Membrane Simulations: Computing Normals to a Surface Represented by Unconnected Points

D. Sulsky
Department of Mathematics and Statistics
University of New Mexico
Albuquerque, NM 87131
sulsky@math.unm.edu

A. R. York II
Applied Research Associates, Inc.
811 Spring Forest Rd, Suite 100
Raleigh, NC 27609
ayork@sed.ara.com

August, 2000

Subject Classifications: 65P99, 73K10, 73K70

Keywords: membranes, connectivity, surface normals, finite differences, finite elements, front tracking

Proposed running head: "Membrane Simulations: Computing Normals"

Send Proofs to:
D. Sulsky
California Institute of Technology
1200 E. California Blvd.
MS 205-45
Pasadena, CA 91125

FAX: 626-449-2677
EMAIL: sulsky@aero.caltech.edu

Abstract

Simulations of fluid-filled membranes are considered where the membrane is discretized by a set of material points that have no logical connections. Since the normal to the membrane surface is necessary for the membrane constitutive model, an algorithm is given for constructing surface normals when the surface is represented by an unconnected set of data points. The main idea is to locally construct a function that has the given surface as an isosurface; gradients of this function provide the normal to the surface. This capability combines advantageous features of surface tracking methods with features of surface capturing methods so that information about the surface location is known from the points on the surface, but topology changes are numerically tractable because the points are not connected.

1 Introduction

Many important physical problems involve interfaces between materials; often these interfaces have nontrivial properties that contribute significantly to the problem solution. Examples are surface tension on the interface between two immiscible fluids, material boundaries in multi-phase flow, and membranes filled with fluid. Other structures such as shocks, vortex sheets, or shear bands, can also be treated as interfaces. Mathematically, interfaces are usually considered to be an infinitesimally thin boundary with no structure through the thickness. Multi-phase or multiple-material problems then reduce to moving boundary problems, with continuum equations applying in each constituent, and with boundary conditions applied at the interface. Such moving boundary problems are notoriously difficult to handle numerically.

Numerical methods for treating interfaces can be broadly separated into two classes, those that 'capture' the interface, and those that 'track' the interface. In the first group, there are methods such as continuum surface force (CSF) [1], level sets [2, 3, 4, 5], and phase-field methods [6, 7, 9, 8, 10, 11]. In this group, the interface is not explicitly represented, but its location is inferred from the computed solution. Typically, a characteristic function or order parameter characterizes each material or phase, and the interface is an isosurface of this function. The second group consists of methods such as front tracking [12, 13, 14], immersed-boundary [15, 16] or immersed-interface methods [17]. In these methods, the interface is explicitly represented, by a surface mesh, by points connected with links, or by a spline; respectively. Interface tracking tends to give better information about the location of the interface, but can be costly if the interface surface needs to be remeshed often because of large deformations or changes in topology. Interface capturing has an advantage when the topology changes, such as the merging of droplets, since this can be handled without logic for breaking and reconnecting points on surfaces.

This paper concerns simulations of fluid-filled membranes, such as automobile airbags or parachutes, and continues the work begun in [18]. A new method is introduced for tracking the membrane surface with a set of material points that have no logical connections, yet allowing computation of a normal to the surface for use in the membrane constitutive model. This membrane representation provides explicit information about the surface location, as in interface tracking methods, but also is amenable to large deformations and changes in topology (such as rupture), as in interface capturing methods. Our simulations use this membrane representation in conjunction with the material-point method for continuum mechanics problems.

The material-point method (MPM) [19, 20, 21] is an extension of the hydrodynamics, particle-in-cell code, FLIP [22, 23] to problems in solid mechanics. In MPM, a fluid or solid body is discretized using an unconnected set of material points that are followed throughout the deformation history. Information from these points is transferred to a background computational grid, where the momentum equation is solved. The grid solution is then used to move the material points and update their properties. In contrast to FLIP, the full stress

tensor is carried by the material points so that history-dependent constitutive models are easily implemented in the MPM.

The method has several advantages, the Lagrangian description provided by the material points naturally tracks the position of bodies and is capable of representing large deformations. Since no connectivity is required between the material points, mesh tangling is not a problem. Referring material-point data to a grid allows an efficient solution of the interactions (linear in the number of material points). Objects of any shape are defined by filling a region with material points, and the computational mesh does not have to conform to the object – greatly simplifying mesh construction. Since material points move in a single-valued velocity field determined from the grid solution, interpenetration of bodies is not possible. This feature makes MPM particularly suited to problems involving many bodies in contact, since no-slip contact can be enforced without a special algorithm.

MPM has been applied to impact, rebound and penetration or perforation problems, and to manufacturing problems such as metal rolling, cutting, extrusion and upsetting. It has also been modified to allow simulations of thin membranes [24, 25], and the contact algorithm has been extended to allow frictional contact. Simulations of granular material [26, 27] have particularly benefited from the addition of frictional contact. In simulations of membranes, material points are placed on the membrane surface. A local coordinate system, in the normal and tangential directions, is used at the surface to project the strain increment onto the plane of the membrane. In-plane stress is then determined from the in-plane strain. This paper describes a new method to compute the normal and tangential directions from the material-point distribution on the surface without needing connectivity among the points. An algorithm of this nature saves setup costs as well as the need to regenerate a mesh during simulations involving large membrane deformations.

Section 2 describes how solids, fluids and membranes are discretized using material points; then the next section presents the new algorithm for computing the normal to a surface from points on the surface, without resorting to meshing the surface with elements or a grid. Section 4 illustrates the construction of normals through a set of examples and also examines accuracy of the algorithm. Higher-order methods are explained in Section 5, with examples given in Section 6. Section 7 reviews a computational cycle of the MPM, in preparation for Section 8 where numerical examples in two dimensions compare simulations of membranes using the new algorithm for constructing surface normals with an algorithm using a surface mesh. Concluding remarks are made in the last section.

2 Initialization

Problem Specification

The MPM is designed to solve problems in continuum mechanics where the governing equations are conservation of mass, momentum and energy. Let a fluid or solid occupy a volume $\Omega(t)$ at time t. The mass density $\rho(\mathbf{x}, t)$, velocity $\mathbf{v}(\mathbf{x}, t)$, Cauchy stress $\boldsymbol{\sigma}(\mathbf{x}, t)$, the specific body force $\mathbf{b}(\mathbf{x}, t)$, and the specific internal energy $e(\mathbf{x}, t)$ are all functions of the current

position $\mathbf{x}$ and time, t. Conservation of mass can be written

$$\frac{d\rho}{dt} + \rho\nabla\cdot\mathbf{v} = 0; \tag{1}$$

linear momentum balance is

$$\rho\frac{d\mathbf{v}}{dt} = \nabla\cdot\boldsymbol{\sigma} + \rho\mathbf{b}; \tag{2}$$

and energy conservation, ignoring conduction and heat sources, is

$$\rho\frac{de}{dt} = \boldsymbol{\sigma} : \frac{d\boldsymbol{\epsilon}}{dt}. \tag{3}$$

The strain rate is related to the velocity gradient, $d\boldsymbol{\epsilon}/dt = \frac{1}{2}[(\nabla\mathbf{v}) + (\nabla\mathbf{v})^T]$. The equations above are to hold for all $\mathbf{x} \in \Omega(t)$. Fluids and solids are distinguished by the constitutive equation relating stress to strain or strain rate. A complete problem specification also requires initial conditions and boundary values.

A discretized version of these equations is required for a numerical solution. In the MPM, a solid body or volume of fluid is described by marking a set of material points in the initial configuration $\Omega(0)$ which are then followed throughout the remainder of the computation. (It is possible to replace one distribution of material points with another during a computation [28]; but, for simplicity, this feature will not be addressed in this paper.) Since the material points are unconnected, they provide a Lagrangian description that is able to represent large deformations without mesh tangling. The material points are also able to represent complicated shapes at low cost because it is easier to fill a region of space with points than it is to construct a mesh that conforms to complicated geometry. Interactions among the material points are computed on a background computational mesh.

Computational Mesh Construction

To begin the discretization, a computational domain of elements is constructed to contain the domain $\Omega(0)$. This mesh of elements can take any convenient form, such as a logically, hexahedral mesh in three dimensions. Let $\mathbf{x}_{i,j,k}$ denote the nodal position for node I, $I = (k-1)N_yN_x + (j-1)N_x + i$, where i, j, and k vary between $1 \le i \le N_x$, $1 \le j \le N_y$, and $1 \le k \le N_z$; respectively. We also use the more compact notation $\mathbf{x}_I$ to denote the position of node I. The index I takes on values from one to the number of nodes, $N_n = N_xN_yN_z$. Note that given I, N_x and N_y, the triple i, j, k can be recovered. Define logical coordinates (ξ, η, ζ) for a point $\mathbf{x}$ in the computational domain by mapping the unit cube onto the hexahedron containing $\mathbf{x}$ through the tensor product of linear shape functions,

$$\begin{aligned}
\mathbf{x}(\xi,\eta,\zeta) &= \zeta'\{\xi'[(1-\eta')\mathbf{x}_{i+1,j,k+1} + \eta'\mathbf{x}_{i+1,j+1,k+1}] \\
&+ \qquad (1-\xi')[(1-\eta')\mathbf{x}_{i,j,k+1} + \eta'\mathbf{x}_{i,j+1,k+1}]\} \\
&+ (1-\zeta')\{\xi'[(1-\eta')\mathbf{x}_{i+1,j,k} + \eta'\mathbf{x}_{i+1,j+1,k}] \\
&+ \qquad (1-\xi')[(1-\eta')\mathbf{x}_{i,j,k} + \eta'\mathbf{x}_{i,j+1,k}]\}.
\end{aligned} \tag{4}$$

In this expression, $\xi' = \xi - i$, $\eta' = \eta - j$, and $\zeta' = \zeta - k$. Equation 4 can be written as a tensor product of b-splines, $\bar{s}$,

$$\mathbf{x}(\xi, \eta, \zeta) = \sum_I \mathbf{x}_I \bar{s}(\xi - i)\bar{s}(\eta - j)\bar{s}(\zeta - k), \tag{5}$$

where

$$\bar{s}(\xi') = \begin{cases} 1 - |\xi'|, & \text{if } |\xi'| \leq 1 \\ 0, & \text{otherwise} \end{cases} . \tag{6}$$

The sum in Equation 5 is over all nodes, but only nodes in the support of $\bar{s}$ contribute.

The product of shape functions in Equation 5 can be written more compactly by introducing the notation

$$s(\boldsymbol{\xi} - \mathbf{I}) = \bar{s}(\xi - i)\bar{s}(\eta - j)\bar{s}(\zeta - k), \tag{7}$$

where the vector $\boldsymbol{\xi}$ has coordinates (ξ, η, ζ) and $\mathbf{I} = (i, j, k)$. Then, Equation 5 becomes

$$\mathbf{x}(\boldsymbol{\xi}) = \sum_I \mathbf{x}_I s(\boldsymbol{\xi} - \mathbf{I}). \tag{8}$$

For future reference, a corresponding shape function in physical space is defined by

$$S(\mathbf{x}(\boldsymbol{\xi}) - \mathbf{x}_I) = s(\boldsymbol{\xi} - \mathbf{I}). \tag{9}$$

Material Point Initialization

Suppose each region occupied by a fluid or solid is described by a set of inequalities: $g^r_\alpha(\mathbf{x}) \leq 0$, where $1 \leq \alpha \leq N_{er}$ and N_{er} is the number of inequalities that describes region r. To initialize the material points, sweep through logical space; place N_{pe} trial points in each element; determine the physical coordinates of the trial point via Equation 4; test each trial point to see if it is in a region (*i.e.* satisfies the inequalities for the region). If the point is in a region, keep it; otherwise, discard it. Points that are inside each region are initialized with material properties for that region, given a mass, initial velocity and possibly an initial stress or strain, depending on the initial conditions for the boundary value problem. The trial points are taken to be equally spaced in logical space; so, in logical space, each material point has a volume $1/N_{ep}$. The mapping, Equation 4, determines the initial volume Ω_p for the material point in physical space. This step gives the material points finite size. The mass, m_p, of a material point is determined using the initial density for the material making up region r, ρ^r_0, times the material point volume, Ω_p, so that $m_p = \rho^r_0 \Omega_p$. The number of material points per element, N_{ep} is an input parameter.

Unlike solid bodies or regions of fluid, membranes are thin structures that are modeled with one material point through the thickness [24, 25]. The initial position of a membrane can be described by the equation of a surface, $\mathbf{x} = \mathbf{x}(u, v)$, parameterized by u and v. The position $\mathbf{x}$ defines the midplane of the membrane which is assumed to have constant thickness. One way to discretize the surface is to place material points equally spaced along the coordinate directions with spacing Δu and Δv. The mass of a material point representing

the surface is the area of an element on the surface times the input density of the membrane (mass per unit volume), times the thickness. Material properties and initial conditions are attributed to each material point. The set of material points constructed on the membrane surface is kept in a list, but no connectivity information is saved or required for subsequent calculations.

3 Computation of Normals Using Linear Shape Functions

Evaluation of the membrane constitutive model requires determining a local coordinate system on the membrane surface, with coordinates given in the normal and tangential directions. The method for constructing the coordinate system described in this section is novel in that it does not rely on the construction of a surface mesh. The computational savings, particularly with two-dimensional surfaces in three-dimensional simulations, can be large – especially if the membrane surface is undergoing large deformations requiring frequent remeshing. The main idea is to regard the membrane surface as the isosurface of a scalar function, $\chi(\mathbf{x})$, defined in the whole domain. Given the value of χ at the material points representing the membrane, determine consistent nodal values. Consistency means that interpolation of nodal values to the membrane position gives the correct value of χ. The nodal values are used to define χ everywhere, and the gradient of χ evaluated at a surface material point gives the normal direction at that point.

More specifically, let χ_p denote the value of $\chi(\mathbf{x})$ at the point $\mathbf{x}_p$ on the membrane surface. Since we want the membrane to correspond to an isosurface of χ, take $\chi_p = 1$ for all points p on the surface. To have consistent nodal values χ_I, we require

$$\chi_p = \sum_I \chi_I S(\mathbf{x}_p - \mathbf{x}_I) \qquad \text{or} \qquad [\mathcal{S}]\{\chi\} = \{\chi^0\}. \tag{10}$$

The rectangular matrix $[\mathcal{S}]$ maps from nodes to membrane points, and has components $\mathcal{S}_{pI} = S(\mathbf{x}_p - \mathbf{x}_I)$. The vector $\{\chi\}$ has the nodal values as components, χ_I, and the vector $\{\chi^0\}$ has the material point values, χ_p, as components. If this equation can be solved for χ_I, then the normal to the surface at $\mathbf{x}_p$ is simply given by

$$\mathbf{n}_p = \sum_I \chi_I \nabla S(\mathbf{x} - \mathbf{x}_I)|_{\mathbf{x}_p}. \tag{11}$$

Typically, there will be many membrane points in an element so that the number of points p is greater than the number of nodes I. Thus, solve Eq. 10 in the least squares sense. To obtain the normal equations, multiply by $S(\mathbf{x}_p - \mathbf{x}_J)$ and sum over p,

$$\sum_I \chi_I \sum_p S(\mathbf{x}_p - \mathbf{x}_I) S(\mathbf{x}_p - \mathbf{x}_J) = \sum_p \chi_p S(\mathbf{x}_p - \mathbf{x}_J). \tag{12}$$

Eq. 12 can be written in matrix form

$$[\mathcal{T}]\{\chi\} = [\mathcal{S}]^T\{\chi^0\}, \tag{13}$$

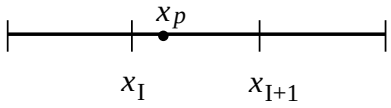


Figure 1: Geometry in one dimension. The element has nodes x_I and x_{I+1} with one material point, x_p, representing the "membrane."

where $[\mathcal{T}]$ is the square matrix with components $\mathcal{T}_{ij} = \sum_p S(\mathbf{x}_p - \mathbf{x}_I)S(\mathbf{x}_p - \mathbf{x}_J)$,
i.e. $[\mathcal{T}] = [\mathcal{S}]^T[\mathcal{S}]$.

Equation 12 can be solved globally, where p ranges over all material points making up the membrane and I and J range over all the nodes in the support of the shape functions centered at the material points. Interestingly, Equation 12 can also be solved element-by-element since only a local description of the surface is necessary to calculate the normal. In that case, for each element, p ranges over the number of material points in the element and I and J range over the nodes in the support of the shape function centered at the material points. So, for a hexahedral mesh in three dimensions, using linear shape functions, the system size is 8×8. The size is 4×4 in two-dimensions, on a quadrilateral mesh with linear shape functions. Actually, any convenient subset of material points and corresponding elements can be used to determine the normal to the surface; this observation will be exploited later.

4 Examples Using Linear Shape Functions

In this section, some examples are used to demonstrate the feasibility of using Equation 10 or Equation 13, to determine nodal values of χ that correspond to $\chi = 1$ on a given surface marked by material points, and then using Equation 11 to determine the normal to the surface. To begin, consider a simple example in one dimension. Although simple, this example is nevertheless, illustrative.

One Dimension

Figure 1 shows a "membrane" which in one dimension is represented by a single point labeled x_p in an element with endpoints x_I and x_{I+1}. In one dimension, Equation 10 reduces to

$$\chi(\xi) = \sum_I \chi_I S(x - x_I) = (1 - \xi)\chi_I + \xi\chi_{I+1},$$

where $\xi(x) = (x - x_i)/(x_{I+1} - x_I)$. The nodal values χ_I and χ_{I+1} are determined so that $\chi(\xi_p) = 1$ where $\xi_p = \xi(x_p)$. This gives only one equation to solve for two unknowns. The equation is

$$\begin{bmatrix} 1 - \xi_p & \xi_p \end{bmatrix} \begin{Bmatrix} \chi_I \\ \chi_{I+1} \end{Bmatrix} = \begin{Bmatrix} 1 \end{Bmatrix}. \tag{14}$$

An obvious solution to this problem is $\chi_I = \chi_{I+1} = 1$. In fact, the value one for all nodes is a solution to the general problem, Equation 10 or Equation 13. Obviously, constant χ does not have a gradient so it is useless for computing a normal direction.

To set up a gradient, choose a random point in the element, $x_q \neq x_p$, and assign the value $\chi_q = 0$ to it. There are now two equations to solve for the two unknowns,

$$\begin{bmatrix} 1-\xi_p & \xi_p \\ 1-\xi_q & \xi_q \end{bmatrix} \begin{Bmatrix} \chi_I \\ \chi_{I+1} \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix},$$

which has the solution

$$\chi_I = \frac{-\xi_q}{\xi_p - \xi_q} \qquad \chi_{I+1} = \frac{1-\xi_q}{\xi_p - \xi_q},$$

as long as $\xi_p \neq \xi_q$. The corresponding solution for χ everywhere is

$$\chi(\xi) = (1-\xi)\frac{-\xi_q}{\xi_p - \xi_q} + \xi\frac{1-\xi_q}{\xi_p - \xi_q},$$

which satisfies $\chi(\xi_p) = 1$ exactly. A normal can be computed from the gradient of χ,

$$\frac{d\chi}{dx} = \frac{d\xi/dx}{\xi_p - \xi_q}, \qquad \frac{d\xi}{dx} = \frac{1}{x_I - x_{I+1}}.$$

A unit normal is determined by normalizing the gradient. The unit normal can have either sign, meaning that either an inward or an outward normal is computed at the surface.

Note that the least squares problem associated with Equation 14 is

$$\begin{bmatrix} (1-\xi_p)^2 & (1-\xi_p)\xi_p \\ \xi_p(1-\xi_p) & \xi_p^2 \end{bmatrix} \begin{Bmatrix} \chi_I \\ \chi_{I+1} \end{Bmatrix} = \begin{Bmatrix} 1-\xi_p \\ \xi_p \end{Bmatrix}.$$

This system is singular, but consistent. There is a one parameter family of solutions, parameterized by the value of χ_{I+1}, and with

$$\chi_I = \frac{-\xi_p}{1-\xi_p}\chi_{I+1} + \frac{1}{1-\xi_p}.$$

The corresponding solution for χ everywhere is

$$\chi(\xi) = \frac{-\xi_p(1-\xi)}{1-\xi_p}\chi_{I+1} + \frac{1-\xi}{1-\xi_p} + \xi\chi_{I+1},$$

which satisfies $\chi(\xi_p) = 1$. Note also that $\chi_I = 1$ when $\chi_{I+1} = 1$. The gradient of χ is nonzero, except when $\chi_{I+1} = 1$, and can be used to compute a normal:

$$\frac{d\chi}{dx} = \frac{d\xi}{dx}\frac{\chi_{I+1} - 1}{1-\xi_p}, \qquad \frac{d\xi}{dx} = \frac{1}{x_I - x_{I+1}}.$$

So, the least squares solution can be used in the underdetermined case, except for the constant solution, and care must be taken when solving the singular system of equations. Adding a random point appears to be the better approach.

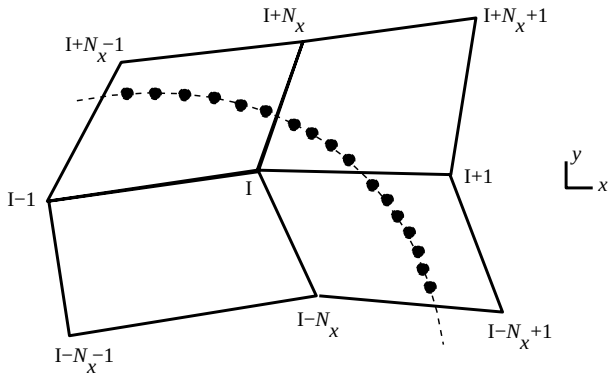


Figure 2: Geometry in two dimensions. The mesh is composed of quadrilateral elements and the "membrane" is represented by a set of material points along a curve.

Two Dimensions

In two dimensions, we consider a mesh of quadrilaterals and a "membrane" represented by points along a curve. Figure 2 illustrates the geometry. If the normals to the membrane are to be computed element-by-element, then there are four unknowns for element I, the four nodal values χ_I, χ_{I+1}, χ_{I+N_x}, and χ_{I+N_x+1}. The system of equations will be underdetermined unless there are at least three membrane points in the element. The fourth equation will come from adding a random point with χ assigned the value zero there. If there are not enough membrane points within an element, then a group of four elements can be joined together and considered as one larger element, presumably with enough membrane points to determine χ. If not, the process can be repeated until enough membrane points are present. Using linear shape functions, straight lines in logical space should be represented exactly. A series of tests are conducted to examine the sensitivity to the placement of the random point. Table I summarizes the results. Three membrane points are placed along a line in an element. Lines have varying slope. For each line, 100 realizations of the random point are generated. In every case, correct normals to the membrane point are computed to within roundoff. However, the condition number of the system (*i.e.* the condition number of the normal equations matrix $\mathcal{T}$ in Equation 13) varies with the position of the random point. Figure 3(a-b) shows two realizations for the case of a line with slope one. The open circles denote the membrane points, the asterisk indicates the random point and the contour lines denote values of χ throughout the element. Note that the membrane points lie on a contour line with $\chi = 1$ and the random point is on a line with $\chi = 0$, as expected. The condition number of the system is larger if the random point is close to being on the membrane. It is possible to have a large condition number, but it is generally within reason, as evidenced by the low median condition number over all the trials. It is possible to monitor the condition number and regenerate the random point if the condition number is deemed too large; however, we have not found this to be necessary in our simulations to date.

For each slope, we have also tested overdetermined cases where each line is represented by 4 or 8 points within the element. Again, the normals to the line at each point are determined

slope	condition number		
	minimum	maximum	median
0.1	3.7e+03	1.3e+08	5.0e+03
0.5	7.8e+01	4.6e+07	1.7e+02
1.0	3.5e+01	4.8e+05	5.9e+01
2.0	2.3e+02	7.3e+16	3.6e+02
10.0	3.7e+03	1.1e+06	5.4e+03
-10.0	3.6e+03	5.1e+06	5.3e+03
-2.0	2.3e+02	1.4e+07	3.3e+02
-1.0	3.5e+01	3.6e+07	6.1e+01
-0.5	3.5e+01	3.6e+07	1.2e+02
-0.1	3.7e+03	2.5e+06	6.0e+03

Table I: Condition number for the normal equations over 100 realizations of the random point for membranes represented by 3 points along lines with a variety of slopes. Normals to these lines can be computed exactly in every case using linear shape functions.

exactly, but the condition number of the system varies with the placement of the random point. A sample of the results is given in Table II, and Figure 3(c-d) shows examples with 4 and 8 points for a line of slope one.

Figure 4(a) shows the results for a nonlinear configuration of membrane points. The membrane location is defined by placing points along the curve $xy + 0.1x + 0.3 = 0$. The bilinear term xy can also be represented exactly for a unit square element by the tensor product of linear shape functions; so, as expected the normals are computed accurately in this case, even with only three membrane points representing the curve. In Figure 4(b), three membrane points are placed along a circle of radius 0.8 centered at the origin. Now, the contour line $\chi = 1$ differs slightly from the exact circular arc, and the normals have an error of 7^o maximum. Moving the random point has little effect on the maximum error; the condition number over 100 trials varies between 1.8e+01 and 4.1e+06, with a median value of 1.1e+02. If the position of the random point is held fixed, as in Figure 4(b), but the number of membrane points is increased, the maximum error also appears to be unaffected; although, the average error decreases. The results of varying the number of membrane points are summarized in Table III. In all cases, the maximum error in the computed normal occurs at the ends of the membrane segment.

Three Dimensions

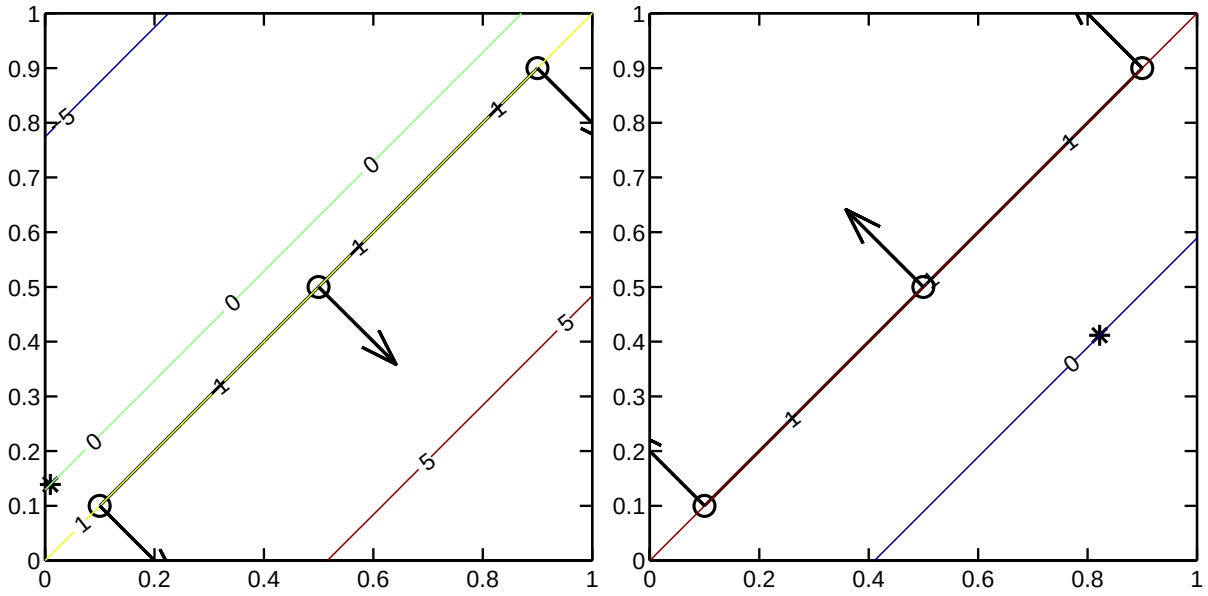
In three dimensions, the normal equations provide an 8 by 8 system of equations that must be solved for the eight nodal values of χ for one element on a hexahedral mesh. The system is underdetermined unless there are at least 7 membrane points and one random point in the element. Examples using membrane points within a unit cube illustrate the construction.

slope 1.0	condition number		
# points	minimum	maximum	median
4	3.2e+01	3.2e+05	7.9e+01
8	3.6e+01	1.7e+09	6.5e+01
slope 2.0	condition number		
# points	minimum	maximum	median
4	1.3e+02	2.9e+06	1.9e+02
8	1.7e+02	9.9e+05	3.4e+02
slope 10.0	condition number		
# points	minimum	maximum	median
4	4.1e+03	5.4e+04	5.8e+03
8	5.7e+03	2.2e+05	7.7e+03

Table II: Condition number over 100 realizations of the random point for membranes represented by 4 or 8 points along lines with a variety of slopes. Normals to these lines can be computed exactly in every case using linear shape functions.

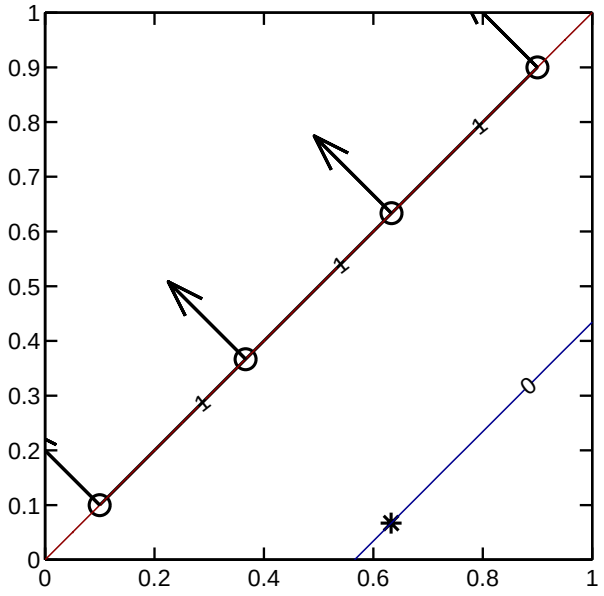
# of points	max error	rms error
3	0.1211	0.0571
4	0.1091	0.0434
8	0.1177	0.0231
16	0.1261	0.0141
32	0.1308	0.0092

Table III: Errors using a varying number of membrane points along a circular arc, keeping the random point fixed as in Figure 4(b). Errors are in radians and measure the deviation of the computed normal from the exact normal to the circle.

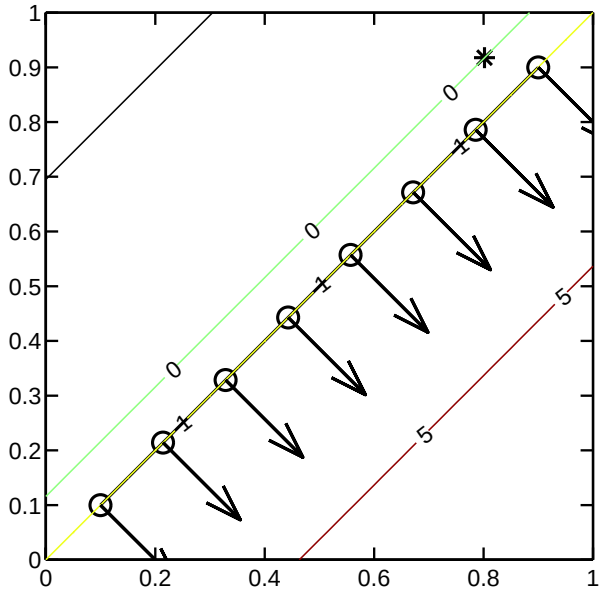


(a) condition number 1.6e+03

(b) condition number 4.5e+01



(c) condition number 3.4e+01



(d) condition number 5.6e+02

Figure 3: Examples of membrane points located along a line of slope one. In (a) and (b) three membrane points are used with two different placements of the random point. In (c) four membrane points are used and in (d) eight are used. Plots show contour lines of χ and the computed normals which have no error using linear shape functions. The exact normals are also plotted but they are indistinguishable from the computed normals.

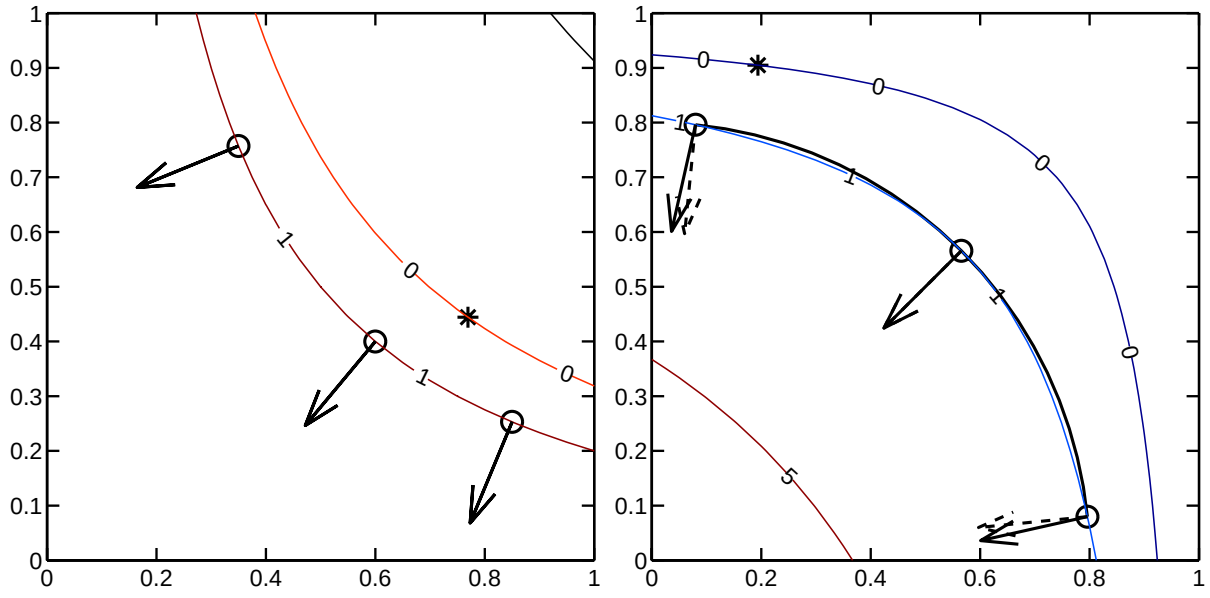


Figure 4: Three membrane points located on the curve (a) $xy + 0.1x + 0.3 = 0$ and (b) $x^2 + y^2 = 0.8^2$. In (a) the normals are computed exactly and in (b) the maximum error is 7^o between the exact (dashed) and computed (solid) normals.

Figure 5(a) shows computed normals where nine membrane points are placed on the plane $0.5x + 0.25y - z = 0$. The linear shape functions produce normals to the surface without error since the plane can be represented exactly. When membrane points are placed along the surface of a sphere $x^2 + y^2 + z^2 = 0.8^2$, Figure 5(b), there is an error in the computed normals. Using 8 membrane points to represent the sphere gives a maximum error of 13^o in the direction of the computed normal, with an rms error of 3^o.

Limitations of Linear Shape Functions

The examples considered so far show that the method can give satisfactory normals. However, there are limitations to using linear shape functions. For example, there are configurations of points that cause the matrix in Equation 13 to be singular or nearly singular. For example, consider a square element of side length one with membrane points on a vertical

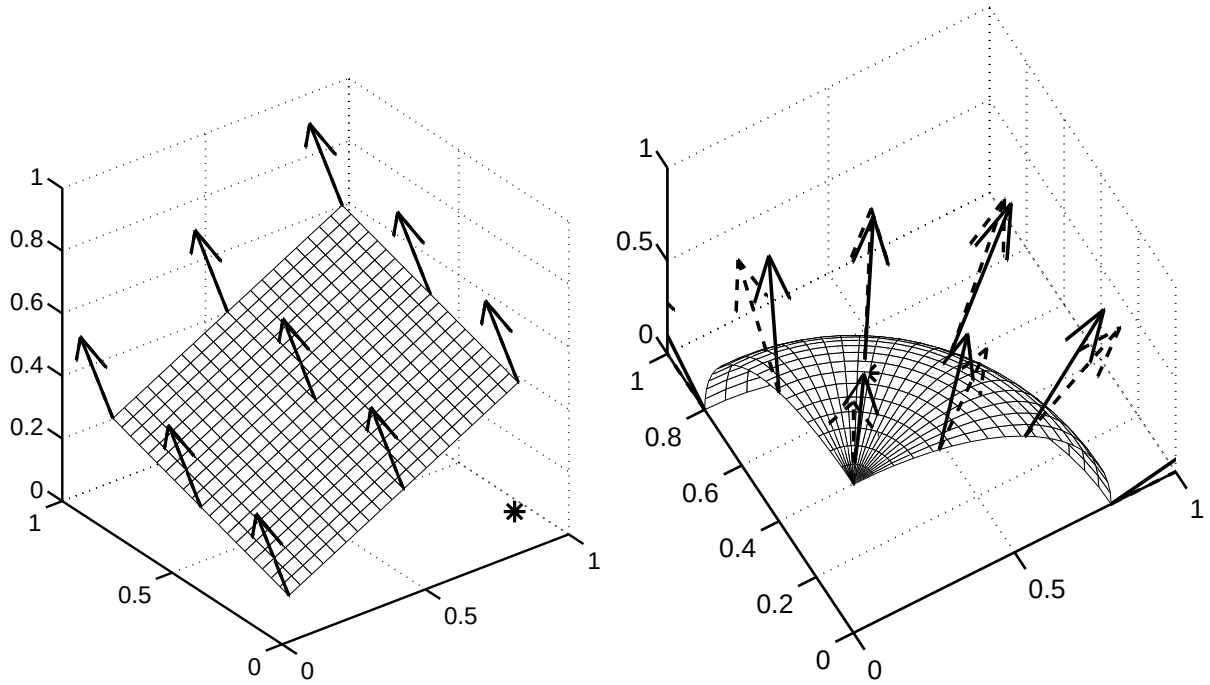


Figure 5: Construction of normals to surfaces in three dimensions using linear shape functions. Membrane points placed on (a) the plane, $0.5x + 0.25y - z = 0$, and (b) the sphere, $x^2 + y^2 + z^2 = 0.8^2$. Exact normals (dashed arrows) and computed normals (solid arrows) are shown.

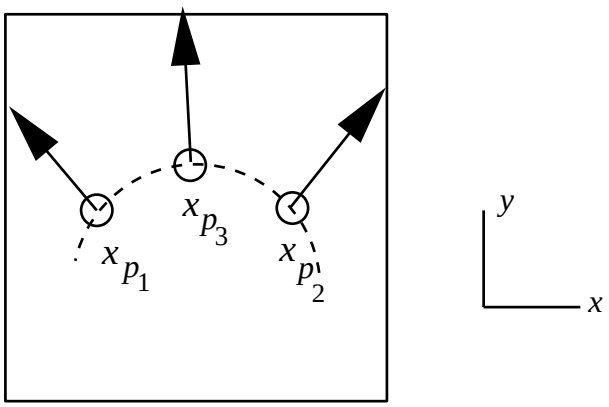


Figure 6: Three membrane points located a curve where x_{p_1} and x_{p_2} have the same logical coordinate, η. Exact normals are shown. Clearly, the normals at x_{p_1} and x_{p_2} do not have the same x-component, so normals computed using linear shape functions will be inaccurate for this configuration.

or horizontal line segment. The system to be solved, Equation 10, has the form

$$\begin{bmatrix} \xi_1(1-\eta_1) & \xi_1\eta_1 & (1-\xi_1)\eta_1 & (1-\xi_1)(1-\eta_1) \\ \xi_2(1-\eta_2) & \xi_2\eta_2 & (1-\xi_2)\eta_2 & (1-\xi_2)(1-\eta_2) \\ \vdots & \vdots & \vdots & \vdots \\ \xi_n(1-\eta_n) & \xi_n\eta_n & (1-\xi_n)\eta_n & (1-\xi_n)(1-\eta_n) \\ \xi_{n+1}(1-\eta_{n+1}) & \xi_{n+1}\eta_{n+1} & (1-\xi_{n+1})\eta_{n+1} & (1-\xi_{n+1})(1-\eta_{n+1}) \end{bmatrix} \begin{Bmatrix} \chi_{i+1,j} \\ \chi_{i+1,j+1} \\ \chi_{i,j+1} \\ \chi_{i,j} \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \end{Bmatrix} \tag{15}$$

where (ξ_k, η_k), $k = 1, \ldots, n$ are the coordinates of the membrane points, and (ξ_{n+1}, η_{n+1}) are the coordinates of the random point. For a horizontal line, $\eta_1 = \eta_2 = \ldots = \eta_n$. Without the last row of the matrix, the first and fourth columns are multiples of one another, as are the second and third. Hence, this matrix is nearly rank deficient and therefore difficult to handle numerically. Similarly for membrane points along a vertical line segment. It is worth noting that perturbing the points generally enables a solution and represents the more likely scenario in a numerical simulation of membranes.

A more serious limitation of linear shape functions exists, even with systems that can be solved for nodal values of χ. Again, consider a unit element. Equation 11, written in components, is

$$((n_p)_x, (n_p)_y)) = (a + b\eta, c + b\xi),$$

where a, b and c are constants that depend on the nodal values of χ,

$$a = \chi_{i+1,j} - \chi_{i,j} \qquad b = \chi_{i+1,j+1} - \chi_{i+1,j} + \chi_{i,j} - \chi_{i,j+1} \qquad c = \chi_{i,j+1} - \chi_{i+1,j+1}.$$

Notice that the x-component, $(n_p)_x$, only depends on η, and the y-component, $(n_p)_y$, only depends on ξ. Consider membrane points as in Figure 6. Since x_{p_1} and x_{p_2} have the same

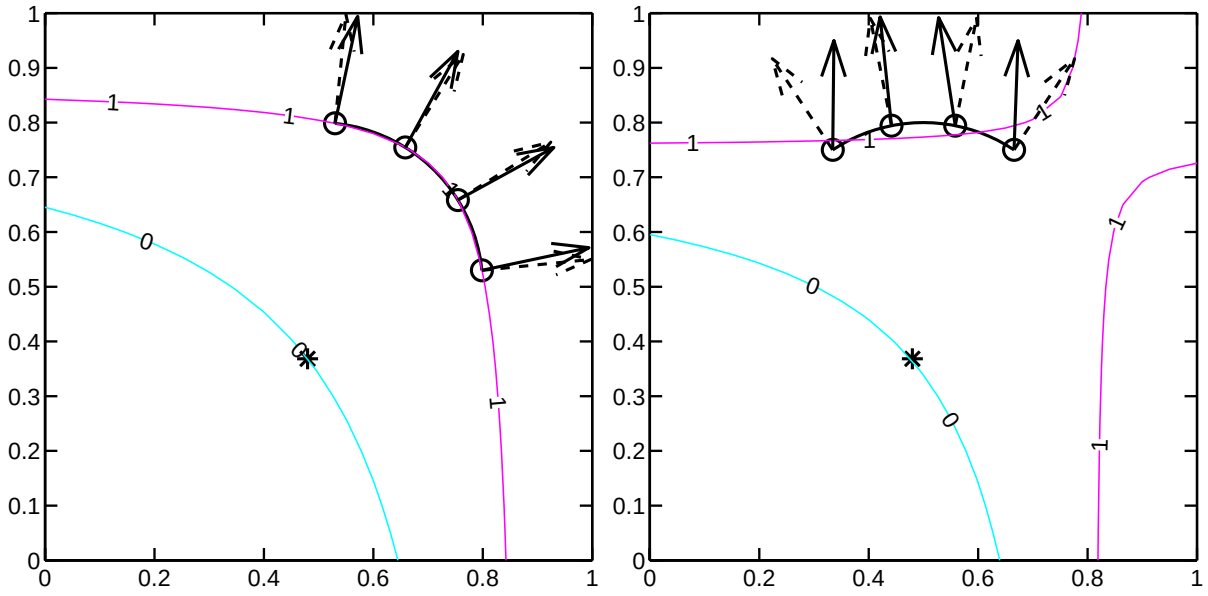


Figure 7: Membrane points are located along a circular arc, $(x-0.5)^2+(y-0.5)^2=0.3^2$. In (a) the x and y components of the membrane points are unique whereas in (b) pairs of points have the same y component. The maximum error is 6^o in (a) and 34^o in (b). The plot shows contours of χ and the computed (solid) and exact normals (dashed).

value of η, no matter how the nodal values of χ are determined, $(n_{p_1})_x$ will equal $(n_{p_2})_x$. Clearly having the same x-component for the normal at x_{p_1} and x_{p_2} is inaccurate in this case. Similar inaccuracies occur if two membrane points have the same y-component. Figure 7 shows two cases where membrane points are located along a circular arc where the center of the circle is at $(0.5, 0.5)$ and the radius is 0.3. In Figure 7(a) the x and y components of the membrane points are unique whereas in (b) pairs of points have the same y component. The inaccuracy in the second case is apparent.

5 Computation of Normals Using High-Order Shape Functions

Linear b-splines that have been used as shape functions in the construction of normals in the last section are one member in a family of b-splines generated by a recursion formula [29]. Starting with splines of order zero,

$$\bar{s}^{(0)}(\xi) = \begin{cases} 1 & 0 \le |\xi| \le \frac{1}{2} \\ 0 & |\xi| > \frac{1}{2} \end{cases}$$

the recursion formula for b-splines of order $n+1$ is

$$\bar{s}^{(n+1)}(\xi) = \int \bar{s}^{(n)}(\xi')\bar{s}^{(0)}(\xi-\xi')d\xi'.$$

Using zeroth order splines is also known as nearest-grid-point interpolation. The first-order spline, $\bar{s}^{(1)}$, is the same function used previously in Equation 6. Quadratic b-splines are given by,

$$\bar{s}^{(2)}(\xi) = \begin{cases} \frac{3}{4} - \xi^2 & 0 \le |\xi| \le \frac{1}{2} \\ \frac{1}{2}(\frac{3}{2} - |\xi|)^2 & \frac{1}{2} \le |\xi| \le \frac{3}{2} \\ 0 & |\xi| > \frac{3}{2} \end{cases}.$$

The support of a b-spline of order n is $[-\frac{n+1}{2}, \frac{n+1}{2}]$. The support of $\bar{s}^{(n)}$ when n is even makes it more convenient to interpolate to element centers rather than to nodes since interpolating to nodes requires additional logic to determine in which octant (in three dimensions) a point lies. When n is odd, interpolation to nodes is more convenient, as with $n = 1$. For the hexahedral element defined by Equation 4, the element center is the point $\mathbf{x}_C = \mathbf{x}(\xi + 1/2, \eta + 1/2, \zeta + 1/2)$. A basis function using quadratic shape functions can be written

$$s^{(2)}(\boldsymbol{\xi} - \mathbf{C}) = \bar{s}^{(2)}(\xi - i - \frac{1}{2})\bar{s}^{(2)}(\eta - j - \frac{1}{2})\bar{s}^{(2)}(\zeta - k - \frac{1}{2}),$$

where the vector $\mathbf{C} = (i + 1/2, j + 1/2, k + 1/2)$ gives the logical coordinates of the element center. The corresponding shape function in physical space is

$$S^{(2)}(\mathbf{x}(\boldsymbol{\xi}) - \mathbf{x}_C) = s^{(2)}(\boldsymbol{\xi} - \mathbf{C}).$$

Higher order b-splines are used similarly to linear shape functions for determining χ and $\nabla\chi$ as in Equations 10 and 11. Using quadratic shape functions, we want to determine values of χ_C at element centers so that

$$\chi_p = \sum_C \chi_C S^{(2)}(\mathbf{x}_p - \mathbf{x}_C),$$

where the sum is over all elements. If we assign χ_p the value one, then again the membrane surface should roughly correspond to an isosurface. The normal to this surface at $\mathbf{x}_p$ comes from taking the gradient,

$$\mathbf{n}_p = \sum_C \chi_C \nabla S^{(2)}(\mathbf{x} - \mathbf{x}_C)|_{\mathbf{x}_p}. \tag{16}$$

6 Examples Using Quadratic Shape Functions

Using quadratic shape functions in one dimension, a single membrane point in element I contributes to three element centers that have logical coordinates, $I-1/2$, $I+1/2$ and $I+3/2$, Figure 8. Since there is only one equation for three values of χ, there is a two parameter family of solutions. The normal equations will be a 3 by 3 singular, but consistent system. Adding a random point still leaves the system underdetermined with a one parameter family of solutions.

In two dimensions the least squares problem is 9 by 9 and will be underdetermined unless there are at least 8 membrane points along with one random point. As before, if there are

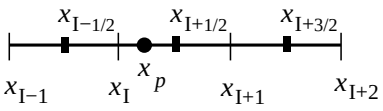


Figure 8: Geometry in one dimension. Element I has nodes x_I and x_{I+1} with one material point, x_p, representing the "membrane." A quadratic shape function overlaps element centers at $x_{I-1/2}$, $x_{I+1/2}$ and $x_{I+3/2}$

.

not enough membrane points, elements can be combined to form a larger element containing more points. Figure 9 shows examples where membrane points are placed along the arc of a circle as in Figure 6. Figure 9(a-b) is the same geometry as Figure 7 except that quadratic shape functions are used to determine χ. Even though the system is underdetermined, we are still able to obtain a solution. In (a) the maximum error is about 3^o and in (b) the maximum error in the direction of the normal is about 19^o. With eight membrane points along the same circular arc, (c) and (d), the system is nonsingular and the error is zero. The calculations for Figure 9 are done by solving the normal equations. Using a QR-decomposition to solve the least squares problem [30] leads to a better conditioned system; and the result for Figure 9(b) is a maximum error of less than one degree.

Exact solutions are not expected using quadratic shape functions with membrane points placed along the cubic curve $y = cx^2(x-1) + x + 0.5$, where c is a constant. Figure 10 shows two examples with $c = 2$ and $c = 4$. In the first case, the maximum error is about 1^o and in the second it is 32^o, with an rms error of about 5^o. The larger error in the second case is attributed to smaller radius of curvature. With $c = 2$, the minimum radius of curvature is about 0.5, and with $c = 4$, it is about 0.2. To have a well resolved membrane, the radius of curvature should roughly be at least on the order of the computational mesh size. The second case is not resolved by the mesh.

In three dimensions, the normal equations will be a system of size 27 by 27 using quadratic shape functions. Figure 11 shows examples where the element is a unit cube, and membrane points are located on a plane and on the surface of a sphere. In Figure 11(a), 30 membrane points are used on the plane $0.5x + 0.25y - z = 0$. Figure 11(b) uses 29 membrane points placed on the surface of the sphere $x^2 + y^2 + z^2 = 0.8^2$ The normals are computed exactly in both cases.

7 Summary of the MPM Computational Cycle

The preceeding algorithm for constructing normals to a surface is combined with the MPM to simulate fluid-filled membranes. In this Section, the MPM algorithm is summarized; details can be found in the references. Initialization of the material points is described in Section 2. Let $\mathbf{x}_p^n$, $p = 1, \ldots, N_p$ denote the current position of material point p at time t^n. Each point at time t^n has an associated mass, m_p, density, ρ_p^n, velocity, $\mathbf{v}_p^n$, Cauchy stress tensor, $\boldsymbol{\sigma}_p^n$, strain, $\mathbf{e}_p^n$, and any other internal variables necessary for the constitutive

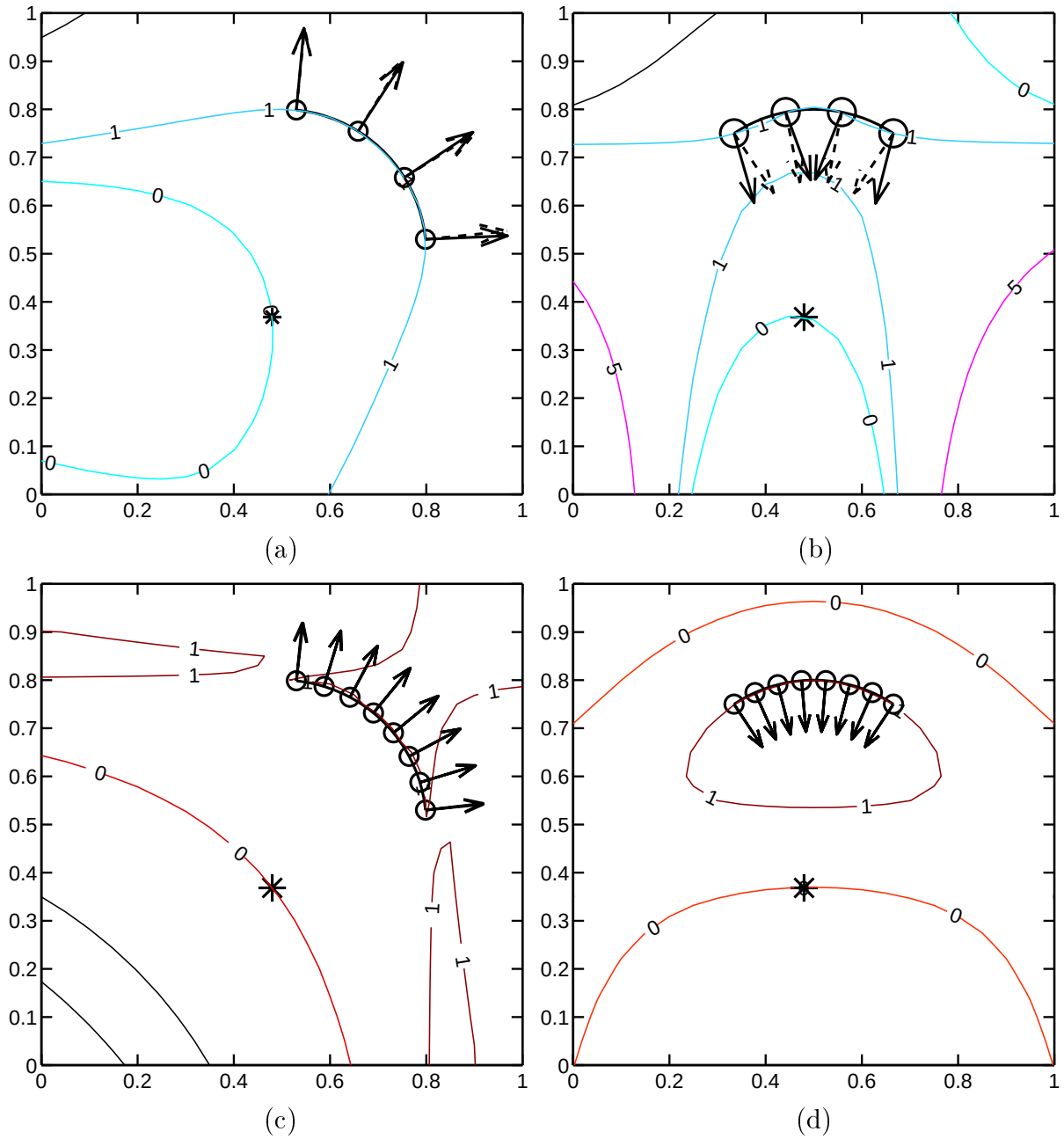


Figure 9: Examples of membrane points located along an arc of the circle $(x-0.5)^2+(y-0.5)^2=0.3^2$. Plots show contour lines of χ, computed normals (solid) using quadratic shape functions, and exact normals (dashed). The maximum error is (a) 3^o and (b) 19^o for the underdetermined systems and zero in (c) and (d) for the nonsingular systems.

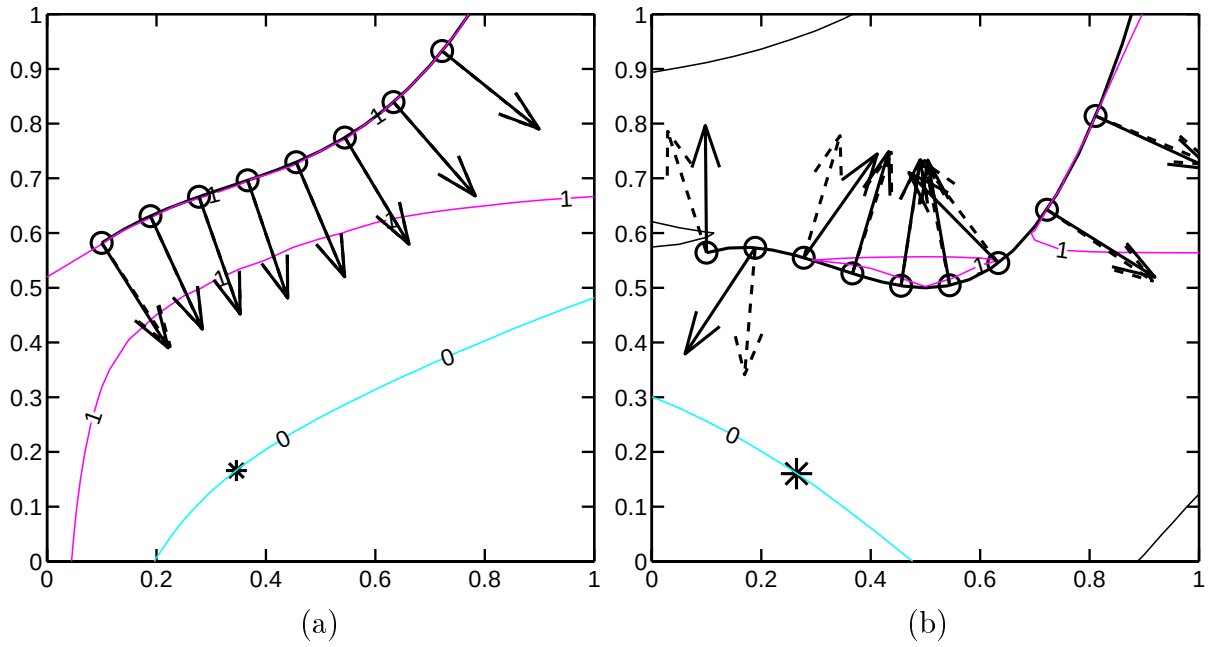


Figure 10: Examples using quadratic shape functions with membrane points located along a cubic curve, $y = cx^2(x-1) + x + 0.5$. In (a) $c = 2$ and the maximum error is 1^o, and in (b) $c = 4$ and the maximum error is 32^o. Plots show contour lines of χ, computed normals (solid) and exact normals (dashed).

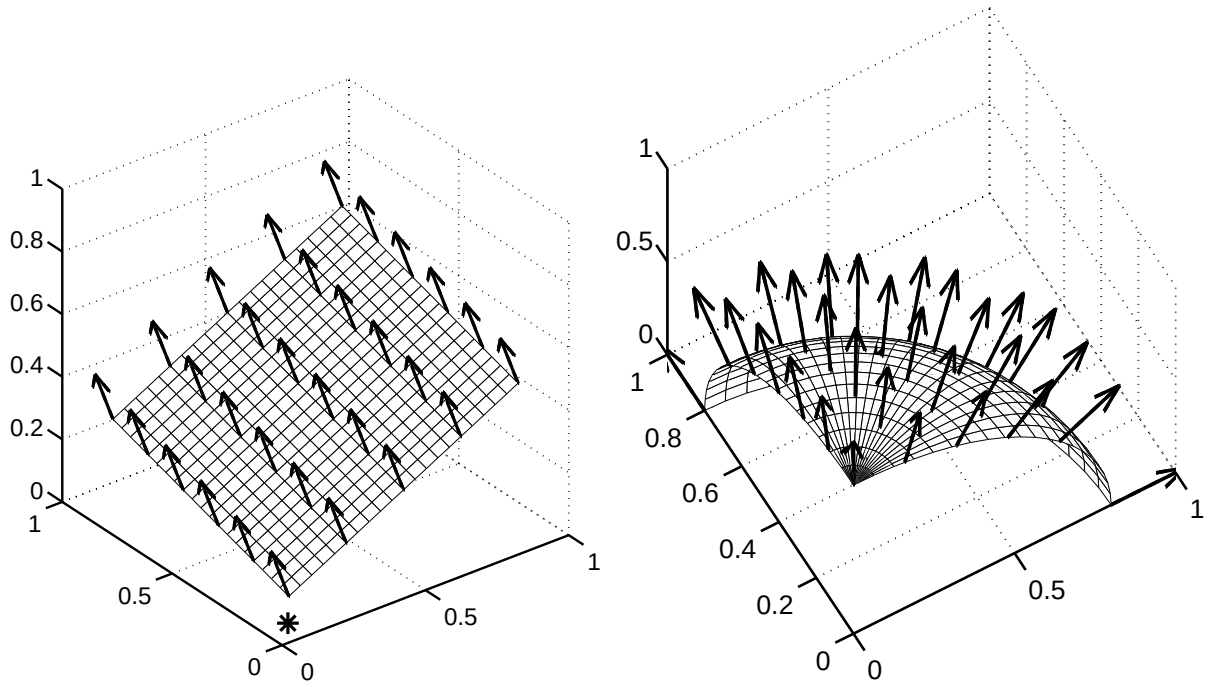


Figure 11: Examples using quadratic shape functions with (a) 30 membrane points located on the plane $0.5x+0.25y-z=0$, and (b) 29 points on the surface of the sphere $x^2+y^2+z^2=0.8^2$. The computed normals are exact.

model. If temperature changes are important, internal energy or temperature may also be ascribed to the material points. The material point mass is constant in time, insuring that the continuity equation is satisfied. Other variables must be updated with reference to conservation of momentum, conservation of energy, or from the constitutive model.

To make the computations tractable, at each time step of a dynamic algorithm, information from the material points is interpolated to a background computational mesh. This mesh covers the computational domain and is chosen for computational convenience. The mesh, described previously in Section 2, is composed of hexahedral elements in three dimensions, and quadrilateral elements in two dimensions. After information is interpolated to the grid, equations of motion are solved on this mesh which is considered to be an updated Lagrangian frame.

To solve the momentum equation on the grid using an explicit FE algorithm, one must know the value of the momentum at the beginning of the time step at the nodal positions. The nodal momentum, $m_I^n \mathbf{v}_I^n$, is the product of the nodal mass and nodal velocity, and each is determined by interpolation,

$$m_I^n = \sum_{p=1}^{N_p} m_p S(\mathbf{x}_p^n - \mathbf{x}_I)$$

$$m_I^n \mathbf{v}_I^n = \sum_{p=1}^{N_p} m_p \mathbf{v}_p^n S(\mathbf{x}_p^n - \mathbf{x}_I).$$

In the above, $S(\mathbf{x} - \mathbf{x}_I)$ is the nodal basis function associated with node I introduced previously (Equation 9).

The internal forces are determined from the particle stresses according to

$$\mathbf{f}_I^{\text{int}} = -\sum_{p=1}^{N_p} \mathbf{G}_{Ip}^n \boldsymbol{\sigma}_p^n m_p / \rho_p.$$

The quantity $\mathbf{G}_{Ip}^n$ is the gradient of the nodal basis function evaluated at the material point position, $\mathbf{G}_{Ip}^n = \nabla S(\mathbf{x} - \mathbf{x}_I)|_{\mathbf{x}_p^n}$. The momentum equation is solved with the nodes considered to be moving with the deformation to give nodal velocities, $\mathbf{v}_I^L$, at the end of this Lagrangian time step of size Δt,

$$m_I^n \frac{\mathbf{v}_I^L - \mathbf{v}_I^n}{\Delta t} = \mathbf{f}_I^{\text{int}} + m_I^n \mathbf{b}_I^n.$$

The external forces at the nodes, $\mathbf{b}_I^n = \mathbf{b}(\mathbf{x}_I)$, are computed easily.

At the end of this Lagrangian step, the new nodal values of velocity are used to update the material points. The material points move along with the nodes according to the solution given throughout the elements by the nodal basis functions

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \sum_{I=1}^{N_n} \mathbf{v}_I^L S(\mathbf{x}_p^n - \mathbf{x}_I).$$

Similarly, the material point velocity is updated via

$$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n + \sum_{I=1}^{N_n} \{\mathbf{v}_I^L - \mathbf{v}_p^n\} S(\mathbf{x}_p^n - \mathbf{x}_I).$$

The sums in these last two equations extend from 1 to N_n where N_n is the number of nodes in the computational mesh.

A strain increment for each material point is determined using the gradient of the nodal basis function,

$$\Delta \mathbf{e}_p = \frac{\Delta t}{2}(\mathbf{G}_{Ip}^n \mathbf{v}_I^L + (\mathbf{G}_{Ip}^n \mathbf{v}_I^L)^T).$$

This strain increment is then used in an appropriate constitutive equation for the material being modeled to update the stress at the material point. The constitutive equation for a membrane point is similar to that for a solid, except the strain increment is projected onto the local, normal-tangential coordinate system for the membrane and only the tangential components of the strain contribute to the stress [24, 25]. Any internal variables necessary in the constitutive model can also be assigned to the material points and transported along with them. Once the material points have been completely updated, the computational mesh may be discarded and a new mesh defined, if desired, and then the next time step is begun.

The material point method has several advantages. The Lagrangian description provided by the material points can undergo large deformations without mesh tangling because the points are not connected. Since the computational mesh is under user control, it can be chosen so that reasonable time steps may be taken in this Lagrangian frame. Usually, the time step is restricted by the CFL condition for an explicit algorithm, where the critical time step is the ratio of the mesh size to the wave speed. Note that this condition depends on the more favorable mesh spacing, not the material point spacing. Since equations are solved in an updated Lagrangian frame on the FE mesh, the nonlinear convective terms troublesome in Eulerian formulations, are not an issue. Finally, the material points transport material properties and internal variables without error.

8 Numerical Examples

The two-dimensional MPM code is capable of treating solids with a variety of constitutive models, membranes and polytropic gasses. In this section, simulations using this code on fluid-filled membranes demonstrate the calculation of normals in a dynamic setting. We have implemented the construction of normals in an element-by-element manner, for elements containing membrane points, as detailed in the preceeding sections of this paper. More specifically, a hybrid algorithm is employed. Quadratic b-splines are generally used in the construction of the normals, provided the element has at least seven membrane points. If there are fewer than seven membrane points in the element, but more than three membrane points, linear b-splines are used. If an element contains fewer than three membrane points, four neighboring elements are joined together to form a larger element and linear b-splines

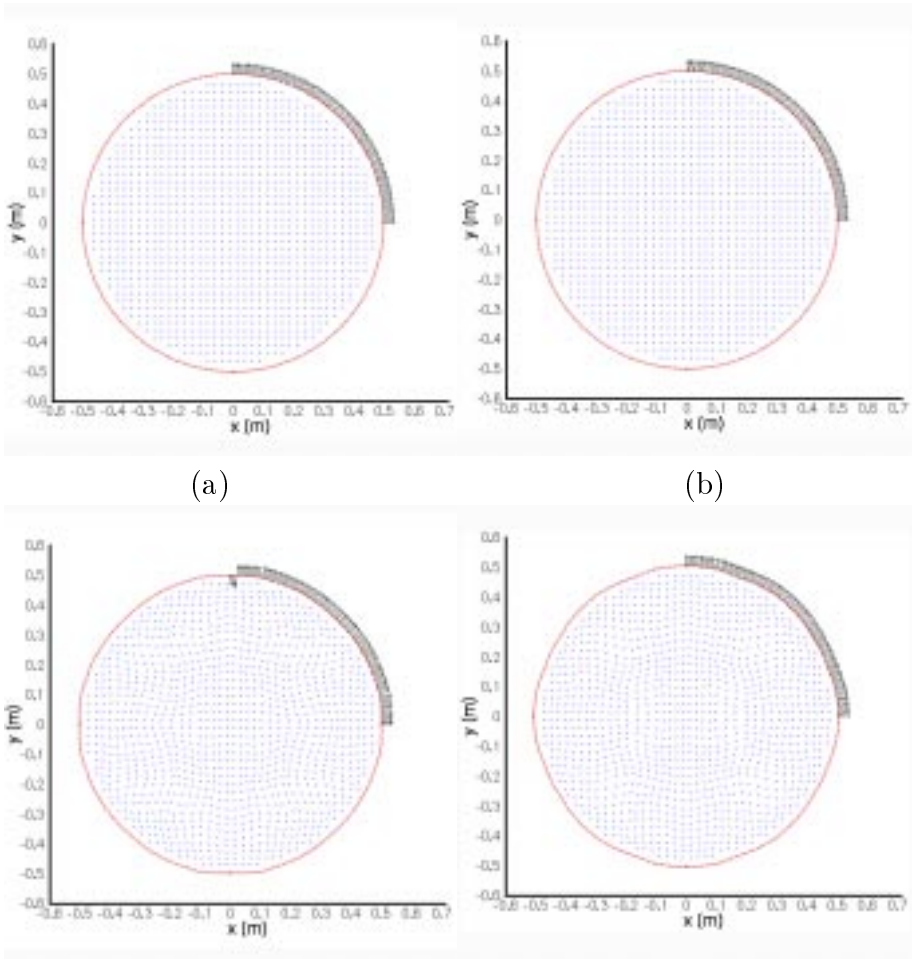
are used in conjunction with all the membrane points in this larger element. This strategy avoids underdetermined systems of equations.

The algorithm for computing normals without connectivity is compared with a more standard approach using connectivity of the membrane points. In two dimensions, the membrane is represented by a line of material points, as sketched in Figure 2. The material points can be generated in a list where the neighboring material points are the neighbors in the list (*i.e.* the material point with position $\mathbf{x}_p$ has neighbors, $\mathbf{x}_{p-1}$ and $\mathbf{x}_{p+1}$). If there are no ruptures, this list is maintained and connectivity is easy to track. The unit tangent vector to the membrane surface is also easily computed; it is the vector in the direction $\mathbf{x}_{p+1} - \mathbf{x}_{p-1}$, normalized by its length.

The first test problem involves an initially slack circular membrane filled with a pressurized gas. Due to symmetry, only one quarter of the circle is used in the simulation. Fig. 12 shows the initial configuration of 91 material points making up one quarter of the membrane, and the internal gas. The gas is represented by 16 material points initially in each computational element. The figure also shows normals to the membrane computed with and without connectivity. The membrane radius is initially 0.5, with a Young's modulus of 10^6 and Poisson's ratio of 0.3, density of 0.5, and thickness 0.1. The internal, gamma-law gas has $\gamma = 1.4$, an initial pressure of 1000, and density of 1.0. Viscosity ($\mu = 0.2$) is added to the gas in order to damp the motion and reach a steady-state during the simulation. The computational grid is a 10×10 mesh of square elements of length 0.1 on each side. A time step of 0.7×10^{-4} is used to satisfy the CFL condition based on the mesh size and elastic wave speed for this time-explicit, dynamic calculation. The units are dimensionless for the examples in this section.

On the left of Fig. 12, are the results of the simulation using using the local isosurface construction of this paper, and on the right, normals are computed using connectivity. From the initial state, the gas expands and pressurizes the membrane. After oscillating, the internal pressure is eventually balanced by the hoop stress in the membrane. The final configuration is in the lower half of the figure for both methods at a time of 0.05. Normal vectors flipped in direction from the majority is a reminder that the local isosurface construction does not distinguish between inward and outward pointing normals. Since the radius of curvature of the membrane is larger than the mesh size, normals are computed quite accurately. Fig. 13 compares the pressure as a function of time for the two methods. The pressure history is quite similar, especially for early times. There is a drift after a large number of timesteps.

A more stringent test is given in the next example. Again, an initially slack membrane is filled with a pressurized gas. However, the initial membrane shape as shown in Fig. 14 is more complicated than the last, circular example. The figure shows the initial dogbone shape, where quarter-symmetry is imposed in the computation. Each frame of the figure has the results using the local isosurface construction on the left and normals computed using connectivity on the right. In this case, 100 material points are used to discretize one quarter of the membrane on a background 10×10 mesh of size 0.2. The gas is represented by 64 material points per computational element. The gamma-law gas with $\gamma = 1.4$ has an initial pressure of 100, density of 1.0, and viscosity of 0.2. The membrane has a Young's



(a) (b)

(c) (d)

Figure 12: Gas-filled membrane expansion initially (top) and at equilibrium (bottom). On the left normals to the membrane are computed using the isosurface construction method and on the right normals are computed using connectivity.

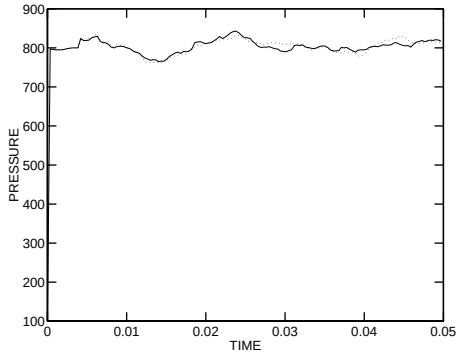


Figure 13: Comparison of the pressure as a function of time in the example of Fig. 12. The dotted line is the history using connectivity and the solid line is for the isosurface reconstruction method.

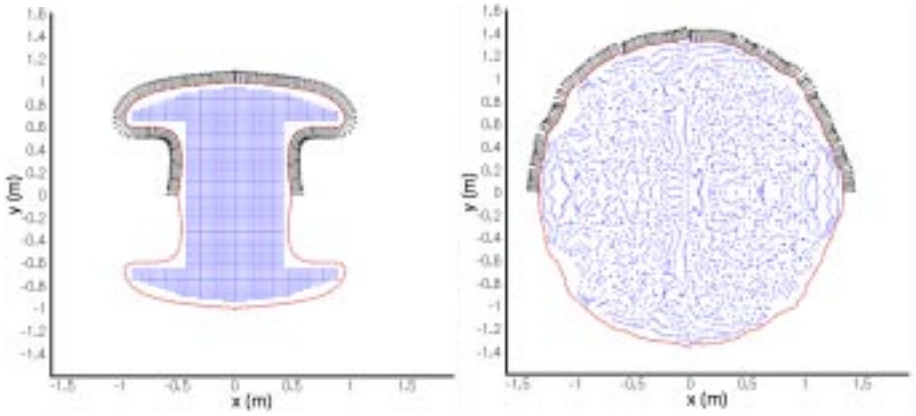


Figure 14: Gas-filled membrane expansion initially and at equilibrium. The left half of each frame has normals computed using the local isosurface construction and the right half uses connectivity.

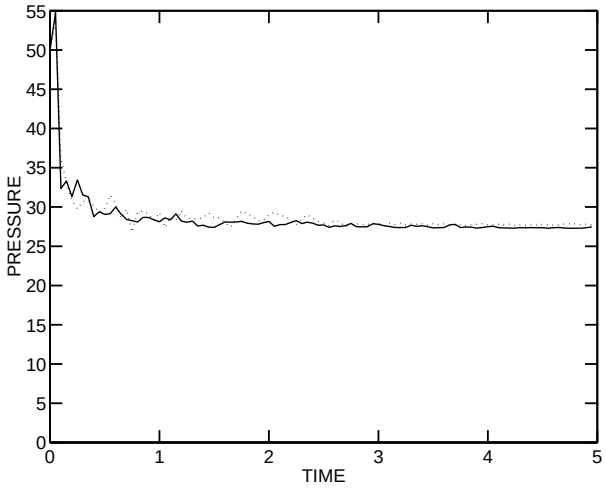


Figure 15: Comparison of the pressure as a function of time in the example of Fig. 14. The dotted line is the history using connectivity and the solid line uses the local isosurface construction.

modulus of 10^6, Poisson's ratio 0.3, density 0.5 and thickness 0.1. The time step is 0.7×10^{-4}. As before, the gas expands to fill the membrane, the gas-membrane system oscillates until viscous damping brings it to rest in an equilibrium configuration where the internal pressure is balanced by the hoop stress in the membrane. The final configuration is also shown in Fig. 14. Fig. 15 shows the pressure as a function of time for this problem, comparing the two methods for computing the normals. The equilibrium pressure is the same with both methods, but there is some variation in the time history.

This problem is challenging because there is a high curvature region that is not well resolved by the mesh at early stages of the deformation. Fig. 16 shows two early time steps in the calculation. Neither method does an entirely satisfactory job in computing the normals to this irregular surface, although connectivity gives better results. As we have seen in some of the test cases in the previous sections of this paper, endpoints of a membrane segment might not have a completely accurate normal and this is occasionally the case in these dynamic simulations for some time steps. Although not a problem in the test cases, we also find that the surface construction allows a function χ with the value 1 as the maximum. In this case, the gradient of χ is zero along the membrane and the normal is not defined. This observation accounts for other time steps where the normals in some element are not accurate. We have experimented with adding an extra random point assigned the value $\chi = 2$ so that a gradient is present, and 1 is not the maximum; however the overall quality of the simulation is not affected.

Initially, membrane points are placed along a smooth surface and Fig. 14 shows the calculation of the normals to that smooth surface is accurate. As the simulation proceeds, the membrane points can have small-scale, high-frequency fluctuations in position. If the surface reconstruction tries to follow this high-frequency noise, the normals will not be

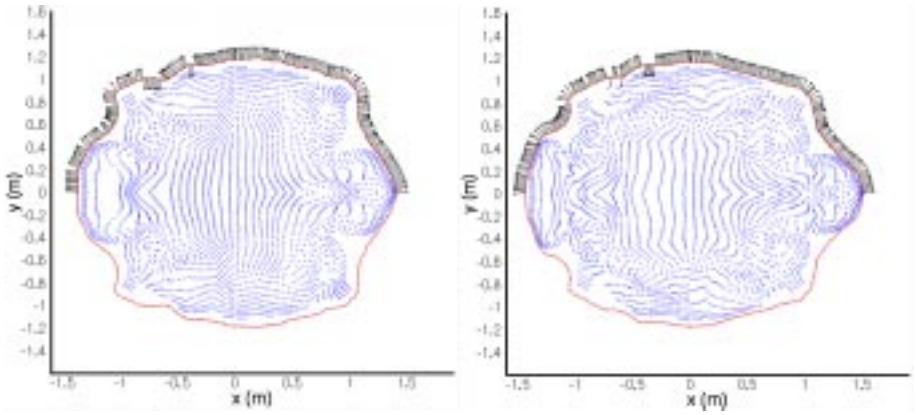


Figure 16: Gas-filled membrane expansion after 4 and 7 timesteps. The left half of each frame has normals computed using the local isosurface construction and the right half uses connectivity.

accurate. Although we have not tried it, some smoothing of the membrane-point positions prior to determining the function χ could be beneficial.

9 Conclusions

This paper introduces an algorithm for constructing surface normals for a surface represented by an unconnected set of points. The main idea is to construct, at least locally, a function that has the given surface as an isosurface. This construction is accomplished by solving a system of equations for nodal values off the surface so that interpolation from the nodes back to the surface gives the correct value. Gradients of this function provide the normal to the surface. Sections 3 and 5 utilize linear and higher-order b-splines for this purpose. Interpolation nodes are chosen on a logically rectangular mesh in two dimensions and on a hexadehral mesh in three dimensions. The examples in Sections 4 and 6 show that satisfactory results can be achieved if the surface curvature is not too large compared with the mesh spacing. It is also seen that the construction can be carried out locally so that only small systems of linear equations need to be solved for each element of the mesh containing membrane points.

The technique for computing surface normals has been combined with the material-point method, a continuum mechanics code. Numerical examples using the new methodology to simulate fluid-membrane interactions are presented in Section 8. A dynamic problem, where a pressurized membrane with complicated geometry is allowed to expand to its equilibrium shape, indicates the feasibility of the method for use in complex modeling and simulation. The treatment of the membrane interface and its interaction with fluid is simple in the MPM and avoids CPU-intensive work necessary in other methods. One major advantage is the ease of meshing bodies and interfaces, since points only have to be identified inside a body or on the membrane surface, and no connectivity is required among the points.

The reconstruction of surface geometry from disconnected points on the surface is potentially useful in many physical problems. This capability essentially combines advantageous features of surface tracking methods with those that capture the surface. Information about the surface location is known from points on the surface, but topology changes are numerically tractable since the points are not connected. Normals to the surface can be computed satisfactorily using quadratic shape functions. For other applications, like surface tension, that also require information about the curvature of the surface, higher-order shape functions are required which allow an additional smooth derivative of the normal. The simplicity of the method makes it feasible to study the complicated physics often associated with interfaces in continua.

References

[1] J.U. Brackbill, D.B. Kothe and C. Zemach, A continuum method for modeling surface tension, J. Comput. Phys., **100**, 335-354 (1992).

[2] S. Osher and J.A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, J. Comput. Phys., **79**, 12-49 (1988).

[3] J.A. Sethian and J. Strain, Crystal-grwoth and dendritic solidification, J. Comput. Phys., **98**, 231-253 (1992).

[4] M. Sussman, P. Smeerka and S. Osher, A level set approach for computing solutions to incompressible 2-phase flow, J. Comput. Phys., **114**, 146-159 (1994).

[5] D. Adalsteinsson and J. A. Sethian, A fast level set method for propagation of interfaces, J. Comput. Phys., **118**, 269-277 (1995).

[6] David Jasnow and Jorge Vinals, Coarse-grained description of thermo-capillary flow, Phys. Fluids, **8**, 660-669 (1995).

[7] J. A. Warren and W. J. Boettinger, Prediction of dendritic growth and microsegregation patterns in a binary alloy using the phase-field method, Acta metall. mater., **43**, 689-703 (1995).

[8] Danan Fan and Long-Qing Chen, Topological evolution during coupled grain growth and Ostwald ripening in volume-conserved 2-D two-phase polycrystals, Acta mater., **45**, 4145-4154 (1997).

[9] B. T. Nadiga and S. Zaleski, Investigations of a two-phase fluid model, Eur. J. Mech., B/Fluids, **15**, 885-896 (1996).

[10] J. Lowengrub and L. Truskinovsky, Quasi-incompressible Cahn-Hilliard fluids and topological transitions, Proc. Royal Soc. London **454**, 2617-2654 (1998).

[11] David Jacqmin, Phase-field numerics of two-phase Navier-Stokes flows, preprint.

[12] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, J. Comput. Phys., **100**, 25-37 (1992).

[13] F. Xabier Garaizar and John Trangenstein, Front tracking for shear bands in an antiplane shear model, J. Comput. Phys., **131**, 54-69 (1997).

[14] James Glimm, John W. Grove, Xiao Lin Li, Keh-ming Shyue, Yanni Zeng, Qiang Zhang, Three-dimensional front tracking, SIAM J. Sci. Comput., **19**, 703-727 (1998).

[15] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys., **25**, 220-252 (1977).

[16] C.S. Peskin and D. M. McQueen, A 3-dimensional computational method for blood-flow in the heart: 1. Immersed elastic fibers in a viscous incompressible fliud, J. Comput. Phys., **81**, 372-405 (1989).

[17] Randall J. LeVeque and Zhilin Li, Immersed interface methods for Stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput., **18**, 709-735 (1997).

[18] A.R. York, Ph.D. Thesis, Mechanical Engineering, UNM, 'Development of Modifications to the Material Point Method for the Simulation of Thin Membranes, Compressible Fluids, and Their Interactions' (1997). Also available as Sandia Report, SAND97-1893, July, 1997.

[19] D. Sulsky, Z. Chen and H. L. Schreyer, A particle method for history-dependent materials, Comput. Meths. Appl. Mech. Engrg., **118**, 179-196 (1994).

[20] D. Sulsky, S.-J. Zhou and H. L. Schreyer, Application of a particle-in-cell method to solid mechanics, Comput. Phys. Commun., **87**, 236-252 (1995).

[21] D. Sulsky and H. L. Schreyer, Axisymmetric form of the material point method with applications to upsetting and Taylor impact problems, Comput. Meths. Appl. Mech. Engrg., **139**, 409-429 (1996).

[22] J. U. Brackbill and H. M. Ruppel, FLIP: A method for adaptively zoned, particle-in-cell calculations in two dimensions, J. Comput. Phys., **65**, 314-343 (1986).

[23] J. U. Brackbill, D. B. Kothe and H. M. Ruppel, FLIP: A low-dissipation,particle-in-cell method for fluid flow, Comput. Phys. Comm., **48**, 25-38 (1998).

[24] A.R. York, D. Sulsky and H. Schreyer, The material point method for simulation of thin membranes, Int. J. Num. Meths. Engng., **44**, 1429-1456 (1999).

[25] A.R. York, D. Sulsky and H. Schreyer, Fluid-membrane interaction based on the material point method, Int. J. Num. Meths. Engrg., **48**, 901-924 (2000).

[26] S. Bardenhagen, J. Brackbill, and D. Sulsky, The material-point method for granular materials, Comp. Meths. Appld. Mechs. Engrng., **187**, 529-541 (2000).

[27] S. Bardenhagen, J. Brackbill, and D. Sulsky, A Numerical Study of Stress Distribution in Sheared Granular Material in Two Dimensions, Phys. Rev. E, to appear.

[28] Giovanni Lapenta and J. U. Brackbill, Control of the number of particles in fluid and MHD particle in cell methods, Comput. Phys. Commun., **87**, 139-154 (1995).

[29] C. DeBoor, A practical guide to splines, Springer-Verlag, New York, 1978.

[30] Gene H. Golub and Charles F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, 1989.