Parallel Hierarchic Adaptive Stabilized Transient Analysis (PHASTA)

**TABLE OF CONTENTS**

[**I. Incompressible Navies-Stokes equation** 2](#_Toc377291966)

[**II. Mappings** 8](#_Toc377291967)

[**III. Implicit time integration** 12](#_Toc377291968)

[**IV. Boundary condition** 17](#_Toc377291969)

# **I. Incompressible Navies-Stokes equation**

Let us state incompressible Navies- Stokes equation (strong form)

|  |  |  |
| --- | --- | --- |
|  |  | ( 1 ) |
|  |  | ( 2 ) |

Inertia (per volume) Divergence of stress

Unsteady Convective Pressure Viscosity Other

acceleration acceleration gradient body

forces

*( 1 )* is mass conversation

*( 2 )* are 3 momentum conservation equation

Finite element approach uses weak form of equations.

Let us derive it:

Introducing new notation *( 2 )*

Let’s write *( 1 )*\*q+; integrating it over our domain Ω (Γ Ω)

|  |  |  |
| --- | --- | --- |
|  |  | ( 3 ) |

In order to get weak form, we will integrate continuity and stress term of momentum equation by partly recall that

|  |  |  |
| --- | --- | --- |
|  |  | ( 4 ) |

Taking *( 4 )* into account we can rewrite *( 3 )* as:

|  |  |  |
| --- | --- | --- |
|  |  | ( 5 ) |

--Weak Galvevkin Form (this is the simplest form for I.N.S equation, which is unstable. On Page 9. We’ll introduce a stabilization term, and on Page 9 we’ll consider an additional mass conservation term (i.e. term, which makes the mass conservation equation satisfied).

Now let’s see what we can do with eqn *( 5 )*, assuming that it doesn’t need any extra terms

In order to solve *( 5 )*, we have to propose trial solution (x,y,z,t) and weight function w(x,y,z)

|  |  |  |
| --- | --- | --- |
| w: |  | ( 6 ) |
|  | ( 7 ) |
| y: |  | ( 8 ) |
|  | ( 9 ) |
|  | ( 10 ) |

Here we used A for weight function and B for trial function

Let’s plug *( 6 )* & *( 7 )* into *( 5 )*

|  |  |
| --- | --- |
|  | ( 11 ) |

Here we wrote momentum eq. contribution in the first term and continuity in the second term:

|  |  |  |
| --- | --- | --- |
|  |  | ( 12 ) |

Where , i=1,2,3 are momentum residuals,  is continuity residual

Provided that & are arbitrary we can conclude that

|  |  |  |
| --- | --- | --- |
|  |  | ( 13 ) |
|  |  | ( 14 ) |

We don’t evaluate integral *( 11 )* globally, we’ll do it from local point of view, which is more efficient. To emphasize the local point of view, we’ll use lower case superscripts a & B.

1d example

local

2

1

1

2

4

3

5

6

e1

e2

e3

e4

e5

global

1

2

4

3

5

6

e1

e2

e3

e4

e5

local

2

1

global

4

3

1

2

4

3

5

6

e1

e2

e3

e4

e5

local

Where

in our example

Now we have to break up the integral

|  |  |  |
| --- | --- | --- |
|  |  | ( 15 ) |

Let’s consider in detail one of elements, and write out the continuity residual for it:

|  |  |  |
| --- | --- | --- |
|  |  | ( 16 ) |

Note: is only the portion of the boundary of that is part of real

How to collect integral back according to *( 15 )*? (Put together all element integrals)

We can generate a connectivity table:

1

2

4

3

e1

e2

e3

|  |  |  |
| --- | --- | --- |
| el # local node # | 1 | 2 |
| 1 | 1 | 2 |
| 2 | 2 | 3 |
| 3 | 3 | 4 |

this is 1d example.

1

2

2

2

e=1

e=2

e=3

1

1

local

1

2

4

3

e=1

e=2

e=3

2

3

global

In the code we have ien (1:npro, 1:hshl), where 1:npro is el # and 1:hshl is local node #

Using this table we can do assemble operation

Which can be in the code as:

# **II. Mappings**

In order to make the integrals easier to evaluate, we need to map elements to a canonical element, e.g. in 2d

-1

1

1

-1

0

Mapping will do for us following transformations:

Derivation

|  |  |  |
| --- | --- | --- |
|  |  | ( 17 ) |

Where will be different for each element

Integrals:

|  |  |  |
| --- | --- | --- |
|  |  | ( 18 ) |
|  |  | ( 19 ) |

--Jacobian of the mapping x()

The advantage of using mappings:

Each element looks the same after applying the mapping all of the elements will be treated by the same routine (if they are the same type, like triangular, quadrilateral, tetrahedron, wedge, brick and pyramid, of course).

Let’s rewrite *( 16 )* using mapping

|  |  |  |
| --- | --- | --- |
|  |  | ( 20 ) |

Q: How do we evaluate integrals?

A: Using Gauss’s quadrature

|  |  |  |
| --- | --- | --- |
|  |  | ( 21 ) |

Where - function value at point , - weight for the quadrature point

We mentioned on Page 2 that Galerkin form *( 5 )* is unstable and we need an extra term to make it stable

|  |  |  |
| --- | --- | --- |
|  |  | ( 22 ) |

Momentum conservation continuity

The first term represents the momentum conservation and the second term represents the continuity.

To make continuity equation satisfied, we also have to add a mass conservation term to the Galerkin form *( 5 )*.

|  |  |  |
| --- | --- | --- |
|  |  | ( 23 ) |

Adding together eqns *( 5 )*, *( 22 )* & *( 23 )*, we can write

|  |  |  |
| --- | --- | --- |
|  |  | ( 24 ) |

Where

|  |  |  |
| --- | --- | --- |
|  |  | ( 25 ) |

Now we can put all terms of eq. *( 24 )* into residual vectors *( 13 )* & *( 14 )*.

Note that

goes to res(1:; 1:3)

goes to res(1:; 4)

* We assemble these global residuals from element level integrations

Let’s state that local form of the residuals (which are stored in -rl(:,:,:) in the code:

|  |  |  |
| --- | --- | --- |
|  |  | ( 26 ) |
|  |  | ( 27 ) |

Note:

In terms *( 15 )*, *( 18 )* & *( 20 )* of eq *( 26 )*, we have the following undefined so far parameters: (e2stab.f)

|  |  |  |
| --- | --- | --- |
|  |  | ( 28 ) |
|  |  | ( 29 ) |
|  |  | ( 30 ) |

Where is the metric tensor:

|  |  |  |
| --- | --- | --- |
|  |  | ( 31 ) |
|  |  |  |

# **III. Implicit time integration**

Let’s describe the iteration, moving from time step n to time step n+1, according to Generalized Alpha Method

Let’s first write it out for velocity vector

|  |  |  |  |
| --- | --- | --- | --- |
|  | =++-) | ( OD 1 ) | |
|  |  | | ( OD 2 ) |
|  |  | | ( OD 3 ) |
|  |  | | ( OD 4 ) |

Same equation for pressure:

|  |  |  |
| --- | --- | --- |
|  |  | ( P 1 ) |
|  |  | ( P 2 ) |
|  |  | ( P 3 ) |
|  |  | ( P 4 ) |

How predictor-multicorrector works?

1) predict =, =

2) invert *( OD 1 )*:

Invert *( P 1 )*

3) evaluate: ;;;

4) evaluate: ,

Reture to 1) increasing the iteration number i until residuals will be close enough to zero

Q: How we will choose ,,in order to get zero residuals?

A: we will use an analog of Newton’s method ( for )

Let’s linearize eqn *( OD 2 )* and *( P 2 )* about  and :

|  |  |  |
| --- | --- | --- |
|  |  | ( 32 ) |
|  |  | ( 33 ) |

Introducing new notation we can rewrite these equations as:

|  |  |  |
| --- | --- | --- |
|  |  | ( 34 ) |
|  |  | ( 35 ) |

Or, in matrix form

|  |  |  |
| --- | --- | --- |
|  |  | ( 36 ) |

Let’s write the alternate form of eq. *( 27 )* for L.N.S tangent of continuity

|  |  |  |
| --- | --- | --- |
|  |  | ( 37 ) |

Now we write matrices K, G D & C at the element level:

|  |  |  |
| --- | --- | --- |
|  |  | ( 38 ) |
|  |  | ( 39 ) |
|  |  | ( 40 ) |
|  |  | ( 41 ) |

Let us write out some derivations in terms of shape function

|  |  |  |
| --- | --- | --- |
|  |  | ( 42 ) |
|  |  | ( 43 ) |
|  |  | ( 44 ) |
|  |  | ( 45 ) |
|  |  | ( 46 ) |

Consider matrices *( 38 )*~ *( 42 )* in detail:

|  |  |  |
| --- | --- | --- |
|  |  | ( 47 ) |
|  |  | ( 48 ) |
|  |  | ( 49 ) |
|  |  | ( 50 ) |

# **IV. Boundary condition**

Generally, we can have 2 types of boundary

Conditions:

1) on

3 names of the same thing

2) on

Plus, we already described “periodical” boundary condition

Let’s consider essential B.C’s

--vector of quantities, which can be prescribed

(we are not considering pressure because this type of boundary condition is almost not applicable for it)

In general, we can express one in terms of another using direction cosines:

In order to have diagonal dominance in matrix , we can recorder r, s, t in such way that

Let’s express q in terms of given ? and solution variables

Inverting this: (for only)

Now we have correct the weight function to be sure that they satisfy homogeneous counterpart of the Dirichlet B.C.:

So substituting we can write

So each possible B.C. can be thought of as a transformation applied to w: