POLITECNICO DI MILANO

Scuola di Ingegneria dei Sistemi

Corso di Studi di Ingegneria Matematica
Orientamento Calcolo Scientifico

Approssimazione numerica di ODE con forzante discontinua per flussi reattivi in mezzi porosi

Relatore: Prof. Luca Formaggia
Correlatore: Dott. Anna Scotti

Massimiliano Beccaria
Matr.754742

ANNO ACCADEMICO 2011-2012

Sommario

Il presente lavoro di tesi si concentra sugli aspetti teorici e numerici della risoluzione di equazioni differenziali ordinarie con forzanti discontinue (ODE-DRH). In particolare le tecniche risolutive proposte nei lavori di Filippov e di Dieci e Lopez sono state implementate all'interno di `LifeV`, libreria di calcolo parallelo ad elementi finiti scritta in C++. Sono state studiate anche tecniche di operator splitting sequenziali ponendo particolare attenzione al problema della conservazione dell'ordine di convergenza degli schemi usati: si sono indagate le criticità legate alle condizioni al bordo dei sottoproblemi derivati dallo splitting e all'utilizzo di metodi multistep. Tali operator splitting sono stati utilizzati per risolvere numericamente equazioni di diffusione avvezione e reazione, quest'ultima non lineare e discontinua, separando i contributi dei vari operatori e sfruttando le tecniche risolutive delle ODE-DRH. In questo contesto si presenta un modello di dissoluzione e precipitazione accoppiato con un modello di Darcy che descrive il processo di formazione e distruzione di cristalli in un mezzo poroso.

Abstract

In the present Master's thesis we discuss some theoretical and numerical aspects concerning the solution of differential equations with discontinuous right-hand sides (ODE-DRH). In particular we implemented the solution strategy proposed in the work of Filippov and Dieci-Lopez in `LifeV`, a C++ parallel computing FEM library. Sequential operator splitting techniques have been studied with a particular focus on the order of convergence: we studied the criticalities related to the imposition of intermediate boundary conditions to the subproblems deriving from the operator splitting and to the use of multistep time integration schemes. The operator splitting was used in the numerical approximation of advection reaction diffusion equations with nonlinear and discontinuous reaction terms. The nonlinear and discontinuous reaction term was separated from the advection-diffusion part of the equation and solved exploiting the suitable techniques for ODE-DRH. In this context we present a model for dissolution and precipitation coupled with a Darcy model that describes the process of production and destruction of crystals in a porous medium.

Indice

Elenco delle figure

Introduzione

Un fenomeno discontinuo è caratterizzato da una variazione brusca, nello spazio o nel tempo, di una grandezza fisica. Questo lavoro si concentra su sistemi di equazioni differenziali ordinarie con forzanti discontinue (identificabili anche con la sigla ODE-DRH). Sistemi di questo tipo sono responsabili della descrizione di fenomeni di svariata natura: si incontrano per esempio nel contesto dei problemi di controllo, dei sistemi biologici e meccanici ed in ambito geologico. In questo lavoro siamo interessati in particolare a problemi di flussi reattivi in ambito geologico, ossia a problemi in cui il flusso di acqua nella porosità delle rocce è responsabile del trasporto di specie chimiche che possono reagire tra loro. Lo studio dei flussi reattivi nel sottosuolo è rilevante per molte applicazioni pratiche, dalla formazione e migrazione di idrocarburi alla simulazione di processi di contaminazione e bonifica, fino allo studio di processi diagenetici. In alcuni casi le reazioni chimiche possono presentare un comportamento discontinuo ed essere quindi descritte da sistemi di equazioni ODE-DRH. Questo comportamento si verifica ad esempio nel processo di generazione degli idrocarburi in cui intervengono fenomeni di ritenzione delle specie prodotte descritti da soglie [2] che introducono una discontinuità nel termine forzante delle equazioni. Un altro caso particolarmente rilevante che sarà considerato in questa tesi è il processo di precipitazione responsabile della formazione di cristalli. La formazione o distruzione di grani solidi dipende dal bilancio di precipitazione e dissoluzione e quest'ultima può essere convenientemente descritta [17] da un termine discontinuo e richiede quindi tecniche numeriche adeguate.

Data la vastità del contesto applicativo dei sistemi di ODE a forzante discontinua è di grande interesse sviluppare tecniche e strumenti di approssimazione numerica volti a simularne efficacemente l'evoluzione.

Per un'equazione differenziale ordinaria esiste un risultato di esistenza, il teorema di Carathéodory, che garantisce l'esistenza di una soluzione nel caso in cui il termine forzante è discontinuo rispetto alla variabile temporale. Il caso di interesse per questo lavoro è però quello in cui il termine forzante è discontinuo dipendentemente dalle variabili di stato del problema. Per questo caso i risultati teorici di esistenza si devono a Filippov [3] e a Dieci e Lopez [8]. Il contesto è quello dei problemi ai valori iniziali descritti da equazioni differenziali ordinarie in cui il termine forzante di destra (il campo) varia in modo discontinuo nel momento in cui la traiettoria della soluzione raggiunge una o più superfici di discontinuità; lontano da esse invece la soluzione preserva la propria regolarità. In generale possono verificarsi diversi fenomeni nel momento in cui una soluzione raggiunge una superficie di discontinuità, in questo lavoro si prenderanno in considerazione solo dinamiche in cui la soluzione rimane continua. In particolare i

casi più significativi dal punto di vista fisico sono quelli in cui la traiettoria attraversa la discontinuità o vi rimane, descrivendo uno *sliding motion*. Il lavoro di Filippov fornisce condizioni per determinare il comportamento della soluzione in corrispondenza dei punti di transizione. Tali condizioni sono state recentemente estese dal lavoro di Dieci e Lopez [8] a casi più generali. Inoltre nello stesso lavoro viene proposta una strategia per l'approssimazione numerica di tali problemi che si basa sull'esatta identificazione del punto di transizione e permette di ottenere soluzioni accurate nonostante la mancanza di regolarità.

Siamo interessati a risolvere equazioni alle derivate parziali che descrivono problemi di diffusione, trasporto e reazione nell'ambito di processi geologici, nel caso in cui il termine di reazione presenti discontinuità. In questo contesto una possibile strategia risolutiva consiste nell'utilizzo di un operator splitting che permetta di separare il termine di reazione dall'equazione di diffusione e trasporto e risolverlo quindi con le tecniche più adatte.

Le tecniche di operator splitting nascono originariamente come tecniche iterative per decomporre complicate equazioni multidimensionali in equazioni monodimensionali di più facile risoluzione. Metodi di questo tipo però possono essere applicati anche al fine di trattare problemi multifisica a derivate parziali fattorizzando l'operatore associato e generando più sottoproblemi, ciascuno legato ai diversi contributi fisici [28], [16], [26]. Se si considera per esempio un problema di diffusione trasporto e reazione si può pensare di splittarlo in tre parti, ciascuna di esse governata da uno dei tre operatori che descrivono l'equazione completa. Diverse tecniche sono state proposte (si veda per esempio [16]), esse si possono suddividere in due macrocategorie: gli splitting sequenziali e quelli iterativi. In questo lavoro ci si concentra sulle tecniche del primo tipo ed in particolare sullo splitting del primo ordine di Yanenko e su quello del secondo ordine noto come splitting di Strang.

Questo lavoro di tesi ha anche come obiettivo l'implementazione efficiente degli algoritmi risolutivi per sistemi di ODE classiche e ODE-DRH e la loro applicazione nell'ambito delle equazioni differenziali a derivate parziali tramite l'utilizzo di tecniche di splitting.

Per questo scopo è stata sviluppata una libreria che implementa, oltre ai metodi di integrazione per ODE classici, le condizioni descritte in [8] e [3] per determinare il comportamento della soluzione di ODE-DRH all'intersezione con le superfici di discontinuità, e le tecniche per preservare l'accuratezza degli schemi di integrazione in tempo. Sia la teoria sia l'algoritmo proposto nel lavoro di Dieci e Lopez sono stati estesi in questa tesi al caso di problemi non autonomi, in cui il termine forzante e/o la superficie di discontinuità presentano una dipendenza dalla variabile temporale. Test numerici sono stati effettuati sia per verificare comportamento e convergenza delle tecniche implementate sia per testare queste ultime sul problema sopracitato di reazioni chimiche discontinue a causa della presenza di una soglia.

In seguito sono state studiate ed implementate tecniche di splitting applicate ad equazioni di diffusione trasporto e reazione al fine di poter separare il contributo dei primi due operatori dall'ultimo. L'obiettivo è quello di poter risolvere problemi in cui il termine di reazione presenta discontinuità; questo viene trattato separatamente tramite le tecniche risolutive per sistemi di ODE-DRH mentre i contributi di diffusione e trasporto vengono risolti con le classiche tecniche FEM. In questa ottica diventa fondamentale splittare correttamente l'equazione di diffusione trasporto e reazione. Fattorizzando il problema come detto sopra ci si trova a dover risolvere un primo sottoproblema, che è ancora nella forma di un'equazione a derivate parziali, e di un secondo sottoproblema che invece è descritto da una ODE-DRH. Sono state studiate opportune

tecniche sui metodi di integrazione in tempo e sulle condizioni al contorno degli operatori splittati al fine di preservare gli ordini di accuratezza teorici degli splitting usati.

Dal punto di vista applicativo si è studiato un modello di precipitazione e dissoluzione descritto in [17]. Esso è costituito da un'equazione a derivate parziali e da un'equazione differenziale ordinaria con forzante discontinua. La prima descrive la diffusione e il trasporto di una specie mobile che, disciolta in un fluido, può reagire, mentre la seconda descrive la dinamica dei cristalli che precipitano o si dissolvono a causa delle reazioni chimiche che avvengono. Grazie all'utilizzo delle tecniche specifiche per ODE-DRH sviluppate è possibile risolvere il problema senza ricorrere, contrariamente a quanto fatto in [17], a una regolarizzazione del termine di reazione che inevitabilmente compromette l'accuratezza della soluzione oltre a introdurre una stiffness non naturale nel problema.

Per concludere, questo modello è stato accoppiato con un modello di Darcy al fine di fornire un campo di moto realistico al trasporto delle specie disciolte nel fluido. L'accoppiamento dei due problemi è molto importante perché da un lato Darcy influenza il trasporto delle specie mobili, dall'altra la soluzione del problema di Darcy dipende dal processo di precipitazione in quanto quest'ultimo provoca una variazione della porosità e quindi della permeabilità del mezzo poroso. L'accoppiamento di queste equazioni ricopre un ruolo molto importante in ambito geofisico, si consideri per esempio il processo di formazione del petrolio. L'espulsione degli idrocarburi dalla roccia madre, ricca di materia organica, può essere modellato come un flusso di Darcy in cui acqua e idrocarburi costituscono due fasi immiscibili. Questo processo di espulsione avviene in concomitanza con una serie di reazioni chimiche che coinvolgono le varie specie di idrocarburi presenti per tutta la durata della loro migrazione nel mezzo poroso.

L'ambiente in cui è stato svolto questo progetto è `LifeV` [10], [11], [22], una libreria di calcolo parallelo a Elementi Finiti scritta in linguaggio C++, al cui sviluppo collaborano il Politecnico di Milano, l'INRIA di Parigi, l'Emory University di Atlanta e l'EPFL di Losanna. Il progetto è iniziato nel 2002 e il codice è rilasciato sotto la licenza LGPL.

E' una libreria scritta in C++ [7] che fa uso di alcune librerie esterne, tra cui `Trilinos` [14] per il calcolo matriciale e `ParMetis` [18] per il partizionamento delle matrici. `LifeV` è un codice parallelo che utilizza il protocollo MPI per la comunicazione attraverso i diversi processori.

La collaborazione con gli altri sviluppatori `LifeV` è stata resa possibile grazie all'utilizzo di `Git`, un programma concepito per la gestione delle versioni [30], [29]. Esso permette infatti un'efficiente gestione dello sviluppo di un programma nel momento in cui a partire da una stessa base gli sviluppatori portano avanti modifiche, aggiornamenti o novità a parti diverse del programma.

Quanto viene presentato in questa tesi fa attualmente parte di `ODE`, un modulo sperimentale di `LifeV`. Al suo interno si trovano tutte le classi necessarie alla risoluzione dei sistemi di ODE-DRH e gli esempi che riproducono i risultati proposti in questa tesi. In particolare la risoluzione del modello proposto in [17] per la precipitazione e dissoluzione e l'accoppiamento dello stesso con un modello di Darcy già implementato in `LifeV`.

La tesi è articolata come segue:

Nel primo capitolo Si affronta il problema della soluzione di ODE-DRH. Viene descritta la struttura del codice implementato, vengono presentati risultati di convergenza e test che

riproducono alcuni casi citati in letteratura.

Nel secondo capitolo si presentano le tecniche di operator splitting usate in questo lavoro evidenziandone pregi ed eventuali limiti e si propongono strategie per preservarne l'accuratezza anche in presenza delle limitazioni incidentalmente incontrate.

Nel terzo capitolo si presenta il modello di precipitazione e dissoluzione e la sua soluzione numerica tramite splitting usando tecniche FEM e la libreria di ODE-DRH implementata.

Nel quarto capitolo si accoppia il modello usato nel capitolo precedente con un modello di flusso di Darcy che fornisce il campo di avvezione.

Nel capitolo conclusivo dopo aver brevemente riassunto i risultati ottenuti in questa tesi, vengono date indicazioni circa i possibili sviluppi del lavoro svolto.

Data l'importanza del lavoro di implementazione nell'ambito di questa tesi, per permettere un più facile utilizzo futuro degli algoritmi implementati si sono forniti all'interno di ciascun capitolo, dove rilevante, alcuni dettagli della struttura del codice sviluppato.

Capitolo 1

Sistemi di ODE con forzanti discontinue

Si vuole studiare il comportamento e l'approssimazione numerica di sistemi di ODE caratterizzati da un termine forzante discontinuo. Sistemi di questo tipo si incontrano in applicazioni di varia natura: sistemi meccanici, applicazioni in ambito biologico e problemi di controllo. In questo lavoro concentreremo la nostra attenzione in particolare su applicazioni geologiche in cui il termine di reazione, accoppiato a fenomeni di trasporto in mezzi porosi, può essere formulato come un sistema di equazioni differenziali ordinarie discontinuo.

1.1 Definizione del problema

Consideriamo un sistema di equazioni differenziali ordinarie con termine forzante discontinuo (ODE-DRH). In generale il problema può essere scritto nella seguente forma:

$$\frac{d\mathbf{y}}{dt} = \begin{cases} \mathbf{f}_1(t,\mathbf{y}), & \text{se } \mathbf{g}(t,\mathbf{y}) < 0 \\ \mathbf{f}_2(t,\mathbf{y}), & \text{se } \mathbf{g}(t,\mathbf{y}) > 0 \end{cases} \tag{1.1}$$

$$\mathbf{y}(t_0) = \mathbf{y}_0 \tag{1.2}$$

Si noti che il termine forzante è discontinuo in corrispondenza di una o più superfici di passaggio definite a livello matematico come la superficie di livello zero di una funzione nelle variabili di stato del sistema e nel tempo ($\mathbf{g}(t,\mathbf{y}) = 0$).
Poichè il termine forzante è discontinuo non è possibile dimostrare l'esistenza della soluzione con il teorema di Peano. Per questo tipo di problema inoltre non risulta applicabile nemmeno il teorema di Caratheodory [31] che richiede che il termine forzante $f(t,\mathbf{y})$ sia discontinuo solo rispetto alla variabile t. Allo scopo di trattare la presenza di questo tipo di discontinuità ci si baserà sulla teoria di Filippov [3] estesa alle condizioni di ordine superiore da Dieci-Lopez in [8] e nella tesi di dottorato [2]. Inoltre, a differenza di quanto fatto in tali lavori, sarà considerata anche la possibilità che il sistema, come evidenziato in (1.1), possa essere non autonomo sia nelle forzanti che nella funzioni di soglia.

In generale consideriamo un termine forzante continuo a tratti in un sottoinsieme limitato S dello spazio $n+1$ dimensionale $(t,\mathbf{y})$, cioè continuo fino al bordo di ogni sottodominio S_k, $k = 1,\dots l$ in cui S è suddiviso. In Fig. 1.1 è rappresentata qualitativamente la traiettoria della soluzione in uno spazio suddiviso in 7 regioni da 3 superfici di discontinuità: si noti che

la traiettoria della soluzione può seguire le superfici di discontinuità o attraversarle (cosa che però non è rappresentata in figura).

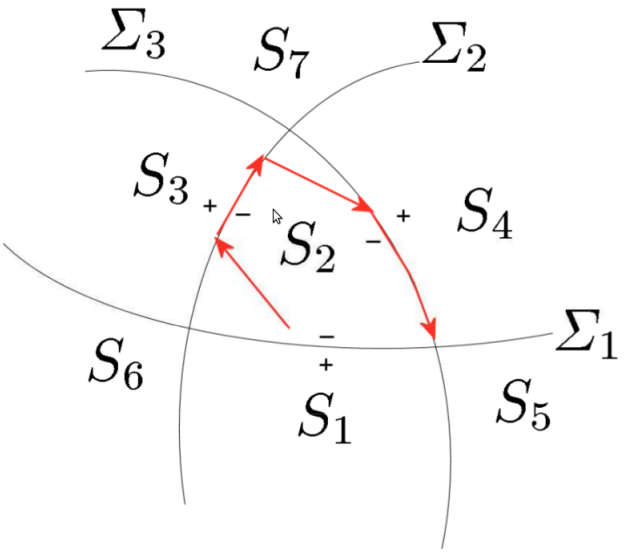


Figura 1.1: Esempio di traiettoria con campi discontinui, tratto da [2].

1.1.1 Analisi di Filippov

Per definire la soluzione nel caso di sistemi del tipo (1.1) è necessario introdurre il concetto di inclusione differenziale [4]. Sia S_i una sua partizione finita di aperti e I l'intervallo di tempo considerato. Si definisce una multifunzione $\mathbf{F}$ associata al termine noto $\mathbf{f}$ nel modo seguente:

Definizione 1. *Data una funzione*

$$\mathbf{f} : I \times \mathbb{R}^n \to \mathbb{R}^n \qquad (t, \mathbf{y}) \to \mathbf{f}(t, \mathbf{y})$$

definita continua a tratti in S, si definisce la multifunzione $\mathbf{F}(t, \mathbf{y})$ per $t \in I$ e $\mathbf{y} \in \mathbb{R}^n$, tale che per ogni $(t, \mathbf{y}^*) \in S$ $\mathbf{F}(t, \mathbf{y})$ è composta dal più piccolo convesso chiuso contenente tutti i valori limite di $\mathbf{f}(t, \mathbf{y})$ per $\mathbf{y} \to \mathbf{y}^*$, a t fissato, per $(t, \mathbf{y}) \in \bigcup_i S_i$.

Un'inclusione differenziale ha la seguente forma:

$$\frac{d\mathbf{y}}{dt} \in \mathbf{F}(t, \mathbf{y}) \qquad \text{per } t \in I. \tag{1.3}$$

Sotto opportune ipotesi [3] si può definire la soluzione di (1.1) come la soluzione dell'inclusione differenziale (1.3) con condizioni iniziali $\mathbf{y}(0) = \mathbf{y}_0$ e

$$\mathbf{F}(t, \mathbf{y}) = \begin{cases} \mathbf{f}_1(t, \mathbf{y}), & \text{se } \mathbf{g}(t, \mathbf{y}) < 0 \\ \overline{co}\,\{\mathbf{f}_1, \mathbf{f}_2\}, & \text{se } \mathbf{g}(t, \mathbf{y}) = 0 \\ \mathbf{f}_2(t, \mathbf{y}), & \text{se } \mathbf{g}(t, \mathbf{y}) > 0, \end{cases}$$

dove l'inviluppo convesso $\overline{co}\,\{\mathbf{f}_1, \mathbf{f}_2\}$ verrà definito in seguito.

L'analisi di Filippov fornisce le condizioni per stabilire il comportamento della soluzione nei punti in cui incontra le superfici di discontinuità. Per definire tali condizioni è necessario introdurre alcune quantità. Sia la variabile di stato y_i appartenente alla funzione di soglia $g_i(t, \mathbf{y}) = 0$ e definiamo con $\mathbf{n}(t, \mathbf{y})$ la normale a tale superficie. Essa potrà essere calcolata come:

$$\mathbf{n}(t, \mathbf{y}) = \frac{\partial g_i/\partial t + \nabla g_i(t, \mathbf{y})}{\|\partial g_i/\partial t + \nabla g_i(t, \mathbf{y})\|}.$$

Per ogni i definiamo la variabile ausiliaria $\gamma_{1,2}(t, \mathbf{y}) = \partial g_i/\partial t + \nabla g_i \cdot \mathbf{f}_{1,2}$ dove, per semplicità, si è omesso di indicare l'indice i della soglia attiva e si considera sempre il campo di forzanti sotto (γ_1) e sopra (γ_2) tale soglia. I termini γ sono interpretabili come le proiezioni dei campi di forzanti corrispondenti sulla normale alla soglia attiva. Sia $(t, \mathbf{y})$ un punto appartenente ad una superficie di discontinuità del sistema, cioè $g_i(t, \mathbf{y}) = 0$ per un i. Si definisce

$$A_{1,2} = \gamma_{1,2}(t, \mathbf{y}). \tag{1.4}$$

Analizzando i segni di $A_{1,2}$ è possibile determinare se, quando la traiettoria interseca una superficie di discontinuità nel punto dato, la soluzione

- attraversa la soglia ed entra in S_1 o S_2 (*crossing motion*)
- prosegue lungo la superficie di discontinuità (*sliding motion*).

In particolare:

1. se $A_1 \cdot A_2 > 0$, la soluzione lascia la soglia (crossing). Essa entra nella regione sotto soglia (S_1) se $A_1 < 0$, entra nella regione sopra soglia (S_2) se $A_2 > 0$. Il campo di forzanti con cui si proseguirà sarà corrispondente alla regione di destinazione.

2. se $A_1 \cdot A_2 < 0$ e $A_1 > 0$ e $A_2 < 0$ si ha una soluzione che rimane sulla soglia (*sliding motion*). Inoltre tale soglia risulta essere attrattiva e le variabili di stato saranno soggette ad un campo di forzanti $\mathbf{f}_F$ che permetterà di proseguire nel loro percorso rimanendo ancorate alla soglia. Tale campo è definibile come:

   $$\mathbf{f}_F(t, \mathbf{y}) = (1 - \alpha(t, \mathbf{y}))\mathbf{f}_1^i(t, \mathbf{y}) + \alpha(t, \mathbf{y})\mathbf{f}_2^i(t, \mathbf{y}). \tag{1.5}$$

   dove i è l'indice della variabile coinvolta nello *sliding motion* e:

   $$\alpha(t, \mathbf{y}) = \frac{\partial g_i(t, \mathbf{y})/\partial t + (\mathbf{n}^T \mathbf{f}_1^i(t, \mathbf{y}))}{\mathbf{n}^T(\mathbf{f}_1^i(t, \mathbf{y})) - \mathbf{f}_2^i(t, \mathbf{y}))}.$$

3. se $A_1 \cdot A_2 < 0$ e $A_1 < 0$ e $A_2 > 0$ si ha uno *sliding motion* repulsivo. Tale condizione risulta instabile e l'unicità della soluzione non è garantita perchè le variabili di stato possono passare spontaneamente in S_1 o S_2.

Tale teoria non è esaustiva delle possibili condizioni che possono verificarsi per un problema di $ODE - DRH$ in quanto non considera i casi in cui i termini $A_{1,2}$ assumano valori nulli. Per poter trattare anche tali casi si utilizza la teoria di Dieci-Lopez che prevede condizioni di ordine superiore. Le condizioni ricavate in [8] sono state generalizzate al caso non autonomo. La

derivazione delle condizioni per il caso non autonomo è stata fatta formalmente introducendo la variabile ausiliaria τ e considerando $t = t(\tau)$ con

$$\begin{aligned} &\frac{dt}{d\tau} = 1, \\ &t = t_0 \quad \text{per } \tau = \tau_0. \end{aligned}$$

In questo modo si costruisce formalmente un sistema autonomo equivalente a quello originario. Omettiamo i dettagli presentando solo i risultati.

1.1.2 Condizioni di ordine superiore

Sia $\mathbf{y} = \mathbf{y}(t)$ una traiettoria soluzione di (1.1) che nel punto $(t, \mathbf{y}(t))$ è di soglia, cioè $g_i(t, \mathbf{y}(t)) = 0$ per un valore di i. Per una funzione $h = h(\tau)$ indicheremo con $[h]_{t^\pm}$ rispettivamente i valori limite per $\tau \to t^+$ e $\tau \to t^-$ Definiamo i seguenti termini ausiliari derivanti dallo sviluppo al secondo ordine delle funzioni γ definite in precedenza:

$$B^\pm_{1,2} = \left[\frac{\partial \gamma_{1,2}}{\partial t} + \nabla\gamma_{1,2}\mathbf{y}'\right]_{t^\pm} \tag{1.6}$$

$$C^\pm_{1,2} = \left[\frac{\partial^2 \gamma_{1,2}}{\partial t^2} + \nabla\gamma_{1,2}\mathbf{y}'' + \mathbf{y}'^T\frac{\partial \gamma_{1,2}}{\partial \mathbf{y}^2}\mathbf{y}'\right]_{t^\pm} \tag{1.7}$$

Chiaramente nelle espressioni precedenti i valori di $\mathbf{y}'$ e $\mathbf{y}''$ sono ottenuti tramite (1.1), e quindi usando $\mathbf{f}_1$ o $\mathbf{f}_2$ (e le loro derivate) a seconda del caso. Ora se $A_1 \neq 0$ e $A_2 \neq 0$ si applica l'analisi al primo ordine di Filippov. Al contrario, i casi in cui $A_1 = 0$ o $A_2 = 0$ determinano un andamento della soluzione che può essere descritto studiando non solo i segni di A, ma anche quelli di B e C.

1. $A_1 = 0$ e $A_2 \neq 0$. Se si arriva da uno *sliding motion* attrattivo ($A_2 < 0$) si dimostra che necessariamente si ha $B_1^- \leq 0$. In tale condizione, si possono verificare i seguenti andamenti delle variabili di stato:

   - $B_1^- < 0$ allora la soluzione lascia la soglia e va in S_1 con il campo corrispondente.
   - $B_1^- = 0$ allora si dimostra che $C_1^- \geq 0$. Inoltre, lo *sliding motion* continua con un campo che è corrispondente a quello in S_1.

   Se la soluzione arriva da sotto soglia si hanno le seguenti possibilità:

   - $A_2 < 0$, allora si applica la analisi del punto precedente.
   - $A_2 > 0$ allora si ha uno *sliding motion* repulsivo ed instabile se $B_1^- < 0$, una soluzione di crossing in S_2 se $B_1^- = 0$ e $C_1^- > 0$.

   Se si arriva da S_2 allora $A_2 < 0$, $y'(t^-)$ è uguale al campo sopra soglia e $y'(t^+)$ è pari al campo sotto soglia. In tal caso si hanno le seguenti possibilità:

   - $B_1^+ < 0$, allora c'è soluzione di crossing in S_1 con il campo corrispondente di forzanti.
   - $B_1^+ > 0$ allora si ha una soluzione di *sliding motion* che rimane con un campo corrispondente ad S_1.

2. $A_2 = 0$ e $A_1 \neq 0$. L'analisi è la stessa di prima con gli indici 1 e 2 ribaltati.

3. $A_1 = 0$ e $A_2 = 0$. In tale condizione si può arrivare a soluzione molto complesse e non del tutto definibili. Tuttavia si hanno dei risultati parziali per casi nei quali si può decidere cosa fare. In particolare se $B_1^- = B_2^- = 0$ lo sliding continuerà, se $B_1^- < 0$, $B_2^- = 0$ e $C_2^- < 0$ c'è crossing in S_1, se $B_1^- = 0$, $B_2^- > 0$ e $C_1^- > 0$ c'è crossing in S_2, se $B_1^- < 0$ e $B_2^- > 0$ c'è una soluzione instabile e si perde l'unicità.

1.2 Soluzione numerica di sistemi ODE-DRH

Nel caso di sistemi con termine forzante discontinuo i metodi di discretizzazione tradizionali possono presentare un decadimento dell'ordine di convergenza previsto dalla teoria nel caso regolare. Per ovviare a questo problema esistono diverse tecniche. In questa tesi ci concentreremo sul metodo proposto in [8], esteso al caso non autonomo, che consiste nel localizzare con precisione il punto di transizione in cui la traiettoria attraversa la superficie di discontinuità, verificare le condizioni di primo e secondo ordine e quindi far ripartire l'integrazione con il termine forzante appropriato. Tecniche alternative a questo approccio possono essere l'adattività del passo di integrazione [12] per ridurre l'errore legato al salto finito del termine noto, o la regolarizzazione del termine noto [23, 17]. L'analisi presentata nella tesi di dottorato [2] mostra che metodi basati sulla localizzazione del punto di transizione forniscono in generale risultati migliori in termini di accuratezza.

Nei paragrafi seguenti verranno forniti dettagli sul metodo numerico utilizzato per integrare il problema (1.1) in prossimità delle superfici di discontinuità di un punto di transizione.

1.2.1 Ricerca punti di transizione

Ci si ponga nel caso in cui la traiettoria della soluzione intersechi una soglia durante l'evoluzione del sistema. In generale, e a meno di usare dei forti infittimenti locali della discretizzazione temporale in prossimità di una superficie di discontinuità, è molto probabile che l'integrazione numerica fornisca una soluzione che, a distanza di un passo temporale e l'altro, appartenga a regioni opposte rispetto alla soglia. E' quindi necessario definire un metodo per localizzare con precisione il punto in cui la traiettoria interseca la discontinuità per poter testare, in corrispondenza di tale istante, le condizioni presentate nel paragrafo precedente e determinare il comportamento della soluzione. Tale localizzazione, se esatta, permette di preservare l'ordine di convergenza dello schema numerico in presenza di campi discontinui.

Detto Δt il passo di discretizzazione temporale e $t_n = t_0 + n\Delta t$, si immagini per esempio di avere una soluzione numerica $\mathbf{y}^{(n)}$ che appartiene all'istante t_n ad una regione S_1 sotto soglia ($g_i(t_n, \mathbf{y}(t_n)) < 0$) mentre all'istante temporale successivo appartiene ad una regione S_2 sopra soglia ($g_i(t_{n+1}, \mathbf{y}(t_{n+1}) > 0$). Si vuole cercare il tempo t^* tale per cui $g_i(t^*, \mathbf{y}(t^*)) = 0$ e la corrispondente soluzione $\mathbf{y}(t^*)$. Tale problema può essere formulato matematicamente come la ricerca dello zero della funzione $G(\eta) = g_i(\eta, \mathbf{y}(\eta))$. Usando il metodo delle secanti si ottiene il seguente algoritmo iterativo di risoluzione:

$$\eta_{k+1} = \eta_k - \frac{\eta_k - \eta_{k-1}}{G(\eta_k) - G(\eta_{k-1})} G(\eta_k)$$

inizializzato con $\eta_0 = t^{(n)}$ e $\eta_1 = t^{(n+1)}$. Come descritto in dettaglio nei paragrafi seguenti tale metodo è stato implementato con un criterio d'arresto basato sulla differenza tra un passo ed il successivo. Si noti inoltre che ad ogni iterazione corrisponde l'integrazione del problema (1.1) su un intervallo temporale per poter valutare $G(\eta_k)$.

È possibile calcolare il t^* di transizione in corrispondenza di ogni tipo di transizione, che si tratti di una soluzione crossing, dell'inizio di uno *sliding motion* o dell'uscita da esso. Si osservi però che tale calcolo non è sempre necessario ai fini dell'accuratezza del metodo. Può essere conveniente definire il t^* di transizione delle variabili di stato anche quando si verificano le condizioni di uscita da una soglia. In tal caso, per campi continui e regolari al di fuori delle superfici di discontinuità, il vantaggio conseguente al calcolo del t^* esatto sarà minore in quanto il campo passando da uno stato di sliding attrattivo difficilmente avrà valori che si discosteranno molto dalla tangente alla soglia, sulla quale è orientato anche il campo $\mathbf{f}_F$ sulla superficie di discontinuità; quindi anche non effettuando una ricerca locale del t^* di uscita i valori di $\mathbf{y}$ saranno una buona approssimazione di quelli esatti.

Ipotizziamo per esempio di avere ad un certo passo temporale $A_1 > 0$ e $A_2 < 0$ e di trovarci quindi in una condizione di *sliding motion* attrattivo. Se al passo successivo si verificano le condizioni $A_1 > 0$ e $A_2 > 0$ la traiettoria deve lasciare la discontiuità ed entrare in S_2. Per ottenere l'istante preciso di uscita in questo caso si cerca lo zero della funzione $A_2(t, \mathbf{y})$. Il procedimento numerico per trovare lo zero di tale funzione è il medesimo illustrato per il caso precedente. Ovviamente, nel caso si vada verso S_1, si ricercherà lo zero di $A_1(t, \mathbf{y})$.

1.2.2 Proiezione su soglia

Si ipotizzi ora invece di essere in condizione di *sliding motion* attrattivo su di una soglia attiva. Anche se i risultati teorici presentati consentono di calcolare il termine forzante corretto nel caso di *sliding motion*, a causa di errori numerici la soluzione approssimata potrebbe discortarsi dalla soglia. Per ovviare a questo problema si implementa un algoritmo di proiezione sulla soglia attiva.

Sia $\widehat{\mathbf{y}}$ un punto in prossimità di una soglia $g_i = 0$ a tempo fissato, allora la sua proiezione $\mathbf{y} = P(\widehat{\mathbf{y}})$ sarà la soluzione del seguente problema di minimo vincolato: si trovi $\mathbf{y}$ tale che

$$q(\mathbf{y}) = \min_{g_i(\mathbf{x})=0} q(\mathbf{x})$$

con $q(\mathbf{x}) = \dfrac{1}{2}(\widehat{\mathbf{x}} - \mathbf{x})^T(\widehat{\mathbf{x}} - \mathbf{x})$.

Tale problema di minimo verrà trattato attraverso moltiplicatore di Lagrange λ per il vincolo di uguaglianza $g_i(\mathbf{x}) = 0$. Si tratta di trovare le radici del seguente sistema non-lineare

$$Q(\mathbf{y}, \lambda) = \begin{bmatrix} \nabla q(\mathbf{y}) + \lambda \nabla g(\mathbf{y}) \\ g(\mathbf{y}) \end{bmatrix},$$

con $\lambda \in \mathbb{R}$. Facendo ricorso al metodo di Newton si dovrebbe risolvere ad ogni iterazione fino a convergenza il sistema

$$\begin{bmatrix} I + \lambda \Delta g(\mathbf{y}^k) & \nabla g(\mathbf{y}^k) \\ \nabla^T g(\mathbf{y}^k) & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}^{k+1} - \mathbf{y}^k \\ \lambda^{k+1} - \lambda^k \end{bmatrix} = -Q(\mathbf{y}^k, \lambda^k), \quad k \geq 0.$$

Per evitare di dover risolvere un vero sistema lineare ad ogni iterazione k si è scelto di utilizzare la tecnica proposta in [8] e di usare quindi il seguente metodo iterativo semplificato:

$$\begin{bmatrix} I & \nabla g(\mathbf{y}^k) \\ \nabla^T g(\mathbf{y}^k) & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}^{k+1} - \mathbf{y}^k \\ \lambda^{k+1} - \lambda^k \end{bmatrix} = -Q(\mathbf{y}^k, \lambda^k), \quad k \geq 0.$$

Questa approssimazione è legittima in quanto ci aspettiamo di essere in ogni caso vicini alla soluzione della minimizzazione. Inoltre in questo modo lo jacobiano approssimato si presenta in una forma facilmente fattorizzabile, dato che

$$\begin{bmatrix} I & b \\ b^T & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ b^T & 1 \end{bmatrix} \begin{bmatrix} I & b \\ 0 & -b^T b \end{bmatrix},$$

La tecnica si può comunque estendere anche al caso di più superfici di discontinuità contemporaneamente attive.

1.3 Dettagli sull'implementazione

Il metodo illustrato è stato implementato in codice C++ strutturato in due librerie. La prima libreria risolve sistemi di ODE classici senza il trattamento di discontinuità. Essa risulta del tutto indipendente dalla seconda e può essere usata come libreria autonoma. La seconda libreria risolve sistemi di ODE DRH. Tale seconda libreria importa i file relativi alla prima e utilizza i solutori ivi implementati per avanzare in tempo con il campo di forzanti definito dall'analisi per il trattamento delle discontinuità presentata nella sezione precedente.

La libreria per la risoluzione di sistemi di ODE classici senza il trattamento di discontinuità importa strutture dati e funzioni delle librerie `Boost` [6] (versione utilizzata 1.48.0) e il pacchetto di algebra lineare `Epetra` di `Trilinos` [14] (versione utilizzata 10.10.1). In particolare `Epetra` è stata utilizzata per la risoluzione dei sistemi lineari derivanti dall'utilizzo del metodo di Newton negli schemi di integrazione impliciti. Poiché la libreria per sistemi di ODE DRH importa la precedente ovviamente richiede essa stessa le medesime librerie. Infine entrambe utilizzano per il passaggio dei dati una libreria di parser utilizzata anche nella libreria `LifeV` sviluppata al MOX in collaborazione con EPFL e Emory University (i dati potranno essere passati con due modalità differenti spiegate più nel dettaglio in seguito).

1.3.1 Libreria per sistemi di ODE classici

La libreria implementa i metodi di Eulero esplicito, Eulero implicito, Runge-Kutta del quarto ordine, Heun, Crank-Nicolson, Adams-Bashforth, Adams-Moulton, BDF2 e BDF3 per la risoluzione di sistemi di equazioni differenziali ordinarie. É tuttavia sufficientemente generica da poter implementare in futiro altri schemi numerici.

Dati di input

La libreria richiede in input l'intervallo temporale (tempo iniziale e tempo finale) in cui si vuole approssimare numericamente la soluzione del sistema, il passo temporale che si vuole utilizzare, il numero di variabili ossia la dimensione del sistema, il termine forzante del sistema,

le condizioni iniziali e il nome del file su cui si vogliono salvare i risultati. Si deve inoltre specificare il metodo di discretizzazione scelto.

Tali variabili di ingresso possono essere fornite attraverso un file esterno o implementate direttamene all'interno del codice. Ovviamente la lettura da file permette di non dover ricompilare il codice se si cambiano i parametri di simulazione. Per quanto riguarda le forzanti esse, nel caso si utilizzi un file di input, saranno passate come stringhe e trattate mediante il parser, altrimenti saranno gestite mediante funtori (puntatori a funzioni). Come si spiegherà più nel dettaglio in seguito, il codice è stato progettato in modo tale che la classe che implementa la risoluzione del sistema risulti del tutto trasparente alle due strategie di lettura del termine forzante che vengono definite esternamente a tale classe.

Se il metodo richiesto è implicito sarà necessario passare anche l'espressione delle derivate del termine forzante, ancora una volta, sia direttamente mediante file di input sia dal codice. Sempre per metodi impliciti è possibile definire i parametri di tolleranza e numero massimo di iterazioni da utilizzare per il metodo di Newton implementato anch'esso ex novo a causa delle specificità del problema che rendono complessa l'integrazione di librerie esterne per la soluzione di sistemi non-lineari. Se tali parametri non vengono definiti si utilizzano valori di default.

Per metodi multistep (Adams-Bashforth e Adams-Moulton) è possibile cambiare il metodo ad un passo da usare per i primi passi del metodo (quando non sono a disposizione abbastanza soluzioni precedenti per usare il corrispondente multistep).

Nel momento in cui viene lanciata la simulazione numerica per la risoluzione del sistema, il codice effettua una serie di controlli sulla coerenza dei dati in ingresso (che il tempo iniziale sia minore del tempo finale, che vengano fornite le derivate nel caso di un metodo implicito...) fornendo un messaggio di errore e arrestando la simulazione nel caso di dati non corretti.

Struttura del codice

La classe centrale della libreria è `Solver` che implementa il risolutore astratto per il sistema di ODE. In essa sono definite le strutture dati delle variabili coinvolte. In particolare le soluzioni vengono memorizzate in un vettore di liste di *double*. La scelta di tale struttura deriva dal fatto che i metodi risolutivi richiederanno per la maggior parte la soluzione al passo precedente per l'avanzamento in tempo nel caso di metodi one-step, o fino a tre passi precedenti per i metodi multistep implementati.

Le forzanti e le eventuali derivate sono salvate come vettori di puntatori ad una classe `Functor`. Tale classe è astratta e definisce l'operatore `()` per la valutazione del valore della funzione al variare delle variabili spaziali e della variabile temporale. L'operatore `()` utilizza una funzione `GetValue` che è la funzione definita *pure virtual* nella classe `Functor`. Le classi `RealFunctor` e `ParsedFunctor` ereditano dalla classe `Functor` astratta, definiscono la corretta struttura dati a seconda rispettivamente del caso di passaggio da funtori e di passaggio da stringa con Parser e fanno l'override corrispondente per la funzione `GetValue`. Questo permette nella classe Solver di usare genericamente puntatori a `Functor` che a seconda del tipo di passaggio corrisponderanno a puntatori ad uno o all'altro tipo concreto. Visto che l'interfaccia all'utente delle due classi concrete derivate da `Functor` è la stessa, l'implementazione della classe `Solver` è del tutto trasparente al tipo di funzioni passate. La scelta di mantenere due tipi di passaggi ha queste motivazioni: in primo luogo, il passaggio da file mediante `Parser` è certamente più comodo per l'utilizzatore, ma se la libreria venisse usata in un codice più esteso

in cui i funtori sono già disponibili per altri scopi (come avverrà anche nel nostro caso per la libreria per ODE-DRH) è utile non costringere l'utente a riscriverli sotto forma di stringhe o di scrivere un file apposito; inoltre, se il passaggio da file è più comodo è altresì vero che una funzione passata come funtore richiederà un tempo per la sua valutazione minore rispetto a una funzione passata con `Parser`. Visto che nel codice vanno fatte molte valutazioni delle forzanti, ciò risulta in un tempo di esecuzione minore per il passaggio con funtori, che in casi complessi può risultare estremamente vantaggioso.

Il metodo centrale della classe `Solver` è `DoIt` che permette di eseguire l'integrazione numerica. Esso contiene la struttura comune a tutti i metodi per il calcolo della soluzione ed utilizza il metodo `AdvancingStep` che è, nella classe `Solver`, *pure virtual*. Tale metodo viene implementato nelle classi figlie che ereditano da `Solver` e definiscono i singoli metodi specifici di avanzamento in tempo elencati in precedenza. Oltre alla funzione `AdvancingStep`, a seconda del tipo di risolutore utilizzato altri metodi saranno soggetti ad overriding nelle classi figlie e altre variabili definite. In particolare per i metodi impliciti si definiscono le variabili di `M_nmax_newton` e `M_toll_newton` che servono per definire rispettivamente il massimo numero di iterazioni e la tolleranza per il metodo di Newton. Esse avranno un valore di default modificabile attraverso la funzione `setNewtonParam`. Sempre per i metodi impliciti si fa l'override anche della funzione `NonLinearFunJac` che definisce, a seconda del metodo, lo jacobiano e la forzante del sistema lineare da risolvere ad ogni passo di Newton. Viene inoltre fatto l'override della funzione `DataisOK` che controlla i dati passati al problema prima di iniziare a risolverlo, per controllare anche che le derivate siano state caricate. Per i metodi multistep (Adams-Bashforth e Adams-Moulton) si aggiunge la variabile `M_OneStepMet` per il metodo ad un passo da usare ai primi passi che di default è Eulero esplicito, ma può essere cambiato con la funzione `setOneStep`. Tale funzione prende valori da un enum `OneStepSol` che può assumere i seguenti valori: OneStepExplicitEuler, OneStepImplicitEuler, OneStepHeun, OneStepRK4, OneStepCrankNicolson.

L'istanziamento di oggetti delle classi concrete relative ai risolutori può avvenire direttamente dichiarando esplicitamente il nome della classe del risolutore corrispondente. Viene però implementata un'altra classe specifica denominata `Creator` che permette di istanziare un puntatore ad un oggetto del risolutore desiderato, ovviamente sia attraverso passaggio da file che da codice, utilizzando i metodi interni chiamati *methods*. Tale classe restituisce un puntatore a `Solver` che ovviamente in base al tipo di risolutore richiesto sarà un puntatore alla classe del risolutore concreto corrispondente. Inoltre tale `Creator` è un singleton, di tale determinata classe verrà creata una e una sola istanza. I *methods*, oltre agli input definiti in precedenza a seconda del tipo di passaggio dei dati, richiedono che sia specificato il metodo da creare definito come un enum `SolverType` che può assumere i seguenti valori: MetExplicitEuler, MetImplicitEuler, MetCrankNicolson, MetHeun, MetRK4, MetAdamsBashforth, MetAdamsMoulton, MetBDF2, MetBDF3.

1.3.2 Libreria per sistemi di ODE-DRH

La seconda libreria implementa il metodo descritto nella sezione precedente per la soluzione di sistemi di equazioni differenziali ordinarie con termine forzante discontinuo. Essa utilizza i metodi di avanzamento in tempo della prima libreria e testa le condizioni per cambiare, dove

serve, il campo di forzanti a cui sono soggette le variabili. Non si considerano casi in cui più soglie sono attive contemporaneamente.

Dati di input

La libreria per sistemi di ODE DRH richiede in ingresso, come la precedente, la finestra temporale in cui si vuole approssimare numericamente la soluzione del sistema (tempo iniziale e tempo finale), il passo temporale che si vuole utilizzare, il numero di variabili, le forzanti del sistema sui due lati di ogni discontinuità, le condizioni iniziali, e il nome del file su cui si vogliono salvare i risultati. Sono inoltre richieste le derivate rispetto al tempo di tali forzanti, il loro gradiente rispetto alle variabili di stato, l'espressione delle superfici di discontinuità, il loro gradiente e le derivate rispetto al tempo. Si deve inoltre specificare il metodo desiderato per la risoluzione del problema al di fuori dalle soglie ed il metodo, eventualmente diverso, da utilizzare lungo la soglia. Per la risoluzione su soglia si utilizzano metodi espliciti quindi se il metodo scelto dall'utente è esplicito viene esteso anche alla soglia, altrimenti viene attribuito di default Runge-Kutta del quarto ordine. Tale metodo è comunque modificabile fornendo l'apposita funzione. Nel caso di metodo generale multistep sarà inoltre modificabile attraverso opportuna funzione il metodo ad un passo corrispondente. I due metodi di passaggio per i dati del problema descritti per la libreria precedente sono validi anche in questo caso e le funzioni sono trattate nei due modi descritti. Si noti inoltre che per il passaggio da file si attribuisce tramite parser di default la funzione identicamente nulla quindi non è necessario scrivere le funzioni e le loro derivate nel caso in cui esse siano identicamente nulle (per esempio nel caso autonomo le derivate in tempo semplicemente non si scrivono).

Struttura del codice

La classe centrale della libreria è OdeDrhSolver che implementa il risolutore di sistemi di ODE DRH secondo la teoria di Dieci-Lopez estesa anche a casi non autonomi. Come nel caso precedente si memorizza la soluzione in un vettore di liste. Si utilizzano inoltre due membri `M_ODESolver` e `M_ODESolver_on` di tipo puntatore a `Solver` generico che verranno utilizzati per avanzare in tempo di un passo al di fuori di soglia e con soglia attiva rispettivamente. Ad ogni transizione si andranno a cambiare i `Functor` specifici di tali metodi attraverso il metodo pubblico `setForcer` della classe `Solver` e si selezionerà il risolutore da utilizzare. Ovviamente nel costruttore tali puntatori a `Solver` generico andranno a puntare ad un oggetto della classe figlia del metodo Solver a seconda del metodo di integrazione richiesto. Le funzioni richieste e date come input saranno definite come vettori di puntatori a `Functor` e trattate come nel caso della libreria precedente a seconda di come vengono passate. Rispetto alla libreria precedente si è aggiunta la variabile `M_time` necessaria per memorizzare i tempi perché, nel caso si verifichino transizioni attraverso le soglie, la soluzione verrà memorizzata non solo in corrispondenza degli istanti di tempo equispaziati da `M_t_start` a `M_t_end` secondo il passo definito dall'utente, ma anche in corrispondenza degli istanti di transizione t^*. Inoltre vi è anche una variabile `M_flag` definita come un vettore di pile di interi che serve per memorizzare le flag delle soluzioni ai vari istanti (-1 se sotto soglia, 0 se su soglia, 1 se sopra soglia) per ogni variabile.

Il metodo fondamentale per l'integrazione del sistema di ODE DRH è `DoIt` che definisce la struttura generale dell'algoritmo. In esso si utilizza il metodo `AdvancingStep` che effettivamente implementa la teoria per la risoluzione di tale tipo di sistema. Ad ogni passo si attua

l'avanzamento in tempo di uno step con il risolutore attivo a seconda delle ultime flag caricate. Si testa poi con una funzione `WhereAmI` la soluzione di arrivo. Se la variabile si trovava inzialmente fuori da soglie e dopo l'avanzamento in tempo la situazione non è cambiata non è necessario testare nulla e `AdvancingStep` restituisce il risultato ottenuto. Se si attraversa una soglia si ricerca con `ZeroGFind` il tempo di arrivo sulla soglia e la soluzione a tale tempo che vengono caricate nelle strutture dati opportune. A tale punto si testano le condizioni e, in base alla teoria, si deduce come la soluzione deve proseguire e si porta a compimento il passo temporale muovendosi con tale nuovo campo di forzanti del tempo mancante. Se si è su una soglia ad ogni passo si testano le condizioni di uscita. Se si rimane sulla soglia si utilizza `project_on` per riproiettare la soluzione numerica sulla soglia attiva. Se si deve uscire si calcola con `ZeroGammaFind` il tempo in cui si deve uscire e si conclude il passo temporale con il nuovo campo. Se ad un avanzamento in tempo si verifica una condizione non definita dalla teoria o si verifica una condizione instabile e con soluzione non unica si restituisce un errore corrispondente e si arresta la simulazione. In ogni caso in cui vi sono cambiamenti nelle flag `M_usedforcer` permette di calcolare le nuove forzanti da usare.

Le funzioni A, B e C servono per calcolare i corrispondenti coefficienti da testare per la teoria di Dieci-Lopez (come definite in (1.4), (1.6) e (1.7)). Si noti come, se per le A le funzioni passate esatte sono sufficienti per il calcolo, per il calcolo delle B e C ($\pm$ e $1, 2$) servirebbero molte più derivate (derivate rispetto alle variabili di stato fino al terzo ordine per le funzioni di soglia, derivate fino al secondo ordine per le forzanti, derivate temporali fino al secondo ordine per le forzanti e derivate temporali fino al secondo ordine per i gradienti di tutte le funzioni di soglia). Si è scelto di non passare tali dati come funzioni esatte al costruttore, ma di approssimarle con i dati che si hanno attraverso un metodo del quarto ordine a differenze finite implementato in `DFGradGamma`, `DFDgammaDt`, `DFdFidt` (rispettivamente per il gradiente di γ, per le derivate temporali di soglie e le derivate temporali delle forzanti su soglia) e con differenze finite per derivate seconde per l'hessiano di γ in `DFHessGamma`. La scelta di approssimare numericamente tali derivate ha lo scopo di rendere il codice più facilmente utilizzabile dall'utente a cui non deve essere richiesto un numero troppo elevato di dati in input per ogni caso. Inoltre si sottolinea che tale approssimazione con tecniche di ordine elevato è accurata e che viene fatta unicamente se le condizioni del secondo ordine sono necessarie (se capita che qualche A risulti nulla). Si aggiunge che tali funzioni di approssimazione vengono utilizzate unicamente per il calcolo di B e C di cui, ai fini del metodo, è necessario solo conoscere il segno o eventualmente il valore nullo.

Sono fornite inoltre funzioni ausiliarie come `setODEOneStep` per cambiare, nel caso di metodo multistep, il metodo ad un passo usato ai primi passi e `setNewtonPar` per cambiare i parametri di Newton nel caso di metodi impliciti che si rifanno a quelli della classe `Solver` della libreria precedente. Per un trattamento opportuno dei metodi multistep si sottolinea che in corrispondenza di una transizione di soglia si cambia il risolutore e quindi se questo è di tipo multipasso viene caricata solo l'ultima soluzione e reinizializzato il metodo stesso.

Nel caso di *sliding motion* su una soglia attiva le forzanti corrispondenti devono essere definite come combinazione lineare dei campi coinvolti come definito in (1.5). Per poter trattare anche questo caso tramite `Functor` come descritto precedentemente si è implementata una classe che deriva da `Functor` (come `RealFunctor` e `ParsedFunctor`) che si chiama `ThresholdFunctor`. Essa carica i dati necessari e fa l'override del `GetValue` di `Functor` in modo tale da definire α in modo corretto ed eseguire la combinazione lineare richiesta. Si noti come il motivo per cui

si usano unicamente solutori espliciti per casi in cui ci si muove su soglia attiva è che in caso contrario servirebbero i gradienti di tali `ThresholdFunctor` da passare al risolutore. Questo vorrebbe dire o passare le derivate corrispondenti da file che sarebbero però in numero molto elevato e non idoneo all'utilizzabilità del codice o approssimarle numericamente. L'ipotesi di approssimarle numericamente è stata esclusa in quanto introdurre un errore numerico sulla valutazione di tali derivate da usare nel metodo di Newton per l'avanzamento in tempo avrebbe peggiorato notevolmente le prestazioni del metodo numerico ed avrebbe reso vano l'utilizzo di un metodo implicito per diminuire l'errore numerico derivante dall'integrazione.

1.4 Risultati

In questo paragrafo vengono riportati alcuni casi test svolti al fine di verificare la correttezza dell'ordine degli schemi di integrazione implementati e per mostrare la maggior efficacia nei casi di sistemi DRH della libreria specifica rispetto agli schemi per ODE classici.

1.4.1 Casi test per libreria ODE

Si mostrano di seguito due problemi risolti con la libreria per sistemi di equazioni differenziali ordinarie classiche con lo scopo di verificare il corretto comportamento degli schemi implementati.

Il primo caso test consiste in una singola equazione differenziale ordinaria

$$\begin{cases} \dfrac{d}{dt}x(t) = (1-2t)x \\ x(0) = 1 \end{cases}$$

la cui soluzione esatta è:

$$x(t) = e^{t-t^2}.$$

Conoscendo la soluzione esatta è stato possibile verificare l'ordine di convergenza degli schemi implementati. Come si può vedere in Fig. 1.2, i risultati sono in accordo con la teoria.

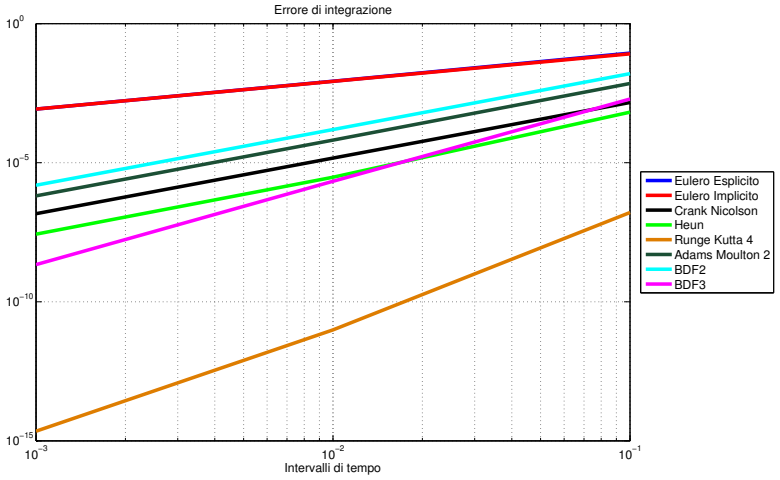


Figura 1.2: Errore di integrazione al variare di Δt.

Il secondo caso test, tratto da [2] rappresenta un semplice caso di interazione fra tre specie chimiche (A_1, A_2, A_3). Si considerano le seguenti reazioni:

$$\begin{aligned} 2A_1 &\rightarrow A_2 + A_3 \\ A_2 &\rightarrow A_3 \\ 2A_2 &\rightarrow A_1 + A_3. \end{aligned}$$

Nell'ipotesi che la velocità di reazione sia fornita dalla legge di Arrhenius e dalla legge dell'azione di massa [9] l'evoluzione in tempo delle specie chimiche può essere descritta dal seguente sistema di ODE non lineari: che possono essere riscritte sotto forma di un sistema di ODE non-lineari:

$$\begin{cases} \frac{d}{dt}x_0(t) = -2{x_0}^2 + {x_1}^2 \\ \frac{d}{dt}x_1(t) = {x_0}^2 - x_1 - 2{x_1}^2 \\ \frac{d}{dt}x_2(t) = {x_0}^2 + x_1 + 2{x_1}^2 \end{cases} \tag{1.8}$$

La soluzione ottenuta con lo schema di Eulero Implicito e un passo di integrazione $\Delta t = 0.01$ è riportata in Fig. 1.3.

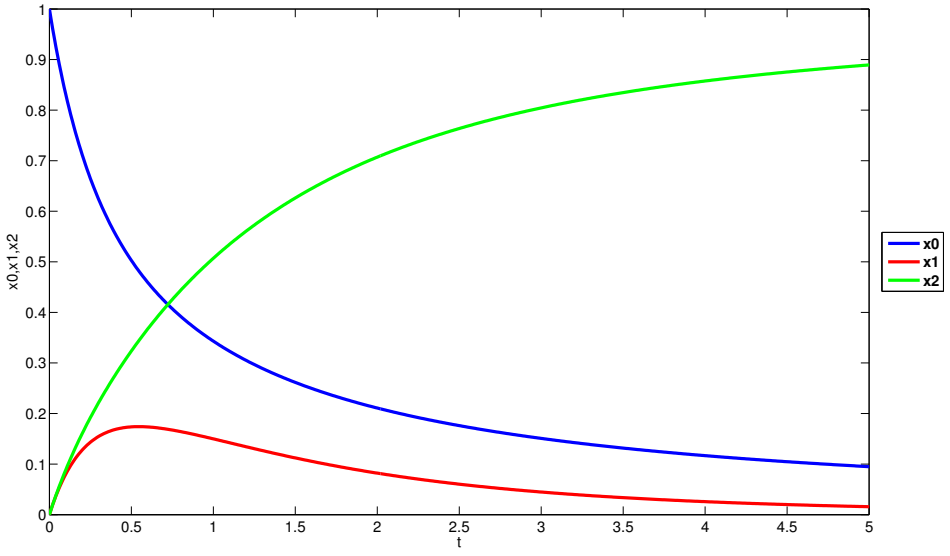


Figura 1.3: Soluzione di (1.8).

1.4.2 Casi test per libreria ODE DRH

In questa sezione verranno per prima cosa presentati due casi test che evidenziano i punti di forza della libreria per sistemi di ODE con forzanti discontinue.

Nel primo caso si considera un'equazione differenziale DRH che presenta una *crossing solution*:

$$\frac{dx}{dt} = \begin{cases} x, & \text{se } x(t) - 2.125 < 0 \\ \dfrac{x}{2}, & \text{se } x(t) - 2.125 > 0 \end{cases}$$
$$x(0) = 1$$

Tale problema è stato risolto con la libreria per ODE classiche e con la libreria ODE DRH per evidenziare la necessità di trattare correttamente la discontinuità del termine forzante. In entrambi i casi sono stati utilizzati gli schemi di Heun e Runge-Kutta con entrambe le librerie e sono stati confrontati gli errori. Come si può vedere in Fig. 1.4 la libreria di ODE non preserva gli ordini di convergenza dell'errore a causa della discontinuità del termine forzante dei campi in $x(t) = 2.125$ mentre quella di ODE DRH grazie alla ricerca esatta del punto di transizione integra correttamente e rispetta gli ordini in accordo con la teoria.

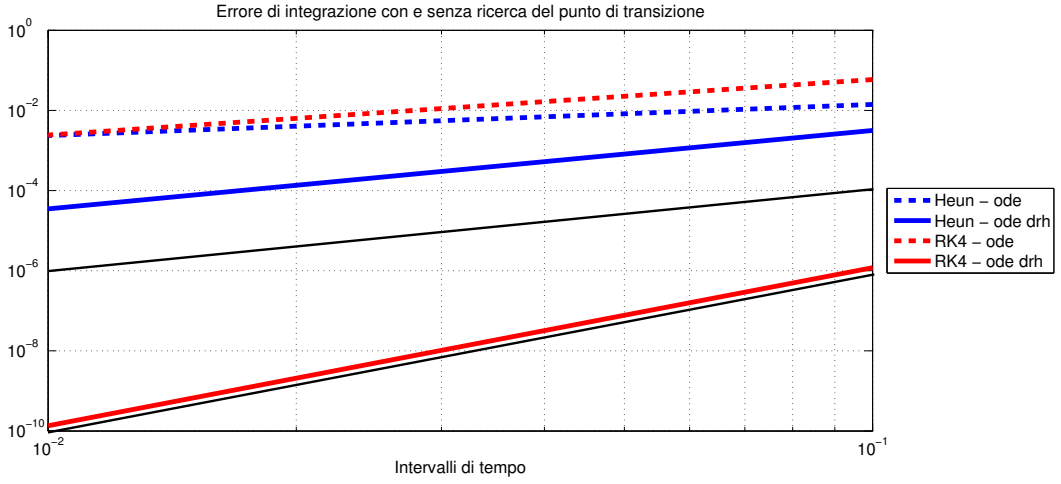


Figura 1.4: Errore di integrazione al variare di Δt. In nero l'ordine di convergenza teorico di Heun (2) e Runge-Kutta (4).

Logicamente infittendo molto la discretizzazione temporale si può riuscire a risolvere esattamente il punto di transizione anche con metodi classici. Per evitare di usare ovunque un Δt troppo piccolo e, conseguentemente, far crescere eccessivamente il costo computazionale della risoluzione del problema è possibile effettuare una ricerca locale e diminuire il passo temporale solo in corrispondenza del punto di transizione sulla soglia.
Questa tecnica permetterebbe di conservare l'ordine di convergenza dello schema di integrazione classico, ma solo nel caso di *crossing solution*; infatti qualora si verifichi una situazione diversa, come uno *sliding motion*, o dovessero rendersi necessarie le condizioni di ordine superiore risulta indispensabile l'utilizzo dei metodi qui sviluppati che sono in grado di determinare il comportamento della soluzione in corrispondenza dei punti di transizione. Si consideri ad esempio il seguente sistema a due equazioni presentato in [8]:

$$\mathbf{f_1} = \begin{bmatrix} x_1 \\ -x_0 + \dfrac{1}{1.2 - x_1} \end{bmatrix} \quad \mathbf{f_2} = \begin{bmatrix} x_1 \\ -x_0 - \dfrac{1}{0.8 + x_1} \end{bmatrix}$$

$$\mathbf{g} = \begin{bmatrix} 2 \\ x_1 - 0.2 \end{bmatrix}$$

$$\mathbf{x_0} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

La soluzione ottenuta con la libreria di ODE DRH, dopo un transitorio iniziale, incontra l'unica soglia attiva per il problema e la attraversa; successivamente il campo oltre la soglia appena attraversata riporta la soluzione sulla linea di discontinuità lungo la quale avviene uno *sliding motion* finché non si verifica la condizione di uscita e non entra in un ciclo infinito come mostrato in Fig. 1.5. Invece, se si cerca di risolvere lo stesso problema con la libreria di ODE si rilevano oscillazioni legate allo *sliding motion*. Nelle Figg. 1.6 e 1.7 sono state messe a confronto le soluzioni ottenute con libreria di ODE DRH e ODE (con passo temporale di un decimo inferiore per quest'ultima) nelle regioni in cui la soluzione entra ed esce dalla linea di discontinuità. In questo tipo di situazioni le oscillazioni, seppur piccole, rimangono anche riducendo il passo temporale Δt quindi, per ottenere una soluzione di buona qualità risulta necessario effettuare la ricerca del punto di transizione e tenere in considerazione le condizioni del primo ordine di Filippov se non addirittura quelle di ordine superiore.

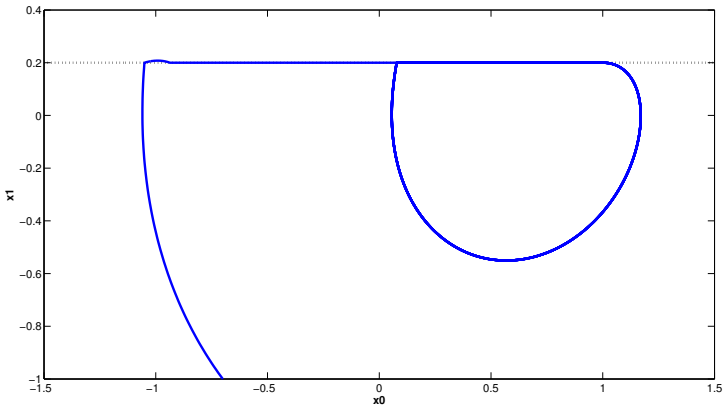


Figura 1.5: Traiettoria nel piano delle fasi, in tratteggio la superficie di discontinuitá.

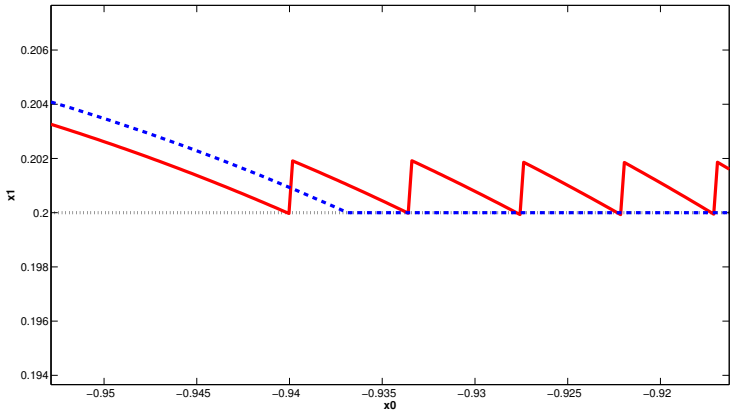


Figura 1.6: Dettaglio all'ingresso dello *sliding motion*; ODE(rosso), ODE DRH(blu).

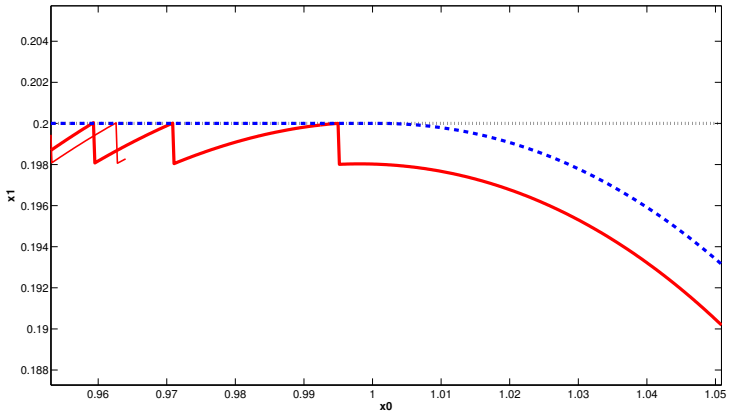


Figura 1.7: Dettaglio all'uscita dello *sliding motion*; ODE(rosso), ODE DRH(blu).

In conclusione si riporta un caso trattato in [2]. Il problema, che può essere considerato come un modello semplificato di un set di reazioni chimiche, è il seguente:

$$\begin{cases} \dfrac{d\mathbf{x}}{dt} = \mathbf{q} \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases}$$

con $\mathbf{q} = [-2{x_0}^2 + {x_1}^2, {x_0}^2 - x_1 - 2{x_1}^2, {x_0}^2 + x_1 + 2{x_1}^2]^T$ e $\mathbf{x}_0 = [1, 0, 0]^T$. Un set di reazioni chimiche può essere descritto da equazioni differenziali discontinue se accoppiato a fenomeni di ritenzione quali ad esempio l'adsorbimento nel caso del problema di generazione di idrocarburi. In questo tipo di processi le molecole generate possono essere intrappolate dalla nanoporosità della roccia fino ad una massima quantità che indichiamo con $z_i(t, \mathbf{x})$ per la specie i-esima. L'evoluzione della concentrazione x_i di ogni specie sarà quindi data dalla velocità di reazione "naturale" q_i al di sotto della massima soglia di ritenzione, e vincolata dalla soglia stessa per $x_i \geq z_i$. Il sistema di equazioni che descrivono i processi di reazione e ritenzione si può quindi

scrivere, per il caso in esame a tre componenti, come

$$\begin{cases} \dfrac{dx_i}{dt} = q_i & \text{se } x_i < z_i \\ \dfrac{dx_i}{dt} = \dfrac{dz_i}{dt} & \text{se } x_i > z_i \\ x_i(0) = x_{0i} \end{cases} \tag{1.9}$$

con $i = 0, 1, 2$ e $\mathbf{z} = x_0[1.1, 0.4, 4]^T$. Si noti che in questo semplice caso la soglia dipende esclusivamente dalla concentrazione della prima componente. Tutte le componenti inizialmente si trovano sotto le rispettive soglie. La variabile x_1 cresce fino ad incontrare la soglia corrispondente ad un quarto della variabile x_0 a $t \approx 0.72$ e si ha che $A_1 > 0$ e $A_2 = 0$ che implica che il campo in S_2 è tangente alla soglia; sono quindi necessarie le condizioni del secondo ordine: B_2^- risulta essere nulla e quindi dalla teoria si avrà uno *sliding motion*. In seguito, per $t \approx 1.36$, A_1 cambia di segno mentre A_2 rimane nullo come anche i termini di ordine superiore nello sviluppo della componente normale alla soglia per il campo 2, quindi $B_2 = 0$ e $C_2 = 0$ e si verificano le condizioni di uscita. Successivamente anche x_2 a $t \approx 2.28$ incontra la linea di discontinuità data da $x_2 < 4x_0$, dal momento che si verificano le stesse identiche condizioni discusse per la variabile x_1 anche in questo caso si ha uno *sliding motion*. La soluzione, che richiede le condizioni di ordine superiore e quindi la valutazione di A, B e C, è riportata in Fig. 1.8 ed è stata ottenuta integrando con passo $\Delta t = 0.01$ il problema (1.9) con un metodo di Adams-Moulton nelle regioni S_1 ed S_2 e Runge-Kutta 4 sulla superficie di discontinuità.

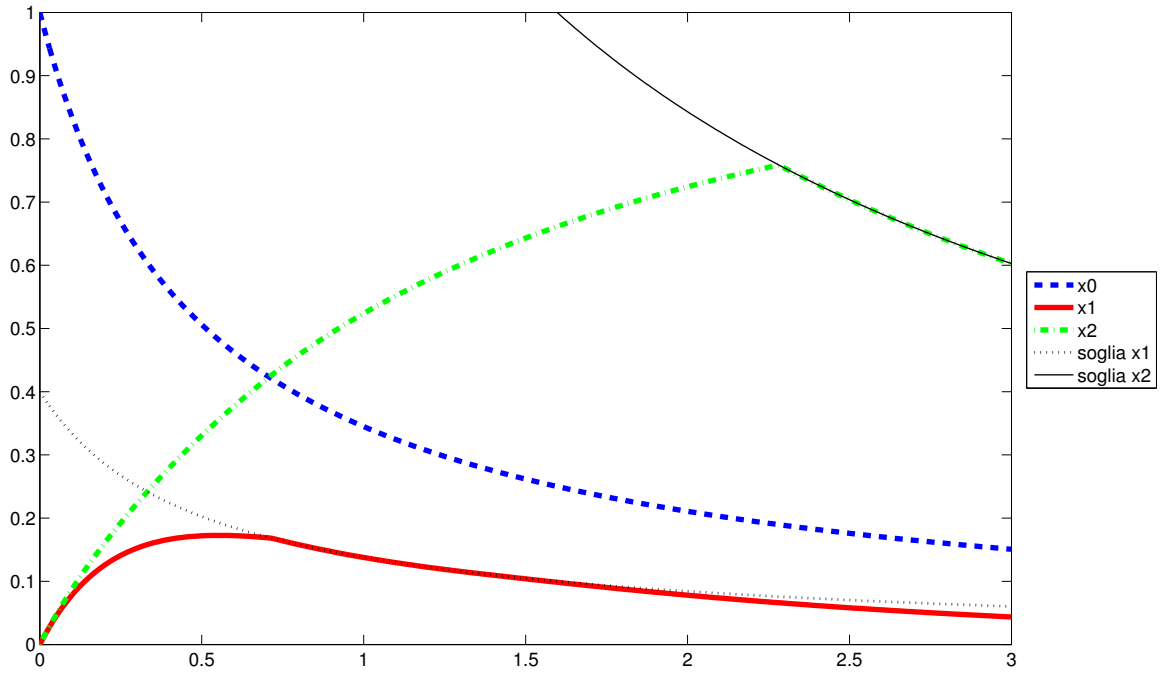


Figura 1.8: Soluzione numerica di (1.9).

La componente x_0 comanda di fatto l'andamento della reazione, la propria evoluzione dipende da sé stessa e dalla componente x_1 mentre le soglie delle altre due dipendono esclusivamente

da essa. Le componenti x_1 e x_2 evolvono liberamente fino ad incontrare le superfici di discontinuità definite da x_0, a questo punto i loro rispettivi termini sorgente saranno identici a quelli della prima componente e quindi tenderanno a farle diminuire fino a farle ritornare sotto soglia. Queste superfici di discontinuità e le rispettive correzioni dei campi di forzanti possono descrivere fenomeni di ritenzione di specie di idrocarburi nel sottosuolo e quindi risultano di particolare interesse in applicazioni di carattere geologico.

Capitolo 2

Operator Splitting per equazioni di diffusione trasporto e reazione

In questo capitolo si procede allo studio di tecniche di splitting per equazioni di diffusione trasporto e reazione col fine particolare di poter separare il contributo dei primi due dall'ultimo. Lo scopo è quello di poter trattare problemi che possono presentare termini di reazione, lineari o nonlineari, discontinui utilizzando tecniche adatte quali i metodi descritti nel Capitolo 1, per poter ottenere soluzioni accurate preservando l'ordine di accuratezza del metodo di integrazione. Vengono presentate le tecniche di splitting di Yanenko e Strang che hanno un'accuratezza in tempo rispettivamente del primo e del secondo ordine. In seguito si indagano due aspetti determinanti nell'utilizzo corretto di queste tecniche: le condizioni al contorno dei sottoproblemi splittati e i metodi di integrazione in tempo usati per risolverli. Per quanto riguarda i dati al bordo si pone l'attenzione sulla necessità di correggerli negli step intermedi dello splitting al fine di risolvere correttamente il problema. Per quanto riguarda gli schemi di integrazione si riporta un risultato di inconsistenza associato all'utilizzo di metodi multistep per la risoluzione dei passi dello splitting.

2.1 Problemi non stazionari di diffusione trasporto e reazione

Si considerano problemi della forma seguente:

$$\begin{cases} \dfrac{\partial u}{\partial t} - \operatorname{div}(\mu \nabla u) + \mathbf{b}\cdot \nabla u + \sigma u = f, & \text{in } Q_T := (0,T)\times\Omega \\ Bu = g, & \text{su } \Sigma_T := (0,T)\times\partial\Omega \\ u = u_0, & \text{su } \Omega \text{ per } t = 0 \end{cases} \tag{2.1}$$

dove μ, σ, f e $\mathbf{b}$ sono funzioni assegnate, l'operatore di bordo B denota una generica condizione al bordo per u (Dirichlet, Neumann, Robin, miste) [25]. Si assuma che Ω sia un dominio limitato in $\mathbb{R}^d$, $d = 2, 3$, con bordo Lipschitziano. Per semplicità si prendono in considerazione condizioni al bordo di Dirichlet. Sia $V = H_0^1(\Omega)$, $V_g = \{u \in H^1(\Omega), u|_{\Sigma_T} = g\}$ e $(\cdot,\cdot)$ il prodotto scalare in $L^2(\Omega)$. Introducendo la forma bilineare $a : V \times V \mapsto \mathbb{R}$,

$$a(u,v) = \int_\Omega \mu \nabla u\cdot \nabla v \, d\Omega + \int_\Omega v\mathbf{b}\cdot \nabla u \, d\Omega + \int_\Omega \sigma u v \, d\Omega \quad \forall u \in V_g, v \in V$$

e dati $f \in L^2(Q_T)$, $u_0 \in L^2(\Omega)$, la formulazione debole del problema (2.1) diviene: trovare $u \in L^2(0,T;V_g) \cap C^0([0,T];L^2(\Omega))$ tale che

$$\begin{cases} \dfrac{d}{dt}(u(t),v) + a(u(t),v) = (f(t),v), \quad \forall v \in V \\ u(0) = u_0. \end{cases}$$

2.2 Operator Splitting

L'operator splitting è una tecnica piuttosto diffusa per l'integrazione in tempo di problemi che presentano un operatore differenziale complesso. Originariamente è nato come tecnica per trasformare operatori multidimensionali in una somma di operatori monodimensionali, attualmente è usato soprattutto per separare i termini associati a diversi fenomeni fisici quali ad esempio trasporto, diffusione e reazione con lo scopo di poter risolvere ogni singolo sottoproblema con il metodo numerico più adatto. Il problema originario risulta così semplificato, al prezzo dell'introduzione di un errore, detto di "splitting", dipendente dal passo temporale di integrazione e da possibili instabilità. Per i problemi di nostro interesse si intende separare il termine di reazione da quelli di diffusione e trasporto, per poter approssimare con metodi numerici ad hoc il termine di reazione, che può essere non-lineare ed eventualmente discontinuo in tempo e nelle variabili di stato. L'idea principale consiste nel dividere il problema originario in due o più equazioni separando gli operatori. Le equazioni vengono poi risolte sequenzialmente, in modo che, ad ogni step temporale, la soluzione di un sottoproblema diventi la condizione iniziale per il successivo.
Si consideri un problema della forma

$$\begin{cases} \dfrac{du}{dt} = F(u) \quad x \in \Omega \\ u(t_0) = u_0, \end{cases} \tag{2.2}$$

dove $F(u)$ indica un operatore differenziale che può essere splittato nei due contributi di diffusione e trasporto (F_{AD}) e reazione (F_R):

$$\begin{cases} \dfrac{du}{dt} = F_{AD}(u) + F_R(u) \quad x \in \Omega \\ u(t_0) = u_0. \end{cases}$$

Per prima cosa si descrive uno splitting di Yanenko del primo ordine che consiste nel creare ad ogni time step (t^n, t^{n+1}) due sottoproblemi risolti sequenzialmente e connessi per mezzo delle condizioni iniziali. Questo comporta che il problema originario (2.2) venga approssimato risolvendo numericamente i due sottoproblemi:

$$\frac{du^*}{dt} = F_{AD}(u^*) \qquad \text{con } t \in (t^n, t^{n+1}) \text{ e } u^*(t^n) = u^n_{sp}, \tag{2.3}$$

$$\frac{du^{**}}{dt} = F_R(u^{**}) \quad \text{con } t \in (t^n, t^{n+1}) \text{ e } u^{**}(t^n) = u^*(t^{n+1}), \tag{2.4}$$

dove u^n_{sp} è la soluzione del problema splittato al passo precedente. Si pone $u^{n+1}_{sp} = u^{**}(t^{n+1})$.

Possono essere costruiti anche splitting di ordine superiore. In particolare uno splitting del secondo ordine, noto come splitting di Strang, può essere costruito ripetendo gli step descritti precedentemente sulla prima metà di ogni intervallo temporale, quindi integrando i due step scambiati di ordine nella seconda metà [28], [16], ottenendo:

$$\begin{aligned}
&\frac{du^*}{dt} = F_R(u^*) && \text{con } t \in (t^n, t^{n+1/2}) \text{ e } u^*(t^n) = u^n_{sp},\\
&\frac{du^{**}}{dt} = F_{AD}(u^{**}) && \text{con } t \in (t^n, t^{n+1}) \text{ e } u^{**}(t^n) = u^*(t^{n+1/2}),\\
&\frac{du^{***}}{dt} = F_R(u^{***}) && \text{con } t \in (t^{n+1/2}, t^{n+1}) \text{ e } u^{***}(t^{n+1/2}) = u^{**}(t^{n+1}),
\end{aligned}$$

e ponendo $u^{n+1}_{sp} = u^{***}(t^{n+1})$.

Abbiamo scelto di eseguire due step di reazione ed uno di diffusione-trasporto (schema RDR), ma è anche possibile eseguire due step di diffusione ed uno di reazione (DRD). Utilizzando un operator splitting possiamo scegliere come integrare in tempo ciascuno dei singoli step. In particolare, dato che il termine di reazione non dipende dallo spazio se non attraverso eventuali coefficienti può essere risolto semplicemente come un'ODE, o un sistema di ODE in ogni singolo nodo, cella o in generale grado di libertà del dominio. Splitting di ordine superiore al secondo possono essere ottenuti, per esempio, usando splitting iterativi, come illustrati in [16]. Tuttavia per gli obiettivi di questo lavoro ci si è limitati alle tecniche citate.

2.2.1 Correzione delle condizioni al contorno

Un aspetto particolarmente rilevante legato all'utilizzo dei metodi di operator splitting è la determinazione delle condizioni al contorno da imporre negli step intermedi. In molti studi relativi ai metodi di splitting applicati ad equazioni di avvezione-diffusione-reazione la strategia adottata consiste nell'adottare, anche nei sottoproblemi intermedi, le condizioni al bordo del problema non splittato. Tale scelta, motivata dalla difficoltà di determinare condizioni prive di un significato fisico per i singoli step dello splitting, non è tuttavia applicabile se le condizioni al bordo sono tempo-dipendenti, se non al prezzo di un decadimento dell'ordine di convergenza [26]. È possibile modificare le condizioni al bordo del problema originario in modo rigoroso per ottenere condizioni corrette che permettono di ottenere la stessa soluzione del problema originario preservando l'ordine di convergenza dello splitting.

Si consideri ad esempio un problema di avvezione-diffusione-reazione con condizioni al contorno di Dirichlet e si ipotizzi di splittarlo in due sottoproblemi di avvezione-diffusione e di reazione. Il problema di diffusione e trasporto risente ovviamente dalle condizioni al bordo. Se nello step dello splitting ad esso associato vengono imposte le esatte condizioni di Dirichlet del problema non splittato si sta in realtà imponento un dato al bordo che contiene informazioni legate anche al contributo reattivo. In questo modo la soluzione del problema di diffusione e trasporto risentirà di questa inconsistenza sull'intero dominio e di conseguenza la soluzione finale, al termine dello step di reazione, sarà diversa da quella del problema originale non splittato.

Al fine di determinare le condizioni al contorno intermedie per i vari passi dello splitting si consideri, senza perdita di generalità, il seguente problema monodimensionale scalare lineare

di diffusione e reazione:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial x^2} + Au, \quad x \in (0,1),\ t > 0 \\ u(x,0) = u_0(x), \quad x \in (0,1) \\ u(0,t) = f(t) \\ u(1,t) = g(t), \end{cases} \tag{2.5}$$

con A costante. Usando il metodo di separazione delle variabili si può ricavare la soluzione analitica nel modo seguente. Ridefinendo

$$u = U + V$$

con

$$V(x,t) = f(t)(1-x) + g(t)x$$

si ha che U soddisfa un'equazione di diffusione e reazione con condizioni di Dirichlet al bordo omogenee la cui soluzione analitica è:

$$U(x,t) = \sum_{n=1}^{\infty} C_n(t)\Phi_n(x) \quad \text{con } \Phi_n(x) = \sin(n\pi x).$$

I coefficienti C hanno la seguente espressione

$$C_n(t) = 2e^{(A-n^2\pi^2)t}\int_0^1 U(x,0)\sin(n\pi x)dx + \int_0^t G_n(s)e^{(A-n^2\pi^2)(t-s)}ds$$

con

$$U(x,0) = u_0(x) - f(0)(1-x) - g(0)x$$

$$G_n(t) = \frac{1}{n\pi}\Big[Af(t) + (-1)^{n+1}Ag(t) - f'(t) + (-1)^n g'(t)\Big].$$

Si consideri ora uno splitting del primo ordine applicato al problema (2.5):

$$\begin{cases} \dfrac{\partial u^D}{\partial t} = \dfrac{\partial^2 u^D}{\partial x^2}, \quad x \in (0,1),\ t > 0 \\ u^D(x,0) = u_0(x), \quad x \in (0,1) \\ u^D(0,t) = F(t) \\ u^D(1,t) = G(t), \end{cases}$$

$$\begin{cases} \dfrac{du}{dt} = Au, \quad t > 0 \\ u(x,0) = u^D(x,\Delta t), \quad x \in (0,1), \end{cases}$$

dove $F(t)$ e $G(t)$ sono le condizioni al bordo intermedie opportunamente corrette. Detto Δt il passo di discretizzazione temporale con cui verrà approssimato numericamente il problema, si può ripetere il procedimento illustrato precedentemente per ricavare le soluzioni analitiche dei due sottoproblemi in ogni intervallo temporale Δt in cui si avanza con la tecnica di splitting. Dato che la tecnica di splitting del primo ordine considerata prevede di avanzare in entrambi gli step da t^n a t^{n+1} si può assumere senza perdita di generalità che i due sottoproblemi (2.3)

e (2.4) abbiano come istante iniziale 0. Il secondo step dello splitting avrà come condizione iniziale la soluzione al tempo Δt del primo step.
Sia $u^D = U^D + V^D$ con

$$V^D(x,t) = F(t)(1-x) + G(t)x$$

si ha che U^D soddisfa un'equazione di diffusione e reazione con condizioni di Dirichlet al bordo omogenee la cui soluzione analitica è:

$$U^D(x,t) = \sum_{n=1}^{\infty} C_n^D(t)\Phi_n(x) \quad \text{con } \Phi_n(x) = \sin(n\pi x).$$

I coefficienti C^D hanno la seguente espressione

$$C_n^D(t) = 2e^{-n^2\pi^2 t}\int_0^1 U^D(x,0)\sin(n\pi x)dx + \int_0^t G_n^D(s)e^{-n^2\pi^2(t-s)}ds$$

con

$$U^D(x,0) = u_0(x) - F(0)(1-x) - G(0)x$$

$$G_n^D(t) = \frac{1}{n\pi}\big[-F'(t) + (-1)^n G'(t)\big],$$

mentre per lo step di reazione la soluzione analitica sarà:

$$U^R(x,t) = e^{At}u^D(x,t), \tag{2.6}$$

che sarà anche la soluzione dell'intero splitting dato che lo step di reazione è l'ultimo nel caso dello splitting del primo ordine che stiamo considerando.

Tale soluzione è ottenuta risolvendo sull'intervallo (t^n, t^{n+1}) il sottoproblema di diffusione con le condizioni al contorno corrette, e utilizzando la soluzione come condizione iniziale per l'ODE (anch'essa risolta sullo stesso intervallo temporale) che descrive la parte reattiva dell'operatore originario. Imponendo che le soluzioni del problema splittato (2.6) e di quello completo (2.5) siano identiche ogni volta che si avanza in tempo di Δt si ricava che le condizioni al contorno per la parte diffusiva vanno corrette nel seguente modo:

$$F(t) = e^{-A\Delta t}f(t), \tag{2.7}$$

$$G(t) = e^{-A\Delta t}g(t). \tag{2.8}$$

Questo tipo di correzione del dato al bordo consiste di fatto nell'"annullare"il contributo reattivo nell'intervallo di tempo Δt prima di applicare l'operatore di diffusione. Benché l'operatore di reazione agisca solo all'interno del dominio le condizioni al contorno del sottoproblema di diffusione sono determinanti perché da esse dipende la soluzione in tutto il dominio. L'unica equazione a richiedere delle condizioni sul bordo è quella di diffusione, ma, come mostrato, si cadrebbe in errore se nel sottoproblema ad essa associato si imponessero le condizioni di Dirichlet del problema non splittato. Questo problema non si verifica nel caso di condizioni di Neumann perché la soluzione al bordo non è più un dato imposto e anche sulla frontiera del dominio si può applicare l'operatore di reazione.

La stessa trattazione appena fatta risulta molto meno banale nel caso dello splitting di Strang. Non è possibile trovare una correzione del dato al bordo tale che la soluzione analitica del problema splittato sia identica a quella del problema non splittato. Tuttavia applicando alle

condizioni al bordo un termine di ritardo come quello presente in (2.7) e (2.8) e proporzionale alla lunghezza del passo di splitting si è verificato sperimentalmente che è possibile preservare l'ordine di accuratezza pari a 2.

Come sarà verificato con esperimenti numerici nella sezione dei Risultati questa correzione permette di conservare l'ordine di convergenza anche nel caso di condizioni al contorno tempo-dipendenti.

2.2.2 Splitting con metodi multipasso

All'interno di uno splitting, ogni step può essere integrato con un metodo di avanzamento in tempo a scelta. Nel caso si vogliano utilizzare splitting di ordine elevato diventa quindi necessario integrare ogni singolo sottoproblema con uno schema di avanzamento in tempo di ordine maggiore o uguale a quello dello splitting. Una possibile scelta sono i metodi multi-step. Tuttavia in questa sezione mostreremo che l'utilizzo di metodi multistep all'interno di operator splitting può introdurre errori di consistenza di ordine Δt o addirittura di ordine 1 e quindi limitare l'ordine di convergenza dello splitting, e in alcuni casi impedirla. Dal momento che questo accade anche quando gli operatori F_{AD} e F_R sono lineari e costanti nel tempo ci limiteremo ad analizzare questo semplice caso.

Ripercorrendo il ragionamento proposto in [15] si ipotizzi di risolvere esattamente il sotto-problema di reazione (2.4) e di applicare un metodo multistep della forma (2.9) a quello di diffusione e avvezione (2.3). Otteniamo che la soluzione intermedia dello step di avvezione-diffusione è data, nell'intervallo nell'intervallo (t_n, t_{n+1}), da:

$$u^*_{n+1} + \sum_{j=1}^{k} \alpha_j u_{n+1-j} = \Delta t \beta_0 F_{AD} u^*_{n+1} + \Delta t \sum_{j=1}^{k} \beta_j F_{AD} u_{n+1-j}, \tag{2.9}$$

dove u_{n+1-j}, $j = 2, ..., k$ sono le soluzioni precedenti, approssimazioni dell'intera equazione non splittata. Ipotizzando di utilizzare come u_{n+1-j} la soluzione esatta di (2.3) si ha che la formula

$$\widetilde{u}^*_{n+1} + \sum_{j=1}^{k} \alpha_j u_{n+1-j} = \Delta t \beta_0 F_{AD} \widetilde{u}^*_{n+1} + \Delta t \sum_{j=1}^{k} \beta_j F_{AD} u_{n+1-j}$$

fornisce una soluzione $\widetilde{u}^*_{n+1}$ che al passo $n+1$ approssima u^*_{n+1} con una precisione di $O(\Delta t^{p+1})$, dove p sono i passi del metodo usato. Per valutare l'errore locale con (2.9) si suppone di partire con gli esatti valori $u_k = u(t_k)$, $k \leq n-1$ di

$$\frac{du}{dt} = F_{AD}(u) + F_R(u) \quad x \in \Omega, \quad u(t_n) = u_n.$$

Usando $u^*(t_n) = u(t_n)$ segue che

$$\|u^*(t) - u(t)\| = O(t_n - t) \max_{t \leq s \leq t_n} \|F_R u(s)\|$$

per $t_{n+1-k} \leq t \leq t_n$. Poiché

$$(I - \Delta t \beta_0 F_{AD})(\widetilde{u}^*_{n+1} - u^*_{n+1}) = \sum_{j=1}^{k} (-\alpha_j + \Delta t \beta_j F_{AD})(\widetilde{u}^*(t_{n+1-j}) - u_{n+1-j}),$$

si ottiene la stima dell'errore locale

$$\|\widetilde{u}^*_{n+1} - u^*_{n+1}\| = O(\Delta t^\nu) \max_{t \le t_n} \|F_R u(t)\|$$

con $\nu = 2$ se $\alpha_j = 0$ $(j = 2, ..., k)$ e $\nu = 1$ altrimenti. Questo significa che se utilizziamo metodi multistep del tipo Backward differentiation formula (BDF) otteniamo un ordine di convergenza globale pari a zero, e pari al massimo a uno nel caso si utilizzino metodi di Adams. Ovviamente la stessa perdita nell'ordine si ottiene scambiando l'ordine dei sottoproblemi splittati e indipendentemente da quale dei due viene risolto con un metodo multistep.

Abbiamo quindi mostrato che anche utilizzando tecniche di splitting sequenziale di ordine elevato (non trattate in questo lavoro) la convergenza sarebbe compromessa dalle problematicità di consistenza legate all'utilizzo di metodi di Adams o BDF. Nella sezione dei Risultati si mostreranno test di convergenza che confermano sperimentalmente questi risultati teorici. É quindi opportuno non usare metodi multistep con gli splitting sequenziali; se si vuole ottenere alti ordini di convergenza si devono usare metodi ad un passo come ad esempio i Runge-Kutta.

2.3 Risultati

In questa sezione presenteremo i risultati di convergenza relativi alle tecniche di splitting descritte.

Si consideri il seguente problema evolutivo di diffusione e reazione:

$$\begin{cases} \dfrac{\partial u}{\partial t} - \mu \Delta u + cu = 0 & \text{in } \Omega = (-1,1) \times (-0.05, 0.05), \quad t \in (0,1), \\ u(-1, y; t) = u(1, y; t) = e^t & \forall y \in (-0.05, 0.05), \\ u(x, y; 0) = x^2, & \text{in } \Omega = (-1,1) \times (-0.05, 0.05), \end{cases} \tag{2.10}$$

dove $\mu = \dfrac{1}{4}x^2$ e $c = -\dfrac{1}{2}$. La soluzione esatta è $u_{ex}(x, y; t) = x^2 e^t$. Si osservi che (2.10) costituisce un caso particolare del problema generale (2.1) in cui il campo di trasporto è nullo e il termine di reazione è lineare in u. Vogliamo separare con un operator splitting il termine di diffusione ed il termine di reazione.

Per la soluzione numerica del sottoproblema di diffusione si utilizza il metodo degli elementi finiti di secondo grado $\mathbb{P}^2$ implementato nella libreria `LifeV`, con lo schema di avanzamento in tempo Crank-Nicolson. In questo modo, dato che la soluzione dipende quadraticamente dalla variabile spaziale, l'errore dovuto alla discretizzazione spaziale è nullo, quindi l'errore calcolato in norma L^2 dipende solo dall'integrazione in tempo e dallo splitting adottato. Per la soluzione numerica del sottoproblema di reazione si utilizza la libreria presentata nel Capitolo 1 e lo stesso schema di avanzamento in tempo. Consideriamo innanzitutto uno splitting del primo ordine di tipo diffusione reazione (DR) che consiste nella soluzione in cascata, ad ogni istante di tempo, del problema di diffusione

$$\begin{cases} \dfrac{\partial u^*}{\partial t} - \mu \Delta u^* = 0, & \text{in } \Omega, \quad t \in (t^n, t^{n+1}) \\ u^*(-1, y; t^{n+1}) = u^*(1, y; t^{n+1}) = e^{t^{n+1} + c\Delta t} \\ u^*(x, y; t^n) = \begin{cases} x^2 & \text{se } t^n = 0 \\ u^n_{sp} & \text{altrimenti} \end{cases} \end{cases}$$

dove u^n_{sp} è la soluzione del problema splittato al passo precedente, e poi di quello di reazione

$$\begin{cases} \dfrac{\partial u^{**}}{\partial t} + cu^{**} = 0, & t \in (t^n, t^{n+1}) \\ u^{**}(t^n) = u^*(t^{n+1}). \end{cases}$$

Al variare del passo di discretizzazione Δt osserviamo, dal grafico di convergenza riportato in Fig. 2.1, che, nonostante i metodi di avanzamento in tempo utilizzati per i due sottoproblemi siano del secondo ordine, globalmente si ottiene una convergenza del primo ordine a causa dell'errore di splitting.

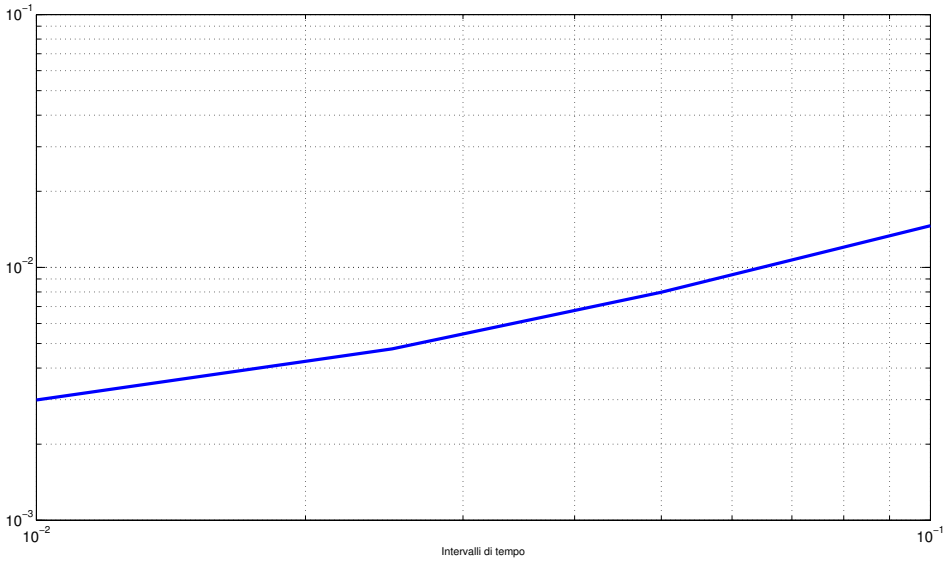


Figura 2.1: Errore dello splitting di ordine 1 con Crank-Nicolson.

Per ottenere una convergenza del secondo ordine è necessario utilizzare uno splitting almeno del secondo ordine, come lo splitting di Strang. In particolare per lo splitting di Strang si è provato ad eseguire i passi, nell'ordine, di reazione diffusione e reazione (RDR), ma anche di diffusione reazione diffusione (DRD), ottenendo i risultati mostrati nelle Figg. 2.3 e 2.2. Si osservi che nel problema in esame le condizioni al contorno di Dirichlet sono tempo-dipendenti. È stato quindi necessario utilizzare nella soluzione dei singoli sottoproblemi condizioni al contorno opportunamente corrette come discusso precedentemente. In particolare per lo splitting di Strang si è provato ad eseguire i passi, nell'ordine, di diffusione reazione diffusione (DRD),

riportati di seguito:

$$\begin{cases} \dfrac{\partial u^*}{\partial t} - \mu \Delta u^* = 0, & \text{in } \Omega, \quad t \in (t^n, t^{n+1/2}) \\ u^*(-1, y; t^{n+1/2}) = u^*(1, y; t^{n+1/2}) = e^{t^{n+1/2} + \frac{c}{2}\Delta t} \\ u^*(x, y; t^n) = \begin{cases} x^2 & \text{se } t^n = 0 \\ u^n_{sp} & \text{altrimenti} \end{cases} \end{cases}$$

$$\begin{cases} \dfrac{\partial u^{**}}{\partial t} + c u^{**} = 0, & t \in (t^n, t^{n+1}) \\ u^{**}(t^n) = u^*(t^{n+1/2}). \end{cases}$$

$$\begin{cases} \dfrac{\partial u^{***}}{\partial t} - \mu \Delta u^{***} = 0, & \text{in } \Omega, \quad t \in (t^{n+1/2}, t^{n+1}) \\ u^{***}(-1, y; t^{n+1}) = u^{***}(1, y; t^{n+1}) = e^{t^{n+1}} \\ u^{***}(x, y; t^{n+1/2}) = u^{**}(t^{n+1}) \end{cases}$$

così come quelli di reazione diffusione e reazione (RDR). Si può osservare nei grafici delle Figg. 2.3 e 2.2 che l'utilizzo di condizioni al contorno non adeguate degrada l'ordine di convergenza teorico dello splitting, che passa dal secondo al primo ordine.

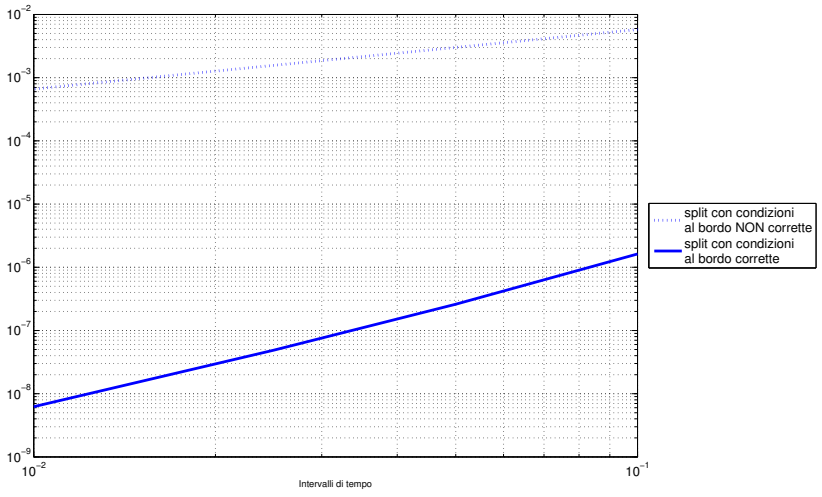


Figura 2.2: Errore dello splitting di Strang DRD con Crank-Nicolson.

Per sottolineare l'importanza di correggere le condizioni al contorno negli step intermedi dello splitting si riportano alcuni grafici dell'andamento dell'errore per il problema (2.10) splittato con la tecnica del primo ordine DR. In Fig. 2.4 sono riportati gli errori $|u_h - u_{ex}|$ lungo una sezione l'intero dominio ad $y = 0$ in tre diversi istanti temporali. Come si può notare l'errore di approssimazione è massimo ai bordi del dominio nel caso in cui si usino le condizioni al contorno non corrette. Diversamente in Fig. 2.5 sono riportati gli errori negli stessi istanti di tempo e ottenuti con la stessa tecnica di splitting, ma correggendo le condizioni al contorno. Appare

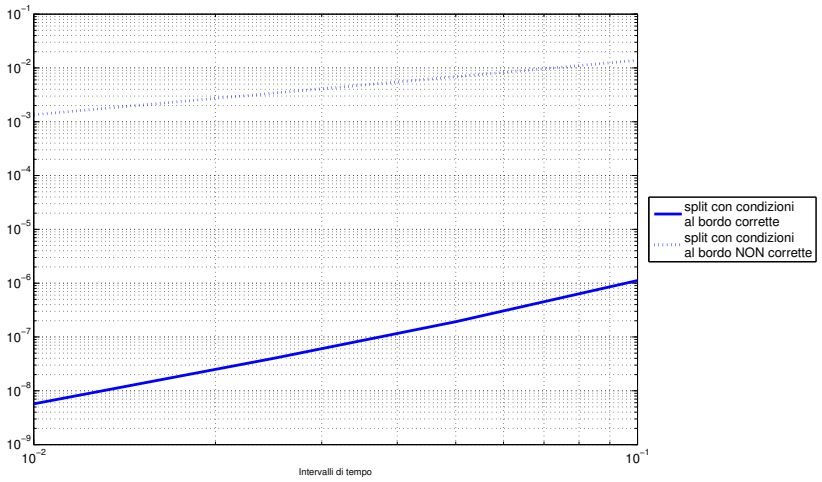


Figura 2.3: Errore dello splitting di Strang RDR con Crank-Nicolson.

evidente che la correzione delle condizioni al bordo per gli step intermedi risulta determinante nella riduzione dell'errore di approssimazione, in particolare agli estremi del dominio.

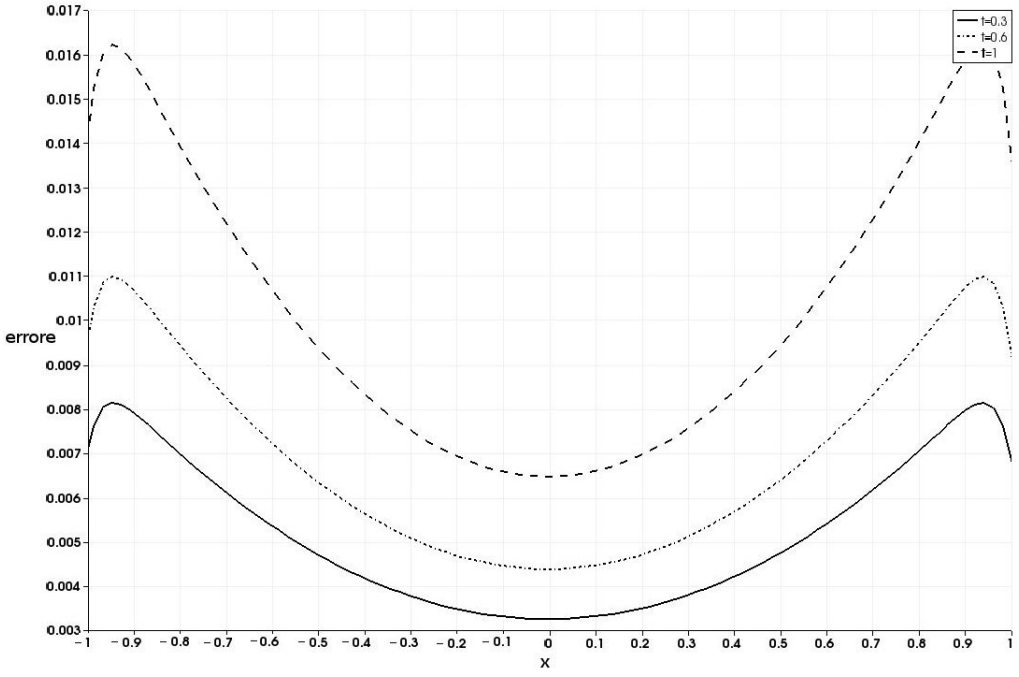


Figura 2.4: Errore per il problema (2.10) senza condizioni al bordo corrette nella sezione $y = 0$ del dominio.

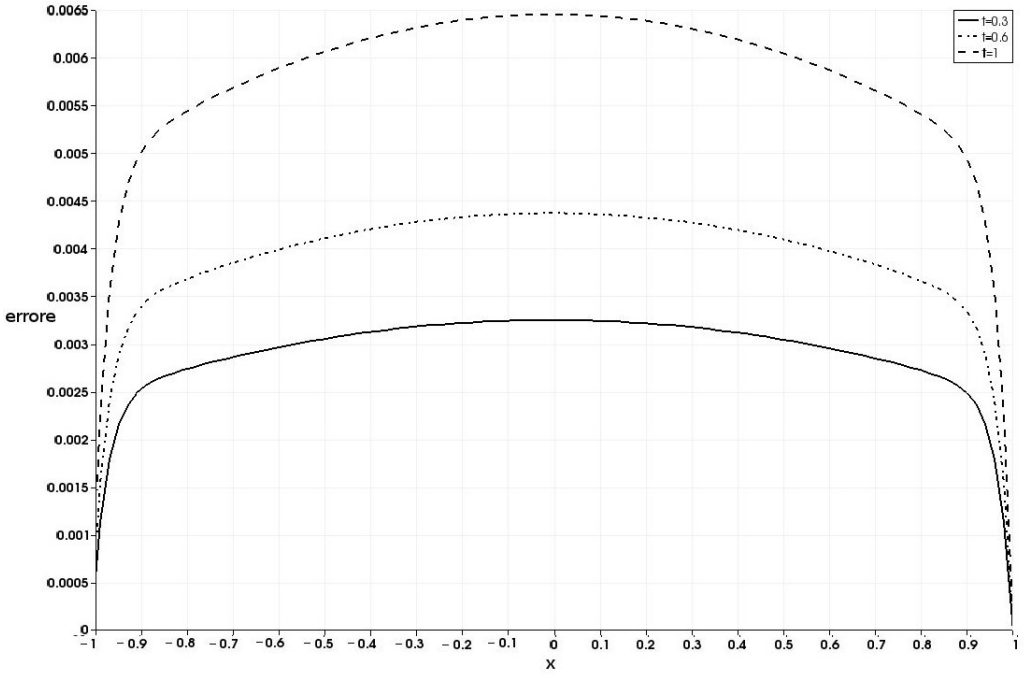


Figura 2.5: Errore per il problema (2.10) con condizioni al bordo corrette nella sezione $y = 0$ del dominio.

Se non si correggono le condizioni di Dirichlet per il sottoproblema di diffusione e si avanza di Δt la soluzione vicino ai bordi di Ω risente di un dato al contorno che contiene anche informazioni legate all'operatore di reazione. Su questi stessi nodi (che sono interni) viene applicato nuovamente l'operatore di reazione, di conseguenza la soluzione dello splitting risulta sbagliata e l'errore massimo è concentrato vicino ai bordi del dominio. La strategia correttiva quindi consiste nell'"annullare"la reazione prima di eseguire lo step di diffusione.

Infine si riportano i risultati dei test di convergenza sugli splitting per cui sono stati utilizzati metodi multistep di Adams e BDF per l'integrazione dei sottoproblemi di reazione e Crank-Nicolson per diffusione e avvezione. In particolare si considerino Adams-Bashforth 3 e BDF 2 con lo splitting di ordine 1. Nella Tabella sono riportati gli errori al tempo finale in norma L^2. Con il metodo di Adams si ottiene un ordine di convergenza di circa 1 mentre BDF 2 risulta addirittura inconsistente confermando il risultato teorico discusso nella sezione precedente.

	$\Delta t = 0.1$	$\Delta t = 0.05$	$\Delta t = 0.01$
Adams-Bashforth 3	0.0311541	0.0153817	0.00422612
BDF 2	0.0355376	0.635172	...

Capitolo 3

Precipitazione e dissoluzione in mezzo poroso

In questo capitolo si considera il problema di un flusso in mezzo poroso in presenza di ioni disciolti in acqua mossi dall'azione combinata di trasporto e diffusione e in grado di reagire. Tali modelli si incontrano in molti casi reali come lo spargimento di agenti chimici e la conseguente contaminazione di falde acquifere, applicazioni biologiche come la formazione di tessuti ed ossa, applicazioni farmaceutiche o il funzionamento di batterie a stato solido.

Ci si concentrerà su un modello alla macroscala, ossia definito alla scala di Darcy: si considera quindi un mezzo continuo in cui i granuli solidi ed i pori non vengono risolti bensì omogeneizzati su di un volume di riferimento, e le equazioni saranno definite ovunque nel dominio di interesse.

Siamo interessati a studiare l'interazione del processo di diffusione e trasporto con le reazioni chimiche responsabili dei processi di precipitazione e dissoluzione degli ioni (anioni e cationi). Tali reazioni portano alla trasformazione degli ioni disciolti in specie immobili, con la formazione di cristalli (precipitazione), e viceversa (dissoluzione).

Si è considerato il modello presentato in [17] e consistente in un'equazione di trasporto diffusione e reazione non lineare e discontinua per gli ioni disciolti accoppiata con un'equazione differenziale ordinaria per l'evoluzione del precipitato. La mancanza di regolarità nel termine di reazione può portare a soluzioni inaccurate nel caso si utilizzino metodi di integrazione in tempo classici. Una possibile soluzione, utilizzata in [17] è la regolarizzazione della discontinuità: tale tecnica verrà messa a confronto con i metodi descritti nel Capitolo 1 e implementati nella libreria ODE-DRH, in grado di gestire equazioni differenziali discontinue grazie alla localizzazione esatta dell'istante di transizione. Tale irregolarità nel termine di reazione può essere gestita con un'approssimazione dipendente da un opportuno parametro o per mezzo della libreria di ODE DRH; entrambe le tecniche verranno testate e messe a confronto.

3.1 Il modello

In questa sezione si presenta e discute la derivazione del modello di dissoluzione e precipitazione in mezzo poroso, proposto in [20] ed utilizzato in [17]. Si considerino due specie di soluti M_1 ed M_2 che nel nostro caso sono rispettivamente cationi ed anioni disciolti in acqua. In aggiunta a queste specie si definisce il solido cristallino $\overline{M}_{12}$ presente nel mezzo. M_1 ed M_2 possono

precipitare e formare $\overline{M}_{12}$ e viceversa il cristallo può dissolversi. La reazione chimica alla base di questo processo può essere descritta nel seguente modo:

$$\overline{M}_{12} \rightleftarrows nM_1 + mM_2, \tag{3.1}$$

dove n ed m sono le valenze di M_1 ed M_2, supposte positive. Poichè M_1 ed M_2 sono specie ioniche la reazione (3.1) descrive il processo che porta alla neutralizzazione della carica elettrica del fluido in cui le specie sono dissolte.
Si definisce con $\mathbf{u} = [u_1, u_2, v]$ le concentrazioni molari di M_1, M_2 e $\overline{M}_{12}$, rispettivamente in $[mmoli/cm^3]$. L'obiettivo è quello di ricavare le equazioni di conservazione della massa. La geologia sottostante e le proprietà del flusso d'acqua sono descritte dal tensore di diffusione/-dispersione $\mathbf{D}$ $[cm^2/s]$ e dal vettore di trasporto $\mathbf{q}$ $[cm/s]$. Si assume che $\mathbf{D}$ sia lo stesso per entrambe le specie. Dal momento che $\overline{M}_{12}$ è una specie immobile il principio di conservazione delle rispettive masse totali conduce alle seguenti equazioni a derivate parziali:

$$\frac{\partial}{\partial t}u_1 + n\frac{\partial}{\partial t}v - \nabla \cdot (\mathbf{D}\nabla u_1 - \mathbf{q}u_1) = 0, \tag{3.2}$$

$$\frac{\partial}{\partial t}u_2 + m\frac{\partial}{\partial t}v - \nabla \cdot (\mathbf{D}\nabla u_2 - \mathbf{q}u_2) = 0. \tag{3.3}$$

Si passa ora alla descrizione della reazione chimica. Siano r_d ed r_p $[mmoli/(s \cdot cm^3)]$ i tassi rispettivamente di dissoluzione e precipitazione, si avrà che

$$\frac{\partial}{\partial t}v = r_p - r_d. \tag{3.4}$$

Generalmente (si veda [20]) si assumono come vere le seguenti proprietà:

- il tasso di dissoluzione r_d del cristallo è costante, cioè

  $$r_d = k_d \quad \text{se } v > 0;$$

- il tasso di precipitazione è dato dalla legge di azione di massa

  $$r_p = k_p r(\mathbf{u}),$$

  dove r nel caso della legge di azione di massa termodinamicamente ideale è:

  $$r(\mathbf{u}) = u_1^n u_2^m.$$

  Si assume che:

  1. $r_p(\cdot) : \mathbb{R} \to [0, \infty)$ è localmente Lipschitz continua in $\mathbb{R}$.
  2. Esiste un unico $u_* \geq 0$ tale che

     $$r_p(u) = \begin{cases} 0 \quad \text{per } u \leq u_*, \\ \text{strettamente crescente per } u \geq u_* \quad \text{con } r(u) \to \infty, \text{ per } u \to \infty. \end{cases}$$

In presenza di precipitato cristallino ($v > 0$) la soluzione si dice satura quando il soluto è sciolto in quantità massima nel solvente e quindi non è possibile scioglierne di più, in queste condizioni il soluto raggiunge la concentrazione massima nella soluzione. Tale condizione, considerando il caso in esame e la (3.4), si ha quando $r_p = r_d$ e quindi

$$r(\mathbf{u}) = k_d/k_p.$$

Le condizioni

$$r(\mathbf{u}) \gtrless k_d/k_p$$

indicano rispettivamente sopra(sotto)-saturazione. Per includere il caso in cui $v = 0$ si deve estendere la definizione del tasso di dissoluzione. Per farlo si assume che valgano, per un generico $v \geq 0$, le seguenti proprietà di una situazione d'equilibrio:

- si possono verificare solo condizioni di saturazione o sottosaturazione;
- se il solido cristallino è presente si ha saturazione;
- in condizioni di sottosaturazione non può essere presente solido cristallino.

Queste ipotesi si traducono nelle seguenti condizioni:

$$0 \leq r \leq \frac{k_d}{k_p}$$

$$v > 0 \;\Rightarrow\; r = \frac{k_d}{k_p}$$

$$r < \frac{k_d}{k_p} \;\Rightarrow\; v = 0,$$

che corripondono a v composto con una funzione proporzionale alla Heaviside $H(\cdot)$ moltiplicata per k_d/k_p:

$$r = \begin{cases} 0, & v \leq 0 \\ \dfrac{k_d}{k_p}, & v > 0. \end{cases}$$

Sia $\Omega \subset \mathbb{R}^2$ il dominio occupato dal mezzo poroso e si supponga che Ω sia aperto, connesso, limitato e poligonale con bordo Lipschitziano Γ. Sia $T > 0$ un tempo arbitrario fissato e $\partial\Omega = \Gamma = \Gamma_D \cup \Gamma_N$ con Γ_D il bordo di Dirichlet e Γ_N quello di Neumann, si definisce:

$$\Omega^T = (0, T] \times \Omega \quad \text{e} \quad \Gamma_{D,N}^T = (0, T] \times \Gamma_{D,N}.$$

Mentre nella realtà le reazioni chimiche avvengono tra cationi ed anioni, per semplicità nel modello considerato si studia una sola specie mobile, quella dei cationi. Detta v la concentrazione molare di precipitato (immobile) e u la concentrazione molare di cationi disciolti in acqua, adimensionalizzando le variabili il problema (3.2) può essere riformulato come:

$$\begin{cases} \dfrac{\partial}{\partial t}(u + v) + \nabla \cdot (\mathbf{q}u - \mu \nabla u) = 0 & \text{in } \Omega^T, \\ u = g & \text{su } \Gamma_D^T, \\ \nabla u \cdot \mathbf{n} = h & \text{su } \Gamma_N^T, \\ u = u_0 & \text{in } \Omega \text{ per } t = 0, \end{cases} \tag{3.5}$$

dove $\mathbf{q}$ rappresenta la velocità del fluido di Darcy, μ è il coefficiente di diffusione e g ed h opportune condizioni al bordo. In questo capitolo considereremo che $\mathbf{q}$ sia un campo noto a divergenza nulla, ipotesi che verrà rimossa nel Capitolo 4 con l'introduzione di un vero accoppiamento fra il modello di precipitazione e Darcy; si ha inoltre che

$$\nabla \cdot \mathbf{q} = 0 \quad \text{in} \quad \Omega, \ \ \forall t \in (0, T).$$

Inoltre il tasso di dissoluzione r_d in questo caso in cui si sono adimensionalizzate le variabili diventa semplicemente la composizione di v con la funzione di Heaviside $H(\cdot)$. Il precipitato, essendo una specie immobile, è soggetto solamente a reazione e non ha alcuna dipendenza spaziale quindi può essere descritto da

$$\begin{cases} \dfrac{\partial v}{\partial t} = r_p(u) - H(v) & \text{in } \Omega^T, \\ v = v_0 & \text{in } \Omega \text{ per } t = 0. \end{cases} \tag{3.6}$$

Si adottano le notazioni standard per l'analisi funzionale: con $(\cdot, \cdot)$ si indica il prodotto interno in $L^2(\Omega)$.

Definizione 2. *Dato* $\mathbf{q} \in H^1(\Omega)$*, con div* $\mathbf{q} = 0$ *in* Ω*, per quasi ogni* $t \in (0, T)$ *trovare*

$$\begin{aligned} &u(t) \in \left\{ v \in H^1(\Omega), v|_{\Gamma_D} = g(t) \right\}, \\ &v(t) \in L^2(\Omega), \end{aligned}$$

tale che

$$\begin{aligned} \big(\partial_t u(t) + \partial_t v(t), \phi\big) + \big(\nabla u(t), \nabla \phi\big) + \big(\mathbf{q} \nabla u(t), \nabla \phi\big) = 0, \quad \forall \phi \in H^1_{0,\Gamma_D}(\Omega) \\ \big(\partial_t v(t), \theta\big) - \big(r_p(u(t)) - H(v(t)), \theta\big) = 0, \quad \forall \theta \in L^2(\Omega). \end{aligned}$$

Essa può essere riformulata come

$$\begin{aligned} \big(\partial_t u(t), \phi\big) + \big(\nabla u(t), \nabla \phi\big) - \big(\mathbf{q} \nabla u(t), \nabla \phi\big) = \big(H(v(t)) - r_p(u(t)), \phi\big), \quad \forall \phi \in H^1_{0,\Gamma_D}(\Omega) \\ \big(\partial_t v(t), \theta\big) = \big(r_p(u(t)) - H(v(t)), \theta\big), \quad \forall \theta \in L^2(\Omega). \end{aligned}$$

Si può dimostrare [17] che questo problema ha un'unica soluzione debole.

3.2 Risultati

In questa sezione si considera la risoluzione numerica del caso test proposto in [17] e illustrato nel paragrafo precedente. Si utilizzano le tecniche di splitting e i metodi per ODE descritti nel Capitolo 1 e nel Capitolo 2.

Si consideri il dominio

$$\Omega := (0,1) \times (0,1), \quad \Gamma_D := \{y : \ x = 0, \ y \in (0,1)\}, \quad \Gamma_N := \partial\Omega \setminus \Gamma_D$$

e i seguenti valori dei parametri

$$\mu = 1, \quad r_p(u) = u, \quad \mathbf{q} = 0.01 y(1-y)\mathbf{e_1}.$$

Si consideri l'intervallo temporale $t \in (0, 0.2)$. Per quanto riguarda le condizioni iniziali si definisce

$$\Omega_v \subset \Omega, \quad \Omega_v := \{(x, y) : \ 0.4 \leq x \leq 0.6, \ 0.4 \leq y \leq 0.6\}$$

e per $t = 0$ si assume $u|_{t=0} = 1$ ovunque in Ω e

$$v|_{t=0} = \begin{cases} 0.2 & \text{per } (x, y) \in \Omega_v, \\ 0 & \text{per } (x, y) \in \Omega \setminus \Omega_v. \end{cases}$$

Ai bordi del dominio sono state imposte condizioni di Neumann omogenee su tutti i lati del quadrato Ω tranne che su uno, dove vengono imposte condizioni di Dirichlet omogenee. Sotto questo tipo di dati ci si aspetta che si generi un processo di dissoluzione in cui la concentrazione di v decresce in Ω_v. Si noti che le suddette condizioni al contorno riguardano solo la variabile u poiché il comportamento di v è descritto da un'ODE quindi non ha alcuna dipendenza spaziale, non necessita di condizioni al bordo e viene risolta numericamente nodo per nodo.
Il problema è stato splittato con la tecnica del primo ordine (2.3) e (2.4) proposta nel Capitolo 2. Per la discretizzazione in spazio si è usata una mesh triangolare strutturata 100×100 ed elementi finiti $\mathbb{P}^1$. Per la discretizzazione temporale si possono discretizzare le derivate temporali e ottenere le approssimazioni (u^n, v^n) di $(u(t_n), v(t_n))$; si è usato uno schema di Eulero implicito con passo di discretizzazione $\Delta t = 0.001$ sia per lo step di diffusione e trasporto sia per lo step reattivo.
Lo stesso problema è stato risolto in [17] con la stessa risoluzione spaziale e temporale, ma utilizzando una tecnica di regolarizzazione per il termine di reazione. In particolare è stata proposta una regolarizzazione del tipo:

$$H_\delta(v) = \begin{cases} 0, & \text{se } v < 0, \\ v/\delta, & \text{se } 0 \leq v \leq \delta, \\ 1, & \text{se } v > \delta, \end{cases} \tag{3.7}$$

con $\delta = 0.1(\Delta t)^{\frac{1}{2}}$. Nelle Figg. 3.2,3.4 sono riportate le soluzioni ottenute in [17] mentre nelle Figg. 3.1 e 3.3 sono riportate le soluzioni ottenute risolvendo la reazione con la libreria di ODE-DRH. Entrambe si riferiscono all'istante finale $t = 0.2$.

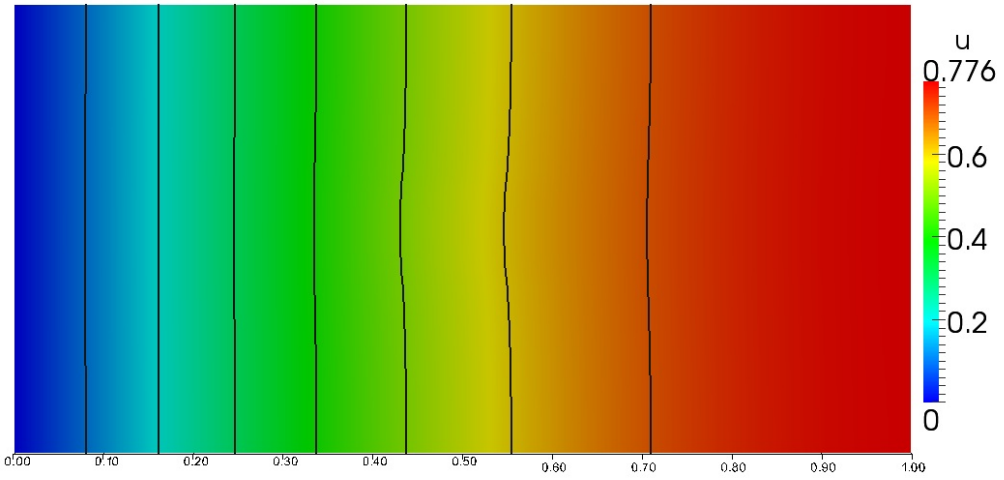


Figura 3.1: Concentrazione di cationi u ottenuta con libreria di ODE-DRH, $t = 0.2$.

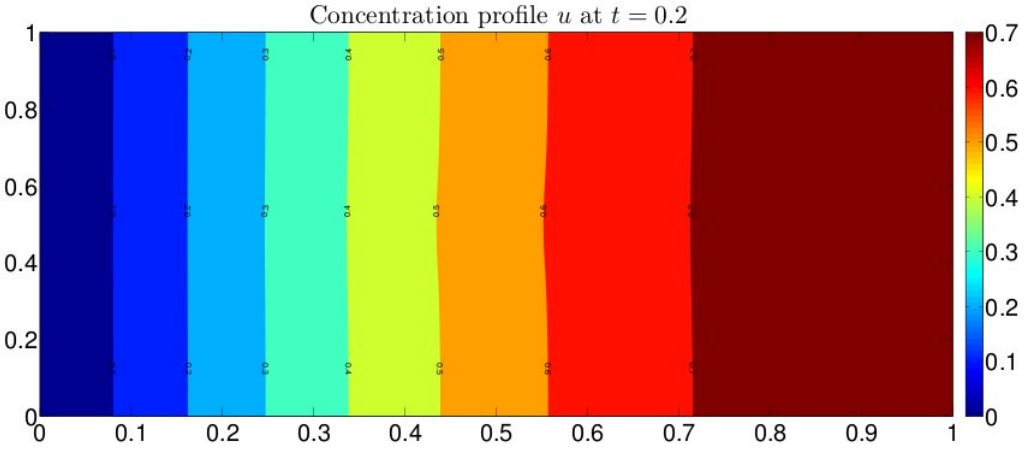


Figura 3.2: Concentrazione di cationi u riportata in [17], $t = 0.2$.

La soluzione presenta uno *sliding motion* attrattivo in corrispondenza della discontinuità data dalla funzione di Heaviside $H(v)$ in $\Omega \setminus \Omega_v$ di conseguenza la concentrazione di cristalli precipitati rimane costante e pari a 0 come nella condizione iniziale. All'interno di Ω_v lo stato iniziale $v = 0.2$ fa sì che non intervengano le condizioni di primo e secondo ordine descritte nella teoria di ODE-DRH poiché v non si trova sulla superficie di discontinuità e la funzione di Heaviside è attiva e pari a 1. Quindi in Ω_v la concentrazione del cristallo precipitato evolve secondo una normale ODE la cui forzante dipende da u e il processo di dissoluzione sarà tanto più rapido quanto più rapida sarà la riduzione di cationi disciolti in acqua. Come si può notare la concentrazione di cationi, che parte dalla condizione iniziale di $u = 1$ sull'intero dominio, all'aumentare del tempo t decresce a causa della condizione di Dirichlet omogenea imposta in

$x = 0$. Nella regione Ω_v in cui si deposita il precipitato si vede l'influenza di v sull'evoluzione di u; dal momento che $v \neq 0$ in Ω_v si attiva la funzione di Heaviside nel termine di reazione dell'equazione di ADR che descrive il comportamento di u. Ricordando la (3.6) si ha che il termine $\dfrac{\partial v}{\partial t}$ in (3.5) cambia di segno con l'attivazione della funzione di Heaviside e quindi il suo contributo si traforma assumendo caratteristiche di produzione anziché di distruzione.

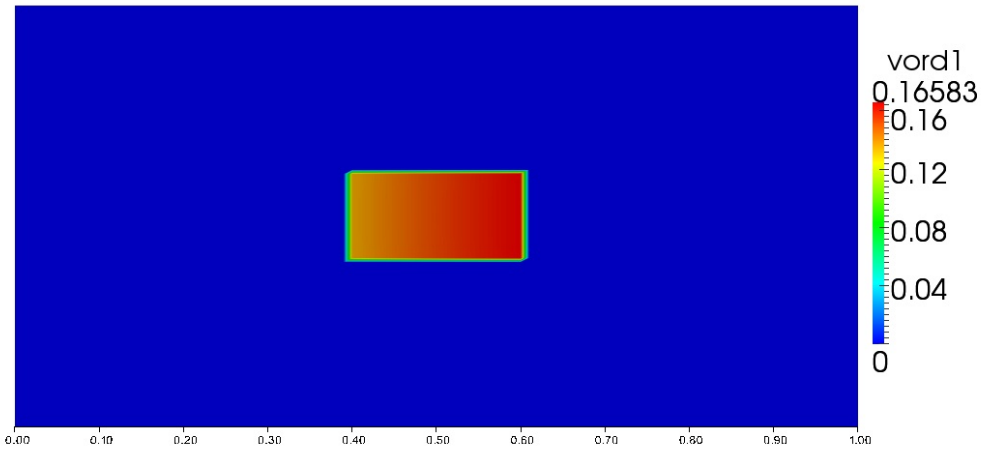


Figura 3.3: Concentrazione di precipitato v ottenuta con libreria di ODE-DRH, $t = 0.2$.

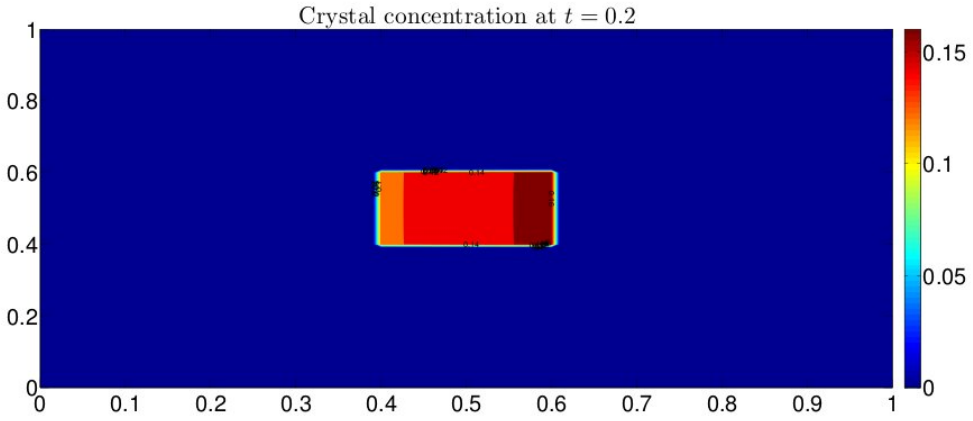


Figura 3.4: Concentrazione di precipitato v riportata in [17], $t = 0.2$.

Per visualizzare più chiaramente l'andamento di u e v sono stati riportati in Fig. 3.5 i rispettivi valori lungo la linea orizzontale passante per $y = 0.5$. La concentrazione di cristallo precipitato in Ω_v dopo 0.2 secondi si è ridotta rispetto alla condizione iniziale e dall'andamento si vede chiaramente la dipendenza da u. Minore è la concentrazione di cationi e maggiore è il decadimento del precipitato; nella regione di sinistra di Ω_v la concentrazione del cristallo è minore poiché u è minore e la precipitazione compensa meno il termine di distruzione dato dalla funzione di Heaviside.

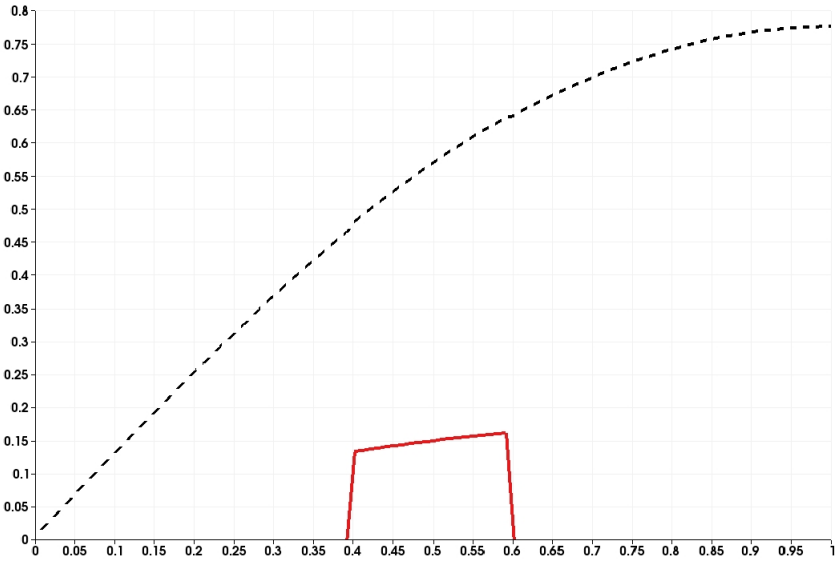


Figura 3.5: Concentrazione di precipitato v (rosso) e di cationi u (tratteggio nero), $t = 0.2$.

É stato anche eseguito un confronto tra la tecnica di regolarizzazione della Heaviside proposta in [17] e la libreria di ODE-DRH. La regolarizzazione in (3.7) si introduce inevitabilmente diffusione numerica. In Fig. 3.6 si possono vedere le concentrazioni di cristallo precipitato ottenute nei due diversi modi; per evidenziare le differenze la scala dei valori visualizzati è stata ristretta all'intervallo $v \in (0, 0.005)$. Nella figura di sinistra si vedono gli effetti della regolarizzazione della funzione di Heaviside mentre nella soluzione ottenuta con la libreria di ODE-DRH (in cui non è stata effettuata alcuna regolarizzazione) la soluzione è esattamente nulla in tutto $\Omega \setminus \Omega_v$ come effettivamente deve essere visto che si trova su di uno *sliding motion* attrattivo sulla superficie di discontinuità $v = 0$. Gli stessi effetti si possono vedere in Fig. 3.7, dove si mostrano le soluzioni numeriche ottenute per v a vari istanti di tempo lungo la sezione $y = 0.5$.

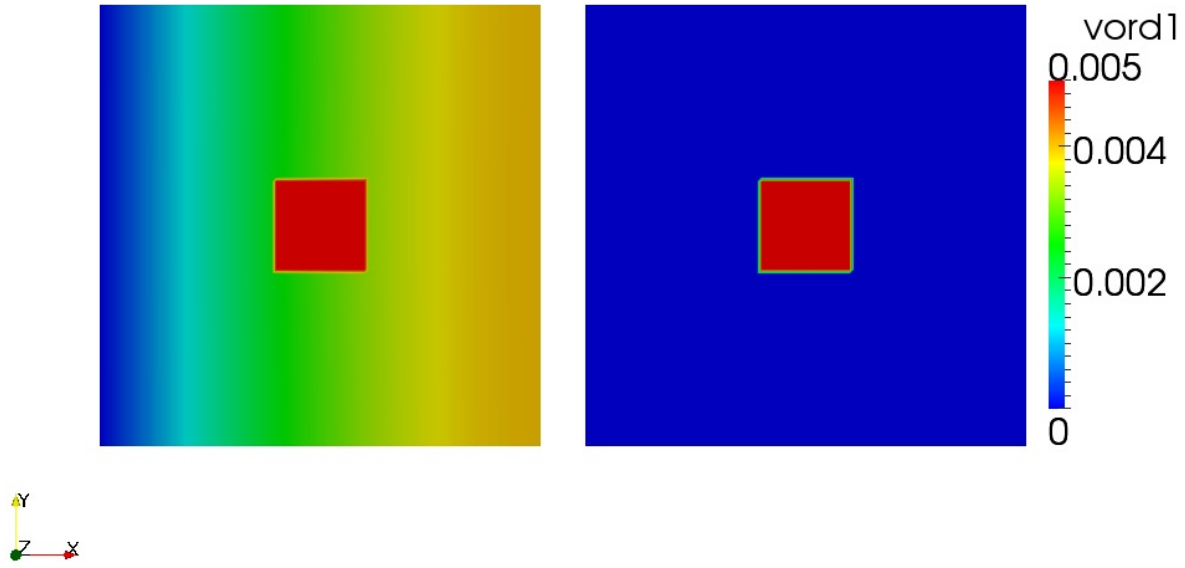
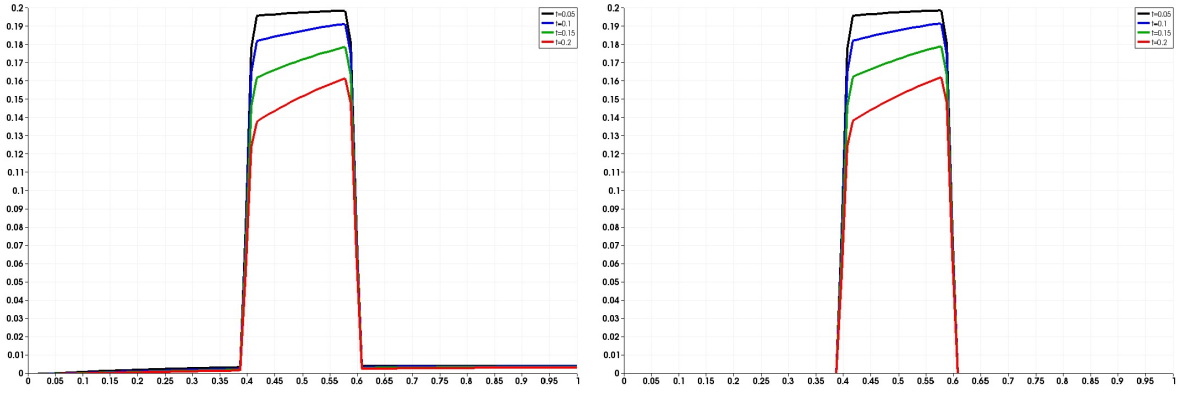


Figura 3.6: Concentrazione di precipitato v con Heaviside regolarizzata (sx) e con ODE-DRH (dx), $t = 0.2$.



(a) Soluzione ottenuta con Heaviside regolarizzata. (b) Soluzione ottenuta con libreria di ODE-DRH.

Figura 3.7: Concentrazione di precipitato v per $t = 0.05, 0.1, 0.15, 0.2$ lungo la sezione $y = 0.5$.

3.3 Dettagli sull'implementazione

Per facilitare l'utilizzo e possibili future estensioni della libreria diamo qui alcuni dettagli sulla implementazione. La risoluzione numerica del problema presentato è basata sulla libreria `LifeV` [10], [11], [22]. Seppure l'implementazione delle ODE-DRH supporta splitting di più alto ordine, in questa fase è stato implementato solo lo splitting del primo ordine. Ad ogni istante di tempo viene prima assemblato e risolto il problema di diffusione e trasporto, poi con la libreria ODE-DRH su ogni nodo viene risolta l'ODE a forzante discontinua che descrive la chimica del problema.

La maggior parte dei dati del problema sono stati impostati tramite un file `data` che viene letto dal programma tramite `GetPot`, un parser che analizza file di input estraendone i dati. In questo file sono presenti tutti i parametri necessari per la discretizzazione temporale (tempo iniziale, finale e passo di discretizzazione), per quella spaziale (dimensione e tipo di mesh), per il precondizionatore delle matrici, per il solutore e per l'esportazione dei risultati. Il termine sorgente, le condizioni iniziali e il campo di trasporto vengono impostati tramite opportune classi o funtori. La mesh viene creata tramite la classe `RegionMesh2DStructured` che riceve in input il numero di elementi nelle due direzioni x ed y e le lunghezze dei lati del dominio nelle rispettive direzioni e genera una mesh triangolare strutturata:

```
// Set up the structured mesh
regularMesh2D( *meshPtr, 0,
               dataFile( ( structuredSection + "nx" ).data(), 4 ),
               dataFile( ( structuredSection + "ny" ).data(), 4 ),
               dataFile( ( structuredSection + "verbose" ).data(), false ),
               dataFile( ( structuredSection + "lx" ).data(), 1. ),
               dataFile( ( structuredSection + "ly" ).data(), 1. ) );
```

In seguito la mesh viene partizionata su vari processori per mezzo di `MeshPartitioner` che richiama opportune routine di `ParMetis`. Nella classe `FESpace` sono invece presenti le informazioni circa lo spazio di approssimazione (grado dei polinomi), le formule di quadratura (`QuadraturRule`) sugli elementi e sulle entità $d-1$ dimensionali e alcune routine di interpolazione di funzioni sullo spazio; gli spazi ad elementi finiti nei quali vengono approssimate la soluzione del problema e il campo di trasporto $\mathbf{q}$ sono creati come puntatori ad oggetti della classe classe `FESpace` fornendo le informazioni sopra citate:

```
// Finite element space of the solution
const ReferenceFE*    refFE;
const QuadratureRule* qR;
const QuadratureRule* bdQr;

refFE = &feTriaP1;
qR    = &quadRuleTria4pt;
bdQr  = &quadRuleSeg1pt;

typedef FESpace< regionMesh_Type, MapEpetra > feSpace_Type;
typedef boost::shared_ptr< feSpace_Type > feSpacePtr_Type;
```

```
feSpacePtr_Type feSpacePtr( new FESpace< regionMesh_Type, MapEpetra >
       ( meshPart, *refFE, *qR, *bdQr, 1, Members->comm ) );
feSpacePtr_Type qfeSpacePtr( new FESpace< regionMesh_Type, MapEpetra >
       ( meshPart, *refFE, *qR, *bdQr, 2, Members->comm ) );
```

Le strutture dati utilizzate per memorizzare vettori e matrici del problema sono quelle di `Epetra`, un pacchetto di algebra lineare contenuto in `Trilinos` e utile sia per il calcolo seriale che parallelo.

```
typedef boost::shared_ptr<VectorEpetra> vectorPtr_Type;
typedef boost::shared_ptr<MatrixEpetra<Real>> matrixPtr_Type;

// Build FE matrices
matrixPtr_Type matM_ptr( new matrix_type( feSpacePtr->map() ) ); //mass
matrixPtr_Type matA_ptr( new matrix_type( feSpacePtr->map() ) ); //stiff

//Fe vectors
vector_type u( feSpacePtr->map()    , Unique ); // cation concentration
vector_type v( feSpacePtr->map()    , Unique ); // precipitate concentration
vector_type f( feSpacePtr->map()    , Unique ); // forcing term
vector_type q( betafeSpacePtr->map(), Unique ); // advection field
```

Le condizioni al contorno vengono impostate tramite la classe `BCManage` ed applicate ad ogni istante di tempo dalle sue routine:

```
// The boundary conditions
BCFunctionBase Zero( zero_scalar );
BCFunctionBase One( one_scalar );

BCHandler bc;

bc.addBC("Top"   ,    TOP,   Natural, Full, Zero, 1);
bc.addBC("Bottom", BOTTOM,   Natural, Full, Zero, 1);
bc.addBC("Left"  ,   LEFT, Essential, Full, Zero, 1);
bc.addBC("Right" ,  RIGHT,   Natural, Full, Zero, 1);

...

// Updating boundary condition
bc.bcUpdate(*(feSpacePtr->mesh()), feSpacePtr->feBd(), feSpacePtr->dof());

bcManage( *matA_ptr, f, *feSpacePtr->mesh(), feSpacePtr->dof(), bc,
          feSpacePtr->feBd(), tgv, t );
```

L'assemblaggio delle matrici per il problema di diffusione-trasporto-reazione viene fatto per mezzo della classe `ADRAssembler` che riceve in input la matrice e le aggiunge i termini corrispondenti alle discretizzazioni spaziali dei vari operatori del problema di diffusione-trasporto, ossia del primo step dello splitting di ordine 1:

```
// Build the ADRAssembler and the matrices
ADRAssembler<regionMesh_Type,matrix_type, vector_type> adrAssembler;

// Set up the ADRAssembler
adrAssembler.setup( feSpacePtr, qfeSpacePtr );

// Assemble A matrix
adrAssembler.addMass( matA_ptr, coeff );
adrAssembler.addDiffusion( matA_ptr, visc );
qfeSpacePtr->interpolate( qFct, q, 0. );
adrAssembler.addAdvection( matA_ptr, q );
```

Una volta assemblate le matrici e il termine noto si risolve il sistema usando ancora una libreria contenuta in `Trilinos`, che implementa il solutore iterativo `AztecOO`.

```
// Definition of the linear solver
SolverAztecOO az_A( Members->comm );
az_A.setDataFromGetPot( dataFile, "solver" );
az_A.setupPreconditioner( dataFile, "prec" );

//Set Up the linear system
az_A.setMatrix( *matA_ptr );

// Solve
az_A.solveSystem( f, u, matA_ptr );
```

Successivamente per risolvere lo step di reazione legato alla chimica del problema si utilizza la libreria di ODE-DRH. L'operatore di reazione, essendo un'equazione scalare, viene risolto nodo per nodo ciclando sugli indici globali della mesh e accertandosi (tramite la variabile booleana `isOwned`) che l'operazione non venga eseguita da più processori quando si lavora in parallelo. Dopo aver creato i vettori con le condizioni iniziali (`CI`), con i funtori delle forzanti (`ForcerDown`, `ForcerUp`), dei gradienti (`GradForcerDown`, `GradForcerUp`) e delle rispettive derivate in tempo (`dForcerDowndt`, `dForcerUpdt`) e analogamente con le superfici di discontinuità (`g`, `GradG`, `dGdt`) si costruisce l'oggetto della classe `OdeDrhSolver` che risolve il sistema di ODE con forzanti discontinue.

```
// Global indexes
std::vector<Int> elemID( ADRsplit_FESpacePtr->
     dof().globalElements(*ADRsplit_FESpacePtr->mesh()) );

for ( UInt i = 0 ; i < elemID.size() ; i++ )
{
    bool isOwned ( u.map().map(Unique)->LID(elemID[i]) > -1. ? true : false );
    if ( isOwned )
    {
        // Construction of the ODE-DRH object
        OdeDrhSolver ChemistrySolver( MetImplicitEuler, t-delta_t, t, delta_t,
```

```
                                2, CI, ForcerDown, ForcerUp, GradForcerDown,
                                GradForcerUp, dForcerDowndt, dForcerUpdt,
                                g, GradG, dGdt, outputfilename );

        // Solve
        ChemistrySolver.DoIt();
    }
}
```

Capitolo 4

Accoppiamento dei modelli di Darcy e di precipitazione-dissoluzione

In questo capitolo si introduce un modello per il flusso in mezzi porosi basato sull'equazione di Darcy con lo scopo di completare il problema di precipitazione/dissoluzione introdotto nel capitolo precedente fornendo un'espressione per la velocità di avvezione degli ioni disciolti, inizialmente considerata assegnata e costante in tempo. Si inizia con la definizione di mezzo poroso e delle sue proprietà; successivamente si introduce il modello di Darcy per la descrizione del flusso e trasporto di un fluido a singola fase definito alla macroscala all'interno di un mezzo poroso. Infine il modello ricavato viene accoppiato al modello di precipitazione-dissoluzione presentato nel capitolo precedente.

4.1 Flussi monofase in mezzi porosi

In questa sezione si presentano le equazioni che descrivono in flusso a singola fase in un mezzo poroso. Si definisce fase una porzione di un sistema chimicamente omogenea e separata dalle altre porzioni da bordi fisici ben definiti. Nel caso di sistemi monofase si considera il moto di un unico fluido, oppure diversi fluidi che siano però perfettamente miscibili gli uni con gli altri. Nel caso di nostro interesse si considera l'acqua come unico fluido presente nel sistema. Un mezzo poroso è un corpo composto da una parte (detta anche matrice) solida e da spazio vuoto che può essere riempito da uno o più fluidi. Il rapporto fra volume dello spazio vuoto e matrice è detto porosità. La modellazione del flusso che utilizziamo (modello di Darcy) è valida sotto le condizioni seguenti:

- lo spazio vuoto deve essere interconnesso;
- le dimensioni dello spazio vuoto devono essere grandi rispetto alla lunghezza del cammino libero medio delle molecole del fluido;
- le dimensioni delle spazio vuoto devono essere sufficientemente piccole in modo tale che il flusso del fluido sia controllato dalle forze adesive all'interfaccia fluido-struttura e da quelle coesive all'interfaccia fluido-fluido nel caso di sistemi multifase. Le velocità del fluido sono sufficientemente piccole da trascurare gli effetti inerziali.

Si consideri un mezzo continuo che riempa il dominio $\Omega \subset \mathbb{R}^3$. Poichè siamo interessati a risolvere il problema alla macroscala i granuli solidi ed i pori non vengono risolti bensì omogeneizzati su di un volume di riferimento, sotto le ipotesi sopracitate il problema può essere descritto dalle equazioni di Darcy. La conservazione della massa è espressa dalla seguente equazione differenziale definita quasi ovunque in Ω:

$$\frac{\partial(\Phi\rho)}{\partial t} + \nabla \cdot \{\rho \mathbf{q}\} = \rho f \quad \text{in } \Omega, \tag{4.1}$$

dove Φ è la porosità del mezzo, ρ la densità in $[kg/m^3]$, $\mathbf{q}$ la velocità macroscopica in $[m/s]$ e f un eventuale termine sorgente in $[s^{-1}]$.

Usando tecniche di media locale o di omogeneizzazione si può mostrare che sotto determinate ipotesi (si veda [24] e [5]) l'equazione di conservazione del momento, nel caso in cui all'interfaccia solido-fluido su scala microscopica si assumano condizioni al contorno di non scorrimento, si riduce alla legge di Darcy a scala macroscopica data da

$$\mathbf{q} = -\frac{\mathbf{K}}{\mu}(\nabla p - \mathbf{f}_v), \tag{4.2}$$

dove p è la pressione del fluido in $[Pa]$, $\mathbf{f}_v$ il vettore delle forze di volume, $\mathbf{K}$ il tensore simmetrico di permeabilità (che in questo lavoro viene supposto isotropo e quindi della forma $\mathbf{K} = k\mathbf{I}$) e μ la viscosità dinamica in $[Pa\,s]$. Questa relazione fu scoperta sperimentalmente per il caso monodimensionale da H. Darcy nel 1856 ed è valida per flussi lenti (in cui gli effetti inerziali possono essere trascurati) di fluidi Newtoniani attraverso un mezzo poroso.

Sostituendo l'equazione (4.2) in (4.1) ed assumendo che il fluido sia incomprimibile si ottiene l'equazione ellittica [24]

$$\begin{cases} \dfrac{\partial \Phi}{\partial t} - \nabla \cdot \left\{ \dfrac{k\mathbf{I}}{\mu}(\nabla p - \mathbf{f}_v) \right\} = f, \\ p = p_D \ \text{ su } \Gamma_D, \\ \mathbf{q} \cdot \mathbf{n} = \eta \ \text{ su } \Gamma_N, \end{cases} \tag{4.3}$$

dove $\Gamma = \partial\Omega = \Gamma_D \cup \Gamma_N$, p_D il dato al bordo di Dirichlet e η un'opportuna funzione per le condizioni ai bordi di Neumann.

4.2 Accoppiamento col modello di precipitazione-dissoluzione

In questa sezione si propone come accoppiare il modello di precipitazione-dissoluzione presentato nel Capitolo 3 con quello di Darcy. Se da un lato la velocità di avvezione dei soluti è determinata appunto dalla legge di Darcy, dall'altro i processi di precipitazione e dissoluzione possono portare a cambiamenti nelle proprietà del mezzo poroso, in particolare nella porosità. L'idea è quindi quella di introdurre una relazione che leghi la concentrazione di cristallo precipitato con la porosità e di conseguenza la permeabilità del mezzo poroso.

Per completezza si riporta il modello di precipitazione-dissoluzione (3.5), (3.6)

$$\begin{cases} \dfrac{\partial}{\partial t}(u+v) + \nabla \cdot (\mathbf{q}u - \mu \nabla u) = 0 & \text{in } \Omega^T, \\ u = g & \text{su } \Gamma_D^T, \\ \nabla u \cdot \mathbf{n} = h & \text{su } \Gamma_N^T, \\ u = u_0 & \text{in } \Omega \text{ per } t = 0, \end{cases} \qquad \begin{cases} \dfrac{\partial v}{\partial t} = r_p(u) - w & \text{in } \Omega^T, \\ v = v_0 & \text{in } \Omega \text{ per } t = 0. \end{cases}$$

Un aumento della concentrazione di precipitato v determina un aumento del volume del materiale e di conseguenza si ha una riduzione nella porosità. Secondo [19] la velocità di variazione di porosità in presenza di processi di precipitazione si può legare alla concentrazione di precipitato con la legge

$$\frac{d\Phi}{dt} = -\sum_i v_i \frac{dc_i}{dt} \tag{4.4}$$

dove c_i sono le concentrazioni molari delle specie precipitate e v_i i rispettivi volumi molari. Nel semplice modello che stiamo utilizzando si considera una sola specie di precipitato. Inoltre, poichè consideriamo equazioni adimensionali, la relazione 4.4 si riduce a

$$\frac{\partial \Phi}{\partial t} = -\frac{dv}{dt}. \tag{4.5}$$

Ad ogni istante di tempo si ha quindi una relazione del tipo

$$\Phi = \Phi_0 - v,$$

dove Φ_0 è posto pari ad 1. Si noti che la porosità descrive una grandezza scalare che varia tra 0 e 1, e che tale proprietà è sempre rispettata nel modello in esame dato che la concentrazione v è sicuramente a sua volta compresa fra 0 ed 1.

Esistono diverse leggi che permettono di esprimere la permeabilità in funzione della porosità della roccia. Si tratta in gran parte di leggi empiriche i cui coefficienti sono calibrati, per vari tipi di roccia, in base a dati sperimentali. In questo lavoro consideriamo una legge piuttosto generale riportata in [21]:

$$k(\Phi) = c\,\Phi^m,$$

dove c è una costante e m è un esponente con valori che possono variare tra 2 e 7. In questo lavoro si è scelto di prendere $m = 2$, inoltre in mancanza di dati realistici si è posto $c = 1$.

Inoltre, la variazione in tempo di porosità costituisce un termine forzante per l'equazione di Darcy. Si sono trascurate le forze di volume ($\mathbf{f}_v = 0$)

$$\widetilde{f} = f - \frac{\partial \Phi}{\partial t}.$$

Infine, il termine sorgente f nell'equazione di conservazione della massa (4.1) si suppone nullo.

4.3 Approssimazione numerica

Analogamente a quanto fatto nel capitolo precedente, il problema è stato splittato con la tecnica del primo ordine (2.3) e (2.4) proposta nel Capitolo 2. Per la discretizzazione in spazio

si è usata una mesh triangolare strutturata 100×100 definita sul dominio $\Omega = (0,1)^2$. Per la discretizzazione temporale si è usato uno schema di Eulero implicito con passo di discretizzazione $\Delta t = 0.01$ sia per lo step di diffusione e trasporto sia per lo step reattivo per l'integrazione del problema al di fuori delle superfici di discontinuità. Nel caso di *sliding motion* sulla discontinuità data da $v = 0$ nella funzione di Heaviside $H(v)$ si è usato uno schema di Runge-Kutta al quarto ordine. Avanzare in tempo in modo esplicito sulla soglia è l'unica alternativa prevista nella libreria di ODE-DRH e quindi si cerca di usare il metodo più accurato possibile.

Per quanto riguarda l'approssimazione numerica del problema di Darcy si è scelto di introdurre una nuova incognita oltre alla pressione: il flusso totale $\mathbf{q}$ dato da (4.2). Usando il flusso totale il problema (4.3) può essere riscritto in forma mista, che è stata introdotta ed analizzata in [27].

Definizione 3. *La formulazione mista del problema (4.3) è trovare una funzione* $p \in \mathcal{C}^1(\Omega)$ *e* $\mathbf{q} \in \left[\mathcal{C}^1(\Omega)\right]^n$ *tale che*

$$\begin{cases} \dfrac{\mu}{k}\mathbf{I}\mathbf{q} + \nabla p = \mathbf{f}_v & \text{in } \Omega, \\ \nabla \cdot \mathbf{q} = \widetilde{f} & \text{in } \Omega, \\ p = p_D & \text{su } \Gamma_D \\ \mathbf{q} \cdot \mathbf{n} = \eta & \text{su } \Gamma_N. \end{cases} \tag{4.6}$$

Si noti che in questa formulazione il flusso di Darcy è un'incognita del problema ed ha la stessa regolarità, in forma forte, della pressione.

Gli spazi in cui si cercano le soluzioni p e $\mathbf{q}$ in forma debole sono

$$\begin{aligned} Q &:= L^2(\Omega), \\ \mathbf{Z} &:= \left\{\tau \in \left[L^2(\Omega)\right]^n : \tau \in \mathbf{H}_{\mathrm{div}}(\Omega)\right\}. \end{aligned}$$

A questo punto si può introdurre la forma mista debole.

Definizione 4. *La formulazione mista debole del problema (4.3) consiste nel trovare* $(\mathbf{q}, p) \in \mathbf{Z} \times Q$ *tale che*

$$\begin{cases} \displaystyle\int_\Omega \frac{\mu}{k}\mathbf{I}\mathbf{q}\tau \, d\Omega - \int_\Omega p\,(\nabla \cdot \tau)\; d\Omega + \int_{\Gamma_N} p\,(\tau \cdot \mathbf{n})\; d\Omega = \int_\Omega \mathbf{f}_v \tau \, d\Omega - \int_{\Gamma_D} p_D\,(\tau \cdot \mathbf{n})\; d\sigma & \forall \tau \in \mathbf{Z}, \\ \displaystyle\int_\Omega (\nabla \cdot \mathbf{q})\, v \, d\Omega = \int_\Omega \widetilde{f} v \, d\Omega & \forall v \in Q. \end{cases}$$

Per la discretizzazione numerica della formulazione mista di Darcy si usano i seguenti sottospazi finito-dimensionali di Q e $\mathbf{Z}$:

- elementi finiti lagrangiani di grado r per le variabili scalari

$$Q_h := \{q_h \in Q : q_h|_K \in \mathbb{P}_r(K)\ \forall K \in \mathcal{T}_h\} \subset Q;$$

  in particolare sono stati usati polinomi $\mathbb{P}_0$ per la pressione.

- elementi di Raviart-Thomas, introdotti in [27], di grado r per le variabili vettoriali

$$\mathbf{Z}_h := \{\mathbf{z}_h \in \mathbf{Z} : \mathbf{z}_h|_K \in \mathbb{RT}_r(K)\ \forall K \in \mathcal{T}_h\} \subset \mathbf{Z},$$

dove $\mathbb{RT}_r(K)$ che è lo spazio ad elementi finiti di Raviart-Thomas sul generico triangolo K della mesh, definito come

$$\mathbb{RT}_r(K) = (\mathbb{P}_r(K))^2 \oplus \mathbb{P}_r(K) \begin{bmatrix} x \\ y \end{bmatrix}.$$

I gradi di libertà sono illustrati in Fig. 4.1 per il grado $r = 0$ con cui sono state eseguite le simulazioni numeriche, in questo caso si ha che $\dim \mathbb{RT}_0(K) = 3$ e le funzioni di base vettoriali mostrate in Fig. 4.2 sono definite come

$$\tau_i^K = \frac{\mathbf{x} - \mathbf{x}_i}{2|K|} \quad \forall \mathbf{x} \in K \qquad i = 1, 2, 3,$$

dove $\mathbf{x}_i$ è la coordinata vettoriale del vertice i-esimo. Le frecce indicano i valori di $\mathbf{u} \cdot \mathbf{n}$ lungo i lati tra un triangolo e quello adiacente, queste quantità sono costanti su ogni lato.

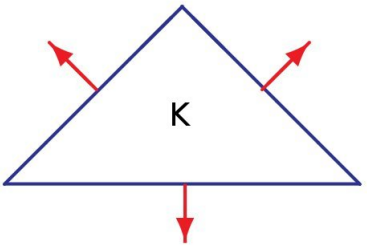


Figura 4.1: Gradi di libertà di $\mathbb{RT}_0(K)$.

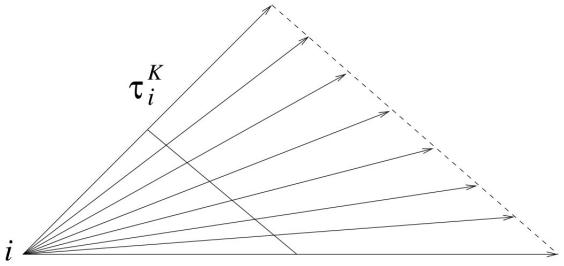


Figura 4.2: Funzioni di base di $\mathbb{RT}_0(K)$.

I metodi misti hanno il vantaggio di approssimare la velocità come un'incognita del problema, mentre nella formulazione primale classica la velocità dovrebbe essere calcolata per derivazione numerica della pressione. Dato che nella formulazione mista l'equazione di continuità non è integrata per parti possiamo aspettarci che sia soddisfatta con un'accuratezza maggiore rispetto ai metodi classici. Infatti, con gli elementi finiti scelti, la conservazione della massa è soddisfatta elemento per elemento. Per questi motivi i metodi misti sono particolarmente indicati per applicazioni come quella di nostro interesse ed in particolare in presenza di forti variazioni del tensore di permeabilità. Inoltre, il calcolo diretto della velocità del fluido è molto conveniente per la successiva soluzione del problema di diffusione e trasporto, [1].

Per imporre la condizione sulla variazione di porosità (4.5) è stata usata un'approssimazione della derivata con differenze finite:

$$-\frac{dv}{dt} = -\frac{v^{n+1} - v^n}{\Delta t}.$$

Quindi, ricordando che $f = 0$, si ha

$$\widetilde{f} = \frac{v^{n+1} - v^n}{\Delta t}.$$

Per il modello di precipitazione-dissoluzione invece sono stati usati elementi finiti $\mathbb{P}_1$.

4.4 Risultati

Il problema è stato risolto nell intervallo $t \in (0, T]$ con $T = 1$ in un dominio $\Omega = (0, 1)^2 \subset \mathbb{R}^2$. Si definisce

$$\Omega^T = (0, T] \times \Omega, \quad \text{e} \quad \Gamma^T = (0, T] \times \Gamma.$$

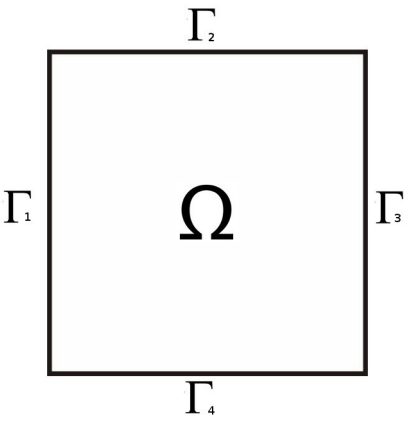


Figura 4.3: Dominio Ω.

Per quanto riguarda le condizioni al contorno, per il problema di Darcy sono stati imposti i valori pressione sui due lati Γ_1 e Γ_3 e condizioni di flusso nullo sui bordi superiore ed inferiore. All'equazione di diffusione-trasporto-reazione che descrive il comportamento dei cationi sono state imposte le stesse condizioni usate nel Capitolo 3, ossia condizioni di Neumann omogenee su Γ_2, Γ_3, Γ_4 e Dirichlet omogeneo su Γ_1. Quindi:

$$
\begin{aligned}
p = 0.5 \quad & u = 0 \quad && \text{su } \Gamma_1, \\
\mathbf{q} \cdot \mathbf{n} = 0 \quad & \nabla u \cdot \mathbf{n} = 0 \quad && \text{su } \Gamma_2, \\
p = 0 \quad & \nabla u \cdot \mathbf{n} = 0 \quad && \text{su } \Gamma_3, \\
\mathbf{q} \cdot \mathbf{n} = 0 \quad & \nabla u \cdot \mathbf{n} = 0 \quad && \text{su } \Gamma_4.
\end{aligned}
$$

La condizione iniziale a $t = 0$ per la concentrazione di cationi è $u|_{t=0} = 1$ ovunque in Ω mentre quella per la concentrazione di precipitato v è diversa nei tre casi considerati. Nel primo caso si è considerato un blocco quadrato Ω_v al centro del dominio:

$$\Omega_v \subset \Omega, \quad \Omega_v := \{(x, y) : \ 0.4 \leq x \leq 0.6, \ 0.4 \leq y \leq 0.6\},$$

tale per cui la concentrazione iniziale di precipitato v, riportata in Fig. 4.4, è:

$$v|_{t=0} = \begin{cases} 0.8 & \text{per } (x, y) \in \Omega_v, \\ 0 & \text{per } (x, y) \in \Omega \setminus \Omega_v. \end{cases} \tag{4.7}$$

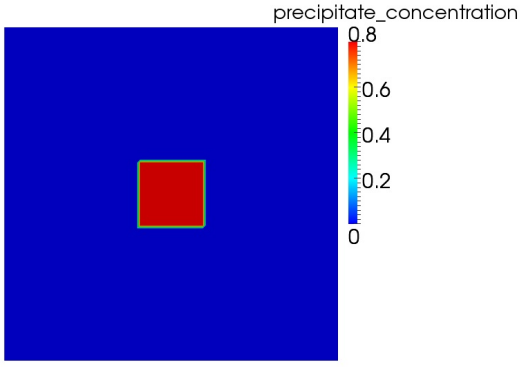
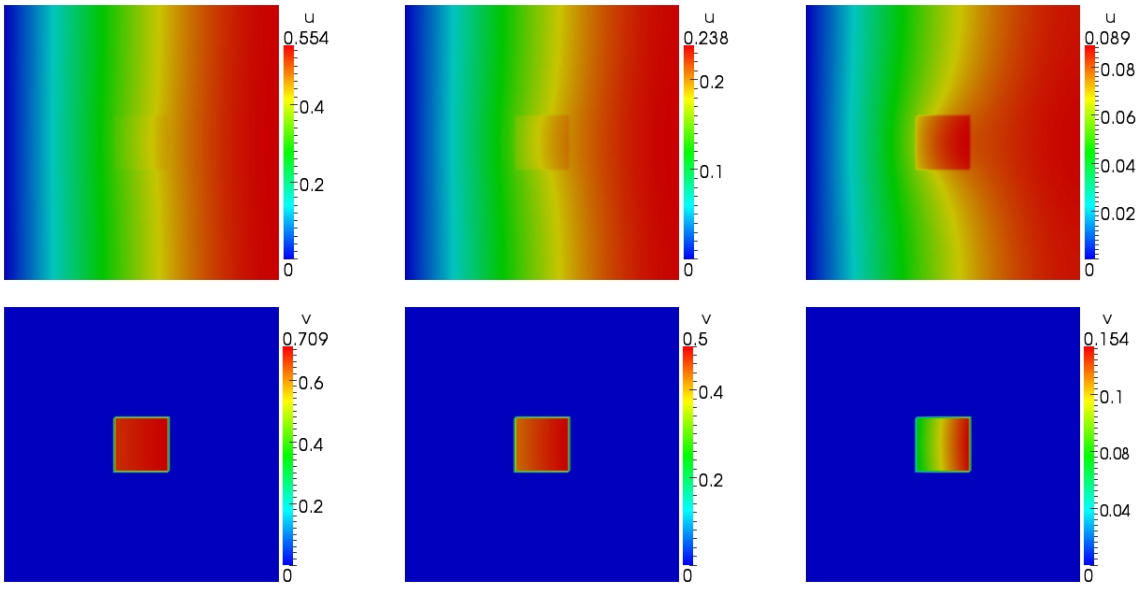


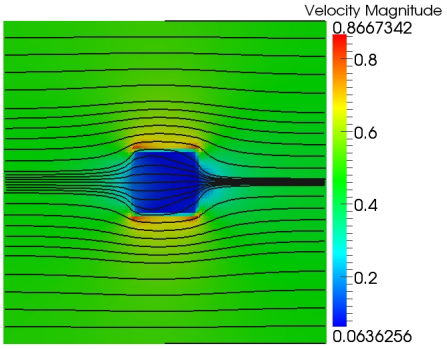
Figura 4.4: Condizione iniziale per v a $t = 0$.

L'andamento delle concentrazioni di cationi e cristalli precipitati è mostrato in Fig. 4.5, mentre la velocità di Darcy corrispondente $\mathbf{q}$ è riportata in Fig. 4.6. La riduzione della concentrazione di cationi determina la dissoluzione del cristallo in Ω_v perché il termine r_d, costante per $v > 0$, supera il termine di precipitazione r_p. Di conseguenza questa variazione della v determina un aumento della porosità del mezzo nelle regioni in cui il cristallo viene maggiormente disciolto. Questo comporta un aumento della velocità $\mathbf{q}$ come si può vedere osservando soprattutto l'evoluzione della componente orizzontale del campo di moto. Si noti anche come la configurazione del cristallo determini una variazione nella componente y del campo di velocità in prossimità degli spigoli di Ω_v.

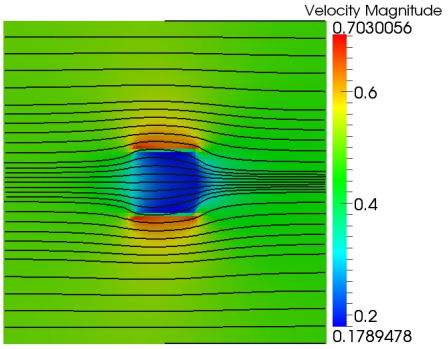


(a) $t = 0.3$. (b) $t = 0.6$. (c) $t = 1$.

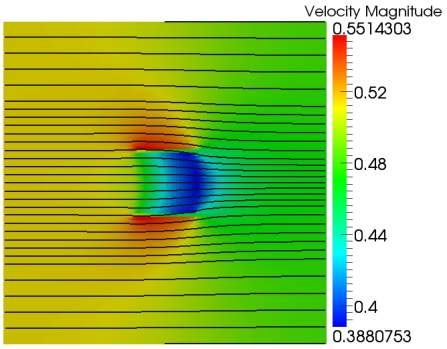
Figura 4.5: Concentrazione di precipitato v e di cationi u con condizione inziale (4.7).



(a) $t = 0.3$.



(b) $t = 0.6$.



(c) $t = 1$.

Figura 4.6: Streamlines del campo di velocità di Darcy $\mathbf{q}$ con condizione inziale (4.7).

Nel secondo caso si è considerata una configurazione a tre blocchi quadrati allinterno dei quali si impone $v = 0.8$ e 0 fuori come mostrato in Fig. 4.7.

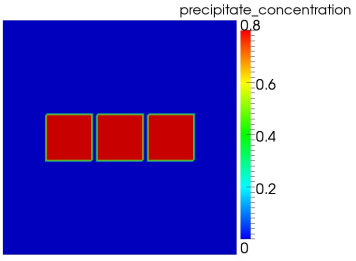
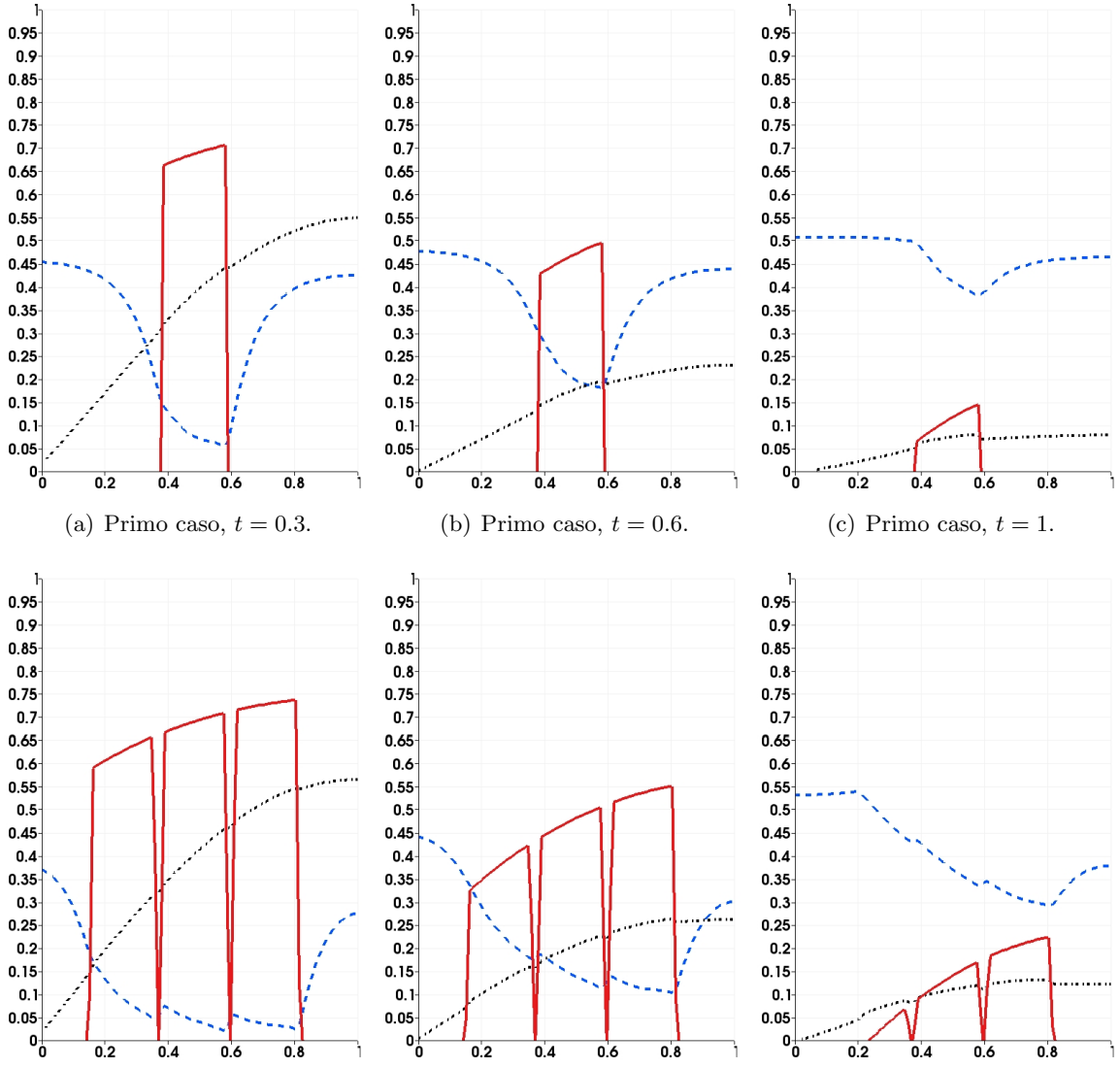


Figura 4.7: Condizione iniziale per v a $t = 0$.

La dinamica del sistema è assolutamente analoga al caso presentato in precedenza, ma come era logico aspettarsi la più estesa concentrazione iniziale di cristalli rallenta maggiormente il flusso nel mezzo poroso. In Fig. 4.8 sono messi a confronto i valori di u, v e $|\mathbf{q}|$ lungo la sezione $y = 0.5$ a diversi istanti di tempo ottenuti nel caso in cui all'instante iniziale è presente un unico blocco quadrato di precipitato e quello in cui ce ne sono tre. A parità di tempo la concentrazione di cationi disciolti è maggiore nel secondo caso perché, a causa della minore intensità del campo di moto il cristallo si dissolve meno.



(a) Primo caso, $t = 0.3$. (b) Primo caso, $t = 0.6$. (c) Primo caso, $t = 1$.

(d) Secondo caso, $t = 0.3$. (e) Secondo caso, $t = 0.6$. (f) Secondo caso, $t = 1$.

Figura 4.8: u (nero $-\cdot$), v (rosso), $|\mathbf{q}|$ (blu $-$) lungo la sezione $y = 0.5$.

Per concludere si è testata una configurazione iniziale in cui si ha sempre $v = 0.8$ in un cerchio di raggio 0.1 posto al centro di Ω (Fig. 4.9).

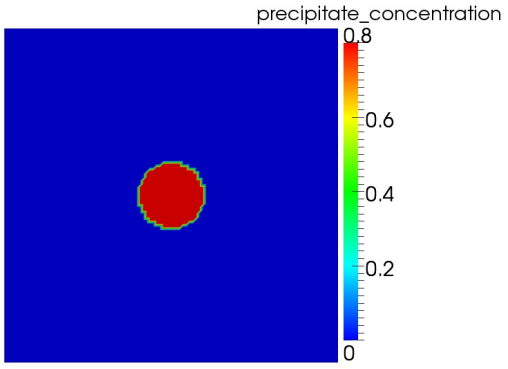
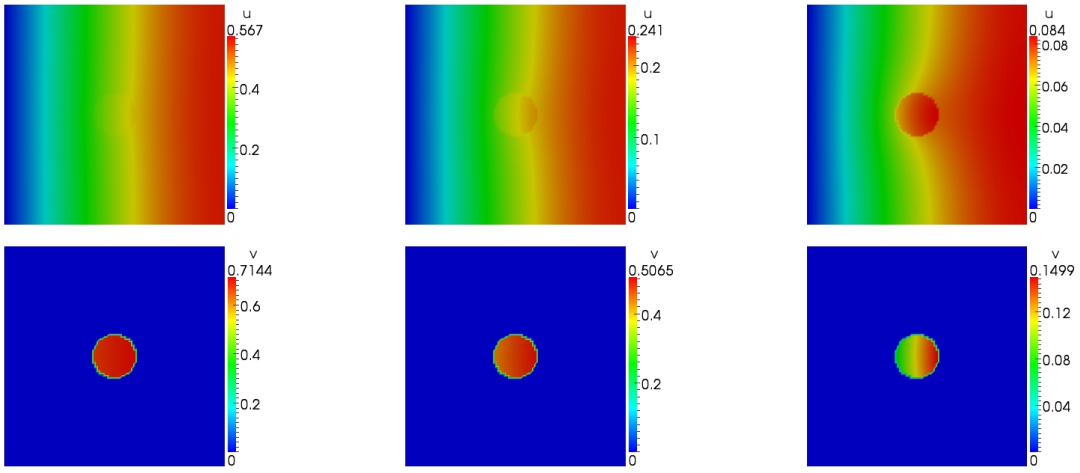


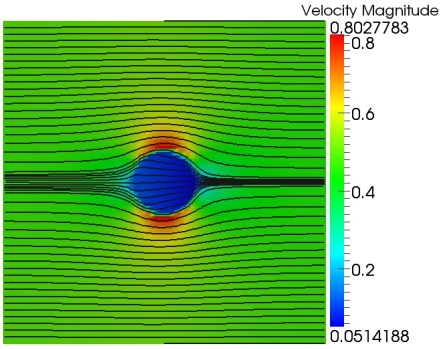
Figura 4.9: Condizione iniziale per v a $t = 0$.



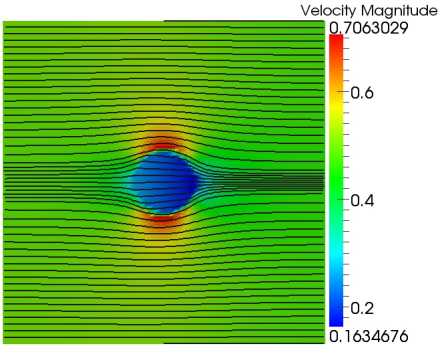
(a) $t = 0.3$. (b) $t = 0.6$. (c) $t = 1$.

Figura 4.10: Concentrazione di precipitato v e di cationi u con configurazione iniziale circolare.

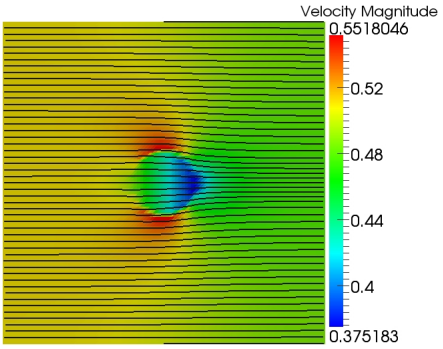
La forma arrotondata della configurazione iniziale di cristallo precipitato fa sì che il fluido incontri meno resistenza rispetto ai casi precedenti. Il campo di moto $\mathbf{q}$ ha intensità maggiore quindi i cationi decrescono poichè vengono trasportati più rapidamente nel mezzo poroso e fanno dissolvere il cristallo. La loro concentrazione finale infatti è leggermente minore rispetto al primo caso.



(a) $t = 0.3$.



(b) $t = 0.6$.



(c) $t = 1$.

Figura 4.11: Streamlines del campo di velocità di Darcy $\mathbf{q}$ con configurazione iniziale circolare.

4.5 Scalabilità del codice

Il problema trattato in questo capitolo è stato sottoposto anche a test di performance su una mesh tridimensionale $32 \times 32 \times 32$ per valutare l'efficienza del codice implementato. In particolare è stata studiata la scalabilità del codice. I calcoli sono stati eseguiti sul calcolatore LAGRANGE del Cilea [13]: 336 nodi, 3200 processori Intel Xeon QuadCore da 3GHz o HexaCore da 2.8GHz, 6400 GB di RAM e una capacità totale pari a circa 50 TB.

Nella Tab. 4.1 sono riportati i tempi di esecuzione che sono stati calcolati facendo una media su tutti gli istanti temporali dei tempi necessari al risolutore di Darcy, all'`ADRAssembler` e alla libreria di ODE-DRH. Come si può vedere i tempi di risoluzione della libreria per le ODE a forzante discontinua scalano quasi perfettamente in maniera lineare. Poiché il problema è scalare esso viene risolto indipendentemente in ogni grado di libertà, ogni processore risolve la ODE-DRH sui nodi assegnatigli senza alcun bisogno di comunicare con gli altri processori.

Per l'ADR e per Darcy il parallelismo è invece fondamentale nel calcolo e assemblaggio delle matrici, nella risoluzione dei sistemi lineari associati alla discretizzazione numerica e nell'esportazione dei risultati. Anche in questo caso comunque sono stati ottenuti dei buoni risultati di scalabilità. Oltre 32 processori i tempi di esecuzione scalano in maniera sublineare per l'ADR e soprattutto per quanto riguarda Darcy. Questo fatto potrebbe essere spiegato dalla dimensione non particolarmente elevata della mesh; i partizionamenti che si formano contengono pochi gradi di libertà e quindi i tempi di comunicazione tra i processori diventano predominanti su quelli di risoluzione.

n	Darcy	ADR	ODE-DRH
2	390.47	27.347	0.49
4	101.89	10.983	0.239
8	39.98	4.67	0.13
16	18.34	2.37	0.067
32	13.81	1.42	0.031
64	20.94	1.068	0.02
128	18.32	0.587	0.01

Tabella 4.1: Tempi di esecuzione del calcolo su n processori.

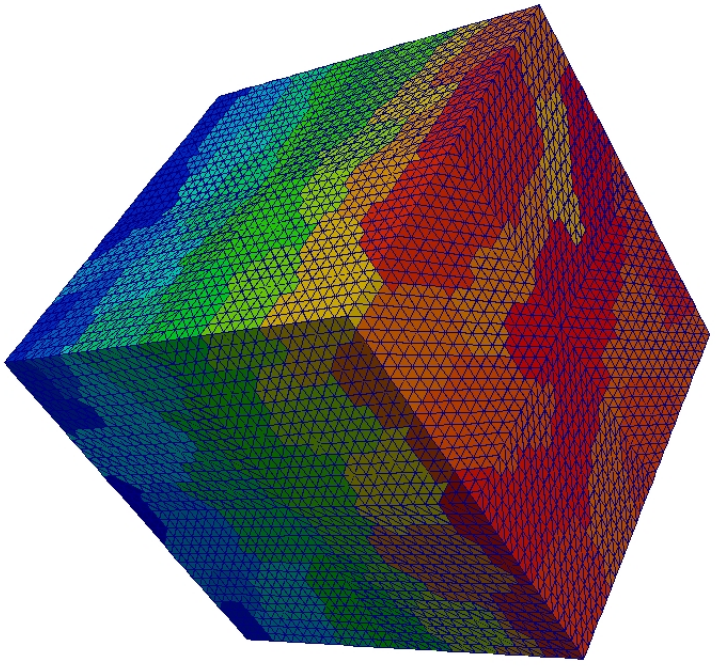


Figura 4.12: Partizionamento della mesh su 128 processori.

4.6 Dettagli sull'implementazione

La risoluzione numerica del problema presentato è basata sulla libreria `LifeV` [10], [11], [22]. In aggiunta agli strumenti utilizzati nel capitolo precedente si è sfruttato anche il modulo `Darcy` (già presente in `LifeV`) per la risoluzione del problema di Darcy accoppiato a quello di precipitazione-dissoluzione. La strategia è praticamente la stessa: ad ogni istante di tempo si risolve Darcy e successivamente l'equazione di diffusione-trasporto-reazione per la u e la ODE a forzante discontinua per la v con uno splitting di ordine 1. Infine si aggiornano la permeabilità e il termine noto di Darcy con i valori di v appena calcolati e si passa all'istante successivo.

Le strutture per il passaggio dei dati e la creazione della mesh sono le stesse presentate nel capitolo precedente. Si definiscono gli spazi ad elementi finiti per il problema di Darcy:

```
// Primal solution parameters
const ReferenceFE*    Darcy_refFE_primal;
const QuadratureRule* Darcy_qR_primal;
const QuadratureRule* Darcy_bdQr_primal;

Darcy_refFE_primal = &feTriaP0;
Darcy_qR_primal    = &quadRuleTria4pt;
Darcy_bdQr_primal  = &quadRuleSeg1pt;

// Dual solution parameters
const ReferenceFE*    Darcy_refFE_dual;
const QuadratureRule* Darcy_qR_dual;
const QuadratureRule* Darcy_bdQr_dual;

Darcy_refFE_dual = &feTriaRT0;
Darcy_qR_dual    = &quadRuleTria4pt;
Darcy_bdQr_dual  = &quadRuleSeg1pt;

// Finite element space of the primal variable
fESpacePtr_Type Darcy_p_FESpacePtr ( new fESpace_Type
     ( meshPart, *Darcy_refFE_primal, *Darcy_qR_primal,
               *Darcy_bdQr_primal, 1, Members->comm ) );

// Finite element space of the dual variable
fESpacePtr_Type Darcy_u_FESpacePtr ( new fESpace_Type
     ( meshPart, *Darcy_refFE_dual, *Darcy_qR_dual,
                 *Darcy_bdQr_dual, 1, Members->comm ) );
```

Si impongono le condizioni al bordo:

```
bcHandlerPtr_Type bcDarcy ( new bcHandler_Type );

BCFunctionBase dirichletBDfun;
BCFunctionBase neumannBDfun;
```

```
dirichletBDfun.setFunction ( dataProblem::dirichlet );
neumannBDfun.setFunction ( dataProblem::neumann1 );

bcDarcy->addBC(    "Top",    TOP,      Natural,    Full,  neumannBDfun,  1 );
bcDarcy->addBC( "Bottom", BOTTOM,      Natural,    Full,  neumannBDfun,  1 );
bcDarcy->addBC(   "Left",   LEFT,    Essential,  Scalar,  dirichletBDfun   );
bcDarcy->addBC(  "Right",  RIGHT,    Essential,  Scalar,  dirichletBDfun   );
```

Si istanzia l'oggetto della classe `DarcySolver` e se ne settano dati e forzanti:

```
// Instantiation of the DarcySolver class
DarcySolverLinear < RegionMesh < geoElement_Type >,
                    SolverAztecOO > darcySolver;

// Set the source term
scalarFctPtr_Type scalarSourceFct ( new scalarSource );
darcySolver.setScalarSource ( scalarSourceFct );

// Set the vector source term
vectorFctPtr_Type vectorSourceFct ( new vectorSource );
darcySolver.setVectorSource ( vectorSourceFct );

// Set the reaction term
scalarFctPtr_Type reactionTermFct ( new reactionTerm );
darcySolver.setReactionTerm ( reactionTermFct );

// Set the inverse of the permeability
matrixFctPtr_Type inversePermeabilityFct ( new inversePermeability );
darcySolver.setInversePermeability ( inversePermeabilityFct );
```

Infine si accoppiano i due problemi aggiornando permeabilità e termine sorgente in base alla concentrazione v calcolata:

```
// Couple the two problems via the permeability matrix
inversePermeabilityFct->addScalarField( vFieldOld );

// Couple the two problems via the scalar source term
scalarSourceFct->addScalarField( vFieldOld );
scalarSourceFct->addScalarField( vField );
```

Per risolvere Darcy rimane solo da chiamare il comando:

```
// Couple the two problems via the permeability matrix
darcySolver.solve();
```

L'implementazione del risolutore per il problema di diffusione-trasporto-reazione e del ciclo iterativo in tempo è la medesima presentata nel Capitolo 3. Ad ogni iterazione va solo aggiunta la chiamata al solutore di Darcy che restituisce il campo di velocità da passare alla funzione `addAdvection` della classe `ADRAssembler`.

Conclusioni

Questo lavoro di tesi aveva come obiettivo l'implementazione di algoritmi risolutivi per sistemi di equazioni differenziali ordinarie con termini forzanti discontinui e l'accoppiamento di queste equazioni con modelli di diffusione-trasporto-reazione e di Darcy.

In una prima fase sono stati analizzati i risultati teorici dovuti a Filippov e Dieci Lopez relativi alle equazioni con discontinuità dipendenti dalla soluzione. La teoria ed il metodo di approssimazione presentati in [8] sono stati estesi al caso più generale di equazioni non autonome e con superfici di discontinuità che dipendono esplicitamente dal tempo. Si è quindi passati all'implementazione e alla validazione di tale metodo. All'interno della libreria di calcolo ad elementi finiti `LifeV` è stato sviluppato un solutore che implementa i metodi di integrazione per ODE classici e le tecniche necessarie a determinare il comportamento della soluzione di ODE-DRH all'intersezione con le superfici di discontinuità. La difficoltà maggiore incontrata in questa parte del lavoro è stata l'organizzazione delle classi dei solutori per ODE e ODE-DRH, le strutture necessarie a gestire ed elaborare i dati del problema e il modo in cui interfacciare le due classi nella complicata procedura di analisi dei vari casi che si possono verificare in problemi di questo tipo. Sugli schemi di integrazione numerica implementati sono stati effettuati test numerici di convergenza dimostrando che la corretta approssimazione dell'equazione in corrispondenza della superficie di discontinuità permette di conservare l'ordine teorico di convergenza degli schemi. Sono stati anche riprodotti risultati proposti in letteratura nell'ambito di reazioni chimiche descritte da equazioni differenziali ordinarie con forzanti discontinue.

Successivamente sono state studiate ed implementate tecniche di splitting applicate ad equazioni di diffusione trasporto e reazione. L'obiettivo era quello di poter risolvere problemi in cui il termine di reazione presenta discontinuità separandone il contributo dall'operatori di diffusione e trasporto. Sono stati implementati con successo splitting del primo e del secondo ordine risolvendo in `LifeV` con la classe `OdeDrhSolver` il sottoproblema di reazione e con elementi finiti quello di diffusione e avvezione. Sono state discusse le criticità legate all'imposizione del dato al bordo per gli operatori splittati proponendo una strategia per ottenere l'ordine di convergenza teorico dello schema di splitting usato.

Le tecniche sopracitate sono state utilizzate per risolvere un caso test di interesse applicativo: il problema di dissoluzione e precipitazione responsabile della formazione di cristalli in un mezzo poroso. Tale modello è descritto da un'equazione a derivate parziali e da un'equazione differenziale ordinaria con forzante discontinua. Dal confronto effettuato col risultato di letteratura è stato possibile validare l'efficienza della libreria di ODE-DRH e dello schema di splitting utilizzato. Un ulteriore punto di forza del codice implementato in questa tesi è la possibilità di

risolvere il problema sopracitato senza ricorrere a una regolarizzazione del termine di reazione, che inevitabilmente compromette l'accuratezza della soluzione oltre a introdurre una stiffness non naturale nel problema.

Le equazioni di diffusione trasporto e reazione discontinua trattate nel problema di precipitazione e dissoluzione sono state accoppiate al modello di flusso di Darcy ipotizzando un legame tra la permeabilità del mezzo poroso e la concentrazione di precipitato. I risultati ottenuti mostrano come la variazione di cristallo precipitato e la velocità di Darcy siano dipendenti l'uno dall'altra.

Le possibilità di ampliamento e di sviluppo futuro del presente lavoro sono numerose. Allo stato attuale il codice implementato per le ODE-DRH è in grado di trattare un unico *sliding motion*, ossia non è possibile che la soluzione del problema si trovi su due superfici di discontinuità contemporaneamente. Occorrerebbe estendere la trattazione ad un caso più generale per poter definire il termine forzante all'intersezione delle soglie e le corrette condizioni per determinare il comportamento della soluzione. Un altro campo in cui c'è ampio margine di studio e di miglioramento è quello dell'operator splitting: si potrebbe lavorare su tecniche ad alto ordine sia nel campo degli splitting sequenziali sia in quello degli splitting iterativi. Infine un'interessante sviluppo futuro consiste nell'applicazione a casi reali, con diversi valori di permeabilità, velocità di reazione e concentrazioni delle specie chimiche coinvolte nel processo di precipitazione e dissoluzione, con lo scopo di valutare l'interazione fra il flusso ed i processi reattivi all'interno di strati sedimentari. Altre possibili applicazioni riguardano modelli di generazione ed espulsione di idrocarburi, stoccaggio di CO_2, analisi di flussi reattivi in ambito ambientale.

Bibliografia

[1] Fumagalli A. "Numerical Modelling of Flows in Fractured Porous Media by the XFEM Method". Tesi di dott. Politecnico di Milano, 2012.

[2] Scotti A. "A numerical model for generation, retention and expulsion of hydrocarbons from source rocks". Tesi di dott. Politecnico di Milano, 2010.

[3] Filippov A.F. "Differential equations with discontinuous right hand sides". In: *Mathematics and its application* (1988).

[4] Cellina A. Aubin J.P. *Differential inclusions*. Springer, 1984.

[5] Jacob Bear. *Dynamics of Fluids in Porous Media*. American Elsevier, 1972.

[6] *Boost C++ libraries*. URL: `http://www.boost.org/`.

[7] Yang D. *C++ and Object Oriented Numeric Computing for Scientists and Engineers*. Springer-Verlag, 2001.

[8] Lopez L. Dieci L. "Sliding motion in discontinuous differential systems: Theory and a computational apporach". In: *Georgia Institute of Technology* (2008).

[9] Scotti A. Formaggia L. "Positivity and conservation properties of some integration schemes for mass kinetics". In: *SIAM J. Numer. Anal.* 49 (2011), pp. 3/1267–1288.

[10] Deparis S. Fourestey G. *LifeV User Manual*. 2010. URL: `www.lifev.org`.

[11] Formaggia L. Gerbeau J.F. Prud'homme C. Fourestey G. Deparis S. *LifeV Developer Manual*. 2010. URL: `www.lifev.org`.

[12] Østerby O. Gear C.W. "Solving ordinary differential equations with discontinuities". In: *ACM transaction on mathematical software* 10.1 (1984), pp. 23–24.

[13] *Gruppo HPC CILEA*. URL: `http://hpc.cilea.it/servizi/calcolo/`.

[14] Willenbring J.M. Heroux M.A. *Trilinos Users Guide*. Rapp. tecn. SAND2003-2952. Sandia National Laboratories, 2003.

[15] Verwer J.G. Hundsdorfer W. "A note on splitting errors for advection-reaction equations". In: *Applied Numerical Mathematics* 18 (1995), pp. 191–199.

[16] Geiser J. "Iterative operator-splitting methods with high-order time integration methods and applications for parabolic partial differential equations". In: *Journal of Computational and Applied Mathematics* 217 (2008), pp. 227–242.

[17] Radu F.A. K. Kumar Pop I.S. *Convergence analysis for a conformal discretization of a model for precipitation and dissolution in porous media*. Rapp. tecn. 12-8. CASA, 2012.

[18] Kumar V. Karypis G. Schloegel K. *ParMetis: Parallel Graph Partitioning and Sparse Matrix Ordering*. 2003. URL: `http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview`.

[19] Guadagnini A. Saaltink M.W. Katza G.E. Berkowitza B. "Experimental and modeling investigation of multicomponent reactive transport in porous media". In: *Journal of Contaminant Hydrology* 120-121 (2011), pp. 27–44.

[20] Hengst S. Knabner P. van Duijn C.J. "An Analysis of Crystal Dissolution Fronts in Flows through Porous Media". In: *Computing* 18/171-185 (1995).

[21] Wangen M. *Physical Principles of Sedimentary Basin Analysis*. Cambridge University Press, 2010.

[22] Deparis S. Malossi C. *LifeV Development Guidelines*. 2011. URL: `www.lifev.org`.

[23] Danca M.F. "On the uniqueness of solutions to a class of discontinuous dynamical systems". In: *Nonlinear Analysis Series B: Real World Applications* 11 (2010), pp. 1402–1412.

[24] Bastian P. "Numerical Computation of Multiphase Flows in Porous Media". Tesi di dott. Kiel University, 1999.

[25] Valli A. Quarteroni A. *Numerical Approximation of Partial Differential Equations*. Springer, 2008.

[26] Garcia-Lopez C.M. Ramos J.I. "Intermediate boundary conditions in operator-splitting techniques and linearization methods". In: *Applied Mathematics and Computation* 94 (1998), pp. 113–136.

[27] Thomas J.M. Raviart P.A. "A mixed finite element method for second order elliptic problems". In: *Lecture Notes in Mathematics* 606:292–315 (1977).

[28] Shadid J. Ropp D. "Stability of operator splitting methods for systems with indefinite operators: advection-reaction-diffusion systems". In: *Journal of Computational Physics* 228 (2009), pp. 3508–3516.

[29] T. Swicegood. *A Pragmatic guide to GIT*. Pragmatic Programmers, 2010.

[30] *Git*. URL: `http://git-scm.com/`.

[31] Rudin W. *Real and complex analysis*. McGraw-Hill, 1975.