Navier – Stokes Equations Finite Element Discretization (as in Libmesh Example 13)

Nestor M. Solalinde

References

J.Donea. *Finite Element Methods for Flow Problems*

J.C.Heinrich. *Intermediate Finite Element Method*

B.Kirk, J.W.Peterson, R.H.Stogner, and G.F.Carey. *libMesh* ...

In[2056]:=

```
Clear["Global`*"]; dim = 2;

For[i = 1, i < dim + 1, i++, e0 = ConstantArray[0, dim]; e0[[i]] = 1; e_i = e0];
For[i = 1, i < dim + 2, i++, e0 = ConstantArray[0, dim + 1]; e0[[i]] = 1; ee_i = e0];
variables = {u, v, w, δ, w1, w2, w3, ukk, uk, vkk, vk, wkk, wk, μ, p, pkk, uold, vold, wold, pold};
wfunctions = {w1, w2, w3, w4};
unknowns = {u, v, w, p};
unknownskk = {ukk, vkk, wkk, pkk};
condition = NonConstants → variables;
Replacer2 = {};
Replacer3 = {};
Replacer4 = {{}, {}, {}, {}}; (* erases weighting functions*)
Replacer5 = {}; (*Second derivatives*)
Replacer6 = {ukk → u, vkk → v, wkk → w}; (*Used for getting tangent stifness matrix*)
Replacer7 = {{}, {}, {}, {}}; (*erases unknownskk*)
X_1 = x; X_2 = y; X_3 = z;
X_dim+1 = p;
For[i = 1, i < Dimensions[unknowns][[1]] + 1, i++,
  For[mm = 1, mm < dim + 1, mm++,
   AppendTo[Replacer7[[i]], Symbol["d" <> SymbolName[unknownskk[[i]]] <> "d" <> SymbolName[X_mm]] → 0];
   AppendTo[Replacer7[[i]], unknownskk[[i]] → 0];
  ]];

For[mm = 1, mm < dim + 1, mm++,
```

```
  For[i = 1, i < Dimensions[unknowns][[1]] + 1, i++,
   AppendTo[Replacer6, Symbol["d" <> SymbolName[unknowns[[i]]] <> "kkd" <> SymbolName[Xmm]]
     → Symbol["d" <> SymbolName[unknowns[[i]]] <> "d" <> SymbolName[Xmm]]];
   AppendTo[Replacer6, Symbol["d2" <> SymbolName[unknowns[[i]]] <> "kkd" <> SymbolName[Xmm] <> "2"]
     → Symbol["d2" <> SymbolName[unknowns[[i]]] <> "d" <> SymbolName[Xmm] <> "2"]] ;
  ]];
For [j = 1, j < Dimensions[wfunctions][[1]] + 1, j++,
  For[i = 1, i < Dimensions[wfunctions][[1]] + 1, i++,
   If[i ≠ j,
    AppendTo[Replacer4[[j]], wfunctions[[i]] → 0];
    For[mm = 1, mm < dim + 1, mm++,
     AppendTo[Replacer4[[j]], Symbol["d" <> SymbolName[wfunctions[[i]]] <> "d" <> SymbolName[Xmm]] → 0]
    ]]]];
For[mm = 1, mm < dim + 1, mm++,
  For[i = 1, i < Dimensions[variables][[1]] + 1, i++,
   AppendTo[Replacer2, D[variables[[i]], Xmm, condition] → Symbol["d" <> SymbolName[variables[[i]]] <> "d'
   AppendTo[Replacer5, D[variables[[i]],
      {Xmm, 2}, condition] → Symbol["d2" <> SymbolName[variables[[i]]] <> "d" <> SymbolName[Xmm] <> "2"]];
  ]];
For[i = 1, i < Dimensions[variables][[1]] + 1, i++,
 AppendTo[Replacer2, D[variables[[i]], t, condition] → Symbol["d" <> SymbolName[variables[[i]]] <> "d" <> !
 AppendTo[Replacer3, Symbol["d" <> SymbolName[variables[[i]]] <> "d" <> SymbolName[t]] → (Symbol[SymbolNam
     - Symbol[SymbolName[variables[[i]]] <> "old"]) / Δt];]
Dc[a_, b_] := D[a, b, condition] /. Replacer2; (*Dc equals partial derivative*)
Scalar2d[a_, b_] := a[[1]] * b[[1]] + a[[2]] * b[[2]];
Grad[V_] := Module[{output = {}}, For[i = 1, i < dim + 1, i++, AppendTo[output, Dc[V, Xi]]]; output]
Dyad[A_, B_] := Sum_{i=1}^{dim} (Sum_{j=1}^{dim} (A[[i, j]] * B[[i, j]]));
SplitByTrial[func_] := Module[{output = {}},
   For[j = 1, j < dim + 2, j++, AppendTo[output, func /. Replacer4[[j]] ]]; output];
Laplacian[func_] := Sum[D[D[func, Xi, condition], Xi, condition], {i, dim}] /. Replacer5 /. Replacer2;
Scalar[A_, B_] := Module[{size = Dimensions[A][[1]], output}, output = Sum[A[[i]] * B[[i]], {i, size}]; outp
Div[A_] := Sum[Dc[A[[i]], Xi], {i, dim}];
SplitByUnknown[V_] := Module[{output = {}, auxiliar},
   For[j = 1, j < dim + 1, j++, AppendTo[output, V - (V /. Replacer7[[j]])]];
```

```
    AppendTo[output, V - (V /. Replacer7[[4]])];
    auxiliar = V;
    For[j = 1, j < 5, j++, auxiliar = auxiliar /. Replacer7[[j]]];
    AppendTo[output, auxiliar];
    Transpose[FullSimplify[output]]];
SplitByBoth[func_] := SplitByUnknown[SplitByTrial[func]];

u1 = u; u2 = v; u3 = w;
u21 = ukk * θ + (1 - θ) * uold;
u22 = vkk * θ + (1 - θ) * vold;
u23 = wkk * θ + (1 - θ) * wold;
p2 = pkk * θ + (1 - θ) * pold;
ΔU1 = ukk - u;
ΔU2 = vkk - v;
ΔU3 = wkk - w;
ΔU4 = pkk - p;
Vkk1 = ukk;
Vkk2 = vkk;
Vkk3 = wkk;



V = Sum[ui * ei, {i, dim}]; (*Velocity Vector*)
W = Sum[wfunctions[[i]] * ei, {i, dim}]; (*velocity weighting functions*)
V2 = Sum[u2i * ei, {i, dim}];
Vkk = Sum[Vkki * ei, {i, dim}];
U = Sum[unknowns[[i]] * eei, {i, dim}] + p * eedim+1;
ΔU = Sum[ΔUi * eei, {i, dim}] + ΔU4 * eedim+1;(*where ukk = u^(k+1) is the value of u for the next iteration, ar
Ukk = U + ΔU;

(*Stress tensor*)
τ[u_] := Module[{t2 = ConstantArray[ConstantArray[0, dim], dim]}, For[i = 1, i < dim + 1, i++,
    For[j = 1, j < dim + 1, j++, t2[[j, i]] = μ * (Dc[u[[i]], Xj])]](*+Dc[u[[j]],xi]) = 0 when div[ρv]=0 *); t2
```

```
(*Terms of the Navier Stokes Equation*)
TEMPORAL = SplitByTrial[Scalar[Dc[ρ * V, t], W]] /. Replacer3 ;
DIFUSSION =  SplitByTrial[Dyad[Grad[W], τ[V2]]]; (*Dyad Product Between Grad[w] and the stress tensor*)
CONVECTION = SplitByTrial[ρ * Scalar[(Scalar[Grad[V2], V]), W]];
PRESSURE = SplitByTrial[Scalar[Grad[p2], W]];
CONTINUITY = SplitByTrial[w3 * Div[Vkk]];
DD = TEMPORAL + CONVECTION + DIFUSSION + PRESSURE + CONTINUITY ;
TANGENT[func_] := Module[{DDDu2 = ConstantArray[ConstantArray[0, dim + 1], dim + 1], i},
   For [i = 1, i < dim + 2, i++,
    DDDu2[[i]] = D[Sum[func[[i]], {i, dim + 1}], Scalar[U, ee_i]]]; SplitByTrial[DDDu2 /. Replacer6]];
NEWTON = TANGENT[DD].ΔU;(*Newton Term*)


Print["VARIABLES"];
Print["ukk,vkk,pkk are the unknowns to solve by the linear system"];
Print[" u,v,p are previous Newton iterations"];
Print[" uold,vold,pold are previous timesteps"];
Print["τ[V] = μ*Grad[V] = ", τ[V]];
Print["V = ", V, " = Velocity Vector"];
Print["Vkk = ", Vkk];
Print["W = ", W, " = Trial Functions for velocities"];
Print["V2 = ", V2];
Print["U = ", U];
Print["Ukk = ", Ukk, " = Unknowns vector"];
Print["ΔU = ", ΔU];
Print["\n PART 1"];
Print["TEMPORAL = Scalar[Dc[ρ*V,t],W] = ", TEMPORAL // MatrixForm];
Print["DIFUSSION = Dyad[Grad[W],τ[V2]] = ", DIFUSSION // MatrixForm];
Print["CONVECTION = ρ*Scalar[(Scalar[Grad[V2],V]),W] = ", CONVECTION // MatrixForm];
Print[" PRESSURE = Scalar[Grad[p2],W] = ", PRESSURE // MatrixForm];
Print["CONTINUITY = w3*Div[Vkk] = ", CONTINUITY // MatrixForm];
Print["DD = TEMPORAL+CONVECTION+DIFUSSION+PRESSURE + CONTINUITY = ", DD // MatrixForm];
Print["DD'[V] = ", TANGENT[DD], "  = Tangent stiffness matrix of DD"];
Print["NEWTON = DD'[u].ΔU = ", NEWTON // MatrixForm]
```

```
Print["\n PART 2"];
force = {f1, f2};
σ = -p * IdentityMatrix[dim] + τ[V];
RESIDUAL[V2_] := Dc[V2, t] + Scalar[V2, Grad[V]] - μ * Laplacian[V2] + Grad[p2] - f;
SUPGTEMPORAL = tsupg * SplitByTrial[Scalar[Scalar[V, Grad[W]], Dc[V, t]] /. Replacer3];
SUPGDIFFUSION = tsupg * SplitByTrial[Scalar[Scalar[V, Grad[W]], -μ * Laplacian[V2]]];
SUPGCONVECTION = tsupg * SplitByTrial[ Scalar[Scalar[V, Grad[W]], Scalar[V, Grad[V2]]]];
SUPGPRESSURE = tsupg * SplitByTrial[ Scalar[Scalar[V, Grad[W]], Grad[p2]]];
SUPGFORCE = tsupg * SplitByTrial[ Scalar[Scalar[V2, Grad[W]], -force]];
SUPGNEWTONDIFFUSION = TANGENT[SUPGDIFFUSION].ΔU;
SUPGNEWTONCONVECTION = FullSimplify[TANGENT[SUPGCONVECTION].ΔU];
SUPGNEWTONPRESSURE = FullSimplify[TANGENT[SUPGPRESSURE].ΔU];
SUPGNEWTONFORCE = FullSimplify[TANGENT[SUPGFORCE].ΔU];
Print["SUPGTEMPORAL = tsupg*Scalar[Scalar[V,Grad[W]],Dc[V,t]] = ", SUPGTEMPORAL // MatrixForm];
Print["SUPGDIFFUSION = tsupg*Scalar[Scalar[V,Grad[W]],-Mu/Ree*Laplacian[V2]] = ", SUPGDIFFUSION // Matr
Print["SUPGCONVECTION = tsupg*Scalar[Scalar[V,Grad[W]],Scalar[V,Grad[V2]]] = ", SUPGCONVECTION // Matri
Print["SUPGPRESSURE = Scalar[Scalar[V,Grad[W]],Grad[p2]] = ", SUPGPRESSURE // MatrixForm];
Print["SUPGFORCE = Scalar[Scalar[V2,Grad[W]],-force] = ", SUPGFORCE // MatrixForm];
Print["SUPGNEWTONDIFFUSION = TANGENT[SUPGDIFFUSION].ΔU = ", SUPGNEWTONDIFFUSION // MatrixForm];
Print["SUPGNEWTONCONVECTION = TANGENT[SUPGCONVECTION].ΔU = \n", SUPGNEWTONCONVECTION // MatrixForm];
Print["SUPGNEWTONPRESSURE = TANGENT[SUPGPRESSURE].ΔU = ", SUPGNEWTONPRESSURE // MatrixForm];
Print["SUPGNEWTONFORCE = TANGENT[SUPGFORCE].ΔU = ", SUPGNEWTONFORCE // MatrixForm];

GLOBALRESULT = {};
AppendTo[GLOBALRESULT, SplitByUnknown[TEMPORAL]];
AppendTo[GLOBALRESULT, SplitByUnknown[DIFUSSION]];
AppendTo[GLOBALRESULT, SplitByUnknown[CONVECTION]];
AppendTo[GLOBALRESULT, SplitByUnknown[PRESSURE]];
AppendTo[GLOBALRESULT, SplitByUnknown[CONTINUITY]];
AppendTo[GLOBALRESULT, SplitByUnknown[NEWTON]];
AppendTo[GLOBALRESULT, SplitByUnknown[SUPGTEMPORAL]];
AppendTo[GLOBALRESULT, SplitByUnknown[SUPGCONVECTION]];
AppendTo[GLOBALRESULT, SplitByUnknown[SUPGPRESSURE]];
AppendTo[GLOBALRESULT, SplitByUnknown[SUPGFORCE]];
AppendTo[GLOBALRESULT, SplitByUnknown[SUPGNEWTONDIFFUSION]];
```

```
AppendTo[GLOBALRESULT, SplitByUnknown[SUPGNEWTONCONVECTION]];
AppendTo[GLOBALRESULT, SplitByUnknown[SUPGNEWTONPRESSURE]];
GLOBALRESULT = GLOBALRESULT /. {Θ → theta, ρ → Rho, μ → Mu, Δt → dt};
```

VARIABLES

ukk,vkk,pkk are the unknowns to solve by the linear system

u,v,p are previous Newton iterations

uold,vold,pold are previous timesteps

$\tau[\mathrm{V}] = \mu{*}\mathrm{Grad}[\mathrm{V}] = \begin{pmatrix} \mathrm{dudx}\,\mu & \mathrm{dvdx}\,\mu \\ \mathrm{dudy}\,\mu & \mathrm{dvdy}\,\mu \end{pmatrix}$

$\mathrm{V} = \{u, v\}$ = Velocity Vector

$\mathrm{Vkk} = \{\mathrm{ukk}, \mathrm{vkk}\}$

$\mathrm{W} = \{\mathrm{w1}, \mathrm{w2}\}$ = Trial Functions for velocities

$\mathrm{V2} = \{\mathrm{uold}\,(1-\theta) + \mathrm{ukk}\,\theta, \mathrm{vold}\,(1-\theta) + \mathrm{vkk}\,\theta\}$

$\mathrm{U} = \{u, v, p\}$

$\mathrm{Ukk} = \{\mathrm{ukk}, \mathrm{vkk}, \mathrm{pkk}\}$ = Unknowns vector

$\Delta\mathrm{U} = \{\mathrm{ukk} - u, \mathrm{vkk} - v, \mathrm{pkk} - p\}$

PART 1

$$\mathrm{TEMPORAL} = \mathrm{Scalar}[\mathrm{Dc}[\rho{*}\mathrm{V},t],\mathrm{W}] = \begin{pmatrix} \frac{(\mathrm{ukk}-\mathrm{uold})\,\mathrm{w1}\,\rho}{\Delta t} \\ \frac{(\mathrm{vkk}-\mathrm{vold})\,\mathrm{w2}\,\rho}{\Delta t} \\ 0 \end{pmatrix}$$

$$\mathrm{DIFUSSION} = \mathrm{Dyad}[\mathrm{Grad}[\mathrm{W}],\tau[\mathrm{V2}]] = \begin{pmatrix} \mathrm{dw1dx}\,(\mathrm{duolddx}\,(1-\theta) + \mathrm{dukkdx}\,\theta)\,\mu + \mathrm{dw1dy}\,(\mathrm{duolddy}\,(1-\theta) + \mathrm{dukkdy}\,\theta)\,\mu \\ \mathrm{dw2dx}\,(\mathrm{dvolddx}\,(1-\theta) + \mathrm{dvkkdx}\,\theta)\,\mu + \mathrm{dw2dy}\,(\mathrm{dvolddy}\,(1-\theta) + \mathrm{dvkkdy}\,\theta)\,\mu \\ 0 \end{pmatrix}$$

$$\text{CONVECTION} = \rho{*}\text{Scalar[(Scalar[Grad[V2],V]),W]} = \begin{pmatrix} \text{w1}\,(u\,(\text{duolddx}\,(1-\theta)+\text{dukkdx}\,\theta)+v\,(\text{duolddy}\,(1-\theta)+\text{dukkdy}\,\theta))\,\rho \\ \text{w2}\,(u\,(\text{dvolddx}\,(1-\theta)+\text{dvkkdx}\,\theta)+v\,(\text{dvolddy}\,(1-\theta)+\text{dvkkdy}\,\theta))\,\rho \\ 0 \end{pmatrix}$$

$$\text{PRESSURE} = \text{Scalar[Grad[p2],W]} = \begin{pmatrix} \text{w1}\,(\text{dpolddx}\,(1-\theta)+\text{dpkkdx}\,\theta) \\ \text{w2}\,(\text{dpolddy}\,(1-\theta)+\text{dpkkdy}\,\theta) \\ 0 \end{pmatrix}$$

$$\text{CONTINUITY} = \text{w3}{*}\text{Div[Vkk]} = \begin{pmatrix} 0 \\ 0 \\ (\text{dukkdx}+\text{dvkkdy})\,\text{w3} \end{pmatrix}$$

$$\text{DD} = \text{TEMPORAL+CONVECTION+DIFUSSION+PRESSURE + CONTINUITY} = \begin{pmatrix} \text{w1}\,(\text{dpolddx}\,(1-\theta)+\text{dpkkdx}\,\theta)+\text{dw1dx}\,(\text{duolddx}\,(1-\theta)+\text{dukkdx}\,\theta \\ \text{w2}\,(\text{dpolddy}\,(1-\theta)+\text{dpkkdy}\,\theta)+\text{dw2dx}\,(\text{dvolddx}\,(1-\theta)+\text{dvkkdx}\,\theta \\ \end{pmatrix}$$

$$\text{DD'[V]} = \begin{pmatrix} \text{w1}\,(\text{duolddx}\,(1-\theta)+\text{dudx}\,\theta)\,\rho & \text{w1}\,(\text{duolddy}\,(1-\theta)+\text{dudy}\,\theta)\,\rho & 0 \\ \text{w2}\,(\text{dvolddx}\,(1-\theta)+\text{dvdx}\,\theta)\,\rho & \text{w2}\,(\text{dvolddy}\,(1-\theta)+\text{dvdy}\,\theta)\,\rho & 0 \\ 0 & 0 & 0 \end{pmatrix} = \text{Tangent stiffness matrix of DD}$$

$$\text{NEWTON} = \text{DD'[u].}\Delta\text{U} = \begin{pmatrix} (\text{ukk}-u)\,\text{w1}\,(\text{duolddx}\,(1-\theta)+\text{dudx}\,\theta)\,\rho+(\text{vkk}-v)\,\text{w1}\,(\text{duolddy}\,(1-\theta)+\text{dudy}\,\theta)\,\rho \\ (\text{ukk}-u)\,\text{w2}\,(\text{dvolddx}\,(1-\theta)+\text{dvdx}\,\theta)\,\rho+(\text{vkk}-v)\,\text{w2}\,(\text{dvolddy}\,(1-\theta)+\text{dvdy}\,\theta)\,\rho \\ 0 \end{pmatrix}$$

PART 2

$$\text{SUPGTEMPORAL} = \text{tsupg}{*}\text{Scalar[Scalar[V,Grad[W]],Dc[V,t]} = \begin{pmatrix} \frac{\text{tsupg (ukk}-\text{uold) (dw1dx}\,u+\text{dw1dy}\,v)}{\Delta\text{t}} \\ \frac{\text{tsupg (dw2dx}\,u+\text{dw2dy}\,v)\,(\text{vkk}-\text{vold})}{\Delta\text{t}} \\ 0 \end{pmatrix}$$

$$\text{SUPGDIFFUSION} = \text{tsupg}{*}\text{Scalar[Scalar[V,Grad[W]],}-\text{Mu/Ree}{*}\text{Laplacian[V2]]} = \begin{pmatrix} -\text{tsupg}\,(\text{dw1dx}\,u+\text{dw1dy}\,v)\,(\text{d2uolddx2}\,(1-\theta)+\text{d2uolddy2}\,(1-\theta)+ \\ -\text{tsupg}\,(\text{dw2dx}\,u+\text{dw2dy}\,v)\,(\text{d2volddx2}\,(1-\theta)+\text{d2volddy2}\,(1-\theta)+ \\ 0 \end{pmatrix}$$

$$\text{SUPGCONVECTION} = \text{tsupg}*\text{Scalar[Scalar[V,Grad[W]],Scalar[V,Grad[V2]]]} = \begin{pmatrix} \text{tsupg}\,(\text{dw1dx}\,u + \text{dw1dy}\,v)\,(u\,(\text{duolddx}\,(1-\theta) + \text{dukkdx}\,\theta) + v\,(\text{duolddy}\,(\\ \text{tsupg}\,(\text{dw2dx}\,u + \text{dw2dy}\,v)\,(u\,(\text{dvolddx}\,(1-\theta) + \text{dvkkdx}\,\theta) + v\,(\text{dvolddy}\,(\\ 0 \end{pmatrix}$$

$$\text{SUPGPRESSURE} = \text{Scalar[Scalar[V,Grad[W]],Grad[p2]]} = \begin{pmatrix} \text{tsupg}\,(\text{dw1dx}\,u + \text{dw1dy}\,v)\,(\text{dpolddx}\,(1-\theta) + \text{dpkkdx}\,\theta) \\ \text{tsupg}\,(\text{dw2dx}\,u + \text{dw2dy}\,v)\,(\text{dpolddy}\,(1-\theta) + \text{dpkkdy}\,\theta) \\ 0 \end{pmatrix}$$

$$\text{SUPGFORCE} = \text{Scalar[Scalar[V2,Grad[W]],}-\text{force]} = \begin{pmatrix} -\text{f1 tsupg}\,(\text{dw1dx}\,(\text{uold}\,(1-\theta) + \text{ukk}\,\theta) + \text{dw1dy}\,(\text{vold}\,(1-\theta) + \text{vkk}\,\theta)) \\ -\text{f2 tsupg}\,(\text{dw2dx}\,(\text{uold}\,(1-\theta) + \text{ukk}\,\theta) + \text{dw2dy}\,(\text{vold}\,(1-\theta) + \text{vkk}\,\theta)) \\ 0 \end{pmatrix}$$

$$\text{SUPGNEWTONDIFFUSION} = \text{TANGENT[SUPGDIFFUSION]}.\Delta\text{U} = \begin{pmatrix} -\text{dw1dx tsupg}\,(\text{ukk} - u)\,(\text{d2uolddx2}\,(1-\theta) + \text{d2uolddy2}\,(1-\theta) + \text{d2udx2}\,\theta + \text{d2udy} \\ -\text{dw2dx tsupg}\,(\text{ukk} - u)\,(\text{d2volddx2}\,(1-\theta) + \text{d2volddy2}\,(1-\theta) + \text{d2vdx2}\,\theta + \text{d2vdy} \\ \end{pmatrix}$$

$$\text{SUPGNEWTONCONVECTION} = \text{TANGENT[SUPGCONVECTION]}.\Delta\text{U} =$$

$$\begin{pmatrix} \text{tsupg}\,(\text{ukk} - u)\,(-\text{duolddy dw1dx}\,v\,(\theta - 1) - \text{duolddx}\,(2\,\text{dw1dx}\,u + \text{dw1dy}\,v)\,(\theta - 1) + (2\,\text{dudx dw1dx}\,u + \text{dudy dw1dx}\,v + \text{dudx dw1dy}\,v)\,\theta) + \text{tsupg}\,(\text{vkk} - v)\,(\\ \text{tsupg}\,(\text{ukk} - u)\,(-\text{dvolddy dw2dx}\,v\,(\theta - 1) - \text{dvolddx}\,(2\,\text{dw2dx}\,u + \text{dw2dy}\,v)\,(\theta - 1) + (2\,\text{dvdx dw2dx}\,u + \text{dvdy dw2dx}\,v + \text{dvdx dw2dy}\,v)\,\theta) + \text{tsupg}\,(\text{vkk} - v)\,(\\ 0 \end{pmatrix}$$

$$\text{SUPGNEWTONPRESSURE} = \text{TANGENT[SUPGPRESSURE]}.\Delta\text{U} = \begin{pmatrix} \text{tsupg}\,(\text{dw1dx}\,(\text{ukk} - u) + \text{dw1dy}\,(\text{vkk} - v))\,(-\theta\,\text{dpolddx} + \text{dpolddx} + \text{dpdx}\,\theta) \\ \text{tsupg}\,(\text{dw2dx}\,(\text{ukk} - u) + \text{dw2dy}\,(\text{vkk} - v))\,(-\theta\,\text{dpolddy} + \text{dpolddy} + \text{dpdy}\,\theta) \\ 0 \end{pmatrix}$$

$$\text{SUPGNEWTONFORCE} = \text{TANGENT[SUPGFORCE]}.\Delta\text{U} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

In[1812]:=

```
Clear[tsupg];
fname = FileNameJoin[{"D:\Documentos 2010\Mathematica Projects", "pre_stokes4cpp.txt"}];
s = OpenWrite[fname];
For[i = 1, i < dim + 2, i++,
  WriteString[s, "F" <> SymbolName[U[[i]]] <> "(i) += JxW[qp]*(\n"];
  For[k = 1, k < Dimensions[GLOBALRESULT][[1]] + 1, k++,
   WriteString[s, "                    +"]; Write[s, -GLOBALRESULT[[k, i, 4]]];
  ];
  WriteString[s, "                    );\n \n"];
 ];



For[i = 1, i < dim + 2, i++,
  For[j = 1, j < dim + 2, j++,
   WriteString[s, "K" <> SymbolName[U[[i]]] <> SymbolName[U[[j]]] <> "(i,j) += JxW[qp]*(\n"];
   For[k = 1, k < Dimensions[GLOBALRESULT][[1]] + 1, k++,
    WriteString[s, "                    +"]; Write[s, GLOBALRESULT[[k, i, j]]];
   ];
   WriteString[s, "                    );\n \n"];
  ]];


Close[s];
```

In[792]:= `Dimensions[GLOBALRESULT]`

Out[792]= {12, 3, 4}