Extending 1D B-spline basis MPM to higher dimensional problems

Literature study

Pascal B.J. de Koster
4302508

The Netherlands, Delft, March 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

Supervisor TU Delft: Dr. M. Möller
Supervisor Deltares: Dr. V. Galavi
Thesis Committee:

ABSTRACT

The material point method (MPM) is a simulation method for ground deformations that combines the benefits of a fixed grid and moving material points. The material points carry the history dependent information of the grid, this information is sent to the grid, on which the equations of motion are solved. Then, the material points undergo the corresponding motion and the particle properties are updated. MPM uses a finite element method to solve the equations of motion over a 2D or 3D grid, but in current versions, the basis functions used are only piecewise linear, which results in a low order for the spatial convergence. This study investigates the possibility to use higher order basis functions in the material point method. For physical and numerical reasons, it is desirable that these basis function are non-negative and C^1-continuous. B-spline basis functions would be very suitable for this purpose and earlier studies for 1D have already showed the potential of B-spline MPM. In this study, a 2D and 3D version of B-spline MPM is investigated, in which the grid is triangular. It turns out that B-spline basis functions are cumbersome to construct for triangulations, and so far only 2D piecewise parabolic basis functions have been considered. These piecewise parabolic basis function show higher order spatial convergence for function reconstruction, which also is required for MPM. The spatial convergence of MPM with these functions is yet to be tested.

CONTENTS

1

INTRODUCTION

In geomechanics, problems arise with large soil deformations. The final 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 finite element method, however this technique has great difficulty dealing with large deformations and history dependent material. In order to capture the history dependent properties, a Lagrangian version of the finite element approach is suitable. In this method, the grid moves along the with the material, which makes it easy to track its properties. However, large deformations will often lead to mesh distortion, resulting in large numerical errors or non-physical solutions. On the other hand, when using a fixed mesh 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 material points with a fixed background mesh. As a fixed mesh 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. Finally, the information is projected back onto the material points, which undergo the corresponding motion. In 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 in terms of a linear combination of basis functions, which are defined in correspondence with the grid.

In order to find accurate solutions for two and three 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 take any arbitrary geometry without the need of mappings, as is common in rectangular grids. Therefore, the grids used in MPM are mostly triangular. The typical choice for basis functions over these triangulations is the set of piecewise linear Lagrangian basis functions. However, this basis is only second order spatial convergent, which means that a fine mesh is required to get a reliable solution. This again results in long computation times. Problems in MPM can have thousands up to millions of degrees of freedom, so computation time and accuracy become a serious problem.

This thesis investigates if higher order basis functions on triangular grids can mitigate this problem. In previous research, the use of higher order Lagrangian basis functions in MPM has been researched for multi-dimensional problems, but a higher order Lagrangian basis turns out to be unsuitable, due to negativity of the basis functions. It is preferable to use non-negative basis functions for 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]. Furthermore, negative entries prevent the mass matrix from being lumped, whereas lumping is most 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. Therefore, the Lagrangian basis functions are not suitable.

Instead of using a Lagrangian basis, this thesis studies the use of a B-spline basis on arbitrary triangulations. These functions are not as straightforward to implementat as the Lagrangian basis, but do have the

property of non-negativity. Earlier studies on B-spline basis for one-dimensional MPM problems show great potential, see [5, 6]. The use of a B-spline basis was also reported to mitigate the effect of grid crossing, which happens when material points move from one element to a neighbouring one. When using the Lagrangian basis, grid crossing errors appear due to discontinuities of the basis functions over the edges of the elements. In most versions of MPM, this error can be mitigated, for example as described by Kafaji [1]. A B-spline basis has the advantage that grid-crossing errors are not expected to occur at all.

The goal of this thesis is to extend the existing one-dimensional MPM with B-spline basis of Tielen et al. [5] to higher dimensions on triangular grids. This research is executed in cooperation with Deltares, a research institute that investigates geomechanical problems with MPM. These problems include underwater slope failure and pile driving. The current MPM version of Deltares uses piecewise linear lagrangian basis functions on triangular grids. This version includes a mitigation for grid-crossing error. A B-spline based MPM could lead to a more accurate and faster method.

In order to compare Lagrangian and B-spline MPM, several benchmark problems will be studied. First, 2D B-spline MPM will be validated by solving one-dimensional problems, and comparing these to their analytic solution. The final goal is to use B-spline MPM will to simulate pile driving and compare B-spline MPM to the method of Deltares. This has yet been done in this literature study though.

The outline of this study is as follows. Chapter 2 start with introducing the equations of motion for continuum mechanics. Then, Chapter 3 continues with discretising the equations of motion, and the material point method for solving these equation is explained. In Chapter 4, B-splines are explained in one-dimension and then extended to higher dimensions on triangular grids. Finally Chapter 5 will compare Lagrangian basis MPM with B-spline MPM.

2

MATHEMATICAL MODEL FOR CONTINUUM MECHANICS

In this chapter, the equations of motion for continuum mechanics will be discussed. These equations of motion are to be solved with the material point method, which is explained in Chapter 3. The equations of motion are presented in such a form such that the are suitable for MPM. For this, first the frame of reference is discussed, because MPM uses a combination between a stationary and moving frame. In the corresponding frame of reference, the conservation of momentum will be used to derive the equations of motion.

2.1. 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 Lagrangian 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. Depending on the frame of reference, the equations of motions will be described differently.

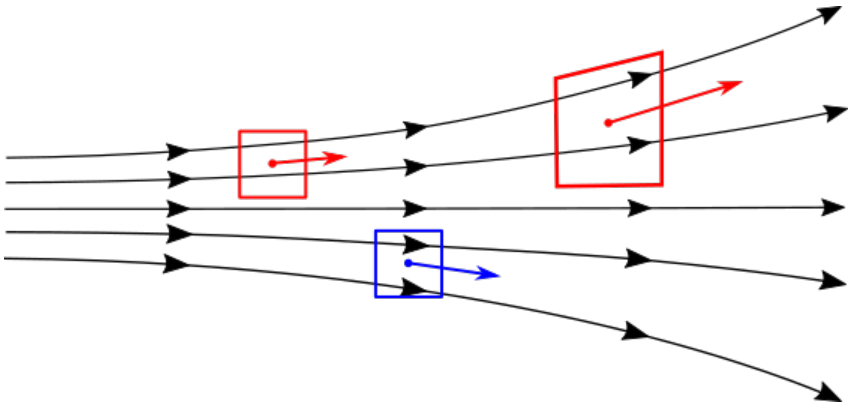


Figure 2.1: Illustration of the Lagrangian (red) and Eulerian (blue) frame of reference in 2D. The control volumes in the Eulerian frame of reference are stationary and material can go in and out. The Lagrangian frame of reference always contains the same material and moves along with it. The shape of the control volume will therefore also change through time.

2.2. Material Derivative

As mentioned in the introduction, the material point method makes use of material points, that move along with the continuum. Therefore the Lagrangian frame of reference will be used. In order to describe the equations of motion of the continuum, differential equations for the displacement, velocity and momentum will be considered. To describe time derivatives for a continuum, the material derivative should be used,

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

In this equation, f describes an arbitrary function, t the time, $\vec{x}$ the position, $\vec{v}$ the velocity and $\nabla\phi$ the gradient of f. Mathematically, the material derivative is the chain rule applied to a function dependent on both time and space, considering that $\boldsymbol{v} = \frac{\partial \boldsymbol{x}}{\partial t}$. Physically, the material derivative can be interpreted to consist of two parts, a partial derivative to the time where all other variables are held constant, and an advection term. For illustration purposes, 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 to the time in Equation 2.1. Second, 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 times the gradient of the temperature in the direction of the flow. 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 flow 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 material derivative reduces to the partial derivative for all purposes in MPM [7],

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

Next, the governing equation ofo motion will be derived in the Lagrangian frame of reference, using the material derivative of Equation 2.2.

2.3. Equations of motion

For most problems in continuum mechanics, the equation of mass conservation would be considered first. However, due to the Lagrangian frame of reference, the control volumes will always contain the same particles. Therefore, mass conservation is already present in MPM and no equations are required for this.

The motion of the continuum is derived from conservation of linear momentum, which corresponds with Newtons second law, $\boldsymbol{F} = m\cdot\boldsymbol{a}$. The equation for momentum equation for a continuum in a Lagrangian frame is given by [8]

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

In this equation, ρ represents the density, $\boldsymbol{\sigma}$ the stress tensor and $\boldsymbol{g}$ the gravitational acceleration, which is a downward vector of $9.81 m/s^2$ unless stated otherwise. In general, the left hand side of this equation is a material derivative instead of a partial derivative, but due to the Lagrangian frame, we can use the partial derivative as was shown in Equation 2.2.

The equation states that acceleration of a material happens due to internal forces, represented by the divergence of the stress tensor, and an external gravitational force. For understanding and handling this equation, it is important to understand its origin. Although the gravitational contribution is straightforward, the internal force is more complicated to understand.

2.3.1. 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 2.2. 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 the plane was put in any vertical direction, then there would be little normal stress on that plane for this example.

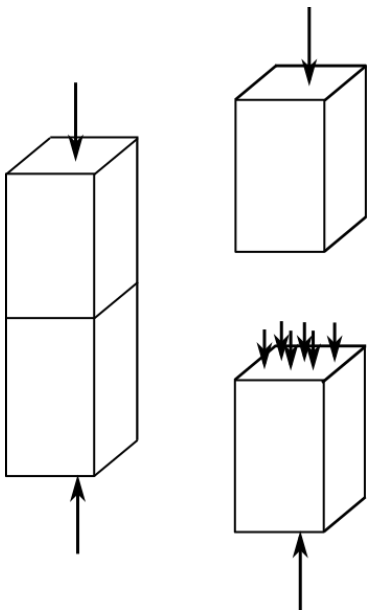


Figure 2.2: Illustration of normal stress in a compressed bar. The stress is normal to any horizontal plane in the vertical bar, and presses on the surface. A similar force field acts on the bottom of the upper half of the bar, but then pushes up instead.

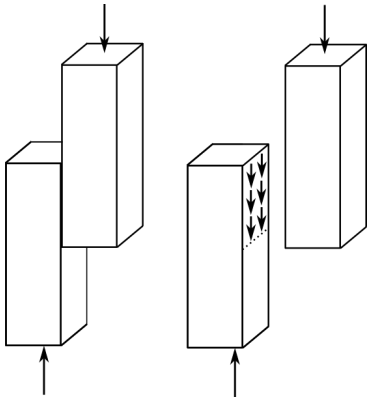


Figure 2.3: Illustration of shear stress when two bars are pressed along each other. The stress is parallel to the plane, and is therefore shear stress.

The shear stress is defined as the force per area parallel to the plane it is exerted on. An example is showed in Figure 2.3. When two blocks are horizontally next to each other, and the right one is pushed down and the left one pushed up, there will be shear stress over the interface. This stress has two component, in the figure, the component in the z-direction is non-zero, but the shear stress in the y-direction is nihil.

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 a infinitely small cube around that point. The stress on opposite planes is equal, as the cube is infinitely small. For this, consider three orthogonal planes, the xy-, the yz and the xz-plane, as is illustrated in Figure 2.4. 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 2.4. 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 in the figure is in steady state, then $\sigma_{ij} = \sigma_{ji}$ and so the stress tensor should be symmetric.

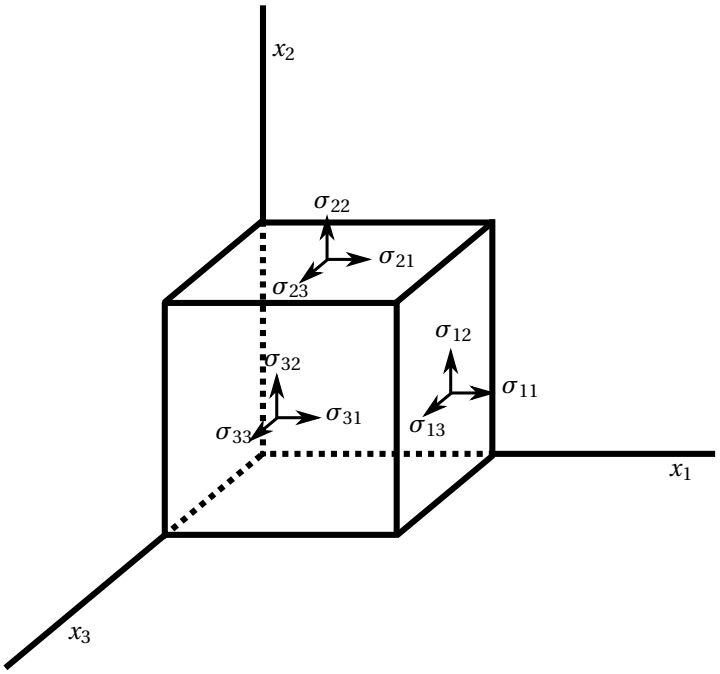


Figure 2.4: The stress components in a single point. The stress depends on the orientation of the plane it is working on. A 3×3 stress tensor describes the full stress in a point.

2.3.2. FROM STRESS TO MOTION

In order to go from the stress within a continuum to the resulting total force, a small volume can be considered. For simplicity, a 2D volume will be considered and the resulting total force on this volume will be calculated. The derivation for a 3D volume is similar. Figure 2.5 shows a square with four 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

$$\begin{aligned} F_x =& \Delta y \left[\sigma_{11}\left(x+\tfrac{\Delta x}{2}, y\right) - \sigma_{11}\left(x-\tfrac{\Delta x}{2}, y\right)\right] + \Delta x \left[\sigma_{21}\left(x, y+\tfrac{\Delta y}{2}\right) - \sigma_{21}\left(x, y-\tfrac{\Delta y}{2}\right)\right] \\ \Rightarrow \frac{F_x}{\Delta x \Delta y} =& \frac{\sigma_{11}\left(x+\tfrac{\Delta x}{2}, y\right) - \sigma_{11}\left(x-\tfrac{\Delta x}{2}, y\right)}{\Delta x} + \frac{\sigma_{21}\left(x, y+\tfrac{\Delta y}{2}\right) - \sigma_{21}\left(x, y-\tfrac{\Delta y}{2}\right)}{\Delta y} \\ \xrightarrow{\Delta x, \Delta y \to 0} f_x(x,y) =& \frac{\partial \sigma_{11}}{\partial x}(x,y) + \frac{\partial \sigma_{21}}{\partial y}(x,y) \end{aligned} \tag{2.4}$$

Note that $\boldsymbol{f}$ is a force density, as corresponds with Equation 2.3. Equation 2.4 can be generalised for any direction. Rename x as x_1 and y as x_2 etcetera. Using the Einstein summation convention, the internal force can be written as [8]

$$f_i = \frac{\partial \sigma_{ji}}{\partial x_j}. \tag{2.5}$$

This hold for 2D as well as for 3D. In divergence notation, this becomes

$$\boldsymbol{f} = (f_1, f_2, f_3) = \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3}\right) \cdot \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} = \nabla \cdot \boldsymbol{\sigma}. \tag{2.6}$$

Note that $\nabla \cdot \boldsymbol{\sigma}$ is a 1 by n vector, so in order to be in correspondence with Equation 2.3, all vectors should be denoted in a similar fashion. It is also possible to first calculate $\nabla \cdot \boldsymbol{\sigma}$, transpose it and then put this in Equation 2.3, in case it is preferable to work with n by 1 vectors.

Adding up both the internal force due to stress and the external force due to gravity, the momentum conservation of Equation 2.3 is derived.

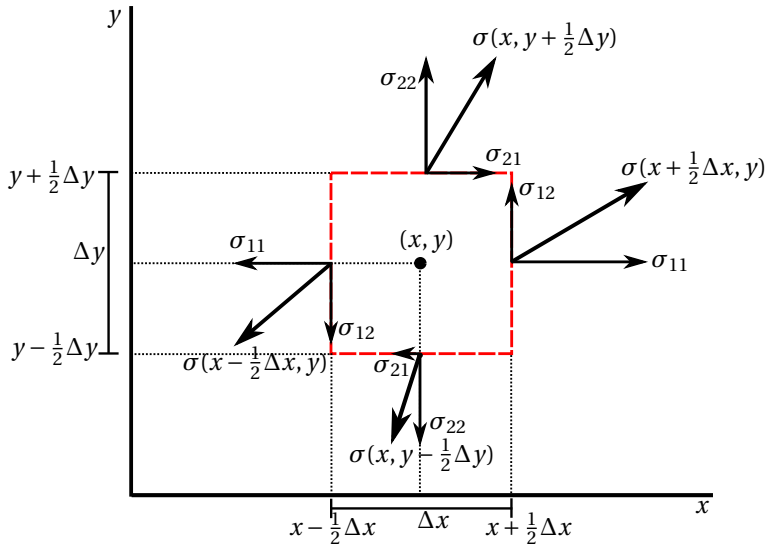


Figure 2.5: A 2D volume of size Δx by Δy. On each of the four faces of the cube, the stress is assumed to be constant. The stress vector working on each of the faces is depicted, decomposed in its x and y components.

2.4. Displacement, strain and stress

In Section 2.3, the equations of motion for a continuum were derived. Equation 2.3 showed that the momentum conservation related the acceleration to the stress of the material. Given the stress in the material, it is possible to find the corresponding acceleration and thus the displacement as well. To make the circle round, it is required to find the new stress from the updated displacement. In order to go from the displacement to the stress, the strain in the material is of interest. The idea is that displacement from the original geometry causes deformation of the material, which is strain. Then a force opposing the deformation tries to get the material back in its original geometry. This force is the stress. The idea is similar to that of a spring being stretched or compressed, though some other effects should also be taken into account. First the strain shall be derived from the deformation of the material, second a constitutive relation will be used to determine the stress from the strain.

2.4.1. Strain

The theory about strain in this part was retrieved from [9] and [1], 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,

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{11} & \frac{1}{2}\gamma_{12} & \frac{1}{2}\gamma_{13} \\ \frac{1}{2}\gamma_{21} & \epsilon_{22} & \frac{1}{2}\gamma_{23} \\ \frac{1}{2}\gamma_{31} & \frac{1}{2}\gamma_{32} & \epsilon_{33} \end{bmatrix}. \tag{2.7}$$

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 $\boldsymbol{u}(\boldsymbol{x})$. To derive the strain, consider a infinitesimal small rectangle of size $\Delta x \times \Delta y$. This rectangle may move and deform over time. Figure 2.6 shows the situation. If the stretching of AB is considered, the length after deformation of the line piece is equal to

$$ab = \sqrt{\left(\Delta x + \frac{\partial u_x}{\partial x}\Delta x\right)^2 + \left(\frac{\partial u_y}{\partial x}\Delta x\right)^2} \tag{2.8}$$

$$= \Delta x \sqrt{1 + 2\frac{\partial u_x}{\partial x} + \left(\frac{\partial u_x}{\partial x}\right)^2 + \left(\frac{\partial u_y}{\partial x}\right)^2}. \tag{2.9}$$

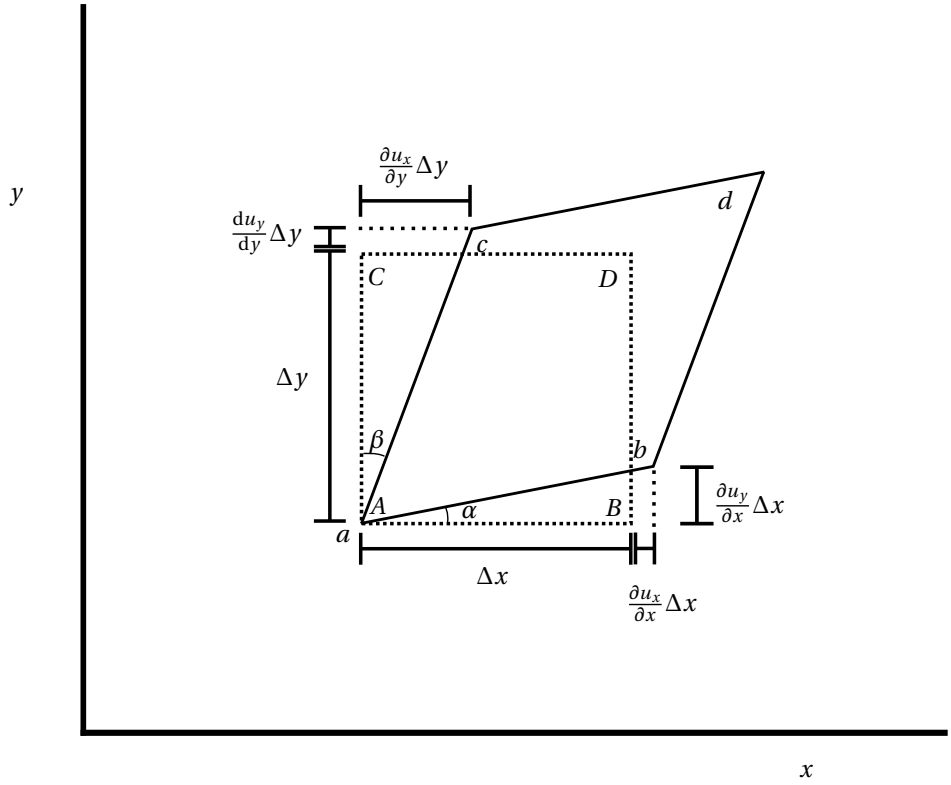


Figure 2.6: A 2D volume element undergoing infinitely small deformation. The deformation in the illustration is exaggerated for the sake of clarity. The dotted rectangle is the original geometry, and the solid quadrilateral is the deformed cube.

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 squareroot can be approximated,

$$ab = \Delta x \sqrt{1 + 2\frac{\partial u_x}{\partial x}} \tag{2.10}$$

$$= \Delta x + \frac{\partial u_x}{\partial x}\Delta x \tag{2.11}$$

Therefore, linepiece 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

$$\epsilon_{ii} = \frac{\partial u_i}{\partial x_i}. \tag{2.12}$$

The shear strain is related to the change in angle between linepieces AB and AC in the original configuration. This change in angle is equal to $\alpha + \beta$ in Figure 2.6. The formulas for α and β are derived from geometric

rules, under the assumption of small displacements and small angles.

$$\tan(\alpha) = \frac{\frac{\partial u_y}{\partial x}\Delta x}{\Delta x + \frac{\partial u_x}{\partial x}\Delta x} = \frac{\frac{\partial u_y}{\partial x}}{1 + \frac{\partial u_x}{\partial x}} \tag{2.13}$$

$$\xrightarrow{\tan\theta \approx \theta,\ \frac{\partial u_x}{\partial x} \ll 1} \alpha = \frac{\partial u_y}{\partial x} \tag{2.14}$$

$$\tan(\beta) = \frac{\frac{\partial u_x}{\partial y}}{1 + \frac{\partial u_y}{\partial y}\Delta y} \tag{2.15}$$

$$\xrightarrow{\tan\theta \approx \theta,\ \frac{\partial v}{\partial y} \ll 1} \beta = \frac{\partial u_x}{\partial y} \tag{2.16}$$

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

$$\gamma_{ij} = \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{2.17}$$

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

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{2.18}$$

This form is in agreement with Equations 2.12 and 2.17.

For MPM the strain is required in differential form. Each time step, the total strain will change slightly. Therefore, the derivative of the strain to the time is needed. The differential form is easily retrieved from Equation 2.18, 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

$$\frac{\partial \epsilon_{ij}}{\partial t} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \tag{2.19}$$

In the next section, the relation between stress and strain is explained.

2.4.2. Constitutive relation

To determine the stress from the strain, a so called constitutive relation is required. The constitutive relation used here was retrieved from [10] and [1]. 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

$$\frac{\partial \sigma_{ij}}{\partial t} = \overset{\nabla j}{\sigma}_{ij} + \omega_{ik}\sigma_{kj} - \sigma_{ik}\omega_{kj}, \tag{2.20}$$

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 result in the rotational acceleration. The spin tensor ω_{ij} is defined as the antisymmetric part of the velocity gradient,

$$\omega_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i}\right). \tag{2.21}$$

The Jaumann rate of stress is defined as

$$\overset{\nabla J}{\sigma}_{ij} = \overset{\nabla H}{\sigma}_{ij} + \frac{\partial \epsilon_{kk}}{\partial t}\sigma_{ij}, \tag{2.22}$$

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. This may be interpreted that compression or stretching causes a force, 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

$$\overset{\nabla H}{\sigma}_{ij} = D_{ijkl}\frac{\partial \epsilon}{\partial t}. \tag{2.23}$$

Substitute above equations into Equation 2.20 to get the final constitutive relation,

$$\frac{\partial \sigma_{ij}}{\partial t} = D_{ijkl}\frac{\partial \epsilon_{kl}}{\partial t} - \frac{\partial \epsilon_{kk}}{\partial t}\sigma_{ij} + \omega_{ik}\sigma_{kj} - \sigma_{ik}\omega_{kj}, \tag{2.24}$$

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

$$D_{ijkl} = \left(K - \frac{2}{3}G\right)\delta_{ij}\delta_{kl} + G\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right). \tag{2.25}$$

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 Poisson ratio ν and the Young's modulus E, which are material properties:

$$K = \frac{E}{3(1-2\nu)} \quad \text{and} \quad G = \frac{E}{2(1+\nu)}. \tag{2.26}$$

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 2.24. Finally this constitutive relation will be used to determine the stress from the strain.

2.5. Initial and Boundary Conditions

With the conclusion of the last section, the equations of motion are complete: displacement causes strain, strain causes stress, stress causes acceleration, and acceleration leads back to displacement. In order to fully define a problem, initial and boundary conditions are required as well.

Let the domain of interest be Ω, and its boundary $\partial\Omega$, see Figure 2.7. 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,

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{U}(t) \quad \text{on} \quad \partial\Omega_u(t). \tag{2.27}$$

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

$$\sigma_{ij}(\boldsymbol{x}, t)n_j = \tau_i(t) \quad \text{on} \quad \partial\Omega_\tau(t), \tag{2.28}$$

where n_j denotes the j^{th} component of the unit vector, normal to the boundary pointing outwards. Prescribed traction corresponds with a Neumann boundary condition. 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. Robin boundary conditions are not considered in this thesis, as they are not used in the benchmarks researched later on.

Initial conditions are required for each of the equations described in differential form. Therefore, at least the displacement and velocity require initial conditions. Furthermore, the stress is also used in differential form, so here another initial condition is required. The acceleration can be directly determined from the stress, and the strain directly from the stress, velocity and the displacement, to for these no initial condition is required. The initial condition are thus given by

$$\boldsymbol{u}(\boldsymbol{x}, t_0) = \boldsymbol{U_0}, \quad \boldsymbol{v}(\boldsymbol{x}, t_0) = \boldsymbol{V_0}, \quad \text{and} \quad \sigma_{ij}(\boldsymbol{x}, t_0) = \sigma_{0ij}. \tag{2.29}$$

Together, the equations of motion, the boundary conditions and the initial conditions can describe a problem for a continuum.

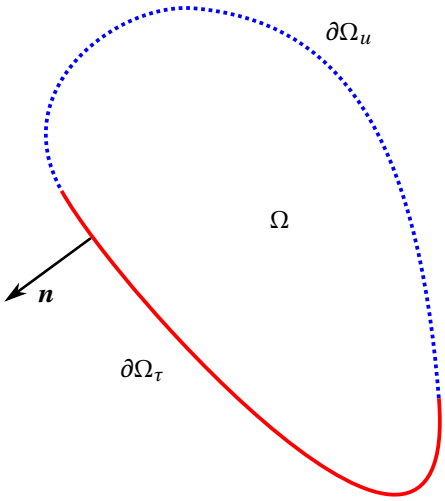


Figure 2.7: A domain Ω with its boundary $\partial\Omega$. The boundary is divided in two parts. On the dotted part $\partial\Omega_u$, the displacement is prescribed, and on the solid part $\partial\Omega_\tau$, the traction is prescribed. $\boldsymbol{n}$ marks an outwards pointing normal vector.

2.6. Weak formulation

In order to simulate the motion, the conservation of momentum in Equation 2.3 should be solved numerically. The material point method uses a finite element method to do so, which requires the weak form of the differential equation. Instead of demanding that the left hand side is exactly equal to the right hand side in the differential equation, the finite element method only requires that the left hand side and right hand side should be equal when both sides are multiplied with an arbitrary test function ϕ from the test space Φ, and then integrated over the domain.

$$\begin{aligned} \rho\boldsymbol{a} &= \nabla\cdot\boldsymbol{\sigma} + \rho\boldsymbol{g} \\ \Rightarrow \int_\Omega \phi\rho\boldsymbol{a}\,\mathrm{d}\Omega &= \int_\Omega \phi\nabla\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega + \int_\Omega \phi\rho\boldsymbol{g}\,\mathrm{d}\Omega \end{aligned} \tag{2.30}$$

Obviously, a solution to the first equation will also be a solution to the weak form. After all, when the first solution holds, then multiplying both sides with an arbitrary test function will still yield the same on both sides, and integrating will not change this either. The other way around is not necessarily true. Consider multiplying with the test function 0, then both sides will always be 0, independent of the stress, velocity and density. Therefore, the formulation in Equation 2.30 is weaker than the original equation. However, when Equation 2.30 holds for every ϕ in Φ, and Φ is sufficiently large, then the solution to the weak form and the original differential equation will be the same. Now 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 [11]. In case that ϕ meets these conditions, then Equation 2.30 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,

$$\begin{aligned} \int_\Omega \phi\nabla\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega &= \int_\Omega \nabla\cdot(\phi\boldsymbol{\sigma})\,\mathrm{d}\Omega - \int_\Omega \nabla\phi\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega \\ &= \int_{\partial\Omega} \phi\boldsymbol{n}\cdot\boldsymbol{\sigma}\,\mathrm{d}\Gamma - \int_\Omega \nabla\phi\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega \\ &= \int_{\partial\Omega_u} \cancelto{0}{\phi}\,\boldsymbol{n}\cdot\boldsymbol{\sigma}\,\mathrm{d}\Gamma + \int_{\partial\Omega_\tau} \phi\boldsymbol{n}\cdot\boldsymbol{\sigma}\,\mathrm{d}\Gamma - \int_\Omega \nabla\phi\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega \\ \Rightarrow \int_\Omega \phi\nabla\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega &= \int_{\partial\Omega_\tau} \phi\boldsymbol{n}\cdot\boldsymbol{\sigma}\,\mathrm{d}\Gamma - \int_\Omega \nabla\phi\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega. \end{aligned} \tag{2.31}$$

When splitting the boundary integral into its part over the Dirichlet and over the Neumann boundary parts, the test function was defined such that it is zero on the Dirichlet part. Therefore, this part can be crossed out

and 2.31 remains. Plug this back into the weak form in Equation 2.30 and this yields the final form of the weak equation. The full problem formulation then becomes

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

$$\int_\Omega \phi \rho \boldsymbol{a} \, \mathrm{d}\Omega = \int_{\partial\Omega_\tau} \phi \boldsymbol{n} \cdot \boldsymbol{\sigma} \, \mathrm{d}\Gamma - \int_\Omega \nabla\phi \cdot \boldsymbol{\sigma} \, \mathrm{d}\Omega + \int_\Omega \phi \rho \boldsymbol{g} \, \mathrm{d}\Omega \tag{2.32}$$

for all $\phi \in \Phi$.

The space $\mathscr{A}$ represents the space of sufficiently smooth functions that respect the essential boundary conditions. This space will later on be specified for MPM. Note that for this equation, all terms on the right hand side are assumed to be known, and only the acceleration needs to be solved. In MPM, this equation will used to solve for the acceleration $\boldsymbol{a}$ at each time step.

3

MATERIAL POINT METHOD

In this chapter, the material point method will be discussed in detail. There are many versions of MPM, but in this thesis the method of Al-Kafaji [1] is followed. This is also the method used by Tielen [5], on whose work this thesis is building, and Deltares follows this procedure as well.

The material point method is explained step by step, with the focus on using higher order basis functions in the space discretisation. Tielen et al. [5] already succeeded to create a high order MPM, but only for 1D, so special attention will also be paid to the differences between 1D and higher dimensional MPM.

First the basic concept of MPM will be explained in Section 3.1, then in Section 3.2 the spatial discetisation is discussed. Section 3.3 finally presents the full algorithm for the time stepping procedure.

3.1. PARTICLE IN GRID METHOD

The material point method is a numerical method to solve the equations of motion for a deforming continuum. MPM makes use of a particle in grid method: there is a stationary background grid with moving material points inside. The vertices of the background grid will often be referred to as *nodes*, and the material points may also be called *particles*. The equations of motion as described in the previous chapter are discretised over time and space and then solved. In order to store the material data, MPM makes use of material points, where all the properties of the material such as stress, velocity and density are stored. 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. This way, 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 2.32 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 **??** illustrates the concept of the particle in grid method of MPM. Because the equations of motion are solved at the stationary background grid, a Lagrangian frame of reference is used. On this frame, no non-linear convective terms appear in the equations of motion, which would have appeared when the equations were solved in the particles directly. Combining material point with a stationary background grid gives the advantage that the history of particles can be tracked easily but at the same time the non-linear convective terms are avoided.

3.2. SPACE DISCRETISATION

In order to solve the momentum equation, it must be discretised on the background mesh. The grids considered in this thesis are triangular. Triangular grid 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 map 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. Also it may be quite difficult to locally refine the mesh. These problems do not appear

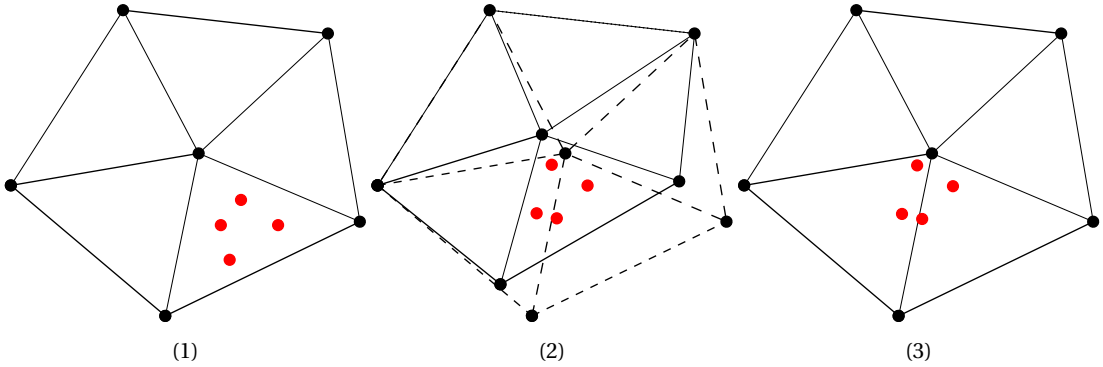


Figure 3.1: A part of a triangular MPM grid with particles inside. The concept of MPM is shown. (1): The particles properties are projected onto the grid. (2): The equations of motion are solved on the grid level. (3): The particle properties are updated and the grid is reset.

when using a triangular mesh. It is very easy to approximate an arbitrary geometry with triangles. Also the local refinement process can simply be done by adding extra nodes locally and running the Delaunay triangulation process again. For large deformation problems, a triangular grid is preferred. Therefore, this thesis studies MPM specifically with a triangular background.

Over the triangular background mesh, a set of n basis functions that span Φ is defined. For each of the basis functions ϕ_i, the weak form for the equation of motion will be solved, which is repeated here for convenience,

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

First however, the acceleration is approximated by a linear combination of basis functions,

$$\boldsymbol{a}(\boldsymbol{x}) \approx \boldsymbol{a}_h(\boldsymbol{x}) = \sum_{j=1}^{n} \hat{\boldsymbol{a}}_j\phi_j(\boldsymbol{x}), \tag{3.1}$$

in which $\boldsymbol{a}_h$ denotes the approximation of the acceleration and $\hat{\boldsymbol{a}}_j$ the coefficients of its corresponding basis function. Note that the acceleration is a vector, so the coefficients in the approximation are also vectors. For each direction, the acceleration has its own coefficients.

Now substitute the approximation $\boldsymbol{a}_h$ into the weak form above, then the equation must hold for each of the basis functions ϕ_i,

$$\int_\Omega \phi_i\rho\sum_{j=1}^{n}\hat{\boldsymbol{a}}_j\phi_j\,\mathrm{d}\Omega = \int_{\partial\Omega_\tau}\phi_i\boldsymbol{n}\cdot\boldsymbol{\sigma}\,\mathrm{d}\Gamma - \int_\Omega\nabla\phi_i\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega + \int_\Omega\phi_i\rho\boldsymbol{g}\,\mathrm{d}\Omega, \quad \text{for } i = 1\ldots n.$$

By exchanging summation and integration, this can be rewritten to

$$\sum_{j=1}^{n}\left(\int_\Omega\phi_i\rho\phi_j\,\mathrm{d}\Omega\right)\hat{\boldsymbol{a}}_j = \int_{\partial\Omega_\tau}\phi_i\boldsymbol{n}\cdot\boldsymbol{\sigma}\,\mathrm{d}\Gamma - \int_\Omega\nabla\phi_i\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega + \int_\Omega\phi_i\rho\boldsymbol{g}\,\mathrm{d}\Omega, \quad \text{for } i = 1\ldots n. \tag{3.2}$$

Now assume that an acceleration function $\boldsymbol{a}_h$ with coefficients $\hat{\boldsymbol{a}}_j$ is found, such that Equations 3.2 hold for each of the basis functions ϕ_i. Note the difference between the function $\boldsymbol{a}(\boldsymbol{x})$ and the coefficient vector $\hat{\boldsymbol{a}}$. The $\hat{}$ will be consistently used to denote a coefficient vector. 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. Therefore, $\boldsymbol{a}_h$ will be a solution for any arbitrary function $\phi \in \Phi$.

Now note that Equations 3.2 is a linear system of equations, and can be rewritten in the form

$$\boldsymbol{M}\hat{\boldsymbol{a}} = \boldsymbol{F}^{trac} + \boldsymbol{F}^{int} + \boldsymbol{F}^{grav}. \tag{3.3}$$

In this equation, $\boldsymbol{M}$ denotes the mass matrix, and $\boldsymbol{F}^{trac}$, $\boldsymbol{F}^{int}$ and $\boldsymbol{F}^{grav}$ respectively denote the traction force, the internal force and the gravitational force. The mass matrix is defined as a n by n matrix, with n the number of basis function or degrees of freedom. Its coefficients are defined as

$$\boldsymbol{M}_{ij} = \int_\Omega \phi_i\rho\phi_j\,\mathrm{d}\Omega. \tag{3.4}$$

The forces are defined as

$$\boldsymbol{F}_i^{trac} = \int_{\partial\Omega_\tau} \phi_i \boldsymbol{n} \cdot \boldsymbol{\sigma} \, \mathrm{d}\Gamma, \tag{3.5}$$

$$\boldsymbol{F}_i^{int} = \int_{\Omega} \nabla\phi_i \cdot \boldsymbol{\sigma} \, \mathrm{d}\Omega, \tag{3.6}$$

$$\boldsymbol{F}_i^{grav} = \int_{\Omega} \phi_i \rho \boldsymbol{g} \, \mathrm{d}\Omega. \tag{3.7}$$

Note that Equation 3.3 is best evaluated when each direction is considered separately, so first only consider the system of equation for the x-direction, then for the y direction, and so on. The directions are decoupled and orthogonal, so this is not a problem.

In order to calculate the integrals, the material points are used as integration points. The values for $\boldsymbol{\sigma}$ and $\boldsymbol{\rho}$ are only known in the material points, so this is the easiest way to approximate the integrals. For the weights, the volume V_p of each particle point is used. The quadrature integration rule then becomes

$$\int_{\Omega} f(\boldsymbol{x}) \, \mathrm{d}\Omega \approx \sum_{p=1}^{n_p} V_p f(\boldsymbol{x}_p), \tag{3.8}$$

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 3.3 are evaluated as

$$\boldsymbol{M}_{ij} = \sum_{p=1}^{n_p} V_p \phi_i(\boldsymbol{x}_p) \rho_p \phi_j(\boldsymbol{x}_p) = \sum_{p=1}^{n_p} m_p \phi_i(\boldsymbol{x}_p) \phi_j(\boldsymbol{x}_p), \tag{3.9}$$

$$\boldsymbol{F}_i^{int} = \sum_{p=1}^{n_p} V_p \nabla\phi_i(\boldsymbol{x}_p) \cdot \sigma(\boldsymbol{x}_p), \tag{3.10}$$

$$\boldsymbol{F}_i^{grav} = \sum_{p=1}^{n_p} V_p \phi_i(\boldsymbol{x}_p) \rho_p \boldsymbol{g} \qquad = \sum_{p=1}^{n_p} m_p \phi_i(\boldsymbol{x}_p) \boldsymbol{g}. \tag{3.11}$$

Here ρ_p, V_p and m_p respectively denote the density, velocity 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.

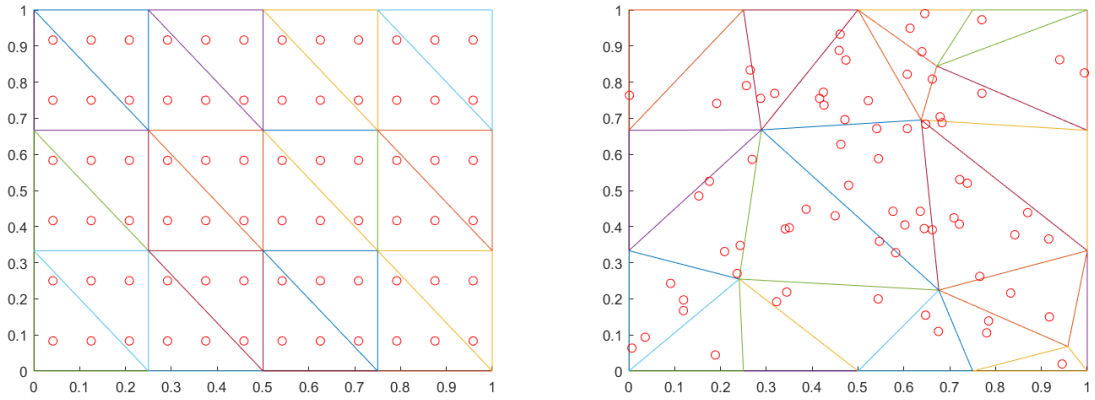
3.3. Solution procedure

In order to solve the equations of motion of a continuum through time, a time stepping procedure is required. The time stepping procedure presented in this section was recovered from [1] and [12]. This time stepping method is used by Deltares 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. It is suggested to initialise the material points and grid, such that each triangular element in the grid starts with about the same number of material points. The reason for this is that the basis functions will be integrated over each triangular element, and an equal distribution of particles per element will give a relatively small quadrature error considering the total number of particles. This will be discussed later on more elaborately. For now, assume that the triangular grid is defined and the material points are initialised inside the grid. At the starting time for the simulation, 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 3.2 illustrates two possible initialisations of the particles in the grid.

Having initialised the grid and the particles, the time stepping procedure can start. During each time step, the following phases are passed.

1. Assemble the mass matrix and force vectors of Equation 3.3 on the background grid. Use the particles as integration points as shown in Equations 3.9-3.11.



(a) Regular

(b) Irregular

Figure 3.2: 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.

2. Solve the coefficient vector $\hat{\boldsymbol{a}}$ from the system of Equations 3.3. 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 showed in Chapter 2.

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.

3.3.1. Assembling momentum conservation equation

In the first phase, Equation 3.3 needs to be assembled. For implementation purposes, it may be easier to consider each of the components x_k of the momentum equation separately, where k goes up to the dimension of the problem. If each direction x_k is considered one at a time, Equation 3.3 is written as

$$\boldsymbol{M}^t \hat{\boldsymbol{a}}^{x_k,t} = \boldsymbol{F}^{trac,x_k,t} - \boldsymbol{F}^{int,x_k,t} + \boldsymbol{F}^{grav,x_k,t}. \tag{3.12}$$

The assembly of the mass matrix and the force vectors are assembled using quadrature integration in the material points, as explained in Section 3.2. Only the assembly for the traction force has not been explained yet. The traction force is assembled by integration over the boundary where a traction force is applied, however this boundary is not well defined. This boundary is not prescribed and may move over time, so instead of tracking a boundary, 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 comes from a solid object pressing on the surface, for example a slab of concrete on top, compressing the continuum. The traction force in each of the boundary particles is then prescribed and the in other particles it is set to 0.

The traction force vector $\boldsymbol{F}^{trac}$ can then be assembled in a similar fashion as the other force vectors,

$$\boldsymbol{M}^t_{ij} = \sum_{p=1}^{n_p} m_p \phi_i(\boldsymbol{x}_p)\phi_j(\boldsymbol{x}_p), \tag{3.13}$$

$$\boldsymbol{F}_i^{trac,x_k,t} = \sum_{p=1}^{n_p} w_p \phi_i(\boldsymbol{x}_p) n_j(\boldsymbol{x}_p)\sigma_{jk}(\boldsymbol{x}_p), \tag{3.14}$$

$$\boldsymbol{F}_i^{int,x_k,t} = \sum_{p=1}^{n_p} V_p \frac{\partial \phi_i}{\partial x_j}(\boldsymbol{x}_p)\sigma_{jk}(\boldsymbol{x}_p), \tag{3.15}$$

$$\boldsymbol{F}_i^{grav,x_k,t} = \sum_{p=1}^{n_p} m_p \phi_i(\boldsymbol{x}_p) g_k. \tag{3.16}$$

3.3.2. Solving the momentum equation

Having assembled the momentum equation on the grid level, next the system of equations should be solved for the acceleration in each direction,

$$\hat{\boldsymbol{a}}^{x_k,t} = \left(\boldsymbol{M}^t\right)^{-1} \boldsymbol{F}^{x_k,t}, \tag{3.17}$$

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

$$\boldsymbol{F}^{x_k,t} = \boldsymbol{F}^{trac,x_k,t} - \boldsymbol{F}^{int,x_k,t} + \boldsymbol{F}^{grav,x_k,t}. \tag{3.18}$$

It is however computationally very expensive to calculate an inverse, and most choices of basis function yield a sparse mass matrix $\boldsymbol{M}$, so a numerical technique can be used to solve this system of equations. 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 the Appendix for more information about solving a system of equations by means of lumping.

Having solved the system of equations and determined the coefficients $\hat{\boldsymbol{a}}^{x_k,t}$, recall that the full acceleration field can be reconstructed from the basis functions as

$$a_h^{x_k,t}(\boldsymbol{x}) = \sum_{i=1}^{n_{dof}} \hat{a}_i^{x_k,t}\phi_i(\boldsymbol{x}). \tag{3.19}$$

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 then the other properties as well.

3.3.3. Updating particle properties

Various different methods within MPM exist for updating the particle properties. The method followed in this thesis was invented by Sulsky is al. [12], and will be referred to as the *modified update algorithm* in this thesis. This method was not the original update algorithm in MPM. This modified algorithm first 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 the grid level velocity field. Afterwards, the strain and stress are determined locally in the particles again.

Given the acceleration coefficient vector $\hat{\boldsymbol{a}}$, the update algorithm first updates the velocity in the particles. This is done by first calculating the acceleration in each particle point using Equation 3.19. Then the velocity is updated by adding the acceleration times the time step size to the old velocity,

$$v_p^{x_k,t+\Delta t} = v_p^{x_k,t} + \Delta t \sum_{i=1}^{n_{dof}} \hat{a}_i^{x_k,t}\phi_i(\boldsymbol{x_p}) \tag{3.20}$$

After determining the new velocity in each of the particles, the modified update algorithm projects the velocity from the particles 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 $v_h^{x_k,t}$ is assumed to be a linear combination of the basis functions ϕ_i,

$$v_h^{x_k,t+\Delta t}(\boldsymbol{x}) = \sum_{j=1}^{n_{dof}} \hat{v}_j^{x_k,t+\Delta t}\phi_j(\boldsymbol{x}). \tag{3.21}$$

For the density weighted L_2-projection on Φ the following system of equation must hold,

$$\int_{\Omega^t} \rho(\boldsymbol{x})\, v_h^{x_k,t+\Delta t}(\boldsymbol{x})\phi_i \,\mathrm{d}\Omega = \int_{\Omega^t} \rho(\boldsymbol{x})\, v^{x_k,t}(\boldsymbol{x})\phi_i \,\mathrm{d}\Omega, \quad \text{for } i = 1 \dots n_{dof}. \tag{3.22}$$

Note that the velocity on the left side is the projection on the space Φ, whereas the velocity on the right side is the velocity field which is only known in the particle points. Because the velocity on the right hand side is only known in the particle point, the integral will again be approximated by using a quadrature rule. The velocity projection on the left hand side can be substituted with Equation 3.21. The L_2-projection is expanded next,

$$\begin{aligned}
\int_{\Omega^t} \rho(\boldsymbol{x})\, v_h^{x_k,t+\Delta t}(\boldsymbol{x})\phi_i \,\mathrm{d}\Omega &= \int_{\Omega^t} \rho(\boldsymbol{x})\, v^{x_k,t}(\boldsymbol{x})\phi_i \,\mathrm{d}\Omega, \\
\Rightarrow \int_{\Omega^t} \rho(\boldsymbol{x}) \left(\sum_{j=1}^{n_{dof}} \hat{v}_j^{x_k,t+\Delta t} \phi_j(\boldsymbol{x}) \right) \phi_i(\boldsymbol{x}) \,\mathrm{d}\Omega &= \sum_{p=1}^{n_p} V_p \rho_p v_p^{x_k,t+\Delta t} \phi(\boldsymbol{x}_p), \\
\Rightarrow \sum_{j=1}^{n_{dof}} \left(\int_{\Omega^t} \phi_i(\boldsymbol{x}) \rho(\boldsymbol{x}) \phi_j(\boldsymbol{x}) \,\mathrm{d}\Omega \right) \hat{v}_j^{x_k,t+\Delta t} &= \sum_{p=1}^{n_p} m_p v_p^{x_k,t+\Delta t} \phi(\boldsymbol{x}_p)
\end{aligned}$$

The last equation must hold for each of the basis functions ϕ_i, $i = 1 \dots n_{dof}$. This results in a system of equations in which the mass matrix pops up again. The system can be written as

$$\boldsymbol{M} \hat{\boldsymbol{v}}^{x_k,t+\Delta t} = \boldsymbol{P}^{x_k,t+\Delta t}. \tag{3.23}$$

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

$$\boldsymbol{P}^{x_k,t+\Delta t} = \sum_{p=1}^{n_p} m_p v_p^{x_k,t+\Delta t} \phi(\boldsymbol{x}_p). \tag{3.24}$$

Equation 3.23 can be solved in a same manner as the acceleration coefficient vector was solved in Equation 3.3. In most practical cases, again lumping is applied to solve the system.

Having determined the projection of the velocity field on the grid level, the displacements and positions of the particles are updated by first calculating the incremental particle displacement during this time step

$$\Delta u_p^{x_k,t+\Delta t} = \Delta t \sum_{i=1}^{n_{dof}} v_i^{x_k,t+\Delta t} \phi_i(\boldsymbol{x}_p). \tag{3.25}$$

Next, the strain is updated using the incremental particle displacement using Equation 2.19,

$$\begin{aligned}
\frac{\partial \boldsymbol{\epsilon}_{ij,p}^{t+\Delta t}}{\partial t} &= \frac{1}{2} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \\
\Rightarrow \Delta \boldsymbol{\epsilon}_{ij,p}^{t+\Delta t} &= \Delta t \frac{1}{2} \left(\frac{\partial \left(\sum_{l=1}^{n_{dof}} \phi_l \hat{v}_l^{x_i,t+\Delta t} \right)}{\partial x_j} + \frac{\partial \left(\sum_{l=1}^{n_{dof}} \phi_l \hat{v}_l^{x_j,t+\Delta t} \right)}{\partial x_i} \right) \\
\Rightarrow \Delta \boldsymbol{\epsilon}_{ij,p}^{t+\Delta t} &= \Delta t \frac{1}{2} \sum_{l=1}^{n_{dof}} \left(\hat{v}_l^{x_i,t+\Delta t} \frac{\partial \phi_l}{\partial x_j}(\boldsymbol{x}_p) + \hat{v}_l^{x_j,t+\Delta t} \frac{\partial \phi_l}{\partial x_i}(\boldsymbol{x}_p) \right)
\end{aligned} \tag{3.26}$$

Next, the strain increment in Equation 3.26 will be used to update the stress. The constitutive relation from Equation 2.24 is discretised for this,

$$\begin{aligned}
\frac{\partial \sigma_{ij}}{\partial t} &= D_{ijkl} \frac{\partial \epsilon_{kl}}{\partial t} - \frac{\partial \epsilon_{kk}}{\partial t} \sigma_{ij} + \omega_{ik} \sigma_{kj} - \sigma_{ik} \omega_{kj}, \\
\Rightarrow \Delta \sigma_{ij,p}^{t+\Delta t} &= D_{ijkl} \Delta \epsilon_{kl,p}^{t+\Delta t} - \Delta \epsilon_{kk,p}^{t+\Delta t} \sigma_{ij,p}^{t} + \Delta \omega_{ik,p}^{t+\Delta t} \sigma_{kj,p}^{t} - \sigma_{ik,p}^{t} \Delta \omega_{kj,p}^{t+\Delta t},
\end{aligned} \tag{3.27}$$

in which $\omega^t_{i,j,p}$ is the discretised spin tensor,

$$\begin{aligned}
\omega_{ij} &= \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i}\right), \\
\Rightarrow \Delta\omega^{t+\Delta t}_{ij,p} &= \frac{1}{2}\Delta t\left(\frac{\partial\left(\sum_{l=1}^{n_{dof}} \phi_l \hat{v}_l^{x_i,t+\Delta t}\right)}{\partial x_j} - \frac{\partial\left(\sum_{l=1}^{n_{dof}} \phi_l \hat{v}_l^{x_j,t+\Delta t}\right)}{\partial x_i}\right), \\
\Rightarrow \Delta\omega^{t+\Delta t}_{ij,p} &= \frac{1}{2}\Delta t \sum_{l=1}^{n_{dof}}\left(\hat{v}_l^{x_i,t+\Delta t}\frac{\partial \phi_l}{\partial x_j}(\boldsymbol{x}_p) - \hat{v}_l^{x_j,t+\Delta t}\frac{\partial \phi_l}{\partial x_i}(\boldsymbol{x}_p)\right).
\end{aligned} \tag{3.28}$$

Note that the basis functions are fully known, and therefore, the derivatives are also known. Therefore, the full right hand side in Equations 3.26 and 3.28 are known. Finally the stress is stress is updated as

$$\sigma^{t+\Delta t}_{ij,p} = \sigma^t_{ij,p} + \Delta\sigma^{t+\Delta t}_{ij,p}. \tag{3.29}$$

Then, 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 $(1+\Delta\epsilon_{11})(1+\Delta\epsilon_{22})(1+\Delta\epsilon_{33})$ for 3D and likewise in lower dimensions. However, as $\Delta\epsilon \ll 1$, the higher order terms are neglected and the expansion factor becomes $\Delta\epsilon_{vol} = (1+\Delta_{11}+\Delta_{22}+\Delta_{33}) = (1+\Delta\epsilon_{kk})$. The full volume update then becomes.

$$V_p^{t+\Delta t} = \Delta\epsilon^{t+\Delta t}_{vol,p} V_p^{t+\Delta t} = \left(1+\Delta\epsilon^{t+\Delta t}_{kk,p}\right) V_p^{t+\Delta t}. \tag{3.30}$$

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

$$\rho_p^{t+\Delta t} = \frac{\rho_p^t}{\left(1+\Delta\epsilon^{t+\Delta t}_{kk,p}\right)}. \tag{3.31}$$

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

$$x^{t+\Delta t}_{k,p} = x^t_{k,p} + \Delta u_p^{x_k,t+\Delta t} \tag{3.32}$$

$$u^{t+\Delta t}_{k,p} = u^t_{k,p} + \Delta u_p^{x_k,t+\Delta t}. \tag{3.33}$$

Note that the particle position must be updated last, and therefore all the function evaluations happened in the old particle positions.

This is the whole algorithm that happens each time step. Figure **??** shown a overview of all the steps that are taken during each time step.

3.4. Boundary conditions

In Chapter 2, the equations of motion were derived along with the boundary conditions. In this section, the implementation of the boundary conditions is discussed. In Section 2.5, it was stated that the boundary was divided into a part on with prescribed displacement and a part with prescribed traction. First the boundary with the traction boundary condition 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. The area of the boundary each represents can be initially determined by dividing the domain into particle areas, areas which each belong to a particle. A possible way of doing so is by using a Voronoi diagram. Then from each of the particle areas, the volume and mass can be initialised, and the surface on the boundary as well. At each time step, the boundary particles can be assigned the total traction force on that specific boundary part. A setback of this method is that the a boundary particle may move away from the boundary, but is still assumed to be a boundary particle. A different way to implement the traction boundary condition is to use a moving mesh. In this method the mesh is moved along with the boundary points by compressing and stretching the mesh. This way the mesh will keep the same structure, but tracks the shape of the boundary much better. Also, the boundary conditions can then

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 3.13-3.16.
2. Solve the acceleration coefficient vector $\hat{\boldsymbol{a}}$ from the system of equations $\boldsymbol{M}\hat{\boldsymbol{a}} = \boldsymbol{F}$ (Equation 3.3. 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 3.20.
4. Reconstruct the new velocity field by projecting the particle data onto the grid. This is done using a L_2-projection onto the basis functions. Assemble Equation 3.21 and solve for the coefficient vector 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 3.33.
6. Find the strain increment at each of the particles using the displacement increment at the particles as done in Equation 3.28.
7. From the strain increment, the stress, volume and density can immediately be updated using Equations 3.29, 3.30 and 3.31.
8. Finally, the total displacement and the positions of each of the particles are updated, as in Equations 3.32 and 3.33

Figure 3.3: The time stepping algorithm.

be assigned to the force vector over the mesh directly, instead of first assigning it to particles. More details can be found in [1] and [13]. For the scope of this thesis and the benchmarks considered, a moving mesh is not considered. The traction boundary condition is implemented by assigning traction forces directly to the particles. This method is easier to implement and the focus of this thesis is the change to higher order basis functions, not the boundary conditions.

The boundary conditions on the displacement boundary $\partial\Omega_u$ is implemented by imposing that the 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, and its center lies just next to the physical boundary. The boundary conditions are imposed on the reconstructed velocity field by setting the degrees of freedom at the boundary to the corresponding value. This is done by changing a number of equations in the system of Equations 3.3. Consider an example in which piecewise linear basis functions are used. A basis function ϕ_i is 1 at the boundary, and on the position of its peak, 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 3.3 by a row of zeros, except for a 1 at the place i^{th} place, and the value v_x in the total force vector in i^{th} row. In this thesis however, the Dirichlet boundaries considered will always by homogeneous, so these boundaries will always be stationary. Therefore, the degrees of freedom at the boundary are simple set to zero for both the acceleration and velocity field.

3.5. Critical time step

The time integration method used in the solution procedure is called the semi-implicit Euler-Cromer scheme [14]. 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 will diverge 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, 15]. The condition is given by

$$\Delta t \leq \frac{h}{\sqrt{E/\rho}}. \tag{3.34}$$

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.

3.6. Grid Crossing Error

Grid crossing errors occur when a particle crosses over from one element to a neighbouring element. 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, by summing over the values in the material points. Recall Equation 3.15,

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

Next consider a piecewise linear basis function and particles as shown in Figure 3.4. 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 crossing are compared. From the illustration, it can be seen that $\frac{\partial \phi}{\partial x}$ is positive constant left of the 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 $\sigma = 1$ over the entire domain and that the volume of each particle is constant V. Then before the crossing, the total internal force is

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

whereas after the crossing, the internal force is

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

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

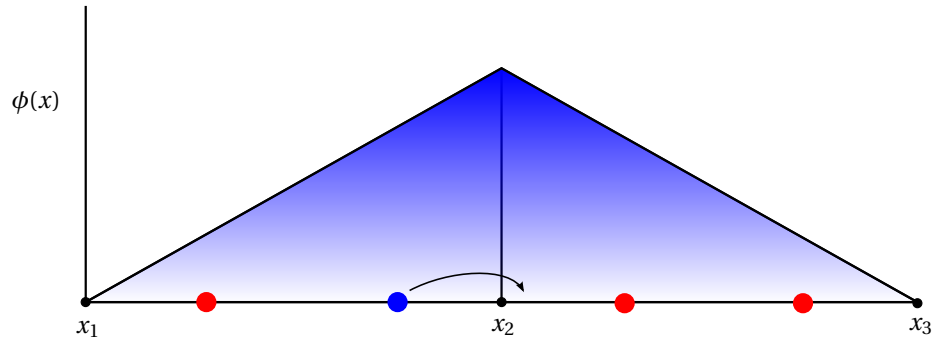


Figure 3.4: Illustration of grid crossing on a 1D grid, the green particle just left of x_2 moves to the position of the blue particle just right of x_2 during one time step. The red particles do not move.

This problem can be mitigated by first reconstructing a global stress function and then using Gaußintegration using fixed Gaußpoints. As the used Gaußpoint are then fixed, grid crossing error do not occur anymore. A second possibility is to use basis function with C^1-continuity over element interfaces. In this thesis, C^1-continuous basis functions will be used, so grid crossing errors are not expected.

4

B-SPLINES ON TRIANGULATIONS

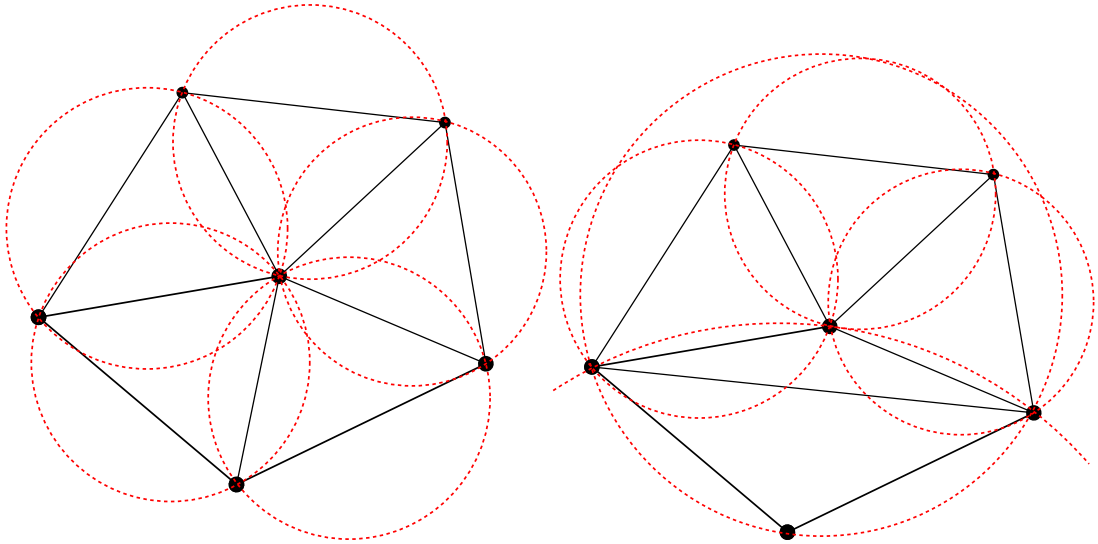
In the last chapter, the numerical algorithm has been described for solving a continuum deformation problem with MPM. The role of the triangular grid and the basis functions defined over the grid have been explained. In this section, a short introduction into the construction of the grid is discussed. Then the choice of basis function over the grid is considered. The classic MPM uses piecewise linear Lagrangian basis functions. More will be explained about this in Section 4.3, and also the shortcomings when considering higher order Lagrangian basis functions. The goal of this thesis however is to replace this basis by one that has continuous derivatives over the full domain and has the property of non-negativity. These conditions all hold for a basis of B-spline functions, which will be considered in Section 4.4. The B-spline basis will be derived and the implementation in a MPM code will be discussed. In Chapter 5 the spatial convergence and computation time of a B-spline based MPM will be compared to classic MPM.

4.1. 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 crashes. For example, when considering a bar that is being stretched, The gid must be large enough for later deformations. 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, triangles without any particles inside. This is not a problem however, elements without particles inside are not a problem for 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.

Next consider the construction of the grid. Triangular grid are easy to construct and refine. One of the most used triangular grid construction method is the Delaunay triangulation procedure [16]. For this procedure, first a set of vertices is defined. Then these vertices are connected such that the domain is divided into vertices and the circumcircle of each triangle contains only its own vertices. A 2D example of a Delaunay triangulation is shown in Figure **??**. For 3D, the procedure is similar, except that tetrahedrons and circumspheres are considered instead of triangles and circumcircles. The procedure avoids as much as possible sliver triangles, which are very long stretched triangles with small angles. Sliver triangles can cause instabilities and are undesirable to work with. As sliver triangles are avoided, the Delaunay triangulation procedure is very suitable 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 there initially 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.

When initialising the grid, the particles are also initialised in MPM. After defining the triangulation, it is adviced to place about the same number of particles in each triangle that lies within the initial domain of the continuum. This is because the error in the quadrature rule for the mass matrix and force vectors in Equa-



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

Figure 4.1: 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 a Delaunay. Two of the circumcircles contain other points. Note that sliver triangles appear as a consequence.

tions 3.13-3.16 depends on the number of integration points. When dividing the particles evenly over the elements, the total integration error is mitigated best.

Once the particles have been distributed over the domain, each should be assigned a 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 4.2.

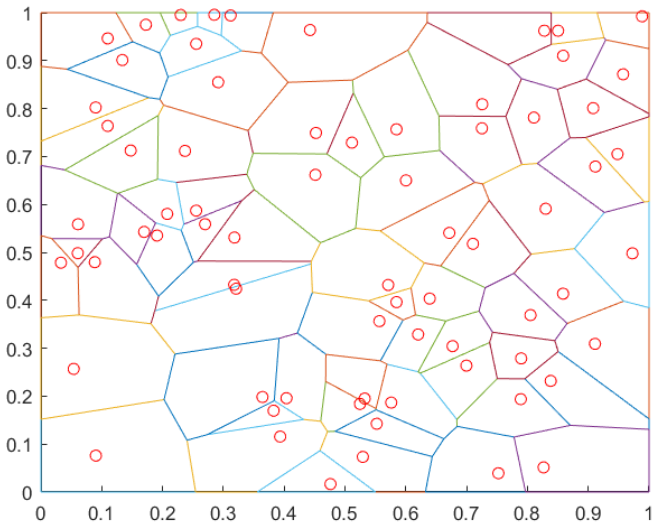


Figure 4.2: A Voronoi diagram of a randomly distributed particles, marked as circles. Not to be confused with the background grid, which this has nothing to do with. The Voronoi diagram is used to initialise the volume of each particle.

4.2. BARYCENTRIC COORDINATES

Before defining the basis functions over the triangulation, a barycentric coordinate system should be discussed. Baycentric 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, then the Cartesian coordinates of a point $(x,)$ can be converted to barycentric coordinates with respect to this triangle by solving

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{4.1}$$

$(\zeta_1, \zeta_2, \zeta_3)$ are the barycentric coordinates of the point with respect to the triangle. Note that there are three coordinates, whereas the 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 4.1. 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 the $(1/3, 1/3, 1/3)$ is the center of mass of the triangle. Figure 4.3 show an example of a triangle and its barycentric coordinates. The barycentric coordinates for a tetrahedron in $\mathbb{R}^3$ are similar to the 2D barycentric coordinates. As the tetrahedron has four vertices, each point is described with four barycentric coordinates, which are determined in a fashion similar to Equation 4.1, but then with an extra vertex and an extra dimension,

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \\ \zeta_4 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{4.2}$$

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

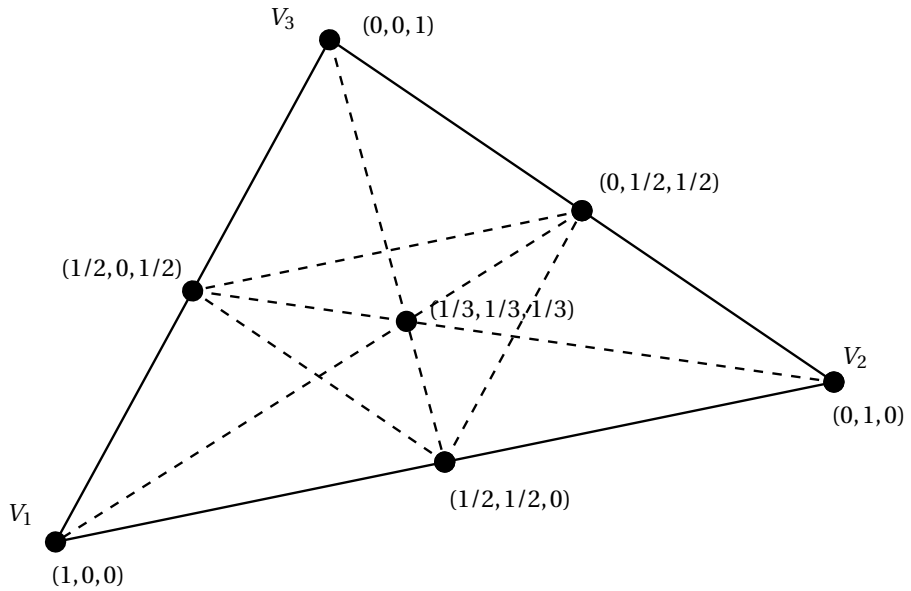


Figure 4.3: A number of barycentric coordinates of an arbitrary triangle with respect to its vertices V_1, V_2, V_3.

4.3. LAGRANGIAN BASIS FUNCTIONS

In this section, the Lagrangian basis functions over triangulation are discussed. The basis functions will be expressed in barycentric coordinates. First however, the one-dimensional Lagrangian basis function 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 4.4 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 function,

$$\phi_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}, \tag{4.3}$$

In this equation, δ_{ij} is the Kronecker delta function, x_j denotes node number j and ϕ_i is the i^{th}. 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 piecewise polynomials. Over each element, the basis functions are smooth polynomials, and over function interfaces, the piecewise polynomials are continuously connected. The derivatives however, are not.

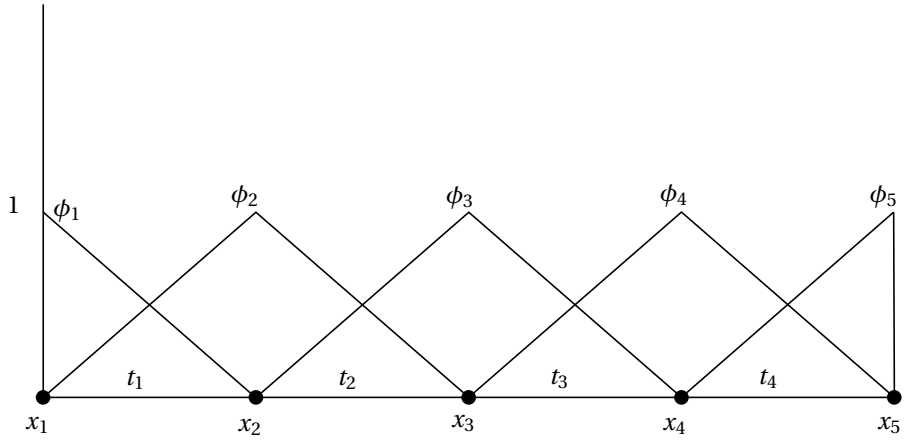


Figure 4.4: A 1D equidistant grid with 5 vertices V and 4 elements t. The piecewise linear Lagrangian basis functions are shown over this grid.

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

$$\phi_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & \text{if } x_{i-1} \leq x < x_i \\ \frac{x-x_{i+1}}{x_i-x_{i+1}} & \text{if } x_i \leq x \leq x_{i+1} \\ 0 & \text{else.} \end{cases} \tag{4.4}$$

This basis function is illustrated in Figure 4.4.

Next consider the piecewise quadratic Lagrangian basis function. For this choise of basis functions, first the grid must be refined. In the middle of each element, an additional node is added, as shown in Figure **??**. The elements are the same as in the unrefined grid however, each element has two points at its borders and an additional one in its middle. Now consider element i. Over this element, three basis functions are non-zero, these are given by

$$\phi_i = \frac{(x-x_{i+1/2})(x-x_{i+1})}{(x_i-x_{i+1/2})(x_i-x_{i+1})} \tag{4.5}$$

$$\phi_{i+1/2} = \frac{(x-x_i)(x-x_{i+1})}{(x_{i+1/2}-x_i)(x_{i+1/2}-x_{i+1})} \tag{4.6}$$

$$\phi_{i+1} = \frac{(x-x_i)(x-x_{i+1/2})}{(x_{i+1}-x_i)(x_{i+1}-x_{i+1/2})}. \tag{4.7}$$

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 4.6. 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.

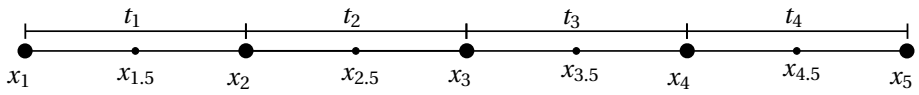


Figure 4.5: A 1D equidistant grid with 5 vertices V and 4 elements t. The grid has been refined with one extra vertex in each element t.

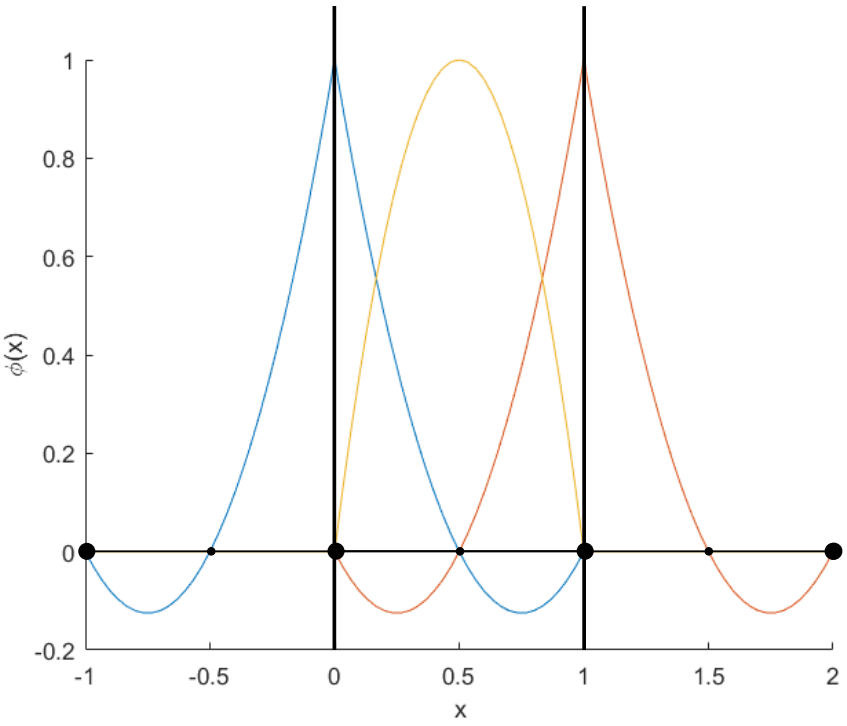


Figure 4.6: The second order basis functions over the refined grid. All non-zero basis functions over the element $[0,1]$ are shown. Two basis functions are also non-zero over neighbouring elements, and the extensions are shown. Each basis function is 1 in exactly one of the grid points, and zero on all the others.

For higher order Lagrangian basis functions, 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

$$\phi_{i+k/(p-1)} = \prod_{j=0, j\neq k}^{p} \frac{x - x_{i+j/(p-1)}}{x_{i+k/(p-1)} - x_{i+j/(p-1)}}. \tag{4.8}$$

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 4.7. Next the Lagrangian basis functions will be considered in two and three dimensions.

4.3.1. Higher dimensional Lagrangian basis

Having defined the one-dimensional basis functions, now the higher dimensional Lagrangian basis functions are considered. The grid in higher dimensions is divided in triangles or tetrahedrons, these are the elements in the grid. In higher dimensions, the Lagrangian basis functions are still piecewise polynomials. In each element, nodes are inserted. 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 piecewise linear case. The elements

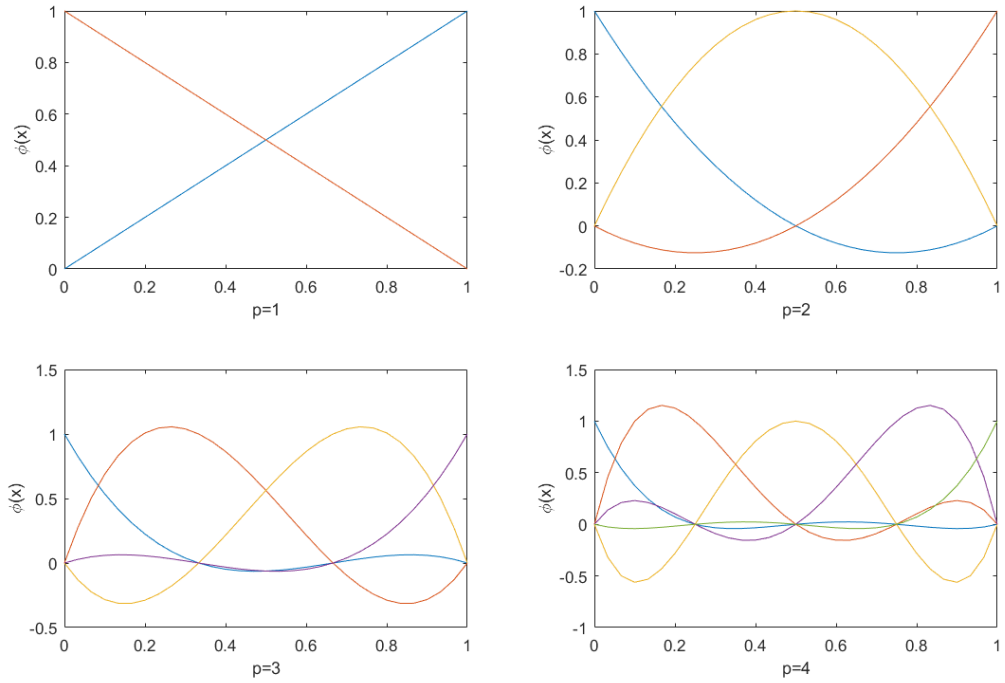


Figure 4.7: The Lagrange basis functions over a single element for various polynomial orders p.

do not have to be refined for this. Each 2D triangular element has three points in the element (the corners of the triangle), and a 3D tetrahedral has four point. Therefore, there are three basis functions over each triangle and four over each tetrahedral. The basis functions are easiest described using the barycentric coordinates of the triangle or tetrahedral. For a 2D triangle, the basis function over the element in barycentric coordinates are

$$\phi_1 = \zeta_1 \tag{4.9}$$
$$\phi_2 = \zeta_2 \tag{4.10}$$
$$\phi_3 = \zeta_3. \tag{4.11}$$

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 4.8 shows one of these basis functions over a single element.

In order to get higher order basis functions over the triangulation, first the grid needs to be refined. If a p^{th} order basis function is required, then first refine each element with $p-1$ nodes in each direction. The nodes corresponding with this refinement are defined as follows. The nodes inserted have 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 $k + l + m = p$. Now let n_{klm} be the node with barycentric coordinates $\frac{1}{p}(k, l, m)$. An example of this refinement is shown in Figure 4.9. In 3D, this procedure is similar.

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

$$\phi_{klm} = \left(\prod_{q=0}^{k-1} \frac{\zeta_1 - \frac{q}{p}}{\frac{k}{p} - \frac{q}{p}}\right) \cdot \left(\prod_{r=0}^{l-1} \frac{\zeta_2 - \frac{r}{p}}{\frac{k}{p} - \frac{r}{p}}\right) \cdot \left(\prod_{s=0}^{m-1} \frac{\zeta_3 - \frac{s}{p}}{\frac{k}{p} - \frac{s}{p}}\right). \tag{4.12}$$

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 4.10. Furthermore, it must be

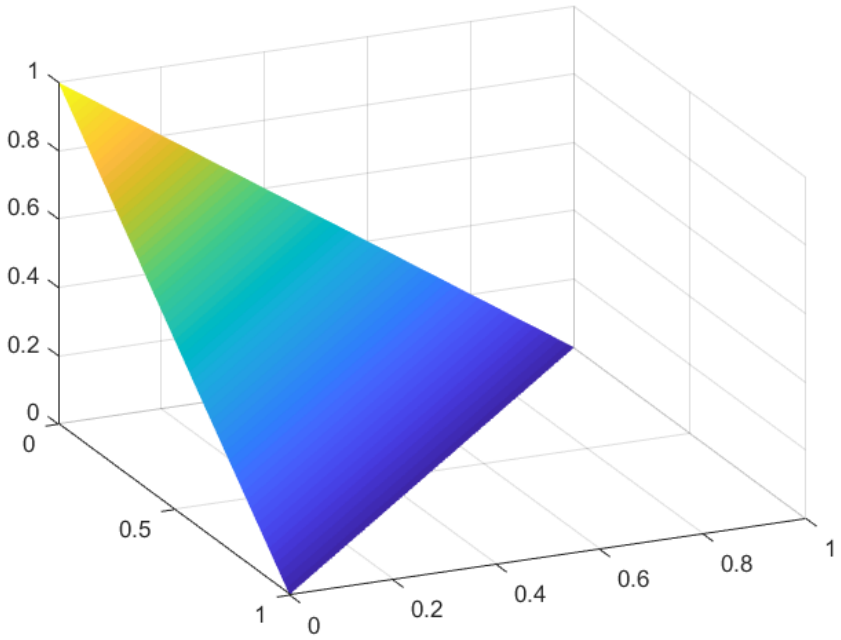


Figure 4.8: A linear basis function over a single triangular element.

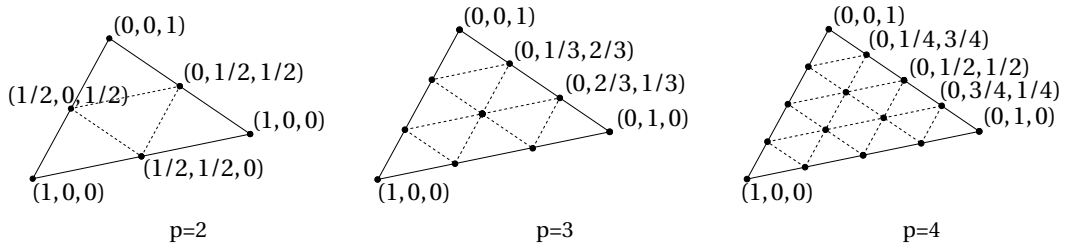


Figure 4.9: Three refinements of an element for order p Lagrange basis functions. The barycentric coordinates are $\frac{1}{p}(k,l,m)$, some of which are shown in the figure.

guaranteed that ϕ_{klm} is equal to 1 in n_{klm}. This is done by ensuring that each term in Equation 4.12 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 quadratic basis functions are shown in Figure 4.11. Also note that each basis function that is not zero on all the edges should be mirrored over the edge it is not zero on. If the basis function is nonzero in a vertex of the element, then the full basis function should be extended over all the elements that include that vertex.

This concludes the definition of Lagrangian basis functions over triangular grids. Although most examples and definitions were only considered for 2D, the procedure for extending this to higher dimensions is similar and often only includes adding a fourth barycentric coordinate ζ_4. Having defined and shown the basis functions, two properties appear that are undesirable for implementation in MPM. The first of these disadvantages is that the functions have discontinuous derivatives over triangle edges. This may cause grid crossing errors, as explained in Section 3.6. Second, all basis functions that are piecewise polynomials of order 2 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 discussed in Section **??**. If the MPM is to be extended 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.

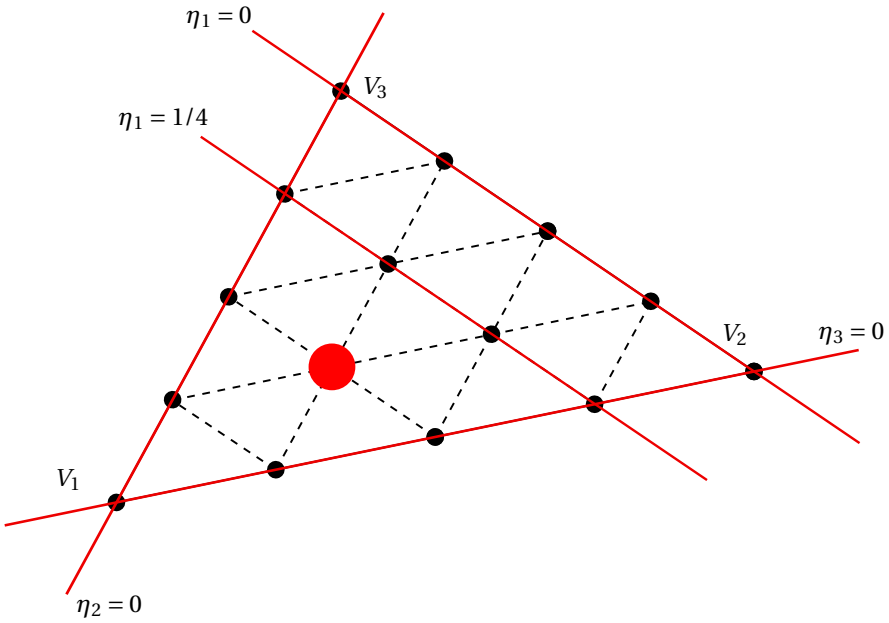


Figure 4.10: The procedure for finding a basis function that is zero in all nodes except for one. In this case, the node with coordinates $(1/2, 1/3, 1/3)$ is considered, marked as the large dot. The function must be zero in all other nodes. The required function corresponding to this grid is of order $p = 4$, 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 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.

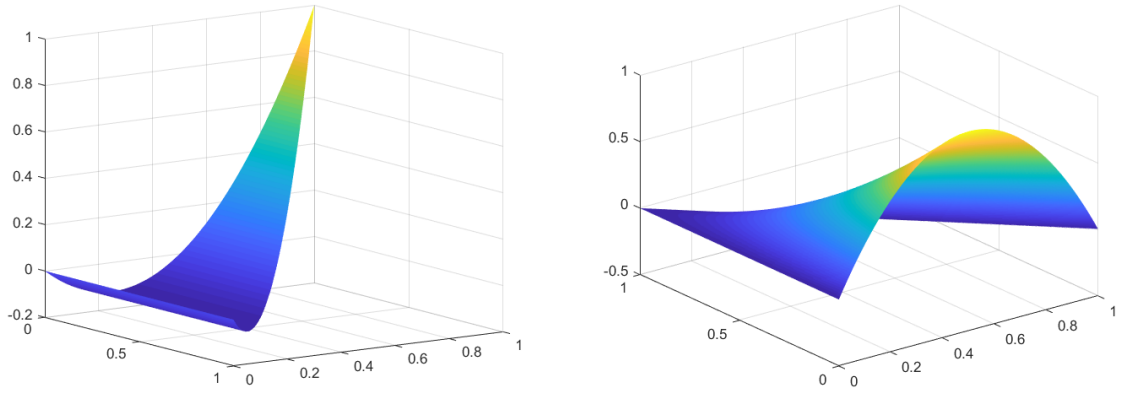
4.4. B-SPLINE BASIS FUNCTIONS IN 1D

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. The extension of these specific basis functions to 2D and 3D will prove hard, and the problems are explained as the end of this section. Although these basis functions are not directly extendable to higher dimensional triangulations, the concept and properties may be used to construct and understand a B-spline basis on triangulations in 2D and 3D.

The desired B-spline basis functions for MPM should satisfy a number of properties. First, the basis functions should be continuously differentiable over the entire domain. This will prevent any grid crossing errors and will ensure that the final solution that no unphysical jumps in the solution will appear. Second, the functions must be non-negative. This will ensure that the mass matrix used in MPM can be lumped. Furthermore, negative terms may cause instabilities, which may cause large errors in the solution. Third, the basis should be a partition of unity. This means that when the basis functions are added summed, the result will be the function 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 fully defined by their *knot vector*. A knot vector is a 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

$$\Xi = \left[\xi_1, \xi_2, \ldots \xi_{n+p+1}\right] \tag{4.13}$$



(a) Vertex basis function

(b) Edge middle basis function

Figure 4.11: 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.

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 nodes at the positions $x = 0,1,2,3,4,5$, with piecewise 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 [**?**]. This recursion formula first defines piecewise constant functions using the knot vector. From the piecewise constants, piecewise linear B-splines $\phi_{i,0}$ are generated, and so on. From the order d B-splines $\phi_{i,d}$, order $d+1$ B-splines $\phi_{i,d+1}$ are generated, up to the desired order p, $\phi_{i,p}$. First define piecewise the constant functions as

$$\phi_{i,0}(\xi) = \begin{cases} 1 & \text{for } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{else.} \end{cases} \tag{4.14}$$

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.

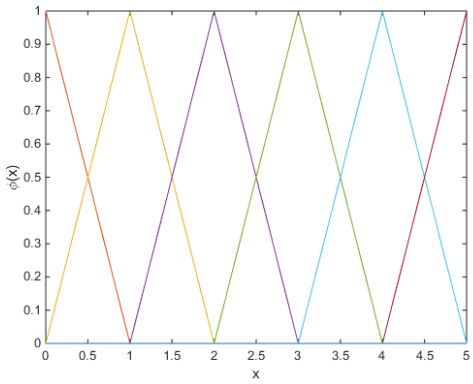
Next, higher order basis functions are constructed recursively from the basis functions of one order lower,

$$\phi_{i,d}(\xi) = \frac{\xi - \xi_i}{\xi_{i+d} - \xi_i}\phi_{i,d-1}(\xi) + \frac{\xi_{i+d+1} - \xi}{\xi_{i+d+1} - \xi_{i+1}}\phi_{i+1,d-1}(\xi). \tag{4.15}$$

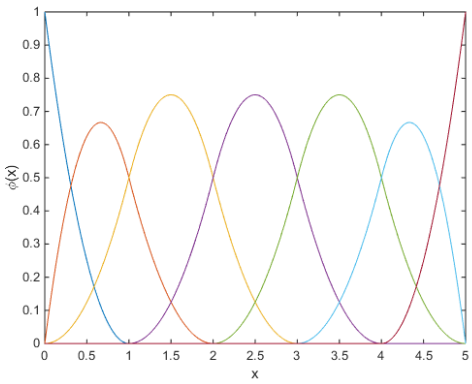
Each higher order basis function is constructed from two lower order basis functions. Note that for basis function $\phi_{i,d}$ the basis function $\phi_{i+1,d-1}$ is required, so with each increase in order, there is one less basis functions than the lower level. It was assumed that ξ has $n+p+1$ elements, 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 4.12 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 have 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 4.13. 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 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.

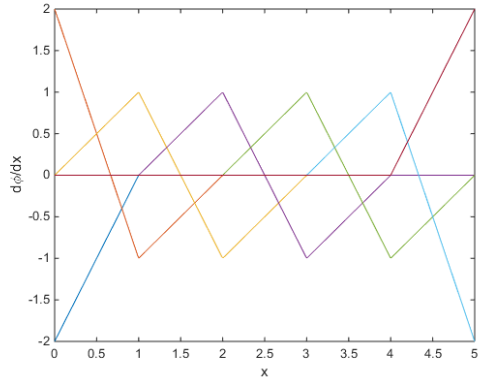
All together, B-spline basis functions are very fit for application in MPM. The basis functions are easy to construct and are very flexible. The knots in the knot vector can easily be spatially refined by adding more nodes closer to each other at places of interest. High order basis function can be generated easily. The basis functions have high continuity over element edges, which can be lowered in accordance with the geometry.



Linear B-splines basis functions.



Quadratic B-spline basis functions.



The derivatives of the quadratic B-spline basis functions.

Figure 4.12: The piecewise linear and piecewise quadratic basis functions, corresponding to the knot vector $\Xi = [0,0,0,1,2,3,4,5,5,5]$. The derivatives of the piecewise quadratic basis functions are also shown, which illustrates the C^1-continuity of the order 2 basis functions.

Finally, the basis functions have the properties of locality, partition of unity and non-negativity. B-spline basis functions are ideal for use in MPM. For higher dimensions, it is possible to use a tensor product of 1D knot vectors, and therefore get a tensor product of 1D basis functions,

$$\phi_{ijk,p}(x,y,z) = \phi^x_{i,p}(x) \cdot \phi^y_{j,p}(y) \cdot \phi^z_{k,p}(z). \tag{4.16}$$

The 3D basis function $\phi_{ijk,p}$ is a tensor product of three 1D basis funtions, 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 4.14. The geometry in this example is hard to represent as a mapping Ψ from a square, parametric domain to the physical domain, and finding an good division and mapping of the domain may prove very hard. Furthermore, it is very hard to locally refine the grid. If the grid should 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 unnecessary local refinement in other parts of the domain as well, as can also be seen in Figure 4.14. This costs more computation time when solving the solution over the grid. The tensor product grid construction is unsuitable for arbitrary geometries, whereas a triangular grid would be much more suitable for this.

When using a triangular grid however, it is very hard to construct B-spline basis functions over arbitrary 2D and 3D triangulations. The Cox-de Boor algorithm does not work on triangulations, and therefore, it is not possible yet to construct B-splines over triangles of any order with a straightforward algorithms. 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,

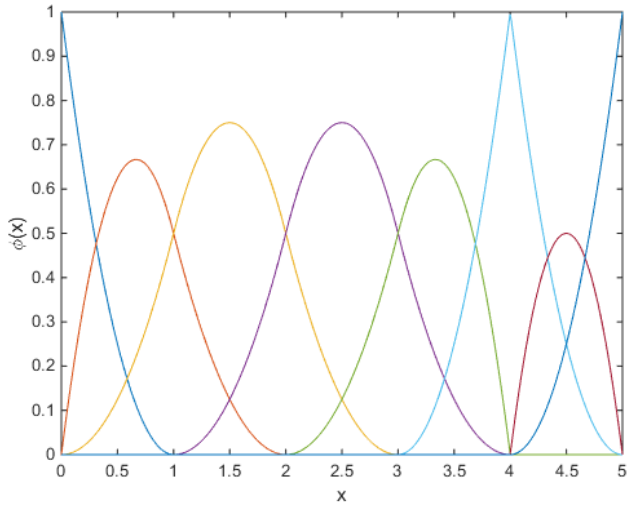


Figure 4.13: The quadratic B-spline basis functions corresponding to the knot vector $\Xi = [0,0,0,1,2,3,4,4,5,5,5]$. Note the C^0 continuity at 4.

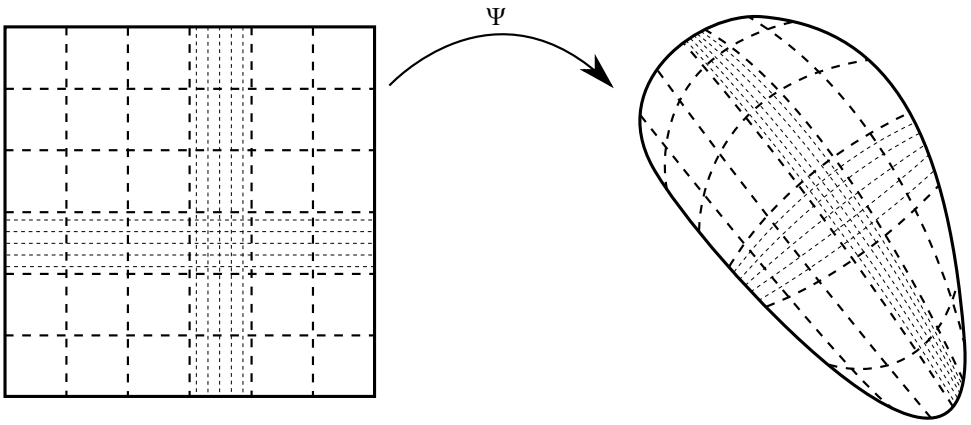


Figure 4.14: A possible mapping Ψ of a square, parametric domain to the physical domain. In the middle, the grid has been refined. The quality of this mapping is unsure, also due to the fact that the square domain has discontinuities at the corners, whereas the physical domain has a smooth boundary.

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 same advantages as the 1D B-spline basis functions should be experienced in MPM.

4.5. 2D QUADRATIC POWELL-SABIN B-SPLINES ON ARBITRARY TRIANGULATIONS

For the construction of B-splines on arbitrary triangulations, there is not yet an general algorithm to generate these for an arbitrary order p on 2D or 3D. Therefore, first the construction of quadratic B-splines will be considered for a 2D triangulation. The construction of these B-splines are not unique; for a certain triangulation there are different possibilities to generate local B-splines over the grid. The B-splines are still required to be locally defined, non-negative, partition of unity and piecewise order p polynomial over each of the (sub-)elements, with C^{p-1}-continuity over the edges. A full research was by Dierckx [**?**] was dedicated to constructing and researching quadratic B-splines over a 2D quadratic grid, which resulted in so called *normalized Powell-Sabin B-splines*. This thesis follows the approach followed 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 poten-

tial of quadratic PS splines has already been proved by Speleers et al. [17] for 2D advection-diffusion-reaction problems, and therefore PS splines are expected to improve 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 node i in the triangulation, three basis functions ϕ_i^q, $q \in \{1,2,3\}$ with local support will be defined associated to that node. The functions will be nonzero on all triangular elements directly around this node, and zero on all other elements. Then any continuous function can be approximated by a linear combination of these basis functions,

$$f(x,y) \approx \hat{f}(x,y) = \sum_{i=1}^{n} \sum_{q=1}^{3} c_{i,q} \phi_i^q(x,y), \tag{4.17}$$

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

$$\phi_i^q(x,y) \geq 0, \quad \text{for all } x,y \in \Omega, \tag{4.18}$$

$$\sum_{i=1}^{n} \sum_{q=1}^{3} \phi_i^q(x,y) = 1, \quad \text{for all } x,y \in \Omega. \tag{4.19}$$

Furthermore, the functions must be piecewise quadratic and be C^1-continuous over (sub-)element edges, so its derivative must be continuous.

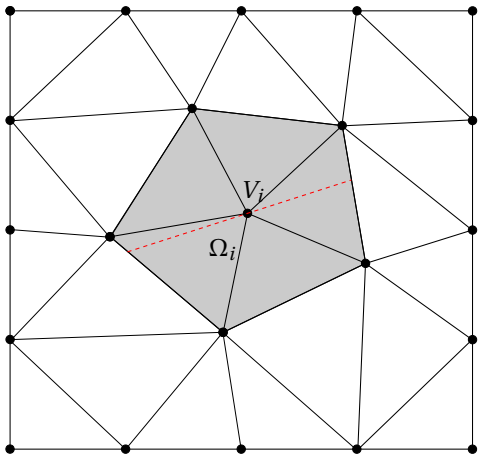
In order to achieve this, first the triangular grid must be refined, each triangular element is divided into smaller sub-elements. After the refinement of the grid, three control points will be chosen for each node, which form a so-called control triangle. The control triangle at each of the nodes will fully define the three basis functions $phi_i^1, \phi_i^2, \phi_i^3$ at each node 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.

4.5.1. Grid refinement

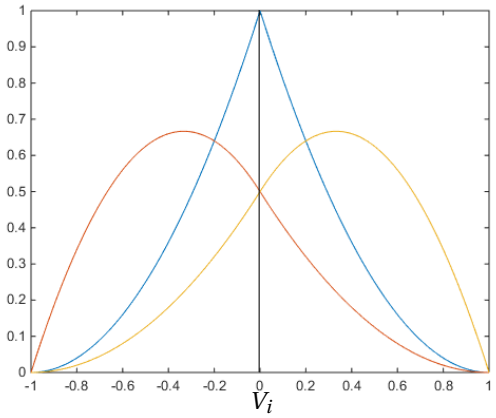
Given a domain Ω, let Δ be a triangulation of Ω. In this section the refinement process of Δ will be explained. First the need for a refinement is explained. Consider a node i and one of its three basis functions ϕ_i^q. This basis function should be locally defined on the triangles directly around node i, let this local domain be called molecule Ω_i. An example of Ω_i is considered in Figure 4.15a. The basis function ϕ_i^q should be a piecewise parabola, in which over each element a parabola is defined with smooth transitions over the boundary of each domain. At the boundary of Ω_i, the value of ϕ_i^q should be zero as well as its derivative, because the function should go over smoothly into the 0 function outside the domain Ω_i. Next consider the piecewise parabola over the dotted line in Figure 4.15a. This line through two elements, and over this line, the basis function should be a C^1-continuous 1D piecewise parabola. The piecewise parabola should smoothly go over into the 0 function at the boundaries of the domain Ω_i. Figure 4.15b shows that it is not possible to construct such a piecewise parabola with just two elements other than the trivial solution $\phi_i^q = 0$. At piecewise parabola must consist of at least three elements to meet the imposed conditions, see Figure 4.15b. 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.

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 4.16, which was constructed with the following steps:

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



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



(b) Piecewise parabolas consisting of two parts.

Figure 4.15: A triangulation (a) of a square domain. A vertex V_i and its corresponding molecule Ω_i are marked. Over this molecule, a 2D piecewise parabola has to be defined, which must be C^1-continuous and smoothly goes to zero at the boundary. Some possibilities for the 2D piecewise 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 piecewise parabolas are inspected. However, all possibilities have discontinuous derivatives over the over the element edges, or do not smoothly go to zero at the boundary.

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

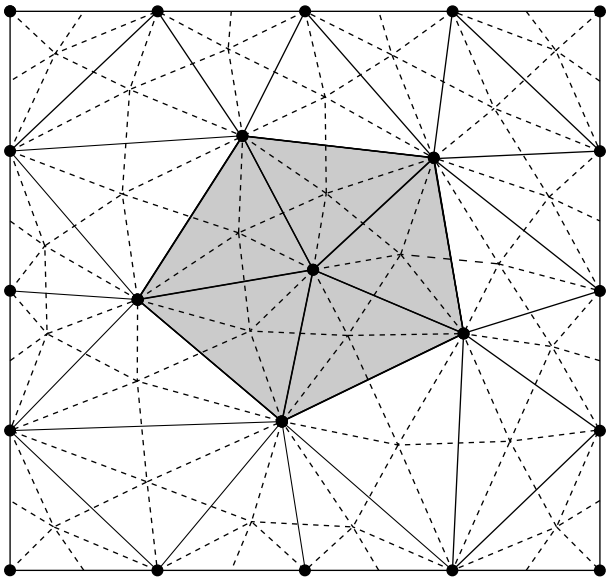
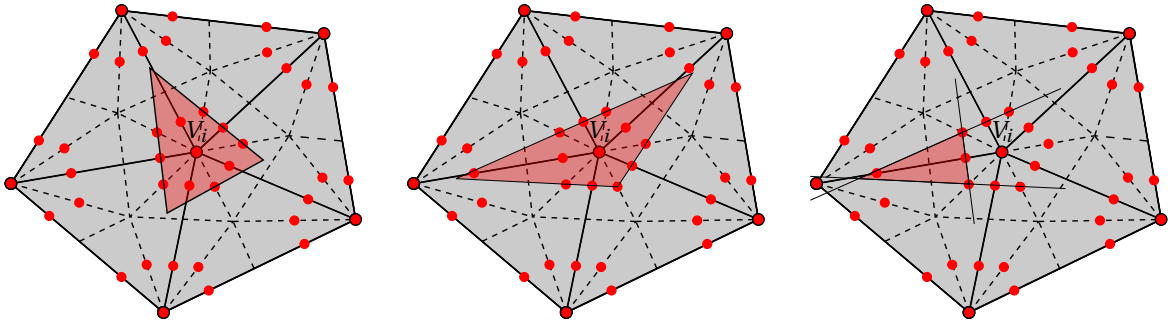


Figure 4.16: A PS refinement Δ^* of the triangulation Δ used earlier.

4.5.2. 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 the vertices V_i and the midpoints of all edges in the PS refinement with V_i at one end. The PS points of a refined triangulation Δ^*



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

Figure 4.17: The PS points and some possible PS-triangles of the marked molecule from the example triangulation of Figure 4.16. 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) shown a triangle which also has three edges in common with the convex hull, but this one does not contain the hull, so it is invalid.

are shown in Figure 5.3. Let PS_i be the set of PS points associated with vertex i, then the control triangle t_i of vertex i must contain all the PS points in PS_i. If the control triangle t_i contains all PS points of PS_i, then non-negativity of the basis functions associated with t_i is assured. The control triangle t_i is not uniquely defined, Figure 5.3 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 PS_i, as this produces PS-splines with good stability properties [17]. 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 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 surface. 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. Given a set of PS points, the edges must be constructed, the intersection points between edges must 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

$$e_{12}(t) = \boldsymbol{x_1} + t\boldsymbol{v_{12}} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \zeta \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}. \tag{4.20}$$

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

$$\begin{aligned} e_1(\zeta_1) &= e_2(\zeta_2) && (4.21)\\ \Rightarrow \boldsymbol{x_1} + \zeta_1 \boldsymbol{v_1} &= \boldsymbol{x_2} + \zeta_2 \boldsymbol{v_2} && (4.22)\\ \Rightarrow \zeta_1 \boldsymbol{v_1} - \zeta_2 \boldsymbol{v_2} &= (\boldsymbol{x_2} - \boldsymbol{x_1}) && (4.23)\\ [\boldsymbol{v_1}\, \boldsymbol{v_2}] \begin{bmatrix} \zeta_1 \\ \zeta_2 \end{bmatrix} &= (\boldsymbol{x_2} - \boldsymbol{x_1}) && (4.24)\\ A\boldsymbol{\zeta} &= \boldsymbol{b}. && (4.25) \end{aligned}$$

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

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

$$S(\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3) = \frac{1}{2} \left| x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \right|. \tag{4.26}$$

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 4.17c. Given again the vertices $\boldsymbol{q}_i, i = 1,2,3$ of a triangle, the barycentric

coordinates of the center with respect to the triangle can be calculated using Equation 4.1. In case one of the barycentric coordinates in negative, the center point is outside the triangle, and the triangle is invalid. With these procedures, the triangles with three common edges with the hull can be constructed.

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

$$\boldsymbol{b_{12}}(\zeta) = \boldsymbol{q}_{12} + \zeta \boldsymbol{v_{12}} = \boldsymbol{q_{12}} + \zeta \left(\frac{\boldsymbol{v}_1}{|\boldsymbol{v}_1|} + \frac{\boldsymbol{v}_2}{|\boldsymbol{v}_2|} \right), \tag{4.27}$$

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, as Figure 4.18 illustrates. This corresponds with 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, an 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.

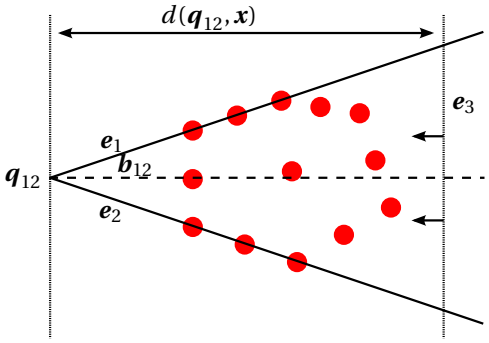


Figure 4.18: Illustration for construction of the last edge for a PS triangle with two shared edges with the convex hull of PS points. The last edge is a line orthogonal to the bisector $\boldsymbol{b}_{12}$ and goes through the PS-point with the greatest distance $d(\boldsymbol{q}_{12}, \boldsymbol{x})$. This corresponds to sliding the $\boldsymbol{e}_3$ to the left until it collides with a PS-point.

4.5.3. PS SPLINE BASIS FUNCTION CONSTRUCTION

In this section, the construction of piecewise quadratic basis function from the PS-triangles is derived. During the derivation, the choice for control triangles will also become clear. For a full derivation, see the papers of Dierckx [18**?**].

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

$$c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2. \tag{4.28}$$

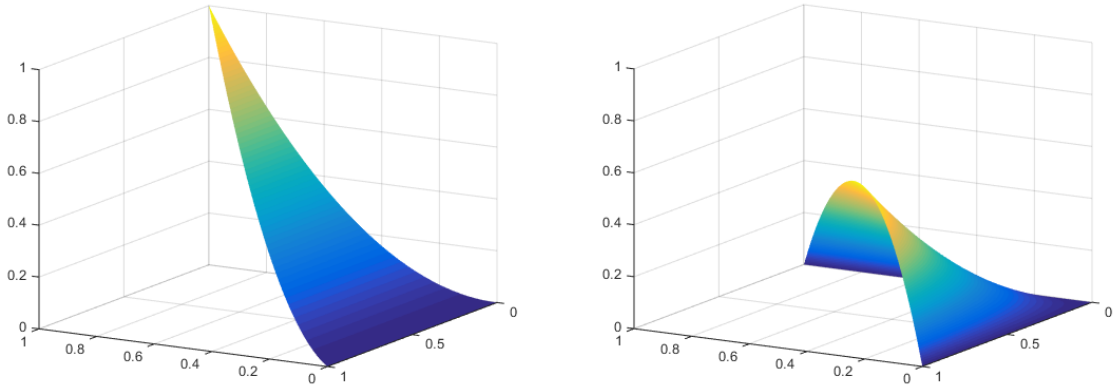
Any parabola can be described using 6 degrees of freedom. Instead of Cartesian coordinates, the parabola can also be described in barycentric coordinates, recall Equation 4.1. Using the barycentric coordinates $\boldsymbol{\zeta}$ of triangle t, it is possible to describe any parabola over a triangle 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

$$B_{i,j,k}(\boldsymbol{\zeta}) = \frac{2!}{i!j!k!} \zeta_1^i \zeta_1^j \zeta_1^k, \tag{4.29}$$

with the properties

$$B^2_{i,j,k}(\boldsymbol{\zeta}) \geq 0 \quad \forall (x,y) \in t, \tag{4.30}$$

$$\sum_{\substack{i+j+k=2, \\ i,j,k \geq 0}} B^2_{i,j,k}(\boldsymbol{\zeta}) = 1 \quad \forall (x,y) \in t. \tag{4.31}$$



(a) Corner parabola

(b) Edge middle parabola

Figure 4.19: Quadratic Bernstein polynomials over a triangle (in Cartesian coordinates).

The Bernstein polynomials of degree 2 can be divided in two groups, the group with indices $(2,0,0),(0,2,0),(0,0,2)$, and the group with indices $(1,1,0),(0,1,1),(1,0,1)$. The first group are parabolas that are maximum in one corner and have both value and gradient 0 at the opposite edge in the triangle. The second group have their maximum halfway one of the edges and are zero in all the other edges. Figure 4.19 shows the shape of these types of functions. These six Bernstein polynomials are linearly independent parabolas, and therefore, they form a basis for all possible parabolas. That is, it is possible to construct any required parabola p as a linear combination of the six Bernstein parabolas,

$$p(x,y) := b(\boldsymbol{\zeta}) = \sum_{\substack{i+j+k=2,\\ i,j,k\geq 0}} b_{i,j,k} B^2_{i,j,k}(\boldsymbol{\zeta}). \tag{4.32}$$

The coefficients $b_{i,j,k}$ are called the Bézier ordinates of the polynomial $p(x,y)$. The polynomial $b(\boldsymbol{\zeta})$ can be schematically represented by filling in all 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 4.20. 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 4.19 with Figure 4.20.

Using the schematic notation using Bézier ordinates, is it very straightforward to define a parabola over a triangle. This notation turns out to be very convenient for multiple reasons. From Equation 4.29 and corresponding Figure 4.19, it can be seen that all Bernstein polynomials with their Bézier ordinates not on edge on edge e are zero on that edge and have 0 gradient on that edge as well. Therefore, the gradient and value at a 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 it the parabolas over each of the elements are required to connect smoothly, i.e. have the same value and gradient over the edge, it is only required that they have the same Bézier ordinates on that edge. The 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 4.21.

It was shown that a 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 nodes V_j, with $j \neq i$,

$$\phi_i^q(V_j) = 0, \quad \text{if } i \neq j, \tag{4.33}$$

$$\nabla\phi_i^q(V_j) = 0, \quad \text{if } i \neq j. \tag{4.34}$$

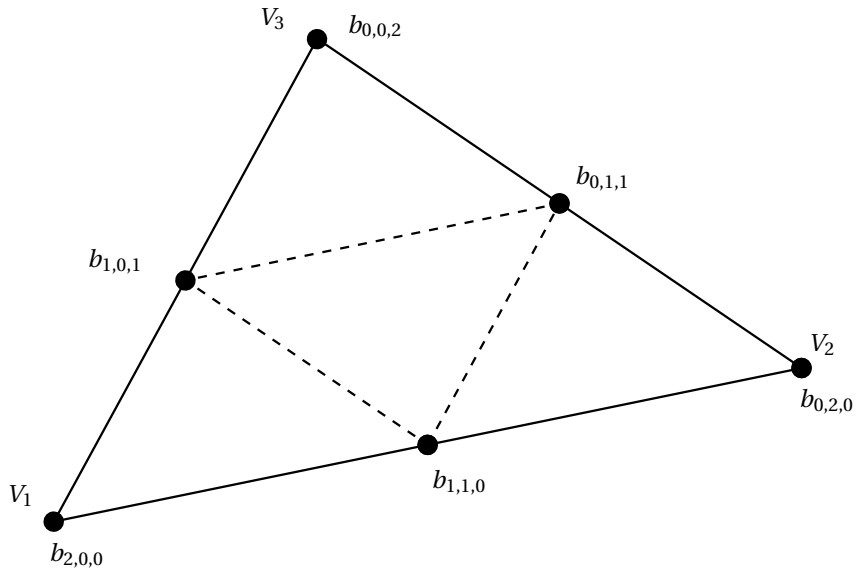


Figure 4.20: The Bézier ordinates of a parabola over a triangle. Each Bézier ordinate is positioned at the location where its corresponding Bernstein polynomial has its maximum (see Figure 4.19.

This also implies 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

$$(\alpha_{i,q}, \beta_{i,q}, \gamma_{i,q}) = \left(\phi_i^q(V_i), \frac{\mathrm{d}\phi_i^q}{\mathrm{d}x}(V_i), \frac{\mathrm{d}\phi_i^q}{\mathrm{d}y}(V_i)\right), \tag{4.35}$$

with the basis function value, its derivative to x and its derivative to y at V_i. Note that this procedure happens 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 4.19, from this it follows that $\phi^1(V) + \phi^2(V) + \phi^3(V) = 1$, and therefore,

$$\alpha_1 + \alpha_2 + \alpha_3 = 1, \tag{4.36}$$
$$\beta_1 + \beta_2 + \beta_3 = 0, \tag{4.37}$$
$$\gamma_1 + \gamma_2 + \gamma_3 = 0. \tag{4.38}$$

These properties must always hold. 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,

$$c_1\alpha_1 + c_2\alpha_2 + c_3\alpha_3 = \hat{f}(x, y), \tag{4.39}$$
$$c_1\beta_1 + c_2\beta_2 + c_3\beta_3 = \hat{f}_x(x, y), \tag{4.40}$$
$$c_1\gamma_1 + c_2\gamma_2 + c_3\gamma_3 = \hat{f}_y(x, y). \tag{4.41}$$

Next, let the x-coordinates of the three vertices $Q_j = (X_j, Y_j)$, $j = 1, 2, 3$ 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:

$$X_1\alpha_1 + X_2\alpha_2 + X_3\alpha_3 = x, \tag{4.42}$$
$$X_1\beta_1 + X_2\beta_2 + X_3\beta_3 = 1, \tag{4.43}$$
$$X_1\gamma_1 + X_2\gamma_2 + X_3\gamma_3 = 0, \tag{4.44}$$

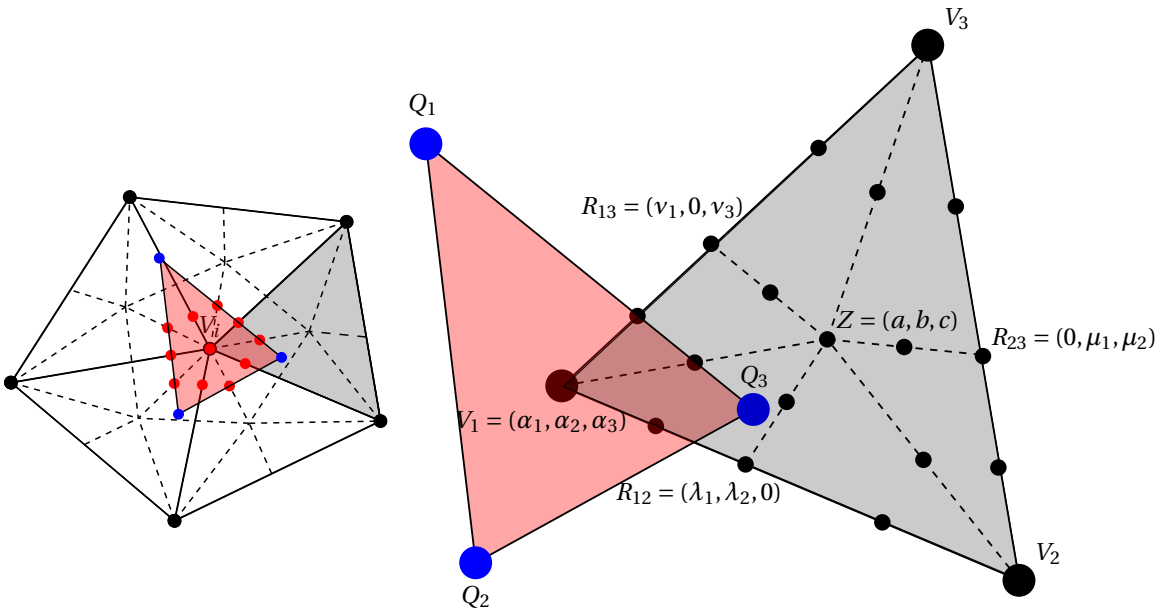


Figure 4.21: Condider again a node V_i of the triangulation Δ. The molecule Ω_i is shown on the left, along with a suitable PS-triangle, marked red. Next, consider only the triangular element marked in grey and the PS-triangle on the right. The locations of the Bézier ordinates are marked to construct 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 nodes.

and

$$Y_1\alpha_1 + Y_2\alpha_2 + Y_3\alpha_3 = y, \tag{4.45}$$
$$Y_1\beta_1 + Y_2\beta_2 + Y_3\beta_3 = 0, \tag{4.46}$$
$$Y_1\gamma_1 + Y_2\gamma_2 + Y_3\gamma_3 = 1. \tag{4.47}$$

The function of the PS-triangle becomes clear, 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 found by considering Equations 4.36, 4.42 and 4.45. Likewise for β with Equations 4.37, 4.43 and 4.46 and for γ with Equations 4.38, 4.44 and 4.47. This results in the following three systems of equations,

$$\begin{bmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{4.48}$$
$$\begin{bmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{4.49}$$
$$\begin{bmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}. \tag{4.50}$$

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 4.1. 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

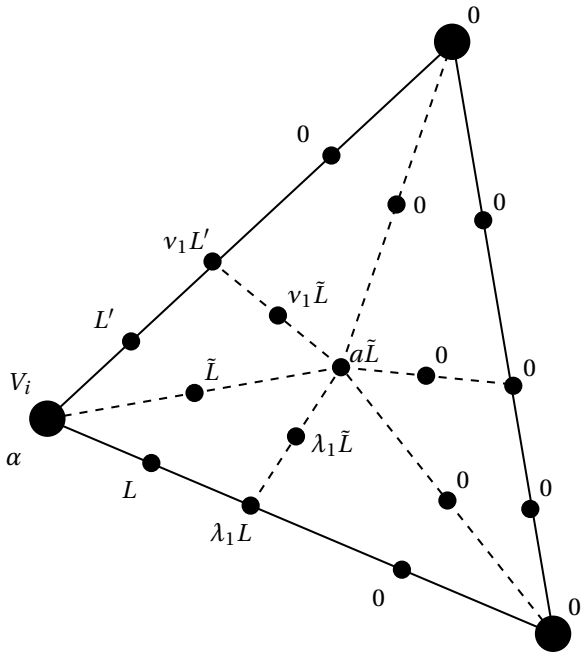


Figure 4.22: The Bézier ordinates for basis function ϕ_i^1, corrsponding to the vertex V_i considered in Figure 4.21. There are three different basis functions, note that for each of these basis functions, α, β and γ should correspond with the triplet $(\alpha_q, \beta_q, \gamma_q)$ corrsponding to PS-triangle vertex Q_q.

approach can be used to find the inverse and give an explicit formula for β and γ,

$$A^{-1} = \begin{bmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1} = \frac{\begin{bmatrix} X_2 - Y_3 & X_2 - X_3 & X_2 Y_3 - X_3 Y_2 \\ Y_3 - Y_1 & X_3 - X_1 & X_3 Y_1 - X_1 Y_3 \\ X_1 - Y_2 & X_1 - X_2 & X_1 Y_2 - X_2 Y_1 \end{bmatrix}}{\begin{vmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{vmatrix}}. \tag{4.51}$$

Then the values of β are given by the first column of the inverse of A and γ by the second column

$$\begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \frac{1}{d} \begin{bmatrix} Y_2 - Y_3 \\ Y_3 - Y_1 \\ Y_1 - Y_3 \end{bmatrix}, \tag{4.52}$$

$$\begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} = \frac{1}{d} \begin{bmatrix} X_2 - X_3 \\ X_3 - X_1 \\ X_1 - X_3 \end{bmatrix}, \tag{4.53}$$

in which d is the determinant of the matrix A

$$d = \begin{vmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{vmatrix}. \tag{4.54}$$

From the PS-triangle, three triplets $\left(\phi^q(V), \frac{\partial \phi^q}{\partial x}(V), \frac{\partial \phi^q}{\partial y}(V)\right) = (\alpha_q, \beta_q, \gamma_q)$, $q = 1,2,3$. Each triplets defines the value, and x- and y-derivative of one of the corresponding basis functions ϕ^q in the node V. From these triplets, the full piecewise quadratic basis functions are uniquely defined with the properties of partition unity, non-negativity and C^1-continuity by an algorithm provided by an earlier work of Dierckx [18]. The resulted basis functions assume 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. 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 $R_{1,2} = (\lambda_1, \lambda_2, 0)$, $R_{1,2} = (0, \mu_2, \mu_3)$ and $R_{1,2} = (\nu_1, 0, \nu_3)$. Then the Bézier ordinates are shown in Figure 4.22.

The coefficients in Figure 4.22 are then given in terms of the triplet $(\alpha_q, \beta_q, \gamma_q) = (\alpha, \beta, \gamma)$

$$L = \alpha + \frac{1-\lambda_1}{2}\overline{\beta}, \tag{4.55}$$

$$L' = \alpha + \frac{1-\nu_1}{2}\overline{\gamma}, \tag{4.56}$$

$$\tilde{L} = \alpha + \frac{b}{2}\overline{\beta} + \frac{c}{2}\overline{\gamma}, \tag{4.57}$$

with

$$\overline{\beta} = \beta(x_2 - x_1) + \gamma(y_2 - y_1), \tag{4.58}$$

$$\overline{\gamma} = \beta(x_3 - x_1) + \gamma(y_3 - y_1). \tag{4.59}$$

Figure **??** showns an example of a resulting piecewise quadratic 2D basis function.

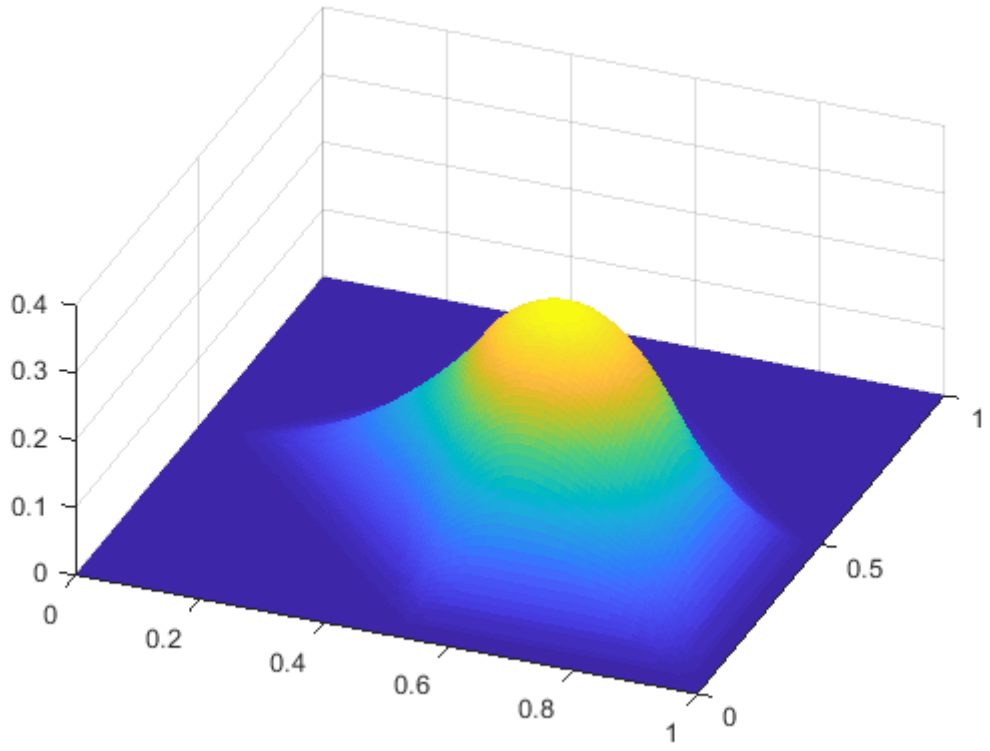


Figure 4.23: A C^1 continuous B-spline basis function over a triangular grid.

5

RESULTS

In this chapter, MPM with B-spline basis functions on a triangular grid will be compared to MPM with standard Lagrange basis MPM. It is expected that B-spline MPM will lead to a higher order of spatial convergence than standard Lagrange MPM. In this thesis, only the B-splines of order $p = 2$ have been investigated yet. These basis functions are expected to show third order spatial convergence, whereas the standard piecewise linear Lagrange basis functions should result in only second order convergence. This difference in convergence happens during the projection of the material properties to the grid. First, it will be shown that the B-spline basis functions indeed show third order convergence for reconstructing a function. Afterwards, B-spline MPM is compared to standard Lagrange MPM.

5.1. L_2-PROJECTION ON B-SPLINE BASIS.

In this section, the constructed second order basis functions are validated. An arbitrary analytic functions will be given, and this function is then projected onto a base of 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 nodes both in the x-direction and in the y-direction. The spacing h is therefore $1/n$, which is also the typical element length. In total, the domain is descretised using n^2 nodes, ordered in equally large squares. From these nodes, a Delaunay triangulation forms the triangular grid. The discretisation of the grid is shown in Figure 5.1.

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

$$f(x,y) = \sin(\pi x)\sin(\pi y) \tag{5.1}$$

will be projected onto the basis of B-splines. The L_2-projection was also used in Equation 3.21. For convenience, the process is repeated here. Let

$$\hat{f}(x,y) = \sum_{j=1}^{n^2}\sum_{r=1}^{3} c_j^r \phi_j^r(x,y) \tag{5.2}$$

be the L_2-projection of $f(x,y)$ onto the B-splines basis functions ϕ_i^q. As there are n^2 nodes in the discretisation, and there are 3 basis functions per node, there are $3n^2$ degrees of freedom. For the L_2-projection $\hat{f}(x,y)$, the following equations must hold,

$$\int_\Omega \hat{f}(x,y)\phi_i^q\,\Omega = \int_\Omega f(x,y)\phi_i^q\,\mathrm{d}\Omega \tag{5.3}$$

$$\int_\Omega \sum_{j=1}^{n}\sum_{r=1}^{3} c_j^r \phi_j^r \phi_i^q\,\mathrm{d}\Omega = \int_\Omega f(x,y)\phi_i^q\,\mathrm{d}\Omega \tag{5.4}$$

$$c_j^r \sum_{j=1}^{n}\sum_{r=1}^{3}\int_\Omega \phi_j^r \phi_i^q\,\mathrm{d}\Omega = \int_\Omega f(x,y)\phi_i^q\,\mathrm{d}\Omega \tag{5.5}$$

$$\boldsymbol{M}\boldsymbol{c} = \boldsymbol{f}, \tag{5.6}$$

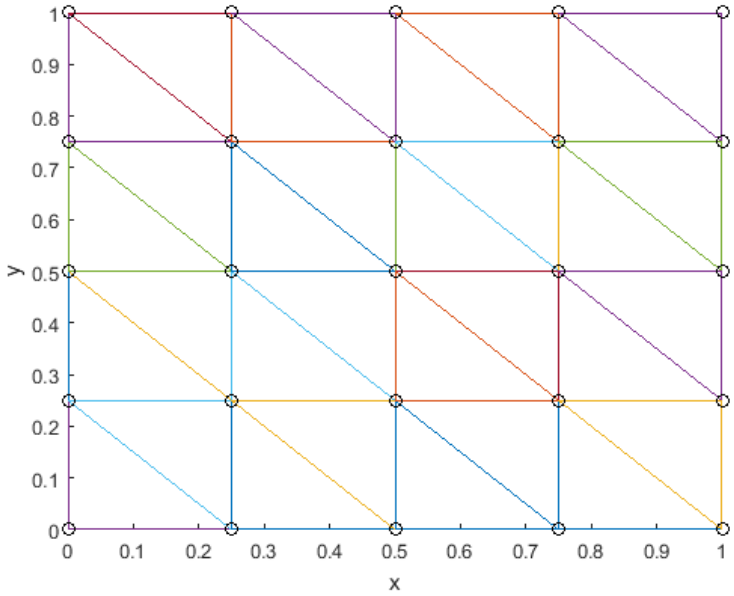


Figure 5.1: A discretisation of the unit square domain Ω, discretised using $n = 5$ nodes in each direction, which results in the shown triangular discretisation Δ.

in which $\boldsymbol{M}_{iq,jr} = \int_\Omega \phi_j^r \phi_i^q \, \mathrm{d}\Omega$ and $\boldsymbol{f}_{iq} = \int_\Omega f(x,y)\phi_i^q \, \mathrm{d}\Omega$. Solving this system for $\boldsymbol{c}$ yield the coefficients for the projection $\hat{f}(x,y)$. The integration of the integrals is done with Gaußintegration with many Gaußpoints, 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,

$$E(h) = \sqrt{\int_\Omega \left(\hat{f} - f\right)^2 \, \mathrm{d}\Omega}. \tag{5.7}$$

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

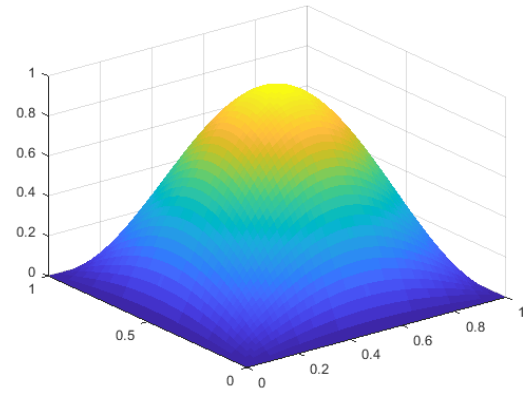
For various values of h, the projection error has been calculated. In Table 5.1, the results are presented. The table shows a reduction in error of about a factor $8 - 10 \approx 2^3$ when the spacing h is halved. This implies that $E(h) = \mathbb{O}(h^3)$, so the B-spline basis shows third order convergence as expected. The basis function, its projection and the error is also shown in Figure 5.2

Table 5.1: The L_2-error for the projection of the function $f(x,y) = \sin(\pi x)\sin(\pi y)$ for different discretisations of the domain.

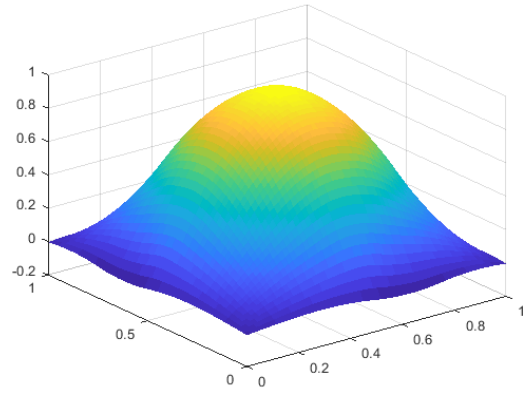
n	n^2	Degrees of freedom	h	E(h)
3	9	27	1/2	1.18575e-02
5	25	75	1/4	1.14331e-03
9	81	243	1/8	1.31622e-04
17	289	867	1/16	1.61689e-05

5.2. BENCHMARK TESTING

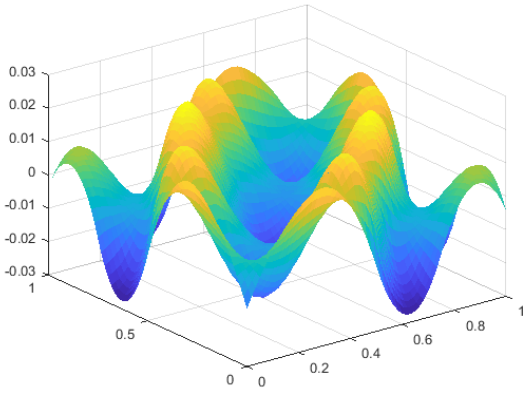
In the last section, the spatial convergence of the B-spline basis was validated. In this section, a benchmark is considered to test the spatial convergence of these basis functions when applied in MPM. The benchmark considered will be a vibrating bar, with both ends fixed. This problem can be described 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 **??** shows a schematic illustration of the situation. The figure also shown a way to represent this one-dimensional problem as a one-dimensional problem. By adding a second, but obsolete direction, the problem may be solved with



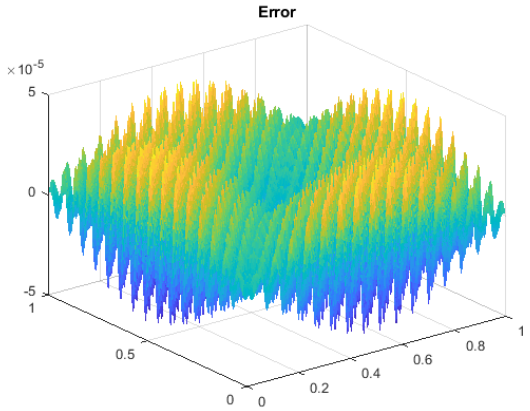
(a) Function $f(x,y) = \sin(\pi x)\sin(\pi y)$.



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



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



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

Figure 5.2: 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 $n = 17$.

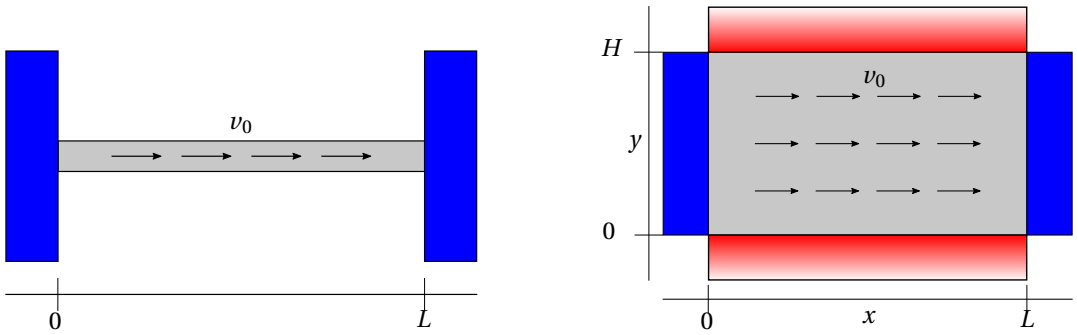
MPM with the described quadratic B-splines over triangulations.

The one-dimensional problem has are homogeneous Dirichlet boundary conditions at $x = 0$ and at $x = L$. 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 $y = 0$ and $y = H$, 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. With these conditions, it is expected that the two-dimensional solution will be only dependent of x, and be uniform 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 for 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 5.2. The parameters are chosen such that the deformations in the bar will be small, i.e. the normal strain in the x-direction $\epsilon_{11} \leq 1\%$. Deformation in the y-direction should not happen, as there will only be an initial velocity in the x-direction. Also, as the Poisson ratio is zero, deformation in the x-direction does not lead to deformation in the y-direction.

The analytical solution for the x-displacement in the vibrating bar with small deformations is derived from the wave equation,

$$\frac{\partial^2 u_x}{\partial t^2} = \frac{E}{\rho}\frac{\partial^2 u_x}{\partial x^2}. \tag{5.8}$$



(a) 2D (b) 1D

Figure 5.3: The benchmark problem of a vibrating bar. This problem can be described as a one-dimensional problem on the left, or as a two dimensional problem on the right. The boundaries on the left and right of the bars are homogeneous Dirichlet for both x and y displacement, the top and bottom boundary conditions are homogeneous Dirichlet for only the y-displacement. An initial velocity field v_0 is imposed over the bar.

Table 5.2: 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	$[kg/m^3]$
Young's modulus	E	100	$[Pa]$
Poisson Ratio	ν	0	$[-]$
Length	L	25	$[m]$
Width	H	1	$[m]$
Velocity	v_0	0.1	$[m/s]$

The boundary conditions are described as

$$\begin{aligned} u_x(0,y,t) &= 0, \\ u_y(0,y,t) &= 0, \\ u_x(L,y,t) &= 0, \\ u_y(L,y,t) &= 0, \\ u_x(x,0,t) &= 0, \\ u_x(x,H,t) &= 0. \end{aligned}$$

The initial conditions are given by

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

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

$$u_x(x,y,t) = \frac{v_0}{\omega}\sin(\omega t)\sin\left(\tfrac{\pi}{L}x\right), \tag{5.9}$$

with

$$\omega = \frac{\pi\sqrt{\frac{E}{\rho}}}{L}. \tag{5.10}$$

Having defined the problem and the analytic solution, the numerical solution using the 2D B-spline MPM can be compared to classic MPM solutions and to the analytic solution. For the 2D B-spline MPM, define a triangular grid with the nodes distributed in 2 rows and 26 columns. Initialise the particles in a similar fashion, 4 rows and 75 columns (3 columns per square element pair), see Figure 5.4 for an illustration of the grid.

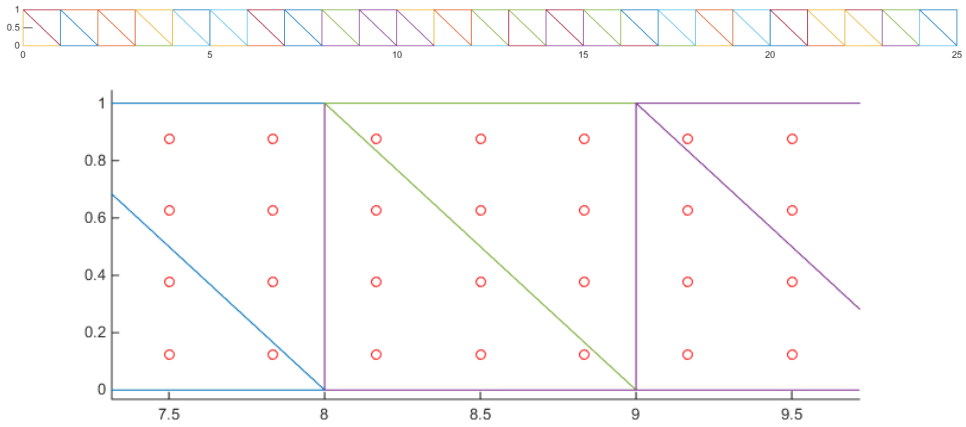


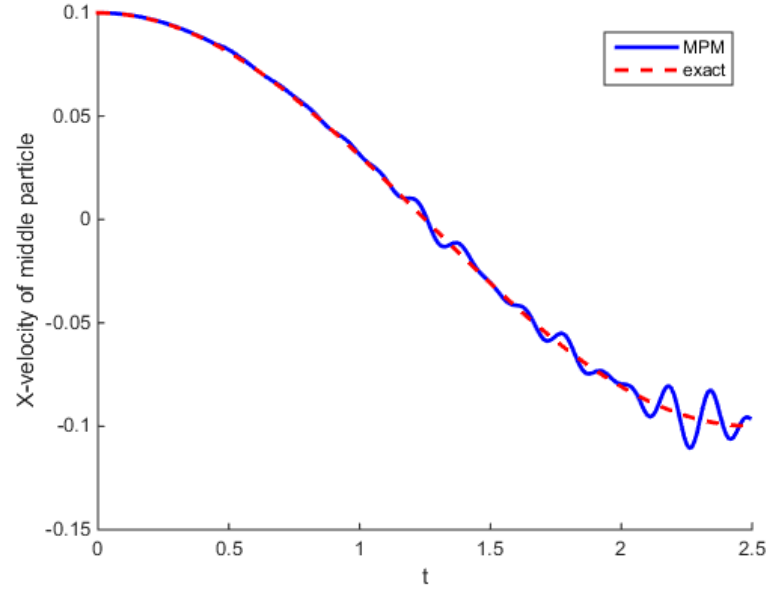
Figure 5.4: The grid for 2D B-spline MPM modelling of a vibrating bar. The triangulation of the domain is shown above, two elements are shown below, along with the initial particle positions, of which there are 96 in each element.

Using this grid, a solution can be calculated with the B-spline basis function MPM and with the classic Lagrange MPM. First consider the classic Lagrange MPM. The numerical solution for the velocity of a particle in the middle of the bar is shown in Figure 5.5. Note that the error in the upper figure is much larger than the error in the lower figure. The sole reason for this is that grid crossing occurs in the upper problem, but not in the lower one. If grid crossing does not occur, the function seems to approximate the exact solution very well.

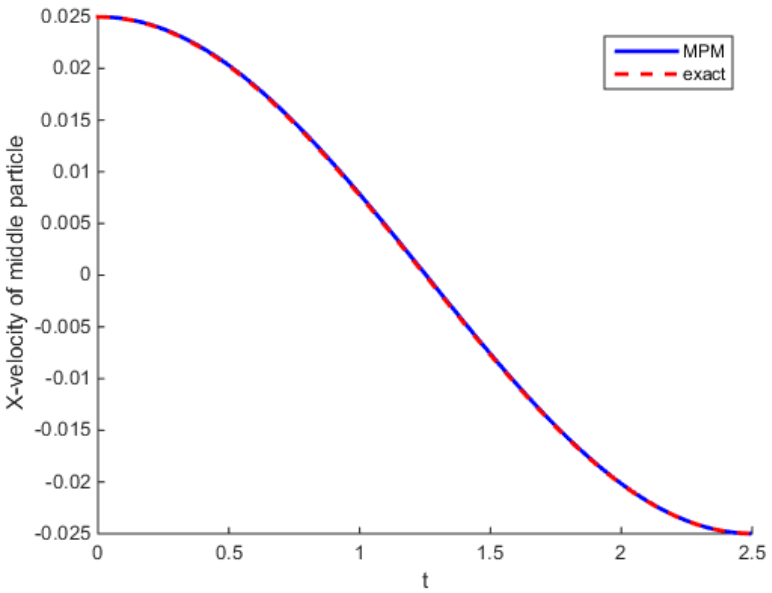
Next consider the same grid, but with B-spline MPM. B-spline MPM has three basis function per node, whereas classic Lagrange MPM has only one basis function per node. Therefore, the same grid yield three times as many degrees of freedom compared classic Lagrange MPM. Therefore, these two cases cannot be compared fairly. First, however, some results for B-spline MPM are considered in general in Figure 5.6. Note that the lower figure converges very well, whereas the upper shows a greater error. The difference lies in the number of particles used per element. For the upper solution, only 6 particles were used per element, which is apparently not enough. This can be explained by considering that each element was divided in 6 sub-element, over each of which a parabola was defined. In order to integrate these properly, many more integration points should be used. It should be noted that grid crossing did happen during this simulation, but no errors seem to appear due to this.

In order to fully check on the convergence rate of B-spline MPM, a fixed time should be chosen, and the L_2-error of the solution should be measured and compared to the number typical element length or the number of degrees of freedom used. This has not yet been done. Questions remain on what quantity to measure, for example, the velocity, stress, displacement or the position. Also the error is not only dependent of the number of elements used, but also of the time step used, the number of particles used, the machine precision and the discretisation of the equations. Currently, a velocity in the y-direction can be measured, which was not originally expected, see Figure 5.7. A y-velocity appear, although the y-velocity seems to go down when more particles per element are used. The frequency with which the particles vibrate correspond to the typical length in the y-direction. This unexpected velocity is most probably a consequence of the quadrature error that comes with numerical integration using the particles.

Besides the number of particles per element, it has been observed that the time step size also contributes to the error. This could be solved by taking a very small time step, which should eliminate the error. Unfortunately, taking very small time steps takes a long time for computations, and therefore, these experiments have not yet been run.

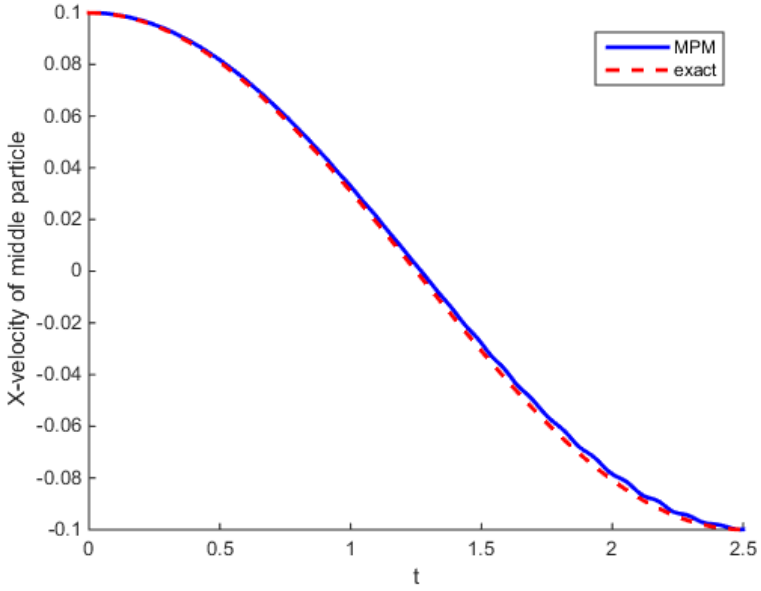


(a) Piecewise linear basis functions, $v_0 = 0.1$, with grid crossing.

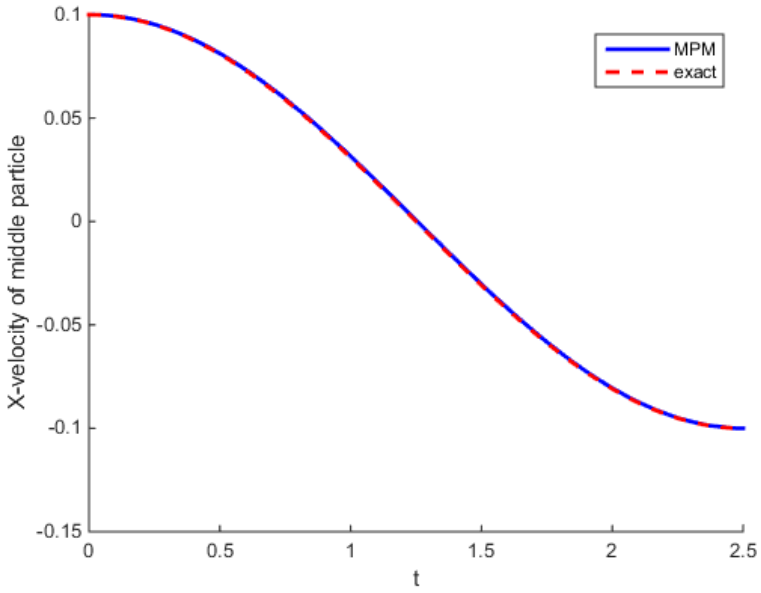


(b) Piecewise linear basis functions, $v_0 = 0.01$, no grid crossing.

Figure 5.5: The solution for the velocity of the vibrating bar using the grid in Figure 5.4. The problem for the upper figure had an initial velocity of $v_0 = 0.025$, in which grid crossing occurred. The lower problem was identical, except that the initial velocity was set to $v_0 = 0.01$ and grid crossing did not occur.



(a) Quadratic B-spline basis functions, 6 particles per element.



(b) Quadratic B-spline basis functions, 24 particles per element.

Figure 5.6: The solution of the vibrating bar using the grid in Figure 5.4 using B-spline MPM. The upper result was obtained by using 6 particles per element as shown in Figure5.4, for the number of particles per element was doubled in both the x- and y-direction to 24 particles per element.

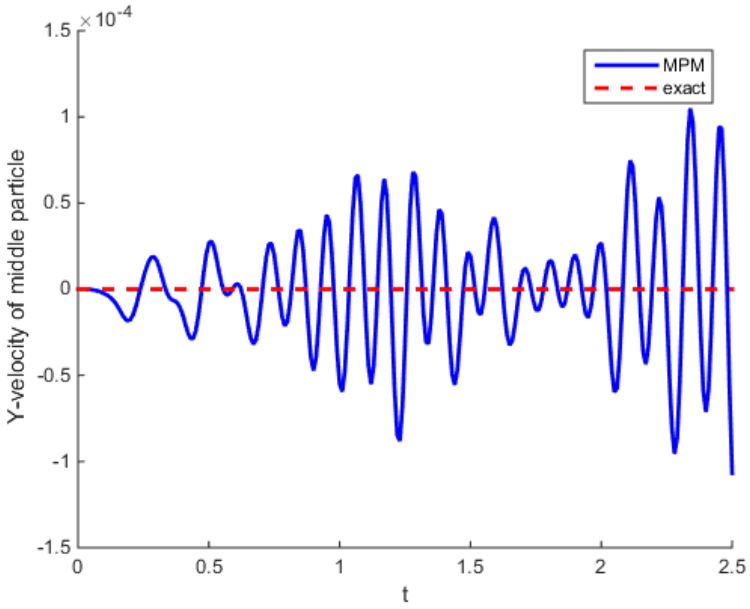


Figure 5.7: The y-velocity of a particle in the middle of the domain, simulated with B-spline MPM.

6

CONCLUSION

A B-spline basis for MPM on arbitrary triangulations may provide MPM with higher order spatial convergence. L_2-projections of functions onto a base of quadratic B-splines show the expected higher order convergence. However, the implementation of a B-spline basis in MPM proves cumbersome, and it is not obvious how to extend quadratic B-splines to higher order B-splines. Test cases have yet to be run to investigate if a B-spline basis will lead to higher order spatial convergence in MPM.

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

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

[9] M. Geers, *Fundamenttals of Deformation and Linear Elasticity*.

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

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

[12] 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).

[13] W. Hu and Z. Chen, *A multi-mesh mpm for simulating the meshing process of spur gears,* Computers & structures **81**, 1991 (2003).

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

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

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

[17] 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).

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

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