



UCSC

**FACULTAD DE
INGENIERÍA**

DEPARTAMENTO
DE MATEMÁTICA
Y FÍSICA APLICADAS

Introducción a la Programación de Elementos Finitos con FreeFem++ y su Aplicación en Mecánica de Fluidos

Sergio Caucao

Departamento de Matemática y Física Aplicadas (DMFA)

Grupo de Investigación en Análisis Numérico y Cálculo Científico (GIANuC²)

XXXVI Jornada de Matemática de la Zona Sur

Universidad Católica de Temuco

24, 25, y 26 de Abril, 2024

Ingresar a:

www.menti.com

6988 3769

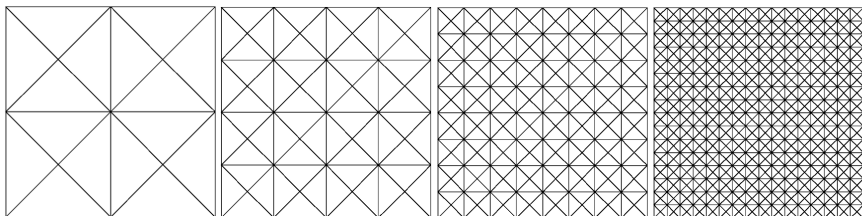
¿Qué entiendes por Método de Elementos Finitos ?

Una respuesta con un poco de historia ...

- El Método de los Elementos Finitos (MEF) es una técnica numérica usada ampliamente en el contexto de ingeniería y aplicaciones varias.
- El MEF es capaz de manejar sistemas complejos para los cuales no puede encontrarse soluciones analíticas explícitas.
- Fue desarrollado por primera vez en 1943 por Richard Courant, quien utilizó el método de Ritz del análisis numérico y la minimización del cálculo variacional para obtener soluciones aproximadas a sistemas vibrantes.
- En un artículo publicado en 1956 por M. J. Turner, R. W. Clough, H. C. Martin, y L. J. Topp, se estableció una definición más amplia del método y análisis numérico. El artículo se focalizaba en la rigidez y la deflexión de estructuras complejas. De aquí se acuñan los términos: Matriz de Rigidez y Vector de carga.

Una respuesta a grandes rasgos ...

- Dada una EDP, por ejemplo:
$$\begin{cases} -\mathbf{D}^{-1} \Delta p = f & \text{en } \Omega, \\ p = 0 & \text{en } \partial\Omega. \end{cases} \quad (\text{Ecuación de Darcy})$$
- Obtener su **formulación variacional** a nivel continuo en H y **discreto en H_h** .
- Particionar/dividir el dominio donde se quiere resolver la EDP, en elementos más pequeños y finitos:




Una respuesta a grandes rasgos ...

- Escoger espacios discretos (polinomiales) H_h contenidos en su contraparte continua H .
- Considerar **funciones bases** φ_n , con n la cantidad de incógnitas, tales que, cualquier función en dicho espacio de dimensión finita se pueda escribir como una combinación lineal de ellas.
- En particular, $p_h = \sum_{i=1}^n \alpha_i \varphi_i$, con α_i incógnitas a calcular.
- Mediante cálculos locales (en cada triángulo de la malla) y un proceso de ensamble de la información, generar la matriz **A** y el vector **F** globales.
- Así, el problema original se reduce a resolver el sistema de ecuaciones lineales:

$$\mathbf{A} \boldsymbol{\alpha} = \mathbf{F}, \text{ con } \boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_n).$$

Programación del Método de Elementos Finitos con FreeFem++

FreeFem++ es un toolbox para resolver ecuaciones diferenciales parciales. Creado y desarrollado desde 1987. Su sitio oficial es: <http://www.freefem.org>.

 [DOCUMENTATION](#) [COMMUNITY](#) [MODULES](#) [SOURCE CODE](#) [GALLERY](#) [EVENTS](#) [TRY IT ONLINE](#) [DONATE](#)

FREEFEM DAYS
15th EDITION - PARIS
FreeFem days 7 & 8 december 2023

```
Load "mesh3"

// Parameters
int nn = 20; // Mesh quality


// Mesh
int[] labs = {1, 2, 2, 1, 1, 1, 2}; // Label mesh3
mesh3 Th = cube(nn, nn, nn, label=labs);
// Remove the 10.5,11.5 domain of the cube
Th = trunc(Th, {x < 0.5} | {y < 0.5} | {z < 0.5}

// Fespace
fespace Vh(Th, P1);
Vh u, v;

// Macro
macro Grad(u) {dx(u), dy(u), dz(u)} //

// Define the weak form and solve
solve Poisson(u, v, solver=Cg)
= int3d(Th) |
  Grad(u)' * Grad(v)
- int3d(Th) |
  1 * v
+ on(1, u=0)
;

// Plot
plot(u, nbiso=25);
```




A high level multiphysics finite element software

FreeFEM offers a fast interpolation algorithm
and a language for the manipulation of data on
multiple meshes.

v4.13
Release notes

Download


All platforms (LGPL 3.0)



Creador: **Frédéric Hecht**

FreeFem++: Algunos Datos Importantes

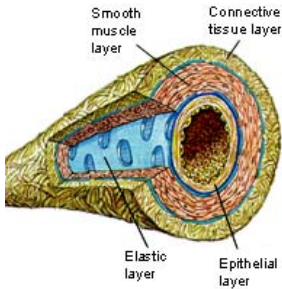
- 1 Los programas creados con FreeFem++ pueden usarse para resolver problemas en multifísica en 2D y 3D.
- 2 FreeFem++ es escrito en C++. Puede ser considerado como un lenguaje de programación en sí mismo.
- 3 Es multiplataforma, se puede instalar en sistemas operativos: MacOS, Windows, y GNU-Linux.
- 4 Para escribir los códigos, podemos utilizar cualquier editor de texto plano: vi, vim, nano, gedit, xed, notepad, etc.
- 5 En linux-mint/ubuntu, lo podemos instalar como: `sudo apt-get install FreeFem++`.
- 6 En general, lo podemos descargar desde el sitio oficial: <http://www.freefem.org>
- 7 Alternativamente, podemos descargar un entorno integrado para FreeFem++: [FreeFem++-cs](#)

Manos al código ... iniciemos con el cursillo!!!

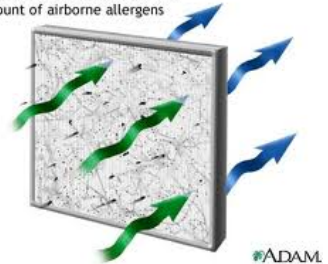
- 1 Motivación
- 2 Ecuación de Darcy: Formulación Primal
- 3 Ecuación de Darcy: Formulación Mixta
- 4 Ecuaciones de Brinkman–Forchheimer

Motivación

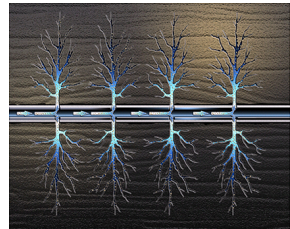
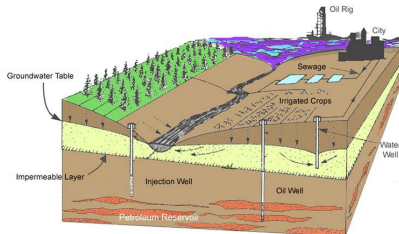
Algunos fenómenos físicos en mecánica de fluidos



A HEPA air filter can reduce the amount of airborne allergens



ADAM



Notaciones en 2D: $\Delta p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}$, $\nabla p = \left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y} \right)^t$, y $\operatorname{div}(\mathbf{u}) = \frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y}$

Ecuaciones de Darcy: presión p – velocidad \mathbf{u} y presión p

$$-\mathbf{D}^{-1} \Delta p = f \quad \text{en } \Omega \iff \begin{cases} \mathbf{D} \mathbf{u} + \nabla p = \mathbf{0} & \text{en } \Omega, \\ \operatorname{div}(\mathbf{u}) = f & \text{en } \Omega, \end{cases}, \quad p = 0 \quad \text{en } \Gamma.$$

Ecuaciones de Stokes: velocidad \mathbf{u} y presión p

$$-\nu \Delta \mathbf{u} + \nabla p = \mathbf{f} \quad \text{en } \Omega, \quad \operatorname{div}(\mathbf{u}) = 0 \quad \text{en } \Omega, \quad \mathbf{u} = \mathbf{0} \quad \text{en } \Gamma.$$

Ecuaciones de Brinkman–Forchheimer: velocidad \mathbf{u} y presión p

$$-\nu \Delta \mathbf{u} + \mathbf{D} \mathbf{u} + \mathbf{F} |\mathbf{u}| \mathbf{u} + \nabla p = \mathbf{f} \quad \text{en } \Omega, \quad \operatorname{div}(\mathbf{u}) = 0 \quad \text{en } \Omega, \quad \mathbf{u} = \mathbf{0} \quad \text{en } \Gamma.$$

Ecuación de Darcy:
Formulación Primal

Problema modelo

$$-D^{-1} \Delta p = f \quad \text{en } \Omega, \quad p = 0 \quad \text{en } \partial\Omega.$$

- p : presión
- D : número de Darcy (permeabilidad)
- $n = 2, 3$: dimensión del espacio
- $\Omega \subseteq \mathbb{R}^n$: dominio poligonal/poliedral
- $\partial\Omega$: frontera de Ω
- $f \in L^2(\Omega)$

Multiplicando por una función test apropiada e integrando por partes, deducimos que

$$-D^{-1} \int_{\Omega} \Delta p q = -D^{-1} \int_{\partial\Omega} \frac{\partial p}{\partial n} q ds + D^{-1} \int_{\Omega} \nabla p \cdot \nabla q = D^{-1} \int_{\Omega} \nabla p \cdot \nabla q.$$

Formulación variacional: Hallar $p \in H_0^1(\Omega)$ tal que

$$a(p, q) := D^{-1} \int_{\Omega} \nabla p \cdot \nabla q = \int_{\Omega} f q =: F(q) \quad \forall q \in H_0^1(\Omega).$$

Tiene solución?

Método de Elementos Finitos

Espacio de Elementos Finitos ($H_h \subset H^1(\Omega)$):

$$H_h := \left\{ q_h \in C(\bar{\Omega}) : q_h|_T \in \mathbb{P}_k(T) \quad \forall T \in \mathcal{T}_h, k \geq 1 \right\}$$

$$H_{h,0} := H_h \cap H_0^1(\Omega)$$

Problema discreto: Hallar $p_h \in H_{h,0}$ tal que

$$a(p_h, q_h) := \mathbf{D}^{-1} \int_{\Omega} \nabla p_h \cdot \nabla q_h = \int_{\Omega} f q_h =: F(q_h) \quad \forall q_h \in H_{h,0}.$$

P1: Tiene solución?

P2: Como calcular p_h ?

P3: $\|p - p_h\|_{1,\Omega} \leq C(p) h^k$,

$h = \max_{T \in \mathcal{T}_h} h_T$ y h_T la mayor longitud entre dos puntos del triángulo T

Fórmula de tasa de convergencia

- Dadas dos particiones/mallas del dominio de estudio Ω , con tamaños de mallas h_{n-1} y h_n , respectivamente, se tienen las estimaciones del error

$$\|p - p_{h_{n-1}}\|_{1,\Omega} \leq C_{n-1}(p) h_{n-1}^k \quad \text{y} \quad \|p - p_{h_n}\|_{1,\Omega} \leq C_n(p) h_n^k,$$

donde k es el orden de convergencia del método dependiendo de la norma con la que es medida el error y el grado polinomial utilizado para dicha aproximación.

- De lo anterior, deducimos que

$$\frac{\|p - p_{h_n}\|_{1,\Omega}}{\|p - p_{h_{n-1}}\|_{1,\Omega}} \approx \left(\frac{h_n}{h_{n-1}} \right)^k.$$

- Así, el orden de convergencia del método puede ser calculado como:

$$k \approx \log \left(\frac{\|p - p_{h_n}\|_{1,\Omega}}{\|p - p_{h_{n-1}}\|_{1,\Omega}} \right) / \log \left(\frac{h_n}{h_{n-1}} \right).$$

- Llevemos a FreeFem++ nuestro primer ejemplo ...

Ejemplo 2D: Tasa de Convergencia para Formulación Primal

Solución para ilustrar tasa de convergencia teórica

- $\Omega := (0, 1) \times (0, 1)$
- $D = 1$
- $p(x, y) = \cos(\pi x) \sin(\pi y)$
- $p_x(x, y) = -\pi \sin(\pi x) \sin(\pi y)$
- $p_y(x, y) = \pi \cos(\pi x) \cos(\pi y)$
- $f = -D^{-1} (p_{xx}(x, y) + p_{yy}(x, y))$

Código FreeFEM++: Darcy-primal_2D.edp I

```

1 //
2 // This code solves the 2D Darcy problem in primal formulation
3 // with nonhomogeneous Dirichlet boundary condition
4 // - (1/D)*\Delta p = f in \Omega, p = pD on \Gamma.
5 //
6 // Global information
7 load "iovtk"; // for saving data in paraview format
8 load "UMFPACK64"; // UMFPACK solver
9 load "Element_P3";
10 //-----
11 // Initial parameters
12 //-----
13 //----- Global parameters
14 int nref = 5;
15 real[int] H(nref); // mesh size
16 real[int] DOF(nref); // degrees of freedom
17 real[int] nbT(nref); // degrees of freedom
18
19 //----- error
20 real[int] Hlp(nref);
21
22 //----- rate of convergence
23 real[int] pHrate(nref-1);
24 //-----
25 // Global data
26 //-----
27 //--- Data RHS
28 func p = cos(pi*x)*sin(pi*y);
29 func px = -pi*sin(pi*x)*sin(pi*y);
30 func py = pi*cos(pi*x)*cos(pi*y);

```

Código FreeFEM++: Darcy-primal_2D.edp II

```

31 func pxx = -(pi^2)*cos(pi*x)*sin(pi*y);
32 func pyy = -(pi^2)*cos(pi*x)*sin(pi*y);
33
34 real D = 1.;
35 func f = -(1./D)*(pxx + pyy);
36
37 --- Macros
38 macro gp [px,py] //
39 macro grad(qh) [dx(qh),dy(qh)] //
40 -----
41 // Defining The Domain
42 -----
43 for(int n = 0; n < nref; n++){
44
45 int size = 2^(n + 2); // space discretization
46 int Gamma = 11;
47
48 border GammaD1(t=0,1){x=t; y=0; label = Gamma;};
49 border GammaD2(t=0,1){x=1; y=t; label = Gamma;};
50 border GammaD3(t=1,0){x=t; y=1; label = Gamma;};
51 border GammaD4(t=1,0){x=0; y=t; label = Gamma;};
52
53 mesh Th = buildmesh(GammaD1(size) + GammaD2(size) + GammaD3(size) + GammaD4(size));
54 //plot(Th,wait=true);
55 -----
56 // Finite element spaces
57 -----
58 fespace Hhp(Th,P1);
59

```

Código FreeFEM++: Darcy-primal_2D.edp III

```

60 fespace Vh(Th,P1); // discrete space to compute the meshsize
61 //-----
62 //                               Defining the bilinear forms and RHS
63 //-----
64 Hhp ph;
65 //----- bilinear forms
66 varf a(ph,qh) = int2d(Th) ( (1./D)*(grad(ph)'*grad(qh)) ) + on(Gamma,ph=p);
67
68 //----- RHS
69 varf rhs(ph,qh) = int2d(Th) ( f*qh ) + on(Gamma,ph=p);
70 //-----
71 //                               Building matrices and Load vector
72 //-----
73 matrix A = a(Hhp,Hhp);
74 //
75 real[int] RHS = rhs(0,Hhp);
76 //
77 set(A,solver = sparsesolver);
78
79 //----- calculating the solution
80 real[int] sol = A^-1*RHS;
81
82 //----- exporting data
83 ph[] = sol;
84
85 //----- calculating the errors
86 Hlp[n] = sqrt(int2d(Th) ( (p - ph)^2 + (gp - grad(ph))'*(gp - grad(ph)) ));
87
88 //----- for the meshsize in Omega

```

Código FreeFEM++: Darcy-primal_2D.edp IV

```

89 Vh h = hTriangle;
90 H[n] = h[] .max;
91 DOF[n] = Hhp.ndof;
92 nbT[n] = Th.nt;
93
94 //----- exporting to Paraview
95 savevtk("Data_Paraview_2D/Darcy-primal_aprox"+n+".vtk",Th,ph,dataname="ph");
96 savevtk("Data_Paraview_2D/Darcy-primal_exact"+n+".vtk",Th,p,dataname="p");
97 }
98 //-----
99 //                               showing the tables
100 //-----
101 cout << " p error in H1 = " << H1p <<endl;
102 for(int n=1; n < nref; n++)
103 pHirate[n-1] = log(H1p[n]/H1p[n-1]) / log(H[n]/H[n-1]);
104 cout << " convergence rate p in H1 = " << pHirate <<endl;
105
106 cout << " mesh size = " << H <<endl;
107 cout << " degrees of freedom = " << DOF <<endl;
108 cout << " number of Triangles = " << nbT <<endl;

```

Observaciones

- El grado polinomial se puede modificar en la opción: `fespace Hhp(Th,P2)`
- Código simple de migrar a 3D ... hagámoslo!!!

Ejemplo 3D: Tasa de Convergencia para Formulación Primal

Solución para ilustrar tasa de convergencia teórica

- $\Omega := (0, 1) \times (0, 1) \times (0, 1)$
- $D = 1$
- $p(x, y, z) = \cos(\pi x) \sin(\pi y) \exp(z)$
- $p_x(x, y, z) = -\pi \sin(\pi x) \sin(\pi y) \exp(z)$
- $p_y(x, y, z) = \pi \cos(\pi x) \cos(\pi y) \exp(z)$
- $p_z(x, y, z) = \cos(\pi x) \sin(\pi y) \exp(z)$
- $f = -D^{-1} (p_{xx}(x, y, z) + p_{yy}(x, y, z) + p_{zz}(x, y, z))$

Implementación Numérica en FreeFEM++: Darcy-primal_3D.edp I

```

1 //
2 // This code solves the 3D Darcy problem in primal formulation
3 // with nonhomogeneous Dirichlet boundary condition
4 // - (1/D)*\Delta p = f in \Omega, p = pD on \Gamma.
5 //
6 // Global information
7 load "msh3";
8 load "medit";
9 load "iovtk";
10 load "UMFPACK64";
11 include "cube.idp";
12 //-----
13 //                               Initial parameters
14 //-----
15 //----- Global parameters
16 int nref = 5;
17 real[int] H(nref); // mesh size
18 real[int] DOF(nref); // degrees of freedom
19 real[int] nbT(nref); // degrees of freedom
20
21 //----- error
22 real[int] Hlp(nref);
23
24 //----- rate of convergence
25 real[int] pHlrate(nref-1);
26 //-----
27 //                               Global data
28 //-----
29 //--- Data RHS

```

Implementación Numérica en FreeFEM++: Darcy-primal_3D.edp II

```

30 func p      = cos(pi*x)*sin(pi*y)*exp(z);
31 func px     = -pi*sin(pi*x)*sin(pi*y)*exp(z);
32 func py     = pi*cos(pi*x)*cos(pi*y)*exp(z);
33 func pz     = cos(pi*x)*sin(pi*y)*exp(z);
34 func pxx    = -(pi^2)*cos(pi*x)*sin(pi*y)*exp(z);
35 func pyy    = -(pi^2)*cos(pi*x)*sin(pi*y)*exp(z);
36 func pzz    = cos(pi*x)*sin(pi*y)*exp(z);
37
38 real D = 1.;
39 func f      = -(1./D)*(pxx + pyy + pzz);
40
41 //--- Macros
42 macro gp [px,py,pz] //
43 macro grad(qh) [dx(qh),dy(qh),dz(qh)] //
44 //-----
45 //                               Defining The Domain
46 //-----
47 for(int n = 0; n < nref; n++){
48
49 int size = n^2 + n + 2.; // space discretization
50 int Gamma = 11;
51
52 int[int] NN = [size,size,size]; // the number of step in each direction
53 real[int,int] BB = [[0,1],[0,1],[0,1]];
54 int[int,int] LL = [[Gamma,Gamma],[Gamma,Gamma],[Gamma,Gamma]]; // left,right,front, back, down
55 , top
56 mesh3 Th = Cube(NN,BB,LL);
57 //medit("cube",Th);

```


Implementación Numérica en FreeFEM++: Darcy-primal_3D.edp III

```

58 //-----
59 //          Finite element spaces
60 //-----
61 fespace Hhp(Th,P13d);
62
63 fespace Vh(Th,P13d); // discrete space to compute the meshsize
64 //-----
65 //          Defining the bilinear forms and RHS
66 //-----
67 Hhp ph;
68 //----- bilinear forms
69 varf a(ph,qh) = int3d(Th) ( (1./D)*(grad(ph)'*grad(qh)) ) + on(Gamma,ph=p);
70
71 //----- RHS
72 varf rhs(ph,qh) = int3d(Th) ( f*qh ) + on(Gamma,ph=p);
73 //-----
74 //          Building matrices and Load vector
75 //-----
76 matrix A = a(Hhp,Hhp);
77 //
78 real[int] RHS = rhs(0,Hhp);
79 //
80 set(A,solver = sparsesolver);
81
82 //----- calculating the solution
83 real[int] sol = A^-1*RHS;
84
85 //----- exporting data
86 ph[] = sol;

```

Implementación Numérica en FreeFEM++: Darcy-primal_3D.edp IV

```

87
88 //----- calculating the errors
89 H1p[n] = sqrt(int3d(Th) ( (p - ph)^2 + (gp - grad(ph))'*(gp - grad(ph)) ));
90
91 //----- for the meshsize in Omega
92 Vh h = hTriangle;
93 H[n] = h[].max;
94 DOF[n] = Hhp.ndof;
95 nbT[n] = Th.nt;
96
97 //----- exporting to Praraview
98 savevtk("Data_Paraview_3D/Darcy-primal_aprox"+n+".vtk", Th, ph, dataname="ph");
99 savevtk("Data_Paraview_3D/Darcy-primal_exact"+n+".vtk", Th, p, dataname="p");
100 }
101 //-----
102 //              showing the tables
103 //-----
104 cout << " p error in H1 = " << H1p << endl;
105 for(int n =1; n < nref; n++)
106 pH1rate[n-1] = log(H1p[n]/H1p[n-1]) / log(H[n]/H[n-1]);
107 cout << " convergence rate p in H1 = " << pH1rate << endl;
108
109 cout << " mesh size = " << H << endl;
110 cout << " degrees of freedom = " << DOF << endl;
111 cout << " number of Tetrahedra = " << nbT << endl;

```

Ecuación de Darcy:
Formulación Mixta

Formulación Mixta

Problema de Darcy en formulación mixta

$$\mathbf{u} = -\mathbf{D}^{-1} \nabla p \quad \text{en } \Omega, \quad \operatorname{div}(\mathbf{u}) = f \quad \text{en } \Omega, \quad p = p_D \quad \text{en } \Gamma$$

$$\mathbf{H}(\operatorname{div}; \Omega) := \left\{ \mathbf{v} = (v_1, v_2) \in \mathbf{L}^2(\Omega) : \operatorname{div}(\mathbf{v}) \in L^2(\Omega) \right\}, \quad \text{con } \operatorname{div}(\mathbf{v}) = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y}.$$

Formulación variacional

Hallar $(\mathbf{u}, p) \in \mathbf{H}(\operatorname{div}; \Omega) \times L^2(\Omega)$ tal que:

$$\begin{aligned} \mathbf{D} \int_{\Omega} \mathbf{u} \cdot \mathbf{v} - \int_{\Omega} p \operatorname{div}(\mathbf{v}) &= -\langle \mathbf{v} \cdot \mathbf{n}, p_D \rangle_{\Gamma} \quad \forall \mathbf{v} \in \mathbf{H}(\operatorname{div}; \Omega), \\ - \int_{\Omega} q \operatorname{div}(\mathbf{u}) &= - \int_{\Omega} f q \quad \forall q \in L^2(\Omega) \end{aligned}$$



G.N. GATICA, *A Simple Introduction to the Mixed Finite Element Method. Theory and Applications*. Springer Briefs in Mathematics. Springer, Cham, 2014.

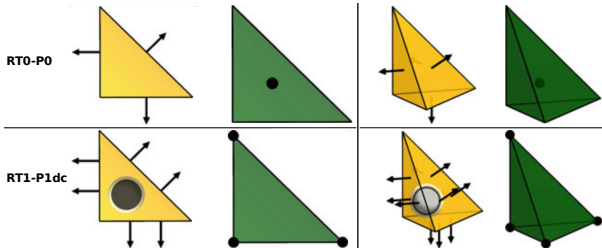
Formulación variacional discreta

Hallar $(\mathbf{u}_h, p_h) \in \mathbf{H}_h^u \times H_h^p$ tal que:

$$\begin{aligned} \mathcal{D} \int_{\Omega} \mathbf{u}_h \cdot \mathbf{v}_h - \int_{\Omega} p_h \operatorname{div}(\mathbf{v}_h) &= -\langle \mathbf{v}_h \cdot \mathbf{n}, p_D \rangle_{\Gamma} \quad \forall \mathbf{v}_h \in \mathbf{H}_h^u, \\ - \int_{\Omega} q_h \operatorname{div}(\mathbf{u}_h) &= - \int_{\Omega} f q_h \quad \forall q_h \in H_h^p. \end{aligned}$$

$$\mathbf{H}_h^u := \left\{ \mathbf{v}_h \in \mathbf{H}(\operatorname{div}; \Omega) : \quad \mathbf{v}_h|_T \in \mathbf{RT}_k(T) \quad \forall T \in \mathcal{T}_h \right\},$$

$$H_h^p := \left\{ q_h \in L^2(\Omega) : \quad q_h|_T \in P_k(T) \quad \forall T \in \mathcal{T}_h \right\}.$$



Tasas de convergencia

P1: Tiene solución?

P2: Como calcular \mathbf{u}_h, p_h ?

P3: $\|\mathbf{u} - \mathbf{u}_h\|_{\text{div};\Omega} + \|p - p_h\|_{0,\Omega} \leq C(\mathbf{u}, p) h^{k+1}$, con $k \geq 0$

Manos al código ... iniciemos con nuestro tercer programa ...

Ejemplo 2D: Tasa de Convergencia para Formulación Mixta

Solución para ilustrar tasa de convergencia teórica

- $\Omega := (0, 1) \times (0, 1)$
- $D = 1$
- $p(x, y) = \cos(\pi x) \sin(\pi y)$
- $p_x(x, y) = -\pi \sin(\pi x) \sin(\pi y)$
- $p_y(x, y) = \pi \cos(\pi x) \cos(\pi y)$
- $\mathbf{u}(x, y) = -D^{-1} (p_x(x, y), p_y(x, y))^t$
- $f = -D^{-1} (p_{xx}(x, y) + p_{yy}(x, y))$

Código FreeFEM++: Darcy-mixto_2D.edp I

```

1 //
2 // This code solves the 2D Darcy problem in mixed formulation with
3 // nonhomogeneous Dirichlet boundary conditions
4 //       $u = -(1/D) * \nabla p \cdot q$  in  $\Omega$ ,  $\text{div}(u) = f$  in  $\Omega$ ,
5 //       $p = p_D$  on  $\Gamma$ .
6 //
7 // Global information
8 load "iovtk";           // for saving data in paraview format
9 load "UMFPACK64";       // UMFPACK solver
10 load "Element_Mixte";
11 //-----
12 //              Initial parameters
13 //-----
14 //----- Global parameters
15 int nref = 5;
16 real[int] H(nref);      // mesh size
17 real[int] DOF(nref);    // degrees of freedom
18 real[int] nbT(nref);    // degrees of freedom
19
20 //----- errors
21 real[int] uerror(nref);
22 real[int] perror(nref);
23
24 //----- rate of convergence
25 real[int] urate(nref-1);
26 real[int] prate(nref-1);
27 //-----
28 //              Global data
29 //-----

```


Código FreeFEM++: Darcy-mixto_2D.edp II

```

30 !--- Data RHS
31 func p   = cos(pi*x)*sin(pi*y);
32 func px  = -pi*sin(pi*x)*sin(pi*y);
33 func py  = pi*cos(pi*x)*cos(pi*y);
34 func pxx = -(pi^2)*cos(pi*x)*sin(pi*y);
35 func pyy = -(pi^2)*cos(pi*x)*sin(pi*y);
36 //
37 real D = 1.;
38 func f   = -(1./D)*(pxx + pyy);
39 //
40 !----- Macros
41 macro u [- (1./D)*px, - (1./D)*py] //
42
43 macro uh [uh1,uh2] //
44 macro vh [vh1,vh2] //
45
46 macro norm [N.x,N.y] //
47 macro div(vh) ( dx(vh[0]) + dy(vh[1]) ) //
48 !-----
49 //
50 Defining The Domain
51 !-----
52 for(int n = 0; n < nref; n++){
53 //
54 int size = 2^(n + 2); // space discretization
55 int Gamma = 11;
56
56 border GammaD1(t=0,1){x=t; y=0; label = Gamma;};
57 border GammaD2(t=0,1){x=1; y=t; label = Gamma;};
58 border GammaD3(t=1,0){x=t; y=1; label = Gamma;};

```

Código FreeFEM++: Darcy-mixto_2D.edp III

```

59 border GammaD4(t=1,0){x=0; y=t; label = Gamma;};
60
61 mesh Th = buildmesh(GammaD1(size) + GammaD2(size) + GammaD3(size) + GammaD4(size));
62 //-----
63 //          Finite element spaces
64 //-----
65 fespace Hhu(Th,RT0);
66 fespace Hhp(Th,P0);
67
68 fespace Vh(Th,P1); // discrete space to compute the meshsize
69 //-----
70 //          Defining the bilinear forms
71 //-----
72 Hhu uh; Hhp ph;
73 //----- bilinear forms
74 varf a(uh,vh) = int2d(Th) ( D*(uh'*vh) );
75 varf b([ph],vh) = int2d(Th) ( -(ph*div(vh)) );
76
77 //----- RHS
78 varf rhs1(uh,vh) = int1d(Th,Gamma) ( -p*(vh'*norm) );
79 varf rhs2(ph,qh) = int2d(Th) ( -(f*qh) );
80 //-----
81 //          Building matrices
82 //-----
83 matrix A = a(Hhu,Hhu);
84 matrix B = b(Hhp,Hhu);
85
86 matrix M;{
87 M = [[ A, B],

```

Código FreeFEM++: Darcy-mixto_2D.edp IV

```

88     [ B', 0] ]};
89 //-----
90 //                               Load vector
91 //-----
92 real[int] RHS1 = rhs1(0,Hhu);
93 real[int] RHS2 = rhs2(0,Hhp);
94
95 real[int] L = [RHS1, RHS2];
96
97 set(M,solver = sparsesolver);
98
99 //----- calculating the solution
100 real[int] sol = M^-1*L;
101
102 //----- exporting data
103 uh1[] = sol(0:Hhu.ndof - 1);
104 ph[] = sol(Hhu.ndof:Hhu.ndof + Hhp.ndof - 1);
105
106 //----- calculating the errors
107 uerror[n] = sqrt(int2d(Th) ( (u - uh)'*(u - uh) + (f - div(uh))^2 ));
108 perror[n] = sqrt(int2d(Th) ( (p - ph)^2 ));
109
110 //----- for the meshsize in Omega
111 Vh h = hTriangle;
112 H[n] = h[].max;
113 DOF[n] = Hhu.ndof + Hhp.ndof;
114 nbT[n] = Th.nt;
115
116 //----- exporting to Praraview

```

Código FreeFEM++: Darcy-mixto_2D.edp V

```

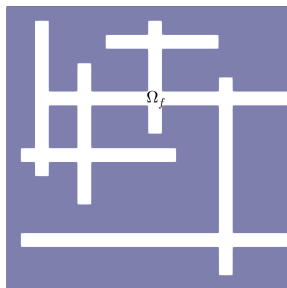
117 savevtk("Data_Paraview_2D/Darcy-mixto_aprox"+n+".vtk",Th,[uh1,uh2,0],ph,dataname="uh ph");
118 savevtk("Data_Paraview_2D/Darcy-mixto_exact"+n+".vtk",Th,[px,py,0],p,dataname="u p");
119 }
120 //-----
121 //                               showing the tables
122 //-----
123 cout << " u error in Hdiv = " << uerror <<endl;
124 for(int n = 1; n < nref; n++)
125   urate[n-1] = log(uerror[n]/uerror[n-1]) / log(H[n]/H[n-1]);
126 cout << " convergence rate u in Hdiv = " << urate <<endl;
127
128 cout << " p error in L2 = " << perror <<endl;
129 for(int n =1; n < nref; n++)
130   prate[n-1] = log(perror[n]/perror[n-1]) / log(H[n]/H[n-1]);
131 cout << " convergence rate p in L2 = " << prate <<endl;
132
133 cout << " mesh size = " << H <<endl;
134 cout << " degrees of freedom = " << DOF <<endl;
135 cout << " number of Triangles = " << nbT <<endl;

```

Fluido en un medio poroso 2D con fracturas

Parámetros

- $\Omega = (-1, 1)^2$
- $D = \begin{cases} 10 & \text{en } \bar{\Omega} \setminus \Omega_f \\ 1 & \text{en } \Omega_f \end{cases}$



$$p = \begin{cases} -0.5(y - 1) & \text{en } \Gamma_{\text{left}}, \\ -0.5(x - 1) & \text{en } \Gamma_{\text{bottom}}, \end{cases}$$

$$p = 0 \quad \text{en } \Gamma_{\text{right}} \cup \Gamma_{\text{top}},$$

Figure: Izquierda: dominio computacional. Derecha: condiciones de contorno.

Código FreeFEM++: Darcy-mixto-fracture_2D.edp I

```

1 //
2 // This code solves the Darcy problem in mixed formulation with
3 // nonhomogeneous Dirichlet boundary conditions in a fracture domain
4 //       $u = -(1/D) * \nabla p$  in  $\Omega$ ,  $\text{div}(u) = f$  in  $\Omega$ ,
5 //       $p = p_D$  on  $\Gamma$ .
6 //
7 // Global information
8 load "iovtk";           // for saving data in paraview format
9 load "UMFPACK64";       // UMFPACK solver
10 load "Element_Mixte";
11 //-----
12 //                               Global data
13 //-----
14 //----- Global parameters
15 real H, DOF, nbT;
16
17 //--- Data RHS
18 real D1 = 10;
19 real D2 = 1.;
20
21 func pLeft = -0.5*(y-1.);
22 func pBottom = -0.5*(x-1.);
23 func f = 0.;
24
25 //----- Macros
26 macro uh [uh1,uh2] //
27 macro vh [vh1,vh2] //
28
29 macro norm [N.x,N.y] //

```

Código FreeFEM++: Darcy-mixto-fracture_2D.edp II

```

30 macro div(vh) ( dx(vh[0]) + dy(vh[1]) ) //
31 //-----
32 //                               Defining The Domain
33 //-----
34 mesh Th = readmesh("Fracture_network-mesh.msh");
35 // Labels setting:
36 // 33: region outside the fracture
37 // 34: region inside the fracture
38 // 1: bottom boundary
39 // 22: right and top boundaries
40 // 4: left boundary
41 //-----
42 //                               Finite element spaces
43 //-----
44 fespace Hhu(Th,RT0);
45 fespace Hhp(Th,P0);
46
47 fespace Vh(Th,P1); // discrete space to compute the meshsize
48 //-----
49 //                               Defining the bilinear forms
50 //-----
51 Hhu uh; Hhp ph;
52 //----- bilinear forms
53 varf a(uh,vh) = int2d(Th,33) ( D1*(uh'*vh) ) + int2d(Th,34) ( D2*(uh'*vh) );
54 varf b([ph],vh) = int2d(Th) ( -(ph*div(vh)) );
55
56 //----- RHS
57 varf rhs1(uh,vh) = int1d(Th,1) ( -pBottom*(vh'*norm) ) + int1d(Th,4) ( -pLeft*(vh'*norm) );
58 varf rhs2(ph,qh) = int2d(Th) ( -(f*qh) );

```

Código FreeFEM++: Darcy-mixto-fracture_2D.edp III

```

59 //-----
60 //                               Building matrices
61 //-----
62 matrix A = a(Hhu,Hhu);
63 matrix B = b(Hhp,Hhu);
64
65 matrix M;{
66 M = [[ A, B],
67      [ B', 0]];}
68 //-----
69 //                               Load vector
70 //-----
71 real[int] RHS1 = rhs1(0,Hhu);
72 real[int] RHS2 = rhs2(0,Hhp);
73
74 real[int] L = [RHS1, RHS2];
75
76 set(M,solver = sparsesolver);
77
78 //----- calculating the solution
79 real[int] sol = M^-1*L;
80
81 //----- exporting data
82 uh1[] = sol(0:Hhu.ndof - 1);
83 ph[] = sol(Hhu.ndof:Hhu.ndof + Hhp.ndof - 1);
84
85 //----- for the meshsize in Omega
86 Vh h = hTriangle;
87 H = h[] .max;

```


Código FreeFEM++: Darcy-mixto-fracture_2D.edp IV

```

88 DOF = Hhu.ndof + Hhp.ndof;
89 nbT = Th.nt;
90
91 //----- exporting to Paraview
92 savevtk("Data_Paraview_2D/Darcy-mixto-fracture_aprox.vtk",Th,[uh1,uh2,0],ph,dataname="uh ph");
93 //-----
94 //              showing the tables
95 //-----
96 cout << " mesh size = " << H <<endl;
97 cout << " degrees of freedom = " << DOF <<endl;
98 cout << " number of Triangles = " << nbT <<endl;

```

Observaciones

- Podemos cargar mallas creadas con malladores externos, como Gmsh o Triangle, siempre y cuando tengan el formato msh soportado por FreeFEM++
- Como este ejemplo no tiene solución analítica, realizamos un estudio cualitativo de los resultados obtenidos. Usemos ParaView!!!

ParaView como una herramienta para crear imágenes de alta calidad

<https://www.paraview.org>

