Scientific Computation

Vinh Phu Nguyen
Alban de Vaucorbeil
Stephane Bordas

The Material Point Method

Theory, Implementations and Applications

Springer

The Material Point Method

Scientific Computation

Vinh Phu Nguyen · Alban de Vaucorbeil ·
Stephane Bordas

The Material Point Method

Theory, Implementations and Applications

Springer

Vinh Phu Nguyen
Department of Civil Engineering
Monash University
Clayton, VIC, Australia

Alban de Vaucorbeil
Institute for Frontier Materials
Deakin University
Geelong, VIC, Australia

Stephane Bordas
University of Luxembourg Campus
Kirchberg
Luxembourg, Luxembourg

ISSN 1434-8322 ISSN 2198-2589 (electronic)
Scientific Computation
ISBN 978-3-031-24069-0 ISBN 978-3-031-24070-6 (eBook)
https://doi.org/10.1007/978-3-031-24070-6

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Computer simulations have become an integral tool in engineering, ranging from civil and mechanical engineering to material sciences and beyond. While the finite element method (FEM) has long been the standard framework for simulations, it reaches its limits when dealing with problems involving large deformation and fractures. To overcome these limitations, meshless methods (MMs) such as smoothed-particle hydrodynamics (SPH) and the material point method (MPM) have emerged as a promising alternative. MPM, in particular, combines the advantages of FEM and MMs, representing the material by a set of particles overlaid on a background mesh. This approach has been successful in simulating a wide variety of large-deformation and complicated engineering problems.

The book not only re-examines previous contributions but also organizes them in a coherent fashion and anticipates new advancements. Sample algorithms for benchmark problems are available on the book's website, allowing researchers and graduate students to modify them and develop their own solution algorithms for specific problems. The goal of this book is to provide students and researchers with a theoretical and practical knowledge of the material point method for the simulation of engineering problems, and to promote further in-depth studies in the field.

This book aims at being a comprehensive guide to the Material Point Method that focuses on its use in solving problems in mechanics, physics and engineering. The book contains nine main chapters that build on each other to provide a detailed understanding of the MPM.

Chapter 1 provides an introduction to the Material Point Method and its advantages over other numerical methods. It also gives an overview of the topics covered in the book.

Chapter 2 covers the mathematical foundations of the MPM. It discusses the principles of continuum mechanics, the kinematic description of motion and the integration algorithms used in the MPM.

Chapter 3 presents different MPM versions that basically adopt different shape functions, e.g. hat functions, B-splines, Bernstein, GIMP and CPDI. Also treated is a new formulation called generalized particle in cell (GPIC) which combines the FEM and MPM that takes advantages of both methods.

Chapter 4 covers the constitutive models used in the MPM. This includes linear elastic isotropic materials, hyperelastic solids and elasto-plastic materials. The chapter also discusses the Johnson-Cook flow model and the algorithm used to compute damage.

Chapter 5 provides some implementation details such as particle generation for simple geometries and for images, initial and boundary conditions, MPM with unstructured grids and visualization of MPM results.

Chapter 6 presents a tutorial MPM code written in MATLAB. This code serves to illustrate the MPM algorithms discussed in the book and it can be used to solve one, two and three dimensional problems.

Chapter 7 describes Karamelo—an open-source parallel C++ package for the material point method. This code can be used to solve large-scale problems as it can be run on multiple processors using MPI. With such an efficient code, we present three dimensional simulations to demonstrate the capability of the MPM in solving large deformation solid mechanics problems.

Chapter 8 presents some advanced topics such as contacts and fracture. Both frictionless and frictional contact are discussed. One notable application of these contact algorithms is the simulation of scratch test—a popular mechanical test to measure a solid's hardness. We then discuss fracture modeling in the framework of the MPM. Application of the MPM to model large strain ductile fracture of metals is then provided.

Chapter 9 discusses the mathematical analysis of the MPM regarding its stability and accuracy. We discuss the conservation of energy and momenta in various MPM variants. We study the convergence behavior of all MPM variants discussed in this book for a problem involving only compression/tension deformation and another problem involving simple shear with superimposed rotation.

Chapter 10 discusses fluid/gases modeling, membrane modeling and heat conduction. With the information provided in this chapter, one can carry out fluid-structure interaction simulations, air-bag simulations and thermo-mechanical simulations.

In addition to the content of the main chapters, the book has several appendices that provide supplementary information. Appendix A discusses the strong and weak form of the momentum equation and their equivalence. Appendix B presents derivation of various CPDI basis functions. Some useful utilities such as how to use an open-source computer algebra system (SageMath and SymPy) to derive CPDI functions, how to use remote machines to run large-scale simulations and consistent units are given in Appendix C. Appendix D gives a short but practical presentation of updated and total Lagrangian, explicit dynamics FEM for nonlinear solid mechanics. Appendix E treats implicit dynamics FEM so that it is easier to understand implicit MPM (even though this is not discussed in this book). Finally, we describe another MPM code in Appendix F, now written in Julia—a new high-level dynamic programming language which is easy to use as Python and as fast as C.

The book also includes several simulations related to these topics. The book provides sample algorithms for benchmark problems, which are available on the

book's website. These algorithms can be modified and used to develop custom solution algorithms for specific problems. The book includes MATLAB, Julia and C++ codes, derivations, and references to other studies in the field.

We would like to thank Prof. Deborah Sulsky at University of New Mexico for reading through the first draft and giving comments. Also, the first author acknowledges the fruitful discussions with Dr. Rebecca Brannon at University of Utah when he started working on the MPM.

Clayton, Australia — Vinh Phu Nguyen
Geelong, Australia — Alban de Vaucorbeil
Luxembourg, Luxembourg — Stephane Bordas

Reference

Zhang, X., Chen, Z., Liu, Y.: The Material Point Method-A Continuum-Based Particle Method for Extreme Loading Cases. Academic Press, Cambridge (2016a)

Contents

Chapter 1
Introduction

The aim of this introductory chapter is to provide an overview of what is the material point method—the topic of the book—and its application domains. We start with a brief introduction to the field of computational sciences and engineering to explain the role of computer simulations using a computational model (Sect. 1.1). The MPM is one of such computational models. The role of experiments cannot be underestimated even for a computational scientist and engineer (Sect. 1.2). Then, we briefly discuss initial-boundary value problems and numerical methods in Sect. 1.3. Next, we talk about mesh-based and mesh-free methods in Sect. 1.4 as the MPM adopts tools from these two classes. This is followed by Sect. 1.5 where we will present a picture of the MPM. Applications of the MPM in various engineering and sciences fields are given in Sect. 1.6. Open source and commercial MPM codes are presented in Sect. 1.7. The layout of the book is given in Sect. 1.8. Finally, our notations are explained in Sect. 1.9.

1.1 Computational Sciences and Engineering

The topic of this book falls within the scope of computational sciences and engineering (CSE). CSE is a relatively new discipline that deals with the development and application of *computational models*, often coupled with high-performance computing, to solve complex problems arising in engineering analysis and design (computational engineering) as well as in natural phenomena (computational science). CSE has been described as the "third mode of discovery" next to theory and experimentation.

Within the realm of CSE these are steps to solve a problem. First, a mathematical model that best describes the problem is selected or developed. This step of model development is done manually by people with sufficient mathematical skills. A majority of mathematical model is developed using calculus (see Remark 1 for a history account) and thus they are continuous models not suitable for digital computers.

V. P. Nguyen et al., *The Material Point Method*, Scientific Computation,
https://doi.org/10.1007/978-3-031-24070-6_1

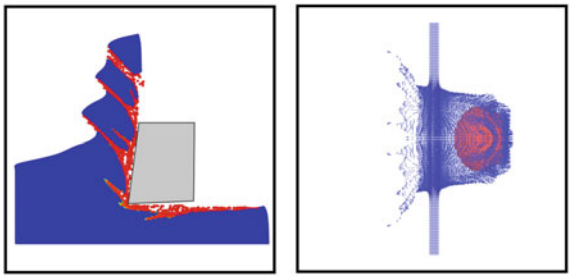

Fig. 1.1 Computer experiments: experiments done with computational models on a digital computer

Second, a computational model of this mathematical model is derived. A computational model is an approximation to the mathematical model and is in a discrete form which can be solved using computers. Third, this discrete model is implemented in a programming language (Fortran in the past and C++ and Python nowadays) to have a computational code or platform. For solid mechanics, popular computational platforms are Abaqus and LS-Dyna. Finally, these computational platforms are used to perform *computer simulations* or *computer experiments* (Fig. 1.1).

Computer simulations are not only useful to solve problems too complex to be resolved analytically, but are also increasingly replacing costly and time consuming experiments. Furthermore, they can provide tremendous information at scales of space and time where experimental visualization is difficult or impossible. And finally, simulations also have a value in their ability to predict the behavior of materials and structures that are yet to be created; experiments are limited to materials and structures that have already been created.

This book presents MPM models i.e., discrete models based on the material point method for the problem of understanding and prediction of the deformation of solids and fluids that undergo very large deformation. The mathematical model for this problem is based on the theory of *continuum mechanics*, see e.g. Malvern (1969). We also discuss computer implementation of these MPM models and provide tutorial MPM codes written in Matlab and Julia and an efficient MPM platform named `Karamelo` which can replace contemporary FE packages such as LS-Dyna and Abaqus for certain problems.

Remark 1 We would like to discuss briefly why calculus is so dominant in sciences and engineering. It all started with the works of Galilei and Newton who discovered that the laws of nature can be unreasonably well described by mathematics, particularly calculus. If the motion of heavenly bodies can be modeled using mathematics, then it is logical to apply it to humanity problems. This was exactly what the geniuses like Bernoulli brothers, Euler, Lagrange, Cauchy had done some 300 years ago. These men developed partial differential equations that can model a wide range of phenomena such as the deformation of fluids, gases and solids. It is the models described by these PDEs that put men on the moon, give us cell phones, computers, radio. Or television. Or ultrasound for expectant mothers, or GPS for lost travelers.

Remark 2 Calculus has two parts: differential calculus and integral calculus. The latter is interesting as we go from finiteness to infinitum. And to solve it numerically, we do the reverse: from infinitum back to finiteness. We replace a solid with infinite number of degrees of freedom by a mesh consisting of just a finite number of degrees of freedom.

1.2 The Role of Experiments in CSE

It is certain that a computational model requires experiments to obtain parameters used in the model. To emphasize the vital role of experiments in sciences and engineering, we consider the interesting article of Boyce et al. (2016) that presents the Sandia fracture challenge to the computational fracture community. The challenge involves the simulation of the fracture of a steel sample of complex geometry. Only a minimum experimental data (tensile test of a steel coupon) was provided to the analysist. Different research groups, who participated in the challenge, used all existing fracture models and none provided a match with the experiment.

Therefore simulations are simply insufficient and thus a combined experiment-simulation programme should be pursued for any problem. It is interesting to know that R. W. Clough, the exact man who coined the term 'finite element method' some 70 years ago, stopped working on the method and switched to experiments (Clough 1980).

1.3 One Dimensional Wave Equation

In science and engineering, one commonly seeks the response to some excitations of a certain kind of system. This system can be mechanical, chemical, biological …To this end, it is common practice to adopt a mathematical model for the system and try to solve it. Usually the model equations are partial differential equations (PDE); for example, the Navier-Stokes equations in fluid mechanics, or the momentum conservation equations in solid mechanics. These differential equations together with both the initial and boundary conditions constitute an initial-boundary value problem (IBVP). In general, solving a boundary value problem by classical analytical methods is almost impossible. Therefore, an approximate solution to the IBVP is sought.

Approximate solutions to an IBVP are obtained by transforming the PDE into a set of algebraic equations. This is achieved by discretizing the space and time domain. Common spatial discretization methods include mesh-based methods such as the finite element method (FEM), the finite volume method (FVM), the finite difference method (FDM) and meshless or meshfree methods (MMs). Time discretization is mostly based on finite differences e.g. the forward Euler method and the leaf-frog method.

In this section, we provide an overview of how to use a numerical method to obtain approximate solutions to IBVPs. This discussion is not meant to be rigorous, but rather to present the basic concepts of numerical methods and a general procedure to go from PDEs, which are difficult to solve, to algebraic equations, which can be handled quite well by nowadays computers. We have decided to present methods using a weak form as the MPM follows this so-called Galerkin method.

For a simple demonstration of the basic concepts in numerical methods, let's consider the one dimensional momentum equation, that governs the deformation of a solid object due to applied external forces:

$$\rho \frac{\partial^2 u}{\partial t^2} = E \frac{\partial^2 u}{\partial x^2} + \rho b \tag{1.1}$$

where $u(x, t)$ is the displacement field, E the Young modulus of the material, ρ the density and b the body force. The spatial domain is $0 \leq x \leq L$ and the time domain is $0 \leq t \leq T$.

For the case of zero body force (i.e. $b = 0$) the above equation becomes the well known one dimensional wave equation written as:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \quad c = \sqrt{\frac{E}{\rho}} \tag{1.2}$$

Remark 3 Solving Eq. (1.2) for $u(x, t)$ with a given c is called *a forward problem*. Inversely, determining c so that Eq. (1.2) has a solution matching a predefined $\bar{u}(x, t)$ is coined *an inverse problem*.

In order for a PDE to have unique solutions, initial and boundary conditions have to be provided. For example, the so-called Dirichlet boundary conditions read

$$u(0, t) = a, \quad u(L, t) = b, \quad t > 0 \tag{1.3}$$

where a, b are some constants. Note that there exists other types of boundary conditions such as Neumann condition and Robin condition. As Eq. (1.2) involves second derivative with respect to t, two initial conditions are required which are given by

$$u(x, 0) = f(x), \quad \dot{u}(x, 0) = g(x) \tag{1.4}$$

where $\dot{u} := du/dt$ and f, g are some functions.

Putting all the above together we come up with the following initial-boundary value problem

$$\begin{aligned} &\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} && \text{(wave equation)} \\ &u(0, t) = a, \quad u(L, t) = b, \quad t > 0 && \text{(boundary conditions)} \\ &u(x, 0) = f(x), \quad \dot{u}(x, 0) = g(x) && \text{(initial conditions)} \end{aligned} \tag{1.5}$$

of which approximate solutions are sought for using a numerical method. Equation (1.5) is called a *strong form* of the wave equation. Some numerical methods such as FDM or collocation methods work directly with this strong form even though they are often less accurate compared with Galerkin methods—those that employ a weak formulation—and unstable. However, these methods are quite efficient as no numerical integration is needed.

The finite element methods (or generally Galerkin based methods) adopt a weak formulation where the partial differential equations are restated in an integral form called the *weak form*. A weak form of the differential equations is equivalent to the strong form. In many disciplines, the weak form has a physical meaning; for example, the weak form of the momentum equation is called the principle of virtual work in solid/structural mechanics.

To obtain the weak form, one multiplies the PDE i.e., the wave equation in this particular context, with an arbitrary function $w(x)$, called the *weight function*, and integrate the resulting equation over the entire domain. That is

$$\int_0^L \left[\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2}\right] w(x)\mathrm{d}x = 0, \quad \forall w(x) \text{ with } w(0) = w(L) = 0 \tag{1.6}$$

The arbitrariness of the weight function is crucial, as otherwise a weak form is not equivalent to the strong form. In this way, the weight function can be thought of as an enforcer: whatever it multiplies is enforced to be zero by its arbitrariness.

Using the integration by parts for the second term, the above equation becomes

$$\int_0^L \frac{\partial^2 u}{\partial t^2} w(x)\mathrm{d}x + c^2 \int_0^L \frac{\partial u}{\partial x}\frac{\partial w}{\partial x}\mathrm{d}x = 0 \tag{1.7}$$

where the spatial derivative of the unknown field, $u(x, t)$, was lowered from two to one. It is a great achievement by a simple derivation as constructing high order approximations of u, so that second derivatives are computable, is much more difficult than constructing linear approximations. Furthermore, the second term is now symmetric, this is significant as the resulting matrix will be symmetric. Symmetric matrices possess nice properties e.g. less storage and real eigenvalues.

The weak form of the wave equation is thus given by: find the smooth function $u(x, t)$ such that

$$\begin{aligned} \int_0^L \frac{\partial^2 u}{\partial t^2} w(x)\mathrm{d}x + c^2 \int_0^L \frac{\partial u}{\partial x}\frac{\partial w}{\partial x}\mathrm{d}x = 0 \\ u(0, t) = a, \quad u(L, t) = b \\ u(x, 0) = f(x), \quad u(x, 0) = g(x) \end{aligned} \tag{1.8}$$

for all $w(x)$ with $w(0) = w(L) = 0$.

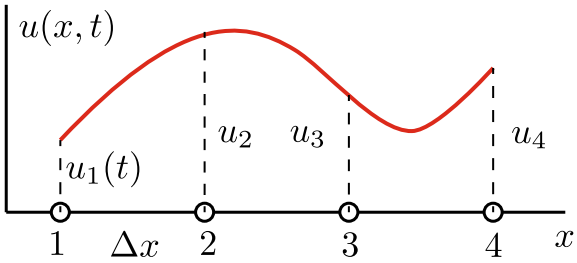

Fig. 1.2 Spatial discretization in one dimension

The basic idea of numerical methods in solving PDEs is to discretize the spatial and temporal domain i.e., instead of working with infinite number of points (or nodes) within the domain of interest $[0, L] \times [0, T]$, one first discretize the spatial domain into a finite number of points x_I, $I = 1, 2, \ldots, n$. Next, the unknown function $u(x, t)$ is approximated using the values of u evaluated at those discrete points x_I (Fig. 1.2), and this approximation is then substituted into the weak form i.e., Eq. (1.8) to obtain a set of ordinary differential equations (ODEs). Finally using any time integration methods of ODEs to advance in time. At this stage, the PDE has been completely transformed into a discrete form—a system of algebraic equations—which can be easily solved by digital computers. This is known as the method of lines. There exists methods which involve full discretization in both space and time, but they are less popular and not further discussed in this book.

The approximation of the unknown field $u(x, t)$ is written as

$$u(x,t) \approx u^h(x,t) = \sum_I^n N_I(x) u_I(t) \tag{1.9}$$

where $N_I(x)$ are the approximation functions or shape functions in the FEM context and $u_I(t)$ denotes the value of u at point I at time instant t and constitutes the unknowns to be solved. The support of node I is defined as the set of points where $N_I(x) \neq 0$. Usually, only a few points are within the support of a given node and thus the shape function is said to have a compact support. And this compact support is crucial to the computational efficiency of the method: the resulting matrices are sparse not full. If the support of $N_I(x)$ is the whole domain, the method is called a spectral method. After having obtained u_I, Eq. (1.9) is used to compute the function at any other points.

Even though there are many choices for the weight functions w, in the Bubnov-Galerkin method, which is the most commonly used method at least for solid mechanics applications, the weight function is approximated using the same shape functions as u. That is

$$w(x,t) = \sum_I^n N_I(x) w_I \tag{1.10}$$

where w_I are the nodal values of the weight function; they are not functions of time.

Now, the numerical solution of the weak form of the wave equation i.e., Eq. (1.8) is thus given by: find u_J such that

$$\int_0^L \left(N_I(x)\ddot{u}_I\right)\left(N_J(x)w_J\right)\mathrm{d}x + c^2\int_0^L \left(\frac{\partial N_I}{\partial x}u_I\right)\left(\frac{\partial N_J}{\partial x}w_J\right)\mathrm{d}x = 0 \tag{1.11}$$

for all w_J. Note that we have used the Einstein summation rule: indices which are repeated twice in a term are summed, see Sect. 1.9 for detail.

The arbitrariness of w_J results in the following system of ordinary differential equations

$$\begin{bmatrix} \int_0^L N_1N_1\mathrm{d}x & \int_0^L N_1N_2\mathrm{d}x & \dots & \int_0^L N_1N_n\mathrm{d}x \\ \int_0^L N_2N_1\mathrm{d}x & \int_0^L N_2N_2\mathrm{d}x & \dots & \int_0^L N_2N_n\mathrm{d}x \\ \vdots & \vdots & \vdots & \vdots \\ \int_0^L N_nN_1\mathrm{d}x & \int_0^L N_nN_2\mathrm{d}x & \dots & \int_0^L N_nN_n\mathrm{d}x \end{bmatrix} \begin{bmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \vdots \\ \ddot{u}_n \end{bmatrix}$$

$$+ c^2 \begin{bmatrix} \int_0^L dN_1dN_1\mathrm{d}x & \int_0^L dN_1dN_2\mathrm{d}x & \dots & \int_0^L dN_1dN_n\mathrm{d}x \\ \int_0^L dN_2dN_1\mathrm{d}x & \int_0^L dN_2dN_2\mathrm{d}x & \dots & \int_0^L dN_2dN_n\mathrm{d}x \\ \vdots & \vdots & \vdots & \vdots \\ \int_0^L dN_ndN_1\mathrm{d}x & \int_0^L dN_ndN_2\mathrm{d}x & \dots & \int_0^L dN_ndN_n\mathrm{d}x \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{1.12}$$

where $dN_I = dN_I/dx$ is the first spatial derivative of the shape function N_I. The integrals in the above equation are called weak form integrals. For this simple 1D problem, they can be exactly computed, but generally, numerical integration is used to evaluate these integrals. We refer to Sect. 2.4 for a discussion on numerical integration.

And Eq. (1.12) can be cast in the following compact equation using a matrix notation

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{0} \tag{1.13}$$

where $\mathbf{u}$ and $\ddot{\mathbf{u}}$ are the vector of displacements and accelerations of the whole problem, respectively. They are one dimensional array of length n; $\mathbf{M}$ and $\mathbf{K}$ are the mass and stiffness matrix—matrices of dimension $n \times n$.

Equation (1.13) is referred to as the *semi-discrete equation* as the time has not been yet discretized. Any time integration methods for ODEs can be used to discretize Eq. (1.13) in time. A time integration scheme is called *implicit* if one has to solve a system of algebraic equations to obtain $\mathbf{u}$ at a time instant t. This system of algebraic equations is very large for practical problems: it is not uncommon to encounter a system of millions of unknowns. On the other hand, it is called *explicit* if one can get $\mathbf{u}$ without solving any system of algebraic equations. Implicit time integration schemes allow large time increments i.e., there are fewer time steps to resolve whereas explicit schemes require small time increments. Generally, explicit schemes are preferred for fast transient problems such as impact simulations.

For time discretization, the time interval $[0, T]$ is partitioned into a number of time steps Δt i.e., the semi-discrete equation is evaluated at discrete time instants $t^m = (m-1)\Delta t$. Assuming that we are at time t and need to advance to time $t + \Delta t$. By using the central difference scheme, which is the most commonly used explicit time integration method, we have

$$\mathbf{u}^{t+\Delta t} = \Delta t^2 \ddot{\mathbf{u}}^t + 2\mathbf{u}^t - \mathbf{u}^{t-\Delta t} \tag{1.14}$$

which allows us to obtain the displacements at $t + \Delta t$ upon substitution of Eq. (1.13) for $\ddot{\mathbf{u}}^t$

$$\mathbf{u}^{t+\Delta t} = -\Delta t^2 \mathbf{M}^{-1}\mathbf{K}\mathbf{u}^t + 2\mathbf{u}^t - \mathbf{u}^{t-\Delta t} \tag{1.15}$$

To avoid inversion of the mass matrix, a technique known as *mass lumping* is often adopted to make $\mathbf{M}$ diagonal.

The final step is to impose Dirichlet boundary conditions e.g. $u(0, t) = a$. If the shape functions N_I have been constructed such that they satisfy the Kronecker delta property then it is pretty straightforward to impose Dirichlet conditions: one simply override the displacements computed in Eq. (1.15) by the prescribed values. In this specific case, simply setting $u_1 = a$ and $u_n = b$ does the job. Shape functions are said to satisfy the Kronecker delta property when they fulfill the following equation

$$N_I(x_J) = \delta_{IJ}, \quad \delta_{IJ} = \begin{cases} 1 & \text{if } I = J \\ 0 & \text{otherwise} \end{cases} \tag{1.16}$$

Therefore, condition $u(0) = a$ becomes $u(0) = \sum_I N_I(0)u_I = u_1 = a$.

What value should be assigned for n or in other words, how many nodes/points should we use? That is the eternal question of computational engineer. There is no theorem which says n should be such and such. A rule of thump is n should be big to have accuracy and not so big to reduce the cost. Practically, one pick an n, do the simulation and evaluate a certain quantity of interest (e.g. the maximum displacement or maximum stress) against analytical solutions (if any) or experimental

data. If a large difference exists, then double n and repeat until a convergence has been obtained. If no solution is available, then one needs to compare the numerical solutions of at least two resolutions (two different n) and they should be close to each other.

Up to this point, how the shape functions N_I are constructed is not yet discussed. In the next section, we discuss this construction of shape functions.

1.4 Mesh-Based and Meshfree Methods

Spatial discretization methods can generally be divided into groups: mesh-based methods (Sect. 1.4.1) and meshless or meshfree methods (Sect. 1.4.2). The three most common mesh-based methods are finite element method (FEM), finite volume method (FVM) and finite difference method (FDM). Herein we focus on FEM since it is the most widely used and commercially available method to date for solid mechanics. Furthermore, the material point method can be considered a variant of FEM.

1.4.1 Mesh-Based Methods

Since its inception about 70 years ago (Courant 1943; Clough 1960),[1] the finite element method has been used with great success in many fields with both academic and industrial applications. They have been the primary computational methodologies in engineering computations for more than half a century. The basic idea is to divide the domain of interest (generally with a complex shape) into a (finite) number of sub-domains called elements (Fig. 1.3). These elements have simple geometry e.g. triangles or quadrilaterals in two dimensions and tetrahedra in three dimensions. The elements are connected at nodes. A field quantity, such as the displacement field, is interpolated by a polynomial defined over the elements. Integrals in the weak form e.g. the stiffness matrix or force vectors are evaluated over individual elements using a quadrature rule (e.g. Gauss quadrature). For deformable solids of which behavior is history dependent (*inelastic solids*) it is the quadrature points where stresses, strains, history variables (such as equivalent plastic strain, damage variables) are stored.

As a simplest description of the FE shape functions we plot in Fig. 1.4 one dimensional linear and quadratic shape functions. The spatial domain is $[0, 4]$ which is divided into four equidistant elements. In the first case of linear elements, each element has two nodes. There are thus five nodes. For the case of quadratic elements, each element has three nodes–two nodes at the extremities and one mid-side node.

[1] For an interesting discount on the history of FEM, we refer to Clough (1980), Gander and Wanner (2012).

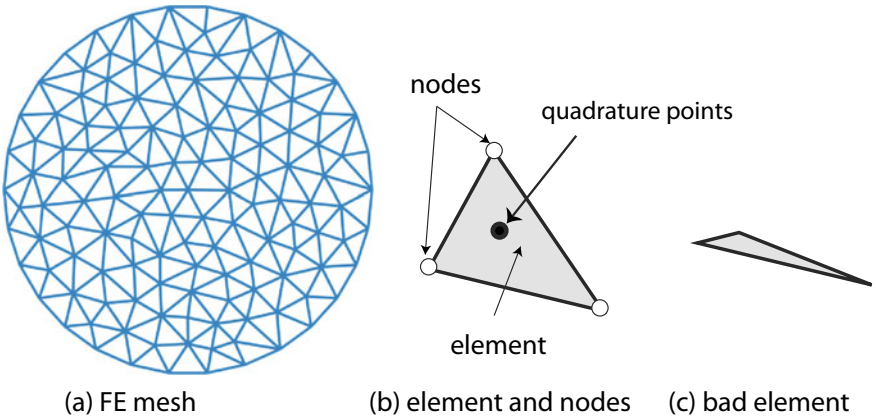

Fig. 1.3 The finite element method: domain is discretized into a number of elements

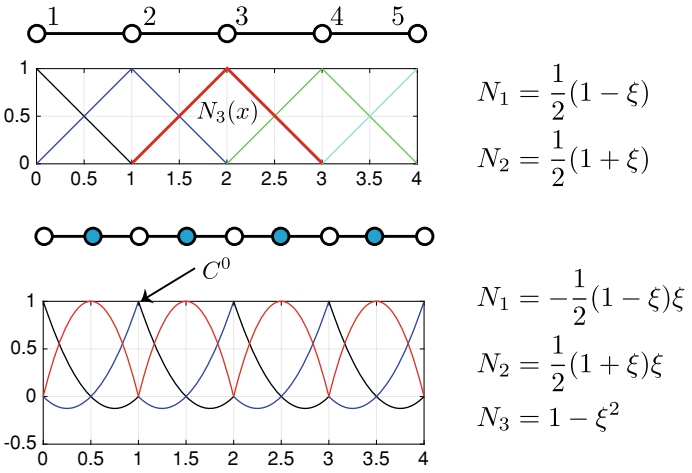

Fig. 1.4 Finite element shape functions in one dimension: linear elements (top) and quadratic elements (bottom). FE shape functions are polynomials defined in the so-called parent domain which is $[-1, 1]$ in one dimension. These shape functions satisfy the Kronecker delta property

Note that even though quadratic functions are smooth they are only C^0 across the element boundaries.

Remark 4 C^0 means that only the functions are continuous across element boundaries but their derivatives are not. This property poses a great challenge in solving PDEs with high order spatial derivatives which occur frequently in structural mechanics (i.e., PDEs that govern the behaviour of beams/plates/shells often involve fourth order derivatives of the primary unknown field).

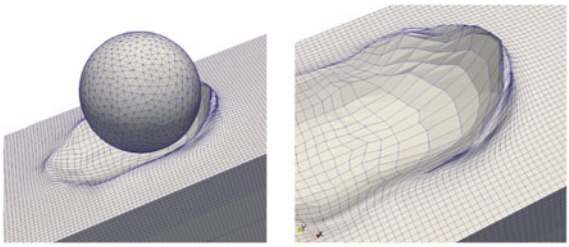

Fig. 1.5 Element distortion in the FEM: simulation of material scratch

The availability of the elements provides a natural way to evaluate the weak form integrals. For example, consider a function f in two dimensions, one can write

$$\int_{\Omega} f(x, y)\mathrm{d}\Omega = \sum_{e} \int_{\Omega_e} f(x, y)\mathrm{d}\Omega = \sum_{e} \sum_{g} f(x_g, y_g) w_g \tag{1.17}$$

where subscript e denotes the elements and subscript g denotes the integration points. The fact that FE shape functions are polynomials and the element domains Ω_e align with the support of the shape functions leads to a very accurate computation of the weak form integrals. This is in sharp contrast to meshfree methods (and also to the material point method) to be discussed shortly.

For problems exhibiting large deformation (precisely large strain) e.g. simulation of manufacturing processes such as extrusion and molding operations, the elements inevitably become distorted which leads to higher errors and even premature termination of the program (Fig. 1.5). This issue of element distortion is often solved using a technique called *remeshing* in which a new mesh with quality elements replaces the old mesh with distorted elements. There are two major problems with remeshing. First, it is a time and human labour consuming task, which is not guaranteed to be feasible in finite time for complex three-dimensional geometries. Second, remeshing requires mapping or projection of information from the old to the new mesh, a step that inevitably introduces error, particularly for inelastic materials involving history variables. The more history variables a material model has the more error this remeshing step will induce.

Another difficulty of FEM is the conversion of a continuum (Fig. 1.6a) into a finite element mesh of good quality (Fig. 1.6b), in a reasonable amount of time and involves least user intervention. This is because solid geometries are created in a CAD (Computer Aided Design) software, and FE simulations are carried out in a FE software that accepts only FE meshes. Meshfree methods were born to remove the remeshing burden of FEM. But it can alleviate the mesh burden as well; even though isogeometric analysis pioneered by Hughes et al. (2005) is probably better for this issue.

Fig. 1.6 Conversion from a CAD (**a**) to a FE mesh (**b**)

(a) A solid in a CAD system (b) Corresponding FE mesh

1.4.2 Meshless Methods

Meshless methods are so named as the space is discretized into a number of points, or particles, in which each point interacts with its neighboring points in a flexible manner: not via a rigid mesh as in the FEM. It should be noted that up to now, there is no unified framework for meshfree methods. This is reflected by the plethora of existing methods[2] in the literature. The oldest developed method is Smoothed Particle Hydrodynamics (SPH) introduced by Gingold and Monaghan (1977), Lucy (1977) which was used for modeling astrophysical phenomena without boundaries such as exploding stars and dust clouds. Nowadays, the SPH is a popular simulation technique in various engineering and science fields. Next, the Generalized Finite Difference Method of Liszka and Orkisz (1980) was proposed followed by the Diffuse Element Method (DEM) by Nayroles et al. (1992), the Element Free Galerkin (EFG) by Belytschko et al. (1994); the Material Point Method (Sulsky et al. 1994), the Reproducing Kernel Particle Method (RKPM) by Liu et al. (1995); the $h-p$ cloud method (Duarte and Oden 1996); the Natural Element Method (Sukumar et al. 1998); the Meshless Local Petrov Galerkin (MLPG) by Atluri and Zhu (1998); the Maximum entropy (Arroyo and Ortiz 2006); the Particle Finite Element Method (PFEM) of Idelsohn et al. (2006), Sabel et al. (2014), the optimal transport meshfree method (OTM) of Li et al. (2010), just to name but the most popular meshfree methods.

In general, all meshless methods share the same characteristic: the domain of interest is completely discretized by nodes (or points or particles) as illustrated by Fig. 1.7. The concept of connectivity in mesh-based methods is replaced by domain of influence which indicates nodes fall within the support of a given node. A meshfree method is characterized by the following items

Collocation or Galerkin formulation DEM, EFG, RKPM, MLPG, NEM are Galerkin meshfree methods i.e., weak form based methods. Galerkin MMs are stable, accurate but computationally expensive. Collocation MMs are ones that approximate the strong form of a PDE (i.e., the PDE itself). One notable collocation MM is the SPH.[3] SPH is classified as a meshfree particle method or more

[2] The wikipedia page on meshfree methods lists about 30 methods and new methods are being created, https://en.wikipedia.org/wiki/Meshfree_methods.

[3] Note that since there are different SPH approximation rules, there thus exists different forms of discrete SPH equations. All of them are used in practice.

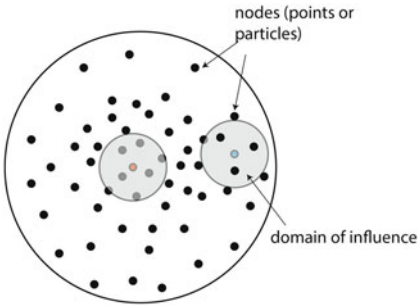

Fig. 1.7 Meshless discretization by a cloud of points

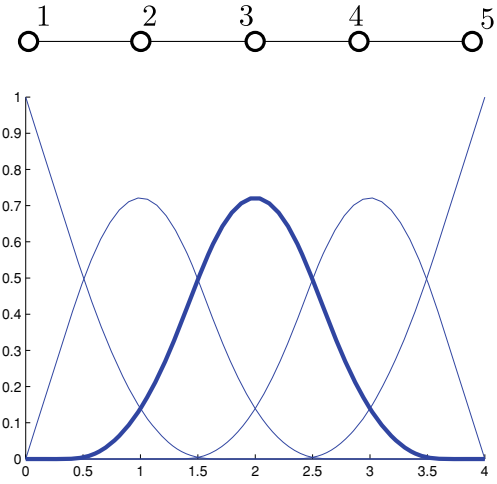

Fig. 1.8 Typical smooth meshfree basis functions: one dimensional MLS functions defined for a set of 5 nodes equally spaced. The highlighted function is the one of the middle node

precisely meshfree Lagrangian particle method because the particles are endowed with physical quantities such as density and volumes. In other words, interpolation points and particles coincide;

Interpolation method Different MMs adopt different high order interpolation techniques: EFG/DEM/MLPG utilizes the MLS (Moving Least Square), in RKPM and SPH kernel estimates are the interpolations. Generally meshfree functions/derivatives are smooth, cf. Fig. 1.8, but computationally expensive than FE functions;

Numerical integration For Galerkin meshfree methods a numerical integration scheme is needed of which one can mention background element/cell technique, nodal integration etc.;

Imposition of essential boundary conditions Most meshfree basis functions do not satisfy the Kronecker delta property thus making enforcement of essential boundary conditions a daunting task.

For a comprehensive consideration of MMs, we refer to various review articles, for instance Belytschko et al. (1996), Babuška et al. (2002), and Nguyen et al. (2008)

which provide computer implementation details including enrichment for weak and strong discontinuities, Hsieh and Pan (2014) for an essential software framework for MMs, Doblaré et al. (2005) (focused on the applications of meshfree methods in biomechanics) and the textbook of Liu and Liu (2003), Liu (2002), Li and Liu (2007), Fasshauer (2007). The last textbook gave a historical account of meshfree approximation theories such as moving least square, radial basis functions etc.

Some MMs have been incorporated into commercial FEA packages such as Abaqus (SPH), LS-Dyna (SPH and EFG), ANSYS (SPH). There exists also purely meshless packages such as NoGrid that implements the finite pointset method for applications in fluid dynamics.

1.5 A Brief Introduction to the MPM

The Material Point Method is one of the latest developments in particle-in-cell (PIC) methods. The first PIC technique was developed in the early 1950s by Harlow (1964), Harlow (2004) at Los Alamos National Laboratory and was used primarily in fluid mechanics. The first PICs suffered from excessive energy dissipation which was overcome in 1986, by Brackbill and Ruppel with the introduction of FLIP-the Fluid Implicit Particle method (Brackbill and Ruppel 1986; Brackbill et al. 1988). In computer graphics, PIC/FLIP has become the de facto standard method for fluid simulations (Zhu and Bridson 2005). The FLIP was later modified and tailored for applications in solid mechanics by Sulsky and her co-workers (Sulsky et al. 1994, 1995b) at University of New Mexico and has since been referred to as the Material Point Method (Sulsky and Schreyer 1996).

In FLIP, the strain and stresses are stored at the cell centers. Yet, in the MPM, they are carried by the particles themselves. Thus, the MPM particles carry the full physical state of the material including position, mass, velocity, volume, stress, temperature etc.. Note that in PIC, the particles carry only position and mass.

The MPM is built on the two main concepts already used in PIC that are the use of Lagrangian material points that carry physical information, and a background Eulerian grid used for the discretization of continuous fields (i.e., displacement field). For a short description of the Lagrangian and Eulerian descriptions, see Fig. 1.9.

1.5.1 Lagrangian Particles and Eulerian Grid

In the MPM, a continuum body is discretized by a finite set of n_p Lagrangian material points (or particles) that are tracked throughout the deformation process. The terms *particle* and *material point* will be used interchangeably throughout this book. In the original MPM, the subregions represented by the particles are not explicitly defined. Only their mass and volume are tracked. In advanced MPM formulations such as GIMP or CPDI, the shape of these subregions is tracked though. Each material point

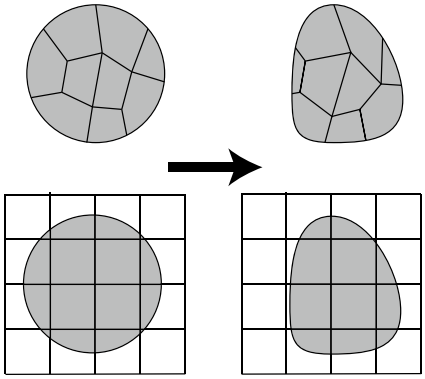

Fig. 1.9 Lagrangian description (top) versus Eulerian description (bottom). In a Lagrangian description, the grid is attached to the solid and thus it deforms during the deformation process of the solid. Each point in the grid is always associated to just one single material point, thus making modeling history-dependent materials easy. The solid boundary is also well defined. However, the grid can become distorted. On the other hand, the Eulerian grid is fixed in space and material flows through the mesh. Mesh distortion never happens

has an associated position $\mathbf{x}_p^t$ $(p = 1, 2, \ldots, n_p)$, mass m_p, density ρ_p, velocity $\mathbf{v}_p$, deformation gradient $\mathbf{F}_p$, Cauchy stress tensor $\boldsymbol{\sigma}_p$, temperature T_p, and any other internal state variables necessary for the constitutive model. Collectively, these material points provide a Lagrangian description of the continuum body. As each material point contains a fixed amount of mass at all time, mass conservation is automatically satisfied.

The original MPM developed by Sulsky is effectively an updated Lagrangian scheme. For this MPM, the space that the simulated body occupies and will occupy during deformation is discretized by a grid, called background grid where the equation of balance of momentum is solved. On the other hand, in the Total Lagrangian MPM (de Vaucorbeil et al. 2020), the background grid covers only the space occupied by the body in its reference configuration. We refer to Fig. 1.10 for a graphical illustration of material points overlaying on a Cartesian grid for both ULMPM and TLMPM. The grid is fixed and the particles are moving over it (Fig. 1.11).

The use of a grid has the following benefits. First, it allows the method to be quite scalable by eliminating the need for directly computing particle-particle interactions. Second, collision is treated easily through this background Eulerian grid (in fact, a non-slip, non-penetration contact is inherent in the method). Third, the momentum equation is solved on the grid, and as there are many fewer grid points than particles, this is a very efficient substitution. Most often, a fixed regular Cartesian grid is used throughout the simulation for efficiency reasons.