosen small to interfere as little as possible, but large enough such that the

simulation remains stable. If the bound is chosen too small, basis functions with insignificant support will not be detected in time, and the simulation remains unstable. If the bound is chosen too large, basis functions will be deactivated when they should be active and new errors are introduced. Therefore, the choice for the bound at which to deactivate a basis function is still vague, inconsistent and in many cases hard to apply.



(a) The particle positions at $t = 0.18$s in the soil column, simulated with PS-spline MPM with deactivating nodes.

(b) The resulting y-displacement in the middle particle.

Figure 5.4: The results for the simulations with a 1×8 block grid using PS-spline MPM with deactivating a basis function i when $M_{ii} < 10^{-2}$. On the left, the particle positions are shown at the moment of greatest deformation, on the right, the result of the simulation for the y-displacement is shown.

Besides the fact that it is hard to determine a proper deactivation bound, this method also has the large drawback that the numerical solution is prone to new errors. From Figure 5.4b, it is visible that deactivating basis functions also inflicts a large error. Due to the deactivation bound, basis functions with insufficient support are deactivated, which causes an additional error when calculating the acceleration and reconstructing the velocity. These degrees of freedom are set to zero, and the system of equations is solved under the condition that the coefficients of these basis functions are zero. Unfortunately, due to the deactivation of basis functions and replacing entries in the right hand side, mass and momentum momentum conservation is no longer guaranteed, and the reconstruction of the acceleration and velocity field will lose quality as well, especially at the boundary where the basis functions were deactivated. As the deactivated basis functions still had significant diagonal entries of $M_{ii} \approx 10^{-2}$, the error will also be significant in the acceleration and velocity field.

5.2.1. Deactivating basis functions: summary

In summary, deactivating basis function can be done when a basis function has few particles left in its molecule, and the basis function values in these particles are very small as well. From the mass matrix, it can be checked if this is the case, and as a result the basis function may be disabled. It has been shown that such an approach has a stabilising effect, but also introduces new errors. The deactivation bound currently has to be chosen relatively large in order to prevent an ill-posed mass matrix, but this causes significant errors in the solution. If these errors can be reduced sufficiently in future research, this method may be promising for further implementation.

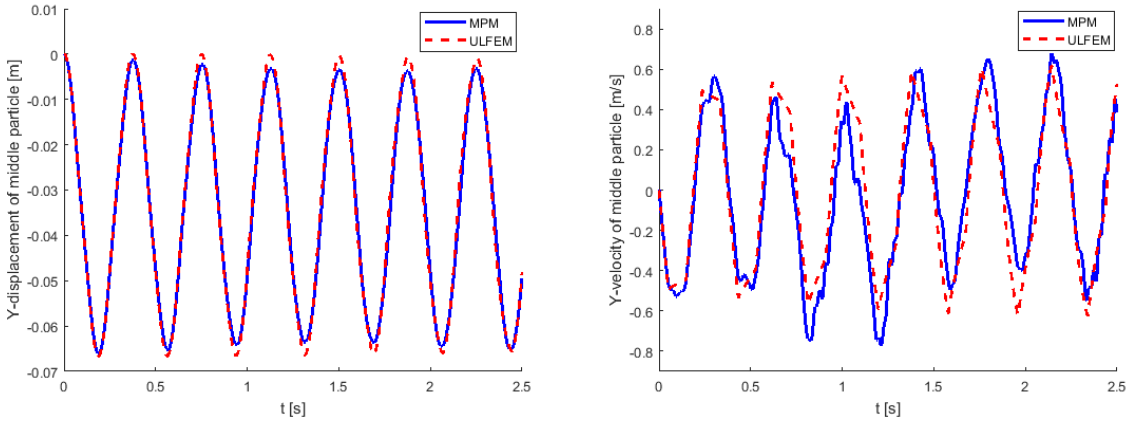
5.3. Possible solution: lumping

The second stability improving method we will consider here is lumping of the mass matrix. For more general details on matrix lumping for MPM, see Appendix A.2. As the instabilities occur when solving an ill-posed system of equations, approximating the solution to this system of equations by lumping may stabilise the simulation. Although lumping is very common in MPM, so far we have not used lumping as mostly coarse grids were used, and lumping on a coarse grid leads to additional errors in the L_2-projection [21]. In this section however, we are in the first place interested in increased stability.

In order to understand why lumping may have a stabilising effect, we first shortly revise the most impor-

tant aspect of lumping. When solving a system of equation $A\boldsymbol{x} = \boldsymbol{b}$ with lumping, the system is replaced by its lumped counterpart, $\tilde{A}\boldsymbol{x} = \boldsymbol{b}$ with A sparse. The lumped matrix $\tilde{A}$ is created by summing the mass of each entire row and put in the diagonal element, which leads to a diagonal matrix $\tilde{A}$. The lumped system of equations is then solved as $x_i = \frac{b_i}{\tilde{A}(i,i)}$, as $\tilde{A}$ is now a diagonal matrix. Lumping is a technique commonly used in MPM and significantly improves the speed of computations, as the coefficient $\boldsymbol{x}$ can be found without solving a system of equation. The cost of lumping is that the spatial accuracy of the L_2-projections decreases. Lumping may also have a stabilising effect: because no longer a system of equations is solved, the ill-posedness of a matrix is no longer a problem, as each coefficient x_i is calculated directly.

Next, we consider simulations with PS-spline MPM with lumping. Recall that the soil column with large deformations would show instabilities for the unadapted PS-spline MPM simulations on a 1×8 block grid. We will now consider the same simulation again, but using a lumped mass matrix for determining the acceleration field and reconstructing the velocity as in Equation 2.18 and Equation 2.37 respectively. The results for lumped PS-spline MPM simulations of a soil column undergoing large deformations with a 1×8 block grid is shown in Figure 5.5.



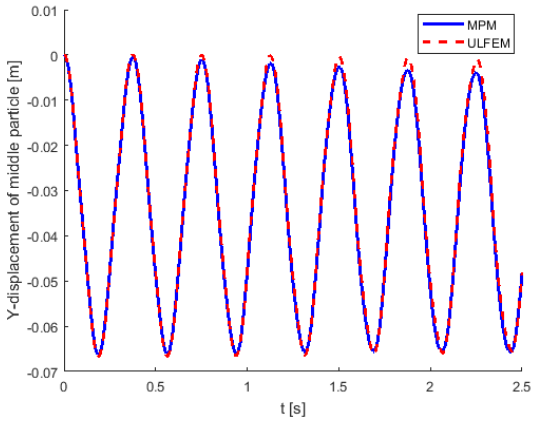
(a) The y-displacement with lumped PS-spline MPM.

(b) The y-velocity with lumped PS-spline MPM.

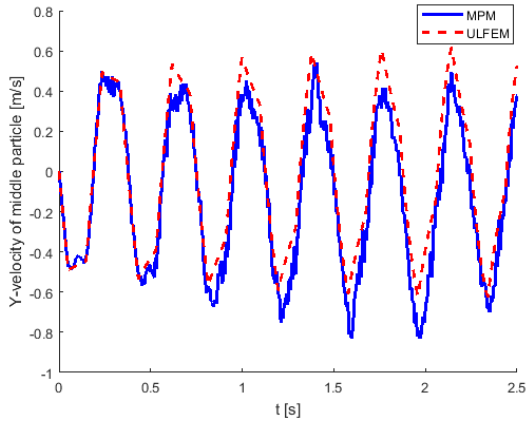
Figure 5.5: The results for the simulations with a 1×8 block grid using lumped PS-spline MPM

Note that the simulation was fully finished for multiple cycles, so we can conclude that the simulation is stable. Also the frequency of the main oscillation is captured very well. However, the amplitude of the displacement shows errors that have not been seen before in PS-spline MPM, and the results for the velocity are much worse than in the non-lumped PS-spline simulation in Figure 4.13. We can conclude that lumping does have a stabilising effect: again elements were completely empty during this simulation, but the simulation was never unstable. However, the increased stability comes at the cost of decreased accuracy of the solution. We have investigated if this error can be compensated by refining the grid into more elements, Figure 5.6 shows results of lumped PS-spline MPM on a 2×16 block grid. Despite the refinement, the results are similar to the less refined case. Apparently, the refinement of the grid is not the main cause of the observed error in the displacement and velocity.

As we know that lumping causes additional errors in the L_2-projection, we will further investigate the spatial velocity profile instead of only investigating the time profile of a single particle. To do so, we have considered again the test case of a vibrating bar undergoing small deformations. This is the most plain test case considered in this thesis: particles rarely leave their elements, all elements are always completely occupied, and therefore, any error should be due to the lumping of the mass matrix. The results are shown in Figure 5.7. For this case, a refined grid of 24×1 blocks was used, but again the solution shows differences from the analytical solution that were not present for unadapted PS-spline MPM. The cause of the error becomes visible from Figure 5.7b: large oscillations occur in the velocity profile through the bar. This is in agreement with the fact that lumping causes additional errors in the L_2 projection. This would also explain the errors in the oscillation in the movement of a single particle: the main movement of the entire bar is captured well, but due to spatial oscillations, errors will also appear in the time profile of a single particle. We have further inspected



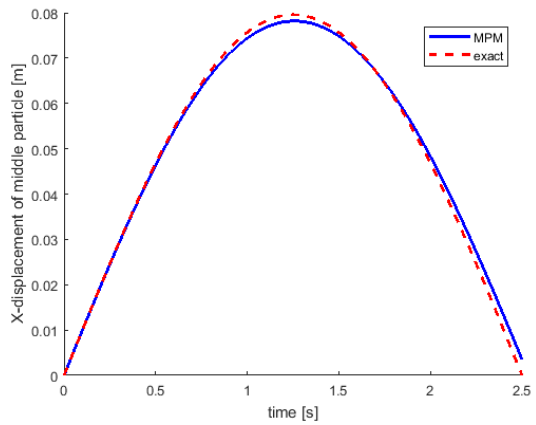
(a) The y-displacement with lumped PS-spline MPM.



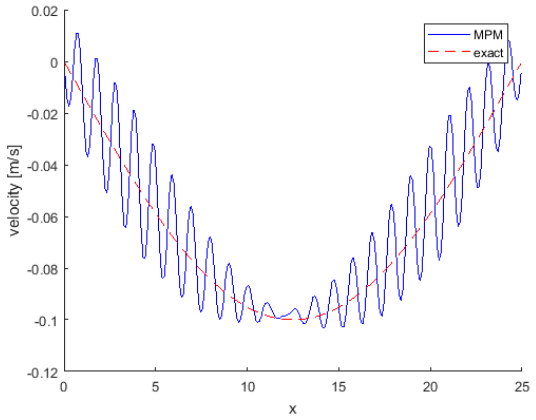
(b) The y-velocity with lumped PS-spline MPM.

Figure 5.6: The results for simulations of a soil column with a 2×16 block grid using lumped PS-spline MPM

the L_2-projection using a lumped mass matrix, but these vibration are not directly caused by the function reconstruction: the lumped L_2-projection of a sin on this basis of PS-splines yield much smaller smaller errors than displayed in Figure 5.7b, however, the oscillation in Figure 5.7b has the same frequency: 24 oscillation for a 24×1 block grid. The magnitude of the error may therefore be caused by the fact that at each iteration, the acceleration is calculated with about the same error due to lumping every time step. The total error in the velocity would then be the sum of all these errors, causing larger oscillations in the velocity profile over time. This is in correspondence with the observation that the errors for the observed cases grow larger over time, but are quite small at the start of the simulation. However, for lumped Lagrange MPM, these large spatial oscillation have not been observed. Further research may be conducted to find out why PS-spline MPM is more effected by lumping than Lagrange MPM, and how these errors can be mitigated for PS-spline MPM. This research would be of great importance for many engineering applications, as lumping is a very common technique within MPM and many applications require lumping to run large simulations in reasonable time.



(a) Vibrating bar, x-displacement of middle particle.



(b) Vibrating bar, the x-velocity through the entire bar at $t = 2.5$ s.

Figure 5.7: The results for simulation of a vibrating bar undergoing small deformation. A 24×1 block grid was used, together with lumped PS-spline MPM. The velocity at $t = 2.5$ s is shown on the right, at this time, the analytical absolute velocity is at its maximum.

5.3.1. LUMPING: SUMMARY

We have found that lumping greatly improves the stability of PS-spline MPM, but at the same time adds significant errors to the solution. Using a lumped mass matrix causes additional errors in the L_2-projection compared to using a consistent mass matrix, and therefore additional spatial inaccuracies can appear. Large errors

can appear as these spatial errors increase over time. This still poses a problem for application in engineering, as lumping is often required for MPM simulations with many degrees of freedom. However, if the lumping errors could be mitigated, lumped PS-spline MPM would be both stable and accurate, and be promising to further develop for engineering applications.

5.4. POSSIBLE SOLUTION: PARTIAL LUMPING

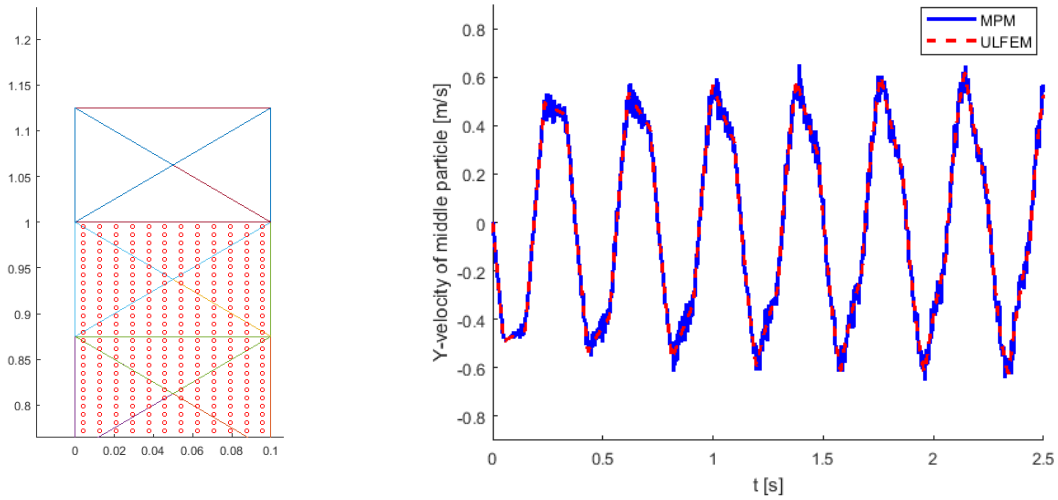
In the previous section, we have observed that lumped PS-spline MPM has very good stability properties, but also additional L_2-projection errors appear. To get a method that is stable, without adding large additional errors, we will also consider an additional method: partial lumping. Recall that the objective of this section is to find a PS-spline MPM method that is stable when elements become partially empty. Lumping has proven successful in stabilising PS-spline MPM, but is accompanied with large errors. The lumping is mainly necessary for preventing that the consistent mass matrix becomes ill-posed due to the basis functions at the partially empty elements. The idea of partial lumping is to only lump the mass matrix for these specific basis functions, and solve the remaining system consistently.

To implement partial lumping, first the basis functions on the edge of the element must be identified. To find the boundary basis functions, we simply check if the corresponding vertex connects a filled element and an empty element. The empty element is per definition outside the domain, and the filled element inside, so a vertex connecting a filled and an empty elements must be a boundary vertex. The Powell-Sabin spline basis functions associated with these vertices are considered boundary functions. During the MPM time stepping process, the rows in the mass matrix will be lumped for specifically these basis functions and solved consistently for all the other degrees of freedom. In order for this procedure to function, the grid simulation needs additional information on where its boundaries are. Additional empty elements are required on the top of the soil column benchmark, to identify the top basis functions as boundary basis functions. Therefore, the grid is extended at the top with an additional block of elements, without any particles inside, see Figure 5.8a. These will then be recognised as empty elements outside the domain, and the basis function at the physical top of the soil column will be recognised as boundary basis functions.

Using the 1×8 grid shown in Figure 5.8a, the soil column simulation has been run again with partially lumped PS-spline MPM. Recall that this test case became unstable when the same grid was used with unadapted PS-spline MPM. The result for the velocity is shown in Figure 5.8b, which looks relatively good. The displacement is not shown here, but it was almost indistinguishable from the ULFEM solution. Altogether, the results are quite good, and very similar to the result we have seen before in Figure 4.13 for the unadapted PS-spline MPM on a very coarse grid. This is to be expected, as most of the system used in partially lumped PS-spline MPM is solved with the consistent mass matrix, and only the boundary basis functions are lumped. The oscillations that can still be observed in the velocity are due to the lumping at the boundary, but the general picture is conserved well enough. However, when the grid was refined further, from 1×8 blocks to 2×16 blocks, the simulation would become unstable again. Apparently, for this case we have not lumped enough basis functions, as it is known that fully lumped PS-spline MPM is stable. To get a method that is both stable and accurate, a balance has to be made between lumping more elements near the boundary for better stability on one hand, and lumping as few basis functions as possible to retain an accurate solution on the other hand. Further research may investigate a better rule to determine a more general balance on whether a basis function should be lumped or not to guarantee stability.

5.4.1. PARTIAL LUMPING: SUMMARY

In summary, partial lumping is a combination between consistent and lumped PS-spline MPM, which only lumps the mass matrix entries of basis functions on the boundary of the domain. As more basis functions are lumped, the method becomes more stable, but less accurate. One simulation was run which showed good results for both the stability and the accuracy of the solution. Further research can be conducted on optimising the balance between stability and accuracy for general cases.



(a) Soil column grid, with an extra block on top. (b) y-velocity of middle particle in a soil column.

Figure 5.8: The results of partially lumped PS-spline MPM for a simulation of a soil column undergoing large deformation. A 1×8 block grid was used with one additional empty block on top

6

CONCLUSION

The material point method (MPM) is a well known finite element technique to numerically simulate large soil deformation problems using a triangular grid. However, it is classically operated with piece-wise linear basis functions, which causes low-order spatial convergence and grid crossing errors due to the discontinuity of the derivatives of the basis functions. The goal of this study was to extend the material point method with higher-order C^1-continuous basis functions on triangulations to solve these issues. For this reason, a C^1-continuous higher-order B-spline basis on arbitrary triangulations has been investigated for MPM, which has led to the use of quadratic Powell-Sabin (PS) splines, which are non-negative and C^1-continuous. It has been shown that the L_2-projection of an analytic function onto a basis of PS-splines shows the expected higher-order spatial convergence. Furthermore, the PS-splines have also successfully been implemented in MPM, and benchmarks have been run for small and large deformations in both a vibrating bar and in a soil column. For the vibrating bar benchmark, higher-order spatial convergence of PS-spline MPM has been observed. Furthermore, grid crossing no longer causes errors when PS-spline MPM is used. In the investigated benchmarks, PS-spline MPM has shown good results when few elements were used, and the method seems promising for further application in problems which require a large number of basis functions: PS-spline MPM may significantly reduce the number of basis functions required and therefore save time. However, PS-spline MPM also faces a number of unsolved issues before it may actually be used in engineering applications.

The largest problem we have encountered with PS-spline MPM is the occurrence of instabilities in the method. If these instabilities occur, particles will have nonphysical speeds and travel out of the domain. In general, sliver elements should be avoided and stability conditions for the time step size should be respected, but a more fundamental problem shows up as well: PS-spline MPM has large difficulties in handling partially empty elements. The values of PS-spline basis functions can become very small in its support, and if all particles in the support are located in such regions, this leads to very small entries in the mass matrix. As a consequence, the mass matrix becomes ill-conditioned, and consistently solving the mass matrix causes instabilities. We have proposed several solutions in this thesis, that have proven to enhance stability, though often at the cost of a loss of accuracy.

One possible solution proposed was to deactivate the PS-spline basis function with insufficient support. If a basis function only attained very small values in all particle locations in its support, it would be deactivated and its coefficient is set to zero. This method has proven effective in stabilising the method, but causes significant errors in the final solution for the displacement. Furthermore, the choice of bound at which basis functions are deactivated is very arbitrary, and may differ greatly for different problems and different grids. All together, this method would require a lot of development to function properly.

The most promising mitigation strategy for the instabilities due to almost empty elements is to incorporate lumping of the mass matrix into the method. In most engineering applications, lumping is an essential part of MPM, as it saves a lot of time for the computations, and therefore, lumping was also incorporated into PS-spline MPM. Lumping is known to have a stabilising effect, and when implemented in PS-spline MPM, the computations became very stable. However, lumping caused large additional errors in the spatial accuracy. It has been observed that the L_2-projection using a lumped matrix shows small spatial errors, but these errors

may be summed over time, resulting in a large spatial error for longer simulations. Further study would be required to solve or mitigate this issue for lumped PS-spline MPM. Instead of using fully lumped PS-spline MPM, it is also possible to use partial lumping. In this method, the consistent mass matrix is used, but only the entries of the basis functions at the boundary that cause the instabilities are lumped. This leads to a more stable method, without adding a very large additional error. As more entries of basis function in the mass matrix are lumped, the method becomes more stable at the cost of a larger additional lumping error.

RECOMMENDATIONS

For follow-up research, the most important issue would be to mitigate the lumping error. For many engineering application, lumping is essential, as it significantly reduces the time required for MPM simulation compared to using a consistent system of equations. Therefore, PS-spline MPM will be less attractive for computations with many degrees of freedom if full lumping is still prone to large errors. This is one of the larger issues of PS-spline MPM and should be further researched with higher priority than the other recommendations. Once this has been resolved, other possible tracks may also be pursued.

Another possible follow-up research may be on the use of function interpolation and Gauss quadrature integration. It may be possible to incorporate Gauss integration into the method, to reduce the integration error, such that less particles can be used for a comparable accuracy. Tielen [19] and Wobbes et al. [22] have shown that function reconstruction for using a Gauss quadrature rule can further improve MPM, and this may also be the case for PS-spline MPM.

In this thesis, we have spent some attention on implementing cubic PS-splines into MPM, but currently this track seems less promising. The errors in PS-spline MPM are no longer dominated by the spatial reconstruction method. Therefore, using cubic PS-splines would have little advantage over quadratic PS-splines. Furthermore, the implementation of cubic splines is more cumbersome than quadratic PS-splines, and the construction of the functions is more expensive as well. For these reasons, we would recommend further study into these basis functions mainly when the spatial function reconstruction of quadratic PS splines becomes a limiting factor for the total error in MPM. Once a higher-order time stepping method and quadrature integration method have been implemented as well, the spatial error may become more dominant again, in which case cubic PS-splines would become interesting.

Another recommendation is to use 2D PS-spline MPM to simulate representative 2D benchmarks, instead of 1D benchmarks with an additional dimension. Some examples of these are pile driving and the collapse of a soil slope. The 1D benchmarks have proven useful to investigate the performance of PS-spline MPM, but the method should also be tested for more advanced 2D benchmarks.

A final recommendation for follow-up research is the extension of PS-spline MPM to 3D. The step from 1D to 2D spline MPM was cumbersome, but scaling up PS-spline MPM from 2D to 3D will most probably prove more straightforward. For most aspects, we expect that the 3D case will be similar to the 2D case, often simply adding one extra coordinate. Further study into 3D PS-spline MPM may prove very beneficial, as for 3D problems, the number of basis functions are often much higher than for 1D and 2D. Therefore higher-order spatial convergence may be of even greater importance to reduce the number of basis functions required than for 1D or 2D.

A

APPENDIX

A.1. STRESS AND STRAIN

For the reader who is not familiar with the concepts of stress and strain, and the resulting motion, we summarise the basics in this appendix.

STRESS

The stress is defined as the internal force, that neighbouring particles exert on each other. Two types of stress can be distinguished, normal stress and shear stress. Normal stress is the force per area in the normal direction of the considered plane. For example, consider the stress over a horizontal plane in a vertical bar, as shown in Figure A.1. The upper part pushes on the lower part and vice versa. This imaginary plane is being pushed on from two sides in a direction normal to the plane. This is the normal stress component of this specific plane. It is also possible that the upper and lower part of the bar are pulling at each other, then the normal stress changes sign. In this thesis, tensile forces will be assumed to be positive, whereas compressive forces are negative. Also note that the normal stress is dependent on the orientation of the plane. If a vertical plane is considered, then there would be no normal stress on that plane, as in this example there is only a force field in the z-direction, which is not a normal direction of a vertical plain.

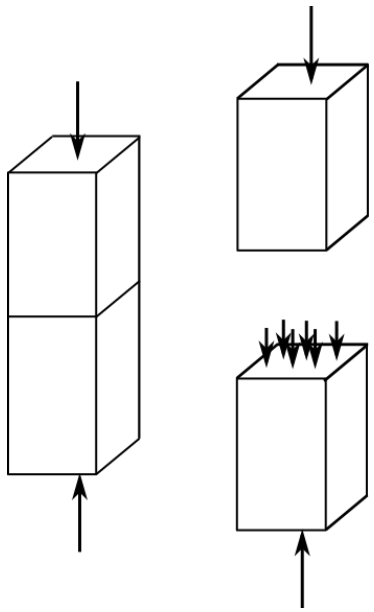


Figure A.1: Illustration of normal stress in a compressed bar.

The shear stress is defined as the force per area parallel to the plane it is exerted on. An example is shown in Figure A.2. When two blocks are horizontally next to each other, and the right one is pushed down and the

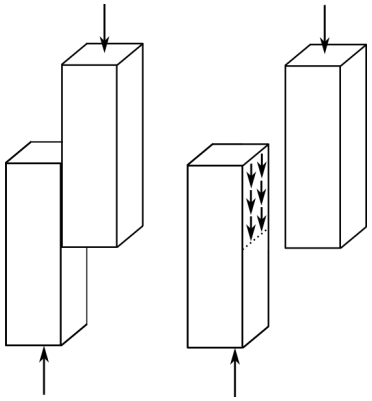


Figure A.2: Illustration of shear stress when two bars are pressed along each other.

left one pushed up, there will be shear stress over the interface. This stress has two components, in the figure, the component in the z-direction is non-zero, but the shear stress in the y-direction is zero.

Having defined the normal and the shear stress of a certain plane, it is possible to describe the full stress in a point by considering an infinitesimal cube around that point. The stress on opposite planes is equal, as the cube is infinitely small, so we only consider three planes: the three planes respectively orthogonal to the $x_1()$-, $x_2()$- and $x_3()$-direction, as is illustrated in Figure A.3. On each of these planes, there is one normal stress component and two parallel stress components. These components can be stored in the stress tensor, where σ_{ij} represents the j^{th} component of the force on the plane normal to the x_i direction, as shown in Figure A.3. Note that when $\sigma_{ij} \neq \sigma_{ji}$ in the cube in the figure, there will be an angular momentum and thus a rotational acceleration. In case the cube is in steady state, then $\sigma_{ij} = \sigma_{ji}$ and so the stress tensor should be symmetric.

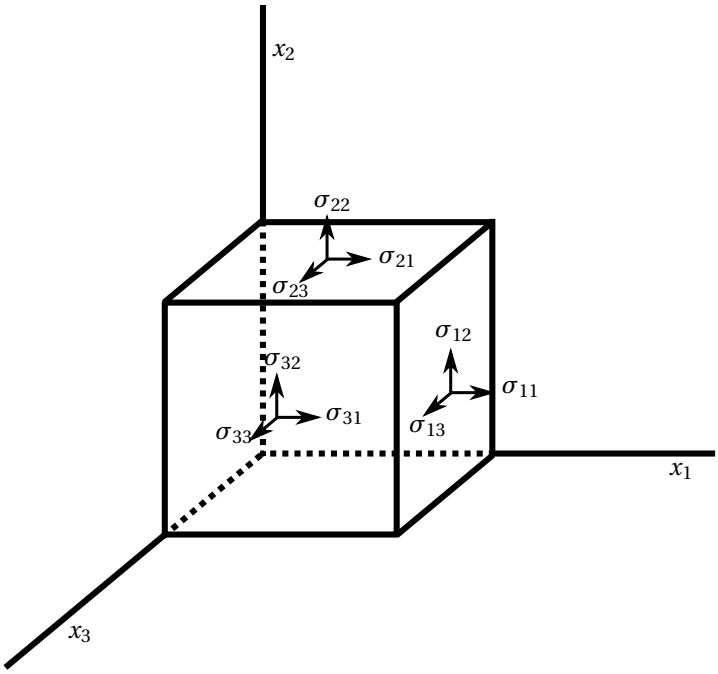


Figure A.3: The stress components in a single point, described as infinitesimal cube.

FROM STRESS TO MOTION

In order to get the resulting force from the stress, a small volume can be considered. A 2D volume will be considered and the resulting total force on this volume will be calculated. Figure A.4 shows a square with four

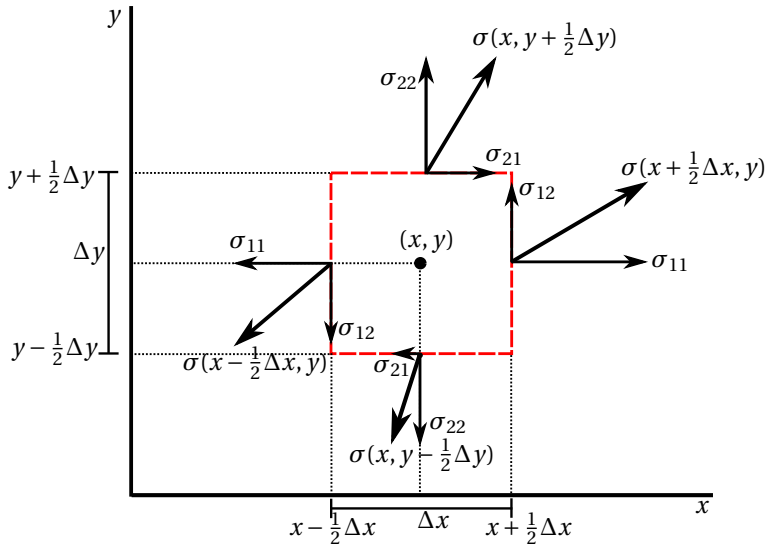


Figure A.4: A 2D volume of size Δx by Δy. On each of the four faces of the cube, the stress is assumed to be constant. The stress vector working on each of the faces is depicted, decomposed in its x and y components.

sides. The center of the rectangle is (x, y), and the corners are at $(x \pm \frac{\Delta x}{2}, y \pm \frac{\Delta y}{2})$. A force field works on each of those sides, and is assumed to be constant over the side. Now consider the total force in the x-direction. This force is equal to

$$F_x = \Delta y \left[\sigma_{11}\left(x+\tfrac{\Delta x}{2}, y\right) - \sigma_{11}\left(x-\tfrac{\Delta x}{2}, y\right)\right] + \Delta x \left[\sigma_{21}\left(x, y+\tfrac{\Delta y}{2}\right) - \sigma_{21}\left(x, y-\tfrac{\Delta y}{2}\right)\right]$$

$$\Rightarrow \frac{F_x}{\Delta x \Delta y} = \frac{\sigma_{11}\left(x+\frac{\Delta x}{2}, y\right) - \sigma_{11}\left(x-\frac{\Delta x}{2}, y\right)}{\Delta x} + \frac{\sigma_{21}\left(x, y+\frac{\Delta y}{2}\right) - \sigma_{21}\left(x, y-\frac{\Delta y}{2}\right)}{\Delta y}$$

By taking the limit of Δx and Δy to zero, we get

$$f_x(x, y) = \frac{\partial \sigma_{11}}{\partial x}(x, y) + \frac{\partial \sigma_{21}}{\partial y}(x, y). \tag{A.1}$$

Note that $\boldsymbol{f}$ is a force density. Equation A.1 can be generalised for any direction. Rename x as x_1 and y as x_2 etc. Using the Einstein summation convention, the internal force can be written as [23]:

$$f_i = \frac{\partial \sigma_{ji}}{\partial x_j}. \tag{A.2}$$

In divergence notation, this becomes

$$\boldsymbol{f} = (f_1, f_2) = \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}\right) \cdot \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} = \nabla \cdot \boldsymbol{\sigma}. \tag{A.3}$$

This accounts for the internal force in the conservation of momentum in Equation 2.3.

Strain

The theory about strain in this part was retrieved from [1] and [24], see these for more details. The strain of a material is the deformation from its original geometry. The strain $\boldsymbol{\epsilon}$ is denoted as a tensor,

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{11} & \frac{1}{2}\gamma_{12} & \frac{1}{2}\gamma_{13} \\ \frac{1}{2}\gamma_{21} & \epsilon_{22} & \frac{1}{2}\gamma_{23} \\ \frac{1}{2}\gamma_{31} & \frac{1}{2}\gamma_{32} & \epsilon_{33} \end{bmatrix}. \tag{A.4}$$

The diagonal element represents the normal strain, whereas the non-diagonal elements denote the shear strain. The strain can be determined from the position dependent displacement $\boldsymbol{u}(\boldsymbol{x})$, similar to the stress

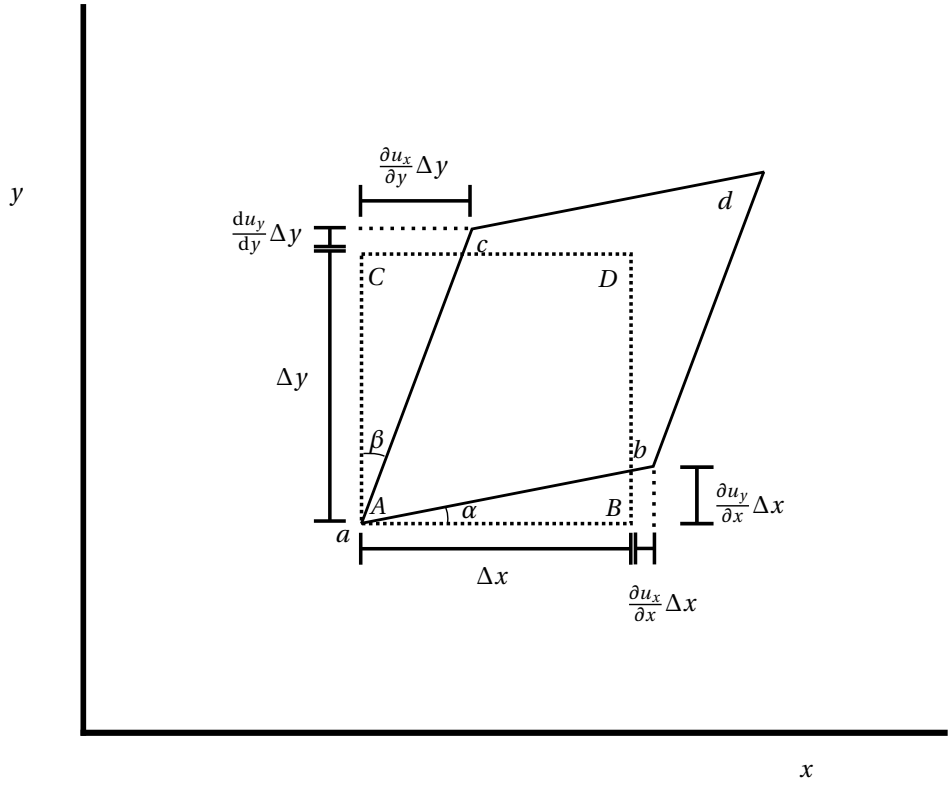


Figure A.5: A 2D volume element undergoing infinitely small deformation. The deformation in the illustration is exaggerated for the sake of clarity. The dotted rectangle is the original geometry, and the solid quadrilateral is the deformed cube.

tensor. To derive the strain, consider an infinitesimal small rectangle of size $\Delta x \times \Delta y$. This rectangle may move and deform over time, as shown in Figure A.5. If the stretching of AB is considered, the length after deformation of the line piece is equal to

$$\begin{aligned} ab &= \sqrt{\left(\Delta x + \frac{\partial u_x}{\partial x}\Delta x\right)^2 + \left(\frac{\partial u_y}{\partial x}\Delta x\right)^2} \\ &= \Delta x \sqrt{1 + 2\frac{\partial u_x}{\partial x} + \left(\frac{\partial u_x}{\partial x}\right)^2 + \left(\frac{\partial u_y}{\partial x}\right)^2}. \end{aligned}$$

Under assumption that $\frac{\partial u_x}{\partial x} \ll 1$ and $\frac{\partial u_y}{\partial y} \ll 1$, the second-order terms can be neglected and the square root can be approximated,

$$\begin{aligned} ab &= \Delta x \sqrt{1 + 2\frac{\partial u_x}{\partial x}} \\ &\approx \Delta x \cdot \left(1 + \frac{\partial u_x}{\partial x}\right) \\ \xRightarrow[\Delta x \to 0]{} ab &= \Delta x + \frac{\partial u_x}{\partial x}\Delta x \end{aligned} \tag{A.5}$$

Therefore, line piece AB has stretched with $\frac{\partial u_x}{\partial x}\Delta x$, so the stretching per unit length is $\frac{\partial u_x}{\partial x}$. In general, the normal strain in direction x_i becomes

$$\epsilon_{ii} = \frac{\partial u_i}{\partial x_i}. \tag{A.6}$$

The shear strain is related to the change in angle of corner A. This change in angle is equal to $\alpha+\beta$ in Figure A.5. The formulas for α and β can be derived from geometric rules, under the assumption of small displacements and small angles.

$$\tan(\alpha) = \frac{\frac{\partial u_y}{\partial x}\Delta x}{\Delta x + \frac{\partial u_x}{\partial x}\Delta x} = \frac{\frac{\partial u_y}{\partial x}}{1+\frac{\partial u_x}{\partial x}} \tag{A.7}$$

$$\left(\tan\alpha \approx \alpha, \frac{\partial u_x}{\partial y} \ll 1\right) \quad \Rightarrow \alpha = \frac{\partial u_y}{\partial x} \tag{A.8}$$

$$\tan(\beta) = \frac{\frac{\partial u_x}{\partial y}}{1+\frac{\partial u_y}{\partial y}\Delta y} \tag{A.9}$$

$$\left(\tan\theta \approx \theta, \frac{\partial u_x}{\partial y} \ll 1\right) \quad \Rightarrow \beta = \frac{\partial u_x}{\partial y} \tag{A.10}$$

Therefore, the total change in angle will be $\alpha+\beta = \frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}$. In general, for infinitesimal deformations this yields a total shear strain of

$$\gamma_{ij} = \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{A.11}$$

Note that the shear strain is symmetric, $\gamma_{ij} = \gamma_{ji}$. The term will appear twice in the strain tensor, and for normalisation, this term appears as $\frac{1}{2}\gamma_{ij}$ in the tensor. Note that the strain can also be written in incremental form

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{A.12}$$

This form is in agreement with Equations A.6 and A.11.

For MPM the strain is required in differential form. Each time step, the total strain will change slightly. Therefore, the derivative with respect to the strain to the time is needed. The differential form is easily retrieved from Equation A.12, as $\frac{\partial u_i}{\partial t} = v_i$, where v_i is the velocity in the x_i direction. Therefore, the strain in differential form becomes

$$\frac{\partial \epsilon_{ij}}{\partial t} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right). \tag{A.13}$$

Constitutive relation

To determine the stress from the strain, a so called constitutive relation is required. The constitutive relation used here was retrieved from [1] and [9]. The constitutive relation gives the change in stress as function of the strain. Note that the Einstein summation convention for repeated indices is used throughout this section. The constitutive relation between stress and strain is given by

$$\frac{\partial \sigma_{ij}}{\partial t} = \overset{\nabla j}{\sigma}_{ij} + \omega_{ik}\sigma_{kj} - \sigma_{ik}\omega_{kj}, \tag{A.14}$$

in which $\overset{\nabla J}{\sigma}_{ij}$ denotes the Jaumann rate of stress and ω_{ij} the spin tensor. The Jaumann rate of stress represents the non-rotational part of the change in stress. The last two terms with the spin tensor represent the torque, which results in the rotational acceleration. The spin tensor ω_{ij} is defined as the anti-symmetric part of the velocity gradient,

$$\omega_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i}\right). \tag{A.15}$$

The Jaumann rate of stress is defined as

$$\overset{\nabla J}{\sigma}_{ij} = \overset{\nabla H}{\sigma}_{ij} + \frac{\partial \epsilon_{kk}}{\partial t}\sigma_{ij}, \tag{A.16}$$

in which $\overset{\nabla H}{\sigma}_{ij}$ denotes the Kirchhoff stress, also known as the Hill stress. The second term represents a non-linear term, which is of importance in case of large deformations. In this thesis, the material is assumed to be

isotropic and linear elastic, so the force caused by stretching the material is proportional to the deformation. This implies a linear inversible relationship between the Kirchhoff stress and strain, independent of direction. In this case, the Kirchhoff stress can be written as

$$\overset{\nabla H}{\sigma}_{ij} = D_{ijkl}\frac{\partial \epsilon}{\partial t}. \tag{A.17}$$

Substituting the equations above into Equation A.14 leads to the final constitutive relation,

$$\frac{\partial \sigma_{ij}}{\partial t} = D_{ijkl}\frac{\partial \epsilon_{kl}}{\partial t} - \frac{\partial \epsilon_{kk}}{\partial t}\sigma_{ij} + \omega_{ik}\sigma_{kj} - \sigma_{ik}\omega_{kj}, \tag{A.18}$$

Only the term D_{ijkl} has not been defined yet. This term gives a linear relation between the Kirchhoff stress and the strain, derived from Hooke's law. The relation is given by

$$D_{ijkl} = \left(K - \frac{2}{3}G\right)\delta_{ij}\delta_{kl} + G\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right). \tag{A.19}$$

The δ_{ij} denotes the Kronecker delta, which is 1 if $i = j$ and zero otherwise. K and G represent the bulk modulus and shear modulus respectively; the bulk modulus is the ratio between pressure and the corresponding decrease of volume, and the shear modulus is the ratio between the shear strain and the resulting shear stress. The bulk modulus and shear modulus are both determined by the Poisson ratio ν and the Young's modulus E, which are material properties:

$$K = \frac{E}{3(1-2\nu)} \quad \text{and} \quad G = \frac{E}{2(1+\nu)}. \tag{A.20}$$

The Poisson ratio ν describes the ratio between transversal expansion to axial compression. That is, when a volume gets compressed in one direction, it will generally expand in directions orthogonal to the compression. The rate at which this happens is the Poisson ratio. Finally, the Young's modulus is the ratio between axial stress and the corresponding axial strain. For all numerical simulations, the Poisson ratio and Young's modulus will be given for a material. With these, the bulk and shear moduli are determined and substituted in Equation A.18. Finally this constitutive relation is used to determine the stress from the strain.

A.2. Matrix lumping

In the material point method, matrix lumping is often used to solve a system of equations

$$M\boldsymbol{a} = \boldsymbol{F},$$

in which M is the mass matrix. The mass matrix is constructed as

$$M_{ij} = \int_\Omega \phi_i \rho \phi_j \, \mathrm{d}\Omega \approx \sum_{p=1}^{n_p} V_p \rho_p \phi_i(\boldsymbol{x}_p)\phi_j(\boldsymbol{x}_p).$$

As the basis functions ϕ have only local support, most ϕ_i and ϕ_j do not share any support at all, and the mass matrix will therefore be sparse. Solving $\boldsymbol{a} = M^{-1}\boldsymbol{F}$ costs $\mathcal{O}(n^3)$ time, with n the number of equations. As the matrix is sparse, several techniques exist to accurately approximate the solution of the system of equation in only $\mathcal{O}(n^2)$ time, but this can often still take too much time, as the number of basis functions n may go up to a million for large problems.

Instead of solving the system of equations in the previously mentioned ways, it is also possible to solve this system of equation with the lumped matrix $\tilde{M}$, which is very common in MPM. The lumped matrix can be constructed from the original mass matrix, which is often referred to as the consistent mass matrix. To construct the lumped mass matrix, all entries or 'mass' of a row in the consistent mass matrix is summed and put on the diagonal, whereas all other entries are set to zero:

$$\tilde{M}_{ii} = \sum_{j=1}^{n} M_{ij},$$

The result is a diagonal matrix $\tilde{M}$, with all the mass of the rows lumped onto the diagonal, hence the name lumped mass matrix. An example is given for a small matrix:

$$M = \begin{bmatrix} 4 & 2 & 0 & 1 \\ 1 & 8 & 1 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 2 & 0 & 1 \end{bmatrix} \quad \Rightarrow \quad \tilde{M} = \begin{bmatrix} 7 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

The system $\tilde{M}\boldsymbol{a} = \boldsymbol{F}$ can then be solved as

$$a_i = \frac{F_i}{\tilde{M}_{ii}}.$$

An important condition for lumping is that no negative entries may occur in the consistent mass matrix. If negative entries do occur, it is possible that the lumped mass matrix can have diagonal entries equal or close to zero, which may result in nonphysical high accelerations.

In MPM, the lumped mass matrix $\tilde{M}$ can either be assembled by first constructing the consistent mass matrix and then lumping, however, we may also use *direct lumping*. For direct lumping, recall the partition of unity property of the basis functions:

$$\sum_{j=1}^{n} \phi_j = 1.$$

Therefore, the lumped mass matrix can be constructed as

$$\begin{aligned} \tilde{M}_{ii} &= \sum_{j=1}^{n} M_{ij} \\ &= \sum_{j=1}^{n} \sum_{p=1}^{n_p} V_p \rho_p \phi_i(\boldsymbol{x}_p) \phi_j(\boldsymbol{x}_p) \\ &= \sum_{p=1}^{n_p} V_p \rho_p \phi_i(\boldsymbol{x}_p) \cdot \sum_{j=1}^{n} \phi_j(\boldsymbol{x}_p) \\ \Rightarrow \tilde{M}_{ii} &= \sum_{p=1}^{n_p} V_p \rho_p \phi_i(\boldsymbol{x}_p) \end{aligned}$$

A.3. Cubic Powell-Sabin spline basis functions

In this section, the construction of cubic Powell-Sabin B-splines will be considered and analysed for use in MPM. The cubic PS-spline basis function are piece-wise cubic polynomials, with C^1 continuity over (sub-)element interfaces. Furthermore, they possess the same properties as quadratic PS-splines: local support, partition of unity and non-negativity. These basis functions have not been implemented in the MPM scheme considered in this thesis, but would be the natural choice for higher-order basis function in case higher-order spatial convergence is desired.

The construction process of the cubic PS-splines is in many ways similar to the construction process of quadratic PS-splines; first the grid is refined, second, from the grid, control triangles are constructed, which are similar to the PS-control triangles for quadratic PS-splines. Finally, the basis functions are created from the control points. First, we will start again with the refinement of the grid.

Given an arbitrary triangulation Δ, the grid may again be refined into a PS-refinement Δ^*. An example of a PS-refinement is shown in Figure A.7. A grid with 5 large elements is divided into 30 sub-elements, 6 for each large element. The sub-elements are marked with dotted lines. The procedure for this refinement is identical to the refinement for quadratic PS-splines, which is repeated here for convenience:

1. Choose an interior point Z_j in each triangle t_j, such that if two triangles t_j and t_m have a common edge, the line between Z_j and Z_m intersects the edge. The intersection point is called Z_{jm}. A way to guarantee the existence of the point Z_{jm} is by choosing all Z_j as the incenter of each triangle t_j, the intersection point of the angle bisectors. Different choices are also possible, but throughout this thesis, the incenters are used an interior points.

2. Connect all Z_j to the vertices of its triangle t_j with straight lines.

3. Finally connect the Z_j to all points Z_{jm} on the edges of triangle t_j with straight lines. In case t_j is a boundary element, t_j will have at least one edge without a neighbouring element, and therefore no point Z_{jm} on this edge to connect Z_j with. In that case, join Z_j with an arbitrary point on that edge. Throughout this thesis, Z_j is connected to the middle of the edge.

The idea behind the construction of each basis functions is that it is a piece-wise cubic polynomial, which consists of a cubic polynomial on each sub-triangle that smoothly connect to the cubic polynomials of the neighbouring sub-triangles. On each sub-triangle, a cubic polynomial can be defined in barycentric coordinates $(\eta_1, \eta_2\eta_3)$ as a linear combination of 10 Bernstein polynomials:

$$p(x,y) := b(\boldsymbol{\zeta}) = \sum_{\substack{i+j+k=3,\\ i,j,k\geq 0}} b_{i,j,k} B^2_{i,j,k}(\boldsymbol{\zeta}). \tag{A.21}$$

The coefficient $b_{i,j,k}$ are known as the Bézier ordinates. This exact same construction was also used for quadratic PS-spline basis functions, except that the used Bernstein polynomials were of order 2, whereas now the Bernstein polynomials are of order 3, and 10 Bernstain polynomials are required instead of 6. Figure A.6 depicts the Bézier ordinates $b_{i,j,k}$ at barycentric coordinates $(\eta_1, \eta_2, \eta_3) = (i/3, j/3, k/3)$ in its corresponding sub-triangle. This notation will be used for defining the Bézier ordinates on a sub-triangle, and these 10 coefficients fully determine the cubic polynomial over the sub-triangle.

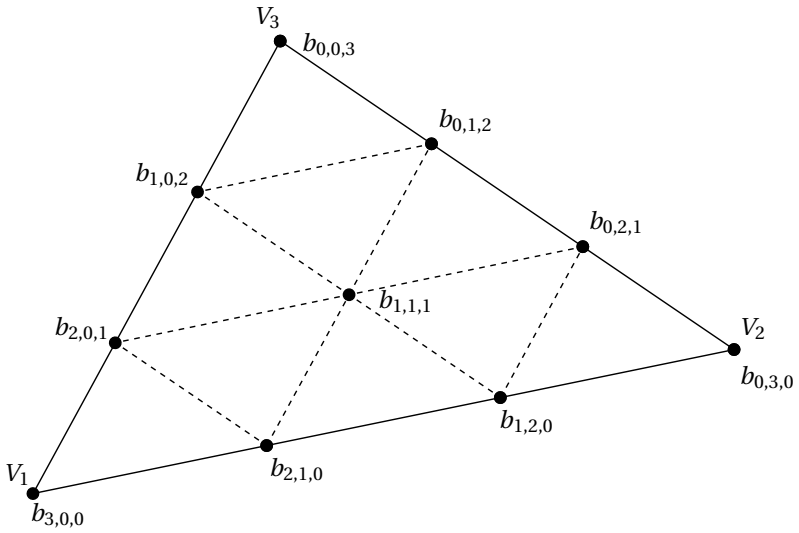


Figure A.6: A sub-triangle in which the location of the Bézier ordinates $b_{i,j,k}$ are marked corresponding to the Bernstein polynomial $B_{i,j,k}$. The location of the Bézier ordinate $b_{i,j,k}$ in barycentric coordinates is $(\eta_1, \eta_2, \eta_3) = (i/3, j/3, k/3)$.

For the construction of the cubic PS-spline basis functions, and determining the Bézier ordinates in each sub-triangle corresponding to the basis function, control triangles are used. To determine suitable control triangles, first a number of PS-points must be defined. For each main vertex V_i, we mark all points in the neighbouring sub-triangles that have barycentric coordinates $(\eta_1, \eta_2, \eta_3) = (i/3, j/3, k/3)$, $\quad i + j + k = 3, \quad 0 \leq i, j, k$. The relevant PS-points are marked in Figure A.7, which are the points in the first shell around V_i, and the points in the second shell around V_i that are not on main-element interfaces. The PS-points in the first shell around V_i are marked red in the figure, and the PS-points in the second ring are marked blue. From these points, the control triangles are defined. The first control triangle is constructed by finding a triangle containing all PS-points in the first shell around V_i, a possible triangle is shown in Figure A.7 in red. In the same figure, the blue control triangles are constructed by considering the two sub-triangles in each element closest to V_i. These two sub-triangles are separated by the interface $V_i - Z_k$, the line between a main vertex and an element center vertex. These two sub-triangles together contain 1 inner shell PS-point (marked red) and 3 outer shell-PS points (marked blue). A control triangle is now created by setting one of the control triangle vertices in the inner shell point, and varying the other two control triangle vertices such that all 3 outer shell-PS points are also contained in the control triangle. This concludes the construction of the control triangles.

The red control triangle represents three cubic PS-spline basis functions, and each blue control triangle represent two cubic PS-spline basis functions, one for each control triangle vertex that had to be chosen. Therefore, there are three basis functions per vertex V_i, similar to quadratic PS B-splines, but also one additional cubic PS B-spline per sub-triangle. Finally, there are also two additional basis function at each boundary edge of a boundary element.

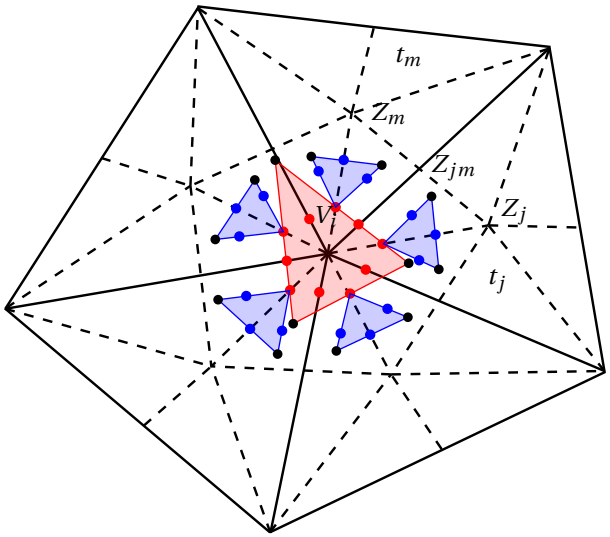
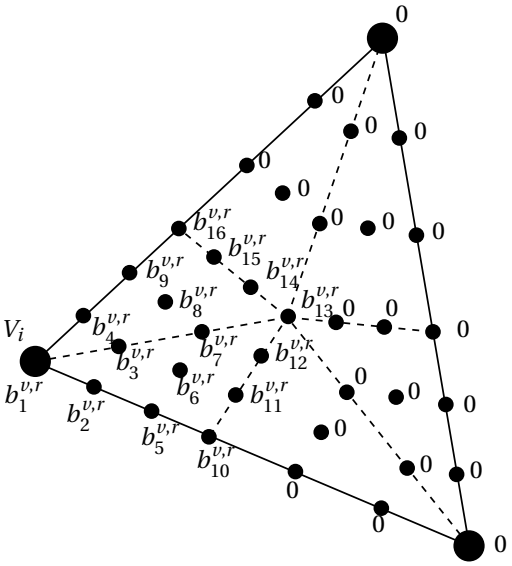
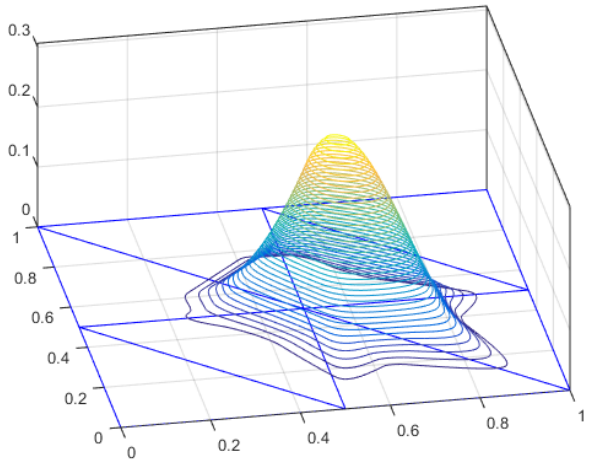


Figure A.7: A PS refinement of a triangulation for cubic splines. Each element is divided into six sub-elements, marked with the dotted lines. For this refinement, a possible choice for control triangles is indicated with red and blue. The black dots mark the control triangle vertices that need to be chosen. Each black dot may be associated with a basis function.

For all three types of basis functions, the non-zero Bézier ordinates are marked in an example element, see Figures A.8a, A.9a and A.10a for the vertex-, sub-triangle- and edge basis functions respectively. In the same Figures, an example is also shown for each type of basis functions. The exact values of the Bézier ordinates for each type of basis function are fully determined by the control triangles. The process for constructing the Bézier ordinates is omitted here, as the main purpose of this section is to show the possibility for cubic PS-spline MPM, but not the exact implementation. For the exact process of determining the Bézier ordinates from the control triangles, we refer to Grošelj and Speleers [18]. More importantly, note that the support of a vertex basis function can be up to an entire molecule around a vertex. For a sub-triangle basis function, the support is up to a full element, and two sub-triangles of a neighbouring element. For the edge basis function, the support is only the two sub-triangles at the boundary of the element on which the edge basis function is defined. Altogether, each element has three control triangles at its vertices, so nine non-zero vertex basis functions. Furthermore, on each element there are its six non-zero sub-triangle basis functions. Finally, each sub-triangle in the element also has two non-zero basis functions, either two edge basis function, or two sub-triangle basis function from the main element neighbouring the sub-triangle. This structure in combination with the non-zero Bézier ordinates can be used for efficiently storing the non-zero basis function on each (sub-)element.

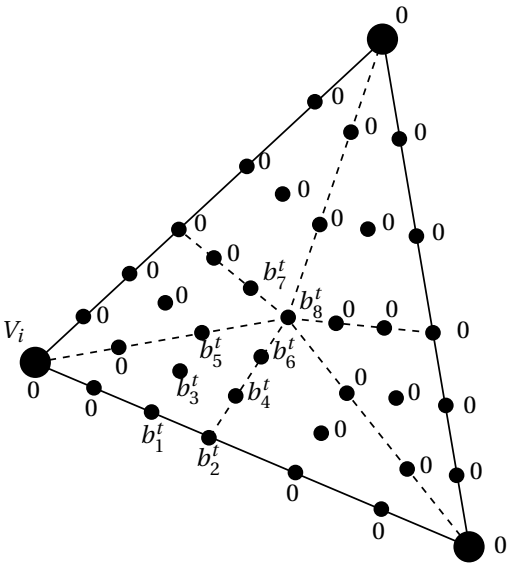


(a) The Bézier ordinates in an element for a vertex basis function.

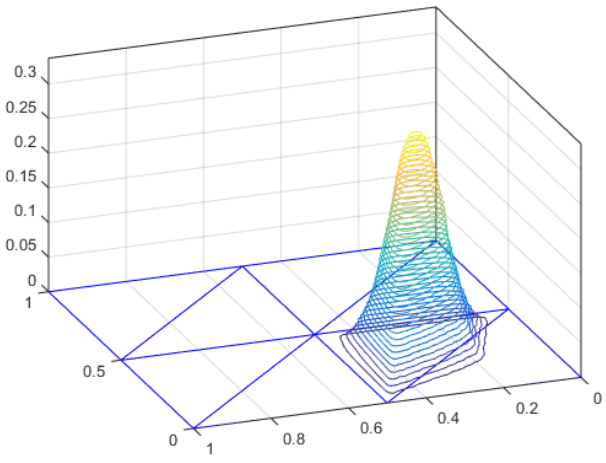


(b) One of the vertex basis functions.

Figure A.8: Construction of a vertex basis function on a triangular grid. The vertex V_i in the left figure would correspond to the middle vertex in the grid in the right plot.

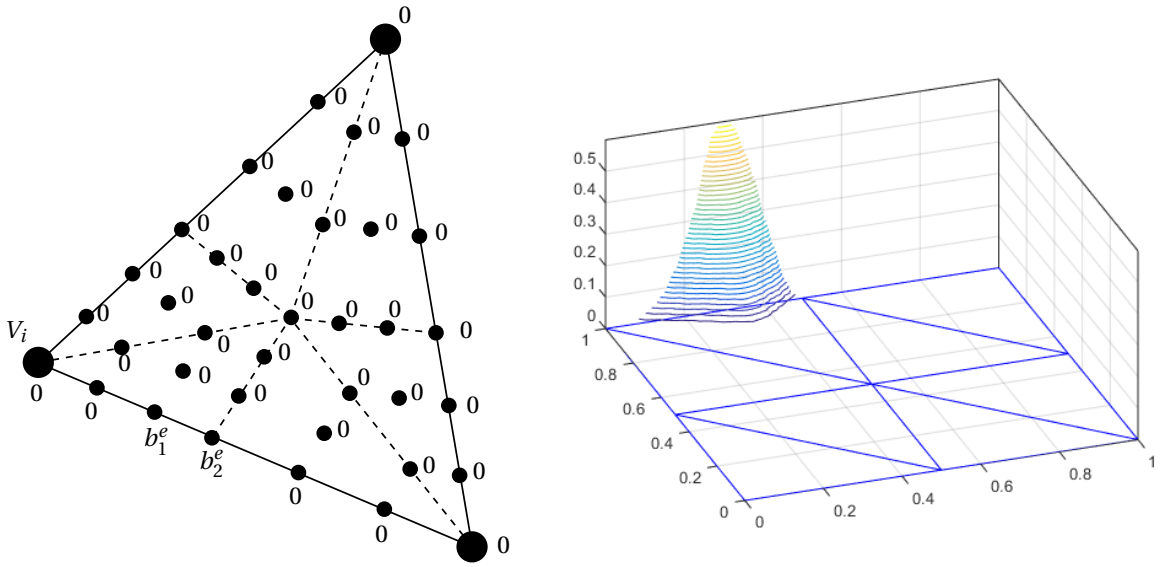


(a) The Bézier ordinates in an element for a sub-triangle basis function.



(b) One of the sub-triangle basis functions.

Figure A.9: Construction of sub-triangle basis function on a triangular grid. The vertex V_i in the left figure would correspond to the bottom middle vertex in the grid in the right plot.



(a) The Bézier ordinates in an element for an edge basis function.

(b) One of the edge basis functions on a triangular grid.

Figure A.10: Construction of boundary edge basis function on a triangular grid. The vertex V_i in the left figure would correspond to the middle vertex at the top of the grid in the right plot.

BIBLIOGRAPHY

[1] I. K. a. Kafaji, *Formulation of a dynamic material point method (MPM) for geomechanical problems* (2013).

[2] M. Gong, *Improving the Material Point Method* (The University of New Mexico, 2015).

[3] D. Sulsky, Z. Chen, and H. L. Schreyer, *A particle method for history-dependent materials,* Computer methods in applied mechanics and engineering **118**, 179 (1994).

[4] S. Andersen and L. Andersen, *Analysis of spatial interpolation in the material-point method,* Computers & structures **88**, 506 (2010).

[5] R. Tielen, E. Wobbes, M. Möller, and L. Beuth, *A high order material point method,* Procedia Engineering **175**, 265 (2017).

[6] M. Steffen, R. M. Kirby, and M. Berzins, *Analysis and reduction of quadrature errors in the material point method (mpm),* International journal for numerical methods in engineering **76**, 922 (2008).

[7] D. Sulsky, S.-J. Zhou, and H. L. Schreyer, *Application of a particle-in-cell method to solid mechanics,* Computer physics communications **87**, 236 (1995).

[8] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodríguez-Ferran, *The Encyclopedia of Computational Mechanics.*

[9] W. Prager, *Introduction to mechanics of continua* (Courier Corporation, 1961).

[10] J. van Kan, A. Segal, and F. Vermolen, *Numerical methods in Scientific Computing* (Delft Academic Press, 2014).

[11] A. Cromer, *Stable solutions using the euler approximation,* American Journal of Physics **49**, 455 (1981).

[12] R. Courant, K. Friedrichs, and H. Lewy, *Über die partiellen differenzengleichungen der mathematischen physik,* Mathematische annalen **100**, 32 (1928).

[13] B. Delaunay, *Sur la sphere vide,* Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk **7**, 1 (1934).

[14] H. Speleers, C. Manni, F. Pelosi, and M. L. Sampoli, *Isogeometric analysis with powell–sabin splines for advection–diffusion–reaction problems,* Computer methods in applied mechanics and engineering **221**, 132 (2012).

[15] T. J. Hughes, J. A. Cottrell, and Y. Bazilevs, *Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement,* Computer methods in applied mechanics and engineering **194**, 4135 (2005).

[16] P. Dierckx, *On calculating normalized powell-sabin b-splines,* Computer Aided Geometric Design **15**, 61 (1997).

[17] P. Dierckx, S. Van Leemput, and T. Vermeire, *Algorithms for surface fitting using powell-sabin splines,* IMA Journal of numerical analysis **12**, 271 (1992).

[18] J. Grošelj and H. Speleers, *Construction and analysis of cubic powell–sabin b-splines,* Computer Aided Geometric Design **57**, 1 (2017).

[19] R. Tielen, *High-order material point method,* (2016).

[20] M. Mieremet, *Numerical stability for velocity-based 2-phase formulation for geotechnical dynamic analysis,* MMJ (2015).

[21] J.-p. Zeng and H.-x. Yu, *Error estimates of the lumped mass finite element method for semilinear elliptic problems,* Journal of Computational and Applied Mathematics **236**, 1993 (2012).

[22] E. WOBBES, M. MOLLER, V. Galavi, and C. Vuik, *Taylor least squares reconstruction technique for material point methods,* .

[23] L. E. Malvern, *Introduction to the Mechanics of a Continuous Medium*, Monograph (1969).

[24] M. Geers, *Fundamenttals of Deformation and Linear Elasticity* (Lecture notes, TU Eindhoven, 2001).

Cover picture retreived from \https://commons.wikimedia.org/wiki/File:Windmills_D1-D4_(Thornton_Bank).jpg, by Hans Hillewaert