Department of Building Technology & Structural Engineering

Aalborg University

Title: The Virtual wave flume – With the SPH method
Theme: Study of the SPH method and using a 2-D model of a wave flume
Project period: 9th -10th semester, February 1 2007– February 1, 2008

Author:

Mads-Peter Hansen

Synopsis

The Smoothed Particle hydrodynamics (SPH) method have been used for modelling CFD problems the last 15 years. The SPH method is, unlike more traditional methods like FEM, not bound by a mesh making it possible to easily handle problems with large deformation or distortion of a free surface. The SPH method is tested for use in building a virtual wave flume corresponding to the ones available at AAU. Sampled time series of wave properties and wave impact from the wave flume are compared with the new virtual flume generated using the open source program SPHysics.

In order to understand the different concepts and parameters in SPH, a number of one dimensional cases are tested and presented as simple examples of how SPH works. Furthermore is the basis of particle approximation with SPH derived and used to approximate Navier-Stokes Equations.

Supervisor: Lars Andersen
Pages in the main report: 97
Pages in the appendix: 74
Number of issues: 5

Preface

This report contains two parts. The first is a review of the theory behind Smoothed Particle Hydrodynamics (SPH) collected from a number of sources. The theory is introduced with a number of simple examples chosen to present the rudimentary of the method. When it is possible the numerical SPH solution is compared to an analytical solution or alternate numerical methods. Following the examples the theory and concepts of the method are explained. In the second part SPH is used to model a wave flume and the computational model is compared with experimental results.

The report is followed by an appendix in which the experiment and the program used for the numerical model are described in greater detail. References to the appendix are made where additional information is needed. Furthermore is an *Appendix CD* attached containing the report as a *pdf* file, the files to compute the examples and numerical models used the in report and a copy of experimental results. References to the Appendix CD are made when additional information is needed.

In this report the figures and tables are numbered consequently in each chapter and accompanied by an explaining text and reference. As an example the third figure in chapter one is named Figure 1.3. To present the code build with base in the SPH theory, boxes are used. The boxes are numbered like figures and tables. Equations used in the project are also numbered consequently in each chapter and named the same way. The first equation in the second chapter is named (2.1).

The source of reference is divided into three types, namely technical literature, scientific articles and web sites, which are placed in a bibliography at the end of the report. If the source of reference is placed before a full stop it refers to the prior sentence. If it is placed after a full stop it refers to the prior section. References are made in the following way:

Books

Reference:	[Author's name et. al.; Year of publication]
Bibliography:	Full name, Year of publication, Title, Publisher, ISBN

Articles

Reference:	[Author's name et. al.; Year of publication]
Bibliography:	Full name, Title, Journal name, Volume, Page

Websites

Reference: [Site name; Year of downloading]
Bibliography: Site name, Date of retrieval, Full site address

In this project MatLab R2007a has been used to make examples, data processing and figure generating while Fortran 77 and 95 has been used to build the numerical model of the wave flume based on the open source code SPHysics 1.0. Therefore a basic knowledge of programming is expected of the reader. The two programs Microsoft Excel 2003 and WaveLab v2.961 have been used for data sampling and the post processing.

The attached Appendix CD contains the following:

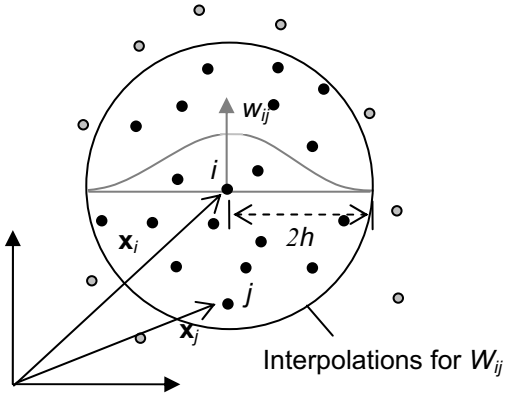
- The report and appendices as PDF files
- Full library of MatLab programs presented in the boxes and the figures of the report
- Microsoft Excel spreadsheets used for calculations and post processing of experimental samples
- WaveLab projects used when sampling and post processing data
- Rewritten Fortran 95 version of the original code for SPHysics 1.0
- Sampled time series from Experiments

Referat – Den virtuelle bølgerende

(This section is an abstract of the report in Danish)

Når marine konstruktioner som moler, vindmøller og bølgeenergianlæg skal projekteres er det ofte nødvendigt at gennemføre en række bekostelige modelforsøg i laboratoriet. Projektet bygger på et ønske om at kunne benytte numeriske modeller til for eksempel at gennemføre dele af et parameter studie og holde det op imod nogle få referenceforsøg. Til det formål undersøges en numerisk metode Smoothed Particle Hydrodynamics (SPH), som første gang nævnes af [Lucy; 1977] men som efter en indledende præsentation af [Monaghan; 1992] har undergået en omfattende udvikling inden for modellering af CFD problemer. Ydermere har udviklingen af hurtigere processorer gjort det realistisk at regne på bølge bevægelsen i en hel bølge rende.

Fordelen ved SPH er at metoden ikke er bundet op på et gitter og derfor kan metoden håndtere de store deformationer og den frie overflade i en bølgebevægelse. Problemet diskretiseres i stedet i en række partikler der hver især repræsenterer massen m_i og vandvolumenet ΔV_i. SPH metoden approksimerer nu en funktion $u(\boldsymbol{x})$ ved hjælp af en interpolationsfunktion $W(\boldsymbol{x_i}-\boldsymbol{x_j},h)$ og interpolation mellem partiklerne i og j. Hvor interpolationslængden h som oftest er en konstant for alle partikler der afgør bredden af $W(\boldsymbol{x},h)$ dvs. hvor stort et interpolations område partiklen nummer i har. Der findes en række mulige interpolations funktioner når fluider skal modelleres men den måske mest kendte er normalfordelingen, der generelt fungerer godt. Princippet i SPH er skitseret herunder sammen med et par af de grundlæggende formler for approksimation af $u(x)$ og den første afledte, bemærk at i og j ikke er tensor notation men i stedet repræsenterer to forskellige partikler:



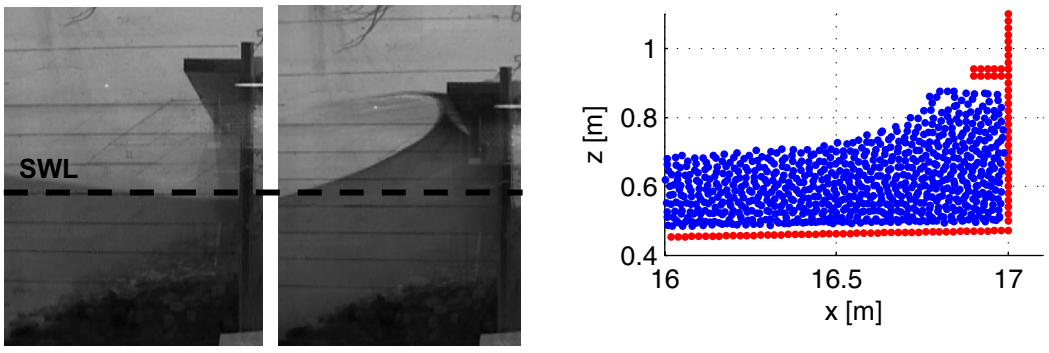
$$\left\langle u\left(\boldsymbol{x}_i\right)\right\rangle = \sum_{j=1}^{N} \frac{m_j}{\rho_j} u\left(\boldsymbol{x}_j\right) W_{ij}$$

$$\left\langle \nabla \cdot u\left(\boldsymbol{x}_i\right)\right\rangle = -\sum_{j=1}^{J} \frac{m_j}{\rho_j} u\left(\boldsymbol{x}_j\right) \cdot \nabla W_{ij}$$

I rapportens første del opstilles en række simple eksempler løst numerisk vha. SPH og den bagvedliggende teori udledes. Formålet er at identificere betydningen af h og $W(\boldsymbol{x}_i-\boldsymbol{x}_j,h)$ samt at sammenligne med de mere kendte numeriske metoder Finite Element og Finite Difference. Med den første del som grundlag er det nu muligt at gå videre til rapportens anden del hvor en virtuel bølgerende modelleres.

I rapportens anden del opstilles en todimensional virtuel bølgerende svarende til den der forefindes i bølgelaboratoriet på Aalborg Universitet. Bølgerenden modelleres ved hjælp af en omskrevet version af programmet SPHysics hvis kildekode i sommeren 2007 blev tilgængelig fra [SPHysics; 2007]. Programmet modellerer bølgerne ved at diskretisere Navier-Stokes ligninger vha. SPH og løse det opstillede ligningssystem eksplicit. Ydermere indeholder SPHysics en række forskellige SPH værktøjer som er blevet udviklet gennem de seneste 15 år og her samlet i et program sammen med det nødvendige script til at generere geometri og en bølgebevægelse.

Den virtuelle bølgerende opstillet i SPHysics er valideret ved at sammenligne den numeriske model med en række forsøg beskrevet i rapportens appendiks. Formålet med valideringen var at fastslå om SPH var i stand til at modellere bølgebilledet i den virtuelle rende, samt modellere bølgeopslaget op under en platform vist herunder med et billede fra forsøgene sammen med den numeriske modellering



Det konkluderes at det ikke er muligt at modellere stabilt med en diskretisering der er høj nok til at lave en præcis sammenligning af forsøg og model i sammenstøds øjeblikket vist herover. I stedet viser en sammenligning af bølgehøjder målt i den virtuelle og virkelige bølgerende en god overensstemmelse hvilket indikerer at bølgebevægelsen og geometriens indflydelse på denne modelleres korrekt. Der er derfor meget at tage fat på i fremtidige projekter.

Table of Contents

Chapter 1

Introduction

When designing breakwaters, offshore wind turbines, wave energy plants or other structures subject to the ocean waves like depicted on Figure 1.1 it is often necessary to make experimental studies in a wave flume. Experimental studies are expensive especially when an extended parameter study is performed where a number of alternate setups are needed in order to determine the optimum design.

Figure 1.1. An ocean wave breaking against a rubble breakwater and a pile outside Helsignør harbour in Denmark during a storm in 2007.

The alternative to extended experimental studies is a computational model of the problem that may be compared to a few tests in order to validate the results. The problem is that many Computational Fluid Dynamics (CFD) methods have trouble with handling models where a free surface is present. To properly model the free surface the method must be able to apply boundary conditions to the free surface, describe the shape and location of the free surface and evolve these with time.

1.1 Computational Methods

The computational/numerical approach is one way to solve fluid problems. In general the pros of using a numerical method are their versatility and the possibility to make any number of different variations of the same problem. But there are a number of different numerical methods available each with their own cons and pros, and while the computational power grows each year more advanced problems becomes manageable. Two well known numerical methods are the Finite Element Method (FEM) and the Finite Difference Method (FDM). Both methods use a grid when solving the governing equations and traditionally FEM uses a Lagrangian grid and FD a Eulerian grid both depicted on Figure 1.2.

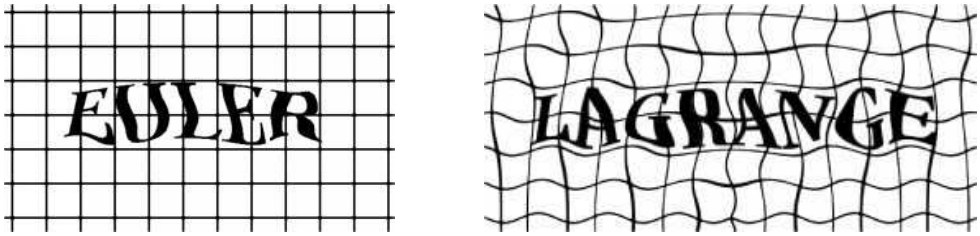


Figure 1.2. On this figure the difference between a Eulerian and a Lagrangian grid is depicted, illustration from [Vesely; 2001].

The Lagrangian approach has a grid attached to the material underneath where each grid node follows the path of the material initially beneath it i.e. the method describes the material. The pros being that with grid nodes along boundaries and interfaces the conditions of free surfaces and moving boundaries are automatically imposed. With irregular grids it is possible to handle irregular geometries. The cons of the Lagrangian when solving CFD problems is its inability to handle large deformations and surfaces that break apart. This leads to a heavily distorted mesh and lack in accuracy.

The Eulerian approach uses a grid fixed in space while the fluids flow across the mesh i.e. the method is a spatial description. Because of the fixed grid the Eulerian method has no problem handling large deformations like it is often the case with fluid problems as the mass momentum and energy is tracked at nodes in the grid or cell boundaries. The cons of the Eulerian method are that it has difficulty handling irregular geometries, moving boundaries or free surfaces because of the fixed grid and its inability to track the time history of points in the fluid.

It is possible to use both methods when solving fluid problems but the cons makes it hard to compute problems with breaking waves and interaction with structures two traits necessary in this project. There have been developments in both meth-

ods in order to make up for these shortcomings. It is possible to interpolate a mathematical description of a free surface into a Eulerian approach and to remesh the Lagrangian grid when deformations grow. Both these improvements demand an extra computational effort and introduce errors.

In order to compute a virtual wave flume with breaking waves it is necessary to use a method that is able to handle large deformations and problems with a free surface and irregular geometries. In this project the chosen approach is to use a particle method. Particle methods are a way to get around the problems that meshes gives traditional and well known methods like the FDM and FEM. In particle methods particle representing is a part of the problem domain and attributes like mass, position, momentum and energy is collected at each particle for the small part of the total domain they represent. The free surface shape and location is described by the location of the particles. If only the fluid is represented by particles the problem domain is simply divided into areas with and without particles, Figure 1.3.

One example of a particle method is the particle in cell method (PIC) where a Eulerian grid is still used to interpolate between the particles. The smoothed particle hydrodynamics method (SPH) is a truly mesh free method with a Lagrangian approach where kernel functions replace the mesh as the mean to interpolate between the particles, Figure 1.3.

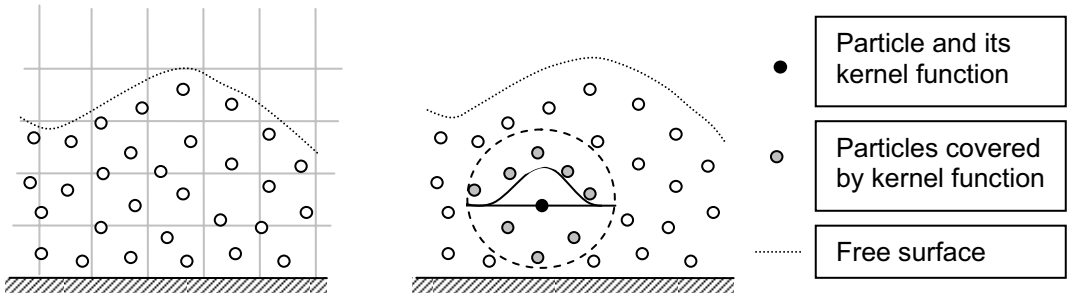


Figure 1.3. The depicted figures are an example of two particle methods in a free surface situation and the methods they use to interpolate between the particle grid or kernel function; PIC (left) and SPH (right).

The PIC and SPH are not the only available particle methods as a wide range of development has been done in recent years but they are among the oldest and most widely examined. It is decided to use the SPH method in order to model waves in a virtual wave flume. The method was chosen because of its lack of grid and because the initial study of the literature showed that the method had already been applied on a lot of similar problems.

1.2 SPH - History and sources

The SPH method is first mentioned in the late seventies by [Lucy; 1977] and has been the target of a great deal of study these past three decades. The SPH method is originally developed for astrophysics where a limited number of particles (planets, stars, galaxies etc.) are needed. It was further developed for the astrophysics during the eighties, and results were published in articles like [Monaghan; 1989] and [Monaghan; 1992].

The first published implementation of the SPH method on a free surface flow is [Monaghan; 1994] where it was demonstrated that the method was capable of modelling a number of free surface problems like a 2-D wave flue depicted in Figure 1.4.

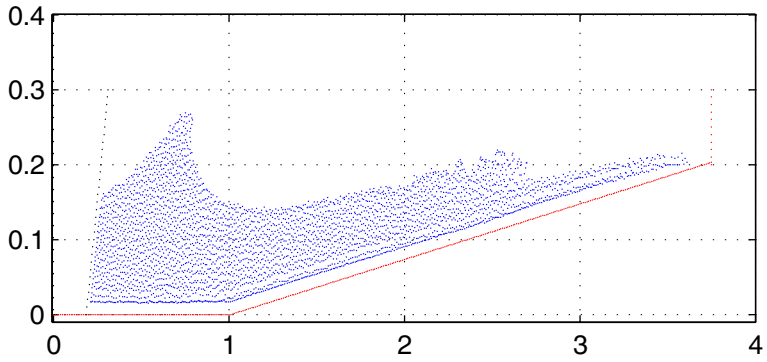


Figure 1.4. Plot of a classic SPH example with breaking waves. The example has been computed using the 2-D version of SPHysics v1.0.

Since the introduction to free surface flows in 1994 extensive work has been conducted to mature the SPH method. The modelling of waves in general have been studied in [Dalrymple et al; 2006] and [Cleary et al; 1999], indicating that the method would be useful for a wide range of wave problems. Adding to this work is wave overtopping studied in [Shao et al; 2006] and a 3-D study of impact with stationary structures, published in [Gómez-Gesteira et al; 2007]. The interaction between fluid and structure is a common denominator for many fluid dynamics problems. In the classic SPH formulation only the pressure on the surface of immobile structures are known, but [Antoci et al; 2007] and [De Vuyst et al; 2005] demonstrated that it is possible to use SPH together with objects deformed by a fast flowing current and coupling the method with another numerical solver, namely the finite element method (FEM). Finally [Colagrossi et al; 2003] has utilized the method to study air entrapment due to violent fluid-structure interaction by a discretization of both water and air while taking advantage of the SPH ability

to handle large deformation and a mix of two elements. If a general introduction to the subject and how to implement it with fluid dynamics is needed, it is available in [Liu; 2003] and [Monaghan; 2005]. An introduction to more specialized subjects like boundary treatment and turbulence is available in [Crespo et al; 2007] and [Issa et al; 2007].

Presently there exists no finished commercial solution using SPH although several are on the way. One developer is [nextlimit.com; 2007]. This is also a good place to witness the full potential of the mesh free method through a number of available animations. A free general SPH code build for fluid dynamics is available together with [Liu; 2003], and recently (July 2007) a free complete SPH program for wave flumes (SPHysics v1.0) has been released. SPHysics is intended to handle a wave flume in 2-D and 3-D and is built in the Fortran 77 language. It is possible to compute a number of basic wave flume situations where two are depicted on Figure 1.4. The program is available on the internet together with a manual explaining parameters and theory behind SPHysics [SPHysics; 2007].

If more knowledge about the SPH method is needed a good place to start is the SPHerics group who among other things hosts a list of SPH literature and articles along with a list of ongoing SPH software projects. [SPHerics; 2007]

1.3 SPH in this report

Smoothed particle hydrodynamics are demonstrated in the first part of this report with the aim to understand the method and as pre study for using it to simulate a free surface flow in a wave flume. The solved differential equations are chosen because they are well known and analytical solutions are possible for a wide range of situations. At the end of each example a simple piece of code is available to demonstrate the basic build of a SPH program used on different problems. The code is written in MatLab but may easily be translated to other programming languages. Following the examples the theory behind SPH is explained and the method is compared with the FDM and FEM used on the presented examples.

In the end the method is prepared for CFD and for use with Navier-Stokes equations and the SPH program SPHysics is presented. The aim of the second part is to compare computational results generated by a custom designed version of SPHysics with experimental results from the wave laboratory validating a virtual wave flume. The different theories used together with the SPHysics version of SPH are presented together with the experimental results in order to determine if the method is able to model the chosen fluid problem.

Chapter 2

SPH Method - Examples

The purpose of the following chapter is to describe how the Smoothed Particle Hydrodynamics (SPH) formulation may be used on a PDE boundary value problem. The first few sections are a quick review of how three different 1-D problems are solved with the method.

2.1 1-D Example - The vibrating string

The chosen PDE is the wave equation given in Equation(2.1), used to solve the problem of a vibrating string. Initial values and the associated analytical solution are given in Equation (2.2) and (2.3) respectively.

$$c^2 \frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2} \tag{2.1}$$

$$u(0,t) = u(L,t) = 0 \quad \wedge \quad u(x,0) = \sin(\pi x) \quad \frac{\partial}{\partial t} u(x,t)\Big|_{t=0} = 0 \tag{2.2}$$

$$u(x,t) = \sin(\pi x)\cos(c\pi t) \tag{2.3}$$

where

c^2 is the wave speed given as (F_E/ρ) [m/s]
$u = u(x,t)$ is the displacement [m]
F_E is the tension in the string [N]
ρ is the unit mass of the string [kg/m]
L is an integer and the length of the problem domain, Figure 2.2 [m]
t is the time [s]

The vibrating string problem is solved numerically by a number of steps using a SPH formulation with the following constants: $F_E = 1$, $\rho = 1$ and $L = 2$. With the given constants is the initial value at $t = 0$ given in Figure 2.1 together with a plot of the $J = 25$ particles in which the problem is discretized. The mass of a single particle m_i is given as J/L.

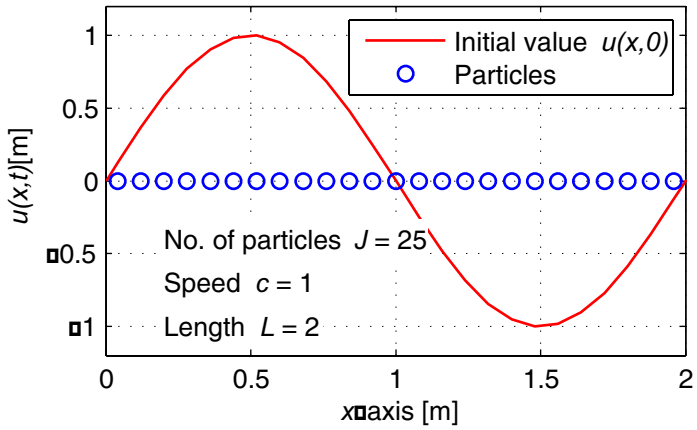


Figure 2.1. Plot of the initial values of the analytical solution given in Equation (2.3) at t=0 and the constants used throughout this example when describing the geometry of the problem.

The vibrating string problem is solved using the SPH procedure. Each step is given as an item on the following pages and key SPH concepts are explained:

- The problem is discretized with N particles. Each particle represents a length $dx = m_i = J/L$ and an initial value of u_i^0, see Figure 2.2.

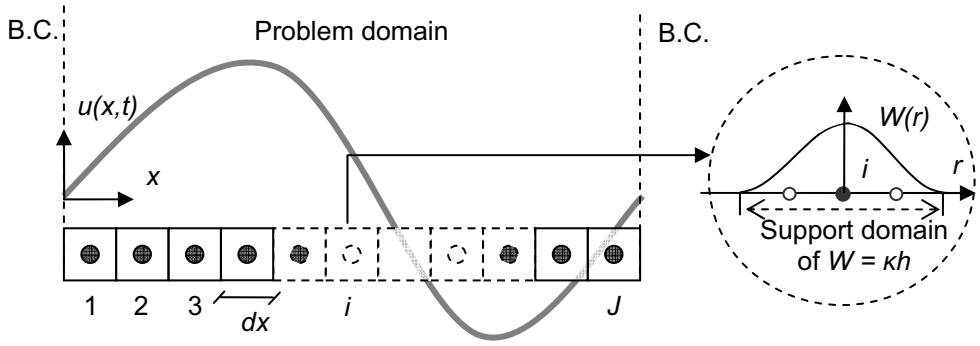


Figure 2.2. The initial values of the vibrating string given in Equation (2.2) depicted as a graph based on Equation (2.3) and discretized into N particles in the problem domain.

The basic unit of the SPH method is the particle. At each particle the momentary knowledge about the field variables like displacement or density is stored. There is no grid connecting the particles. To replace the grid is the kernel function of a single particle *W(r)* introduced on Figure 2.2. The value of a single field function at particle *i* is found by interpolating between all the particles in the support domain. The kernel constant κ depends on the choice of kernel, and the smoothing length *h* depends on the discretization of the problem. The SPH method is in (2.4) used to approximate the derivative of the field function *u(x,t)* in *J* particles.

$$\left\langle \frac{\partial u_i^n}{\partial x} \right\rangle = \sum_{j=1}^{J} dx_j u_j^n \nabla_i W_{ij} \qquad \frac{\partial u_i^n}{\partial x} \approx \left\langle \frac{\partial u_i^n}{\partial x} \right\rangle \tag{2.4}$$

Where

$( )_i$ is the particle where the field function is approximated

$( )_j$ is a particle in the problem domain

$( )^n$ is the time step number

∇_i is the gradient taken with respect to particle *i*

W_{ij} is the kernel function with the coordinate input $r_{ij} = \Delta x = x_i - x_j$

- Particle approximation (2.4) and the derivative of Kernel functions $\nabla_i W_{ij}$ are used to determine the second-order spatial derivative (2.5). The discrete form of Equation (2.1) is finally given as (2.6).

$$\left\langle \frac{\partial^2 u_i^n}{\partial x^2} \right\rangle = \sum_{j=1}^{J} dx_j \left(\frac{\partial u_j^n}{\partial x} \right) \nabla_i W_{ij} \tag{2.5}$$

$$\left\langle \frac{\partial^2 u_i^{n+1}}{\partial t^2} \right\rangle = c^2 \sum_{j=1}^{J} dx_j \left(\sum_{j=1}^{J} dx_j u_j^n \cdot \nabla_i W_{ij} \right) \nabla_i W_{ij} \tag{2.6}$$

where

$$\nabla_i W_{ij} = \frac{x_i - x_j}{r_{ij}} \frac{\partial W_{ij}}{\partial r_{ij}} \tag{2.7}$$

$$\frac{\partial u_i^n}{\partial x} = \left. \frac{\partial u}{\partial x} \right|_{x=x_i, t=t_n} \tag{2.8}$$

It is (2.6) that is solved numerically in MatLab. The code is available in Box 2.1. A plot of the discretized problem into 25 particles and associated kernel functions

is depicted on Figure 2.3 where the smoothing length is chosen to be $h = dx = 0.09$. The Gaussian kernel function (2.9) and its spatial derivative (2.10) used to solve this problem has in practice a kernel constant $\kappa = 4$ resulting in a support domain with the width of 0.38 m.

$$W(r) = \exp\frac{\left(-r^2/h\right)}{h\sqrt{\pi}} \tag{2.9}$$

$$\frac{\partial}{\partial x}W(r) = -1.1284\frac{r}{h^3}\exp\frac{\left(-r^2\right)}{h^2} \tag{2.10}$$

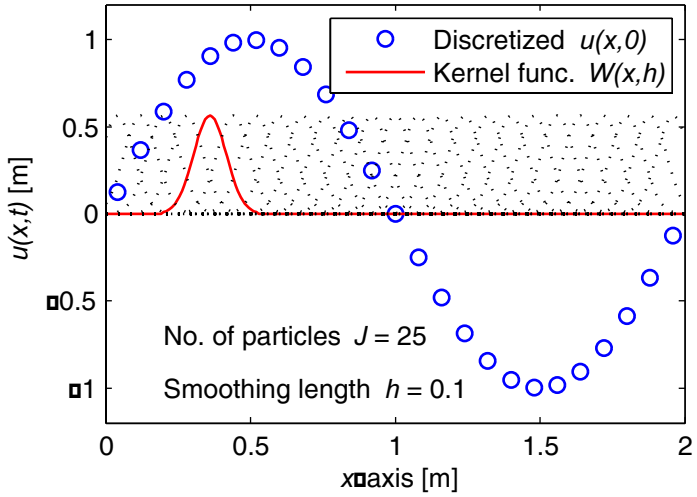


Figure 2.3. A plot of the discretized problem at t=0 and the associated kernel functions showing the spread of the support domain. The kernel function of particle five is marked. The derivative of the kernel functions are used to approximate a numerical solution of (2.1)-(2.3).

- The ODE is solved using an explicit integration algorithm like the Euler method with two steps (2.11) and (2.12). [Cullen et al, 2001]

$$\frac{du_i^{n+1}}{dt} = \frac{du_i^n}{dt} + \left\langle \frac{\partial^2 u_i^{n+1}}{\partial t^2} \right\rangle \cdot dt \tag{2.11}$$

$$u_i^{n+1} = u_i^n + \frac{\partial u_i^{n+1}}{\partial t} \cdot dt \tag{2.12}$$

- The whole procedure is repeated through the desired number of time steps, for instance one period T. The result is a numerical solution of the wave

equation shown in Figure 2.4 for four different time steps and a total time of T=2 sec. The finished MatLab code is given in Box 2.1.

Because the x-displacement is infinitesimal it is only necessary to define $\Sigma(\nabla W_{ij})$ once and store the resulting matrix *dKernelvalue* when writing the numerical solution of Box 2.1 . The boundaries of the problem are handled with the introduction of bo = 2 virtual ghost particles on each side of the problem domain. The ghost particles are introduced to solve the problem of truncated support domains close to the boundaries and they have displacements equal to their opposite number on the other side of the boundary. This is discussed in details in section 3.5.

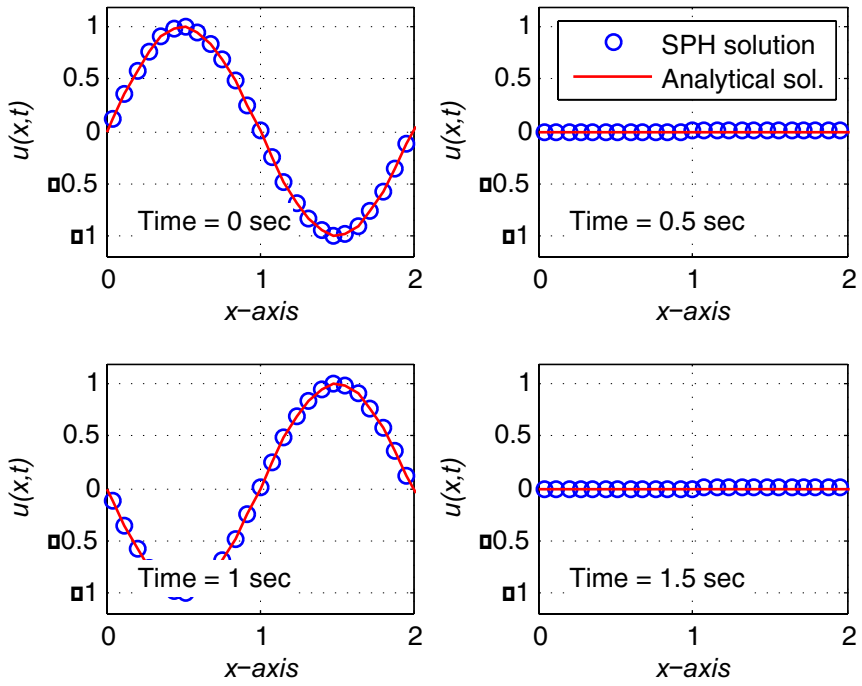


Figure 2.4. Numerical solution of the wave equation with the SPH method plotted at four different time steps together with the corresponding analytical solution Equation (2.3).

The method described above is limited to problems in which the distance between the particles dx is a constant. This is the case with a one dimensional problem like for instance the vibrating string or heat conduction problems. In case of particles moving between each other, i.e. changing their x-coordinate, the following item is added:

- Particle approximation is performed once every time step i.e. the use of the particles depend on the present distribution. When this is not the case it is only necessary to use the kernel functions once as it is done in the code, Box 2.1.

Box 2.1. SPH Code 1-D Vibrating String

```
%Definition of constants

N=25;           %Number of particles (boundary particles excluded)
L=2;            %Length of string
c=1;            %Wave speed c^2 = F/rho
dx=L/N;         %Distance bewteen particles
bo=2;           %Number of boundary particles on each side of problem domain
n=2*bo+N;       %Total number of particles
h=1.1*dx;       %Smoothing length for all particles
dt=0.001        %Size of timestep
Totalsteps=2000;

%Definition of coordinates, Particle volume, initial values u(x)
%and d/dx*u(x)=v

Xcord_point = [-bo*dx+dx/2:dx:L+bo*dx-dx/2]';
ParticleVol(1:n,1) = dx;
u_ini(1:n,1) = sin(pi*Xcord_point(1:n,1));
v_ini = zeros(n,1);

%Defining a matrix with values of d/dx*W used to find the derivate in each
%timestep

for i=1:n
    for j=1:n
        Xdif = Xcord_point(j,1)-Xcord_point(i,1);
        dKernelvalue(i,j) = -1.1284*Xdif/h^3*exp(-Xdif^2/h^2); %(2.10)
    end
end

%Loop to solve Equation (2.1)

for step=1:Totalsteps
    dFunction = zeros(n,1);
    ddFunction = zeros(n,1);

    %First round of SPH diff. 1st derivate (2.4)
    for j=1:n
        dFunction(j,1) = sum(dKernelvalue(:,j).*ParticleVol.*u_ini);
    end

    %Second round of SPH diff. 2nd derivate (2.5)
    for k=1:n
        ddFunction(k,1) = sum(dKernelvalue(:,k).*ParticleVol.*dFunction);
    end

    %Updating the bo boundary particles
    uBC =-1*[flipud(v_old(bo+1:2*bo,1)); flipud(v_old(n-2*bo+1:n:bo),1))];
    vBC =-1*[flipud(u_old(bo+1:2*bo,1)); flipud(u_old(n-2*bo+1:n:bo),1))];
    v_old =[uBC(1,:); v_old(bo+1:n-(bo),1); uBC(2,:)];
    u_old =[vBC(1,:); u_old(bo+1:n-(bo),1); vBC(2,:)];

    %Numerical integration - Simple Euler & Update of
    %initial values (2.11)
    v_new = v_ini+ddFunction.*(dt*c^2);
    u_ini = u_ini+v_new.*dt;
    v_ini = v_new;
end
```

2.2 1-D Example - Moving particles

This example is included to demonstrate the use of SPH together with moving particles, i.e. the particle approximation needs to be done once at every time step as the inter-dependence of the particles changes. The chosen example is two trains of moving particles colliding, Figure 2.5. Each particle represents a piece of the string with the length *dx* and mass *m*.

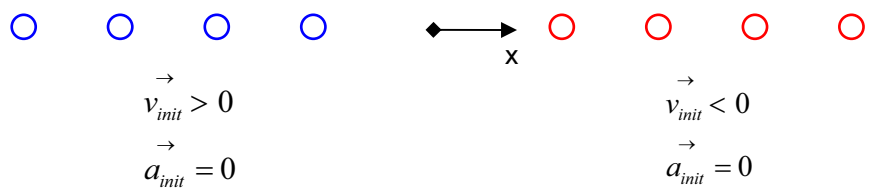


Figure 2.5. This plot shows the initial distribution of the J colliding particles and the initial directions of speed v_{init} and acceleration a_{init}. All particles in a train moves with the same velocity and acceleration.

As a result the governing PDE is the dimensional wave equation given by (2.1) and the Gaussian kernel (2.9) is used for the particle approximation. One problem when using Equation (2.4) is that the spatial derivative of a constant is not necessarily equal to zero. This is an issue in the problem depicted above because the initial values of speed and acceleration are constants throughout their respective trains. This will result in an error in the particle approximation before the two trains collides. The problem is solved using an identifier (2.13) and rewriting the particle approximation of (2.4) to (2.14). The proof is available in Appendix A.

$$\nabla \cdot u(x) = \frac{1}{\rho}\left[\nabla \cdot \left(\rho u(x)\right) - u(x) \cdot \nabla \rho\right] \tag{2.13}$$

The new discrete form of (2.1) is now given as (2.15).

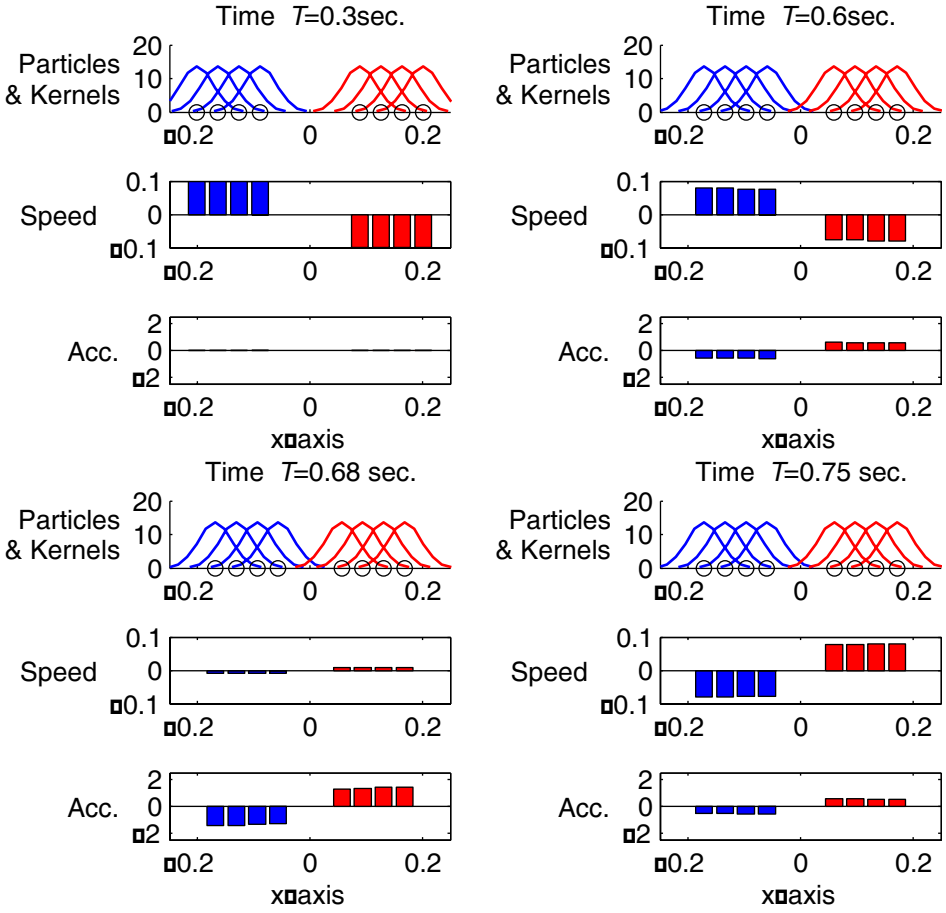
$$\left\langle \frac{\partial u_i^n}{\partial x_i} \right\rangle = \frac{1}{m_i}\left[\sum_{j=1}^{J} m_j \left(u_j^n - u_i^n\right) \cdot \nabla_i W_{ij}\right] \tag{2.14}$$

$$\left\langle \frac{\partial^2 u_i^{n+1}}{\partial t^2} \right\rangle = \frac{k^2}{m_i^2}\left[\sum_{j=1}^{J} m_j \left[\sum_{j=1}^{J} m_j \left(u_j^n - u_i^n\right) \cdot \nabla_i W_{ij}\right] \cdot \nabla_i W_{ij}\right] \tag{2.15}$$

Another big difference from Section 2.1 is the update of particle position (x-coordinate) at every time step (2.16).

$$x_i^{n+1} = x_i^0 + u_i^{n+1} \tag{2.16}$$

Solving the problem in MatLab with eight particles (four in each train) and using the rewritten equations generates a series of situations depicted in Figure 2.6. Note how the kinetic energy is constant in Figure 2.7 this is due to the new discrete formulation (2.15). The particles have a h=1.1dx, a tension F=50 N, T=1 sec. and the Gaussian kernel function (2.9). There are no boundaries in this example.



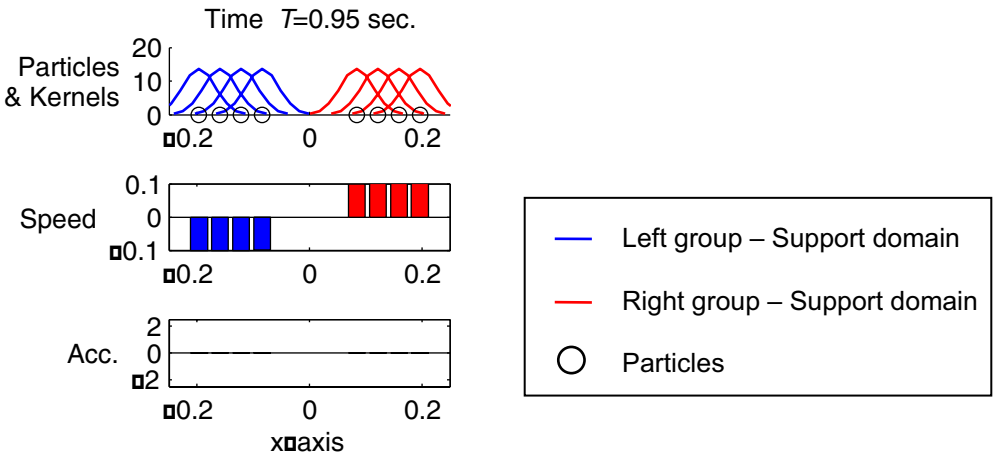


Figure 2.6. This plot shows the simple example of two trains colliding together with the particle support domain and the momentary distribution of acceleration and speed at five different time steps. The problem is started with $v_{init} = \pm 0.1$, $a_{init} = 0$ and a time step $dt = 0.001$.

To prove that the plotted solution is feasible the potential and kinetic energy of the system are calculated. The kinetic energy E_{kin} and potential energy E_{pot} are computed as a total for the whole system of particles using Equations (2.17) and (2.18) .

$$E_{kin}^{n} = \sum_{i=1}^{J} \frac{1}{2} m_i \left(v_i^n \right)^2 \tag{2.17}$$

$$E_{pot}^{n} = E_{pot}^{n-1} + \sum_{i=1}^{J} F_i^n \, s_i^n = E_{pot}^{n-1} + \sum_{i=1}^{J} \left(m_i a_i^n \right) \left(v_i^n dt \right) \tag{2.18}$$

where

F is the force conducting work on the particle [N]

s is the total displacement of the particle in a single time step [m]

A plot of the energy belonging to the problem from Figure 2.6 is depicted in Figure 2.7.

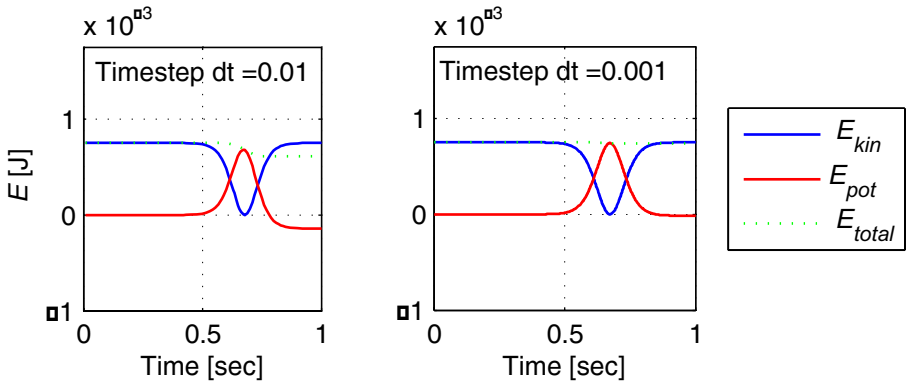


Figure 2.7. Plot of the total potential and kinetic energy with two different time steps and the Euler method. Both are made with basis in the problem depicted in Figure 2.6.

The energy depicted in Figure 2.6 shows that the problem with four moving particles is solved correctly using the discrete formulation of (2.15) and that the quality of the solution depends on the size of the time step dt which is clear from the difference between Figure 2.7 left and right. During the collision the particles are slowing down and kinetic energy is converted to potential energy and back again on a one to one basis if the time is discretized with a $dt = 10^{-3}$. The total of kinetic and potential energy E_{total} throughout the solution is equal to the initial amount of energy which together with the behaviour of the particles in Figure 2.6 makes it reasonable to believe that the problem is solved correctly. The code used to solve the problem is depicted in Box 2.2.

Box 2.2. SPH Code 1-D Colliding particles (Wave equation)

```
%Definition of constants

J = 4;                  %Number of particles in each side
w = 0.5;                %Width of problem
b = 0.2;                %Distance between the two trains of particles
dx = (w-b)/(2*J);       %Distance between particles
h = 1.1*dx;             %Smoothing length
dt = 0.001;             %Time step
F = 50;                 %Tension in problem trains
Totalsteps = 1500;      %Total number of time steps

%Definition of coordinates, Particle mass, Particle density, Particle %volume,
Elasticity vector and speed vector

Xcoor_pil = [-w/2:dx:-w/2+J*dx]; Xcoor_pir = [b/2:dx:w/2];

for i=1:1:length(Xcoor_pil)-1
    Xcoor_pal(i,1)=abs(Xcoor_pil(1,i+1)-Xcoor_pil(1,i))/2+Xcoor_pil(1,i);
    Xcoor_par(i,1)=abs(Xcoor_pir(1,i+1)-Xcoor_pir(1,i))/2+Xcoor_pir(1,i);
    particlemass_l(i,1)=abs(Xcoor_pil(1,i+1)-Xcoor_pil(1,i));
    particlemass_r(i,1)=abs(Xcoor_pir(1,i+1)-Xcoor_pir(1,i));
end

Xcoor_particle=[Xcoor_pal; Xcoor_par];              %Particle coordinates
particlemass =[particlemass_l; particlemass_r];     %Particle mass
particledensity(1:2*J,1)=1;                         %Particle density
particlevolume=particlemass./particledensity;       %Particle volume
E(1:2*J,1)=F;                                       %Particle elasticity
C=E/particledensity;                                %Particle Wavespeed

%Iinitial Displacement (Uini) and speed (Vini)
Uini(1:J,1)=0; Uini(J+1:2*J,1)=0; Vini(1:J,1)=0.1; Vini(J+1:2*J,1)=-0.1;

%Loop to solve the equation
Xcoor_particle0 = Xcoor_particle;

for step = 1:Totalsteps
    %Defining a matrix with values of d/dr*W used to find the derivate
    for i=1:length(Xcoor_particle)
        for j=1:length(Xcoor_particle)
            Xdif = Xcoor_particle(j,1)-Xcoor_particle(i,1);
            dKernelvalue(i,j) = -1.1284*Xdif/h^3*exp(-Xdif^2/h^2);
       end
    end

    dFunction = zeros(2*J,1); ddFunction = zeros(2*J,1);

    %First round of SPH diff. 1st derivative (2.14)
    for j=1:length(Xcoor_particle)
        dFunction(j,1)=1/particledensity(j,1)*sum(dKernelvalue(:,j).*...
            (particlemass.*(Uini-Uini(j,1))));
    end

    %Second round of SPH diff. 2nd derivative (2.15)
    for j=1:length(Xcoor_particle)
        ddFunction(j,1)=1/particledensity(j,1)*sum(dKernelvalue(:,j).*...
        (particlemass.*(dFunction-dFunction(j,1))))*C(j,1);
    end

    %Numericalintegration - Simple Euler & Update of coordinates (2.16)
    Vini = Vini+ddFunction.*dt;
    Uini = Uini+Vini.*dt;
    Xcoor_particle = Xcoor_particle0+Uini;
end
```

2.3 1-D Example - Collision with boundary

This example is included to demonstrate the use of SPH together with moving particles like in Section 2.2. Furthermore is two boundaries implemented at each end of the problem domain. The modeling of these boundaries is the main subject of the next few pages. The chosen problem is in 1-D and consists of one train with particles and two walls, Figure 2.8.

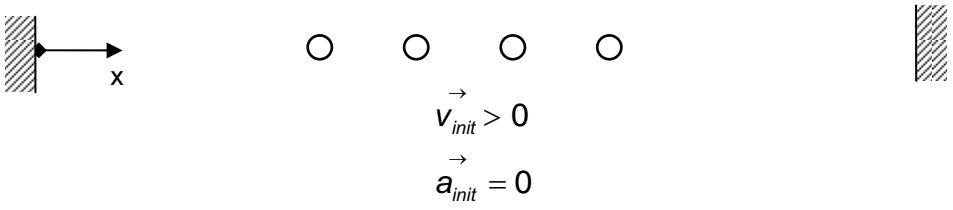


Figure 2.8. This plot shows the simple 1-D example with one train off J particles and two solid boundaries at each end of the problem domain. All particles in a train are moving with the same initial v and a.

Equation (2.1) is again used as governing PDE and particle approximation is done with the discrete form of (2.1) given in Section 2.2 as (2.15). The position of the particles are updated once each time step (2.16).

The modelling of boundaries was briefly described in Section 2.1. The theory behind ghost particles outside the problem domain is described in Equation (2.19)-(2.21) for a one dimensional problem with stationary boundaries.

$$\boldsymbol{x}_{iG} = 2\boldsymbol{x}_B - \boldsymbol{x}_i \tag{2.19}$$

$$u_{iG} = -u_i \tag{2.20}$$

$$v_{iG} = -v_i \tag{2.21}$$

where

$( )_B$ is the coordinates of the boundary

$( )_G$ is the ghost particles outside the problem domain

The removal of one particle train and the ghost particles is implemented in the code of Box 2.2. The new script given in Box 2.3 is used on a problem where h=1.1*dx*, a tension *F*=50 N, T=1.5 sec and the Gaussian kernel function (2.9). Five different resulting time steps are depicted on Figure 2.9.

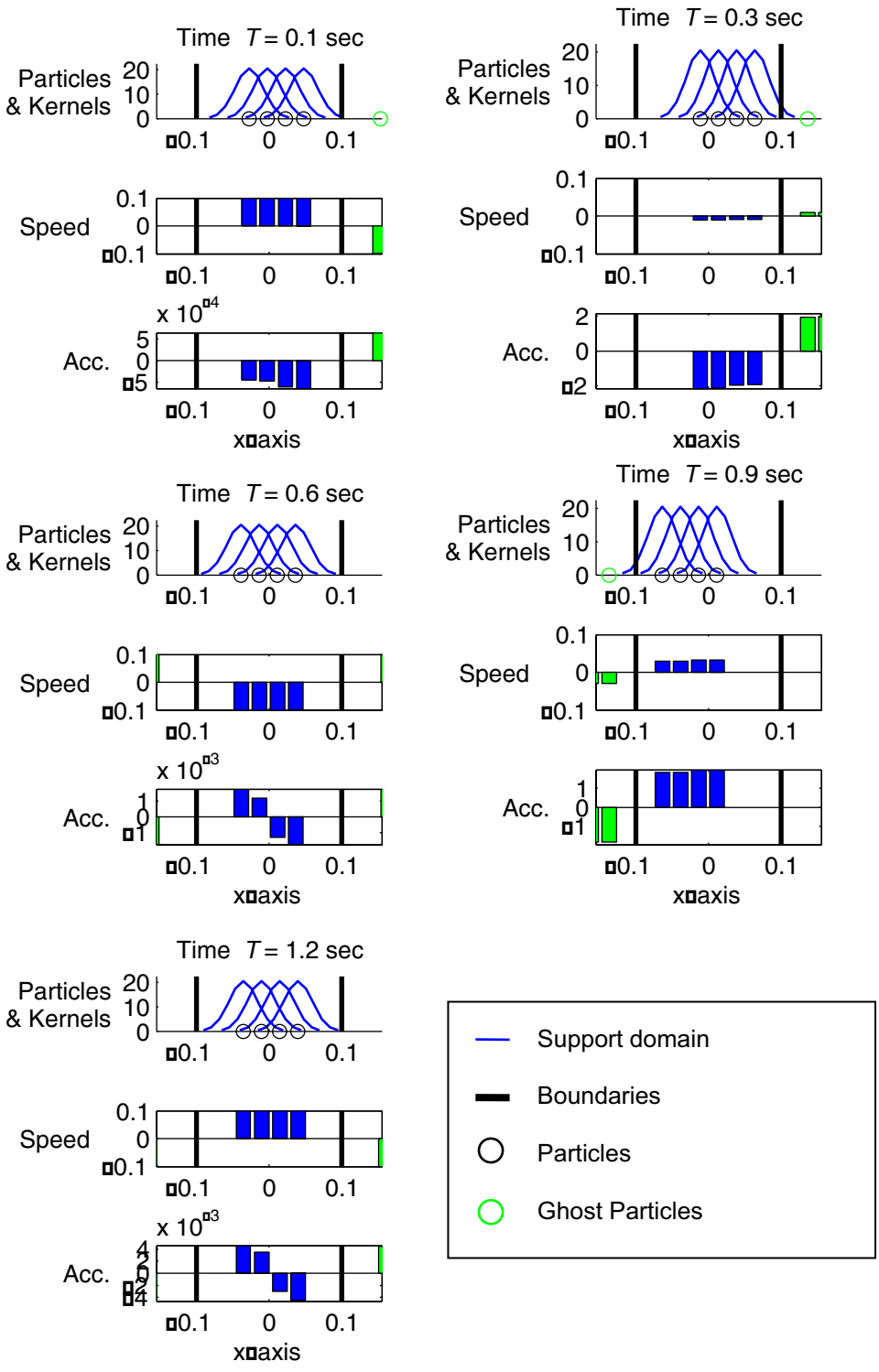


Figure 2.9. This plot shows the simple example of one train colliding with the boundaries at five different time steps. The problem is started with $v_{init} = 0.1$, $a_{init} = 0$ and a time step $dt = 0.001$.

The energy of this situation is computed as described in Section 2.2 and the result is depicted in Figure 2.10.

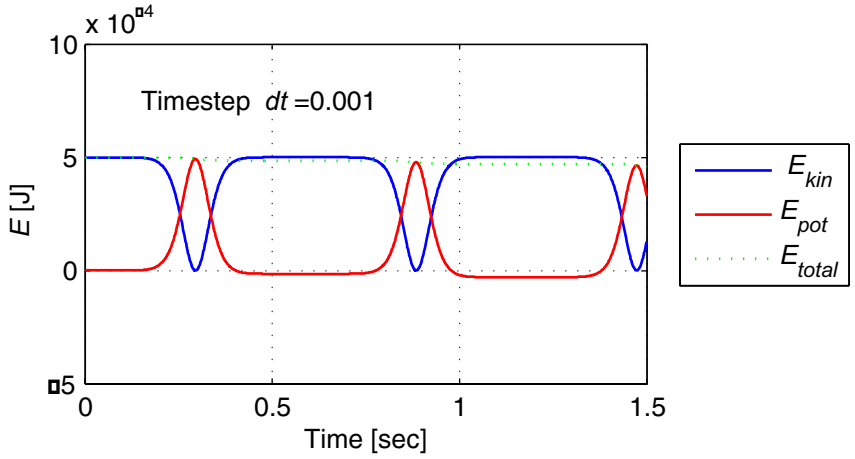


Figure 2.10. The computed kinetic and potential energy of the particle train colliding with the two boundaries of the problem domain.

The results depicted on Figure 2.9 and Figure 2.10 shows that the energy of the system is preserved. The particle train is reaccelerated when approaching the boundaries and subsequently repelled. The error in the SPH solution grows at every collision and one way to minimize it is to take smaller time steps or choose another numerical integration method. Solving the problem with a different kernel function leads to a slightly different solution this subject is addressed in Section 3.3.

The script used in this solution is given in Box 2.3. In the script a fixed number of ghost particles are applied at each boundary but this would not be feasible in a problem with several thousand particles. In this case only the particles closer than $\kappa h/2$ to the boundary would need a ghost i.e. the support domain is truncated by the boundary. It is also possible to solve the boundary problem with a fixed number of repellent particles who will repel any particles getting close this is not feasible for this simple example, but is widely used when solving problems with a complicated geometry. The different kinds of boundary particles are given a short introduction in Section 3.5.

Box 2.3. SPH Code 1-D Particle collision with boundary (Wave equation)

```
J = 4;                  %[m] number of particles in train
boundary = [-0.1 0.1];  %[m] width of problem domain
wt = 0.10;              %[m] width of particle train
dx = (wt)/(J);          %[m] distance between particles
V_ini = 0.1;            %[m/sec] Initial speed of all particles
F = 50;                  %[N] Tension in problem trains
rho = 1;                %[kg/m] Unit mass of all particles
h = 1.1*dx;             %[m] Smoothing length
dt = 0.001;             %[sec] Time step
Totalsteps = 1500;      %Total number of time steps

%Definition of coordinates, Particle mass, Particle density, Particle
%%volume, Elasticity vector and speed vector
Xcoor_t = [-wt/2:dx:wt/2];                          %Train pieces
for i=1:1:length(Xcoor_t)-1                         %Particle coordinates
    Xcoor_pa(i,1) = abs(Xcoor_t(1,i+1)-Xcoor_t(1,i))/2+Xcoor_t(1,i);
end
particlemass(1:3*J,:) = dx;                         %Particlemass
particledensity(1:3*J,1) = rho;                     %Particle unit mass
particlevolume = particlemass./particledensity;     %Particlevolume
F(1:3*J,1) = F;                                     %Elasticity of train

%Initial Displacement (Uini), acceleration (aini) and speed (Vini)
Uini(1:J,1) = 0; aini = Uini; Vini(1:J,1) = V_ini;

%Loop to solve the equation
Xcoor_particle0 = Xcoor_pa;
for step = 1:Totalsteps
    %Generating/Udating ghost particles (position, displacement, speed)
    Xcoor_all = [2*boundary(1,1)-flipud(Xcoor_pa(:,1));
                 Xcoor_pa(:,1);
                 2*boundary(1,2)-flipud(Xcoor_pa(:,1))];
    Uini_all = [-flipud(Uini); Uini; -flipud(Uini)];
    Vini_all = [-flipud(Vini); Vini; -flipud(Vini)];

    %Defining a matrix with values of d/dr*W used to find the derivate
    for i=1:length(Xcoor_all)
        for j=1:length(Xcoor_all)
            Xdif = Xcoor_all(j,1)-Xcoor_all(i,1);
            dKernelvalue(i,j) = -1.1284*Xdif/h^3*exp(-Xdif^2/h^2);
       end
    end
    dFunction = zeros(3*J,1); ddFunction = zeros(3*J,1);
    %First round of SPH diff. 1st derivative
    for j=1:length(Xcoor_all)
        dFunction(j,1) = 1/particledensity(j,1)*sum(dKernelvalue(:,j).*...
                (particlemass.*(Uini_all-Uini_all(j,1))));
    end
    cnum = F./particledensity;
    %Second round of SPH diff. 2nd derivative
    for j=1:length(Xcoor_all)
        ddFunction(j,1) = 1/particledensity(j,1)*sum(dKernelvalue(:,j).*...
                (particlemass.*(dFunction-dFunction(j,1))))*(cnum(j,1));
    end

    %Numericalintegration - Simple Euler & Update of coordinates
    Vini_all = Vini_all+ddFunction.*dt;
    Uini_all = Uini_all+Vini_all.*dt;
    Vini = Vini_all(J+1:length(Vini_all)-J,1);
    Uini = Uini_all(J+1:length(Uini_all)-J,1);

    Xcoor_pa = Xcoor_particle0+Uini;
end
```

2.4 Sub conclusion

The purpose of this Chapter has not been to give a finished SPH program ready to implement directly on a fluid. It has instead demonstrated the how the method are used when solving simple PDE boundary value problems. It has been shown that the method with the chosen discretization and choice of time step is able solve the problems.

The chosen examples was solved using the simplest parts of the SPH method and greater accuracy could be archived with a higher order time stepping like a 4^{th} order Runge-Kutta or another choice in kernel function. A wide variety of different kernel functions are available and a selection of these is presented in Section 3.3. Another logical step to improve the presented solutions would be a more advanced way to search for particles in a given support domain. The problem is limited with stationary particles but if the two collision examples were expanded to more than 1-D would the number of particles grow with a power of two. A number of possibilities are available and a method is introduced when it becomes necessary later in this report.

The examples shown in this chapter are used as a reference to explain the finer points of the method in the next few chapters.

Chapter 3

SPH Method – Theory

The examples presented in Chapter 2 were all made with the SPH method and the a few key elements of the theory was explained along the way. These four examples are now used as a basis for deriving the theory behind the SPH method and significance of the base concepts of the method as they are described in [Liu, 2003] and [Monaghan, 2005]. This chapter is a basis for the methods used to solve more advanced PDE with moving particles in 2-D.

3.1 SPH interpolation

On Figure 3.1, a problem domain discretized into J particles is depicted. Within the problem domain is the kernel function of a single particle depicted together with the support domain Ω and its surface S. The figure is used as a base when explaining the theory of the SPH method and three basic concepts of the method.

- Kernel functions W and ∇W are applied to interpolate between the particles like it is depicted on Figure 2.3 and Figure 3.1. The particles within the support domain of W are utilized in the particle approximation.

- The smoothing length h defines together with a kernel constant κ the support domain of the kernel function like as depicted in Figure 3.1.

- The Particle approximation is the discretization of a problem domain like (2.1) into J particles and the subsequent numerical approximation with the help of kernel functions. Like all numerical solutions, SPH is dependent on the discretization. Furthermore also on the choice of kernel functions and smoothing length h has an influence on the accuracy.

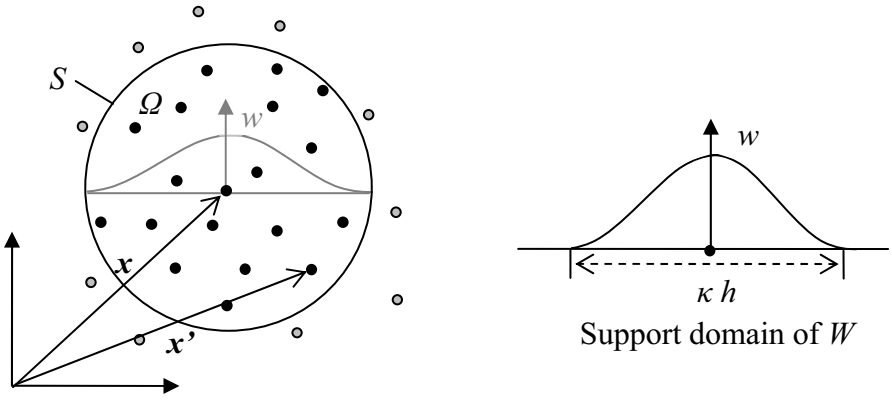


Figure 3.1. The principle of the SPH method, the value of the field function is determined with the use of an integral representation, a kernel function W and the remaining points in the support domain Ω with the surface S.

In order to explain the theory of particle approximation and kernel functions, it is necessary to start with an integral representation of the field function and its derivative within the support domain.

3.2 Integral representation

First step is the integral interpolant of the form (3.1) for the quantity *u(x,t)*. [Liu,2003]

$$u(\boldsymbol{x}) \approx \langle u(\boldsymbol{x}) \rangle = \int_{\Omega} u(\boldsymbol{x}') W(\boldsymbol{x}-\boldsymbol{x}', h)\, d\boldsymbol{x}' \tag{3.1}$$

where

*W(**x**-**x'**,h)* is a smoothing / kernel function
h is the smoothing length defining the support domain of *W(**x**-**x'**,h)*
x and ***x'*** are three dimensional position vectors, Figure 3.1
*d**x'*** is an infinitesimal volume

Equation (3.1) is also known as a kernel approximation and this is marked in SPH by using an angle bracket *⟨u(x,t)⟩*. The method is derived from the exact solution where *u*(*x*) is continuous within *Ω* and *W(**x**-**x'**,h)* is equal to the Dirac delta function *δ(**x**-**x'**)*.

When representing the first derivative of *u(x,t)*, the differential operation on the function is moved and performed on the kernel function instead used in (2.4). This is possible by replacing *u(x)* in (3.1) with $\nabla \cdot u(x)$ giving rise to Equation (3.2). It follows by the use of the divergence product rule for a vector field followed by the divergence theorem.

$$\begin{aligned}\left\langle \nabla \cdot u(\boldsymbol{x}) \right\rangle &= \int_{\Omega} \left(\nabla \cdot u(\boldsymbol{x}') \right) W(\boldsymbol{x} - \boldsymbol{x}', h)\, d\boldsymbol{x}' \\ \left\langle \nabla \cdot u(\boldsymbol{x}) \right\rangle &= \int_{\Omega} \nabla \left(u(\boldsymbol{x}') W(\boldsymbol{x} - \boldsymbol{x}', h) \right) d\boldsymbol{x}' - \int_{\Omega} u(\boldsymbol{x}') \cdot \nabla W(\boldsymbol{x} - \boldsymbol{x}', h)\, d\boldsymbol{x}' \\ \left\langle \nabla \cdot u(\boldsymbol{x}) \right\rangle &= \int_{S} u(\boldsymbol{x}') W(\boldsymbol{x} - \boldsymbol{x}', h) \cdot \boldsymbol{n} dS - \int_{\Omega} u(\boldsymbol{x}') \cdot \nabla W(\boldsymbol{x} - \boldsymbol{x}', h)\, d\boldsymbol{x}'\end{aligned} \quad (3.2)$$

where

n is the normal vector

The surface integral is removed because the kernel function is defined to have compact support from (3.7) i.e. the surface integral of the kernel function is equal to zero leaving the following representation of the spatial derivative.

$$\left\langle \nabla \cdot u(\boldsymbol{x}) \right\rangle = - \int_{\Omega} u(\boldsymbol{x}') \cdot \nabla W(\boldsymbol{x} - \boldsymbol{x}', h)\, d\boldsymbol{x}' \quad (3.3)$$

This is not the case when the kernel function is truncated by the boundary of the problem domain as it is the case in Figure 2.3. How this problem is solved and how boundary conditions are applied is discussed in Section 3.5.

3.3 Kernel functions

New kernel functions have been derived continuously during the last 30 years and the purpose of this chapter is to give a short review of the functions here divided into three different categories represented by examples in Figure 3.2 and given as equations in Table 3.1.

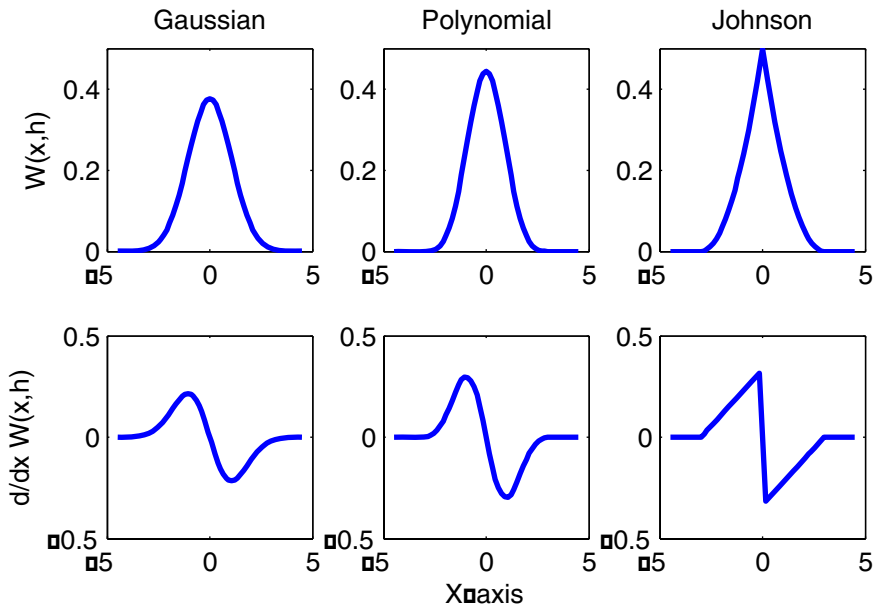


Figure 3.2. This is a plot of the three different types of kernel functions and their first derivative. The kernel functions are all plotted in 1-D around x=0 and smoothing length of h=1.5.

The three depicted Kernel functions share a number of traits. They are all made with a kernel factor $\kappa = 4$, and they are all nonnegative a typical choice for many smoothing functions [Monaghan, 2005] and [Liu, 2003]. One important difference between the Gaussian Kernel and the rest are its lack of compact support i.e. that the function value is zero outside the defined support domain.

- The Gaussian Kernel has a form common for a variety of smoothing functions and is made with the help of the well known exponential function e^x. The lack of a support domain implies that all particles in the problem domain are used in the approximation.

- The Polynomial kernel functions are on Figure 3.2 represented by the Cubic Spline, a kernel function build with two polynomials each representing a part of the function. There is a wide range of polynomials available in the literature, ranging from the second to the fifth order.

- The Johnson Kernel is a second order polynomial and represents the kernel functions which do not share the bell shaped form of the two first kernel types in Figure 3.2. It was designed to be superior to the Cubic spline in that it always increases as two particles moves closer and decrease when they move apart.

3.3.1 Major kernel properties

The Gaussian kernel function given in 1-D by Equation (3.4) and plotted above is used widely throughout examples in this text because of it is well known and easy to use in a simple code. The Gaussian kernel (3.4) is now also used as a base to discuss the conditions, a kernel function should observe in order to be useful in SPH interpolation.

$$W\left(x,h\right)=\frac{\exp\left(-x^2/h^2\right)}{\left(h\sqrt{\pi}\right)} \tag{3.4}$$

The Gaussian kernel (3.4) is given for a one dimensional problem. On Figure 3.3 is the kernel functions used to depict the three basic kernel conditions given by (3.5)-(3.7).

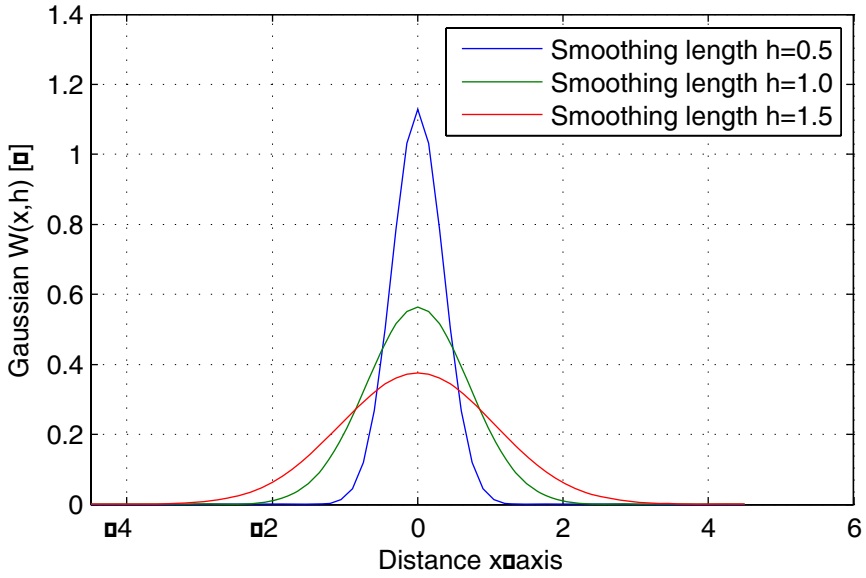


Figure 3.3. Plot of the Gaussian kernel with three different values of h showing the three basic kernel conditions given by Equation (3.5)-(3.6) in one dimension.

$$\int_{\Omega} W\left(\boldsymbol{x}-\boldsymbol{x}',h\right)d\boldsymbol{x}'=1 \tag{3.5}$$

$$\lim_{h\to 0} W\left(\boldsymbol{x}-\boldsymbol{x}',h\right)=\delta\left(\boldsymbol{x}-\boldsymbol{x}'\right) \tag{3.6}$$

$$W\left(\boldsymbol{x}-\boldsymbol{x}',h\right)=0 \qquad \left|\boldsymbol{x}-\boldsymbol{x}'\right|>\kappa h \tag{3.7}$$

where

κ is a kernel scaling factor depending on the choice of W often equal to four.

The conditions secure that the function is interpolated correctly and that the particle approximation approaches the function value. This is closely linked to the integral interpolant of (3.1) as a kernel functions with these conditions h=0 and would be equal to Diracs delta function making the solution exact. The last condition (3.7) transforms the approximation from a global to a local operation. This condition is the big disadvantage of the Gaussian kernel as it does not have a closely defined support domain.

3.3.2 List of Kernel functions

All the three types off kernel functions depicted in Figure 3.2 observe like the Gaussian kernel the conditions of Equation (3.5) - (3.7). They are presented in the following Figure 3.4 and Table 3.1 with the constant α_d given in Table 3.2 and R equal to (3.8).

$$R = \frac{|\boldsymbol{x} - \boldsymbol{x}'|}{h} \tag{3.8}$$

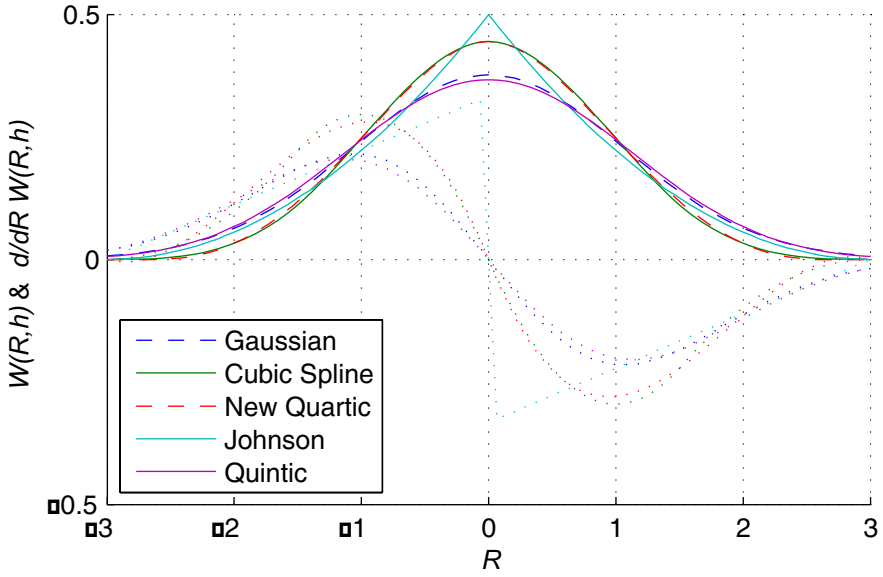


Figure 3.4. On this plot of a wide range of Kernel functions and their derivatives (dashed) with a smoothing length h=1.5 and plotted with reference to R. This makes it possible to depict the difference in scaling factor κ between the different functions.

Table 3.1. List of kernel functions discussed in this chapter available for 1, 2 and 3-D [Liu, 2003]

Kernel name	Equation	Eq. no
Gaussian	$W(R,h)=\alpha_d \exp\left(-R^2\right)$	(3.9)
Cubic spline	$W(R,h)=\begin{cases}\alpha_d\left(\frac{2}{3}-R^2+\frac{1}{2}R^3\right) & 0\le R\le 1\\ \alpha_d\,\frac{1}{6}\left(2-R^2\right) & 1\le R\le 2\\ 0 & 2<R\end{cases}$	(3.10)
New quartic	$W(R,h)=\alpha_d\left(\frac{2}{3}-\frac{9}{8}R^2+\frac{19}{24}R^3-\frac{5}{32}R^4\right)\quad 0\le R\le 2$	(3.11)
Quintic	$W(R,h)=\begin{cases}\alpha_d\left((3-R)^5-6(2-R)^5+15(1-R)^5\right) & 0\le R\le 1\\ \alpha_d\left((3-R)^5-6(2-R)^5\right) & 1\le R<2\\ \alpha_d(3-R)^5 & 2\le R\le 3\end{cases}$	(3.12)
Johnson	$W(R,h)=\alpha_d\left(\frac{3}{16}R^2-\frac{3}{4}R+\frac{3}{4}\right)\quad 0\le R\le 2$	(3.13)

Table 3.2. List of the constant α_d used together with kernel functions in Table 3.1 [Liu, 2003]

Kernel name	Eq. no.	K	1-D (α_d)	2-D (α_d)	3-D (α_d)
Gaussian	(3.9)	-	$1/(h\pi^{0.5})$	$1/(\pi h^2)$	$1/(\pi^{3/2}h^3)$
Cubic spline	(3.10)	4	$1/h$	$15/(\pi h^2)$	$3/(2\pi h^3)$
New Quartic	(3.11)	4	$1/h$	$15/(7\pi h^2)$	$315/(208\pi h^3)$
Quintic	(3.12)	6	$1/(120h)$	$7/(478\pi h^2)$	$3/(359\pi h^3)$
Johnson	(3.13)	4	$1/h$	$2/(\pi h^2)$	$5/(4\pi h^2)$

The Cubic Spline (3.10) is one of the most commonly used kernel functions, while the New Quartic (3.11) made to mimic the formers good qualities. The shape of the two functions is similar as it is depicted on Figure 3.4 the difference being that the New Quartic kernel has only one piece which makes a difference for the second derivate of the kernel function [Liu, 2003].

3.3.3 Comparison of Kernel functions

In order to display the difference between the five presented kernel functions the vibrating string example from Section 2.1 is used. A single particle is plotted through an entire period solved numerically with different kernels and compared to the analytical solution, Figure 3.5.

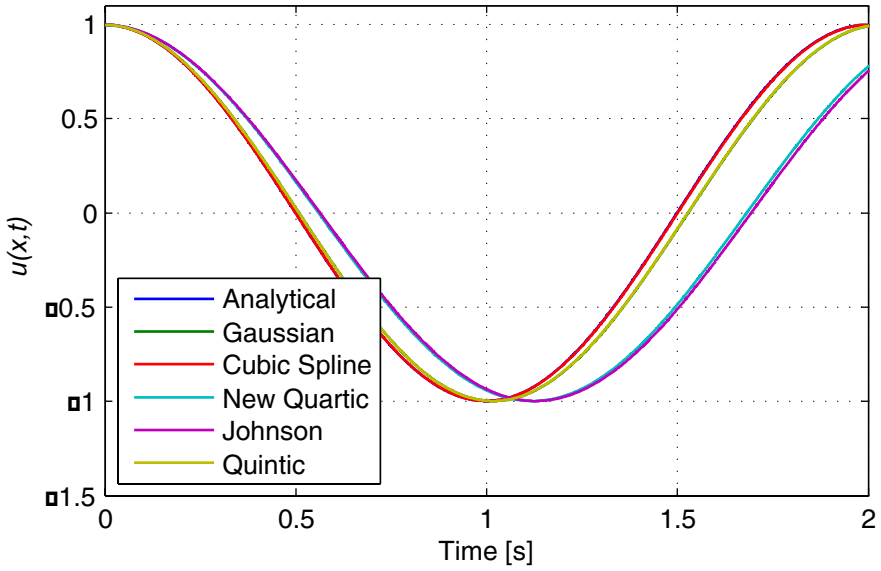


Figure 3.5. Numerical solution of the wave equation with the settings from Section 2.1 used to plot a single particle through a whole period T=2 sec. with five different kernel functions and constant smoothing length.

The plot in Figure 3.5 shows how that it is possible to solve the vibrating string problem with different kernel functions i.e. the plot of a point through a whole period T shows that the SPH solutions roughly follows the analytical solution. The Johnson and New Quartic kernels are evidently not suited to solve the system with the chosen parameters. Closer study of the New Quartic kernel reveals that $W(R,h) < 0$ when $R \approx 2$ making it critical how the parameters are chosen because it will influence a particle approximation where particles are placed in this area. As it is evident from Table 3.3 is the New Quartic giving good results with the optimum value of h.

Because the shape of the kernel function is dependent on h, is not possible to compare all five kernels with the same input of h and do a precise comparison. As an alternative the optimum value of h has been determined for each kernel function. The period elongation evident in Figure 3.5 is used to estimate the precision of the five different kernel functions in the 1-D case. The period of the SPH solution is determined using a zero down crossing analyses and given in Table 3.3.

Table 3.3. Measured periods T found with the help of a ZeroDown crossing analysis with five different kernel functions and their optimum values of h compared to the analytical period of T_{ana}=2 sec. Computation was running for 11 seconds.

Wave no.	**Analytical**	**Gaussian**	**Cubic Spline**	**New Quartic**	**Johnson**	**Quintic**
h	**-**	***h* = 1.00**	***h* = 1.10**	***h* = 1.00**	***h* = 1.25**	***h* = 0.90**
1	2.000	2.036	2.003	2.022	2.132	2.027
2	2.000	2.036	2.003	2.021	2.133	2.026
3	2.000	2.037	2.002	2.021	2.613	2.027
4	2.000	2.036	2.003	2.021	-	2.027
5	2.000	2.036	2.003	2.021	-	2.027

The periods in Table 3.3 show that although the kernel functions appear similar on Figure 3.5 it is necessary to determine the best kernel function for a given problem. The period elongation may be brought further down to a few parts of a thousand with a better discretization of time and space. As evident from Table 3.3, the Cubic Spline is superior when solving this example in 1-D. This is also concluded by [Monaghan, 2005] with respect to 1-D problems in general.

3.4 Smoothing length

The kernel function and smoothing length h are together comparable to the shape functions in the well known FE method. The smoothing length determines the size of the support domain i.e. the number of particles used to approximate the value of $u(\mathbf{x})$, Figure 3.1. The size of h directly influences the accuracy of a solution. A small value of h will mean that the number of particles in the support domain is too small to make an accurate SPH approximation while a large h may result in local properties being smoothed out. The size of h is also crucial when choosing the right time step dt for the time integration how to choose the right combination of h and dt is further explained in Section 5.2.8.

In the 1-D situation with stationary particles like the vibrating string example a smoothing length equal to $1.1 \cdot dx$ provides the most accurate result. This is depicted on Figure 3.6 based on the example from Section 2.1. This complies with [Liu, 2003] where the optimum number of neighboring particles is given as five in one dimension if h=1.2 and κ=2.

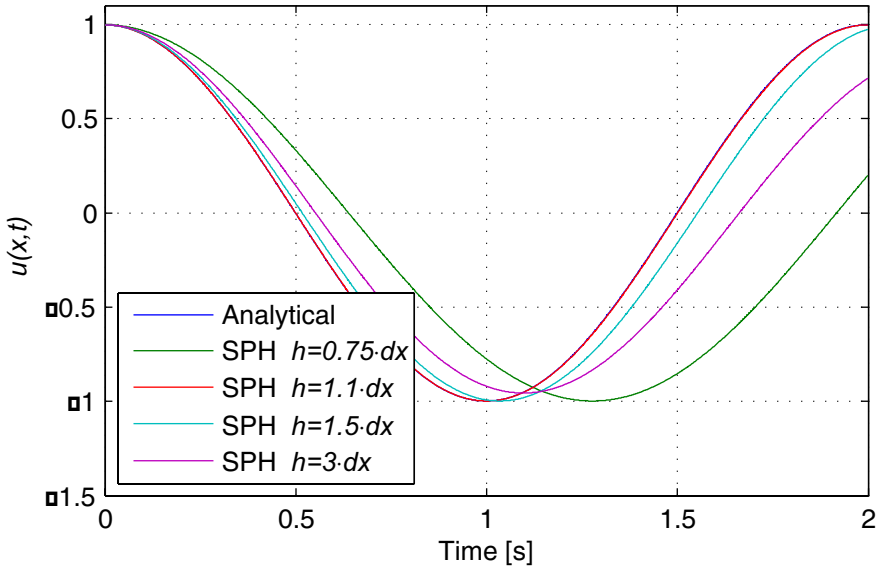


Figure 3.6. Numerical solution of the wave equation with the settings from Section 2.1 used to plot a single particle through a whole period T=2 sec. with four different values of h.

The situation depicted on Figure 3.6 made with stationary particles evenly distributed in the problem area and one smoothing length. If the distribution is retained and two different sizes of h are used the result becomes as depicted in Figure 3.7. Now the quality of the solution is depended on how far the support domain of particles with h_2 extends into other parts of the problem domain.

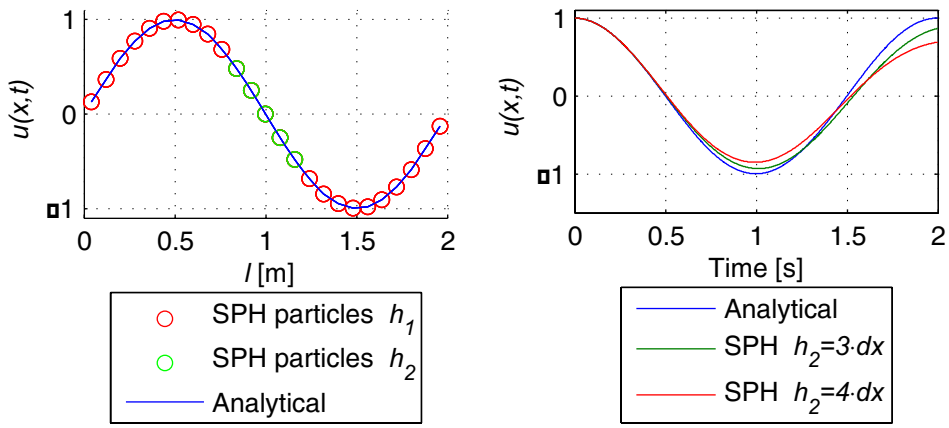


Figure 3.7. The Numerical solution of the wave equation with two different smoothing lengths h_1 and h_2,. h_1 is equal to 1.1dx and h_2 is variable.

3.5 SPH Boundaries

Boundaries are a problem when using SPH if the support domain of the kernel function extends beyond the boundaries of the problem. Only particles inside the boundary contributes to the summation and as the kernel function is truncated and there is no compact support (3.7). As a consequence the particle approximation derived in (3.2) is no longer exact this is depicted Figure 3.8 and summarized in (3.14).

$$W\left(\boldsymbol{x}-\boldsymbol{x}',h\right)\neq 0 \qquad \left|\boldsymbol{x}-\boldsymbol{x}'\right|>\kappa h$$
$$\Rightarrow \int_S u\left(\boldsymbol{x}'\right)W\left(\boldsymbol{x}-\boldsymbol{x}',h\right)\cdot\boldsymbol{n}dS\neq 0 \tag{3.14}$$

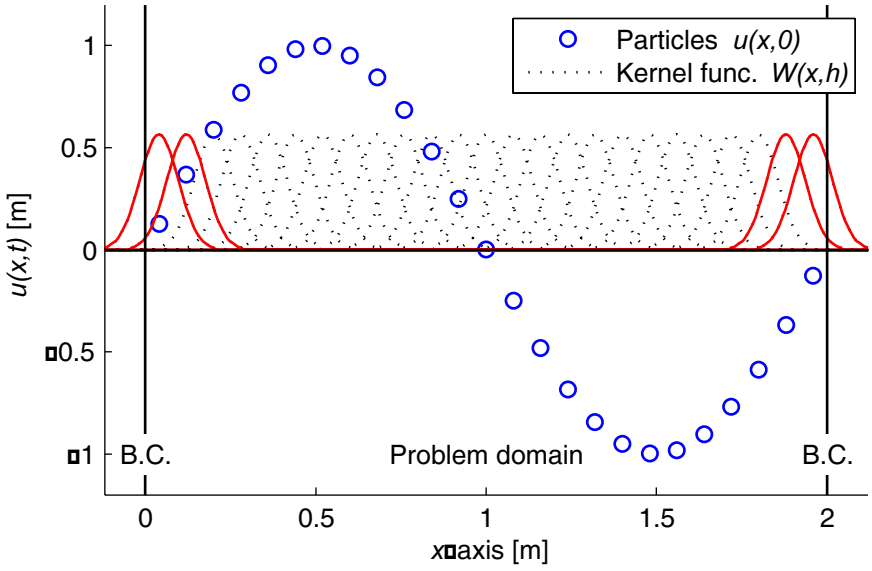


Figure 3.8. The support domain of the kernel functions (red) are truncated by the boundaries leading to a wrong integral representation of the function u(x,t). The solution is to add more virtual particles beyond the boundary.

A simple example of the consequences, if nothing is done at the boundaries, is shown on Figure 3.11 where particle approximation is used to compute the density distribution. The approximated density is wrong close to the boundaries.

Furthermore most the boundaries be able repel particles when they get close to a boundary to keep them inside the problem domain. An example of this ability is given in Section 2.3 where to boundaries are modelled to contain a particle train. Two different approaches are predominating when solving the boundary problem:

- Ghost Particles: Each particle close to a boundary has a ghost particle on the other side of the boundary with opposite values of the field variables displacement and speed. This method is used in the examples of Chapter 2 because of its simplicity but is not feasible with complicated geometries [Colagrassi et al, 2003].

- Repellent Particles: Are placed as boundaries of the problem domain. The particles in the problem domain are repelled as they approach the boundaries. This method is flexible and use full for complicated geometries. It is described in detail by [Monaghan, 2005].

The math behind ghost particles was introduced for the 1-D problem in Section 2.3. The ghost particles are implemented in the solution of the vibrating string problem as it is depicted on Figure 3.9.

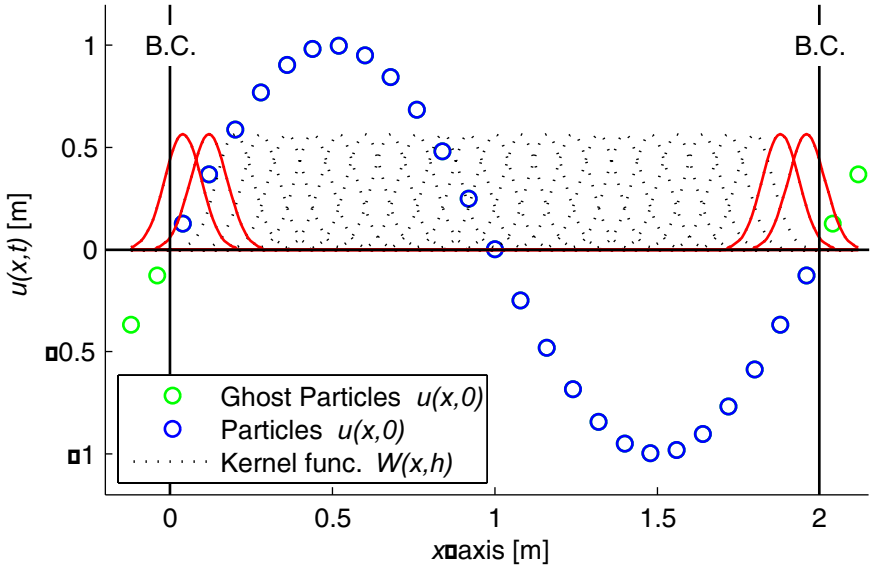


Figure 3.9. Figure showing the discretization of the problem in 25 particles and the addition of 2 ghost particles beyond each boundary (mirror). The ghost particles are in this problem equal to the negative value of their opposite number because of the boundary condition u(0, t)=u(L,t)=0.

As shown on Figure 3.9 is the value of the ghost particles based on their opposite number on the other side of the boundary with a truncated support domain. Because of boundary conditions (2.2) the sign of the particle is changed this is implemented in the MatLab code, Box 2.1. The necessary number of ghost particles depends on the smoothing length h, i.e. the extent of the kernel function. With the a the choice of h and $W(r,h)$ depicted on Figure 3.9 two ghost particles are necessary. [Liu, 2003]

3.6 Particle approximation

Particle approximation is the numerical method used to approximate the value of *u(x,t)* at a specific particle *i*, like it is done with the vibrating string in Equation (2.6). The integral approximation of (3.1) is basically approximated by the summation over a number of particles, where the infinitesimal volume ***dx'*** is rewritten as a finite volume of the particle *j* ΔV_j in (3.15).

$$m_j = \Delta V_j \rho_j \;\Rightarrow\; \Delta V_j = \frac{m_j}{\rho_j} \tag{3.15}$$

where

m_j is the mass of particle *j*
ρ_j is the density of particle *j*

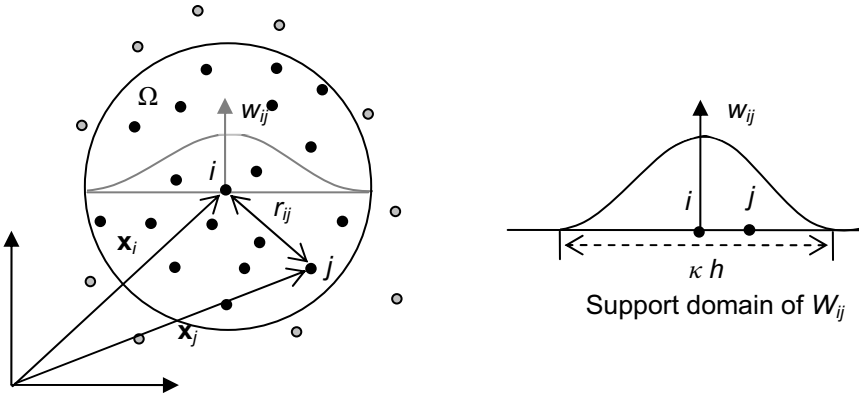


Figure 3.10. Replot of Figure 3.1 showing the SPH notation used in this report when the field functions is approximated at discretized particles.

Using (3.15) together with an integral interpolant, it is easily seen that the partial approximation of a value in a given point *i* is given as (3.16) with the notation shown on Figure 3.10.

$$\begin{aligned} \langle u(\boldsymbol{x}) \rangle &= \int_{\Omega} u(\boldsymbol{x'}) W(\boldsymbol{x} - \boldsymbol{x'}, h)\, d\boldsymbol{x'} \\ \langle u(\mathbf{x_i}) \rangle &= \sum_{j=1}^{N} \frac{m_j}{\rho_j} u(\mathbf{x_j}) W_{ij} \end{aligned} \tag{3.16}$$

where

J is the number of particles within the support domain of the particle i

W_{ij} is the kernel function $W(\mathbf{x}_i-\mathbf{x}_j,h)$

Substituting the function $u(x)$ with the density function ρ leads to a SPH approximation of the density given as (3.17) and used to calculate and plot the unit mass of the vibrating string, Figure 3.11.

$$\rho_i = \sum_{j=i}^{J} m_j W_{ij} \tag{3.17}$$

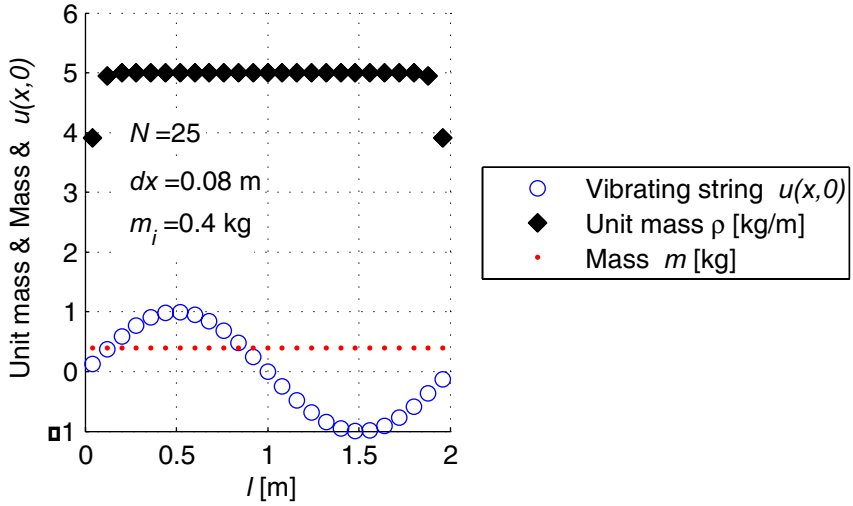


Figure 3.11. The plot is an example of particle approximation using the particle mass m_i of the vibrating string to interpolate the unit mass function ρ. Gaussian kernel and h=dx.

The Figure 3.11 shows the plotted particle approximation of the unit mass of the vibrating string. The density is inaccurate near the boundaries because the kernel functions are truncated by the boundary as mentioned in Section 3.2. This has taken into account when solving the vibrating string problem by using mirror particles at the boundary as it is described in Section 3.5.

The spatial derivative of the field function at particle i is in a similar fashion derived with particle approximation Equation (3.3) and (3.15).

$$\left\langle \nabla \cdot u\left(\mathbf{x}_i\right)\right\rangle = -\sum_{j=1}^{J} \frac{m_j}{\rho_j} u\left(\mathbf{x}_j\right) \cdot \nabla W_{ij} \tag{3.18}$$

where

∇W_{ij} is the gradient taken with respect to the particle j

It is possible to rewrite (3.18) and make it possible to take the spatial derivative of W with respect to only one variable r. Where the variable r is a spherical coordinate starting at the particle i as it is depicted on Figure 3.12.

(3.19)

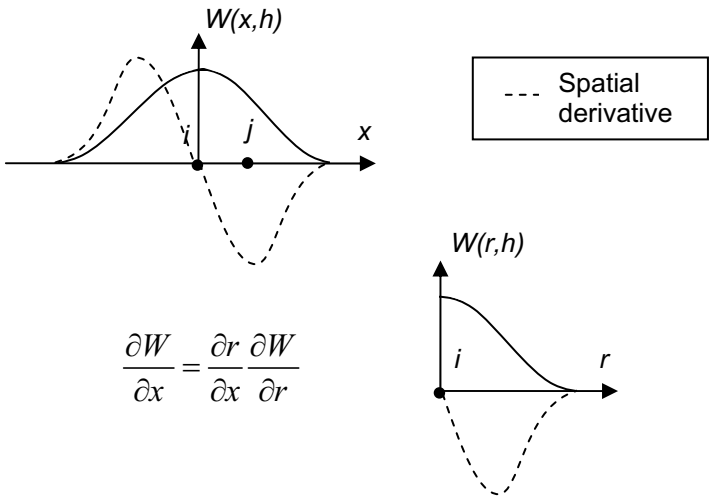


Figure 3.12. The spherical coordinate r and its derivative depicted in 1-D and the rewriting between the x and r coordinate.

In a spherical coordinate system a length and a direction are needed, the direction is in this case supplied by a direction vector defined as (3.20).

$$\boldsymbol{r}_d = \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{r_{ij}} \tag{3.20}$$

where

$$r_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2} \tag{3.21}$$

The rewritten particle approximation for the spatial derivative is given as (3.22). Notice that the negative sign in (3.18) is removed because the derivative now is taken with respect to particle i.

$$\left\langle \nabla \cdot u\left(\boldsymbol{x}_i\right)\right\rangle = \sum_{j=1}^{J} \frac{m_j}{\rho_j} u\left(\boldsymbol{x}_j\right) \cdot \nabla_i W_{ij} \tag{3.22}$$

where

$$\nabla_i W_{ij} = \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{r_{ij}} \frac{\partial W_{ij}}{\partial r_{ij}} \tag{3.23}$$

With (3.22) is possible to approximate the first derivative of *u(x)* the second derivative necessary to solve the vibrating string problem is found by simply using (3.22) twice leading to equation used for particle approximation with the particle string problem.

$$\left\langle \nabla^2 \cdot u\left(\boldsymbol{x}_i\right) \right\rangle = \sum_{j=1}^{J} \frac{m_j}{\rho_j} \left\langle \nabla \cdot u\left(\boldsymbol{x}_i\right) \right\rangle \cdot \nabla_i W_{ij} \tag{3.24}$$

It is possible to rewrite the spatial derivative using an identifier as it has been shown in the example with colliding particles Section 2.2. These alternate ways to determine the derivative of the field function are useful in a number of situations. The first alternative (3.25) is used to take the derivative of a field function with a constant value to ensure that the derivative is equal to zero [Monaghan, 2005].

$$\left\langle \nabla \cdot u\left(\boldsymbol{x}_i\right) \right\rangle = \frac{1}{\rho_i} \left[\sum_{j=1}^{J} m_j \left(u\left(\boldsymbol{x}_j\right) - u\left(\boldsymbol{x}_i\right) \right) \cdot \nabla_i W_{ij} \right] \tag{3.25}$$

The second alternative has not been used in this report but it is needed when solving the Naviar-Stokes equations with SPH in order to conserve the linear and angular momentum [Vesely, 2001] and introduces symmetry in the approximation who will limit the error when particles are not evenly distributed.

$$\left\langle \nabla \cdot u\left(\boldsymbol{x}_i\right) \right\rangle = \rho_i \left[\sum_{j=1}^{J} m_j \left(\frac{u\left(\boldsymbol{x}_j\right)}{\rho_j} + \frac{u\left(\boldsymbol{x}_i\right)}{\rho_i^2} \right) \cdot \nabla_i W_{ij} \right] \tag{3.26}$$

The used identifiers and how to rewrite (3.22) is presented in Appendix A. The full toolbox of SPH equations has now been presented and assembled in Box 3.1.

Box 3.1 The SPH basic toolbox

In this box are all the necessary SPH equations for particle approximation and useful rewritten equations presented. The theory behind the equations is explained in Section 3.2 and 3.6 and kernel functions are available in Table 3.1.

Particle approximation of a field function:

$$\left\langle u\left(\boldsymbol{x}_{\mathbf{i}}\right)\right\rangle = \sum_{j=1}^{N} \frac{m_j}{\rho_j} u\left(\boldsymbol{x}_{\mathbf{j}}\right) W_{ij} \tag{3.16}$$

$$W_{ij} = W\left(\boldsymbol{x}_i - \boldsymbol{x}_j, h\right)$$

Particle approximation of the first derivative:

$$\left\langle \nabla \cdot u\left(\boldsymbol{x}_i\right)\right\rangle = \sum_{j=1}^{J} \frac{m_j}{\rho_j} u\left(\boldsymbol{x}_j\right) \cdot \nabla_i W_{ij} \tag{3.22}$$

$$\nabla_i W_{ij} = \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{r_{ij}} \frac{\partial W_{ij}}{\partial r_{ij}} \tag{3.23}$$

Particle approximation of the first derivative - Rewritten Appendix A:

$$\left\langle \nabla \cdot u\left(\boldsymbol{x}_i\right)\right\rangle = \frac{1}{\rho_i}\left[\sum_{j=1}^{J} m_j \left(u\left(\boldsymbol{x}_j\right) - u\left(\boldsymbol{x}_i\right)\right) \cdot \nabla_i W_{ij}\right]$$

$$\left\langle \nabla \cdot u\left(\boldsymbol{x}_i\right)\right\rangle = \rho_i\left[\sum_{j=1}^{J} m_j \left(\frac{u\left(\boldsymbol{x}_j\right)}{\rho_j} + \frac{u\left(\boldsymbol{x}_i\right)}{\rho_i^2}\right) \cdot \nabla_i W_{ij}\right]$$

Density approximation:

$$\rho_i = \sum_{j=i}^{J} m_j W_{ij} \tag{3.17}$$

Example of kernel function (Gaussian) and its derivative 1-D:

$$W\left(r\right) = \exp\frac{\left(-r^2/h\right)}{h\sqrt{\pi}} \qquad \frac{\partial}{\partial x} W\left(r\right) = -1.1284 \frac{r}{h^3} \exp\frac{\left(-r^2\right)}{h^2}$$

3.7 Sub conclusion

The SPH toolbox presented in Box 3.1 is the basics tools necessary to use the SPH method to make numerical solutions.

The kernel function and the smoothing length are two the two basis parameters in SPHysics and it were evident how the changing of one would change the quality of the solution. The kernel functions presented here have all been used in SPH although the Cubic Spline and the Gaussian are by far the most common. It has not been discussed in this chapter that it is possible to allow the smoothing length to vary with time for instance dependent on the density distribution. Sources like [Liu, 2003] describe this possibility in order to enhance the solution when working with moving particles in two and three dimensions. The area is not a part of this project and the smoothing length will remain a constant as is the norm the remaining SPH literature used in this report.

Boundary conditions are an area where a number of possibilities are presented in the SPH literature. The two approaches presented in this chapter the basic ways to approach the problem and the mirrors used in the examples of Chapter 2 one are maybe the most simple and stable example. Nevertheless are repellent particles the more common approach when particles are moving as it is easier to handle stationary boundary particles.

The next step is to broaden the theory for use with a specific numerical problem namely the computation of a virtual wave flume. To this end is the SPH solution of Navier-Stokes equations derived in Section 5.2. Together with the derived particle approximation of Navier-Stokes equation are a number of different filters and methods presented who have been developed to improve the solution of CFD problems.

Chapter 4

Study of the SPH method 1-D

The SPH method has in many ways taken the best from the traditional grid based method like the Finite Difference method (FDM) and the Finite Element method (FEM). Where FDM is an example of a direct discretization of the strong form approach to describe the physical governing equations and FEM represents the weak form approach. The SPH method is characterized as a mesh free weak form particle method but is unlike FDM and FEM using a kernel functions instead of stiffness and mass matrices to interpolate between the particles.

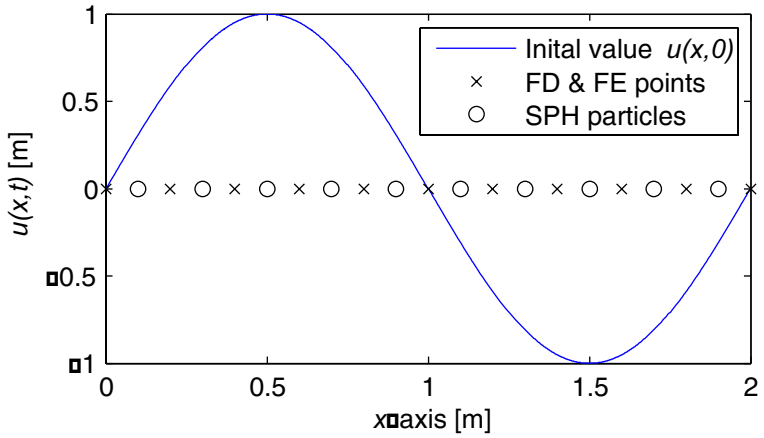


Figure 4.1. Plot of the initial values of the 1-D vibrating string problem discretized with one SPH particle in each FD / FE element.

In this Chapter is the three methods are compared using the vibrating string problem from Section 2.1, cf. Figure 4.1. The scope is to discuss the difference between the ways the three methods handle problem variables like stiffness and the distribution of mass problems where the simple analytical solution does not work. The Euler method is again used for explicit time integration. The governing equa-

tion for the vibrating 1-D string is given in equation (4.1) where c^2 is the wave speed, F_E the tension in the string (Elasticity) and ρ is the unit mass (density).

$$\frac{\partial^2 u}{\partial t^2} = \frac{F_E}{\rho}\frac{\partial^2 u}{\partial x^2} = c^2\frac{\partial^2 u}{\partial x^2} \tag{4.1}$$

4.1 FDM method

The FDM method is based on the strong form equations i.e. a direct discretization of the governing equations using equation (4.2) where the mass is lumped at *J-1* discrete notes.

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} \tag{4.2}$$

The result is stiffness matrix $\mathbf{K_{FD}}$ and a mass matrix $\mathbf{M_{FD}}$ ($\mathbf{M}=\Delta x/\rho\cdot\mathbf{I}$) where $\mathbf{I}$ is an identity matrix. The system is solved by the use of the finite element formulation of dynamics system:

$$\mathbf{M}\ddot{\boldsymbol{u}}(t) + \mathbf{C}\dot{\boldsymbol{u}}(t) + \mathbf{K}\boldsymbol{u}(t) = \boldsymbol{f}(t) \tag{4.3}$$

$$\mathbf{M}\ddot{\boldsymbol{u}}(t) + \mathbf{K}\boldsymbol{u}(t) = \mathbf{0} \tag{4.4}$$

Where the damping matrix $\mathbf{C}$ and external force $\boldsymbol{f}$ are zero in this problem, equation (4.4).

4.2 FEM method

The FEM method is based on a weak form approach i.e. the strong forms (governing equations) are multiplied with a virtual field (shape functions). The weak form will approach the strong form as the number of elements goes against infinity. A linear shape function (4.5) is used to calculate the element displacement and distribute the mass of the element, where $x_{e,start}$ is the first point of the element..

$$\Phi(x_e) = \left[\left(1 - \frac{x_e}{L_e}\right)\frac{x_e}{L_e}\right] \qquad x_e = x - x_{e,start} \tag{4.5}$$

The element matrix (4.6) is used to build the stiffness matrix $\mathbf{K}_{\mathrm{FE}}$ and mass element matrix (4.7) is used to build the mass matrix $\mathbf{M}_{\mathrm{FE}}$.

$$K_e = \frac{F_E/\rho}{\Delta x}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{4.6}$$

$$M_e = \frac{\rho \Delta x}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{4.7}$$

The system is again solved based on the short form of the finite element solution (4.4) and the Euler integration algorithm, cf. Figure 4.2.

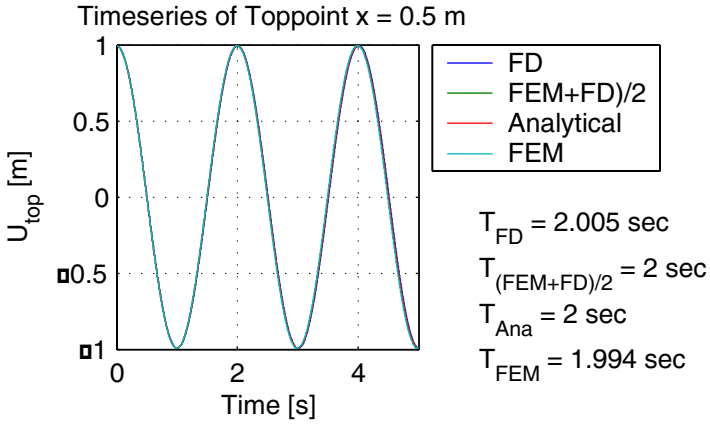


Figure 4.2. A comparison of the two methods FD and FEM with identical stiffness and two different levels of FEM mass distribution compared to an analytical solution at the point u(0.5,t).The period elongation computed with zero-down crossing is depicted in the lower right corner.

The optimum distribution of the mass is the mean of the $\mathbf{M}_{FD}$ and $\mathbf{M}_{FEM}$. This evident from Figure 4.2 where the three different possibilities are depicted showing that the FEM solution defined as (FEM+FD/2) and the analytical solution share the same period i.e. no period elongation. This FEM solution is used in Section 4.4 when comparing with the SPH solution and changing the wave speed in part of the string.

4.3 Comparison with SPH

The three methods are compared with a plot of a single point in the solution through a single period, Figure 4.4. The problem is solved using the standard values from Section 2.1 with $L = 2$, $T = 2$ and $c = 1$ while the SPH method is used with $h = 1.1dx$ and the Cubic Spline kernel function. The discretization is 25 SPH

particles and 26 FD/FE points and dt = 0.001. The result after 10 seconds is depicted on Figure 4.3.

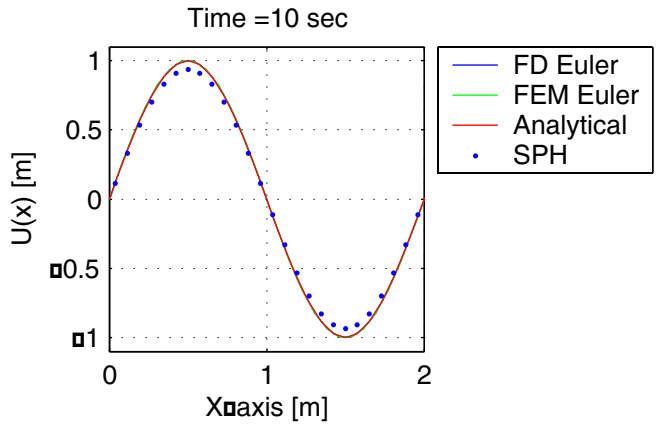


Figure 4.3. A comparison of the three methods SPH, FD and FEM with identical stiffness and mass distribution compared to an analytical solution after 10 seconds.

It is evident from the comparison of the three methods that there is a period elongation i.e. they are all a bit faster/slower than the analytical solution. This is depicted on Figure 4.4 by following the time history of a single point in the solution $u(0.5,t)$.

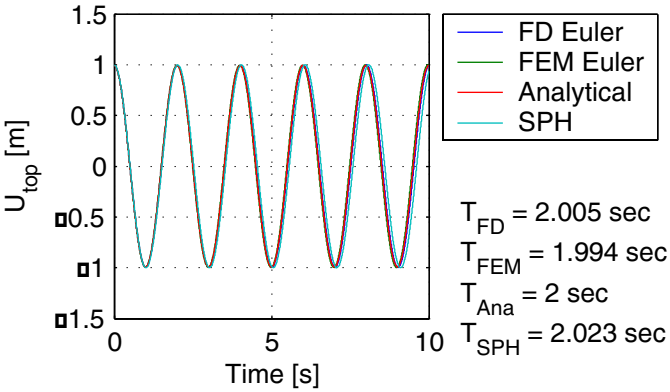


Figure 4.4. A comparison of the three methods SPH, FD and FEM with identical stiffness and mass distribution compared to an analytical solution at the point u(0.5,t).The period elongation computed with zero-down crossing is depicted in the lower right corner.

It is evident from Figure 4.4 that the three methods compute a solution which is in close to the analytical solution. The presented figures also show that although

FEM and SPH are both weak formulations with mass interpolated over several particles they do necessarily agree about the solutions of the problem.

4.4 Variable tension and density

Having compared the three methods in the linear case with the same density and tension throughout the string it was shown that the FEM solution with $\mathbf{M}_{(FEM+FD)/2}$ was equal to the analytical solution, cf. Figure 4.2. In the following this solution is used as a comparison to the SPH method.

A simple change in the problem presented problem would be to change the tension or unit mass in part of the string while keeping the original period of the motion. The following configuration would for instance keep the period stable as the wave speed is not changed:

$$\begin{aligned} &E_0 = 1 \quad \rho_0 = 1 \qquad c_0 = E_0/\rho_0 = 1 \\ &E_1 = 4 \quad \rho_1 = 4 \qquad c_0 = E_0/\rho_0 = 1 \\ &E_2 = 1/4 \quad \rho_2 = 1/4 \qquad c_0 = E_0/\rho_0 = 1 \end{aligned}$$

The string is divided in two peaces, one and two each with a length of one. The result would be that the period is maintained but the two parts of the string will have very different elevation compared to the original reference situation (with E_0 and ρ_0), cf. Figure 4.5.

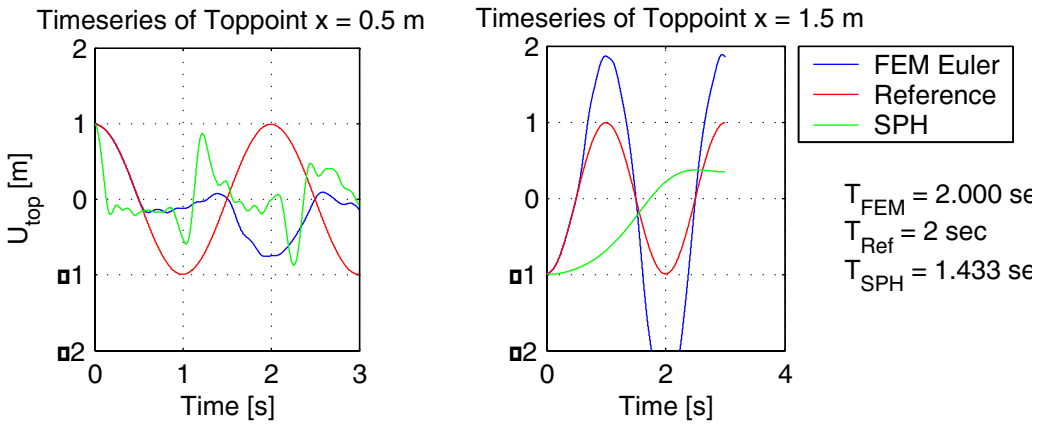


Figure 4.5. The history in the two initial top points u(0.5,t) and u(1.5,t) it is evident how the two pieces of the string have huge difference in stiffness and mass. The SPH on the other hand is not able to model this kind of problem.

It is evident that the SPH solution is unable to model the depicted situation in the same way as optimum FEM solution. This is properly due to the mesh free nature of the method and the looser connection the kernels represent, all the information about the first derivative is collected from the neighbours as $\nabla W(0,h) = 0$, cf Figure 3.2. An increase in the number of particles does not cause any difference in this case.

4.5 Sub conclusion

The SPH method has a number of problems when it comes to modelling a connection when the two peaces of string have a great difference in material parameters. Although this is a weakness when working with the impulse in a continuing peace of string the same abilities makes it possible for SPH to model the large deformations in fluids. It was demonstrated that it with the FEM method is possible to write a numerical solution equal to the analytical with the right distribution of mass and stiffness.

Chapter 5

Virtual Wave Flume

This marks the beginning of the second part of the report. In the first part the basic SPH theory was presented and used to solve simple PDE problems. The next step is to implement it in 2D in order to model a virtual wave flume. The wave flume has already been used as a model in several SPH experiments, cf. [Monaghan & Kos; 1999] and [Gotoh et al; 2004]. Their experiments show that it is indeed possible to generate a 2-D wave situation using SPH and their different approaches to the problem also show some of the options available with the SPH method. The combination of methods presented in this chapter is another possibility and known alternatives are given when the methods are presented.

The virtual wave model must be able to generate breaking waves and model nonlinear impact situations. It is in this kind of problems the SPH method because of its particle nature might be an asset. The governing equations of this kind of fluid problems are the Navier-Stokes equations together with a turbulence model to describe the viscosity.

After the governing equations have been chosen it is necessary to choose the other capabilities of the flume. A virtual wave capable of modelling the flume in the laboratory needs a number of abilities: (1) It must be possible to build a number of different geometries and possibly obstacles in the flume. (2) It must be possible to fill the flume with water to fit arbitrary geometry and solve the governing equations of this fluid. (3) The solution must be able to handle a free surface. (4) It must be able to generate waves either by introducing a paddle or possibly an initial displacement of water. (5) The chosen boundaries must be able to handle an arbitrary geometry and allow for a moving paddle. Furthermore it would be an asset if it was possible to measure pressure on the boundary. The capabilities of a virtual flume are depicted in Figure 5.1.