EN2210 Continuum Mechanics

Project

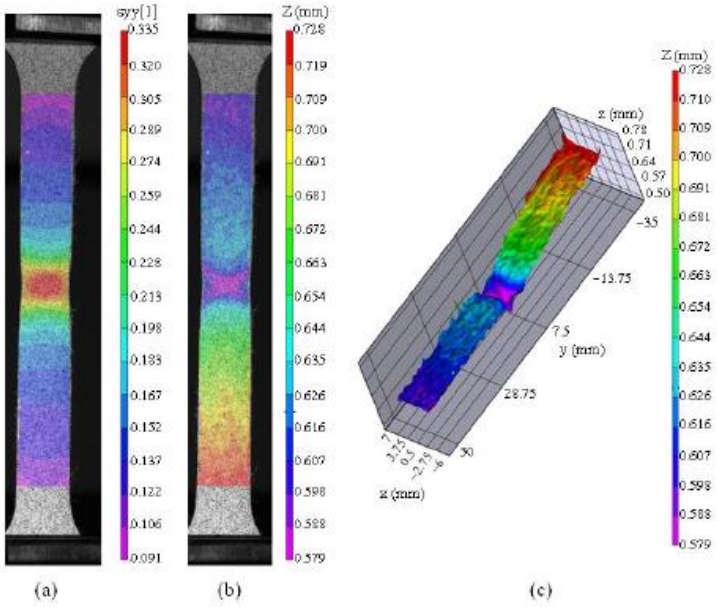
Computing Deformation Measures from Digital Image Correlation Data

Synopsis

You will write a MATLAB code to compute measures of deformation and motion from 3D digital image correlation measures.

1. Introduction

Digital image correlation is a powerful and comparatively inexpensive technique for measuring deformation fields in materials and structures with micron scale resolution. The basic principle is to take a series pictures of a speckle pattern on a surface during deformation, and to determine displacement fields by matching similar regions in two successive images. An example of displacement contours extracted from DIC measurements on the surface of a tensile specimen of steel [1] is shown in the figure.



In the example shown, only the surface of the specimen can be observed. In transparent materials, it is also possible to the full 3D displacement field in a specimen. Professor Frank's lab uses such a system to measure deformation in a soft gel substrate beneath a living cell [2]. If the constitutive response of the substrate is known, the tractions exerted by the cell can be determined. Professor Frank and his students have written software to extract strain fields from their DIC displacement measurements [3]. Your mission in this project is to write a similar MATLAB program.

Professor Franck's lab has kindly provided sample data sets to analyze. The data files contain three components of displacements measured at a set of points arranged in a regular *x,y,z* grid inside a gel specimen subjected to some external loading. In one case, the gel was subjected to uniaxial compression (as a test); in the other, a cell was placed on the surface of the gel. The cell adheres to the substrate and begins to move over its surface, and so exerts tractions on the gel surface. The DIC method was used to measure the displacements in the gel as a function of time.

Your goal will be to read these data, and then compute various deformation, strain, and strain rate measures in the gel, and display the results graphically. You can choose what measures you compute,

how you compute them, and how you will display the data, and can add whatever bells and whistles you like to your code – it could be just a basic MATLAB script, or a more fancy GUI driven code.

2. Project Deliverables:

Your mission is to provide:

1. A written report that describes what your code can do, with instructions for its use, and provides some examples of its performance. The examples could include a validation showing that the code gives correct results for a known strain field; and then some examples of fields
2. A MATLAB code that will read the data files provided and plot strain and strain rate measures of interest.

The due date for both is **Wednesday Oct 10.**

3. Data files

Two matlab files with representative datasets have been provided.

1. The file called "compression_data.mat" contains displacements measured in a specimen that was loaded externally to induce a roughly uniform state of strain in the material
2. The file called "Cell_data.mat" contains displacements measured beneath a cell placed on the surface of the gel.

Coordinates and displacements are all in microns.

Before running the program on these datasets, it is a good idea to test it with a displacement field that has a perfectly uniform strain field. You can use either grid in the datafile for this purpose, and simply replace the experimentally measured displacements with displacements corresponding to a known, uniform strain field, i.e.

$$u_i = E_{ij} x_j$$

with a predetermined value for E_{ij}. All the methods described below should be able to determine the strain tensor exactly.

4. Interpolating displacement fields

Your mission will be to compute, and plot, deformation measures in the specimen. You can choose what deformation measures you would like to compute – but you will most likely want to start by calculating the deformation gradient or displacement gradient.

To do this, you will have to interpolate the displacement data provided by Professor Franck's students. Again, you can choose how to do this. Two suggestions are given below. If you want to do something fancier, you could also think of using a meshless interpolation scheme, but these are quite cumbersome to implement and are recommended only for people who are finding their graduate programs too trivial….

4.1 Computing displacement gradients using finite-difference methods.

It is easiest to illustrate this approach in two dimensions. Consider a generic point in the grid $x_i^{(0)}$ in the picture. Suppose that we would like to determine the displacement gradient

$$G_{ij} = \frac{\partial u_i}{\partial x_j}$$

at the point of interest. To do so, we could assume that the displacement gradient is uniform in the immediate neighborhood of the point of interest. This means that we could compute the displacement at any nearby point from the formula

$$u_i = u_i^{(0)} + G_{ij}\left(x_j - x_j^{(0)}\right)$$

Of course, we don't know G_{ij} so we can't do this yet – but we can use this formula to determine the value of G_{ij} that matches this formula with the known values of displacement at neighboring grid points as accurately as possible. For example, we could select G_{ij} to minimize the least-squared error

$$E = \frac{1}{2}\sum_a \left(u_i^a - u_i^{(0)} - G_{ij}\left[x_j^a - x_j^{(0)}\right]\right)\left(u_i^a - u_i^{(0)} - G_{ik}\left[x_k^a - x_k^{(0)}\right]\right)$$

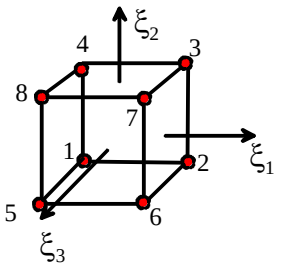
where the sum is taken over as many neighboring grid points as is convenient – for an interior point you could sum over the six (in 3D) nearest points; or if you wish, you can include a larger number of points, which will tend to smooth the data. How do we minimize this? Just use elementary calculus – set the derivatives with respect to all the components of G_{ij} to zero, with the result

$$G_{ij}\sum_a (x_j^a - x_j^{(0)})(x_k^a - x_k^{(0)}) = \sum_a (u_i^a - u_i^{(0)})(x_k^a - x_k^{(0)})$$

This is a system of 9 linear equations for the 9 unknown components of G_{ij}. Assembling the system of equations is a bit tedious, but can be done with four nested loops over *i,j,k,a*. Ask if you don't see how to do this!

4.2 Computing displacement gradients using finite-element interpolation

The displacement fields can also be interpolated using finite element shape functions. In this approach, sets of eight grid points are taken to lie at the corner of a linear hexahedral element, as shown in the figure. The standard finite element procedure is then used to interpolate values of displacement at the corners of this element. We begin by introducing interpolation functions $N^a(\xi_i)$ for each element in terms of a local, dimensionless, coordinate system within the element. The coordinates satisfy $-1 \le \xi_i \le +1$, and the interpolation functions are



$$N^1 = (1-\xi_1)(1-\xi_2)(1-\xi_3)/8 \quad N^2 = (1+\xi_1)(1-\xi_2)(1-\xi_3)/8$$
$$N^3 = (1+\xi_1)(1+\xi_2)(1-\xi_3)/8 \quad N^4 = (1-\xi_1)(1+\xi_2)(1-\xi_3)/8$$
$$N^5 = (1-\xi_1)(1-\xi_2)(1+\xi_3)/8 \quad N^6 = (1+\xi_1)(1-\xi_2)(1+\xi_3)/8$$
$$N^7 = (1+\xi_1)(1+\xi_2)(1+\xi_3)/8 \quad N^8 = (1-\xi_1)(1+\xi_2)(1+\xi_3)/8$$

To save you some typing and tedious algebra, MATLAB scripts that compute the shape functions and their derivatives are provided in the Appendix. The displacement field and the position of a point inside an element are computed in terms of the interpolation functions using the formulas

$$u_i = \sum_{a=1}^{N_e} N^a(\xi_j)u_i^a \qquad x_i = \sum_{a=1}^{N_e} N^a(\xi_j)x_i^a$$

where $N^a(\xi_j)$ denote the shape functions, u_i^a, x_i^a denote the displacement values and coordinates of the nodes on the element, and N_e is the number of nodes on the element.

The displacement gradient at an arbitrary point within the element can then be computed from

$$G_{ij} = \sum_{a=1}^{N_e} \frac{\partial N^a}{\partial x_j} u_i^a$$

where the shape function derivatives can be computed using the formula

$$\frac{\partial N^a}{\partial x_j} = \frac{\partial N^a}{\partial \xi_i} \frac{\partial \xi_i}{\partial x_j}$$

where

$$\frac{\partial \xi_j}{\partial x_i} = \left(\frac{\partial x}{\partial \xi} \right)^{-1}_{ji} \qquad \frac{\partial x_i}{\partial \xi_j} = \frac{\partial}{\partial \xi_j} \left(\sum_{a=1}^{N_e} N^a(\xi) x_i^a \right) = \sum_{a=1}^{N_e} \frac{\partial N^a(\xi)}{\partial \xi_j} x_i^a$$

It is simplest to use this procedure to compute the displacement gradient at the center of each element. It is possible, but not really necessary, to determine values at the original grid points as well – one procedure would be to average the displacement gradients for all the elements connected to each grid point, or you could do more fancy things like do a least-squares fit of the values at the grid points to interior values.

5. Data processing and plotting

Once you have computed displacement gradients, it is straightforward to compute other quantities that characterize deformations. Examples might include

- The Jacobian J = det(**F**)
- Left and right stretch tensors, and/or their principal values and directions; strain invariants
- The rotation tensor, and its axis/angle (Rodriguez representation)
- The velocity gradient; the stretch rate; the spin tensor, the vorticity vector (or its magnitude and direction)

You can plot the data using the matlab contour() function; or you can plot contour data on slices through the specimen using the matlab slice() function. To get you started, a sample MATLAB script is provided in the appendix, which reads the experimental data set, and plots contours of the displacement components using slice(). Matlab is not ideal for visualizing 3D meshes and data, unfortunately - If you need to plot complicated data on 3D grids Tecplot is far better.

6. References

1. Jason Coryell, Josh Campbell, Vesna Savic, John Bradley, Sushil Mishra, Shashank Tiwari, Louis Hector, Jr. "Tensile deformation of quenched and partitioned steel - a third generation high strength steel," Supplemental Proceedings, Vol **2**, Materials Properties, Characterization and Modeling, 555-562 2012

2. Franck, C., Hong, S., Maskarinec, S.A., Tirrell, D.A., Ravichandran, G., "Three-Dimensional Full-Field Measurements of Large Deformations in Soft Materials using Confocal Microscopy and Digital Volume Correlation," Experimental Mechanics, **47**, 427-438, 2007.

3. Jennet Toyjanova, Eyal Bar-Kochva, Christian Frank, personal communication, 2012.

Appendix 1: Grading Rubric

1. Report: 10 points.
    - Report describes all code capabilities; convincing demonstration of accuracy; clear and thorough instructions for use with appropriate examples; well organized; clearly written in good English; graphs properly labeled, etc (10 points)
    - Good report but with minor errors 8 points
    - Adequate report, but difficult to follow; incomplete 6 points
    - Major errors or omissions 4 points
2. Matlab Code 15 points
    - Correct, well commented, user-friendly code with extensive capabilities 15 points
    - Good, correct, well commented and user friendly MATLAB code but capabilities not as extensive as other submissions 12 points
    - Functioning code, but difficult to read or use, or limited capabilities for data processing or visualization 9
    - Major errors, or only minimal capabilities 6

Appendix 2: Basic Matlab Script

```
function displacement_processing

close all

load('compression_data.mat') % Read the compression data file
% load('Cell_data.mat') % Read the cell data file
%
% Each data file contains:
% nx, ny, nz, nt = no. grid points in x,y,z, directions and # time steps
% xyz(1:3,nn) are xyz coords of grid point number nn.
% uvw(1:3,nn,i) are the xyz components of displacemnt for point nn
%                at time step i
% The order of the grid points is such that
%      xyz(i,1:nx) is the first row of points along the x direction,
%                  with y=z=0
%      xyz(i,nx+1:2*nx) is the second row of the top plane with z=0
%      xyz(i,nx*ny+1) is the start of the second plane with z=dz
%
% Create figure
    figure1 = figure;

for t = 1:nt % Loop over time steps
   for k = 1:nz
     for j = 1:ny
        for i = 1:nx
           nn = i+(j-1)*nx + (k-1)*ny*nx;
           x(i,j,k) = xyz(1,nn);
           y(i,j,k) = xyz(2,nn);
           z(i,j,k) = xyz(3,nn);
           u(i,j,k) = uvw(1,nn,t);  % This plots x component of displ.
        end
     end
   end
```

```
%
% MATLAB slice expects the order of coords to be indexed
% with y varying most rapidly, then x, then z.
%
    P = [2 1 3];
    X = permute(x, P);
    Y = permute(y, P);
    Z = permute(z, P);
    V = permute(u, P);

    clf % Clear the figure
    axes1 = axes('Parent',figure1,...
    'Position',[0.13 0.14047619047619 0.648571428571429 0.78452380952381],...
    'FontSize',14,...
    'CLim',[-2.64384412765503 1.82826542854309]);
    view(axes1,[-37.5 30]);
    grid(axes1,'on');
    hold(axes1,'all');
    slice(X,Y,Z,V,xyz(2,end)/2,xyz(1,end)/2,xyz(3,end)/2)
    % Create xlabel
    xlabel('x (microns)','FontSize',14);
    % Create ylabel
    ylabel({'y (microns)'},'FontSize',14);
    % Create zlabel
    zlabel({'z (microns)'},'FontSize',14);
    % Create colorbar
    colorbar('peer',axes1,...
    [0.864177927927928 0.127272727272727 0.0225225225225226
0.690078870900789],...
    'FontSize',14);
   % Create textbox
    annotation(figure1,'textbox',...
    [0.783494208494208 0.89821722113503 0.195945945945948
0.0639269406392694],...
    'String',{'Displacements (microns)'},...
    'HorizontalAlignment','center',...
    'FontSize',14,...
    'FitBoxToText','off',...
    'LineStyle','none');

    pause(0.1)
end

end

function N = shapefunctions(nelnodes,ncoord,elident,xi)


   N = zeros(nelnodes,1);
%
%  1D elements
%
  if (ncoord == 1)
    if (nelnodes==2)
      N(1) = 0.5*(1.+xi(1));
      N(2) = 0.5*(1.-xi(1));
```

```
    elseif (nelnodes == 3)
      N(1) = -0.5*xi(1)*(1.-xi(1));
      N(2) =  0.5*xi(1)*(1.+xi(1));
      N(3) = (1.-xi(1))*(1.+xi(1));
    end
%
%  2D elements
%
  elseif (ncoord == 2)
%
%    Triangular element
%
    if ( nelnodes == 3 )
      N(1) = xi(1);
      N(2) = xi(2);
      N(3) = 1.-xi(1)-xi(2);
    elseif ( nelnodes == 6 )
      xi3 = 1.-xi(1)-xi(2);
      N(1) = (2.*xi(1)-1.)*xi(1);
      N(2) = (2.*xi(2)-1.)*xi(2);
      N(3) = (2.*xi3-1.)*xi3;
      N(4) = 4.*xi(1)*xi(2);
      N(5) = 4.*xi(2)*xi3;
      N(6) = 4.*xi3*xi(1);
%
%    Rectangular element
%
    elseif ( nelnodes == 4 )
      N(1) = 0.25*(1.-xi(1))*(1.-xi(2));
      N(2) = 0.25*(1.+xi(1))*(1.-xi(2));
      N(3) = 0.25*(1.+xi(1))*(1.+xi(2));
      N(4) = 0.25*(1.-xi(1))*(1.+xi(2));
    elseif (nelnodes == 8)
      N(1) = -0.25*(1.-xi(1))*(1.-xi(2))*(1.+xi(1)+xi(2));
      N(2) = 0.25*(1.+xi(1))*(1.-xi(2))*(xi(1)-xi(2)-1.);
      N(3) = 0.25*(1.+xi(1))*(1.+xi(2))*(xi(1)+xi(2)-1.);
      N(4) = 0.25*(1.-xi(1))*(1.+xi(2))*(xi(2)-xi(1)-1.);
      N(5) = 0.5*(1.-xi(1)*xi(1))*(1.-xi(2));
      N(6) = 0.5*(1.+xi(1))*(1.-xi(2)*xi(2));
      N(7) = 0.5*(1.-xi(1)*xi(1))*(1.+xi(2));
      N(8) = 0.5*(1.-xi(1))*(1.-xi(2)*xi(2));
    end
%
  elseif (ncoord==3)

    if (nelnodes == 4)
      N(1) = xi(1);
      N(2) = xi(2);
      N(3) = xi(3);
      N(4) = 1.-xi(1)-xi(2)-xi(3);
    elseif (nelnodes == 10)
      xi4 = 1.-xi(1)-xi(2)-xi(3);
      N(1) = (2.*xi(1)-1.)*xi(1);
      N(2) = (2.*xi(2)-1.)*xi(2);
      N(3) = (2.*xi(3)-1.)*xi(3);
      N(4) = (2.*xi4-1.)*xi4;
      N(5) = 4.*xi(1)*xi(2);
```

```
      N(6) = 4.*xi(2)*xi(3);
      N(7) = 4.*xi(3)*xi(1);
      N(8) = 4.*xi(1)*xi4;
      N(9) = 4.*xi(2)*xi4;
      N(10) = 4.*xi(3)*xi4;
    elseif (nelnodes == 8)
      N(1) = (1.-xi(1))*(1.-xi(2))*(1.-xi(3))/8.;
      N(2) = (1.+xi(1))*(1.-xi(2))*(1.-xi(3))/8.;
      N(3) = (1.+xi(1))*(1.+xi(2))*(1.-xi(3))/8.;
      N(4) = (1.-xi(1))*(1.+xi(2))*(1.-xi(3))/8.;
      N(5) = (1.-xi(1))*(1.-xi(2))*(1.+xi(3))/8.;
      N(6) = (1.+xi(1))*(1.-xi(2))*(1.+xi(3))/8.;
      N(7) = (1.+xi(1))*(1.+xi(2))*(1.+xi(3))/8.;
      N(8) = (1.-xi(1))*(1.+xi(2))*(1.+xi(3))/8.;
    elseif (nelnodes == 20)
      N(1) = (1.-xi(1))*(1.-xi(2))*(1.-xi(3))*(-xi(1)-xi(2)-xi(3)-2.)/8.;
      N(2) = (1.+xi(1))*(1.-xi(2))*(1.-xi(3))*(xi(1)-xi(2)-xi(3)-2.)/8.;
      N(3) = (1.+xi(1))*(1.+xi(2))*(1.-xi(3))*(xi(1)+xi(2)-xi(3)-2.)/8.;
      N(4) = (1.-xi(1))*(1.+xi(2))*(1.-xi(3))*(-xi(1)+xi(2)-xi(3)-2.)/8.;
      N(5) = (1.-xi(1))*(1.-xi(2))*(1.+xi(3))*(-xi(1)-xi(2)+xi(3)-2.)/8.;
      N(6) = (1.+xi(1))*(1.-xi(2))*(1.+xi(3))*(xi(1)-xi(2)+xi(3)-2.)/8.;
      N(7) = (1.+xi(1))*(1.+xi(2))*(1.+xi(3))*(xi(1)+xi(2)+xi(3)-2.)/8.;
      N(8) = (1.-xi(1))*(1.+xi(2))*(1.+xi(3))*(-xi(1)+xi(2)+xi(3)-2.)/8.;
      N(9)  = (1.-xi(1)^2)*(1.-xi(2))*(1.-xi(3))/4.;
      N(10) = (1.+xi(1))*(1.-xi(2)^2)*(1.-xi(3))/4.;
      N(11) = (1.-xi(1)^2)*(1.+xi(2))*(1.-xi(3))/4.;
      N(12) = (1.-xi(1))*(1.-xi(2)^2)*(1.-xi(3))/4.;
      N(13) = (1.-xi(1)^2)*(1.-xi(2))*(1.+xi(3))/4.;
      N(14) = (1.+xi(1))*(1.-xi(2)^2)*(1.+xi(3))/4.;
      N(15) = (1.-xi(1)^2)*(1.+xi(2))*(1.+xi(3))/4.;
      N(16) = (1.-xi(1))*(1.-xi(2)^2)*(1.+xi(3))/4.;
      N(17) = (1.-xi(1))*(1.-xi(2))*(1.-xi(3)^2)/4.;
      N(18) = (1.+xi(1))*(1.-xi(2))*(1.-xi(3)^2)/4.;
      N(19) = (1.+xi(1))*(1.+xi(2))*(1.-xi(3)^2)/4.;
      N(20) = (1.-xi(1))*(1.+xi(2))*(1.-xi(3)^2)/4.;
    end
  end

end

%
%================= SHAPE FUNCTION DERIVATIVES ======================
%
function dNdxi = shapefunctionderivs(nelnodes,ncoord,elident,xi)

  dNdxi = zeros(nelnodes,ncoord);
%
% 1D elements
%
  if (ncoord == 1)
    if (nelnodes==2)
      dNdxi(1,1) = 0.5;
      dNdxi(2,1) = -0.5;
    elseif (nelnodes == 3)
      dNdxi(1,1) = -0.5+xi(1);
      dNdxi(2,1) =  0.5+xi(1);
```

```
      dNdxi(3,1) = -2.*xi(1);
    end
%
%  2D elements
%
  elseif (ncoord == 2)
%
%    Triangular element
%
    if ( nelnodes == 3 )
      dNdxi(1,1) = 1.;
      dNdxi(2,2) = 1.;
      dNdxi(3,1) = -1.;
      dNdxi(3,2) = -1.;
    elseif ( nelnodes == 6 )
      xi3 = 1.-xi(1)-xi(2);
      dNdxi(1,1) = 4.*xi(1)-1.;
      dNdxi(2,2) = 4.*xi(2)-1.;
      dNdxi(3,1) = -(4.*xi3-1.);
      dNdxi(3,2) = -(4.*xi3-1.);
      dNdxi(4,1) = 4.*xi(2);
      dNdxi(4,2) = 4.*xi(1);
      dNdxi(5,1) = -4.*xi(2);
      dNdxi(5,2) = -4.*xi(1);
      dNdxi(6,1) = 4.*xi3 - 4.*xi(1);
      dNdxi(6,2) = 4.*xi3 - 4.*xi(2);
%
%    Rectangular element
%
    elseif ( nelnodes == 4 )
      dNdxi(1,1) = -0.25*(1.-xi(2));
      dNdxi(1,2) = -0.25*(1.-xi(1));
      dNdxi(2,1) = 0.25*(1.-xi(2));
      dNdxi(2,2) = -0.25*(1.+xi(1));
      dNdxi(3,1) = 0.25*(1.+xi(2));
      dNdxi(3,2) = 0.25*(1.+xi(1));
      dNdxi(4,1) = -0.25*(1.+xi(2));
      dNdxi(4,2) = 0.25*(1.-xi(1));
    elseif (nelnodes == 8)
      dNdxi(1,1) = 0.25*(1.-xi(2))*(2.*xi(1)+xi(2));
      dNdxi(1,2) = 0.25*(1.-xi(1))*(xi(1)+2.*xi(2));
      dNdxi(2,1) = 0.25*(1.-xi(2))*(2.*xi(1)-xi(2));
      dNdxi(2,2) = 0.25*(1.+xi(1))*(2.*xi(2)-xi(1));
      dNdxi(3,1) = 0.25*(1.+xi(2))*(2.*xi(1)+xi(2));
      dNdxi(3,2) = 0.25*(1.+xi(1))*(2.*xi(2)+xi(1));
      dNdxi(4,1) = 0.25*(1.+xi(2))*(2.*xi(1)-xi(2));
      dNdxi(4,2) = 0.25*(1.-xi(1))*(2.*xi(2)-xi(1));
      dNdxi(5,1) = -xi(1)*(1.-xi(2));
      dNdxi(5,2) = -0.5*(1.-xi(1)*xi(1));
      dNdxi(6,1) = 0.5*(1.-xi(2)*xi(2));
      dNdxi(6,2) = -(1.+xi(1))*xi(2);
      dNdxi(7,1) = -xi(1)*(1.+xi(2));
      dNdxi(7,2) = 0.5*(1.-xi(1)*xi(1));
      dNdxi(8,1) = -0.5*(1.-xi(2)*xi(2));
      dNdxi(8,2) = -(1.-xi(1))*xi(2);
     end
%
```

```
%     3D elements
%
   elseif (ncoord==3)

     if (nelnodes == 4)
       dNdxi(1,1) = 1.;
       dNdxi(2,2) = 1.;
       dNdxi(3,3) = 1.;
       dNdxi(4,1) = -1.;
       dNdxi(4,2) = -1.;
       dNdxi(4,3) = -1.;
     elseif (nelnodes == 10)
       xi4 = 1.-xi(1)-xi(2)-xi(3);
       dNdxi(1,1) = (4.*xi(1)-1.);
       dNdxi(2,2) = (4.*xi(2)-1.);
       dNdxi(3,3) = (4.*xi(3)-1.);
       dNdxi(4,1) = -(4.*xi4-1.);
       dNdxi(4,2) = -(4.*xi4-1.);
       dNdxi(4,3) = -(4.*xi4-1.);
       dNdxi(5,1) = 4.*xi(2);
       dNdxi(5,2) = 4.*xi(1);
       dNdxi(6,2) = 4.*xi(3);
       dNdxi(6,3) = 4.*xi(2);
       dNdxi(7,1) = 4.*xi(3);
       dNdxi(7,3) = 4.*xi(1);
       dNdxi(8,1) = 4.*(xi4-xi(1));
       dNdxi(8,2) = -4.*xi(1);
       dNdxi(8,3) = -4.*xi(1);
       dNdxi(9,1) = -4.*xi(2);
       dNdxi(9,2) = 4.*(xi4-xi(2));
       dNdxi(9,3) = -4.*xi(2);
       dNdxi(10,1) = -4.*xi(3)*xi4;
       dNdxi(10,2) = -4.*xi(3);
       dNdxi(10,3) = 4.*(xi4-xi(3));
     elseif (nelnodes == 8)
       dNdxi(1,1) = -(1.-xi(2))*(1.-xi(3))/8.;
       dNdxi(1,2) = -(1.-xi(1))*(1.-xi(3))/8.;
       dNdxi(1,3) = -(1.-xi(1))*(1.-xi(2))/8.;
       dNdxi(2,1) = (1.-xi(2))*(1.-xi(3))/8.;
       dNdxi(2,2) = -(1.+xi(1))*(1.-xi(3))/8.;
       dNdxi(2,3) = -(1.+xi(1))*(1.-xi(2))/8.;
       dNdxi(3,1) = (1.+xi(2))*(1.-xi(3))/8.;
       dNdxi(3,2) = (1.+xi(1))*(1.-xi(3))/8.;
       dNdxi(3,3) = -(1.+xi(1))*(1.+xi(2))/8.;
       dNdxi(4,1) = -(1.+xi(2))*(1.-xi(3))/8.;
       dNdxi(4,2) = (1.-xi(1))*(1.-xi(3))/8.;
       dNdxi(4,3) = -(1.-xi(1))*(1.+xi(2))/8.;
       dNdxi(5,1) = -(1.-xi(2))*(1.+xi(3))/8.;
       dNdxi(5,2) = -(1.-xi(1))*(1.+xi(3))/8.;
       dNdxi(5,3) = (1.-xi(1))*(1.-xi(2))/8.;
       dNdxi(6,1) = (1.-xi(2))*(1.+xi(3))/8.;
       dNdxi(6,2) = -(1.+xi(1))*(1.+xi(3))/8.;
       dNdxi(6,3) = (1.+xi(1))*(1.-xi(2))/8.;
       dNdxi(7,1) = (1.+xi(2))*(1.+xi(3))/8.;
       dNdxi(7,2) = (1.+xi(1))*(1.+xi(3))/8.;
       dNdxi(7,3) = (1.+xi(1))*(1.+xi(2))/8.;
       dNdxi(8,1) = -(1.+xi(2))*(1.+xi(3))/8.;
```

```
        dNdxi(8,2) = (1.-xi(1))*(1.+xi(3))/8.;
        dNdxi(8,3) = (1.-xi(1))*(1.+xi(2))/8.;
      elseif (nelnodes == 20)
        dNdxi(1,1) = (-(1.-xi(2))*(1.-xi(3))*(-xi(1)-xi(2)-xi(3)-2.)-(1.-
xi(1))*(1.-xi(2))*(1.-xi(3)))/8.;
        dNdxi(1,2) = (-(1.-xi(1))*(1.-xi(3))*(-xi(1)-xi(2)-xi(3)-2.)-(1.-
xi(1))*(1.-xi(2))*(1.-xi(3)))/8.;
        dNdxi(1,3) = (-(1.-xi(1))*(1.-xi(2))*(-xi(1)-xi(2)-xi(3)-2.)-(1.-
xi(1))*(1.-xi(2))*(1.-xi(3)))/8.;

        dNdxi(2,1) = ((1.-xi(2))*(1.-xi(3))*(xi(1)-xi(2)-xi(3)-
2.)+(1.+xi(1))*(1.-xi(2))*(1.-xi(3)))/8.;
        dNdxi(2,2) = (-(1.+xi(1))*(1.-xi(3))*(xi(1)-xi(2)-xi(3)-2.)-
(1.+xi(1))*(1.-xi(2))*(1.-xi(3)))/8.;
        dNdxi(2,3) = (-(1.+xi(1))*(1.-xi(2))*(xi(1)-xi(2)-xi(3)-2.)-
(1.+xi(1))*(1.-xi(2))*(1.-xi(3)))/8.;

        dNdxi(3,1) = ((1.+xi(2))*(1.-xi(3))*(xi(1)+xi(2)-xi(3)-
2.)+(1.+xi(1))*(1.+xi(2))*(1.-xi(3)))/8.;
        dNdxi(3,2) = ((1.+xi(1))*(1.-xi(3))*(xi(1)+xi(2)-xi(3)-
2.)+(1.+xi(1))*(1.+xi(2))*(1.-xi(3)))/8.;
        dNdxi(3,3) = (-(1.+xi(1))*(1.+xi(2))*(xi(1)+xi(2)-xi(3)-2.)-
(1.+xi(1))*(1.+xi(2))*(1.-xi(3)))/8.;

        dNdxi(4,1) = (-(1.+xi(2))*(1.-xi(3))*(-xi(1)+xi(2)-xi(3)-2.)-(1.-
xi(1))*(1.+xi(2))*(1.-xi(3)))/8.;
        dNdxi(4,2) = ((1.-xi(1))*(1.-xi(3))*(-xi(1)+xi(2)-xi(3)-2.)+(1.-
xi(1))*(1.+xi(2))*(1.-xi(3)))/8.;
        dNdxi(4,3) = (-(1.-xi(1))*(1.+xi(2))*(-xi(1)+xi(2)-xi(3)-2.)-(1.-
xi(1))*(1.+xi(2))*(1.-xi(3)))/8.;
        dNdxi(5,1) = (-(1.-xi(2))*(1.+xi(3))*(-xi(1)-xi(2)+xi(3)-2.)-(1.-
xi(1))*(1.-xi(2))*(1.+xi(3)))/8.;
        dNdxi(5,2) = (-(1.-xi(1))*(1.+xi(3))*(-xi(1)-xi(2)+xi(3)-2.)-(1.-
xi(1))*(1.-xi(2))*(1.+xi(3)))/8.;
        dNdxi(5,3) = ((1.-xi(1))*(1.-xi(2))*(-xi(1)-xi(2)+xi(3)-2.)+(1.-
xi(1))*(1.-xi(2))*(1.+xi(3)))/8.;
        dNdxi(6,1) = ((1.-xi(2))*(1.+xi(3))*(xi(1)-xi(2)+xi(3)-
2.)+(1.+xi(1))*(1.-xi(2))*(1.+xi(3)))/8.;
        dNdxi(6,2) = (-(1.+xi(1))*(1.+xi(3))*(xi(1)-xi(2)+xi(3)-2.)-
(1.+xi(1))*(1.-xi(2))*(1.+xi(3)))/8.;
        dNdxi(6,3) = ((1.+xi(1))*(1.-xi(2))*(xi(1)-xi(2)+xi(3)-
2.)+(1.+xi(1))*(1.-xi(2))*(1.+xi(3)))/8.;
        dNdxi(7,1) = ((1.+xi(2))*(1.+xi(3))*(xi(1)+xi(2)+xi(3)-
2.)+(1.+xi(1))*(1.+xi(2))*(1.+xi(3)))/8.;
        dNdxi(7,2) = ((1.+xi(1))*(1.+xi(3))*(xi(1)+xi(2)+xi(3)-
2.)+(1.+xi(1))*(1.+xi(2))*(1.+xi(3)))/8.;
        dNdxi(7,3) = ((1.+xi(1))*(1.+xi(2))*(xi(1)+xi(2)+xi(3)-
2.)+(1.+xi(1))*(1.+xi(2))*(1.+xi(3)))/8.;
        dNdxi(8,1) = (-(1.+xi(2))*(1.+xi(3))*(-xi(1)+xi(2)+xi(3)-2.)-(1.-
xi(1))*(1.+xi(2))*(1.+xi(3)))/8.;
        dNdxi(8,2) = ((1.-xi(1))*(1.+xi(3))*(-xi(1)+xi(2)+xi(3)-2.)+(1.-
xi(1))*(1.+xi(2))*(1.+xi(3)))/8.;
        dNdxi(8,3) = ((1.-xi(1))*(1.+xi(2))*(-xi(1)+xi(2)+xi(3)-2.)+(1.-
xi(1))*(1.+xi(2))*(1.+xi(3)))/8.;
        dNdxi(9,1)  = -2.*xi(1)*(1.-xi(2))*(1.-xi(3))/4.;
        dNdxi(9,2)  = -(1.-xi(1)^2)*(1.-xi(3))/4.;
```

```
      dNdxi(9,3)  = -(1.-xi(1)^2)*(1.-xi(2))/4.;
      dNdxi(10,1)  = (1.-xi(2)^2)*(1.-xi(3))/4.;
      dNdxi(10,2)  = -2.*xi(2)*(1.+xi(1))*(1.-xi(3))/4.;
      dNdxi(10,3)  = -(1.-xi(2)^2)*(1.+xi(1))/4.;
      dNdxi(11,1)  = -2.*xi(1)*(1.+xi(2))*(1.-xi(3))/4.;
      dNdxi(11,2)  = (1.-xi(1)^2)*(1.-xi(3))/4.;
      dNdxi(11,3)  = -(1.-xi(1)^2)*(1.+xi(2))/4.;
      dNdxi(12,1)  = -(1.-xi(2)^2)*(1.-xi(3))/4.;
      dNdxi(12,2)  = -2.*xi(2)*(1.-xi(1))*(1.-xi(3))/4.;
      dNdxi(12,3)  = -(1.-xi(2)^2)*(1.-xi(1))/4.;
      dNdxi(13,1)  = -2.*xi(1)*(1.-xi(2))*(1.+xi(3))/4.;
      dNdxi(13,2)  = -(1.-xi(1)^2)*(1.+xi(3))/4.;
      dNdxi(13,3)  = (1.-xi(1)^2)*(1.-xi(2))/4.;
      dNdxi(14,1)  = (1.-xi(2)^2)*(1.+xi(3))/4.;
      dNdxi(14,2)  = -2.*xi(2)*(1.+xi(1))*(1.+xi(3))/4.;
      dNdxi(14,3)  = (1.-xi(2)^2)*(1.+xi(1))/4.;
      dNdxi(15,1)  = -2.*xi(1)*(1.+xi(2))*(1.+xi(3))/4.;
      dNdxi(15,2)  =  (1.-xi(1)^2)*(1.+xi(3))/4.;
      dNdxi(15,3)  = (1.-xi(1)^2)*(1.+xi(2))/4.;
      dNdxi(16,1)  = -(1.-xi(2)^2)*(1.+xi(3))/4.;
      dNdxi(16,2)  = -2.*xi(2)*(1.-xi(1))*(1.+xi(3))/4.;
      dNdxi(16,3)  = (1.-xi(2)^2)*(1.-xi(1))/4.;
      dNdxi(17,1) = -(1.-xi(2))*(1.-xi(3)^2)/4.;
      dNdxi(17,2) = -(1.-xi(1))*(1.-xi(3)^2)/4.;
      dNdxi(17,3) = -xi(3)*(1.-xi(1))*(1.-xi(2))/2.;
      dNdxi(18,1) = (1.-xi(2))*(1.-xi(3)^2)/4.;
      dNdxi(18,2) = -(1.+xi(1))*(1.-xi(3)^2)/4.;
      dNdxi(18,3) = -xi(3)*(1.+xi(1))*(1.-xi(2))/2.;
      dNdxi(19,1) = (1.+xi(2))*(1.-xi(3)^2)/4.;
      dNdxi(19,2) = (1.+xi(1))*(1.-xi(3)^2)/4.;
      dNdxi(19,3) = -xi(3)*(1.+xi(1))*(1.+xi(2))/2.;
      dNdxi(20,1) = -(1.+xi(2))*(1.-xi(3)^2)/4.;
      dNdxi(20,2) = (1.-xi(1))*(1.-xi(3)^2)/4.;
      dNdxi(20,3) = -xi(3)*(1.-xi(1))*(1.+xi(2))/2.;
    end
  end

end
```