Lecture No. 13

1-D Numerical Quadrature

- Element matrix and vector contributions can be computed using numerical quadrature.
- In numerical quadrature, an integral is approximated by a linear combination of integrand values, such that the quadrature is exact for polynomial integrands up to a degree determined by the order of the quadrature formula.
- There are two main classes of formulae:

  Newton-Cotes formulae: generally less efficient but simple integrands at equispaced points.

  Gaussian Quadrature: sampling points are not equispaced: fewer evaluations are required than Newton-Cotes.

$$I = \int_{-1}^{+1} f(\xi) d\xi = \sum_{i=1}^{n} w_i f(\xi_i) + E_n$$

where

w_i = weighting factor and ξ_i = coordinate of the ith integration point

n = total number of integration points

$$E_n = \text{error} = 0\left(\frac{d^{2n}f}{d\xi^{2n}}\right)$$

n	exact for	i	ξ_i	w_i
1	linear	1	0	2
2	cubic	1	$+1/\sqrt{3}$	+1
		2	$-1/\sqrt{3}$	+1
3	quintic	1	0	8/9
		2	$+\sqrt{15/5}$	5/9
		3	$-\sqrt{15/5}$	5/9
4	septimal	1,2	± 0.86113631	0.34785485
		3,4	± 0.33998104	0.65214515

- The Gaussian quadrature formulae are exact for polynomials for degree 2n-1. The integration points and weights were derived on this premise.
- The formulae will also be exact for any integer degree polynomial below this order.
- Gaussian Quadrature also has slightly superior error coefficients as compared to Newton-Cotes.

Motivation for Numerical Quadrature

1. Flexibility in choosing interpolating functions at the execution time of the program. Thus we can implement the program such that any order interpolation may be specified at the time of execution. The quadrature formulae are then used to evaluate matrices.
2. We can integrate functions which are difficult to or can not be integrated in closed form:

$$c_f \frac{1}{(\eta + h)} (u^2 + v^2)^{1/2} u$$

3. When the terms are complex and interpolating functions are of high order, it may be simpler and/or more efficient to use quadrature formulae.

Example

$$\underline{M}^{(n)} = \frac{L_n}{2}\int_{-1}^{+1}\begin{bmatrix}\phi_1\\ \phi_2\\ \phi_3\end{bmatrix}[\phi_1 \quad \phi_2 \quad \phi_3]d\xi$$

$$\phi_1 = \xi(\xi-1)/2 \qquad \phi_2 = (1-\xi^2) \qquad \phi_3 = \xi(1+\xi)/2$$

The highest product will involve quartic terms. We must select $n = 3$ and use 3 integration points which are specified as:

i	ξ_i	w_i
1	0	8/9
2	$+\sqrt{15/5}$	5/9
3	$+\sqrt{15/5}$	5/9

Thus $I = \frac{8}{9}f(0) + \frac{5}{9}f\left(\frac{-\sqrt{15}}{5}\right) + \frac{5}{9}f\left(\frac{+\sqrt{15}}{5}\right)$ where $f = \underline{\phi}^T\underline{\phi}$

Let's evaluate $\underline{\phi}$ at the 3 integration points:

$$\underline{\phi}\Big|_{\xi_1} = \left[\frac{0(0-1)}{2}, (1-0^2), \frac{0(1+0)}{2}\right] = [0,1,0]$$

$$\underline{\phi}\Big|_{\xi_2} = \left[\frac{\frac{\sqrt{15}}{5}\left(\frac{\sqrt{15}}{5}-1\right)}{2}, \left(1-\frac{15}{25}\right), \frac{\frac{\sqrt{15}}{5}\left(1+\frac{\sqrt{15}}{5}\right)}{2}\right] = [-0.08730, 0.4000, 0.68730]$$

$$\underline{\phi}\Big|_{\xi_3} = [0.68730, 0.40000, -0.08730]$$

Now take the products of the $\underline{\phi}$.

$$\frac{8}{9}\underline{\phi}^T\underline{\phi}\Big|_{\xi_1} = \frac{8}{9}\begin{bmatrix}0\\1\\0\end{bmatrix}[0 \quad 1 \quad 0] = \begin{bmatrix}0 & 0 & 0\\0 & 0.8889 & 0\\0 & 0 & 0\end{bmatrix}$$

$$\frac{5}{9}\underline{\phi}^T\underline{\phi}\Big|_{\xi_2} = \frac{5}{9}\begin{bmatrix}-0.08730\\0.40000\\0.68730\end{bmatrix}[-0.08730 \quad 0.4000 \quad 0.68730]$$

$$= \begin{bmatrix}0.004234 & -0.019400 & -0.03333\\-0.019400 & 0.088889 & 0.152733\\-0.03333 & 0.152733 & 0.262434\end{bmatrix}$$

$$\frac{5}{9}\underline{\phi}^T\underline{\phi}\Big|_{\xi_3} = \frac{5}{9}\begin{bmatrix} 0.68730 \\ 0.40000 \\ -0.08730 \end{bmatrix}[0.68730 \quad 0.4000 \quad -0.08730]$$

$$= \begin{bmatrix} 0.262434 & 0.152733 & -0.03333 \\ & 0.088889 & -0.019400 \\ & & 0.004234 \end{bmatrix}$$

Adding all the contributions together:

$$\underline{M}^{(n)} = \frac{L_n}{2}\left\{\frac{8}{9}\left(\underline{\phi}^T\underline{\phi}\right)\Big|_{\xi_1} + \frac{5}{9}\left(\underline{\phi}^T\underline{\phi}\right)\Big|_{\xi_2} + \frac{5}{9}\left(\underline{\phi}^T\underline{\phi}\right)\Big|_{\xi_3}\right\} = \frac{L_n}{2}\begin{bmatrix} 0.2666 & 0.13333 & -0.06666 \\ & 1.06666 & 0.13333 \\ & & 0.26668 \end{bmatrix}$$

- This matrix is identical to the one we developed analytically. The procedure is easy to implement, even for very high order interpolation.
- We note that when a number of known variable coefficients are included (e.g. $\underline{V}, \underline{D}$ etc.), it may be more economical to perform numerical integration.
- Finally we note that we always evaluate the interpolating function vector $\underline{\phi}$ (and/or $\underline{\phi}_{,\xi}$) at the integration points and <u>then</u> form products.

Order of the interpolating function

	$\underline{\phi}$	$\underline{\phi}_{,\xi}$
linear	1	0
quadratic	2	1
cubic	3	2
quartic	4	3

Order of products in our C-D formulation:

Interpolation	Order	$\underline{M^{(n)}}$ $\underline{\phi\phi}$	$\underline{A^{(n)}}$ $\underline{\phi\phi\phi_{,\xi}}$	$\underline{B^{(n)}}$ $\underline{\phi_{,\xi}\phi_{,\xi}}$	N required for Gaussian Quadrature*
Linear	1	2	2	0	2
Quadratic	2	4	5	2	3
Cubic	3	6	8	4	5
Quintic	4	8	11	6	6
INOR	INOR	2*INOR	3*INOR-1	2*(INOR-1)	

*In order for the integration to be exact.

Gaussian Quadrature is exact for degree 2n-1

n	Exact for
1	1
2	3
3	5
4	7
5	9
6	11

Time discretization for the C-D Equation

The system of global equations we wish to solve:

$$\underline{M}\underline{u}_{,t} + \left(\underline{A} + \underline{B}\right)\underline{u} = \underline{P}$$

let

$$t_{j+1} = t_j + \Delta t$$

Using a weighted implicit/explicit discretization we have

$$\underline{M}\left[\frac{\underline{u}_{j+1} - \underline{u}_j}{\Delta t}\right] + \left[\underline{A}_{j+1} + \underline{B}_{j+1}\right]\theta\underline{u}_{j+1} + \left[\underline{A}_j + \underline{B}_j\right](1-\theta)\underline{u}_j = \theta\underline{P}_{j+1} + (1-\theta)\underline{P}_j$$

$\theta =$ implicit fraction

$$\theta = \text{implicit fraction} = \begin{cases} 0 \rightarrow fully\ explicit \\ 1 \rightarrow fully\ implicit \\ 0.5 \rightarrow Crank - Nicolson \end{cases}$$

$\underline{M}$ does not have time varying components (except if the grid changes with time)

$\underline{A}$ and $\underline{B}$ do not have time varying components, we can have $V(t)$ and $D(t)$

$$\left\{\underline{M} + \Delta t\theta\left(\underline{A}_{j+1} + \underline{B}_{j+1}\right)\right\}\underline{u}_{j+1} = \underline{M}\underline{u}_j + (\theta - 1)\Delta t\left(\underline{A}_j + \underline{B}_j\right)\underline{u}_j + \Delta t\theta\underline{P}_j$$

- Recall that

$$\underline{M}^{(n)} = \int_{\Omega^{(n)}} \underline{\phi}^T \underline{\phi} dx$$

$\underline{M}^{(n)}$ is not time varying (unless the domain changes in time).

$$\underline{A}_j^{(n)} = \int_{\Omega^{(n)}} \underline{\phi}^T \underline{\phi} \underline{V}_j^{(n)} \underline{\phi}_{,x} dx$$

If $\underline{V}_j^{(n)}$ varies in time, we must re-evaluate it at every time step.

$$\underline{B}_j^{(n)} = D_j^{(n)} \int_{\Omega^{(n)}} \underline{\phi}_{,x}^T \underline{\phi}_{,x} dx$$

If $D^{(n)}$ varies in time, we must re-evaluate this matrix at every time step

Fully explicit time discretization

$$\underline{\underline{M}}\underline{u}_{j+1} = \underline{\underline{M}}\underline{u}_j + \Delta t\left(\underline{\underline{A}}_j + \underline{\underline{B}}_j\right)\underline{u}_j + \Delta t\underline{P}_j$$

- The explicit FE solution requires the solution of a matrix!
- We only need evaluate and decompose the system matrix once since it will never vary in time (even when *V* and *D* vary with time). This makes implementation more efficient since for direct matrix solvers:

  $0(n^3)$ operations for decomposition of matrix

  $0(n^2)$ operations for back substitution to solve for $\underline{u}_j$

Therefore if we need not modify the system matrix at every time step (this is indeed the case with a fully explicit scheme) we save at least a factor of $0(n)$ operations. However we note that the explicit scheme will have stability constraints on the size of the time step.

Scheme not fully explicit

Then the system matrix equals:

$$\underline{S} = \underline{M} + \Delta t\theta\left(\underline{A}_{j+1} + \underline{B}_{j+1}\right)$$

If either $\underline{A}_j^{(n)}$ or $\underline{B}_j^{(n)}$ vary in time due to variations in material properties *V* and *D* we must:

1. re-assemble the updated system matrix
2. re-decompose the system matrix

This involved at least $0(n^3)$ operations per time step.

We note that if *V* and *D* do not vary temporally, we need only assemble and decompose one for all time steps. Thus this would make the partially implicit scheme require the same order of operations per time step as the fully explicit scheme ($0(n^3)$ operations): However

i) The partially implicit system matrix is not symmetrical whereas the fully explicit is. Explicit storage and solution per time step is somewhat more efficient.

ii) If $\theta \geq 0.5$ we have no constraints on time step for stability.

Finally we note that if the grid varies spatially in time (e.g. due to flooding), we would always have to re-set and re-solve all matrices.

Symmetry of Coefficient Matrices

Explicit Schemes:

System matrix = $\{\underline{M}\}$

Therefore symmetric matrix (since $\underline{M}$ is symmetric)

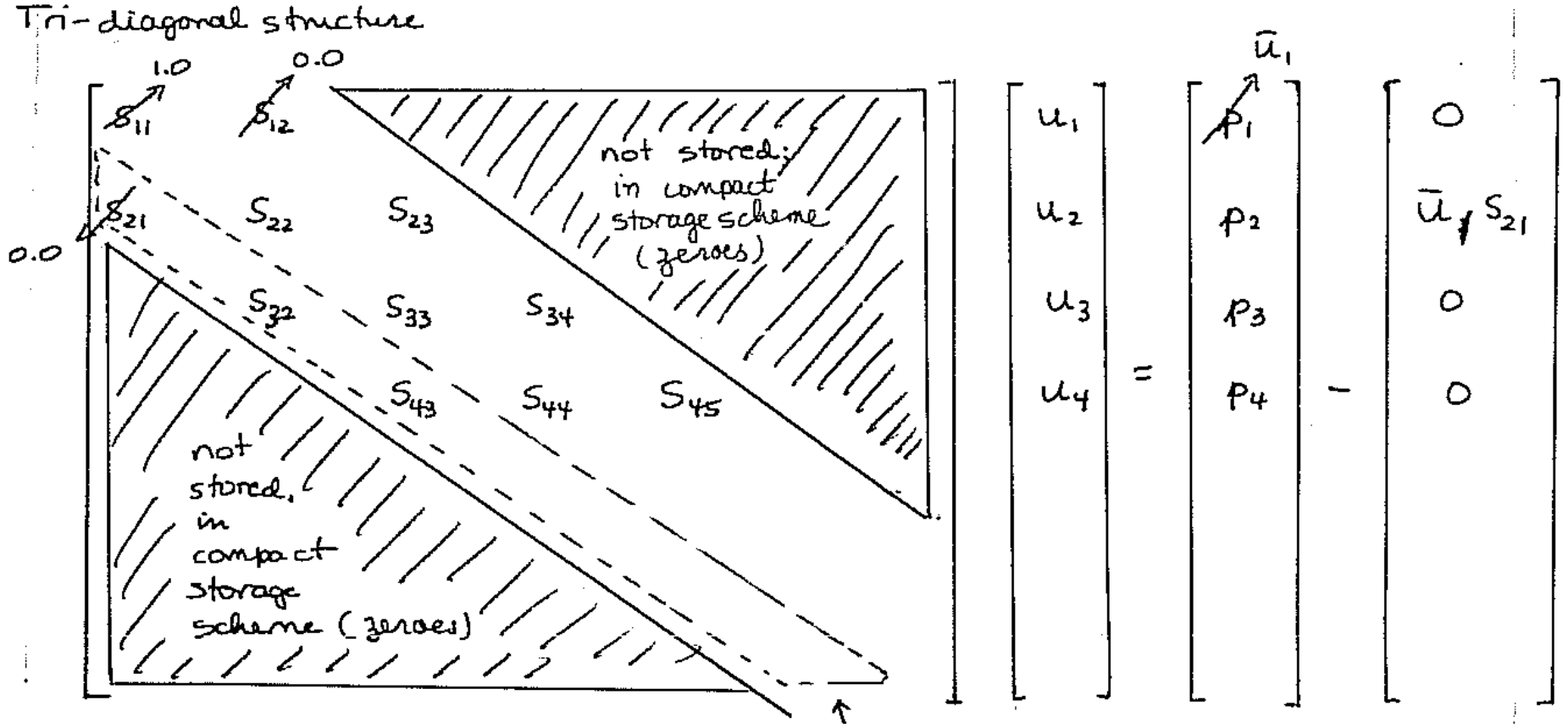
Implicit Schemes: (or partially implicit)

System matrix = $\{\underline{M} + \Delta t\theta(\underline{A}_{j+1} + \underline{B}_{j+1})\}$

Non-symmetric, since convection matrix $\underline{A}_{j+1}$ is not symmetric.

We must use a full matrix.

Introduction of essential b.c.'s into the symmetrical system matrix



Introduce b.c. $u_1 = \bar{u}_1$

Steps for full storage mode:

1. Zero out *row 1* and put 1 on the diagonal

2. Replace p_1 with $\bar{u}_1$

3. Now the system matrix is no longer symmetrical. Therefore we multiply the coefficients in column 1 by $\bar{u}_1$ and subtract from the right hand side. Then zero out column 1 in the system matrix.

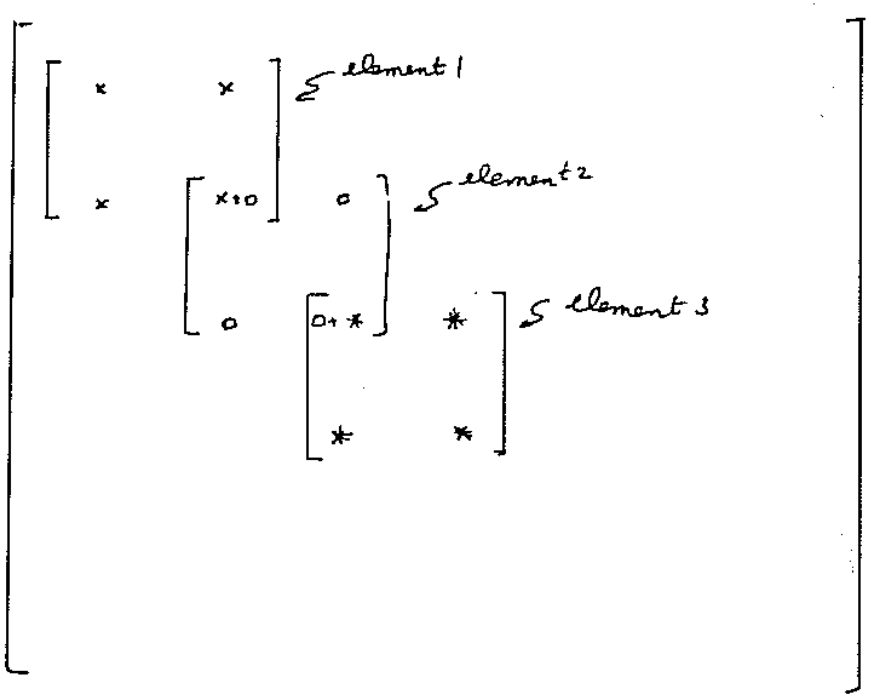
Steps for symmetrical storage mode:

1. Subtract off (row 1)^T multiplied by $\bar{u}_1$ from r.h.s. (this is equivalent to subtracting column 1 with the exception of the diagonal term)
2. Now zero out the 1st row and pout 1 on the diagonal and replace p_1 with $\bar{u}_1$

Global Matrix Structure for 1-D Spatial Problems:

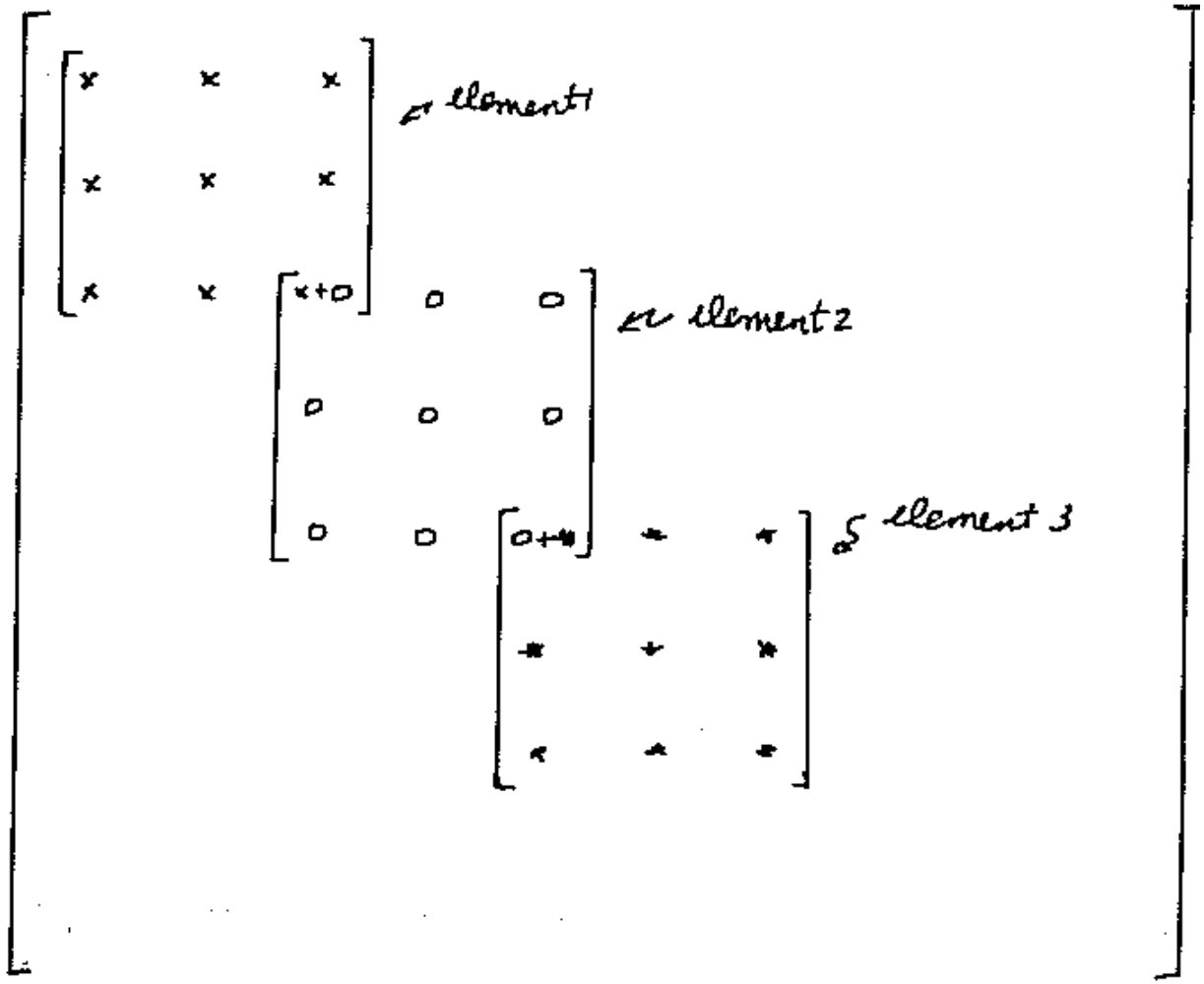
For linear Lagrange each element has 2 nodes and results in a 2x2 elemental matrix.

When considering functional continuity requirements and assembling into a global matrix we obtain a tri-diagonal matrix:



Note that the equations are associated with global nodes

For quadratic Lagrange, each element has 3 nodes and 3x3 elemental matrices which assemble into the following penta-diagonal global matrix:



Lumping of Mass Matrix

Producing a diagonal matrix from a banded mass matrix (typically for the matrix associated with the time derivative term)

$$\underline{M} \rightarrow \underline{D}$$

where $\underline{M}$ is a banded matrix and $\underline{D}$ is a diagonal matrix.

- We note that the inversion of a diagonal matrix is trivial

$$\underline{D} \rightarrow \underline{D}^{-1}$$

  We only take the reciprocal of each of the diagonal elements

Explicit Schemes

C-D equation:

$$\underline{M}\underline{u}_{j+1} = -\Delta t\left(\underline{A}_j + \underline{B}_j\right)\underline{u}_j + \Delta t\underline{P}_j + \underline{M}\underline{u}_j$$

Let's "diagonalize" $\underline{M} \rightarrow \underline{D}$

$$\begin{bmatrix} x & x & & & \\ x & x & x & & \\ & x & x & x & \\ & & x & x & x \\ & & & x & x \end{bmatrix} \rightarrow \begin{bmatrix} x & & & & \\ & x & & & \\ & & x & & \\ & & & x & \\ & & & & x \end{bmatrix}$$

$$\underline{D}u_{j+1} = -\Delta t\left(\underline{A}_j + \underline{B}_j\right)\underline{u}_j + \Delta t\underline{P}_j + \underline{D}u_j$$

The solution now becomes trivial at each time step. Simply invert $\underline{D} \rightarrow \underline{D}^{-1}$. Note that in the non-lumped case, we do not use an inversion process and instead must apply triangularization and back-substitution. For the lumped case:

$$\underline{D}^{-1}\underline{D}u_{j+1} = -\Delta t\underline{D}^{-1}\left(\underline{A}_j + \underline{B}_j\right)\underline{u}_j + -\Delta t\underline{D}^{-1}\underline{P}_j + \underline{D}^{-1}\underline{D}u_j$$

Therefore

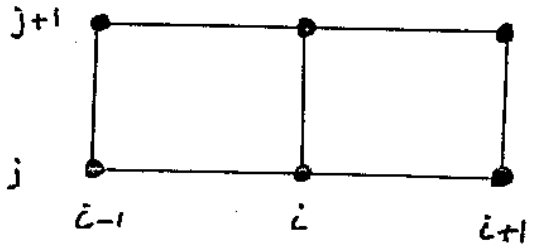
$$u_{j+1} = u_j - \Delta t\left(\underline{D}^{-1}\underline{A}_j + \underline{D}^{-1}\underline{B}_j\right)u_j + \Delta t\underline{D}^{-1}\underline{P}_j$$

Now, instead of back substituting for $\underline{u}_{j+1}$ we can directly solve. This simplifies things a lot!!!

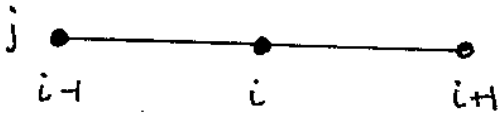
Molecules

$$\underline{M}\left(\frac{u_{j+1}-u_j}{\Delta t}\right) \qquad \left(\underline{A}_j + \underline{B}_j\right)\underline{u}_j$$

non lumped



non-lumped



Lumped

Lumped is not feasible!

Note that it is not possible to lump the convection or diffusion matrices. The spatial derivative terms would entirely lose their meaning! This is true for both implicit and explicit schemes.

Partially or Fully Implicit Schemes

$$\{\underline{M} + \Delta t\theta(\underline{A}_{j+1} + \underline{B}_{j+1})\}\underline{u}_{j+1} = (\theta - 1)\Delta t(\underline{A}_j + \underline{B}_j)\underline{u}_j + \Delta t\theta\underline{P}_{j+1} + \Delta t(\theta - 1)\underline{P}_j + \underline{M}\underline{u}_j$$

We can lump $\underline{M} \rightarrow \underline{D}$. We can not lump $\underline{A}$ or $\underline{B}$. Thus lumping does not reduce the band structure of the system matrix to a diagonal matrix. Therefore there is no reason to lump in a partially implicit scheme.

General Effects of Lumping

1. For an explicit scheme: Lumping drastically increases efficiency (particularly in 2-D/3-D applications).
2. Lumping is not useful in increasing efficiency in implicit schemes.
3. When comparing explicit to implicit schemes we should not that explicit schemes always involve stability constraints on time step. The time step constraint on the lumped explicit scheme will increase computational effort which may not be offset by increased efficiency of lumping, i.e. it may be better to use a partially implicit non-lumped scheme (in particular a C-N scheme).
4. Lumping always deteriorates the phase propagation characteristics of the FE scheme (for both C-D, N.S, wave equation, etc.)

5. The lumped explicit scheme using Lagrange linear elements yields the same nodal equations as the central FD explicit (FTCS) scheme. Therefore lumped FE scheme will be no worse than the FD scheme.

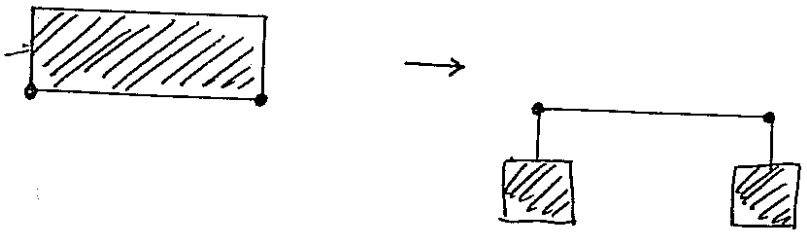
Implementation of Lumping

- A lumped diagonal matrix is obtained by adding the off-diagonal terms to the diagonal locations and then zeroing the off-diagonal locations.

$$\underline{M}^{(n)} = \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \rightarrow \underline{D}^{(n)} = \begin{bmatrix} x+x+x & 0 & 0 \\ 0 & x+x+x & 0 \\ 0 & 0 & x+x+x \end{bmatrix}$$

- Thus we lump the mass matrix but not the convection of diffusion matrices. Recall also that the mass matrix is associated with the time derivative (mass x acceleration = forces).
- Physical interpretation of lumping: replace a continuous string by a massless string with the distributed mass now being concentrated as "beads" at the nodes.

  For $p = 1$ (constant)



- Mathematical Interpretation of Lumping

$$\underline{M}^{(n)} = \frac{L_n}{2}\int_{-1}^{+1}\begin{bmatrix}\phi_1\\ \phi_2\end{bmatrix}[\phi_1 \quad \phi_2]d\xi = \frac{L_n}{2}\int_{-1}^{+1}\begin{bmatrix}\phi_1\phi_1 & \phi_1\phi_2\\ \phi_2\phi_1 & \phi_2\phi_2\end{bmatrix}d\xi$$

- Each term represents a quadratic.
- Lumping may be accomplished by using a Newton-Cotes quadrature rule which is one order. This will introduce some type of additional truncation error!
- Recall that Newton-Cotes uses integration points at evenly spaced points. Using 1 order too low and integration formula for the mass matrix will result in these integration points corresponding with the nodes.
- Newton-Cotes Quadrature

$n = 1 \qquad w_o = 1 \qquad \xi_o = -1$

$\qquad\qquad w_1 = 1 \qquad \xi_o = +1|$

This formula is exact for a linear polynomial. Furthermore the integration points correspong to the nodes.

- Let's examine the terms in the mass matrix:

$$\int_{-1}^{+1} \phi_1 \phi_1 d\xi = \sum_{j=0}^{1} w_j \phi_1 \phi_1 |_{\xi_j}$$

However

$$\phi_1 = \frac{1}{2}(1-\xi)$$

$$\phi_2 = \frac{1}{2}(1+\xi)$$

thus

$$\int_{-1}^{+1} \phi_1 \phi_1 d\xi = 1(1)(1) + (1)(0)(0) = 1$$

Similarly:

$$\int_{-1}^{+1} \phi_1 \phi_1 d\xi = \sum_{j=0}^{1} w_j (\phi_1 \phi_1)|_{\xi_j} = (1)(0)(1) + (1)(1)(0) = 0$$

In general:

$$\underline{M}^{(n)} \approx \underline{D} = \frac{L_n}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

This is the same matrix achieved by using exact quadrature and then "lumping" the terms directly onto the diagonal terms.

- However some error has been introduced into the evaluation of $\underline{M}$ since we used too low an order Newton Cotes rule. This leads to an additional truncation error.

  This explains why lumped formulations have poorer accuracy characteristics as compared to non-lumped schemes.

  i.e. phase errors are much more pronounced in lumped schemes, more oscillations are apparent.
- We can lump any matrix with non-derivative terms, e.g. mass matrix, force distribution matrices (e.g. bottom friction, Coriolis)