Laurea magistrale
in ingegneria matematica

Progetto per il corso di
Programmazione Avanzata per il Calcolo Scientifico.

Implementazione in LifeV dell'algoritmo di Riduzione Gerarchica di Modello

Progetto svolto da:
Matteo Carlo Maria Aletti
Matr. 783045
Andrea Bortolossi
Matr. 783023

Anno Accademico 2012–2013

Indice

Capitolo 1

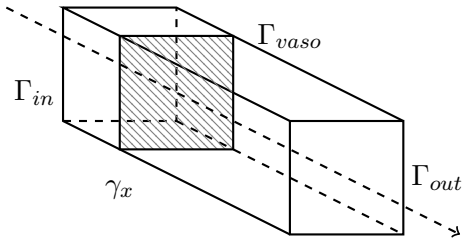
Introduzione

1.1 Nozioni base

L'obbiettivo primale del progetto è stato di implementare in LifeV un risolutore ADR 3D, basato sulla tecnica di Riduzione Gerarchica di Modello. Il problema trattato è il seguente:

$$\begin{cases} -\mu\Delta u + \mathbf{b}\cdot\nabla u + \sigma u = f & \text{in } \Omega \\ u = u_{in} & \text{su } \Gamma_{in} \\ \frac{\partial u}{\partial \mathbf{n}} = 0 & \text{su } \Gamma_{out} \\ u = 0 & \text{su } \Gamma_{vaso} \end{cases} \tag{1.1}$$

$$\Omega = \bigcup_{x\in\Omega_{1D}} \gamma_x \tag{1.2}$$



Si consideri il dominio Ω, come l'unione di slice poste trasversalmente alla direzione longitudinale del tubo a sezione rettangolare, la quale verrà indicata d'ora in poi con Ω_{1D}:

Lungo le slice γ_x vengono utilizzate funzioni spaziali differenti rispetto a quelle utilizzate lungo Ω_{1D}. Si consideri infatti per Ω_{1D}, lo spazio funzionale $V_{1D} = H^1_{\Gamma_{in}}(\Omega_{1D})$, mentre sulla generica γ_x si introducano le basi modali $\{\varphi_k(y,z)\}$ ortonormali in $L^2(\gamma_x)$, con $k \in \mathbb{N}$. Quest'ultime definiscono su γ_x

lo spazio funzionale $V_{\gamma_x} := span\{\varphi_k\}$. Definiamo ora il sottospazio generato solo dai primi m modi ovvero $V_{\gamma_x}^m := span\{\varphi_1, ..., \varphi_m\}$ e combiniamolo con V_{1D}, ottenendo il seguente spazio ridotto:

$$V_m := \left\{ v_m(x,y,z) = \sum_{k=1}^{m} \varphi_k(y,z)\tilde{v}_k(x), \ \ con \ \ \tilde{v}_k \in V_{1D} \right\} \tag{1.3}$$

L'ortogonalitá in $L^2(\gamma_x)$ implica che i coefficienti $\tilde{v}_k$ in (1.3) sono il risultato del seguente prodotto scalare per $k = 1, ..., m$:

$$\tilde{v}_k(x) = \int_{\gamma_x} \varphi_k(y,z) v_m(x,y,z)\, dydz$$

La convergenza di una soluzione u_m tale che soddisfi il problema (1.1) è garantita osservando che:

- $V_m \subset V \ \forall m \in \mathbb{N}$, ossia che lo spazio ridotto V_m è conforme in V;
- $\lim\limits_{x \to +\infty} \left(\inf\limits_{v_m \in V_m} || \, v - v_m \, || \right) = 0$ per ogni $v \in V$, ossia che vale la proprietà di approssimazione di V_m rispetto a V;

È possibile dimostrare che le ipotesi di conformità e approssimazione sono ancora valide in una trattazione con dato di Dirichlet non omogeneo sulle pareti del tubo ([**?**]).

1.2 Forma matriciale

La risoluzione del problema ADR può avvenire quindi sullo spazio ridotto V_m. Dunque, per ogni $m \in \mathbb{N}$ si riconosca il seguente problema ridotto del problema originale (1.1), trovare $u_m \in V_m$ tale che $\forall v_m \in V_m$:

$$\int_\Omega (\mu \nabla u_m \nabla v_m + \mathbf{b} \nabla u_m v_m + \sigma u_m v_m)\, d\Omega = \int_\Omega f v \, dxdy \quad (1.4)$$

Si adoperi l'espansione tramite i coefficienti di Fourier della $u_m(x,y,z) = \sum_{j=k}^{m} \tilde{u}_j(x)\varphi_j(y,z)$ dove:

$$\tilde{u}_j(x) = \int_{\gamma(x)} u_m(x,y,z)\varphi_j(y,z)\, dydz$$

e si considerino le funzioni test $v_m = \vartheta(x)\varphi_k(y,z)$ con $\vartheta(x) \in V_{1D}$ e $k = 1, ...m$. Il problema assume la seguente forma:

$$\begin{aligned}
&\sum_{j=1}^{m} \Bigg[\int_\Omega \mu \nabla(\tilde{u}_j(x)\varphi_j(y,z)) \nabla(\vartheta(x)\varphi_k(y,z))\, dxdydz \\
&+ \int_\Omega \mathbf{b} \nabla(\tilde{u}_j(x)\varphi_j(y,z)\vartheta(x)\varphi_k(y,z)\, dxdydz \\
&+ \int_\Omega \sigma \tilde{u}_j(x)\varphi_j(y,z)\vartheta(x)\varphi_k(y,z)\, dxdydz \Bigg] \\
&= \int_\Omega f\vartheta(x)\varphi_k(y,z)\, dxdydz
\end{aligned} \quad (1.5)$$

Svolgendo l'operatore gradiente si ottiene:

$$\begin{aligned}
&\sum_{j=1}^{m} \Bigg[\int_\Omega \mu(\partial_x \tilde{u}_j \partial_x \vartheta \varphi_j \varphi_k + \tilde{u}_j \vartheta \partial_y \varphi_j \partial_y \varphi_k + \tilde{u}_j \vartheta \partial_z \varphi_j \partial_z \varphi_k)\, dxdydz \\
&+ \int_\Omega (b_1 \partial_x \tilde{u}_j \varphi_j + b_2 \tilde{u}_j \partial_y \varphi_j + b_3 \tilde{u}_j \partial_z \varphi_j)\vartheta\varphi_k\, dxdydz \\
&+ \int_\Omega \sigma \tilde{u}_j \vartheta \varphi_j \varphi_k\, dxdydz \Bigg] \\
&= \int_\Omega f\vartheta\varphi_k\, dxdydz
\end{aligned} \quad (1.6)$$

Definito N il numero di nodi scelti uniformemente distribuiti lungo Ω_{1D}, si determina una partizione T_h, dove $h = |\Omega_{1D}|/(N-1)$ è il passo spaziale. Introduciamo lo spazio agli elementi finiti lungo Ω_{1D} definito come segue

$$X_h^r = \left\{\psi_h \in C^0(\Omega_{1D}) : \psi_h|_K \in \mathbb{P}_r, \forall K \in T_h\right\}$$

Nella successiva implementazione del metodo si è considerato per semplicità, una base F.E.M. di primo grado. Possiamo quindi esprimere i coefficienti di Fuorier nel seguente modo: $\tilde{u}_j(x) = \sum_{s=1}^{N} u_{js}\psi_s(x)$.

Si ottiene dunque la formulazione matriciale del nostro problema, trovare $\mathbf{u} \in \mathbb{R}^{N*m}$ tale che $\forall\psi_l$ e $\forall\varphi_k$, con $l = 1, ...N$ e $k = 1, ...m$ si ha che:

$$\begin{aligned}\sum_{j=1}^{m}\sum_{s=1}^{N} u_{js}\Bigg[&\int_\Omega \mu(\partial_x\psi_s\partial_x\psi_l\varphi_j\varphi_k + \psi_s\psi_l\partial_y\varphi_j\partial_y\varphi_k + \psi_s\psi_l\partial_z\varphi_j\partial_z\varphi_k)\,dxdydz \\ &+ \int_\Omega (b_1\partial_x\psi_s\varphi_j + b_2\psi_s\partial_y\varphi_j + b_3\psi_s\partial_z\varphi_j)\psi_l\varphi_k\,dxdydz \\ &+ \int_\Omega \sigma\psi_s\psi_l\varphi_j\varphi_k\,dxdydz\Bigg] \\ &= \int_\Omega f\psi_l\varphi_k\,dxdydz \quad (1.7)\end{aligned}$$

Si osservi che il doppio indice $''js''$, in realtà scorre un vettore, la rimappatura in un solo indice può facilmente essere dedotta ottenenedo che $[\mathbf{u}]_{js} = \mathbf{u}[(j-1)N + s]$. La matrice generata ha quindi dimensioni $(mN)^2$, tuttavia fissata la frequenza delle soluzione e della funzione test è possibile identificare un blocco che corrisponde ad un problema monodimensionale. Se utilizziamo, in direzione x, gli elementi finiti di grado 1, il blocco risulta tridiagonale e, in questo caso, la matrice ha un numero di elementi non zero pari a $m^2(3N-2)$. Il pattern di sparsità per un caso con m=3 e N=14 è riportato in figura **??**. La matrice dei coefficienti è dunque sparsa ed inoltre il pattern è noto a priori, queste informazioni hanno permesso un assemblaggio più veloce in sede implementativa.

In generale il problema che si porrebbe ora sarebbe la scelta della base modale. Esistono svariati metodi al fine di determinare la natura della base modale, tuttavia questa problematica va al di fuori degli scopi di questo elaborato. Seguendo le linee guida in (e qua ci autocitiamo!!!!) scegliamo la base modale in grado di garantire le condizioni di parete:

$$\varphi_j(y,z) = sin\left(\frac{\alpha}{\pi L_y}y\right) sin\left(\frac{\beta}{\pi L_z}z\right) \quad \lambda_j = \alpha^2 + \beta^2 \qquad (1.8)$$

1.3 Implementazione integrali

Nel caso i coefficienti del problema ADR siano dipendenti dalla sola coordinata x o risultino fattorizabili lungo la direzione x e il piano ortogonale, il

risultato finale di HiMod è la trasformazione di un problema ADR full 3D a m^2 problemi ADR 1D accoppiati con coefficienti modificati opportunatamente dalle funzioni modali a seconda della coppia di frequenze considerata. Nel caso non ricadiamo in tale ipotesi vale comunque la scomposizione in problemi 1D ma risulta più delicata l'integrazione. Nel caso si fattorizza anche μ proiettandola sulle basi modali, non penso che dia dei buoni risultati, tuttavia è fattibile

$$\begin{array}{ll} \partial_x\psi_s\partial_x\psi_l & \int_{\gamma_x} \mu\varphi_j\varphi_k \, dydz \\ \partial_x\psi_s\psi_l & \int_{\gamma_x} \varphi_j\varphi_k \, dydz \\ \psi_s\psi_l & \int_{\gamma_x} (\mu\partial_y\varphi_j\partial_y\varphi_k + \mu\partial_z\varphi_j\partial_z\varphi_k + b_2\partial_y\varphi_j\varphi_k + b_3\partial_z\varphi_j\varphi_k + \sigma\varphi_j\varphi_k) \, dydz \end{array} \tag{1.9}$$

Capitolo 2

Descrizione classi

2.1 Modalspace

Modalspace é una classe base da cui derivano le figlie di modalspace che nascono dal momento che tipo di condizioni al bordo di parete vengono scelte.

Perché si é scelto questo tipo di ereditá? Gli autovalori generati e le rispettive basi sono proprietá esclusiva del tipo di p roblema ai limiti che si intende istanziare. Dunque le classi figlie si preoccupano di calcolare l'appropriata successione di autovalori e inoltre registrano i valori nodali delle funzioni di base sulla griglia di quadratura scelta. I dati vengono raccolti nei membri:

- Eigenvalues
- Mphi
- Mdphi

Fissandoci sulla slice di riferimento $[0, 1] \times [0, 1]$, si osserva che l'unico parametro che occorre ad una figlia per generare i materiale elencato precedentemente la specifica delle condizioni di bordo anche mtot!!.

Mi sta venendo un dubbio, il discorso della factory mi aveva intrigato, tuttavia mi stavo chiedendo, ma noi cosa registriamo di preciso?Un oggetto modalspace DDDD a cui peró vanno specificati mtot, gli autovalori e i valori delle basi?Dovrei registrare modalspaceDDDD per ogni possibile valore di mtot!!!Forse mi sfugge qualcosa....

Modalspace raccoglie gli elementi e le operazioni comuni a tutti i suoi figli tra questi vi sono sicuramente:

- Dimensioni della slice
- Numero modi utilizzato

- Regola di quadratura
- Calcolo coefficienti Fourier g(y,z) - vettore [g1 g2 ...]
- Calcolo del coefficiente Fuourier di g(x,y,z) - reale (g(x))

In realtá Eigenvalues appartiene ancora a Modalspace sará il caso di sposatarlo?Fare attenzione a findmyzmax() che definito in Modalspace.cpp, lo teniamo perché le funzioni comuni fanno usa di tale membro?Dovremmo costruire il getter nelle classi figlie secondo me sarebbe piú coerente

Stabiliti infatti i valori nodali delle funzioni di base e delle loro derivate tutte le basi modali istruite possono essere trattate alla stessa maniera. Sono comuni infatti i seguenti metodi (con j si intende l'indice legato alla soluzione e con k quello legato alla funzione test):

- ComputePhiPhi $\int_{\gamma_x} \varphi_j(y,z)\varphi_k(y,x)\,dydz$
- ComputeDyPhiPhi $\int_{\gamma_x} \partial_y\varphi_j(y,z)\varphi_k(y,x)\,dydz$
- ComputeDzPhiPhi $\int_{\gamma_x} \partial_z\varphi_j(y,z)\varphi_k(y,x)\,dydz$
- ComputeDyPhiDyPhi $\int_{\gamma_x} \partial_y\varphi_j(y,z)\partial_y\varphi_k(y,x)\,dydz$
- ComputeDzPhiDzPhi $\int_{\gamma_x} \partial_z\varphi_j(y,z)\partial_z\varphi_k(y,x)\,dydz$
- ComputePhi $\int_{\gamma_x} \varphi_k(y,x)\,dydz$

L'elemento principale di questa classe é ConstructModalBasis. In realtá tale metodo non é altro che un wrappers che facilita l'istanziamento della classe da parte dell'utente. Infatti dietro tale metodo si nasconde una chiamata alla factory dove sono schedati i possibili figli di ModalSpace. Qua possiamo aggiungere il codice di ConstructModalBasis cosí si capisce cosa nascondiamo.

Mi sono convinto che l'approccio ereditarietá é quello migliore, tuttavia dovremmo vedere il reale utilizzo che facciamo di questa factory

2.2 HiModAssembler HiModView

Sono momentaneamente separate ma é chiaro che devono appartenere alla stessa classe, ovvero a quella che sintetizzerá insieme Modalspace e FESpace 1D (ovvero quello costruito sulla fibra di supporto).

HiModView e HiModAssembler devono continuamente lavorare con gli elementi di Modalspace e FESpace 1D non sarebbe il caso di instaurare un legame piú intimo?Magari specificando l'amicizia di HiMod con Modalspace e FESpace?Troppo incasinato?Si velocizza il tutto (non occorre infatti passare dai noiosi getters)?

Gli unici membri di HiModAssembler e HiModView sono fespace e modalbasis (in HiModAssembler c'é anche etfespace ma direi che dobbiamo toglierlo e non farlo creare nel main, quella é sicuramente una questione interna di come abbiamo voluto implementare il calcolo dei coefficienti della matrice, inoltre aumentiamo in leggibilitá).

Ecco le uilitá di HiModAssembler:

- AddADRProblem
- interpolate
- Addrhs (costante e functionType)
- Addrhsfunctor
- AddDirichletBCIn (Momentanemente via penalizzazione)

Ecco invece le utilitá di HiModView:

- funCoeff3D (genera i valori nodali su una griglia partendo dal vettore soluzione o da una funzione)
- normL2 (dato il vettore che sputa fuori funCoeff3D ne fa la normaL2)
- ConvergeFile (crea un file di output gestibile tramite getpot, utilizzo limitato al testconvergence)