INDICE

CAPITOLO 1

La simulazione come strumento di ricerca scientifica

Lo sviluppo della tecnologia è perseguito costantemente dall'umanità al fine di ottenere metodologie, strumenti, macchinari, che adempiano ad uno scopo particolare. Ogni elemento elaborato dalla mente umana attraverso i secoli ed i millenni della sua evoluzione è servito, o serve tuttora, per lo svolgimento di precise azioni; così il treno serve per gli spostamenti, le forbici per tagliare ecc.

Il computer invece e uno strumento che può essere utilizzato per molti scopi differenti: scrivere, calcolare, gestire immagini, comunicare con gli altri, vendere, comprare, e così via quasi fino all'infinito. Il fatto che il computer possa essere utilizzato per moltissimi scopi risiede nella sua stessa natura, infatti, esso non gestisce cose fisiche bensì informazioni e quasi tutto può essere tradotto in informazioni e quindi può essere manipolato dal computer.

In questa sede l'utilizzo del computer cui facciamo riferimento è quello per scopi scientifici ed in particolare come strumento di ricerca nelle mani di uno scienziato. Lo scienziato può gestire, con un computer, in modo variabile, i dati a sua disposizione; può per esempio immagazzinare notevoli quantità di dati, può trasformare quantità di dati in grafici o in animazioni, al fine di rendere possibile una miglior comprensione di essi, grazie alla notevole capacità dell'uomo di cogliere correlazioni tra le immagini visive.

Ancora lo scienziato può servirsi del computer per ricreare la realtà che vuole studiare all'interno di simulazioni. Scopo dello scienziato "è quello di conoscere e capire la realtà"[1] e l'adempimento a questo compito avviene attraverso l'osservazione della realtà stessa, in modo sistematico e scientifico e della successiva raccolta di dati ed elaborazione di teorie, solitamente

[1] Domenico Parisi (2001) , *SIMULAZIONI – La realtà rifatta nel computer*, p. 10, il Mulino

quantitative, che spieghino i fenomeni osservati attraverso le regole, le cause, i meccanismi che li originano. Così, con l'utilizzo della simulazione, si offre un nuovo strumento allo scienziato per l'adempimento del suo compito.

1.1 Le scienze

Per meglio comprendere le potenzialità che le simulazioni offrono allo sperimentatore, è opportuno porsi alcune domande.

Che cosa è la scienza? Cosa la distingue dalle altre attività umane?

Riflettendo su cosa sia la scienza, ci viene in mente la rilevanza che gli scienziati danno ai fatti empirici ed all'osservazione diretta di questi; una considerevole parte del tempo che lo scienziato adopera per i suoi studi è rivolta alla descrizione, alla misurazione, all'osservazione della realtà.

L'analisi che lo studioso esegue, deve essere diretta, cioè percepita con i propri sensi, evitando così di considerare tutto ciò che è interpretazione o racconto. A volte si rende necessario l'ausilio di metodologie o sistemi e di strumentazioni specifiche, senza le quali l'indagine stessa non sarebbe possibile, basti pensare a microscopi, necessari per le analisi molecolari, ai telescopi, utilizzati per le scienze spaziali, e molti altri ancora. Altre volte lo studio si concentra non direttamente sui fenomeni che si vuole analizzare, ma sulle conseguenze, sui resti di questi, come accade in paleontologia, in archeologia ed in generale in tutte le scienze storiche. Tuttavia quale che sia la situazione di quelle citate sopra, il soggetto compie osservazioni, analisi, studi in prima persona.

La motivazione che spinge gli scienziati a dedicare tanto del loro tempo per i fatti empirici risiede nel fatto che l'osservazione della realtà è la prova della veridicità o meno delle loro teorie.

Altresì accade per le scienze dell'uomo, che, contrariamente alle scienze della natura, si limitano a formulare teorie dimostrabili unicamente in sede di dissertazione e di ragionamento, senza poter offrire dati empirici a prova delle teorie. Non si verificherà mai, quindi, che una teoria filosofica possa essere confutata da fatti empirici, poiché tali fatti non esistono; differentemente tale situazione si potrà sicuramente presentare, ad esempio, per una teoria sulla genetica.

Si potrebbe obiettare allora che, chiunque nella vita di tutti i giorni osserva la realtà per farsi proprie idee di questa e per comportarsi conseguentemente ad esse; ciò è fuorviante, infatti, lo scienziato è caratterizzato dalla sistematicità, dalla metodicità, dalla accuratezza, dalla professionalità delle sue osservazioni ed analisi.

Lo scienziato cerca:

- Di evitare di trarre conclusioni generali da osservazioni singole che potrebbero non essere correttamente rappresentative.
- Controlla le condizioni ambientali in cui si svolgono le osservazioni, al fine di escludere l'influenza di fattori estranei.
- Utilizza una descrizione dei fatti di tipo quantitativo cosicché i risultati delle osservazioni risultino il più possibile oggettive e precise.
- Si assicura dell'oggettività delle sue osservazioni, cioè, si accerta del fatto che, se al suo posto ci fosse stato un altro scienziato, questi avrebbe verosimilmente tratto le sue stesse conclusioni.

Tuttavia, per quanto tale aspetto empirico risulti essere di primaria importanza nel definire cosa sia la scienza, non sono da meno le teorie stesse, che dopo ripetute analisi di fenomeni, ne spiegano la dinamica, l'essenza, le cause che li generano, li prevedono, li controllano.

Così il semplice osservare e descrivere accuratamente ed oggettivamente i fatti risulta essere conoscenza della realtà, non comprensione di questa; soltanto scoprendo cosa genera il fenomeno, lo scienziato, è in grado di dare una vera

spiegazione della realtà. Vengono così formulate teorie scientifiche, insiemi di regole, concetti, idee, attraverso le quali lo scienziato comprende la realtà, oltre che conoscerla, individuando le motivazioni per cui si verifica un evento piuttosto che un altro.

Detto questo, si può affermare che, perché una teoria possa chiamarsi tale, è necessario che da tale teoria sia possibile estrapolare delle previsioni, cioè delle anticipazioni sui fatti che si dovrebbero osservare nella realtà, dati determinati presupposti. Se ciò non si verifica, o perché non si possono estrapolare previsioni, o perché le predizioni ottenute non hanno riscontro con la realtà, allora non si può parlare di teoria scientifica.

Ciò differenzia decisamente le scienze e le relative teorie, da altre teorie, come quelle filosofiche, religiose, le quali non danno precise previsioni sulla realtà e nemmeno si adoperano nel valutare empiricamente e nel sottoporre ad analisi le teorie, come invero accade nell'ambito scientifico. L'aspetto empirico è considerabile come elemento di giunzione tra la realtà e le teorie che la spiegano: la realtà è compresa attraverso le teorie su di essa e le teorie sono verificate, dimostrate, provate, dal manifestarsi di alcuni eventi.

L'immagine qui data della scienza, non deve tuttavia trarre in inganno, facendo pensare alle due attività, quella di osservazione della realtà e quella di creazione di una teoria, come due parti separate; esse devono essere viste come due facce di una stessa moneta, la cui esistenza è legata alla presenza di entrambe.

Nemmeno si deve pensare a tali attività come assolutamente oggettive, infatti: "la scienza non è necessariamente oggettiva. Tuttavia la scienza per chiamarsi scienza deve necessariamente cercare di essere oggettiva".[2] La frase appena citata vuole significare che lo scienziato decide di elaborare una teoria, non così per caso, senza motivo, ma perché è interessato all'argomento, perché nutre delle speranze di ottenere determinati risultati ecc, e l'osservazione della realtà è anch'essa influenzata dagli occhi con la quale è vista. Quest'aspetto può

[2] Domenico Parisi (2001), *SIMULAZIONI – La realtà rifatta nel computer*, p. 20, il Mulino

essere un problema serio se non affrontato nella dovuta maniera, poiché rischia di far incorrere in errori lo scienziato il quale, allora, adopererà determinate regole comportamentali per prevenire inconvenienti.

Esso assumerà:

- Che la ricerca scientifica è un'impresa collettiva, nel senso che ogni scienziato ha la possibilità di poter verificare egli stesso la robustezza di qualsiasi teoria proposta da altri colleghi.
- Che elemento basilare di una teoria è la formulazione quantitativa delle teoria e non solo quella qualitativa o descrittiva; questo permette di avere una visione più nitida, oggettiva e scevra da influenze del linguaggio.
- Si utilizzano esperimenti in laboratorio, così lo scienziato può controllare le condizioni nelle quali i fenomeni avvengono ed anzi li provoca, comprendendo bene quali sono gli elementi e le concomitanze di essi che ne causano il verificarsi.

A questo punto è possibile dare risposta alle domande iniziali, infatti possiamo dire che ciò che contraddistingue e caratterizza la scienza è il legame tra le teorie elaborate ed i fatti empirici cui tali teorie si riferiscono. Così non è scienza l'elaborazione di teorie priva di analisi dei fatti empirici ed altrettanto non è sufficiente a far sì che si parli di scienza, l'osservazione dei fatti empirici, senza che essi vengano poi elaborati in teorie che ne possano dare una spiegazione, una comprensione e quindi delle previsioni (dati alcuni fattori).

1.2. Che cos' è la simulazione

L'analisi e lo studio dei fenomeni sociali, come l'economia, stanno facendo largo uso della simulazione supportata dall'uso di computer, affiancandola e talvolta sostituendola alle due metodologie tradizionali, quella della formalizzazione matematica e quella della trattazione verbale

Frequentemente nel campo della ricerca economica l'espressione del modello in oggetto è di tipo matematico-statistico, spesso complicato e ciò comporta delle difficoltà di trattazione che vengono superate attraverso semplificazioni, al fine di ottenere equazioni più semplicemente risolvibili, con la conseguenza di rendere il sistema troppo semplice rispetto a ciò su cui si basa.

La trattazione puramente verbale rende invece complesso per il ricercatore stesso, individuare con precisione le implicazioni dei risultati ottenuti e la verifica dei legami di tipo quantitativo tra le varie grandezze considerate.

Attraverso le teorie scientifiche, che sono insiemi di concetti ed idee, lo scienziato tenta di carpire i meccanismi, le cause, i fattori che generano gli eventi nella realtà; elaborati tali concetti, questi devono essere espressi, formulati, formalizzati, al fine di permettere allo stesso scienziato di continuare lo studio, ma anche per permettere la dissertazione scientifica tra i vari colleghi, spesso fondamentale nel procedimento di evoluzione delle teorie. Fino a pochi anni fa le teorie sono state espresse o attraverso l'uso del linguaggio, con tutte le precisazioni che ne derivano, come neologismi creati *ad hoc,* oppure, ridefinizioni di significati di parole già esistenti, ecc., ma anche attraverso l'uso di simbologie quantitativo-matematiche, oppure attraverso l'utilizzo di sistemi grafici, che servono a cogliere la capacità comprensione della nostra vista.

Quale che sia la metodologia usata per la formalizzazione delle teorie, rimane il fatto che esse fanno comunque uso di simbologie. La simulazione, quindi, che cosa è?

La simulazione riesce a superare in parte limiti delle metodologie tradizionali suddette: "le simulazioni sono un nuovo modo di esprimere le teorie che non usa simboli"[3]. In una simulazione il modello viene rappresentato come un programma al computer che può essere utilizzato sia per la descrizione di teorie qualitative sia per quella di teorie quantitative; il limite della trattazione tradizionale è qui superato grazie al potenza del calcolatore il quale è in grado di rappresentare relazioni anche non lineari tra le variabili trattate. Tutto ciò che concerne concetti, meccanismi, processi postulati dalla teoria, non vengono descritti a parole o rappresentati in formule, ma vengono esplicitati in un programma che poi rende una rappresentazione di quelli che sono i fenomeni che si vuole studiare.

Attraverso la simulazione si cerca di ricreare i fenomeni che si vuole analizzare, piuttosto che descriverli, prevederli o spiegarli come la metodologia tradizionale farebbe; l'iter lavorativo prevede la formulazione di ipotesi e di teorie. Quindi possiamo dire che le teorie espresse attraverso le forme tradizionali del linguaggio, dei simboli matematici e dei grafici, si limitano a spiegare la realtà, mentre le simulazioni la ricreano attraverso il programma che gira nel computer; in tal modo i fenomeni che si vuole studiare sono riprodotti grazie alla macchina di calcolo.

Per contro, si potrebbe facilmente obbiettare che anche i programmi fatti al computer sono costituiti da simboli, quelli utilizzati per i vari linguaggi di programmazione (Objective C, Java, Basic, ecc.), per cui anche le simulazioni, in quanto facenti uso di programmi, sono espresse con simboli; è comunque palese una differenza, che confuta tale obbiezione, è cioè che i linguaggi usati per la formulazione delle teorie, attraverso metodologie tradizionali, sono destinati ad altri individui, mentre i codici, con cui si scrivono i programmi, sono volti a permettere l'interazione tra uomo e macchina; sarà poi il prodotto che risulta dalla simulazione che avrà il compito di permettere la comprensione della teoria ad altri individui.

[3] Domenico Parisi (2001), *SIMULAZIONI – La realtà rifatta nel computer*, p. 32, il Mulino

Analizzando meglio la differenza sopra esposta, si può dire che i linguaggi matematico-statistici, i grafici ed il linguaggio comune, sono simboli nel senso che rappresentano un concetto, uno schema mentale, che si visualizza nella mente del destinatario dell'informazione, quando il simbolo è percepito; invece le simbologie utilizzate per la programmazione, non devono produrre significati nella mente dei fruitori del programma (diverso è per il programmatore, che deve poter conoscere i vari significati dei simboli che utilizza, per la creazione di un programma).

Così l'utilizzo delle simulazioni permette un'evoluzione formale del fare ricerca, infatti, non sarà più necessario che gli utilizzatori delle teorie stesse comprendano tutti i passaggi che portano a determinati risultati, poiché a questi verrà già data la predizione ricavata dalla teoria stessa, che, come è stato detto precedentemente, deve corrispondere alla realtà. Compito del ricercatore, nel creare una teoria diventa così, la formalizzazione della teoria stessa in un codice di programmazione e l'osservazione dei risultati che il programma genera (visualizzati attraverso il monitor), risultati che sono apprezzabili quanto combaciano con la realtà, cioè quando offrono una predizione di questa.

1.3. Un metodo differente per la creazione di teorie scientifiche

L'utilizzo delle simulazioni ovviamente non impedisce l'elaborazione mentale di teorie, di concetti, di formulare ipotesi, tuttavia ne cambia il metodo e l'approccio tradizionale. Qualora le simulazioni venissero dopo la formulazione della teoria nella mente dello scienziato, nulla cambierebbe rispetto all'approccio tradizionale, ma ciò si verifica raramente, in quanto lo scienziato che utilizzi il metodo della simulazione interagisce con essa nel formulare la teoria originata dai propri studi. Si promuove così un continuo passare da mente a simulazione,

nella elaborazione delle teorie e per tale motivo si può affermare che la metodologia di approccio alla ricerca è diverso da quello di uno scienziato che usa i metodi tradizionali.

Ulteriore differenza rispetto alla "tradizione" è che lo scienziato può elaborare le proprie predizioni sulla base della sua teoria e poi, invece di verificarle osservando la realtà, può ricreare l'ambiente in una simulazione all'interno del computer e verificare i risultati ottenuti. Così uno scienziato che si serva di simulazioni può sempre, oltre che formulare le teorie nella propria mente, elaborarne delle predizioni, come è sempre stato fatto, e fare verifiche di queste attraverso l'utilizzo di realtà simulata all'interno di calcolatori. Diversamente accade nella scienza tradizionale, infatti lo scienziato non può fare altro se non controllare osservando direttamente la realtà, per valutare la bontà predizioni ricavate dalle teorie.

Non deve tuttavia essere sopravvalutato l'esperimento mentale, pur avendo a disposizione supporti tecnologicamente avanzati come i computer, poiché attraverso la costruzione mentale degli esperimenti, si riescono ad ottenere notevoli vantaggi, come la comprensione più approfondita dei propri studi, l'individuazione di eventuali limiti o problemi, la coerenza o no delle varie parti che la compongono. L'utilizzo della simulazione non deve quindi essere motivo di abbandono di alcuni passi essenziali della ricerca.

La simulazione dà un taglio più oggettivo agli esperimenti mentali dello scienziato, supportandolo ed aiutandolo attraverso l'oggettività dei risultati generati dai suoi calcoli, ricavati dalla teoria elaborata dalla scienziato stesso; lo studioso genera la teoria e la predizione da essa, ma è poi il computer che ne verifica la correttezza attraverso i risultati della simulazione.

1.4. Realtà e realtà artificiale

E' stato detto che la forma di controllo che lo scienziato ha a disposizione nelle sue valutazioni è il controllo diretto della realtà, o l'analisi di ciò che accade nella simulazione. Quanto appena detto può essere ampliato e completato dal seguente concetto: "Le simulazioni sono teorie ma non sono solo teorie, sono anche realtà. Naturalmente si tratta di realtà artificiale, cioè di realtà costruita da noi. Ma esse sono realtà come sono realtà altri prodotti esterni del comportamento umano, gli artefatti tecnologici, i prodotti della comunicazione, le opere d'arte"[4].

Al fine di poter comprendere la bontà di tale affermazione risulta necessario esprimere una definizione di realtà o almeno fornire alcuni elementi che la caratterizzino e la identifichino presi nel loro insieme. Ecco tre criteri per indicarla.

La realtà è:

1. Quello che può essere percepito attraverso un'azione dei sensi o delle nostre percezioni (dolore, movimenti del corpo).
2. Quello su cui possiamo esercitare delle azioni.
3. Quello che può costituire un vincolo, una limitazione alle nostre azioni ovvero un mezzo per svolgerle.

Una distinzione che possiamo fare tra gli abitanti del nostro mondo è che, per gli animali diversi dall'uomo, salvo alcuni casi, esiste solo un tipo di realtà, che è la realtà naturale, mentre per l'uomo esiste sia la realtà naturale, la quale è tale, indipendentemente dall'esistenza degli esseri umani ed ha caratteristiche che non dipendono da azioni dell'uomo, sia la realtà artificiale che è frutto dell'operato dell'uomo. Quindi anche le simulazioni, artefatto dell'uomo, sono realtà artificiale e soddisfano le tre regole sopra enunciate. La percezione avviene

[4] Domenico Parisi (2001), *SIMULAZIONI – La realtà rifatta nel computer*, p. 42, il Mulino

attraverso lo schermo, la nostra azione su di essa, è permessa dall'uso di periferiche quali sono la tastiera ed il mouse, ed infine la simulazione è un vincolo perché ci permette di fare alcune cose ma altre no, ed è mezzo grazie al quale possiamo svolgere determinate azioni.

Emerge così un'altra novità: le simulazioni sono "realtà". Innanzitutto le simulazioni sono teorie e quindi sono dei mezzi per comprendere la realtà, tuttavia sono anche realtà, questo comporta un'ulteriore novità e cioè, che le teorie (espresse attraverso la simulazione) siano realtà; ciò differisce dalla tradizione, che propone una visione non solo differente, ma opposta, asserendo che le teorie sono una cosa mentre la realtà è un'altra.

Quest'ultimo aspetto presenta la sua importanza quando ad essere oggetto di studio sono fenomeni di per sé unici o del passato. Normalmente si avrebbero delle difficoltà nell'affrontare studi su questi fenomeni per vari motivi che riguardano le metodologie con la quale vengo formalizzate le teorie e cioè, generalizzando, astraendo dal particolare, dall'individuale. Diversamente accadrebbe con l'uso delle simulazioni, le quali essendo realtà, sono in grado di riprodurre una copia delle realtà uniche che si va studiando. Tale elemento va a favore di scienze quali la storia, la psicologia ed altre scienze dell'uomo.

CAPITOLO 2

Vantaggi applicativi delle simulazioni

L'utilizzo di simulazioni permette di superare alcuni limiti delle tradizionali metodologie volte all'analisi ed alla formazione di teorie scientifiche. Grazie all'automatizzazione dei processi di derivazione delle predizioni, attraverso l'uso delle simulazioni, si rende oggettiva quella verifica delle teorie che lo stesso scienziato tradizionalmente operava privatamente, con tutti i dubbi di soggettività che ne derivavano.

Quando una teoria è espressa come simulazione, i risultati che da essa ne derivano, possono essere considerati come le predizioni derivate dalla teoria stessa; quando una simulazione dà determinati risultati si è sicuri che questi discendano dalla teoria. Inoltre le simulazioni, possono essere utilizzate come laboratori sperimentali molto più potenti di quanto non siano i veri e propri laboratori e nei quali è possibile ricreare ambienti, che altrimenti non sarebbero studiabili nel classico laboratorio, per vari motivi che più avanti verranno analizzati. Ancora, le simulazioni permettendo di ricreare ambienti, nel quali eseguire esperimenti, permettono quindi di creare ambienti che non esistono, i "mondi possibili"; in questo modo le possibilità di analisi da parte dello scienziato si ampliano notevolmente, garantendo anche una maggior comprensione di quella che è la realtà "vera". A tale proposito si deve citare un notevole passo avanti per la scienza che le simulazioni permettono, cioè la gestione di sistemi complessi che diversamente non potrebbero essere studiati con metodologie tradizionali a causa dei limiti cognitivi, mnemonici e di calcolo che la mente umana presenta in rapporto a quanto un calcolatore può fare.

Inoltre adottare le simulazioni significa essere spinti verso una visione meno disciplinare della scienza poiché le simulazioni sono un metodo di ricerca adattabile ad ogni aspetto della realtà.

2.1. Automatizzazione del processo di derivazione delle predizioni

Quando una simulazione "gira" in un computer, essa genera dei risultati i quali corrispondono alle predizioni che tradizionalmente uno scienziato elabora sulla base di una propria teoria e che confronterà con osservazioni empiriche. Il fatto che le predizioni vengano generate automaticamente, attraverso le procedure di calcolo eseguite dal computer, sulla base di indicazioni, di regole, di vincoli, definiti all'interno del programma che definisce la teoria, rappresenta un notevole passo avanti che risolve più di un problema.

Gli errori nel derivare le predizioni diminuiscono; infatti, se fosse lo stesso scienziato a formulare le ipotesi di risultato, egli potrebbe imputare a determinati risvolti della sua teoria certi eventi, quando invece, non vi è nessun legame tra le due cose, o almeno non vi è il legame rilevato dallo scienziato stesso. Lo scienziato può commettere errori per vari motivi, ad esempio, possono essere causati dal desiderio di credere che certe predizioni derivino dalla teoria. Con l'utilizzo delle simulazioni si trova soluzione a questo problema, poiché i risultati sono generati all'interno del computer; per cui se la simulazione produce determinati risultati vuole dire che effettivamente c'è una relazione tra i fatti e la teoria espressa dallo studioso.

Secondo vantaggio che l'uso delle simulazioni ci offre è la garanzia di avere una teoria chiara, priva di parti carenti da un punto di vista esplicativo; la ragione di questo è dovuta al fatto che tentando di tradurre in programma una teoria poca chiara, o non si riesce ad esplicitarla in un linguaggio di programmazione, oppure essa non "gira" nel computer.

Un ulteriore vantaggio è legato al fatto che le simulazioni possono essere viste come laboratori virtuali e ciò comporta che cambiando anche solo alcuni elementi all'interno di esse si possono creare notevoli varietà di condizioni da cui derivare predizioni; le simulazioni sono quindi una fonte inesauribile di dati e di risultati che possono essere ricavati da condizioni ambientali che lo stesso studioso può cambiare a suo piacimento. Le loro potenzialità vanno ben oltre quelle dei classici laboratori in cui sono condotti solitamente gli esperimenti.

Riassumendo, si può affermare che in generale, le simulazioni rendono più oggettiva la verifica privata (verifica interna), che lo scienziato fa della propria teoria. La verifica interna, spesso, è poco considerata in rapporto alla verifica empirica dei fatti (verifica esterna), poiché tradizionalmente è svolta dallo scienziato in forma privata. Tuttavia la poca considerazione di questa verifica non ha giusti fondamenti, perché lo stabilire se una predizione empirica dipenda direttamente da una teoria non è cosa così semplice ed ovvia; può ben capitare, infatti, che vi sia un teoria da un lato e che vi sia una predizione da un altro, ma che la prima non spieghi il divenire della seconda. Ma è evidente che il comprendere se una predizione discenda o no da una teoria è fondamentale perché in questo modo si stabilisce la bontà di una teoria, qualora si abbia anche la conferma dei fatti empirici. Se invece la predizione, pur essendo confermata dai fatti, non deriva dalla teoria, non si potrà comunque dire che la teoria data sia la spiegazione dei risultati analizzati.

Le motivazioni che stanno alla base del fatto che una teoria non sia in grado di spiegare una predizione, dalla quale comunque sembra invece derivare, possono essere di svariate tipologie e spesso è difficile individuarle in maniera oggettiva. Tale difficoltà nello spiegare è anche dimostrata dal fatto che le teorie espresse quantitativamente, rispetto a quelle espresse qualitativamente, sono migliori; questo proprio perché l'esprimere i concetti in maniera analitica, quantitativa, rende maggiormente verificabile ed oggettivo il ricavare delle predizioni, rispetto al ricavarle da teorie espresse attraverso simbologie qualitative (come ad es. il linguaggio). Tuttavia tali aspetti espositivi delle teorie

possono essere usati con disinvoltura in particolari campi scientifici che possiamo definire scienze della natura (matematica, fisica, chimica, ecc.), mentre altri campi, le scienze dell'uomo, difficilmente si prestano a definizioni di tipo quantitativo. Grazie all'uso di simulazioni si può superare quest'ostacolo e si può essere sicuri che, se la simulazione gira nel computer, i risultati ottenuti da questa, discendono dalle teorie su cui la simulazione si basa. Inoltre la mole di dati e di possibili differenziazioni delle situazioni in cui applicare la teoria studiata, rendono la verifica attuata con la simulazione più ricca e completa di quella che lo studioso può fare nella propria mente.

L'uso della simulazione favorisce, tra gli scienziati, la formazione di teorie sempre più particolareggiate e sempre più precise a livello espositivo, usando terminologie non vaghe e non ambigue, assicurando così che tutto ciò che serve alla spiegazione della teoria stessa, sia presente e mai tralasciato. A tale proposito, si può citare quanto un noto informatico, *Donald Knuth* (l'autore del software Tex), disse: "la scienza è quella che capiamo e che sappiamo spiegare ad un computer".

2.2. Laboratori sperimentali virtuali

Un altro vantaggio è che le simulazioni possono essere considerate come laboratori sperimentali virtuali che possiedono maggiori potenzialità rispetto ai laboratori reali. Il lavoro in laboratorio, di qualunque genere esso sia, reale o virtuale, permette allo scienziato di fare analisi e controlli sulle situazioni che sono oggetto dei suoi studi, manipolandole come meglio crede, per indagare vari aspetti della sua teoria. Tuttavia l'utilizzo di laboratori reali è possibile soltanto in alcuni campi della scienza, mentre i laboratori virtuali sono utilizzabili da tutte le scienze e quindi per tutti i fenomeni che gli scienziati possono voler studiare.

Le implicazioni ed i vantaggi dovuti a quest'ultima affermazione sono numerosi e richiedono un'analisi più approfondita.

Innanzitutto si possono studiare fenomeni che unicamente per ragioni fisiche non possono essere analizzati in un laboratorio reale, come anche, fenomeni dilatati molto nel tempo e che quindi per ovvi motivi pratici non potrebbero essere oggetto di interesse in un "classico" laboratorio sperimentale. Esempi chiarificatori di queste due tipologie di fenomeni potrebbero essere studi attuati su regioni particolarmente estese, su zone spaziali, sull'intero pianeta, od anche riguardanti condizioni che si sviluppino in secoli, millenni o milioni di anni. Ma nemmeno possono essere sperimentate condizioni relative a situazioni che avvengono molto lontano dal luogo in cui si opera, poiché non potrebbero essere oggetto di manipolazioni da parte dello scienziato, basti pensare a ciò che accade nelle profondità marine. Ancora, quei fenomeni che pur essendo riproducibili e studiabili in laboratorio, risultino difficilmente modificabili e manipolabili dallo sperimentatore o le cui analisi risultino semplicemente molto costose.

Usando le simulazioni nessuno dei problemi sopra citati si può presentare, poiché grazie alle potenzialità di calcolo di un computer è possibile ricreare, analizzare, modificare e manipolare ognuna delle situazioni che presentano le caratteristiche poco fa indicate. Attraverso l'uso di laboratori virtuali, si possono simulare fenomeni enormi oppure piccolissimi, che si estendono molto nel tempo e che distano molto da noi, oppure che richiedono l'uso di costosissime apparecchiature o che risultano difficilmente manipolabili.

In secondo luogo, grazie all'uso di simulazioni, si possono gestire fenomeni risalenti ad epoche lontane del passato e che quindi non possono essere studiati come si potrebbe fare con fenomeni del presente. Nel laboratorio tali eventi non possono essere analizzati semplicemente perché non sono più riproducibili le condizioni in cui si sono verificati. Una simulazione invece può riprodurre sia condizioni del passato, sia del presente rendendo così possibili analisi di fenomeni non studiabili in altro modo.

Ulteriore pregio delle simulazioni è che grazie ad esse è possibile lo studio di fenomeni caratterizzati da notevole complessità, senza dover per forza isolare alcuni fattori per permettere le analisi, cosa questa che avviene nel caso in cui si adoperi un laboratorio "canonico". Infatti, accade che si debbano isolare aspetti di uno stesso fenomeno e che debbano essere analizzati separatamente per permettere di comprendere la dinamica di quanto accade. Questo tuttavia può funzionare per alcuni fenomeni, ma non per altri, mentre nel caso si usi una simulazione, non essendo necessario attuare analisi separate, è possibile ampliare il raggio della fenomenologia complessa che si può studiare.

A spiegare questo limite delle sperimentazioni nei laboratori tradizionali, può essere il fatto che l'insieme di tutti i fattori necessari a definire la situazione in cui deve essere creato l'esperimento va tenuta sotto controllo dallo scienziato stesso e quindi richiede la compresenza di tutti gli elementi nella testa dello scienziato; tuttavia la mente umana ha dei limiti di attenzione e di elaborazioni dati, per questo le semplificazioni, di cui si è parlato sopra, si rendono necessarie al fine di gestire l'esperimento. Detto questo si capisce come alcuni esperimenti non possano essere eseguiti, a causa dell'impossibilità di semplificare a sufficienza le condizioni dell'esperimento stesso, al fine di permettere una gestione da parte dello scienziato sperimentatore. Utilizzando invece una simulazione è il computer a sopperire ai limiti della mente umana, infatti, non è più necessario memorizzare tutti gli elementi di un esperimento e tutte le implicazioni che questi generano tra di loro, poiché vengono gestiti dal calcolatore, grazie alle sue capacità di elaborazione dati.

Ancora, si può dire che, grazie alle simulazioni, sono eseguibili quegli esperimenti che altrimenti non potrebbero essere attuati in un laboratorio reale poiché moralmente discutibili, se non del tutto inaccettabili. Ad esempio, infatti, non è etico modificare la struttura corporea di un essere umano, per poterne valutare le conseguenze, mentre ciò può essere fatto, senza nessun problema etico, in una simulazione.

Infine si può far notare che le simulazioni sono in generale più vantaggiose di una sperimentazione tradizionale, poiché si possono eseguire in un tempo più contenuto ed inoltre, per quanto possano risultare complessi e lunghi, la stesura del codice e tutti i preparativi necessari ad una simulazione, saranno sempre meno costosi, sia in termini di tempo sia in termini di risorse, di ciò che va fatto per un esperimento in laboratorio.

2.3. La creazione di mondi possibili

Quando viene espressa una teoria secondo la metodologia tradizionale, utilizzando il linguaggio verbale e quello matematico, le predizioni che ne derivano, possono essere verificate unicamente attraverso l'osservazione del mondo reale, verificando quindi come realmente le cose sono nella realtà. Tutto ciò è normale, poiché, compito delle scienze è quello di farci capire, attraverso lo sviluppo delle teorie, com' è il nostro mondo.

Grazie all'utilizzo di simulazioni le possibilità si ampliano. E' già stato detto che la simulazione può ricreare, seppur con qualche semplificazione, la realtà e quindi seguendo un logico ampliamento di quest'affermazione, si può dire che le simulazioni sono in grado di creare mondi reali e non; in altre parole, possono creare quelli che stiamo definendo "mondi possibili". Un mondo possibile, non è il mondo reale, ma è un mondo cui sono applicati almeno alcuni dei principi che regolano il mondo reale. Così la simulazione può creare materiali possibili, organismi possibili, menti possibili, società possibili, ecc. L'utilizzo di questi mondi possibili può essere ricondotto ad una maggior comprensione del mondo reale, attraverso lo studio di una possibile evoluzione societaria, si può ad esempio comprendere quali sarebbero le implicazioni di determinate ideologie; idem dicasi nel caso si voglia derivare delle predizioni su tipologie di organismi, infatti, questi potrebbero essere ricreati, con qualche modifica, in un mondo

possibile e si potrebbe così verificare se il comportamento permane immutato o cambia. Successivamente, qualora le teorie inizialmente espresse vengano confermate, si avrebbe un rafforzamento di queste, poiché sia nel mondo reale, sia in quello possibile, il comportamento rimane lo stesso.

Non si deve tuttavia pensare che le simulazioni siano il primo caso in cui gli esseri umani creano mondi possibili, infatti, l'umanità ha creato cose che nella realtà non esistono e vive in mondi che nella realtà prima non esistevano, ma che ora esistono; quasi tutto ciò che accompagna la nostra vita attualmente fa parte di mondi possibili. Attraverso l'uso di simulazioni è possibile creare cose che non esistono, semplicemente simulandole all'interno di un computer. Il vantaggio rispetto a quanto l'uomo ha fatto fino ad ora è lampante, e riguarda il maggior grado di libertà che si gode nel creare mondi possibili all'interno di simulazioni (si pensi ai vincoli etici) ed il minor costo (componente non trascurabile).

Inoltre spesso la creazione di nuove tecnologie è una conferma della teoria che sta alla base della creazione dell'oggetto in questione, questo perché le cose che vengono costruite sono basate sulle nostre conoscenze scientifiche e quindi costituiscono una conferma indiretta della bontà delle nostre teorie. Questo non accade sempre, poiché possono essere costruiti oggetti, teorie, strumenti, senza che via sia nessuna scienza esplicita che sta alla base, sfruttando solamente esperienza, pratica ed intuizione. Storicamente vi sono conferme di tale situazione, e più precisamente si può affermare che nel seicento si è verificata la rivoluzione scientifica e nel settecento la rivoluzione industriale, ma è soltanto dal novecento in poi che la scienza è base per la creazione di ciò che è "nuovo". Nonostante questo, anche oggi una discreta porzione di tecnologie, utili ed efficaci per la nostra società, non hanno base scientifica su cui fondarsi: esempi ne sono la medicina, sia del corpo, sia della mente, l'intelligenza artificiale che ha generato interessanti ritrovati altamente tecnologici, senza che alla base vi fosse una reale comprensione di come funzionano la mente e l'intelligenza.

Quindi anche da quest'ultimo punto di vista le simulazioni di mondi possibili sono una novità poiché alla base di queste ci deve per forza essere una

teoria esplicita, cosa che si è appena detto sopra non essere sempre vera per le pratiche operative. Sotto quest'aspetto la simulazione di un mondo possibile (come quella del mondo reale), è una prova di una teoria esplicita che sta alla base della creazione della simulazione stessa.

Come conferma delle teorie le simulazioni di mondi possibili, si riferiscono a quelle che sono state definite verifiche interne, cioè quelle verifiche che servono per determinare se una teoria è sufficientemente dettagliata, completa e se spiega veramente ciò a cui si riferisce. Ovviamente le simulazioni di mondi possibili non servono per quelle che sono state definite verifiche esterne, infatti, per queste si necessita del controllo della realtà. Tuttavia nel caso i fenomeni che si vogliano studiare siano complessi, è già un notevole passo avanti il fatto di poterli studiare attraverso simulazioni e di poterne ricavare le verifiche interne.

2.4. Lo studio dei sistemi complessi e la non disciplinarità

Si è visto fino ad ora che le simulazioni offrono alla scienza una potente alternativa alla metodologia classica dello sperimentare. Tuttavia la simulazione non è una semplice alternativa a disposizione della scienza per attuare esperimenti, bensì offre qualcosa di più, aggiunge uno strumento che modifica la percezione che lo scienziato ha della realtà e il modo con il quale esso si organizza per studiarla. L'utilizzo delle simulazioni come metodo di ricerca muta la concezione che lo scienziato ha della realtà. Tali mutazioni riguardano due differenti aspetti: innanzitutto la realtà non sarà più vista come insieme di sistemi semplici, ma come insieme di sistemi complessi ed in secondo luogo diventerà meno disciplinare, cioè sarà meno divisa in sottodiscipline.

2.4.1. Lo studio dei sistemi complessi

La tendenza dell'uomo è quella di pensare che la realtà sia fatta di sistemi semplici. In un sistema semplice una singola causa produce un singolo effetto, quindi sapendo che si è verificata la causa si può prevedere che si verificherà la conseguenza e quindi si potrà, nella realtà, far sì che la causa si verifichi o no a seconda che l'effetto sia desiderato o meno. Capita anche che l'effetto sia prodotto da più cause, anche se in genere queste non sono molte, tuttavia l'effetto di ogni singola causa è individuabile e quindi scindibile dagli effetti delle cause concomitanti; questo vuole dire che le diverse cause non interagiscono tra di loro e per questo gli effetti di ogni causa possono essere isolati. Altre caratteristiche dei sistemi semplici meritano di essere considerate:

- I vari stati di un sistema sono prevedibili, quindi il futuro di un sistema semplice è prevedibile, purché se ne conoscano gli stati precedenti.
- Se un sistema semplice si trasforma nel tempo, le modalità di cambiamento che lo caratterizzano sono prevedibili.
- Se un sistema semplice riceve "perturbazioni" da un evento esterno, l'effetto di tale sistema è proporzionale all'entità della perturbazione stessa, questo poiché non sono relazionati gli effetti facenti parte del sistema con gli effetti derivanti dalla perturbazione, quindi una perturbazione grande produrrà un grande effetto, mentre una perturbazione piccola produrrà un piccolo effetto.
- Due sistemi semplici che partano da diverse condizioni iniziali, avranno uno sviluppo differente che sarà proporzionale all'entità delle diversità. Per condizioni di partenza molto simili, non saranno notabili diversità particolarmente rilevanti.

- Un sistema semplice può essere isolato dal suo contesto, quindi il contesto in cui si opera non è funzionale ai risultati del sistema stesso.

- Un sistema semplice non presenta rapporti di causazione reciproca, quindi se A influenza B, B non può influenzare A, cioè un elemento del sistema ne influenza un altro ma non viceversa; inoltre se un sistema "vive" in un ambiente, l'ambiente stesso può influenzare il sistema ma non viceversa.
- Un sistema semplice tende a non essere parte di una gerarchia di sistemi, in cui un sistema, posizionato ad un certo livello in una scala gerarchica, è elemento di un altro sistema posizionato ad un più alto livello della scala gerarchica; in sistemi gerarchici vi è reciproca influenza tra un sistema e l'altro.
- Le varie parti che compongono il sistema semplice e che concorrono a formare gli effetti dello stesso, sono ben individuabili.
- Un sistema semplice può essere riprodotto in copie identiche.

All'essere umano sembra di comprendere veramente qualcosa quando oggetto della sua analisi è un sistema semplice, infatti, in un sistema di questo tipo, possono essere individuate le cause che generano gli effetti all'interno dello stesso sistema, si effettuano manipolazioni varie per ottenere effetti sempre differenti, ne si può prevedere il futuro, inoltre non è necessario che vi sia un ambiente da sfondo e si può riprodurre un sistema semplice identicamente ad un altro.

Tuttavia a ben guardarla la realtà è più fatta di sistemi complessi che di sistemi semplici, con l'aggravante che le caratteristiche dei primi sono l'opposto delle caratteristiche dei secondi.

In un sistema complesso infatti, non vi è indipendenza fra i suoi elementi e quindi essi concorrono unitamente e non unitariamente alla formazione delle cause, così le relazioni tra le cause sono non lineari e non sommatorie (come nel caso dei sistemi semplici); in questo modo non è possibile isolare un elemento da un altro al fine di eliminare un effetto piuttosto che un altro dal sistema in questione. Gli elementi di un sistema complesso sono solitamente moltissimi e

diversi uno dall'altro, tra di loro si influenzano localmente, cioè, un elemento interagisce solo con un ristretto numero di altri elementi che compongono il sistema e da tali interazioni emergono effetti relativi all'intero sistema che non sono deducibili o prevedibili, pur avendo conoscenza della struttura dei singoli elementi e delle interazioni reciproche.

Ulteriori caratteristiche appartenenti ai sistemi complessi che li differenziano dai sistemi semplici sono:

- Gli stati futuri di un sistema complesso non sono solitamente prevedibili sulla base degli stati precedenti.
- Le trasformazioni nel tempo di un sistema complesso non sono prevedibili, o lo sono poco.
- Le perturbazioni cui un sistema è sottoposto, producono degli effetti che spesso non sono proporzionali all'entità della perturbazione; così, perturbazioni di notevoli dimensioni possono essere assorbite dal sistema senza che gli effetti siano mutati nella stessa proporzione.
- Le condizioni iniziali influenzano notevolmente l'evolversi di un sistema complesso, così variazioni appena percettibili all'inizio, sono causa di notevoli differenze al trascorrere del tempo.
- L'ambiente su cui un sistema complesso poggia è rilevante, per questo non può essere tralasciato.
- Sono presenti, all'interno di sistemi complessi, rapporti di causazione reciproca; un elemento ne influenza un altro, ma è anche influenzato da esso; capita anche che l'intero sistema sia influenzato dall'ambiente su cui poggia e, a sua volta, influenza l'ambiente.
- I sistemi complessi, solitamente, sono inseriti in gerarchie di sistemi, dove un sistema, posizionato ad un determinato livello, è parte di un più ampio sistema, di rango superiore, e tra essi vi sono influenze reciproche.

- Non è ben identificabile il ruolo che ogni singolo elemento di un sistema complesso ha all'interno della generazione degli effetti, non è quindi isolabile al fine di individuare la conseguenza che produce.
- Non si possono fare copie identiche di sistemi complessi.

A livello di pensiero comune i sistemi semplici sono in stretto legame con un qualcosa che è comprensibile, che è prevedibile e controllabile, mentre i sistemi complessi sono ciò che appare meno prevedibile, meno controllabile e meno comprensibile. La scienza in accordo con tale visione si è occupata principalmente di sistemi semplici. Inoltre è comprensibile, sempre rifacendosi a quanto appena detto, la situazione che ha portato le scienze della natura ad evolversi maggiormente delle scienze dell'uomo, infatti le prime sono caratterizzate da una maggioranza di sistemi semplici, mentre le seconde, al contrario, sono caratterizzate da una maggioranza di sistemi complessi.

Quindi la ragione per cui gli scienziati si sono principalmente orientati verso lo studio di sistemi semplici, tralasciando i sistemi complessi, è dovuta al fatto che i sistemi semplici si prestano ad essere meglio studiati con gli strumenti tradizionali della scienza. Fino a che le teorie saranno elaborate nella testa degli scienziati e verranno trasmesse agli altri attraverso il linguaggio, come è stato fino ad ora, le teorie che potranno essere studiate saranno solo quelle relative a sistemi semplici, i quali possono essere elaborati dalle capacità cognitive dell'uomo, mentre i sistemi complessi non possono essere gestiti dalla mente umana perché formati da troppe parti che hanno troppe interazioni tra loro. Lo stesso dicasi per gli strumenti matematici tradizionali della scienza, i quali risultano appropriati per lo studio di sistemi semplici ma diventano inadatti quando li si applica ai sistemi complessi, i quali presentano numerose variabili e complesse interazioni tra queste. A tutto ciò si aggiunga che i sistemi complessi presentano delle caratteristiche di imprevedibilità e di irripetibilità che gli strumenti tradizionali hanno difficoltà a gestire.

Nonostante i problemi appena citati siano già di una certa rilevanza, l'inadeguatezza dei metodi tradizionali nel trattare i sistemi complessi appare ancor più evidente nell'uso del metodo degli esperimenti di laboratorio. Si sa che il metodo sperimentale è un elemento fondamentale della scienza, tuttavia questo è adatto ai sistemi semplici e non a quelli complessi. In laboratorio lo scienziato opera dei cambiamenti su una causa, facendo in modo che si verifichi o che non si verifichi, ovvero variandone quantitativamente il valore, per poi osservare l'effetto delle sue manipolazioni. Questo metodo è però valido per i sistemi semplici, in cui gli effetti sono il prodotto di singole cause; qualora fossero più cause a concorrere alla formazione di un effetto, sarebbe comunque possibile isolare le singole cause, poiché la loro azione nel generare un effetto è indipendente da quella delle altre cause. Invece esattamente il contrario accade per i sistemi complessi, nei quali gli effetti sono la risultanza di molte cause che tra loro interagiscono, quindi non ha senso isolare una singola causa per poi studiare gli effetti che si generano nell'intero sistema. Inoltre le manipolazioni che vengono effettuate in fase di sperimentazione, sulle condizioni presenti nel sistema richiedono una certa isolabilità del sistema stesso dall'ambiente cui si poggia; questo, ancora, funziona per i sistemi semplici, ma non per i sistemi complessi che tendono a non essere isolabili dall'ambiente in cui "vivono", infatti, per questi, anche piccole variazioni dell'ambiente in cui vivono, possono causare effetti completamente differenti. O ancora, il metodo sperimentale, richiede che i fenomeni siano ripetibili e riproducibili; questo può essere fatto per i sistemi semplici, ma non per i sistemi complessi, che come detto sopra, sono molto sensibili a variazioni dell'ambiente iniziale, per cui risulta impossibile la riproducibilità e la ripetibilità dei fenomeni stessi.

Attualmente la scienza si sta rendendo conto, o meglio, sta accettando il fatto che la realtà è composta di sistemi complessi, più che di sistemi semplici. I sistemi complessi, compaiono in ogni angolo del nostro mondo: "sono presenti nei fenomeni meteorologici, nella dinamica dei fluidi, nel processo di sviluppo attraverso il quale un corredo genetico viene tradotto nelle caratteristiche

dell'organismo adulto, nelle interazioni tra i miliardi di cellule che determinano quello che chiamiamo il comportamento degli organismi e, nel caso degli esseri umani, la loro vita mentale, nelle interazioni tra le diverse componenti di un ecosistema, nelle interazioni tra gli individui tra di loro e con le strutture e istituzioni sociali che costituiscono le società umane e determinano il modo in cui cambiano nel tempo, nelle interazioni tra soggetti economici che costituiscono un mercato".[5]

Appurato il fatto che la realtà è per buona parte composta di sistemi complessi e che analizzarne una parte, riducendola a sistemi semplici, pur essendo buona cosa, non è per nulla sufficiente, rimangono da ricercare e studiare gli elementi concettuali e metodologici adatti allo studio dei sistemi complessi, visto che gli strumenti tradizionali non sono sufficienti.

Soltanto recentemente la scienza ha elaborato gli strumenti adatti allo studio dei sistemi complessi e tra questi le simulazioni sono uno dei più importanti, se non il più importante. Forse se non fossero esistiti i computer e le simulazioni, probabilmente i sistemi complessi non sarebbero stati studiati e nemmeno scoperti dalla scienza.

Le teorie espresse attraverso le metodologie tradizionali sono adatte ai sistemi semplici, ma non a quelli complessi, perché sono vincolate dai limiti cognitivi, mnemonici, di ragionamento dell'uomo; infatti la teoria deve essere tenuta a mente dallo scienziato che la elabora. La teoria di un sistema complesso, invece, è anch'essa complessa, poiché riguarda molti elementi, ha molte parti, e tutte queste interagiscono tra loro creando una struttura che non può essere gestita dalla mente umana a causa di limitazioni, di memoria, di attenzione e di ragionamento. Tutto cambia quando la teoria è espressa attraverso l'uso di un computer, infatti, le potenzialità del computer di memorizzare, di elaborare dati, sono decisamente maggiori di quelle dell'uomo; il computer è in grado di gestire una teoria complessa. L'uomo può inserire uno alla volta gli elementi che compongono il sistema, può modificare la teoria che è incorporata nel

[5] Domenico Parisi (2001), *SIMULAZIONI – La realtà rifatta nel computer*, p. 76, il Mulino

programma, può verificare il funzionamento del tutto, può manipolare le variabili, tuttavia è il computer che "vede" la teoria nel suo insieme, che elabora i dati e fornisce i risultati allo scienziato, che così può comprenderne il funzionamento.

Il fatto che l'uomo non sia in grado di gestire un sistema complesso con la propria mente e che abbia bisogno di ricondurlo ad un sistema semplice per poterlo elaborare e studiare, rende necessario l'impiego delle simulazioni nel computer come metodo per poter progredire nello studio di tali fenomeni che altrimenti non sarebbero analizzabili.

Inoltre, il fatto che la tradizionale sperimentazione in un laboratorio reale sia adatta per i sistemi semplici ma non per quelli complessi, dipende da una caratteristica delle simulazioni che le contraddistingue in modo fondamentale dalla metodologia tradizionale di sperimentazione.

La metodologia tradizionale procede attraverso l'*analis*i dei fenomeni, mentre il metodo che sfrutta le simulazioni offre una *sintesi* di fenomeni. Il metodo tradizionale opera un'analisi della realtà, nel senso che parte dalla realtà, la scompone nelle sue parti e ricostruisce i fenomeni ricomponendo le varie parti con la teoria ed il ragionamento; ad esempio volendo studiare comportamenti della società umana, questa viene scomposta negli individui che la compongono e poi, sarà compito delle teorie e dei ragionamenti basati sulle teorie dimostrare come attraverso l'analisi dei vari componenti, si possa arrivare al fenomeno societario di partenza.

Le simulazioni invece sintetizzano la realtà nel senso che partendo dalle varie componenti si arriva a comprendere cosa accade quando queste interagiscono tra loro; girando nel computer la simulazione mostra come le varie componenti di cui un sistema è composto, interagendo tra loro, formino il sistema in analisi. L'assunto su cui le simulazioni si basano è che la realtà, per essere conosciuta, non va semplicemente analizzata, ma va ricreata, partendo dalle sue componenti base.

La scienza tradizionale cerca di comprendere la realtà così com'è, invece attraverso le simulazioni, si cerca di capire la realtà ricreandola. Il principio è: "se riesco a riprodurre la realtà, questo vuole dire che l'ho capita".[6]

Forse i sistemi semplici possono essere studiati e capiti semplicemente analizzandoli, ma i sistemi complessi, per essere compresi, devono essere ricreati.

2.4.2. La non disciplinarità

L'evoluzione della scienza moderna, nata nel Seicento, ha portato notevoli progressi e questa situazione ha portato con sé le suddivisioni disciplinari. Con il progredire della scienza e l'approfondirsi delle conoscenze è iniziato un processo di frammentazione della scienza con il formarsi di nuove discipline e di sottodiscipline scientifiche. Non si può più dire che oggi esistano la scienza della natura e la scienza dell'uomo; la scienza della natura è oggi suddivisa in fisica, chimica, biologia, con tutte le relative sottodiscipline, mentre la scienza dell'uomo è suddivisa in psicologia, sociologia, antropologia, economia, storia, con tutte le sottodistinzioni e specializzazioni. La suddivisione in discipline è giustificata dal fatto che i fenomeni che la realtà presenta, quando vengono studiati, sono talmente tanti e diversi, che appare impossibile che dei singoli scienziati o delle singole strutture organizzative, possano conoscerli e capirli tutti.

C'è da dire anche che la frammentazione disciplinare non è stata semplicemente una conseguenza dell'evoluzione delle conoscenze scientifiche ma è anche stata l'elemento che ha permesso tutti i progressi ottenuti. Infatti, l'approfondimento di una materia è permesso quando lo scienziato può concentrarsi unicamente sul fenomeno che sta studiando, tralasciando altri campi della scienza.

---

[6] Domenico Parisi (2001), *SIMULAZIONI – La realtà rifatta nel computer*, p. 79, il Mulino

E’ anche vero che in parte la suddivisione delle discipline scientifiche è un ostacolo per lo sviluppo delle conoscenze scientifiche, infatti, è palese che la realtà è un tutto integrato, un insieme di fenomeni collegati uno all’altro, non un insieme di fenomeni che sono indipendenti tra loro, come tende a vederla una scienza suddivisa in discipline e sottodiscipline. La frammentazione disciplinare è da ostacolo al progresso della conoscenza ed alla comprensione della realtà per vari motivi, innanzitutto, non è possibile comprendere alcuni fenomeni se non considerando questi correlati ad altri ed una scienza divisa in discipline avrebbe difficoltà a carpire tali collegamenti poiché riguardano più branche della scienza stessa. Inoltre per la mente umana comprendere qualcosa, significa trovare il generale nel particolare per cui, la scienza non può essere un semplice elenco di discipline e sottodiscipline, ciascuna delle quali avente i suoi meccanismi e processi generali per spiegare una parte della realtà, ma deve trovare i meccanismi ed i processi che siano in grado di spiegare fenomeni appartenenti a differenti discipline. Ancora si può dire che la realtà non è un semplice elenco di fenomeni, ma è stata caratterizzata da processi evolutivi, nei quali a fenomeni se ne sono sostituiti o aggiunti altri; la scienza deve aiutarci a comprendere la realtà e per questo deve poter ricostruire il processo storico del susseguirsi di fenomeni e spiegarne il perché. Tuttavia per poter spiegare tale processo evolutivo è necessario possedere un insieme di concetti, teorie e metodi di ricerca che si possano applicare ugualmente sia all’inizio sia alla fine del processo di evoluzione e trasformazione; questo però è proprio quello di cui non dispone una scienza suddivisa in sottodiscipline.

Le teorie elaborate secondo le metodologie tradizionali colgono soltanto parti ristrette della realtà e quindi l’osservazione dei fenomeni non può che riguardare aspetti particolari dei fenomeni; in tale situazione le suddivisioni disciplinari sono necessarie, ma rimane che la disciplinarietà della scienza può essere da freno alla comprensione della realtà.

A volte la scienza tenta di superare tale limite ricorrendo alla interdisciplinarità e cioè mettendo insieme più scienziati appartenenti a diverse

discipline per affrontare e risolvere problemi appartenenti ad una sola disciplina. Tuttavia l'interdisciplinarità presenta dei problemi giacché il dialogo tra scienziati diversi, abituati a trattare fenomeni di tipo diverso, con metodi diversi, con regole diverse, risulta difficoltoso. Quindi il problema della disciplinarità rimane. Ciò che servirebbe per comprendere meglio la realtà è un qualcosa che stesse prima della formazione delle discipline. Le simulazioni presentano la caratteristica di essere non disciplinari, per cui la scienza che si dota, tra i suoi strumenti di ricerca del metodo simulativo, otterrà una minor disciplinarità ed una maggior comprensione della realtà.

Le simulazioni sono intrinsecamente non disciplinari, perché risolvono molti dei problemi a causa dei quali la scienza è dovuta ricorrere alla suddivisione in sottodiscipline. Infatti, le grandi risorse di memoria e di calcolo a disposizione dei computer soppiantano le "carenze" di memoria e di calcolo della mente umana. Lo scienziato solitamente ha familiarità solo con una tipologia di esperimenti e difficilmente riesce a gestire le interazioni tra i vari fenomeni. Il computer non ha questi problemi, perché si può tranquillamente costruire un programma che contenga dati riferiti a fenomeni di tipo diverso e la simulazione che ne deriva può incorporare una teoria che tenga in considerazione le interazioni fra i vari fenomeni. In una simulazione i dati possono essere inseriti passo dopo passo da più scienziati, aggiungendo elementi, senza perderne altri. Non ci sono limiti nell'inserire conoscenze all'interno di un computer, a differenza di quanto invece si può fare con la mente di un uomo.

Le simulazioni, a differenza delle scienze disciplinari, non usano metodi diversi a seconda delle tipologie di fenomeni che vengono studiati. La suddivisione in discipline e sottodiscipline vede anche una frammentazione dei metodi, che queste usano per la ricerca, frammentazione che è anche più ampia di quella delle discipline stesse; quindi se i metodi sono tanti non si può nemmeno pretendere che uno scienziato padroneggi e sia esperto di più metodi e questa situazione non fa che giustificare ed alimentare la suddivisione in discipline. Anche per tale problema le simulazioni offrono una soluzione che riduca la

disciplinarità della scienza, in quanto la simulazione è un metodo che si può adattare allo studio di fenomeni di qualunque tipo. Una volta compreso cosa sono le simulazioni, capiti i loro limiti, le loro possibilità, i loro aspetti tecnici, queste possono essere usate per lo studio di qualsiasi fenomeno e per ciò significa che le simulazioni possono diventare un elemento comune a tutte le scienze, con un potere unificante e non disciplinare nei confronti della scienza nel suo complesso.

Dovendo cogliere l'aspetto unificante delle simulazioni e quindi la loro essenza si deve prima considerare qual'è il metodo con il quale lo scienziato affronta la ricerca; gli scienziati tendono a semplificare la realtà per poterla studiare e comprendere, però, quando ciò si verifica per le simulazioni, essendo esse realtà (virtuale), riescono a mantenere la completezza e l'integrità della realtà "reale".

CAPITOLO 3

Problemi delle simulazioni

3.1. Critiche confutabili

Le simulazioni come ogni cosa, presentano dei problemi e delle limitazioni. Tuttavia, la maggior parte delle critiche rivolte alle simulazioni sono dovute più che altro ad una mancata comprensione delle simulazioni stesse, e non ai veri e propri limiti che le contraddistinguono.

Per prima cosa le simulazioni vengono criticate per rappresentare in modo troppo semplice la realtà, senza considerare che qualunque teoria, elaborata secondo le metodologie tradizionali della scienza, è comunque una semplificazione della realtà ed è proprio grazie a questa caratteristica che per l'uomo è possibile comprendere l'essenza dei fenomeni, che altrimenti, a causa della loro complessità, non sarebbero comprensibili. Questo limite può essere già ora, anche se in parte, superato, poiché non si deve dimenticare che a supportare le simulazioni vi sono grandi risorse di memoria e di calcolo del computer che, con il tempo, permettono l'introduzione di nuove caratteristiche e dettagli, che rendano la simulazione il più vicino possibile alla realtà.

Le simulazioni sono inoltre criticate perché non dicono nulla di nuovo rispetto a quanto non si sappia già, in quanto ricreando la realtà la riproducono e basta; questo non è vero perché spesso dalle simulazioni emergono fenomeni che non erano previsti, sulla base delle conoscenze che si avevano della realtà. Inoltre il fatto che una simulazione riproduca risultati identici alla realtà, deve essere percepito come una conferma della bontà della teoria che sta alla base della simulazione.

Ancora, viene criticato il fatto che sia inutile simulare la realtà, poiché non la si conosce approfonditamente; ciò è errato perché sarebbe inutile simulare, o in generale studiare, qualcosa che è già conosciuto alla perfezione. E' evidente, che tale critica ha delle basi molto deboli e se appoggiata dovrebbe confuare tutto il sistema scientifico in generale e non solo il metodo simulativo.

Altra problematica sollevata è relativa al fatto che le simulazioni non ci fanno capire cosa accade realmente nel sistema che viene simulato. Questo è vero fino ad un certo punto, in quanto lo scienziato manipolando i vari elementi può comprendere, sulla base dei risultati ottenuti, quali siano le dinamiche che si sono svolte nell'ambiente simulato.

3.1.1. Le simulazioni sono una semplificazione della realtà

Spesso le simulazioni sono criticate perché sono una semplificazione della realtà. Gli scienziati ben sanno che la realtà è estremamente complessa e quanta fatica ed impegno richieda la scoperta e la comprensione dei fenomeni, quindi sembra a loro che le simulazioni siano qualcosa di troppo semplice per poter spiegare la realtà; addirittura le simulazioni vengono a volte prese come dei giochi, divertenti, ma incapaci di essere un elemento utile per aiutare a comprendere la realtà.

Tale critica è eccessiva e denota una mancata comprensione di quello che realmente sono le simulazioni. Le simulazioni sono innanzitutto la traduzione in programma per computer di una teoria, e tutte le teorie, per assolvere al loro compito, cioè quello di far comprendere i fenomeni, sono semplificate rispetto alla realtà. Le teorie, quindi sono utili, proprio perché semplificano ciò che rappresentano e permettono così all'uomo di cogliere l'essenza dei fenomeni, liberandoli dalla complessità e dalla varietà. Non è per cui corretto e sensato criticare le simulazioni perché sono troppo semplificate, in quanto le teorie espresse in modo tradizionale, sono anch'esse delle semplificazioni della realtà e

riusciamo a comprenderle proprio perché sono più semplici della realtà. Ciò che preme è capire se le semplificazioni sono orientate nella giusta direzione e se non sono omessi degli elementi fondamentali per capire l'essenza del fenomeno che viene studiato. Questo però vale per ogni teoria, espressa come simulazione o espressa secondo i modi tradizionali; quindi una simulazione che semplifica è una simulazione a posto, purché semplifichi nel modo giusto, altrimenti, in caso contrario, non sarebbe sensato abbandonare la simulazione, piuttosto andrebbe corretta, facendo sì che semplificazioni avvengano in modo corretto.

Un elemento che giustifichi in parte questa critica tuttavia c'è, infatti, le simulazioni sono innanzitutto l'espressione della realtà attraverso il computer, però sono anche dei sistemi che pretendono di riprodurre la realtà ed i suoi fenomeni. Di conseguenza, possono essere criticate perché la rappresentazione che danno della realtà è troppo semplificata. Per l'essere delle teorie, le simulazioni, non possono che semplificare, però essendo delle riproduzioni della realtà, dovrebbero rispecchiare molti aspetti, caratteristiche e dettagli dei fenomeni che riproducono.

Ciò che è importante sottolineare adesso, è che, mentre le teorie ed il metodo della sperimentazione tradizionali rimarranno sempre semplificazioni della realtà, le simulazioni, con il tempo, con l'innovazione tecnologica e con lo sviluppo della tecnica simulativa, diventeranno sempre più sofisticate ed elaborate, rendendo così la rappresentazione dei fenomeni che studiano, sempre più vicina alla realtà.

3.1.2. Le simulazioni non dicono nulla di nuovo

Altra critica che viene rivolta alle simulazioni è il fatto che queste non ci dicano nulla di nuovo rispetto a quanto non si sappia già della realtà che si sta studiando; quindi viene spontaneo affermare che se nulla di nuovo ci viene detto, è inutile utilizzarle, poiché sono solo uno spreco di tempo.

A questa critica bisogna innanzitutto controbattere affermando che spesso si verifica che la simulazione non restituisca proprio quello che ci si aspettava, ma che produca dei fenomeni che differiscono da quelli che si erano usati come basi di partenza. In alcuni casi le differenze possono essere così significative che i fenomeni ottenuti ci stupiscono a tal punto da non poter ricondurre la realtà, usata come partenza per l'esperimento, ai risultati dell'esperimento stesso. Si può così affermare che è assolutamente errato proporre come critica per le simulazioni il fatto che queste si limitino a ridirci quanto già sappiamo sulla realtà.

Inoltre le simulazioni ci dicono qualcosa che comunque non sapevamo sui fenomeni che studiamo e cioè ci informano sull'effettivo contenuto empirico delle teorie che sono alla base dei nostri esperimenti. Infatti, è notevole l'importanza rivestita dalla dimostrazione empirica di una teoria al fine che questa possa essere definita scientifica; tale contenuto empirico coincide con le predizioni che si possono ricavare, sulla base di un teoria, e nel successivo controllo della corrispondenza di queste con la realtà. Spesso capita che lo scienziato non conosca tale contenuto scientifico ma le simulazioni sono in grado di chiarire questo punto, offrendo così allo studioso qualcosa di nuovo.

I risultati delle simulazioni non sono semplici repliche della realtà anche perché i fenomeni empirici non incorporano già una teoria elaborata da uno scienziato che cerca di spiegarli e di comprenderli, mentre i fenomeni simulati, anche se producono gli stessi risultati della realtà, si fondano su teorie interpretative della realtà. Per cui una simulazione è importante anche se riproduce fenomeni empirici che sono già noti; il fatto che i risultati dell'ambiente simulato coincidano con la realtà, significa che il fondamento teorico, che sta alla base della simulazione è corretto.

Va anche ricordato che rispetto a qualche anno fa durante il quale gli scienziati avevano a che fare con teorie e realtà, ora si trovano davanti un terzo elemento, i risultati delle simulazioni, elemento che può essere classificato come un ibrido dei primi due, in quanto è realtà, perché percepito attraverso i nostri

sensi, ma è anche l'espressione di una teoria. Questo può lasciare perplesso lo scienziato, ed indurlo a diffidare ed a trovare sempre nuove critiche, peraltro non ponderando quelle attuabili nei confronti dei metodi tradizionali, ma rimane una novità concettuale basilare relativa al metodo della sperimentazione.

3.1.3. La non profonda conoscenza della realtà rende inattuabili le simulazioni

I critici delle simulazioni espongono un'ulteriore argomentazione affermando che: il fatto che le simulazioni vogliano riprodurre la realtà è un conto, però della realtà l'uomo sa molto poco e quindi risulta inutile riprodurla, perché il risultato sarebbe una riproduzione priva di significato scientifico, a causa della sua incompletezza e penuria di particolari.

Ma, se fosse realmente così e se si dovesse veramente aspettare di conoscere perfettamente la realtà prima di poterla simulare, non si capirebbe a cosa potrebbero servire le simulazioni; il senso di tale affermazione è che se già si conosce tutto su un argomento a nulla servirebbe studiarlo ancora, poiché già si conosce ogni suo funzionamento, ogni sua particolarità, ecc.

La simulazione serve proprio quando non si conosce a fondo qualcosa e lo si vuole studiare, per poterne comprendere l'essenza, per ricavare delle spiegazioni a dati fatti empirici. Ciò è vero in due sensi. Innanzitutto, la simulazione è una teoria ed in quanto tale è utile per andare al di là di un semplice insieme di fatti e per poter comprendere perché tali fatti si verifichino; questo è il compito delle teorie in generale e quindi anche delle simulazioni. In secondo luogo le simulazioni possono essere da guida per la scoperta di altri fenomeni, rispetto a quelli che si stanno studiando; per ciò possono essere considerate come guida per indirizzare la ricerca scientifica, nello scoprire se i "nuovi" fenomeni, ricavati dai risultati delle simulazioni, abbiano un riscontro nella realtà. Quest'aspetto è comune anche alle teorie scientifiche espresse in

modo tradizionale, che oltre a spiegare fenomeni già noti, possono indicare l'esistenza di altri fenomeni e così spingere gli studiosi ad analizzarli. Una simulazione, qualora dia dati discordanti con quelli della realtà, può indurre lo scienziato a modificare la teoria che ne sta alla base, oppure può suggerire lo studio di nuovi fenomeni che prima non si erano considerati, quindi stimolare il ritorno allo studio della realtà per il reperimento di nuovi dati, al fine di costruire altre simulazioni.

3.1.4. Le simulazioni riproducono, ma non spiegano

Le simulazioni pur essendo in grado, in alcuni casi, di riprodurre il fenomeno studiato, non permettono di comprendere più a fondo tale aspetto della realtà; è questa una delle critiche che viene indirizzata contro il metodo delle simulazioni. Questo per vari motivi. Una prima limitazione è relativa al fatto che le simulazioni essendo espresse in linguaggio informatico, possono essere comprese nella loro essenza solo da esperti informatici; a questa obiezione si può facilmente obiettare che anche le teorie espresse in modo tradizionale hanno alla loro base un contenuto matematico, che per essere compreso, deve presupporre un determinato livello di conoscenza della matematica. Questo vuole dire che anche le teorie "classiche" richiedono determinate conoscenze specifiche per essere comprese e quindi anch'esse non possono essere intese nella loro essenza da chiunque. Nel loro significato, invece, possono essere comprese anche senza una base matematica, nella maggior parte dei casi, ma questo accade anche per le simulazioni, che attraverso il monitor del computer, ci fanno capire cosa succede nell'ambiente simulato.

Altro aspetto sottolineato dai critici, è il fatto che nelle simulazioni si conosce la teoria che sta alla base, quindi gli elementi che la compongono, ed i risultati che emergono dal suo girare in un computer, però nulla si può dire realmente su ciò che accade all'interno della simulazione stessa; in altre parole si

sa cosa si inserisce e quale risultati dà una simulazione, ma non si capisce il perché.

Tuttavia anche a questa critica si può facilmente obiettare che, come detto nel capitolo precedente, le simulazioni sono laboratori virtuali, per cui lo scienziato può tranquillamente modificare a suo piacimento gli elementi che costituiscono la simulazione per vedere quali siano gli effetti provocati dai cambiamenti; inoltre tale operazione è meglio gestibile in una simulazione piuttosto che in un laboratorio reale. Questo può aiutare lo scienziato a comprendere come una simulazione funzioni e quali siano le dinamiche degli eventi per poi comprendere il perché dei risultati.

Ancora se questo non fosse sufficiente, lo studioso potrebbe tranquillamente tradurre in linguaggio matematico la teoria che sta alla base della simulazione, in modo da poter comprendere secondo metodologie tradizionali quello che si verifica nel fenomeno simulato.

3.2. Reali problemi delle simulazioni

Come tutte le cose, anche le simulazioni, hanno dei limiti reali che sono sottolineati giustamente dai critici, ma non consistono in veri e propri problemi che minano l'efficienza di questa nuova metodologia di ricerca. Tuttavia tali limiti sono per lo più dovuti alla nostra incapacità di usare correttamente questo metodo ed è quindi verosimile che possano essere superati in futuro, grazie alla maggior dimestichezza degli scienziati con le simulazioni.

Lo scienziato che utilizza le simulazioni corre il rischio di legarsi troppo ai mondi simulati che crea, attuando semplicemente la verifica interna, tralasciando l'importante verifica esterna; la verifica esterna è un elemento della ricerca che non può essere tralasciato, se si vuole rendere questa tecnica un buon metodo per fare sperimentazione. Si è inoltre visto che essendo le simulazioni delle teorie,

anch'esse tendono a semplificare la realtà che studiano, però spesso si verifica il problema delle semplificazioni che avvengono nella direzione sbagliata; capita così a volte che si semplifichi proprio un elemento che è fondamentale per la comprensione del fenomeno analizzato. Oppure si verifica quasi l'opposto e così vengono inseriti particolari e dettagli nella simulazione che la rendono più complessa, ma il problema che emerge è la non corrispondenza dei dettagli inseriti con la realtà che viene simulata.

3.2.1. La tendenza a fare poca verifica esterna

Uno dei più rilevanti problemi delle simulazioni è legato agli scienziati che le usano; questi spesso tendono a dare maggior peso alle verifiche interne a discapito delle verifiche esterne. La verifica interna consiste nel valutare se derivino effettivamente le predizioni che si pretende che da essa siano ricavabili; ovviamente ci si deve occupare anche della verifica di quelle implicazioni che pur derivano dalla teoria ma che non servono affinché la teoria risulti verificata. La verifica interna di una teoria è un elemento importante della ricerca scientifica, soprattutto quando i fenomeni della realtà che vengono studiati sono complessi e complesse sono anche le teorie che ne stanno alla base e cercano di spiegarle. Tuttavia la ricerca scientifica non può fermarsi alla semplice verifica interna, ma deve adoperarsi per eseguire anche la verifica esterna. La verifica esterna è semplicemente il controllo dei risultati ottenuti in fase di simulazione con la realtà ad essi relativa, si controlla cioè che quello che è accaduto nella simulazione corrisponda ai fatti empirici. Si verifica di frequente che in questa verifica esterna la ricerca sia un po' carente; spesso lo scienziato che studia i fenomeni attraverso la simulazione si accontenta di produrre la simulazione, studiarne i risultati, modificare le condizioni per analizzare le mutazioni del sistema, ma non si interessa molto di verificare qual è la reale corrispondenza tra

i risultati ottenuti e la realtà osservabile. Si potrebbe obiettare a tali affermazioni che uno scienziato, prima di costruire una simulazione, abbia ben presente un fenomeno, che poi è il soggetto che vuole analizzare, e che quindi al momento dell'ottenimento dei risultati, faccia implicitamente un'analisi con la realtà dalla quale ha ricavato lo spunto per il suo studio. Ciò è vero, ma quello che in questa sede si vuole criticare è che il confronto tra risultati della simulazione e realtà, non avviene in modo sistematico, come invece dovrebbe essere, per stabilire poi la bontà dell'analisi condotta. Per far sì che le simulazioni possano diventare uno strumento valido per la ricerca scientifica è necessario che i raffronti su detti siano espliciti, dettagliati ed ampi. Il pericolo individuato è quindi il troppo coinvolgimento dello scienziato nei suoi mondi artificiali ed il tralasciare il punto di vista della realtà empirica, che invece deve essere ben presente allo studioso. Le motivazioni di tale mancato raffronto, possono anche essere dovute al fatto che la mole di lavoro richiesta per la costruzione di una simulazione è notevole, quindi lo scienziato si trova spesso in difficoltà nel dover dedicare ulteriore tempo ad una verifica esterna sistematica ed approfondita o nel dedicarsi allo studio della letteratura empirica. Altra ragione è che il metodo della simulazione è ancora poco diffuso all'interno delle discipline tradizionali, dove invece notevole è la familiarità con la letteratura empirica. Inoltre, non bisogna dimenticare che le simulazioni possono essere usate a fini unicamente pratici per ricavare ad esempio nuove tecnologie; in questo caso il confronto con la realtà è superfluo poiché non è parte degli obiettivi che il ricercatore si prefigge.

3.2.2. Le semplificazioni spesso non avvengono nel senso giusto

Altro problema è relativo al fatto che, come si è detto precedentemente, le simulazioni sono una semplificazione della realtà; le simulazioni sono prima di tutto delle teorie ed in quanto tali non è un loro difetto semplificare, ma è una loro caratteristica. Il problema di queste semplificazioni è che devono essere

indirizzate nella maniera corretta; in altre parole si deve semplificare quello che è giusto semplificare e cioè gli elementi che non sono particolarmente rilevanti per il fenomeno in questione.

Il limite che emerge da molte simulazioni è che poca attenzione viene dedicata a questa distinzione tra aspetti irrilevanti ed aspetti rilevanti; è anche vero che non ci sono regole che indichino quali siano le semplificazioni giuste e quali no, però è anche vero che una maggior attenzione e riflessione sull'argomento, potrebbero risultare utili nella scelta delle semplificazioni.

Facendo un esempio si potrebbe considerare l'uso delle simulazioni per lo studio della mente e del comportamento attraverso l'utilizzo di reti neurali, "che sono delle teorie simulative del comportamento, esplicitamente ispirate alle caratteristiche fisiche del sistema nervoso che controlla il comportamento dell'organismo".[7] Una semplificazione scorretta potrebbe essere l'ignorare l'ambiente in cui l'organismo vive e con cui interagisce, non includendolo nella simulazione. La simulazione consisterebbe quindi in una semplice riproduzione della rete neurale. L'ambiente sarebbe sostituito dal ricercatore stesso che deciderebbe quali stimoli devono giungere al soggetto simulato e poi ne valuterebbe le risposte. La scorrettezza della semplificazione è evidente, poiché gli organismi viventi si caratterizzano proprio per la loro evoluzione dei comportamenti che permettono l'interazione con l'ambiente che li ospita e gli input, che ricevuti dall'ambiente, sono elemento che indirizza l'evoluzione comportamentale. In questo caso quindi molte semplificazioni si possono fare, ma non quella di tralasciare la considerazione dell'ambiente in cui l'organismo vive.

[7] Domenico Parisi (2001), *SIMULAZIONI – La realtà rifatta nel computer*, p. 100, il Mulino

3.2.3. Dettagli non corrispondenti alla realtà

Questo problema potrebbe essere individuato come l'opposto di quello appena sopra. Le simulazioni sono per necessità semplificate, però devono comunque essere in grado di costituire un sistema che funzioni, che giri nel computer, che riproduca il fenomeno reale che si va studiando.

Per ottenere questo però può capitare che siano inseriti nelle simulazioni dettagli inutili, o addirittura arbitrariamente scelti dal ricercatore, cioè non aventi riscontro nella realtà. Il problema a questo punto è legato all'interpretazione che viene data ai risultati, infatti, il ricercatore dovrà capire qual è il contributo che i dettagli arbitrari danno ai risultati della simulazione. È ovvio che potrebbe verificarsi che i dettagli inseriti arbitrariamente dal ricercatore diano un contributo distorsivo all'andamento della simulazione; in tal caso, qualora lo studioso non si rendesse conto del contributo dato da tali dettagli, si annullerebbe la buona riuscita dell'esperimento.

CAPITOLO 4

Il progetto SUM

Prima di addentrarsi più approfonditamente nel progetto SUM, è necessario fare una panoramica su quelli che sono gli strumenti su cui il modello è basato, e cioè fare una breve rassegna sulla programmazione ad oggetti, sul linguaggio di programmazione utilizzato l'*objective C* e sul supporto software offerto da *Swarm*.

4.1. La programmazione ad oggetti

Lo sviluppo dei linguaggi di programmazione ad oggetti si basa su estensioni ed innovazioni di linguaggi già esistenti; tramite quest'evoluzione si possono costruire ambienti di sviluppo sempre più funzionali ed efficienti. I linguaggi di programmazione ad oggetti sono in grado di rendere la stesura dei programmi molto più intuitiva, veloce da sviluppare e soprattutto più flessibile all'inserimento di eventuali modifiche. L'utilizzo di questi linguaggi permette di impostare differentemente la struttura del programma; tale nuova struttura non è tuttavia di così facile comprensione per i non addetti ai lavori, infatti presenta aspetti concettuali che vanno compresi prima di potersi dedicare alla costruzione di un ambiente basato su oggetti.

Gli ambienti che permettono di sviluppare programmi con linguaggi orientati agli oggetti, necessitano dei seguenti elementi:

- Una o più librerie di oggetti che fungono da struttura dell'ambiente.
- Strumenti di sviluppo.
- Linguaggio di programmazione ad oggetti

L'insieme delle librerie, contiene le definizioni degli oggetti più comuni e delle funzioni che si possono utilizzare semplicemente richiamandole in determinati punti del programma, quando sono necessarie. Gli strumenti di sviluppo permettono di strutturare le diverse applicazioni in un pc traducendo da codice di programmazione a linguaggio macchina il programma che si sta creando (compilazione) e di verificare se il lavoro svolto attraverso la programmazione sia corretto, non in termini sintattici, ma in termini di funzionamento (controllo con il *debugger)*. Il linguaggio di programmazione ad oggetti è l'insieme di convenzioni che permettono di comunicare con la macchina e di inserire così le informazioni che struttureranno l'ambiente; i principali linguaggi di questo tipo sono *objective C* e *java*. Per il progetto SUM si è usato il linguaggio *objective C*.

Analizzando i linguaggi di programmazione in generale si deve operare una distinzione tra l'interfaccia e la sua implementazione (funzionamento). L'interfaccia è ciò che si può vedere di un'applicazione, mentre l'implementazione è il funzionamento di questa.

Il principio su cui la programmazione ad oggetti si basa è quello di raggruppare informazioni relative al funzionamento di un'entità presente nel programma, con le funzioni che vengono utilizzate da questa; il tutto prende il nome di *oggetto*. All'interno di un *oggetto* sono definite, o meglio sono richiamate, tutte le *funzioni* necessarie al suo funzionamento e queste sono unite all'interno dell'*oggetto* stesso dalle informazioni (relative al suo funzionamento), che lo caratterizzano. Il risultato di tale modo di costruire il programma è quello di avere più *oggetti* creati uno distintamente dall'altro, con ognuno la sua zona di memoria; tali *oggetti* possono essere messi in relazione l'uno con l'altro attraverso sistemi gerarchici che possono essere molto articolati e quindi permettono di strutturare e ricreare fenomeni reali complessi che altrimenti sarebbe molto difficile gestire. La difficoltà di gestione che, attraverso la programmazione ad oggetti viene superata, è dovuta al fatto che l'*oggetto* viene creato dal programmatore con tutte le informazioni e le funzioni necessarie, però

tale *oggetto*, collegato con altri *oggetti* del programma, viene gestito dall'ambiente stesso e non necessita del controllo del programmatore il quale si deve limitare a costruirne la struttura ed a relazionarlo con le altre parti del programma.

Un programma costruito con linguaggio di programmazione ad oggetti è sostanzialmente composto da un insieme di *oggetti*, tra loro relazionati, ognuno dei quali avente un preciso compito; il sistema di relazione tra i vari oggetti è gestito da *messaggi* che i vari oggetti si scambiano comunicando tra loro; l'attivazione di tali messaggi è operata grazie ai *metodi* che sono l'espressione informatica di un funzionamento o di una funzione. Poiché ogni *oggetto* ha il suo funzionamento, può interpretare differentemente dagli altri ogni *messaggio* che riceve, il tutto sulla base della struttura propria di ogni *oggetto*.

In un linguaggio di programmazione tutti gli oggetti della stessa tipologia appartengono a quella che viene denominata *classe*; la classe rappresenta un prototipo di *oggetto* che può essere generato più volte per creare tanti *oggetti* quanti si vuole. Volendo sfruttare una metafora, si può pensare ad uno stampo, con il quale si possono generare *oggetti* tutti aventi le stesse caratteristiche. Gli *oggetti* facenti capo ad una stessa *classe*, devono possedere le stesse caratteristiche di comportamento ed utilizzare le stesse funzioni; secondo questa logica, è possibile generare *classi* e *sottoclassi* che importino le funzioni ed i comportamenti della *classe* origine e poi si caratterizzino per ulteriori funzioni e comportamenti. Questo è ciò che si verifica, ad esempio in SUM, per ciò che riguarda la *classe BasicSumAgent*, dalla quale tutte le tipologie di agenti operanti nel modello ereditano i comportamenti base. In questo modo è possibile creare una complessa rete di dipendenze tra i vari elementi del programma; rete che permette una gestione più semplice dell'intero sistema, anche se bisogna avere ben presente le relazioni che intercorrono tra i vari elementi. Il fatto di creare *oggetti* che siano in relazione tra loro, e non un unico programma comprendente tutto, permette di rappresentare in modo più semplice e più gestibile tutti i fenomeni di tipo complesso che la realtà offre; questo è un passo avanti verso il

tentativo di comprendere che cosa accade in determinati fenomeni reali e nel tentativo di interpretare l'incidenza di ogni elemento sul risultato finale.

4.1.1. Il linguaggio di programmazione "objective C"

Il linguaggio di programmazione *objective C* deriva dal linguaggio *C* "puro" e per questo la sintassi che lo contraddistingue è molto simile a quella del linguaggio da cui si origina. In *objective C* gli *oggetti* sono gestiti come in tutti gli altri linguaggi di programmazione ad oggetti e quindi rispettano quanto detto sopra. Anche per ciò che riguarda l'interazione tra i vari oggetti si sfrutta il sistema dei messaggi che vengono inviati da un oggetto all'altro permettendone il collegamento; infatti per far sì che un oggetto svolga una determinata azione è sufficiente che un altro oggetto invii un messaggio con indicazione del metodo e delle funzioni da utilizzare. Anche le *classi* rispecchiano quanto detto nel paragrafo sopra, essendo dei modelli per la costruzione di altri *oggetti*; gli *oggetti* che vengono creati grazie ad una *classe* vengono definiti istanze di classe. Usando il linguaggio *objective C*, le *classi* godono della proprietà additiva, e cioè le *classi* che derivano da altre *classi* ne ereditano le funzioni ed i metodi. Le *classi* sono definite in due differenti file che rappresentano aspetti differenti della stessa classe:

- L'interfaccia che dichiara tutti i metodi e le variabili della classe.
- L'implementazione tramite la quale si definiscono metodi e le variabili della classe.

L'interfaccia e l'implementazione sono definiti in due differenti file, anche se i compilatori non lo richiederebbero, questo per convenzione e per ordine; l'interfaccia ha solitamente estensione *.h*, mentre l'implementazione ha estensione *.m*. L'abitudine è quella di inserire un solo oggetto per classe, questo

sempre per una questione di ordine e di chiarezza. Il file con estensione *.h* contiene i nomi di tutte le variabili e di tutti i metodi che si intende richiamare nella classe in questione, mentre nel file *.m* si inseriscono le regole di comportamento dell'oggetto stesso.

4.1.2. Swarm

Il progetto *Swarm* è stato ideato da Chris Langton nel 1994, all'Istituto di Santa Fe nel New Messico; molti furono gli sviluppatori che si operarono per portare avanti il progetto. Inizialmente il progetto *Swarm* fu concepito per sistemi operativi *Unix* in grado di supportare l'ambiente grafico *X Windows*. La prima versione beta di *Swarm* fu rilasciata nel 1995, ma nel Gennaio del 1997 fu rilasciata ufficialmente le versione 1.0 la quale funzionava solamente su sistemi *Solaris* e *Linux*. Nell'Aprile del 1998 fu rilasciata la versione 1.1 e, grazie al pacchetto *Cygnus Win32*, *Swarm* poteva finalmente essere utilizzato anche su sistemi operativi *Windows 95/NT*. Verso la fine del 1999 fu rilasciata la versione 2.0 nella quale si introduceva il supporto per il linguaggio *Java* che fino a quel momento era stato tralasciato. Questa innovazione permise ai programmatori *Java* di poter introdurre all'interno dei propri modelli tutte le librerie *Swarm*.

Swarm è un insieme di librerie informatiche che forniscono un potente strumento per la creazione e la gestione di simulazioni. Il codice *Swarm* è orientato agli oggetti e fornisce un insieme di strumenti atti a favorire la creazione di simulazioni che sfruttano la programmazione ad oggetti. Il primo oggetto ad essere creato è l'*Observer* il quale crea a sua volta l'interfaccia utente ed inizializza le istanze del *ModelSwarm*; il *ModelSwarm* a sua volta crea i livelli sottostanti e la creazione delle azioni e delle attività. L'oggetto *ModelSwarm* è generalmente definito come una sottoclasse dell'oggetto *Swarm* e rappresenta l'oggetto al quale è attribuito il compito di costruire gli agenti. Il *ModelSwarm* è

anche in grado di fornire a ciascun agente un indirizzo di memoria e di programmare le proprie attività.

In *Swarm* esistono due metodi di particolare rilevanza che meritano un approfondimento e sono:

- il metodo buildObjects.
- il metodo buildActions.

Il metodo buildObjects introduce le istruzioni per la creazione sia degli oggetti appartenenti alla classe in questione sia tutte le istruzioni di cui la stessa classe necessita per creare i suoi oggetti.

Il metodo buildActions crea gli oggetti di due classi fondamentali: l'*ActionGroup* e lo *Schedule*. L'*ActionGroup* è l'oggetto che contiene l'insieme di azioni ed eventi che devono essere svolti simultaneamente durante la fase di simulazione. Lo *Schedule* è l'oggetto che contiene la sequenza tramite la quale vengono mandati in esecuzione gli elementi inseriti nell'*ActionGroup*.

Un altro elemento fondamentale per l'ambiente *Swarm* è rappresentato dalle *liste*. In *Swarm* le liste sono una tipologia di oggetto con una propria classe di appartenenza definita *List*. Una lista è un oggetto all'interno del quale è definita una precisa struttura di singoli oggetti dove per ciascuno dei quali è definita la propria zona di memoria e la propria posizione nei confronti degli altri oggetti. Una lista può essere considerata un contenitore dotato di un elenco di tutti gli elementi che vi sono in esso.

4.2. Che cos'è SUM

Sum, acronimo di *Surprising (Un)realistic Market* è un progetto di simulazione di un mercato di borsa. In questo mercato si vuole vedere come si comportino, ricostruendo la struttura di un mercato reale, gli agenti artificiali che

vi operano e quali siano i risultati, da un punto di vista aggregato, dei loro comportamenti.

L'autore,[8] ha evitato di introdurre in questo modello ogni semplificazione relativa alla formazione dei prezzi, come ad esempio l'utilizzo del banditore, ma anzi ha ricreato la struttura del book telematico operante nei mercati reali; in questo modo il prezzo in SUM è generato, istante dopo istante, dal comportamento degli operatori che vi operano, i quali prendono le loro "decisioni", in merito all'operatività sul mercato, sulla base di regole che li caratterizzano, ma assolutamente senza condizionamenti da parte del programma stesso. Così accade che ad esempio l'operatore *random*, scelga casualmente se acquistare o vendere, oppure che l'operatore *stoploss*, decida di acquistare o vendere, soltanto quando il prezzo del titolo abbia raggiunto un determinato valore. Gli operatori, chiamati agenti, inviano i loro ordini al mercato e, come nella realtà, questi vengono inseriti nel book, dal lato denaro o dal lato lettera a seconda che siano rispettivamente ordini di acquisto o di vendita; il book manda in esecuzione immediatamente gli ordini che trovano una controparte mentre mette in lista, in ordine decrescente gli ordini di acquisto ed in ordine crescente gli ordini di vendita rimanendo così coerente con le modalità utilizzate nei mercati reali.

I risultati più interessanti emersi dal funzionamento di questo modello sono relativi al formarsi di bolle speculative e di relativi *crash*, con un discreta facilità; l'autore analizzando la situazione interpreta i dati asserendo che tale fenomeno sia generato dalla struttura del book stesso e non tanto dalla tipologia di operatori che agiscono sul mercato o da eventi esterni (che fino a poco fa non erano presenti in SUM); anche se a tale proposito si è notato che le bolle ed i *crash* si formano più frequentemente e di maggior entità, quando vi sono particolari tipologie di operatori e cioè quelli imitatori di mercato. Ma ciò che

[8] Pietro Terna, Università di Torino, Dipartimento di Scienze Economiche e Finanziarie G. Prato

conta veramente e che a discapito delle tipologie di operatori le bolle ed i *crash* del mercato si generano comunque.

Più tipologie di agenti sono state inserite e più modifiche sono state fatte ma i risultati sono sempre i medesimi.

Le varie tipologie di agenti che operano in SUM, sono state inserite per far sì che nel mercato fossero rappresentate alcune tipologie di operatori che nella realtà agiscono sui mercati di borsa, contribuendo a caratterizzarli (così sembra) e determinandone il comportamento particolare, per il quale non è ancor stata individuata una spiegazione matematica che lo giustifichi e che lo spieghi. Proprio questa mancata comprensione della realtà giustifica l'utilizzo di simulazioni per cercare di comprendere la realtà studiata.

Attualmente, a differenza delle prime versioni del modello in cui vi era un solo titolo, sono presenti più titoli di borsa (scelti a piacere dall'utente) ed il relativo *future*, che ha appunto come sottostante l'indice dei titoli di SUM; la scelta del titolo su cui operare è attualmente impostata in modo casuale, ipotesi questa coerente con la realtà, per la maggior parte degli operatori. A tale regola fanno eccezione alcuni operatori che per il loro modo di operare necessitano di concentrarsi sempre su un titolo oppure sempre su tutti; verrà indicato in questa breve trattazione quali di questi agenti fanno eccezione alla regola generale su definita.

4.2.1. La struttura di SUM

Il modello SUM è strutturato su due livelli, il livello *"observer"*, che fungendo da sonda all'interno del sistema vero e proprio, mostra i risultati di ciò che avviene nel mercato simulato, ed il livello *"model"*, che rappresenta il modello vero e proprio, vale a dire l'insieme degli agenti ed il mercato in se stesso.

In SUM sono presenti molte *classi* la cui definizione e le cui azioni, determinano il funzionamento della simulazione, alcune di queste sono *classi* da cui altre classi ereditano dei comportamenti, altre invece per ora sono *classi* a sé stanti, che svolgono delle operazioni svolte unicamente da esse (sono solitamente quelle degli agenti che operano nel mercato).

Le *classi* da cui altre *classi* ereditano informazioni o *metodi* comportamentali sono:

- *BasicSumAgent*: è la superclasse che definisce la struttura di ogni agente operante in SUM, tutti gli agenti ne ereditano i comportamenti.

- *BasicSumRuleMaster:* è la superclasse che definisce le regole per ogni tipologia di *Rule Master* i quali ereditano da questo tutti i metodi.

- *Book:* è la *classe* che gestisce il funzionamento del mercato simulato, grazie a tale *classe* si regola il sistema delle contrattazioni sia in fase di pre-apertura, sia in fase di apertura. Quindi in tale *classe* vengono definiti tutti i *metodi* che determinano le modalità di inserimento ordini, le modalità di abbinamento degli ordini.

- *CurrentAgent:* è una *classe* di servizio, utilizzata per trasferire il messaggio di agire a tutti gli agenti che popolano il mercato.

- *Matrix, Matrix2, MultiMatrix:* sono *classi* che definiscono i metodi per permettere l'uso di vettori e di matrici, che nel modello vengono gestite.

- *Quota:* in essa viene definita la *probe* che contengono le informazioni relative alle previsioni fatte dagli agenti cognitivi, ed in particolare contengono le percentuali relative alle previsioni che sono risultate corrette.

- *ModelSwarm:* è la *classe* che sta alla base del modello, in essa vengono creati e definiti tutti gli agenti e tutte le altre *classi*; in essa viene inizializzata anche la *classe book*, cuore della simulazione. Viene anche definito in questa sezione la scansione temporale degli eventi, che si verificheranno all'interno della simulazione.

- *ObserverSwarm:* è la *classe* che serve per osservare cosa accade all'interno del modello; permette l'interfaccia grafico con l'utente.

Vi sono quindi le *classi* di tutti gli agenti che operano nel mercato, che sono costruite in modo da caratterizzare l'operato di ciascuno di essi; alcune caratteristiche tipiche di un agente vengono utilizzate anche per costruire altri agenti, ai quali vengono poi ulteriormente date delle caratteristiche aggiuntive; è il caso del *randomAgent*, ad esempio.

Le varie tipologie di agenti operanti in SUM sono:

- *RandomAgent*: è l'agente principale che opera nel mercato simulato ed è uno di quelli che più rappresenta le varie tipologie di agenti che nella realtà operano, o meglio, è quello i cui comportamenti, possono rappresentare l'operato di più tipologie di individui nella realtà borsistica. Le scelte che tale agente attua sono casuali, nel senso che sceglie casualmente (con probabilità pari a 0,5) se acquistare o vendere; per ciò che riguarda il prezzo a cui l'agente Random inserisce l'ordine, questo viene scelto applicando uno

switch (il cui importo è generato in modo casuale), all'ultimo prezzo trattato su quel titolo; quindi le conoscenze che il *randomAgent* possiede, si limitano all'ultimo prezzo eseguito. Il valore dello switch che l'operatore casuale applica al l'ultimo prezzo eseguito, che egli considera come base di partenza per inserire l'ordine, può essere modificato, correggendo i limiti massimo e minimo del range in cui tale switch viene generato casualmente; le variabili da modificare sono il *minCorrectingCoeff* ed il *maxCorrectingCoeff*. E' uno tra le due tipologie di agenti più rappresentative della compagine reale di operatori perché, nella realtà (soprattutto in questi ultimi anni) ci sono molti investitori che operano senza possedere le competenze adatte, senza avere nessuna nozione di base di carattere economico, senza un preciso motivo legato al titolo in questione od alle condizioni macroeconomiche. Il risultato di tali operatori è ben rappresentato dalle operazioni svolte dal *randomAgent* in SUM.

- *ImitatingAgent:* rappresenta l'altra categoria di agenti che meglio rappresentano la popolazione reale di investitori, infatti tale tipologia di agente suddiviso ulteriormente in due tipologie di agenti (*marketImitatingAgent* e *locallyImitatingAgent*), imita l'azione degli agenti che formano il mercato.

  Il *marketImitatingAgent*, decide di acquistare con probabilità definita dalla variabile *asymmetricBuySellProb*, se il prezzo medio dall'istante *t-2* all'istante *t-1* è salito, mentre decide di vendere, sempre con probabilità pari a *asymmetricBuySellProb*, se il prezzo medio dall'istante *t-2* all'istante *t-1* è sceso. Le decisioni in termini di prezzo sono prese moltiplicando un valore casuale per l'ultimo prezzo eseguito (come il *randomAgent*).

Il *locallyImitatingAgent* è invece un agente che imita localmente il mercato prendendo le proprie decisioni di acquisto o vendita, sulla base di quanto la media degli agenti ha fatto; la media delle operazioni degli agenti è data dal *localHistoryLenght*. A seconda di quale delle due operazioni prevalga, quella in vendita o quella in acquisto, l'agente opererà, imitandola (sempre però con probabilità *asymmetricBuySellProb*); il prezzo viene fissato anche i questo caso moltiplicando l'ultimo prezzo per un valore casuale.

Questa tipologia ben spiega quella parte di operatori che segue ogni forma di consiglio o di azione di altri individui, chiunque essi siano. Basti pensare a frasi tipo: "ho comprato, perché un amico ben informato me lo ha consigliato", oppure "ho comprato perché il mio promotore finanziario me lo ha consigliato"; come questi si potrebbero fare altri esempi, criticabili o meno da un punto di vista della razionalità, ma quello che è rilevante ora individuare è il fatto che tale situazioni nella realtà si verificano e quindi vanno rappresentate in un modello di borsa come SUM, che cerca di cogliere l'effetto globale di un insieme di variabili che concorrono alla formazione di un evento, rappresentato dal prezzo dei titoli.

Tale tipologia di agenti, se in percentuale alta rispetto agli altri agenti, per le sue caratteristiche operative, è fortemente destabilizzante per il mercato che in SUM si genera, in quanto imitando l'operato di altri genera svuotamenti asimmetrici delle parti del book causando il rapido formarsi di bolle speculative. Lo svuotarsi delle parti del book rappresenta eccessi di domanda o di offerta sul titolo in questione, che non vengono colmate per un periodo di tempo (variabile, ma consistente) e che alimentano le bolle speculative ed i relativi *crash*. Tale azione, come si

approfondirà più avanti è limitata dall'operato di un agente arbitraggista.

- *StoppLossAgent:* è un agente che opera secondo due diverse modalità: senza memoria del passato, oppure con memoria del passato. Nel caso in cui operi senza memoria del passato, se l'ultimo prezzo eseguito è aumentato (diminuito), rispetto al prezzo medio di un dato intervallo di tempo, di un tasso maggiore del *maxLossRate*, l'agente compra (vende) al prezzo corrente. Nel caso in cui operi con memoria del passato, se l'ultimo prezzo eseguito è aumentato (diminuito), rispetto al prezzo medio di un dato intervallo di tempo, ad un tasso maggiore od uguale al parametro *maxLossRate*, l'agente compra (vende) se ha una posizione *short* (*long*). In entrambi i casi, se il prezzo corrente non supera lo *stop loss*, l'agente si comporterà come un agente casuale. Tale agente non crea effetti che alimentano le anomalie di borsa in quanto opera fondamentalmente come un RandomAgent, con l'aggiunta però di una caratteristica in più e cioè quella di trattare anche i titoli sulla base di uno *stop loss*; tuttavia in casi in cui il prezzo corrente superi lo *stop loss*, l'effetto è quello di alimentare la direzionalità del mercato. Si potrebbe criticare tale impostazione asserendo che un operatore che fa usa dello *stop loss*, non lo manterrebbe fisso per tutto il periodo di mantenimento del titolo, ma lo modificherebbe a seconda dell'andamento delle quotazioni e di altri eventi che ora non è importante ricordare. Tale critica è corretta, se però si pensa che qui si vuole vedere cosa si verifica in un mercato se alcuni agenti utilizzano lo *stop loss* e non cosa capita alla ricchezza personale dell'agente *stop loss*, si comprende che tale critica non ha senso di esistere (almeno per ora) in SUM; quando, se mai accadrà, saranno inserite le ricchezze personali dei vari operatori allora avrà

senso modificare in parte la struttura di tale agente, rendendolo conforme a quello reale.

- *aNNForecastAppAgent:* è una tipologia di agente che opera sulla base di previsioni eseguite da un altro soggetto presente nel mercato, il quale ha solo la funzione di generare previsioni, il *forecastAgent*, senza preoccuparsi di doverle utilizzare per operare (come invece fa il *BPCTagent*). Il *forecastAgent*, genera previsioni su tutti i titoli presenti sul mercato, compreso il future sull'indice, sarà poi il *aNNForecastAppAgent* che sceglierà in modo casuale quale previsione seguire e di conseguenza su quale titolo operare. Le previsioni sono generate sulla base di calcoli prodotti da una rete neurale artificiale. Le caratteristiche della rete neurale sono: tot nodi input dati dal valore di *dataWindowLenght* (rappresentante variazioni dei prezzi medi giornalieri di ogni giorno precedente al valore *nAheadForecast*), tot nodi hidden pari a *dataWindowLenght/2*, un unico output dato dalla previsione pari a *nAheadForecasting* giorni avanti. Tali previsioni sono quindi utilizzate dall'*aNNForecastAppAgent*, nel seguente modo: ogni giorno l'operatività di tali agenti è definita dalla probabilità data dal valore indicato nella variabile, *aNNforecastAppAgentActDailyProb* ed acquistano o vendono a seconda che il valore della previsione sia maggiore di *[1 + aNNInactivityRange]* o minore di *[1-aNNInactivityRange]*; se la previsione è maggiore di *[1 + aNInactivityRange]* l'acquisto avverrà con probabilità pari a *asimmetricBuySellProb*, nel caso in cui la previsioni sia minore di *[1 - aNInactivityRange]* gli operatori venderanno con la solita probabilità. Nel caso in cui il valore previsto sia compreso nel *range*, gli agenti non agiranno.

- *BPCTAgentA:* sono agenti "cognitivi", costruiti sulla base di un complesso meccanismo di determinazione delle loro azioni, che si avvale della metodologia dei *cross target*. Gli agenti, sulla base della struttura della rete neurale sviluppano coerenza interna tra due azioni, quella di acquisto e quella di vendita e sviluppano congetture sugli effetti liquidità e quantità azioni. Sono fissati anche degli obiettivi esterni di tre tipi differenti: accrescere la liquidità, accrescere la quantità di azioni detenute, accrescere entrambi.[9]

- *BPCTAgentB:* sono anch'essi strutturati facendo utilizzo di reti neurali che presentano la tecnica dei *cross target*; sono un'evoluzione degli agenti cognitivi di tipo A, visti al punto sopra. A differenza degli agenti di tipo A, sviluppano congetture oltre che sugli effetti liquidità e quantità di azioni, anche sugli effetti ricchezza dall'agente valutata al prezzo di chiusura e ricchezza dell'agente valutata sulla base del prezzo previsto dall'agente neurale. L'output dal lato delle azioni è sempre uno e determina la decisione di acquisto o di vendita.

  L'obiettivo sia dell'agente di tipo A e sia di tipo B è quello di sviluppare una coerenza interna tra l'azione di vendita e l'azione di acquisto. Tale tipologia di agente (cognitivo), attraverso la metodologia dei *cross target*, sviluppa con l'apprendimento la capacità di prendere in modo coerente delle decisioni, cioè è in grado di prendere delle decisioni sulle azioni da compiere, per ottenere determinati risultati.

[9] Ovviamente non è possibile ottenere contemporaneamente l'accrescimento di liquidità e di azioni possedute; tale possibilità viene comunque inserita per completezza dello schema della rete neurale stessa.

- *ArbitrageurAgent:* è un agente che cerca di ricavare profitti operando su due differenti mercati di titoli azionari, quello a pronti e quello a termine. Tale tipologia di agente individua disallineamenti tra future sull'indice e suo sottostante (i titoli) ed opera mettendo ordini su entrambi i mercati, di segno opposto. Per una trattazione più approfondita si rimanda al Cap. 9 par. 2 della presente tesi.

- *AvatarAgent:*[10] è una classe di agenti, che è impersonata da umani, nel senso che un umano può interagire con il modello SUM, attraverso tali agenti. Le operazioni sono decise liberamente dall'umano, ma affinché queste possano essere inserite nel modello, è necessario che l'umano abbia personificazione in SUM in un agente *avatar*, che viene gestito dal modello come un qualsiasi altro agente e le cui caratteristiche operative non hanno precisa definizione nel codice informatico, perché sono relative a ciò che l'umano in questione desidera fare.

La scansione temporale all'interno di SUM, prevede le seguenti azioni:

- *Clean:* il book di borsa viene "pulito", in questo modo le proposte che il giorno precedente sono state inserite, vengono eliminate.

- *Pre-apertura:* gli agenti valutano se inserire o no ordini di acquisto o di vendita; tale fase serve a far sì che il book non sia vuoto in fase di apertura, gli ordini che vengono inseriti in tale fase, sono poi passati alla fase di contrattazione continua, dove se abbinabili vengono conclusi.

[10] Il termine avatar, indica la personificazione di una divinità in un umano; in questo caso indica la personificazione di un umano nel modello SUM.

- *Negoziazione continua:* gli agenti decidono se acquistare o vendere, inseriscono nel book i loro ordini, che se trovano contropartita vengono chiusi, altrimenti sono messi in lista secondo l'ordine di prezzo previsto dal lato del book e secondo la relativa priorità temporale, per venire poi eseguiti successivamente nell'arco della giornata (qualora trovino contropartita).

- *Contabilità:* al termine della giornata di borsa si procede con il calcolo del prezzo medio relativo ai titoli ed alla ricchezza degli agenti calcolata sia sulla base della liquidità, sia sulla base del controvalore dei titoli.

4.2.2. L'interfaccia grafico di SUM

Il modello SUM presenta un interfaccia grafico che permette all'utente una migliore interazione con il modello, garantendo una maggior facilità di utilizzo, rispetto al dover cambiare i valori delle variabili agendo direttamente sul codice di programmazione; l'interfaccia grafico consente l'utilizzo di SUM anche a chi non è in grado di modificare il codice, ma vuole semplicemente fare esperimenti per osservare cosa capita nella realtà simulata di SUM. Tale interfaccia è gestito dalla classe *"observer"* e presenta tre finestre di dialogo, di cui una è una semplice tastiera che permette di immettere i comandi utili al funzionamento del modello, mentre le altre due permettono di modificare alcune impostazioni al fine di mutare alcune condizioni del mercato.

In questo paragrafo verrà fatta una presentazione completa di tutti i parametri che l'utente può liberamente modificare, in modo da permettere al lettore di interagire con il programma senza conoscerne il codice ed il linguaggio di programmazione.

La finestra tastiera, denominata "*ProCtrl*" si presenta nel seguente modo e non necessita di nessun commento:

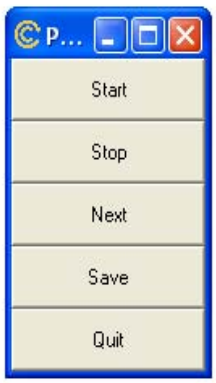

Figura 1

Attraverso questa tastiera è possibile far iniziare la simulazione (tasto start), bloccarla (tasto stop) in qualsiasi momento, uscire dal programma (tasto quit), salvare i dati (tasto save).

La finestra di dialogo dell'*obsever* è la seguente:

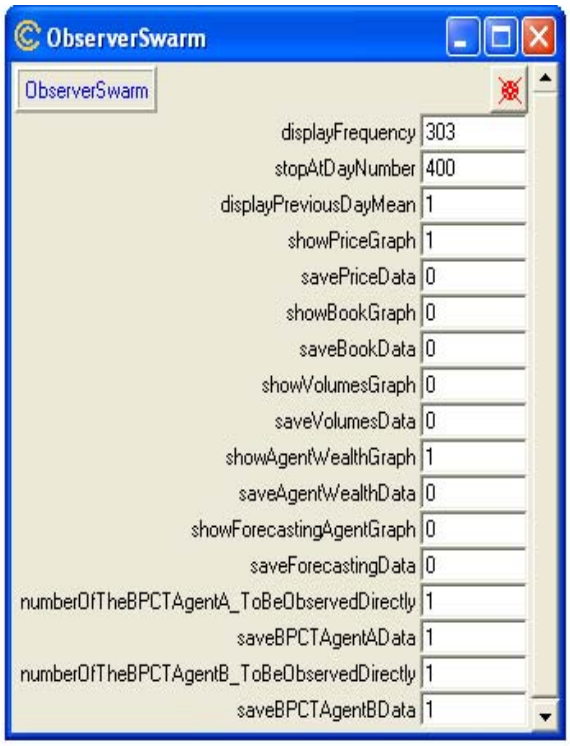

Figura 2

I parametri che l'utente può modificare a suo piacimento, per fare esperimenti di vario tipo, modificando sostanzialmente le condizioni di mercato, sono i seguenti:

- *DisplayFrequency:* indica la frequenza con la quale vengono aggiornate le finestre che riportano le varie tipologie di grafici possibili; tale frequenza può essere inserita in valore pari ad uno oppure in valore pari ad un multiplo del numero degli agenti. Seguendo quest'ultima possibilità, si ha il risultato di ottenere sul

grafico il valore relativo all'ultimo prezzo di ogni giorno della simulazione, per tale motivo è consigliabile tale scelta.

- *StopAtEpochNumber:* indica il numero di giorni che la simulazione produrra, al raggiungimento del quale si fermerà.

Le variabili che seguono hanno la valenza di interruttori, nel senso che, inserendo il valore "1", si attiva la funzione, mentre inserendo il valore "0", si disattiva la funzione. Tali variabili switch, sono:

- *DisplayPreviousDayMean:* permette di visualizzare il grafico del prezzo medio relativo all'andamento di ogni titolo.

- *SavePriceData:* permette di salvare i dati relativi alla simulazione in un apposito file.

- *ShowBookGraph:* permette di visualizzare l'andamento del volume di contrattazione su ogni titolo.

- *SaveBookData:* permette di salvare i dati relativi al *bookGraph*, in un apposito file.

- *ShowVolumesGraph:* permette di visualizzare l'andamento dei volumi di contrattazione.

- *SaveVolumesData:* permette di salvare i dati relativi al *volumesGraph*, in un apposito file.

- *ShowAgentWealthgraph:* permette di visualizzare il grafico della ricchezza degli agenti.

- *SaveAgentWealthGraph:* permette di visualizzare il grafico della ricchezza degli agenti.

- *ShowForecastAgentGraph:* permette di visualizzare il grafico con le previsioni dell'agente neurale.

- *SaveForecastData:* permette di salvare i dati relativi alle previsioni dell'agente neurale.

- *NumberOfTheBPCTAgentA_ToBeObserverDirectly:* permette di decidere il numero di *BPCTAgentA* di cui si vuole visualizzare il grafico con, liquidità, quantità di azioni e ricchezza.

- *SaveTheBPCTAgentAData:* permette di salvare i dati relativi al grafico dei *BPCTAgentA*, in un apposito file.

- *NumberOfTheBPCTAgentA_ToBeObserverDirectly:* permette di decidere il numero di *BPCTAgentA* di cui si vuole visualizzare il grafico con, liquidità, quantità di azioni e ricchezza.

- *SaveTheBPCTAgentBData:* permette di salvare i dati relativi al grafico dei *BPCTAgentB*, in un apposito file.

La finestra di dialogo (sempre gestita dall'observer), che permette di modificare alcune delle variabili del *Model* è la seguente:

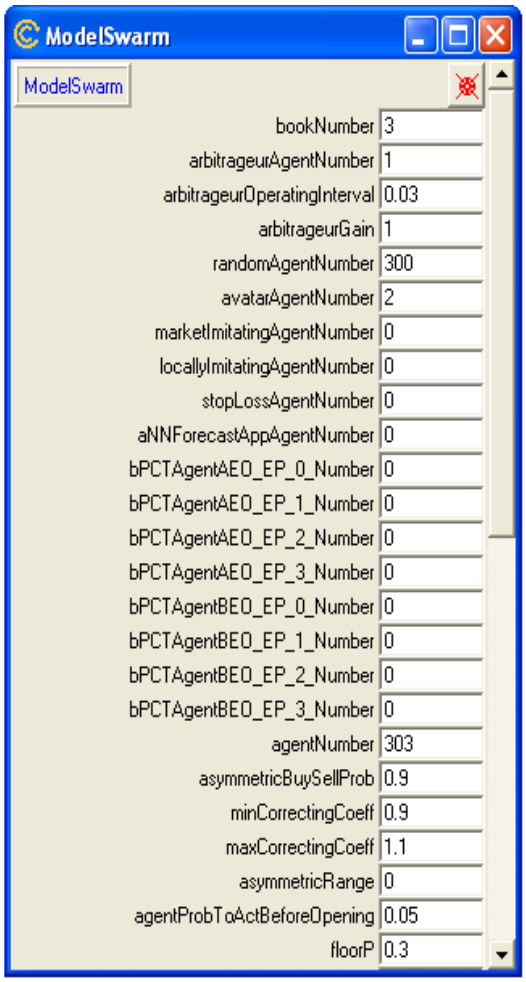

Figura 3

- *BookNumber:* permette di inserire il numero di book che si vuole inserire nelle contrattazioni del mercato.

- *ArbitrageurAgentNumber:* permette di inserire il numero di agenti arbitraggisti.

- *ArbitrageurOperatingInterval:* permette di decidere l'ammontare in termini percentuali del livello dei costi delle operazioni da arbitraggio; oltre questo scostamento minimo tra valore del future e valore teorico del future, l'arbitraggista riterrà proficuo svolgere le sue operazioni.

- *ArbitrageurOperatingIntervalFixed:* permette di attivare, inserendo il valore "1", la modalità a costi fissi per le operazioni sul *future*, eseguite dagli operatori arbitraggisti; inserendo il valore "0" la modalità dei costi è quella variabile e segue i parametri sopra esposti.

- *ArbitrageurGain:* è un parametro switch che, se attivato (inserendo il valore 1), permette di far sì che l'arbitraggista operi considerando di guadagnare sempre da ogni operazione. Disattivando tale funzione l'arbitraggista opererà qualsiasi volta individui uno scostamento del *future* dal suo valore teorico, indipendentemente dal fatto che ottenga un profitto.

- *RandomAgentNumber:* permette di scegliere il numero di agenti random da far operare in SUM.

- *AvatarAgentNumber:* permette di inserire il numero di agenti gestiti da umani che si vuole far operare in SUM.

- *MarketImitatingAgentNumber:* permette di scegliere il numero di agenti imitatori di mercato da far operare in SUM.

- *LocallyImitatingAgentNumber:* permette di scegliere il numero di agenti imitatori locali di mercato da far operare in SUM.

- *StopLossAgentNumber:* permette di scegliere il numero di agenti che utilizzano lo *stop loss* da far operare in SUM.

- *ANNForecastAppAgentNumber:* permette di scegliere il numero di agenti che operano sulla base delle previsioni fornite dal *aNNForecastAgent*, da far operare in SUM.

- *BPCTAgentAEO_EP_0_Number:* permette di inserire il numero di *BPCTAgentA* che non utilizzano obiettivi esterni (EO).

- *BPCTAgentAEO_EP_1_Number:* permette di inserire il numero di *BPCTAgentA* che hanno come obiettivo esterno l'accrescimento della liquidità.

- *BPCTAgentAEO_EP_2_Number:* permette di inserire il numero di *BPCTAgentA* che hanno come obiettivo esterno l'accrescimento del numero di azioni in portafoglio.

- *BPCTAgentAEO_EP_3_Number:* permette di inserire il numero di *BPCTAgentA* che hanno come obiettivo esterno sia l'accrescimento della liquidità, sia l'accrescimento del numero di azioni in portafoglio.

- *BPCTAgentBEO_EP_0_Number:* permette di inserire il numero di *BPCTAgentB* che non utilizzano obiettivi esterni (EO).

- *BPCTAgentBEO_EP_1_Number:* permette di inserire il numero di *BPCTAgentB* che hanno come obiettivo esterno quello di accrescere la ricchezza valutata al prezzo di chiusura.

- *BPCTAgentBEO_EP_2_Number:* permette di inserire il numero di *BPCTAgentB* che hanno come obiettivo esterno quello di accrescere la ricchezza valutata al prezzo previsto.

- *BPCTAgentBEO_EP_3_Number:* permette di inserire il numero di *BPCTAgentB* che hanno come obiettivo esterno sia l'accrescimento della ricchezza valutata al prezzo di chiusura e sia al prezzo previsto.

- *AgentNumber:* numero totale degli agenti operanti nel mercato.

- *AsymmetricBuySellProb:* con tale valore si indica la probabilità di operare, per gli agenti che utilizzano strategie imitative, in accordo con le strategie stesse.

- *MinCorrectingCoeff:* indica il valore minimo del coefficiente correttivo utilizzato per determinare il prezzo da inserire negli ordini di compravendita.

- *MaxCorrectingCoeff:* indica il valore massimo del coefficiente correttivo utilizzato per determinare il prezzo da inserire negli ordini di compravendita.

- *AsymmetricRange:* correzione aggiuntiva a quella descritta sopra che permette di far sì che l'agente adotti un comportamento asimmetrico.

- *AgentProbToActBeforeOpening:* permette di determinare la probabilità dell'agente di agire prima dell'apertura.

- *FloorP:* il valore *floor* del prezzo.

Attraverso la barra di scorrimento è possibile, visualizzare gli altri parametri che l'utente può modificare (figura 4):

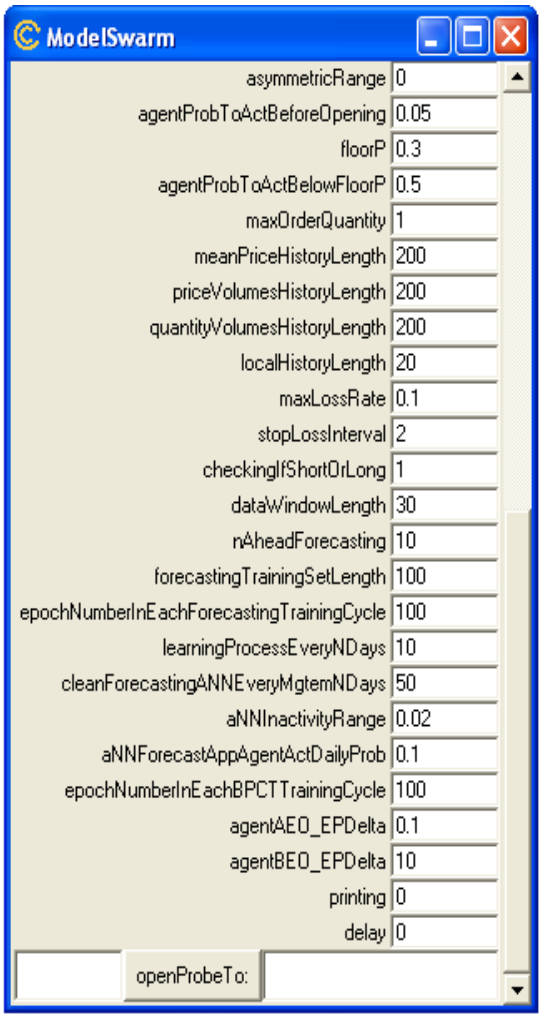

Figura 4

I parametri corrispondenti a questa parte della finestra di dialogo sono i seguenti:

- *AgentProbToActBelowFloorP:* indica il valore della probabilità di agire quando il prezzo raggiunge il valore *floor*.

- *MaxOrderQuantity:* indica il valore massimo del quantitativo che ogni agente può inserire negli ordini di compravendita.[11]

- *MeanPriceHistoryLenght:* indica la lunghezza del vettore che contiene la serie storica di prezzi medi dei titoli, utilizzata dagli agenti imitativi del mercato.

- *PriceVolumeHistoryLenght:* indica la lunghezza del vettore che contiene la serie storica dei volumi relativi a determinati prezzi, utilizzata dagli agenti imitativi del mercato.

- *QuantityVolumesHistorylenght:* indica la lunghezza del vettore che contiene la serie storica dei volumi dei titoli, utilizzata dagli agenti imitativi di mercato.

- *LocalHistoryLenght:* lunghezza del vettore che registra le azioni degli altri agenti, tale valore è utilizzato dagli agenti che imitano localmente.

[11] Gli agenti in SUM, possono contrattare una sola azione alla volta; quando si varia il parametro "*maxOrderQuantity"*, si indica che un agente possa trattare un quantitativo di azioni superiore ad "1", ma tecnicamente, avviene che lo stesso agente opererà per "n" volte consecutive invece che immettere un quantitativo pari ad "n" azioni in un unico ordine.

- *MaxLostRate:* valore massimo di perdita che l'agente *stop loss* è disposto a sopportare come tasso di perdita; oltre tale valore attiverà la strategia dello *stop loss*.

- *StopLossInterval:* indica la lunghezza dell'arco di tempo preso in considerazione dall'agente *stop loss*.

- *CheckingIfShortOrLong:* è un parametro *switch*, se attivato (inserendo il valore "1") fa sì che l'agente *stop loss* operi avendo memoria del passato; se disattivato (inserendo il valore "0") fa sì che l'agente *stop loss* operi senza memoria delle sue posizione passate.

- *DataWindowLenght:* il numero di dati utilizzati come input dalle rete neurale artificiale; i dati sono espressi come indici calcolati rapportando il prezzo medio del giorno *t* con il prezzo medio del giorno *(t – nAheadForecasting)*.

- *NAheadForecast:* numero di giorni i cui valori vengono previsti dalla rete neurale; l'output generato dalla rete è sotto forma di rapporto tra il prezzo al tempo *nAheadForecast* e l'ultimo prezzo medio disponibile.

- *ForecastTrainingSetLenght:* numero di dati (input ed output attesi) utilizzati per fare apprendimento sulla rete neurale artificiale; tale set di valori viene aggiornato con il passare del tempo.

- *EpochNumberInEachForecastTrainingCycle*: indica il numero di epoche che vengono utilizzate nella fase di apprendimento della

rete neurale artificiale; per ogni epoca si considera un set di valori pari al valore del parametro sopra indicato.

- *LearningProcessEveryNDays:* indica il numero di giorni che intercorrono tra una fase di apprendimento e l'altra, quindi tale valore indica ogni quando la fase di apprendimento, definita sulla base delle variabili indicate sopra, deve essere ripetuta.

- *CleanForecastANNEveryMgtemNDays:* indica ogni quanti giorni la fase di apprendimento debba ricominciare da capo, assegnando nuovi pesi alle matrici delle funzioni utilizzate dalla rete neurale; in tale modo si ha la possibilità di tener conto nella fase di apprendimento dei cambiamenti della struttura dei prezzi.

- *ANNInactivityRange:* indica il range di variazione del prezzo all'interno del quale l'agente non opera; se la previsione è compresa tra [1 - aNNInactivityRange] e [1 + aNNInactivityRange], l'agente non opera.

- *ANNForecastAppAgentActDailyProb:* indica la probabilità di agire sul mercato.

- *EpochNumberInEachBPCTTrainingCycle:* indica il numero di epoche di apprendimento della rete neurale artificiale; l'apprendimento è applicato ogni giorno sugli ultimi 10 giorni, insieme che viene aggiornato di giorno in giorno.

- *AgentAEO_EPDelta:* indica il valore del parametro che viene utilizzato per misurare l'effetto degli obiettivi esterni della rete neurale artificiale degli agenti BPCT di tipo A.

- *AgentBEO_EPDelta:* indica il valore del parametro che viene utilizzato per misurare l'effetto degli obiettivi esterni della rete neurale artificiale degli agenti BPCT di tipo B.

- *Printing:* inserendo vari valori è possibile attivare la stampa su monitor di una vasta serie di informazioni relative ad alcuni elementi della simulazione di borsa.

- *Delay:* indica il ritardo in termini di secondi che si può attivare per le operazioni sul mercato degli *randomAgent*; tale ritardo serve per permettere agli *avatarAgent* di avere il tempo di inserire gli ordini senza che la simulazione sia troppo veloce.

4.2.3. Qual è lo scopo che SUM si prefigge

Il modello si sta sempre più evolvendo grazie al lavoro congiunto degli studenti e del professor Terna, con lo scopo di rendere sempre più simile alla realtà la simulazione; tuttavia non va persa di vista una logica di fondo che alimenta il progetto SUM, e cioè il fatto che la tendenza debba essere quella di replicare la realtà nei suoi aspetti essenziali e fondamentali, per poi osservare come tali regole di fondo determinino il verificarsi ed il susseguirsi degli eventi simulati. Tale atteggiamento serve a far sì che SUM e le simulazioni in generale possano offrire un modo alternativo allo studio della realtà, nella fattispecie dell'economia. Quindi non si deve pensare che lo scopo sia quello di copiare la realtà in tutti i suoi elementi, perché ciò porterebbe inevitabilmente ad un fallimento del progetto in quanto questo genererebbe dei risultati identici a quelli della realtà semplicemente per il fatto che si è forzato il verificarsi degli eventi all'interno del mondo simulato. La simulazione deve vivere su determinate

regole fondamentali, ma muoversi secondo processi liberi. I risultati ottenuti da una simulazione forzata non permetterebbero a questa di dare delle nuove informazioni allo studioso e nemmeno gli permetterebbero di comprendere qualcosa in più di quello che già si sapeva prima di svolgere la sperimentazione.

Il susseguirsi delle modifiche sta dando risultati interessanti, a volte non previsti anche se prevedibili.

4.3. Possibili evoluzioni future

Il modello SUM pur essendo già corposo, presenta ancora delle lacune per poter essere definito rappresentativo della realtà, anche se coglie nell'essenza il meccanismo delle contrattazioni di borsa e quindi ne riproduce i comportamenti che evidenziano i problemi che si possono generare sulla base delle caratteristiche della sua stessa struttura. Basti pensare al risultato più preoccupante, ma anche più stimolante, che emerge in tutte le sperimentazioni fatte con il modello SUM e cioè, il formarsi di bolle "speculative" e dei conseguenti *crash* borsistici. Questo risultato si crea nella struttura più semplice di SUM ma anche nelle sue versioni più evolute che sono per ora quella che comprende la trattazione sul mercato di più titoli e del future sul loro indice e quella che introduce agenti che sono sensibili ad eventi esogeni, gli *eventAgent*. Si capisce bene che visti i risultati delle sperimentazioni, la causa delle bolle speculative non è (solamente) la presenza di fattori esogeni, ma è insita nel sistema in sé e nel comportamento degli operatori; anzi si può dire che le bolle speculative si formino comunque e che piuttosto siano ulteriormente alimentate dal comportamento degli agenti sensibili ad eventi esterni.

Le modifiche da fare sono ancora molte e lasciano spazio in varie direzioni, dalla compartecipazione di agenti umani e di agenti artificiali alla

ricerca di realismo nella struttura del book, alla introduzione di sempre più sofisticate modifiche volte a rendere il modello sempre più ricco di tipologie di strumenti finanziarie e di elementi caratterizzanti il mondo reale. Il tutto senza perdere di vista l'obiettivo, che non è quello di non far sì che gli eventi si verifichino perché il programmatore imposta il modello in modo tale da instradarlo alla formazione di determinate situazioni, bensì quello di ricreare (con le opportune semplificazioni) le condizioni della realtà e poi di stare ad osservare cosa nel modello accade.

4.3.1.L'ampliamento del contratto future

Per rimanere vicino alla trattazione di questa tesi, un naturale ampliamento di questo lavoro potrebbe essere l'eliminazione di alcune semplificazioni che pur essendo state opportunamente giustificate, non rendono il sistema completamente aderente nelle sue parti alla realtà.

Per comprendere il motivo di questo suggerimento di modifica, è necessario richiamare quello che doveva essere un percorso di sperimentazione ipotizzato su indicazione del professor Terna. Si era deciso che dopo aver introdotto il *future* sull'indice di borsa sarebbe stato necessario, qualora le operazioni dell'arbitraggista non fossero state sufficienti a mettere in equilibrio il mercato a pronti ed il mercato a termine, introdurre una forma di cognitività a tutti gli altri agenti, relativamente a ciò che il *future* sull'indice rappresentasse. Tale ipotesi è stata suffragata da colloqui con operatori del settore i quali indicavano, come causa principale del mantenimento di coerenza tra valore del *future* e suo valore teorico il fatto che gli operatori stessi nel loro complesso abbiano cognizione di ciò che il future rappresenta; questo significa che vi è l'opinione, diffusa tra gli operatori, che non siano le operazioni di arbitraggio a generare condizioni di equilibrio di mercato, ma che sia l'insieme degli operatori,

che avendo conoscenza di ciò che un *future* rappresenta, mantiene le sue quotazioni ad un livello tale che esso sia in linea con il suo valore teorico.

Sarebbe quindi interessante introdurre un aspetto cognitivo tra gli altri agenti, relativamente a ciò che rappresenta il *future* sull'indice, per vedere in sede sperimentale cosa si verifichi nel mercato simulato. Sarebbe ipotizzabile uno scenario in cui, essendo gli operatori istruiti in merito a cos'è un *future*, l'allineamento sarebbe garantito in parte anche dalle loro azioni oltre che quelle attuali dell'arbitraggista; sarebbe inoltre ipotizzabile pensare ad una riduzione dell'attività dell'arbitraggista in quanto questo si troverebbe minori condizioni in cui operare, rispetto a quelle che si presentano ora.

Altra possibile evoluzione, ma ben più lontana in quanto di più difficile applicazione è l'introduzione della ricchezza personale degli operatori; tale aspetto li vincolerebbe da un punto di vista operativo in quanto avrebbero a disposizione una liquidità limitata.

Ovviamente qualora si introducesse in SUM la ricchezza personale degli operatori, fin dal primo giorno di contrattazioni, ciò li vincolerebbe da un punto di vista degli acquisti/vendite. In tale condizione, ritornando al contratto *future*, sarebbe necessario introdurre l'ente garante del buon andamento dei contratti, la Cassa di Compensazione e Garanzia. Questo per evitare che a scadenza del contratto vi sia la possibilità che una delle controparti risulti insolvente. Sempre in questo caso sarebbe interessante vedere quali problemi avrebbe l'operatore arbitraggista nell'eseguire le sue operazioni, soprattutto in relazioni alle situazioni in cui non possedendo sufficiente denaro o i titoli da vendere, debba chiedere denaro a prestito o vendere titoli allo scoperto. Tali problemi nell'eseguire le operazioni di arbitraggio genererebbero dei probabili disallineamenti tra *future* e suo valore teorico, proprio come accade nella realtà per tali motivi. Tuttavia per poter inserire tali modifiche andrebbe implementata anche la variabile tasso d'interesse che permetterebbe di corrispondere un dato compenso per il denaro dato a prestito e nello stesso tempo permetterebbe agli

operatori di mercato di avere un ulteriore mezzo per far fruttare il loro denaro, quello del deposito bancario.

Come si vede da questa breve trattazione le modifiche che si potrebbero inserire sono molte, tuttavia quello che va ancora ricordato è che ciò che si prefigge il progetto non è quello di emulare la realtà in tutto, ma quello di coglierne gli aspetti essenziali per osservare cosa accade quale sia il funzionamento del mercato di borsa; quindi magari a ben veder alcune delle modifiche qui indicate potrebbero non rivelarsi realmente utili nell'incrementare lo spessore sperimentale del progetto SUM, ma potrebbero semplicemente contribuire a renderlo più complesso senza tuttavia aggiungere risultati interessanti.

4.3.2. Le modifiche imminenti

Rispetto alle modifiche citate nel paragrafo precedente, vi sono altre possibili ampliamenti del progetto SUM, che sono in via di sviluppo o che presto saranno sviluppati; tali modifiche sono, primo, l'ampliamento della possibilità di interazione degli umani con il mercato simulato, promovendo una sessione di sperimentazioni che coinvolgano un buon numero di agenti umani e secondo, l'inserimento delle recenti modifiche attuate al book telematico.

La partecipazione di un buon numero di agenti umani al mercato di borsa, potrebbe dare risultati inattesi; infatti, sarebbe interessante verificare cosa le componenti cognitive di agenti umani, potrebbero causare in un mercato di borsa simulato, popolato da agenti che seguono sempre e schematicamente delle regole precise. Si potrebbe, infatti, ipotizzare che gli agenti umani non siano così schematici e razionali nell'applicare le loro regole. Inoltre si potrebbe verificare se le dinamiche di prezzo riscontrate con soli agenti informatici si ripresentino nelle stesse condizioni quando ad operare sono anche gli umani.

L’introdurre maggior realtà nel book è una modifica che, riguardando il cuore del modello, permetterebbe di incrementarne di molto la valenza sperimentale; andrebbero introdotte le aste di chiusura e magari tutte le limitazioni che la Consob impone su quelle variazioni di prezzo che troppo si discostano dalle quotazioni del giorno precedente. Tale aspetto potrebbe avere riscontro sulla formazione delle bolle speculative e sui *crash* che ipoteticamente si potrebbero ridurre.

CAPITOLO 5

Gli strumenti derivati

Gli strumenti derivati sono nati per permettere di gestire più agevolmente il rischio, necessità questa, sorta in seguito all'elevata variabilità dei mercati finanziari tipica degli ultimi due decenni. Si definiscono derivati in ragione del fatto che il loro valore "deriva" dal prezzo dell'attività sottostante al contratto.

Tali strumenti possono essere catalogati in base alla natura del sottostante:

- *Commodity derivative*: sono contratti che hanno per oggetto attività reali, quali ad es. oro, petrolio, caffè, cereali.

- *Financial derivative*: sono contratti basati su attività finanziarie, che si distinguono a loro volta in strumenti derivati aventi come sottostante tassi d'interesse, valute, azioni, indici azionari.

Possono essere altrimenti distinti in riferimento alle caratteristiche tecniche dei diversi strumenti. In sintesi, si possono individuare un limitato numero di operazioni di base, negoziazione a pronti, negoziazione a termine, opzioni, le cui combinazioni originano la totalità dei prodotti derivati:

- Il contratto di *swap*, con il quale le controparti assumono l'impegno a scambiarsi flussi monetari in entrata ed in uscita e a compiere, ad una certa data posteriore, l'operazione inversa;

- Il contratto *future*, che implica un impegno inderogabile, per l'acquirente e per il venditore del contratto, ad effettuare una data prestazione ad una data scadenza; tale impegno può essere estinto mediante un'operazione di segno contrario prima della scadenza.Il Fib30, contratto sull'indice Mib30, è un future.

- Il contratto di opzione, che conferisce all'acquirente, dietro pagamento di un premio la facoltà di ritirare o consegnare una determinata attività ad un certo prezzo e ad una data scadenza futura o entro la stessa.

In ognuna delle categorie sopra citate sono individuabili più varianti rispetto alle figure contrattuali originarie. Inoltre, in seguito al continuo proliferare di nuovi strumenti "ibridi", i quali amalgamano in varia misura le caratteristiche delle tipologie sopra citate per incontrare le esigenze degli operatori più sofisticati, la suddivisione appena fatta assume connotati sempre più sfumati.

Ulteriore distinzione si può fare in merito ai mercati nei quali vengono negoziati gli strumenti. Si possono individuare, da questo punto di vista, due grandi categorie:

- I mercati organizzati

  Presentano caratteristiche omogenee, quali:

  la standardizzazione dei contratti, che ne aumenta la fungibilità, la regolamentazione delle transazioni, il rispetto di rigide norme a salvaguardia dell'integrità del mercato, la presenza di uno specifico organismo chiamato "*clearing house*". Tale organismo detto anche "Cassa di Compensazione e Garanzia", fungendo da controparte legale di tutti gli acquirenti e di tutti i

venditori, facilita l'esecuzione dei contratti e soprattutto ne garantisce il buon fine.

La standardizzazione e la presenza della Cassa di Compensazione e Garanzia permettono di chiudere anticipatamente una posizione con estrema facilità, garantendo una significativa flessibilità nell'uso dei contratti negoziati nei mercati regolamentati.

In questi mercati si negoziano future ed opzioni.

- I mercati fuori borsa

  Vengono anche definiti "*over-the-counter*" (*OTC*); sono caratterizzati dall'assenza di luogo fisico per lo svolgimento delle negoziazioni, dalla mancanza di quotazioni ufficiali, dalla quasi mancanza di standardizzazione dei contratti, sia per ciò che riguarda gli importi, sia per le scadenze, dalla liberalità dei contratti, dall'assenza di organismi centrali di compensazione.

  I contratti scambiati su tali mercati vengono privatamente gestiti tra le istituzioni finanziarie ed i loro clienti, costituendo per questi ultimi strumenti personalizzati per la gestione del rischio; infatti i contratti OTC hanno per oggetto attività e/o scadenze non disponibili sui mercati regolamentati.

  Swap ed opzioni vengono contrattati su questi mercati.

5.1. Il future sull'indice di borsa

Il future sull'indice di borsa è un contratto con il quale due soggetti si accordano per acquistare o vendere un paniere di titoli azionari inclusi nell'indice oggetto del contratto, ad un prezzo prefissato ed ad una data stabilita.

Il mercato azionario italiano, ha avviato le negoziazioni del future sull'indice di borsa dal 28 novembre 1994; è così nato *l'Italian DErivatives Market (Idem)*, il mercato nazionale per i prodotti derivati. Il mercato italiano si

discosta dalla prassi internazionale in quanto gli stessi organi di controllo, che gestiscono il mercato principale, si occupano di quello dei derivati contro la soluzione internazionale che pone i mercati degli strumenti derivati in un ambito regolamentare parallelo a quello della borsa valori, con organi di controllo distinti.

A differenza di altri contratti *future* (come l'*interest rate future* e il *currency future*), quello su indici di borsa non presenta un'attività sottostante materialmente negoziabile. Questo comporta che il *future* sull'indice abbia le seguenti caratteristiche:

- Per convenzione il valore dell'attività sottostante è pari al valore dell'indice moltiplicato per un valore monetario. Nel caso del Fib30 si ha che ciascun punto dell'indice vale 5 € (10.000 £).

- Non è possibile effettuare la consegna fisica del sottostante, infatti, sarebbe poco pratico e molto costoso provvedere alla consegna materiale di un rilevante numero di titoli nella stessa porzione con la quale essi risultano essere presenti nell'indice di riferimento. Il regolamento avviene tramite denaro (*cash delivery*) per le posizioni ancora aperte.

Gli scopi perseguibili dall'utilizzatore degli *stock index future* sono configurabili diversamente in base all'atteggiamento assunto nei confronti del rischio e distinguibili in: speculativo, di copertura, di arbitraggio.

- Lo speculatore apre una posizione deliberatamente esposta al rischio; acquista uno *stock index future* auspicando un rialzo dei titoli azionari che compongono l'indice, o viceversa vende il *future* attendendosi un ribasso dei corsi. In generale lo speculatore trova più appetibile l'investimento sul *future* piuttosto che agire direttamente sul mercato sottostante, questo è dovuto all'effetto

leverage che caratterizza gli strumenti derivati ed inoltre ai più contenuti costi di transazione rispetto a quelli del mercato a pronti. La figura dello speculatore serve in un mercato a bilanciare disequilibri tra domanda ed offerta.

- L'hedger utilizza il *future* sull'indice di borsa per tutelarsi da rischi derivanti dal possesso di un determinato portafoglio titoli. La sua operazione consiste nel vendere lo *stock index future*, assumendo una posizione contraria a quello del proprio portafoglio titoli, in modo da avere una protezione parziale o totale, da ribassi dei corsi azionari. L'*hedger* trasferisce il rischio su altri soggetti che sono disposti ad assumerlo, gli speculatori. Ovviamente tale copertura si può attuare anche su un portafoglio *short* (cioè con titoli venduti allo scoperto), però in questo caso l'operatore sarebbe classificabile non come *hedger*, bensì come speculatore; questo poiché l'investimento in titoli è considerato tale quando il rendimento cui si mira è quello offerto dalla distribuzione dei dividendi, non quello ottenibile dalla differenza tra prezzo di vendita e prezzo di acquisto (capital gain).

- L'arbitraggista agisce contemporaneamente sul mercato a pronti e sul mercato a temine, cercando dei disallineamenti di quotazione tra le attività sottostanti al *future* sull'indice (mercato a pronti) ed il *future* sull'indice (mercato a termine); l'intento di quest'operatore è quello di lucrare piccoli importi ma su grandi volumi, senza rischio. L'arbitraggista svolge due importanti compiti, aumenta lo spessore e la liquidità del mercato e fa sì che i prezzi a termine siano allineati con quelli a pronti.

E' da notare il fatto che, nonostante gli *stock index future*, come peraltro tutti gli altri prodotti derivati, siano nati per soddisfare esigenze di copertura e quindi di riduzione del rischio, abbiano ormai assunto connotazioni speculative; questo perché gli strumenti derivati, caratterizzati dall'effetto leva, permettono agli speculatori di ottenere percentuali di guadagno elevate pur con variazioni contenute del sottostante.

Alcune indagini in merito all'utilizzo dei *future*, hanno appunto dimostrato tale fatto, rilevando che l'utilizzo dei future per copertura ammonta ad un 5-20% della totalità dei contratti, la quota a titolo di arbitraggio è pari al 5-10% mentre il rimanente 70-80% dei contratti è stipulato ai fini del trading.

5.2. Diffusione e successo dello "stock index future"

La crescita dei volumi negoziati e l'elevata diffusione dei *future* sull'indice sono giustificati da diversi elementi, peraltro comuni a tutti i prodotti derivati:

- La flessibilità insita nello strumento. L'utilizzo di questa tipologia di contratti permette di soddisfare le esigenze di molti investitori e consente a questi di coprire posizioni o di ribaltarle (da *long* a *short* e viceversa) con estrema facilità e tempestività, permettendo l'adeguamento del portafoglio al mutare delle aspettative di mercato.

- La maggior convenienza sia per ciò che riguarda i costi, sia per la liquidità offerta dall'operatività in *future* (ed anche in *option*), rispetto alle analoghe posizioni nei mercati a pronti.

- La tendenza a delegare agli investitori istituzionali la gestione di una parte sempre più rilevante di fondi; tali investitori istituzionali, diversificando il proprio portafoglio su diversi mercati, utilizzano gli *stock index future* per meglio gestire il rischio.

- Lo sviluppo della tecnica dell'indicizzazione dei portafogli che si limita a replicare passivamente l'andamento di un determinato indice di mercato. Gli *stock index future* sono ampiamente utilizzati per la gestione di portafogli diversificati, basati sui medesimi indici di mercato collegati al *future*.

- La forte volatilità delle attività sottostanti. La situazione che viene a crearsi è un processo che si autoalimenta: si ha la crescita della domanda di copertura da parte degli operatori avversi al rischio, un conseguente stimolo della domanda di tipo speculativo che a sua volta comporta l'azione da parte degli operatori arbitraggisti mettendo alla prova la rapidità dei meccanismi di aggiustamento dei prezzi.

- La concorrenza tra le varie piazze finanziarie. Questa si manifesta con l'introduzione, da parte di borse più avanzate, di strumenti derivati su attività finanziarie di paesi esteri. Così facendo i mercati nazionali sono spinti ad azioni di difesa allo scopo di frenare l'uscita dei volumi di contrattazione a favore di mercati esteri.

5.3. Il mib30, indice di riferimento per lo stock index future del mercato italiano, il fib30

5.3.1. Caratteristiche

Per la creazione dello *stock index future* italiano (fib30) si utilizza come base l'indice mib30, il quale presenta le seguenti caratteristiche:

- E' un indice campionario, cioè considera non la totalità dei titoli quotati sulla piazza finanziaria italiana, bensì un numero limitato, individuato in 30.

- E' un indice aperto, nel senso che la sua composizione viene normalmente rivista due volte l'anno, in marzo e settembre. Gli eventuali aggiornamenti hanno efficacia a partire dal terzo giorno di borsa aperta successivo alla prima scadenza degli strumenti derivati ad esso collegati. Qualora si verificassero circostanze eccezionali, quali ad esempio scissioni di società, la revisione potrebbe essere posticipata o anticipata, previa comunicazione al mercato, al fine di non effettuare interventi troppo ravvicinati.

- Presenta una buona capacità di replica delle performance del intero listino, poiché i 30 titoli che ne compongono il paniere, rappresentano circa il 70% della capitalizzazione di borsa ed il 75% degli scambi totali. Alcuni calcoli svolti dal Consiglio di borsa, evidenziano la buona capacità dell'indice mib30 di replicare l'andamento della borsa italiana, individuando in una percentuale del 99% le variazioni, intervenute nell'indice generale, che il mib30 è stato in grado di cogliere e rappresentare con il suo andamento.

- La sua costruzione e semplice, grazie a precise regole che limitano l'arbitrarietà nella selezione dei titoli. Tali regole mirano ad individuare i titoli più liquidi ed a maggior capitalizzazione, facendo sì che l'indice sia in grado di rappresentare i mutamenti strutturali relativi a liquidità e capitalizzazione dei titoli quotati sulla borsa italiana. I primi 10 titoli dell'indice rappresentano circa il 70% di questo, i primi 15 raggiungono l'80% e i primi 20 arrivano al 90%; questo rende più agevole la pianificazione e l'esecuzione di adeguate operazioni di copertura.

5.3.2. La costruzione dell'indice

La costruzione dell'indice può essere distinta in due fasi di lavoro: la prima che riguarda l'individuazione dei titoli da inserire nell'indice, la seconda che comporta il calcolo dell'indice medesimo.

La selezione dei titoli viene operata dal Consiglio di borsa il quale ne comunica la lista al mercato almeno dieci giorni prima che la modifica abbia efficacia. L'individuazione dei titoli è legata a parametri quali la liquidità e la capitalizzazione rappresentati in una indicatore detto ILC.

La procedura svolta è la seguente:

- Per ogni titolo quotato, si calcola la capitalizzazione media giornaliera (CapMG), risultante dal prodotto del numero di titoli in circolazione per la media dei prezzi ufficiali nei sei mesi precedenti ed il volume medio giornaliero degli scambi relativo allo stesso semestre VolMG.

- Rapportando le due grandezze precedenti si ottiene il cosiddetto coefficiente alfa del titolo:

$$\text{Alfa (titolo)} = \text{CapMG} / \text{VolMG}$$

- Tramite la stessa metodologia, si determina l'analogo indicatore per l'interno mercato e si calcola il rapporto alfa di mercato:

$$\text{Alfa(mkt)} = \text{CapMG(mkt)} / \text{VolMG(mkt)}$$

- Si costruisce, per ciascun titolo, l'indicatore di liquidità e capitalizzazione ILC, che pondera i due elementi nel modo seguente:

$$\text{ILC(titoli)} = \text{CapMG} + \text{alfa(mkt)} * \text{VolMG}$$

- Si pongono i titoli quotati in ordine decrescente di ILC.

- Si procede all'eliminazione di quei valori mobiliari che, pur presentando un elevato valore di ILC, rientrano nelle seguenti fattispecie:

    1. Tra le varie categorie di azioni (ordinarie, privilegiate, di risparmio) di una medesima società, quelle con ILC più basso. Al fine di immunizzare l'indice, almeno

parzialmente, dal rischio specifico relativo alla società emittente si include un solo titolo per emittente.

2. Azioni con periodo di negoziazione ufficiale non sufficientemente significativo, fatta eccezione per i titoli di nuova quotazione purché con capitalizzazione sufficiente.
3. Titoli con alfa superiore a 10.000, con preciso scopo di evitare l'inclusione di quei titoli ad elevata capitalizzazione ma con scarsa liquidità.
4. Titoli per i quali si teme, al momento della selezione, il successivo venir meno dei requisiti di capitalizzazione e di liquidità o della stessa quotazione.

- Si scelgono i primi 30 titoli che faranno parte dell'indice e si considereranno come riserve i primi cinque esclusi. Tali titoli di riserva vengono utilizzati qualora nel periodo di validità del paniere si debba procedere alla sostituzione di titoli che subiscono la cancellazione dal listino, la sospensione dalla quotazione per un periodo superiore a dieci giorni consecutivi di borsa aperta o che, in seguito ad altri eventi, incorrono in consistenti ed accertate perdite di liquidità e/o di capitalizzazione. Ovviamente in caso di modifica della composizione dell'indice, viene effettuato il ricalcalo dei pesi di tutti i titoli; in questo caso si utilizza come prezzo base quello di apertura del giorno stesso in cui diviene efficacie l'aggiornamento e come numero base delle azioni quello presente sul listino ufficiale tre giorni di borsa aperta antecedenti l'intervento.

- L'indice è il seguente:

$$\text{Mib30} = \frac{\sum_{i=1}^{30} \frac{p(t)}{p(0)} \cdot p(0) \cdot q(0)}{\sum_{i=1}^{30} p(0) \cdot q(0)} \cdot 10.000$$

Dove:

- $p_i(0)$: prezzo base di ogni azione, è il prezzo di apertura della seduta in cui si effettua l'aggiornamento dell'indice o, in mancanza di questo, l'ultimo prezzo del giorno precedente, cioè quello dell'ultimo contratto concluso;
- $p_i(t)$: prezzo corrente, è l'ultimo prezzo registrato da ciascun titolo;
- $q_i(0)$: è il numero di titoli;

I prezzi considerati sono quelli *tel-quel*, quindi non soggetti a rettifica in caso di stacco dei dividendi. Lo scostamento nelle quotazioni che incide sui contratti *future* con scadenza giugno e settembre, periodo in cui si concentra lo stacco dei dividendi della maggior parte dei titoli inclusi nell'indice, viene misurato in termini di punto indice dal Consiglio di Borsa e reso noto agli operatori.

CAPITOLO 6

Lo Stock Index future

I *future* sono operazioni di compravendita a termine che vengono negoziate su mercati di borsa organizzati; esistono sia *future* su attività reali (come quelli sulle materie prime), sia su attività finanziarie (come quelli su tassi d'interesse o su indici finanziari). In questa sede interessa offrire una breve rassegna relativamente ai *future* di tipo finanziario ed in particolare sarà preso in considerazione, il *future* sull'indice del mercato di borsa italiano. Innanzitutto tali attività finanziarie si caratterizzano per il notevole grado di standardizzazione, relativo a più elementi caratterizzanti il contratto: l'oggetto, l'importo, la scadenza, orari e regole di contrattazione, tipologia ordini, modalità di liquidazione. Concludere un contratto nelle vesti di futuro acquirente, significa assumere una posizione lunga nei confronti di una data attività, mentre concludere un contratto nelle vesti di futuro venditore, significa assumere una posizione corta, nei confronti di una data attività. Sui mercati *future* le scadenze dei contratti sono in numero limitato; solitamente vi sono quattro scadenze di contratti in un anno e la maggioranza delle borse *future* si attiene, in merito alle date di scadenza, ai mesi di marzo, giugno, settembre e dicembre. Il mese in cui un contratto scade viene definito mese di scadenza; anche il giorno della settimana in cui il contratto scade è standardizzato e vi è perfino un termine orario entro il quale liquidazione deve avvenire. Tuttavia nel giorno di liquidazione, pur esistendo il sottostante di uno *stock index future*, si preferisce procedere con la liquidazione a mezzo pagamento (*cash settlement*).

6.1. L'esempio italiano

L'esempio italiano di *stock index future* è il Fib30, il cui valore monetario si ottiene moltiplicando il valore dell'indice di borsa di riferimento, il Mib30, per un controvalore pari a 5€; così ad esempio, avendo l'indice pari a 23.000 si avrà un Fib30 pari al valore di 23.000 * 5 €, cioè 115.000 €.

Analizzando il panorama internazionale il nostro future si colloca a livello intermedio rispetto ai valori più alti dello S&P500 e alle poche migliaia di euro degli indici danese e spagnolo.

La scelta operata dal nostro paese è motivata da due considerazioni:

- Evitare che con la fissazione di un valore troppo basso del *future*, si dia spazio ad occasioni di speculazione decisamente rischiose, operate da parte di piccoli investitori attirati dalle possibilità di guadagno (l'effetto leva l'aspetto più attraente), senza nessuna finalità reale di *hedging* ed in mancanza di un'adeguata conoscenza della finanza e dei mercati stessi.

- Per contro, si è voluto evitare che il valore scelto fosse troppo alto, per quanto tale condizionare fosse particolarmente allettante ai fini di un contenimento dei costi operativi del sistema telematico, per non allontanare la fascia di operatori di medie dimensioni portatori di liquidità sul mercato.

6.2. Tick e scadenze

Attualmente il movimento minimo del prezzo è di 5 punti dell'indice, mentre precedentemente era fissato in 1 punto percentuale.

Le scadenze sono standardizzate e fissate nei mesi di: marzo, giugno, settembre e dicembre.

Per ciò che riguarda le quotazioni, queste sono presenti nel numero di tre contemporaneamente, ovvero la più vicina e le due immediatamente successive.

6.3. Modalità di negoziazione e tipologie di proposte

Il future viene trattato sul sistema telematico, in continua, dalle 9.30 alle 17.30.

Rispetto al mercato azionario è stata eliminata l'asta di apertura, ciò è dovuto a due considerazioni:

- Una di ordine tecnico: il pacchetto informativo acquistato contemplava soltanto la contrattazione continua e le modifiche avrebbero ulteriormente posticipato l'avvio delle negoziazioni.

- Una di ordine commerciale: tale mercato è principalmente utilizzato da operatori professionisti le cui preferenze sono rivolte ai mercati continui.

Il sistema è di tipo misto perché include caratteristiche del sistema "*order-driven*" e figure del sistema "*quote-driven*", che sono i market maker.

Il sistema *order-driven* consente:

- L'instaurarsi di un trattamento paritario degli ordini, essendo questi eseguiti secondo le priorità di prezzo e di tempo.

- L'esplicarsi di un elevato livello di competitività fra intermediari le cui proposte sono tutte presenti sul book.

- La riduzione dello spread denaro/lettera, in quanto è sempre possibile inserire ordini all'interno della miglior forchetta.

Tuttavia un sistema *order-driven* puro, sarebbe caratterizzato da eccessiva volatilità in quanto i prezzi tenderebbero a fluttuare seguendo l'andamento di domanda ed offerta; la presenza dei *market maker* funge da stabilizzatore.

I servizi offerti dal sistema telematico sono:

- La diffusione di proposte negoziali.
- La conclusione automatica dei contratti.
- L'informazione sui contratti conclusi.

Per tali servizi, gli operatori autorizzati devono corrispondere una quota fissa pari a 10.000 € annui per postazione e una quota variabile pari a 1,14 € per contratto concluso.

Le proposte di negoziazione che gli operatori possono inserire sono di 4 tipi:

- singole: in acquisto ed in vendita.

- combinate (COMBO) standard: tali proposte standardizzate, consistono in un contemporaneo ordine di acquisto e di vendita, per

pari importo, ma su scadenze differenti del future. Sono standard perché predefinite dal sistema e considerano la scadenza più vicina per la prima operazione e quella immediatamente successiva per la seconda operazione. Tali proposte compaiono su un book a parte sul quale viene direttamente quotato lo spread tra le due scadenze del contratto.

- combinate non standard: sono altre combinazioni di ordini che, in quanto non standard, devono essere specificate al momento dell'immissione ed eseguite contemporaneamente e totalmente.

- quotazioni denaro/lettera: ovviamente riservate ai soli market maker.

Gli ordini possono essere immessi a "prezzo di mercato" o "con limite di prezzo". Gli ordini vengono classificati mediante la stessa metodologia utilizzata per il mercato a pronti, quindi le proposte vengono ordinate in ordine decrescente di prezzo per gli acquisti ed in ordine crescente di prezzo per le vendite, in caso di parità di prezzo si considera l'ordine temporale di immissione.

Le proposte a prezzo di mercato, vengono abbinate automaticamente dal sistema ad una o più delle migliori proposte di senso contrario. Le proposte con limite di prezzo, invece, vengono concluse solo se dall'altra parte del book vi sono proposte con prezzo uguale.

Per le proposte singole o combinate standard, possono essere inseriti ulteriori parametri che ne definiscono specifiche modalità di esecuzione relative alla validità temporale o alla quantità eseguibile. Alcuni di questi parametri sono:

- FAK (*fill and kill*): esegui totalmente o parzialmente, cancella l'ineseguito.
- FAS (*fill and store*): rimane nel book fino a scadenza.

- FOK (*fill or kill*): esegui totalmente o cancella.
- GED (*good end of day*): valido fino a fine seduta.
- GTC (*good till cancel*): valido sino a cancellazione.
- GTH (*good till hour*): valido fino ad orario specificato.

6.4. I margini di sicurezza

6.4.1. Margine iniziale

Come per ogni contratto future la stipulazione di un contratto *stock index future*, sia esso in acquisto (posizione *long*), sia esso in vendita (posizione *short*), prevede il versamento di un margine iniziale alla Cassa di compensazione e garanzia (*Clearing house*). Tale margine non è contemplato per i contratti che chiudono una precedente posizione.

L'importo richiesto come margine è alquanto contenuto in rapporto al valore nominale del contratto stesso, ciò è giustificato dalla funzione propria del margine iniziale, e cioè fronteggiare le eventuali perdite di un giorno di contrattazione e contemporaneamente limitare l'investimento iniziale dell'operatore. In Italia la percentuale del margine, che viene definito dalla Cassa stessa, non può essere inferiore al 4%; questa percentuale è stata individuata sulla base di uno studio relativo alla volatilità dell'indice Comit30 (si deve ricordare che il Mib30 deriva dall'indice Comit30), in particolare si è rilevato che lo scarto quadratico medio delle variazioni percentuali giornaliere dell'indice è pari 1,6. Se si ipotizzasse per le suddette variazioni una distribuzione normale, prendendo due volte lo scarto quadratico medio si coprirebbe il 96% dei casi.

Inizialmente gli intermediari membri della *clearing house* dovevano versare un margine iniziale pari al 5,2% del valore nominale del contratto; in

seguito dal maggio del 1995, a causa della maggior volatilità effettiva dei prezzi, tale margine è stato alzato fino al 6,5%. Da ciò si desume quindi che l'effetto leva del Fib30 è pari al 15,38, che l'inverso di 6,5; con ciò significa che con l'importo equivalente al valore nominale del Fib30 si possono acquistare 15 contratti future. Ad esempio, se il Fib30 quotasse 23.000, la somma di 115.000 €, consentirebbe l'acquisto di 15 contratti future poiché per ciascun contratto sarebbe sufficiente versare la somma iniziale pari a 7.666,67 € a fronte del margine di garanzia richiesto dalla *clearing house*.

Tale versamento può essere corrisposto in contanti oppure in titoli di Stato: nel caso del pagamento in contanti si ha diritto ad una remunerazione pari al tasso *Ribor* (*Rome interbank offered rate*, cioè il tasso lettera sulla lira interbancaria) medio mensile meno 1,3%, nel secondo caso si possono conferire i titoli trattati sull'Mts (Mercato all'ingrosso dei titoli di Stato), eccetto i Cte, che saranno valutati all'85% del valor nominale. Gli operatori a loro volta richiedono ai propri clienti margini iniziali maggiori, pari circa la 10%. Il margine iniziale viene trattenuto dalla *clearing house* fino alla chiusura della posizione e viene utilizzato a fini di copertura in caso che l'operatore sia inadempiente.

Sono necessarie alcune precisazioni in quanto le cose nella realtà sono più complesse, infatti, la *clearing house* italiana, per adeguarsi a quanto in vigore in altre borse *future*, utilizza il sistema di calcolo dei margini iniziali sviluppato dalla *Option Clearing Corporation di Chicago*, noto come *Tims* (*Theoretical Intermarket Margins System*). Con tale meccanismo, i margini iniziali non rimangono fissi, ma subiscono variazioni giornaliere.

Il motivo di tale ricalcalo va ricercato nella necessità della Cassa di Compensazione e Garanzia di avere a disposizione un margine iniziale continuamente aggiornato rispetto all'evoluzione dei prezzi e dei conseguenti rischi, per essere sempre in grado, in caso di insolvenza di un operatore, di liquidare le posizioni alle condizioni di mercato più sfavorevoli senza rimetterci del proprio. Tale esigenza è tanto più necessaria quanto più complessa è la posizione aperta dall'operatore, che può comprendere *future* e *option* su diversi

prodotti e con diverse scadenze; in queste situazioni il ricalcalo dei margini secondo il sistema *Tims* permette di adeguare prontamente la garanzia ritenuta sufficiente per la copertura.

Il principio cui è ispirato il metodo del *Tims* e fondato sulla valutazione del rischio contenuto in un portafoglio di *future* e/o *option* , ipotizzando una variazione dei prezzi giornaliera pari ad un ammontare massimo definito "intervallo di margine", in modo contrario e quindi avverso alla posizione del portafoglio considerato. I valori teorici dei contratti *future* e/o *option* che emergono da tale calcolo vengono considerati come valori teorici di liquidazione, cioè come i prezzi di mercato a cui la *Clearing House* potrebbe, nel peggiore dei casi ipotizzabili, liquidare le posizioni di un aderente che risulta essere inadempiente.

Successivamente, cioè a fine giornata, si calcolano i costi/ricavi di liquidazione e l'effettivo valore corrente (prezzo di chiusura del derivato) di ogni contratto. L'importo ottenuto è il nuovo margine iniziale; questo comporta che se le garanzie (in contanti e/o in titoli) che l'aderente ha depositato precedentemente presso la *Clearing House* sono sufficienti, nessun altro versamento è richiesto, alternativamente, qualora non ci fosse copertura dei margini, l'aderente è tenuto a ulteriori versamenti, volti al reintegro delle garanzie, che devono essere versati entro le ore 9:00 del giorno successivo, pena la chiusura coatta della posizione in essere.

6.4.2. Margine di variazione

Sulla base del sistema *marking-to-market*, tutte le operazioni aperte vengono liquidate alla fine di ogni giorno, generando un addebito o un accredito sul conto che ciascun operatore autorizzato ha in essere con la Cassa di Compensazione e Garanzia; tale posizione è riaperta allo stesso prezzo di chiusura del precedente giorno. Praticamente accade che al termine di ogni

giorno di borsa la *Clearing House* determina la variazione dei prezzi del contratto *future* risultante dalla differenza tra la quota di chiusura del giorno corrente e quella del precedente. Tale variazione deve essere moltiplicata per 5 € e rappresenta un addebito o un accredito da regolare in contanti, a seconda della posizione assunta dall'operatore; le eventuali perdite devono essere coperte entro le ore 9,00 del giorno successivo, prima, quindi, dell'apertura della nuova giornata borsistica, pena la chiusura automatica del contratto da parte della stessa *Clearing House*. Tuttavia la Cassa può chiedere dei versamenti infragiornalieri di margini in seguito a variazioni rilevanti ed improvvise dei prezzi, oppure, quando ritenga una posizione assunta da un cliente troppo rischiosa; l'arco di tempo entro il quale il cliente deve adempiere all'obbligo è decisamente contenuto, questo in seguito al motivo che spinge la *Clearing House* a richiedere tali versamenti aggiuntivi.

La struttura del sistema dei margini ed il regolamento quotidiano delle posizioni permettono di limitare il rischio di controparte e di permettere alla Cassa di chiudere eventuali posizioni di clienti inadempienti senza rimetterci del suo. Tutto ciò è a salvaguardia ed a tutela dell'integrità del mercato e della sicurezza del mercato.

6.4.3. Prezzo di chiusura

Il prezzo di chiusura giornaliero dello *stock index future* viene calcolato dalla *Clearing House* come media ponderata per le quantità, dei prezzi dell'ultimo 10% dei contratti scambiati nell'arco della giornata borsistica. Quando si tratta di contratti su scadenze successive alla più vicina, qualora si verifichino carenze di liquidità del contratto, si può procedere calcolando il prezzo di chiusura su una percentuale di titoli più elevata rispetto al 10%, nello specifico si può arrivare fino al 30%; alternativamente il prezzo di chiusura può

risultare dalla media delle migliori quotazioni denaro/lettera di un arco di tempo, scelto dalla Cassa stessa, che non deve essere inferiore a 10 minuti.

L'ultimo giorno di contrattazione il prezzo dell'indice è dato dai prezzi di apertura dei titoli componenti l'indice di riferimento. Qualora durante l'ultimo giorno di contrattazione alcuni titoli del paniere sottostante non aprano, si calcola la media aritmetica delle medie ponderate dei cinque migliori spread denaro/lettera. Qualora alcuni titoli siano sospesi, si utilizzano i prezzi degli ultimi contratti conclusi, anche in data antecedente.

Questo sistema vuole agevolare l'operatore arbitraggista che, il giorno di scadenza, potrà così chiudere la posizione aperta sul mercato dei titoli, quello cash, senza subire il *basis risk*, cioè il rischio derivante dall'andamento non ben sincronizzato tra il prezzo dell'indice *cash* ed il prezzo del *future*.

6.4.4. Il giorno di liquidazione

Il termine delle contrattazioni di uno *stock index future* è dato dal momento in cui sono stabiliti i prezzi di apertura dei titoli che compongono l'indice di riferimento, nel terzo venerdì del mese di scadenza del contratto *future*. Il primo giorno lavorativo successivo all'ultimo giorno di liquidazione, viene effettuato il regolamento. Qualora il contratto giungesse a scadenza, a titolo di consegna, si procederebbe al regolamento per contanti, sistema del *cash settlement*; tale sistema prevede che si riceva o si paghi, l'importo pari alla differenza fra il prezzo finale di regolamento a scadenza (*exchange delivery settelment price*) ed il prezzo di chiusura del giorno precedente.

La liquidazione anticipata, invece, usata molto più frequentemente, consiste nella stipulazione di un contratto di segno opposto rispetto alla posizione originariamente aperta, versando oppure ottenendo l'importo della differenza fra il prezzo negoziato e quello del giorno precedente.

Nel giorno di scadenza del contratto le negoziazioni si concludono alle ore 10,30, dopo che è stato determinato il prezzo di apertura dei trenta titoli; la nuova scadenza è quotata a partire dal primo giorno borsistico successivo.

Da un punto di vista operativo il fatto che le negoziazioni siano bloccate alle 10.30 preclude la possibilità di operare a fini di copertura (*hedging*) sfruttando il contratto in scadenza; tali strategie possono essere realizzate attraverso l'utilizzo dei contratti *future* sulle due scadenze successive le cui quotazioni continuano normalmente.

CAPITOLO 7

La Cassa di Compensazione e Garanzia, la Clearing House italiana

A differenza delle negoziazioni che si attuano nei mercati fuori borsa,[12] nei mercati future vi è la presenza di un organo di controllo detto Clearing House; tale ente, solitamente rappresentato da una società per azioni, posseduta dagli stessi intermediari che partecipano alle contrattazioni dei future, si propone di assicurare la compensazione ed il buon fine dei contratti e di emanare regolamenti che disciplinino l'operatività del mercato future ad integrazione delle disposizioni previste dalla legge.

La *Clearing House* è un organismo che si prefigge l'obiettivo di garantire il buon esito dei contratti; per fare ciò si interpone tra la due parti che stipulano il contratto. Ciò comporta che la controparte per ogni operatore che stipuli un contratto *future*, su un indice di borsa, su valute, su materie prime, sia la *Clearing house* stessa. In questo modo la *Clearing House* è controparte, nella stipulazione di un contratto *future*, sia dell'acquirente, sia del venditore del contratto. Tale metodologia operativa è stata strutturata per far sì che sulla stipulazione dei contratti *future* non pesasse il rischio di controparte, infatti la *clearing house* è in grado di gestire l'eventuale inadempienza degli operatori garantendo così la solvibilità per ogni contratto. Ancora il versamento dei margini di garanzia richiesto agli acquirenti/venditori e la liquidazione giornaliera di ogni operazione, consente alla *Clearing House* di non rischiare di doverci rimettere del proprio qualora un operatore risulti inadempiente.

[12] Definiti mercati *Otc*, acronimo di *Over The Counter*.

7.1. Ruolo

In Italia la *Clearing House* è la Cassa di Compensazione e Garanzia, essa si pone come controparte di ogni operatore e garantisce il buon fine dei contratti. La Cassa come ogni altra *Clearing House* è in grado di coprire l'inadempienza degli operatori in virtù della sua mancata esposizione al rischio di variazioni di prezzo in quanto fungendo da venditore per tutti i compratori e acquirente per tutti i venditori, non detiene posizioni aperte sul mercato.

Per poter operare sull'*idem*, il mercato italiano dei derivati, è condizione indispensabile, l'adesione alla Cassa di Compensazione e Garanzia; tale adesione può assumere diverse configurazioni a seconda delle caratteristiche degli enti che aderiscono:

- Adesione generale: è riservata agli enti creditizi ed alle Sim con un patrimonio netto non inferiore a circa 52 milioni di euro; agli aderenti è concesso svolgere attività di compensazione per conto proprio, per conto della propria clientela e per conto dei propri aderenti indiretti.

- Adesione individuale: è riservata agli enti creditizi ed alle Sim con un patrimonio netto non inferiore a circa 5,2 milioni di euro; consente di svolgere attività di compensazione per conto proprio e per conto dei propri clienti.

- Adesione indiretta: è riservata alle banche e alle Sim in difetto dei requisiti precedenti ed agli agenti di cambio, impone l'espletamento dell'attività di compensazione per il tramite del proprio aderente generale.

Qualora si verifichi che un aderente generale oppure un aderente individuale sia inadempiente, la Cassa di Compensazione stessa provvede, innanzitutto, a sospendere l'aderente dalle contrattazioni, salvo il fatto che si ritenga che il pagamento possa avvenire entro tre ore, e successivamente provvede a liquidare le posizioni contrattuali utilizzando le disponibilità, sia in titoli, sia in contante detenuti dall'aderente inadempiente; nel caso in cui le disponibilità dell'aderente non siano sufficienti la Cassa Di Compensazione provvede con i propri mezzi.

La Cassa può differire la liquidazione delle posizioni detenute dall'aderente generale per conto di aderenti indiretti; ciò serve a valutare l'eventuale disponibilità di altri aderenti indiretti ad assumere tali posizioni contrattuali.

Se fosse un aderente indiretto ad essere inadempiente, sarebbe compito dell'aderente generale comunicare alla Cassa l'inadempienza e procedere alla liquidazione delle posizioni; nel caso in cui le disponibilità dell'aderente indiretto non siano sufficienti ad estinguere il debito, sarà l'aderente generale a dover intervenire con i propri mezzi.

7.2. Gli operatori

Le categorie che possono operare sul Fib 30 sono: i *broker*, i *dealer*, i *market maker*, a condizione che siano aderenti diretti (adesione generale e adesione individuale) o aderenti indiretti, come già ricordato sopra.

- I *broker*, possono inserire nel book solo ordini della propria clientela.

- I *dealer*, possono esporre nel book solo ordini per conto proprio.

- I *market maker,* il cui ruolo può essere ricoperto solo da Sim di negoziazione per conto proprio e banche, con requisiti patrimoniali superiori rispettivamente a 5,2 e 10,3 milioni di euro. Devono iscriversi in un apposito albo e oltre all'inserimento di ordini per conto proprio si impegnano ad immettere quotazioni denaro/lettera al fine di assicurare la necessaria liquidità e continuità delle negoziazioni.

  Non godendo di particolari privilegi, i *market maker* non devono sottostare a particolari vincoli; per tutta la durata delle contrattazioni, devono fornire proposte in acquisto e in vendita per almeno 10 contratti sulle prime due serie di scadenze, tuttavia non sono soggetti a limitazioni sullo *spread* applicato.

7.3. I costi

Per quanto attiene ai costi che gli operatori devono corrispondere alla Cassa di Compensazione e Garanzia per poter operare sul mercato dei derivati, sono previsti dal sistema importi relativi a quote ed importi relativi a commissioni. E' prevista una quota fissa annuale, che è differente a seconda della natura dell'aderente ed una importo a titolo di commissione, che varia in base alla tipologia di contratto stipulato.

Per la quota fissa gli importi sono pari a:

- 15.000 euro circa per gli aderenti generali
- 7.500 euro circa per gli aderenti individuali
- 2.500 euro circa per gli aderenti indiretti

La commissione che viene applicata ad ogni contratto stipulato sul Fib 30 è pari a 0,43 euro.

A questi costi vanno aggiunte le commissioni per la copertura dei costi di gestione di titoli costituiti a garanzia; il costo è pari allo 0,02 % per mese (o

frazione di esso), calcolato sul saldo massimo del valore nominale dei titoli depositati da ciascun aderente nel mese di riferimento.

Inoltre su ogni intermediario che operi sull'idem (*Italian Deridvatives stock Market*) grava una quota annuale pari a circa 10.000 euro ed un costo per operazione di 1,13 euro, per il collegamento con il servizio di contrattazione e liquidazione del sistema telematico.

CAPITOLO 8

Il prezzo del Fib30

Il Fib30, presenta due tipologie di contratto, il contratto *long future* (acquisto del *future*) ed il contratto *short future* (vendita del *future*). Si deve comunque prendere in considerazione il fatto che nonostante si possano porre in essere due tipologie differenti di contratto, il rischio che le caratterizza è, come presto verrà descritto, esattamente speculare.

Il *long future* viene utilizzato quando ci sono aspettative di rialzo di mercato, viceversa il *short future* viene utilizzato quando ci sono aspettative di ribasso di mercato. Questa differenza di utilizzo vale per le strategie che si prefiggono di ottenere rendimenti affrontando un coefficiente di rischio, diversa è invece la situazione che si presenta quando ad operare è un arbitraggista, il quale per definizione, nell'attuare le sue operazioni, non sopporta nessun rischio e si preoccupa di individuare discrepanze tra due o più mercati e di attuare operazioni che gli permettano di lucrare sfruttando tali anomalie (temporanee).

Il generarsi delle discrepanze di mercato dipende da vari motivi che saranno analizzati successivamente, ma è dato dal fatto che la quotazione del *future* non sempre è allineata con quello che dovrebbe essere il suo valore reale, indicato da quello che viene definito prezzo teorico del *future*. Ciò è dovuto alla difficoltà di individuare prontamente quale dovrebbe essere il valore teorico del future, il quale, come si vedrà più avanti, dipende da più fattori, che in alcuni casi sono stimati (dividendi) e non certi. Si può allora dire che il valore teorico del *future* è suscettibile di variazioni dovute all'azione congiunta di più fattori e questo genera ulteriori difficoltà nel calcolo immediato di quello che dovrebbe essere il valore reale del *future*, da parte di molti operatori. Tuttavia tra il ventaglio di possibilità che il Fib30 lascia agli operatori, l'individuazione del prezzo teorico del *future* ed il suo raffronto con la quotazione corrente interessa

principalmente le operazioni di arbitraggio, le quali necessitano di tale differenza per essere attuate; ciò significa che se il mercato fosse perfettamente efficiente l'arbitraggista non riuscirebbe ad operare perché non si creerebbero le condizioni necessarie alle sue operazioni. Meno rilevante è quest'aspetto per le operazioni di copertura (*hedging*) e per le operazioni di speculazione (*trading*).

8.1. Operazioni long e short

Per calcolare l'utile o la perdita di un contratto *future* in acquisto è sufficiente calcolare la differenza tra il prezzo del *future* al tempo T ed il prezzo del *future* al momento della stipulazione del contratto. Nel caso di un *future* in acquisto (*long future*) il profitto massimo è teoricamente illimitato, mentre la perdita massima è pari all'intero valore del contratto, poiché il livello minimo che il prezzo del *future* può raggiungere è pari a zero.

Anche per calcolare l'utile o la perdita di un contratto *future* in vendita è sufficiente calcolare la differenza tra il prezzo del *future* al tempo T ed il prezzo del *future* al momento della stipulazione del contratto. Tuttavia il profilo di rischio di un *future* in vendita (*short future*) è speculare rispetto a quello di un *future* in acquisto e quindi si ha che l'utile massimo è limitato all'importo del contratto, mentre la perdita massima è teoricamente illimitata.

In Figura 5 è rappresentato il profilo di rischio di un *Long Future*, mentre in Figura 6 quello di un *Short Future*.

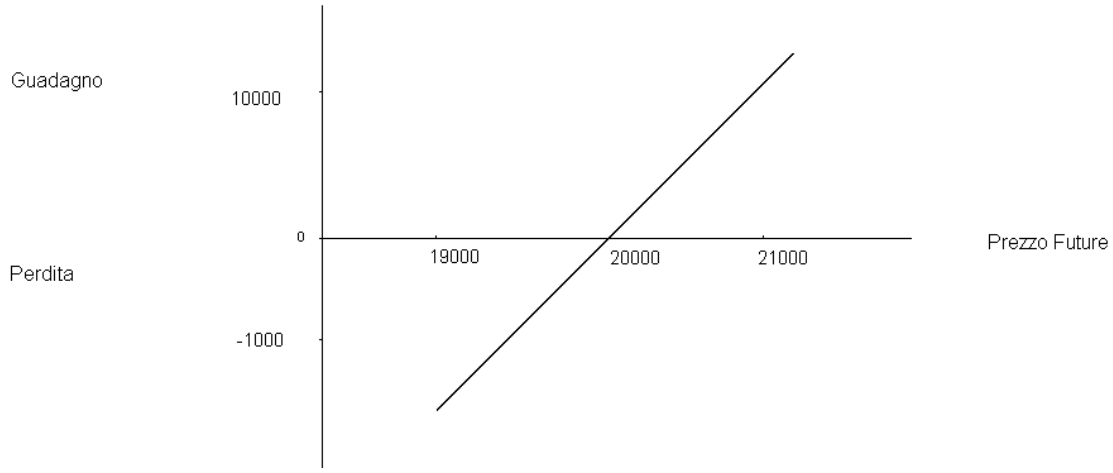

Figura 5

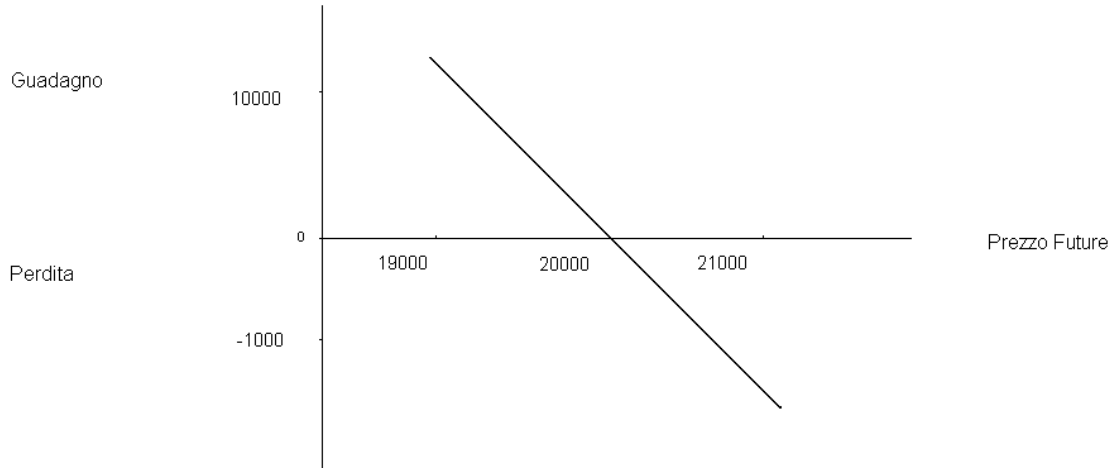

Figura 6

8.2. La determinazione del prezzo del Fib30

Nonostante il suo nome, il *future*, non incorpora nessuna aspettativa degli operatori nei confronti del futuro andamento del suo sottostante. Per meglio comprendere questo aspetto caratterizzante il contratto si può fare riferimento ad un esempio di scenario reale. Si supponga infatti di trovarsi di fronte ad uno scenario in cui a seguito di vicende geopolitiche od economiche negative si diffondano, tra gli operatori, aspettative di instabilità economica. A seguito di tali eventi, non vi è nessun motivo che la spinta alle vendite dei valori mobiliari si riversi nei confronti dei contratti *future*, quando vi è la disponibilità di un mercato a pronti. Saranno poi le fluttuazioni del sottostante ad influenzare il prezzo teorico del contratto *future* e quindi anche la sua quotazione; qualora questo non si dovesse verificare, nel mercato entrerebbero arbitraggisti a riequilibrare con le loro operazioni il mercato a pronti con il mercato a termine.

Inoltre bisogna dire che il prezzo del *future* differisce sempre dal prezzo dell'indice e ciò dipende da più fattori che contemporaneamente concorrono alla formazione del prezzo del *future*. Per comprendere meglio quale siano i fattori responsabili di tale differenza, si potrebbe immaginare una situazione in cui non vi sia differenza i due valori sopra citati. In tale condizione si potrebbero ipotizzare due investimenti equivalenti, per la precisione:

- L'operatore 1 acquista un contratto Fib30, con scadenza tre mesi.
- L'operatore 2 acquista direttamente le azioni dell'indice Mib30 (il sottostante del Fib30) nelle stessa proporzione in cui queste sono presenti nell'indice Mib30, investendo una somma di valore pari al valore odierno del Mib30.

Al fine di rendere realmente omogenee le due forme di contratto nell'esempio che si sta considerando, si devono fare alcune ulteriori precisazioni:

- Le due tipologie di investimento avvengono al medesimo prezzo pari a 21000

- Il tasso d'interesse per attività prive di rischio è del 3% annuo, corrispondente allo 0,75% trimestrale.
- I dividendi medi relativi ai titoli dell'indice Mib30 ammontano al 2%, cioè allo 0,05% su base trimestrale.
- Non vengono presi in considerazione i margini di variazione sul contratto Fib30.

Schematicamente si può scrivere che:

Operatore 2:

- Investimento oggi: pari al prezzo dell'indice al tempo 1
- Risultato tra tre mesi: prezzo dell'indice al tempo 2 + quota dividendi

L'operatore 2 ha acquistato oggi le azioni comprese nell'indice Mib30, nella medesima proporzione, investendo un controvalore pari a 21000. Al termine del periodo d'investimento (tre mesi), l'operatore 2 deterrà in portafoglio le azioni dell'indice Mib30 per un controvalore ottenuto dalla valorizzazione di queste ai prezzi dell'indice Mib30 al tempo 2 (cioè dopo tre mesi); in più avrà maturato una quota di dividendi pari allo 0,5%.

Ipotizzando tre scenari differenti sul futuro andamento dell'indice, rispettivamente:

- Il valore dell'indice non varia
- Il valore dell'indice cresce del 10%
- Il valore dell'indice scende del 10%

Si potranno definire così le tre alternative per l'investimento del nostro operatore 2, rispetto agli scenari futuri ipotizzati:

A) indice invariato: 21000 + 105 = 21105

B) incremento indice (+10%): 21100 + 105 = 21205

C) decremento indice (-10%): 20900 + 105 = 21005

Sempre schematicamente si può scrivere che:

Operatore 1:

- Investimento oggi: controvalore pari al Prezzo dell'indice al tempo 1, investito in Bot con scadenza tre mesi.
- Risultato tra tre mesi: Prezzo dell'indice al tempo 2 - Prezzo del *future* + Montante su Bot

L'operatore 1 invece acquista il *future* sull'indice e, non dovendo effettuare nessun esborso immediato (poiché abbiamo ipotizzato nulli i margini di garanzia)[13] per quest'acquisto, potrà impiegare diversamente il suo denaro; investirà in titoli privi di rischio, nella fattispecie in Bot con scadenza tre mesi, per un ammontare pari al controvalore del prezzo del *future*, cioè per 21000. Trascorso i tre mesi, l'operatore 1 dovrà saldare il suo debito ed acquistare così l'indice sottostante, chiudendo la sua posizione *future*; il suo investimento sarà pari alla differenza tra prezzo del *future* ed il prezzo dell'indice al tempo due, incrementato del capitale investito in Bot a tre mesi e degli interessi relativi maturati. Considerando sempre i tre scenari futuri, già menzionati sopra si avranno le tre alternative per l'investimento dell'operatore 1:

Si potranno così definire le tre alternative di investimento per l'operatore 1, sempre considerando i tre scenari sopra ipotizzati:

A) indice invariato: $(21000 - 21000) + 21000 * 1{,}0075 = 21157{,}5$

B) incremento indice (+10%): $(21100 - 21000) + 21000 * 1{,}0075 = 21257{,}5$

C) decremento indice (-10%): $(20900 - 21000) + 21000 * 1{,}0075 = 21057{,}5$

[13] Tale ipotesi non è così limitativa come sembra a prima vista, poiché nella realtà è possibile dare dei titoli privi di rischio a garanzia dell'acquisto di un contratto future, come forma di margine iniziale; di fatto questi titoli fruttano interessi all'operatore in future.

Le due possibilità di investimento, per come sono state ipotizzate nel nostro esempio, risultano uguali sia in termini di controvalore investito, sia in termini di rischio (qualsiasi fluttuazione di prezzo dell'indice ha il medesimo impatto sulle due tipologie di investimento). Tuttavia, come è ben evidente dai dati presentati, l'acquisto del Fib30, permette all'operatore 1 di ottenere un guadagno maggiore (perdita minore) grazie agli investimenti fatti in Bot a tre mesi, rispetto al guadagno (perdita) dell'operatore 2 limitato alla maturazione di una quota dividendi.

Questa situazione di sistematico vantaggio per l'operatore in *future* non è sostenibile all'interno di un mercato efficiente, infatti altri investitori acquisterebbero contratti *future*, facendone salire il prezzo, fino a che la convenienza nell'esporsi a termine fosse maggiore di quella nell'esporsi a pronti; così facendo si arriverebbe ad un punto in cui le due alternative, acquistare il future,oppure acquistare direttamente l'indice, si equivarrebbero.

Per definizione, in un mercato efficiente, tali condizioni di sistematico vantaggio per una posizione rispetto ad un'altra non si verificano, tuttavia nei mercati finanziari tali condizioni, pur se non sistematicamente, si verificano con una certa frequenza e ciò determina la possibilità per determinati operatori di effettuare operazioni di arbitraggio.

Calcolando il punto di equilibrio tra le due alternative di investimento a cui il mercato tende, si ottiene il valore teorico del future:

P.zzo Indice 2 + Divid. = (P.zzo Indice 2 – P.zzo future) + Montante inv. in Bot

Si può semplificare la nostra equazione, poiché "Prezzo Indice 2" compare a sinistra ed a destra del segno di uguaglianza; considerando inoltre "r" come il tasso d'interesse l'equazione può essere riscritta sviluppando il "Montante investito in Bot", ottenendo:

*Dividendi = - Prezzo Future + [Prezzo Indice 1 * (1 + r)]*

Risolvendo l'equazione in funzione del Prezzo Future si otterrà:

*Prezzo Future = [Prezzo Indice 1 * (1 + r)] – Dividendi*

Volendo sostituire i valori relativi ai nostro esempio, si avrà che il valore teorico del future, deve essere:

*Prezzo Future = 21000 * (1 + 0,0075) – 21000 * 0,005 = 21052,5*

Riscrivendo la formula e riassumendo quanto fino qua ottenuto grazie all'esempio, si otterrà:

Prezzo teorico Future = Prezzo Indice 1 + Interessi – Dividendi

8.3. I fattori che influenzano il prezzo teorico del future

Dai risultati ottenuti nel paragrafo sopra, si può facilmente comprendere quali siano i fattori da cui dipende il prezzo teorico del *future*, anche definito come *fair value*. Esso dipende da quattro variabili:

- Il prezzo dell'indice sottostante.
- Il trascorrere del tempo.
- Il tasso d'interesse (relativo ad impieghi privi di rischio).
- I dividendi (relativi ai titoli contenuti nel sottostante).

Per meglio comprendere l'influenza di queste variabili, che nella realtà agiscono contemporaneamente, rendendo difficile l'individuazione precisa di quello che dovrebbe essere il valore teorico del *future*, conviene analizzarle singolarmente, mantenendo invariate le altre.

Valore dell'indice sottostante: più cresce il valore dell'indice sottostante, più cresce il *fair value* del *future.* La variazione di prezzo del *future* non è tuttavia di eguale misura rispetto a quella del sottostante; questo è ovvio ed intuibile sulla base della formula che è stata esaminata sopra.

Esempio

Tasso d'interesse: 3% annuo Tempo: 3 mesi Dividendi: 2% annuo

Indice = 21000 Valore Teorico del future = 21052,5
Indice = 21100 Valore Teorico del future = 21152,75
Indice = 21200 Valore Teorico del future = 21253

Tempo rimanente alla scadenza: più lontana è la scadenza, più alto è il valore del *future*, questo solo se il tasso per impieghi senza rischio è maggiore dei dividendi.

Esempio

Prezzo Indice: 21000 Tasso d'interesse: 3% Dividendi: 2% annuo

1 giorno alla scadenza Valore Teorico del future = 21000,58
1 mese alla scadenza Valore Teorico del future = 21017,5
3 mesi alla scadenza Valore Teorico del future = 21052,5

Tasso d'interesse: più il tasso d'interesse su attività prive di rischio è elevato, maggiore è il valore del future. Quando si verifica che il tasso d'interesse

è minore dei dividendi, il *future* fa sconto sull'indice; viceversa, quando il tasso d'interesse è maggiore dei dividendi, cosa molto più comune, il *future* fa premio sull'indice.

Esempio

Prezzo Indice: 21000 Tempo rimanente: 3 mesi Dividendi: 2% annuo

Tasso d'interesse = 1% Valore Teorico del future = 20947,5
Tasso d'interesse = 5% Valore Teorico del future = 21105
Tasso d'interesse = 6% Valore Teorico del future = 21157,5

Livello dei dividendi: più i dividendi relativi all'indice sottostante sono elevati, minore sarà il *fair value* del *future*.

Esempio

Prezzo Indice: 21000 Tasso d'interesse: 3% Tempo a scadenza: 3 mesi

Dividendi = 1% Valore Teorico del future = 21105
Dividendi = 3% Valore Teorico del future = 21000
Dividendi = 4% Valore Teorico del future = 20947,5

8.4. Il valore teorico ed il valore di mercato del future

Solitamente i prezzi di mercato del *future* rispecchiano i valori teorici del *future*, tuttavia, come già accennato sopra, si verificano delle situazioni in cui vi sono delle differenze tra il valore teorico ed il valore di mercato del *future*, e può capitare che tali discrepanze siano sufficientemente ampie da consentire ad alcune tipologie di operatori, gli arbitraggisti, di assumere posizioni, sia sul mercato a termine (quello dei *future*), sia sul mercato a pronti (quello dei titoli), per lucrare guadagni sicuri dalla mancanza di coerenza tra i due valori. Tali operatori continueranno ad operare fin tanto che non saranno annullate le discrepanze, o fin tanto che le discrepanze glielo consentiranno. Quindi si può ben verificare che vi siano delle differenze tra valore teorico del *future* e valore di mercato ma che non vi siano le condizioni necessarie al fine di fare operazioni di arbitraggio. L'operato degli arbitraggisti, oltre ad essere proficuo per essi, lo è anche per il mercato in generale, in quanto tali operazioni servono a riportare in equilibrio i mercati su cui sono attuate, permettendo così una coerenza tra prezzo teorico del *future* e valore reale del *future* e rendendo nel suo complesso il mercato efficiente.

Le cause che determinano questi scostamenti che creano opportunità di arbitraggio sono molteplici:

- L'esistenza di due prezzi, uno relativo alla domanda ed uno relativo all'offerta.
- L'incidenza delle commissioni, che varia da operatore ad operatore.
- La prevalenza, in un determinato istante, nel mercato, dei venditori o dei compratori.
- Il differente trattamento fiscale, soggettivo a seconda dell'operatore a cui si riferisce.

- La differenza di trattamento fiscale relativa al future ed al sottostante.

La differenza tra il prezzo del sottostante (prezzo *cash*) e prezzo del *future* sull'indice (I - F) viene denominata *basis*; la *basis* è composta di due elementi:

- *Carry basis*: data dalla differenza tra il valore dell'indice ed il valore teorico del *future*, il *fair value*, ($I - F_v$). Sulla base della formula indicata nel paragrafo precedente, si può dire che la *carry basis* (base del riporto) è uguale alla differenza tra dividendi ed interessi. Alla scadenza tale differenza sarà nulla, infatti si dice che per definizione il valore del *future* a scadenza è uguale a quello dell'indice; non è difficile capirne il perché, in quanto, se si è al giorno di scadenza del contratto *future*, tasso d'interesse e quota dividendi, non potranno concorrere a determinare il prezzo teorico del *future*, poiché il contratto scade oggi.

- *Net basis*: anche detta *value basis*, è data dalla differenza tra *fair value* del *future* e quotazione del *future* (F_v - F). Questa differenza indica se il *future* è sottovalutato o sopravvalutato, in quanto ne misura lo scostamento dal suo prezzo teorico. Ciò significa che se la *net basis* (base netta) è diversa da zero, esiste teoricamente possibilità di arbitraggio; ovviamente prima di porre in essere un'eventuale operazione di arbitraggio si devono considerare i vari costi delle operazioni che si dovranno eseguire per portare a termine l'arbitraggio stesso, quindi nella realtà non è sufficiente che la *net basis* sia uguale a zero.

Riassumendo:

$$Basis = carry\ basis + net\ basis$$

In valori:

$$I - F = (I - F_v) + (F_v - F)$$

Qualora il prezzo di mercato del contratto *future* (*F*) fosse maggiore del suo prezzo teorico (F_v), sarebbe conveniente acquistare i titoli inclusi nel paniere dell'indice, cioè il sottostante, e contemporaneamente vendere il contratto *future*. Tale operazione di arbitraggio viene definita *cash and carry*. L'arbitraggista deve quindi agire contemporaneamente sui due mercati.

Il tutto può essere verificato con un esempio:

- Prezzo dell'indice pari a 21000.
- Tasso d'interesse su attività prive di rischio pari a 3% annuo.
- Dividendi pari a 2% annuo.

Supponiamo che il prezzo di mercato del *future* sia 21100, cioè superiore al suo *fair value* (date le condizioni fissate nell'esempio), pari a 21052,5; realizzando un arbitraggio *cash and carry* si avrà:

- Vendita *future* a 21100
- Acquisto titoli componenti l'indice per un controvalore pari a 21000
- Costo finanziamento della posizione (per tre mesi) pari a 157,5
- Incasso quota dividendi 103,5
- Guadagno da arbitraggio pari a 46

In numeri:

$$+ 21100 - 21000 - 157{,}5 + 103{,}5 = 46$$

Per semplicità non si è tenuto conto di quelli che sono i costi di transazione; nella realtà questi costi sono presenti e l'arbitraggista entrerà in posizione da arbitraggio solo quando riuscirà ad ottenere un profitto (sicuro) al netto di tali costi, che tuttavia possono essere differenti a seconda degli operatori.

Il fatto che i costi di transazione possano differire a seconda degli operatori indica che le soglie oltre le quali una posizione di arbitraggio può essere eseguita, cambiano a seconda degli operatori. Alla voce generica "costi di transazione" si devono comprendere: lo *spread* denaro/lettera sul *future* e sui titoli (componenti l'indice sottostante), i bolli, le commissioni su *future* e titoli. Tali costi possono essere espressi in termini di punti indice, ad esempio nel nostro caso se il loro ammontare fosse stato pari a 200, il punto oltre il quale sarebbe stato conveniente fare un arbitraggio sarebbe individuabile sommando 200 al valore teorico del *future*, quindi pari a (21052,5 + 200) = 21252,5.

Nel caso contrario, in cui il prezzo di mercato del future (F) fosse maggiore del prezzo teorico del *future* (F_v), risulterebbe conveniente per l'operatore arbitraggista, vendere i titoli inclusi nell'indice sottostante, utilizzare i fondi ricavati da tale vendita per acquistare titoli privi di rischio e contemporaneamente acquistare il contratto *future*; tale operazione viene denominata *riverse cash and carry*. Differentemente dalla posizione *cash and carry* nella *riverse cash and carry*, si perdono i dividendi relativi ai titoli venduti, ma allo stesso tempo si maturano gli interessi sui titoli privi di rischio; avendo poi fissato il prezzo di riacquisto di mercato del portafoglio, attraverso il contratto future, si otterrà un guadagno da arbitraggio.

Facendo un esempio:

- Prezzo dell'indice pari a 21000
- Tasso d'interesse su attività prive di rischio pari a 3% annuo.
- Dividendi pari a 2% annuo.

Supponiamo che il prezzo di mercato del *future* sia 20500, cioè inferiore al suo *fair value* (date le condizioni fissate nell'esempio), pari a 21052,5; realizzando un arbitraggio riverse *cash and carry* si avrà:

- Vendita titoli a 21000
- Ricavo da impiego fondi in titoli privi di rischio, pari a 157,5
- Acquisto del *future* al prezzo di 20500
- Mancati dividendi per importo pari a 103,5

- Guadagno da arbitraggio pari a 554

In numeri:

$$+ 21000 + 157{,}5 - 20500 - 103{,}5 = 554$$

8.5. Le strategie attuabili con il Fib30

Già si è fatto cenno alla possibilità di eseguire arbitraggi, attraverso opportune strategie, sfruttando il contratto Fib30, tuttavia vi sono altre operatività che il Fib30 consente e sono le strategie di *hedging* (copertura) e le strategie di *trading* (speculazione).

Le strategie di arbitraggio: l'operatore assume contemporaneamente posizioni contrarie, una sul mercato a pronti ed una sul mercato a termine, sfruttando le inefficienze dei mercati, il tutto, eseguito, per ottenere piccole somme di guadagno, ma solitamente su grossi volumi. Elemento fondamentale di tale tecnica è l'assenza di rischio, in teoria; in pratica vi sono alcune difficoltà che rendono l'operato degli arbitraggisti tutt'altro che semplice. Un primo rischio che l'arbitraggista può incontrare è il rischio di esecuzione, infatti, nonostante l'operatore si avvalga della tecnica del *program trading*, che gli consente di negoziare notevoli quantità di titoli in un breve lasso di tempo, può verificarsi un mutamento delle condizioni di mercato, tra l'individuazione del possibile arbitraggio e la sua completa realizzazione; il mutamento di queste condizioni potrebbe annullare completamente la possibilità di fare profitti. Un secondo rischio a cui l'arbitraggista si espone è la non sempre agevole vendita del portafoglio allo scoperto oppure in alternativa il possibile elevato costo del prestito titoli per l'arbitraggista che non possiede il sottostante. Si può quindi dire che per realizzare con profitto un arbitraggio *riverse cash and carry*, la differenza

tra prezzo teorico del *future* e prezzo di mercato deve raggiungere un'ampiezza maggiore rispetto a quella necessaria per l'arbitraggio *cash and carry*. Alcuni studi empirici, hanno individuato nella difficoltà di vendita allo scoperto dei sottostanti, la maggior frequenza dei casi di sottovalutazione del prezzo del *future* rispetto a quelli di sopravvalutazione;[14] quindi la maggior frequenza di possibilità di attuare l'operazione *riverse cash and carry*, che, in seguito alle difficoltà citate, non si riesce a porre in essere e che provoca disallineamenti tra *future* e sottostante anche consistenti.

Le strategie di hedging: tali operazioni vengono realizzate per ridurre il rischio sistematico associato alla detenzione in portafoglio di titoli azionari. Questa strategie si basa sulla considerazione che l'investimento sul mercato azionario è soggetto a due tipologie di rischio, il rischio non sistematico (o intrinseco), che si può ridurre attraverso la diversificazione di portafoglio, ed il rischio sistematico (o di mercato), che colpisce il mercato in sé e quindi non può essere ridotto con la diversificazione.

Se un operatore detiene una posizione azionaria e prevede un ribasso delle quotazioni, anziché vendere l'intero portafoglio, può vendere un certo numero di contratti *future* sul sottostante. Se il ribasso si verifica realmente, le perdite subite sul suo portafoglio, saranno compensate dai guadagni ottenuti dal contratto *future*; qualora il ribasso non si verificasse e la situazione rimanesse invariata l'operatore dovrebbe sopportare solo i costi relativi alla creazione della posizione *short* sul *future*; invece, se si verificasse un rialzo dei corsi azionari l'operatore subirebbe una perdita sul contratto *future* compensata da un incremento di valore del proprio portafoglio titoli, oppure direttamente un guadagno in caso di vendita di questo.

Nell'attuare tale tecnica vi sono alcune difficoltà di particolare rilevanza, che consentono a pochi operatori di attuare veramente l'operazione raggiungendo

[14] Si veda: D.MODEST-M. SUNDARESAN, 1993 pp.15-42, *The relationship between spot and futures prices in stock index futures markets: some preliminary evidence*, in "Journal of Futures Markets", n. 3.

l'obiettivo sperato (la copertura). Tali difficoltà riguardano la correlazione fra le oscillazioni del portafoglio *cash* e le oscillazioni del contratto *future*, infatti l'indice sottostante al *future* su esso, non è necessariamente correlato con ogni forma di portafoglio che comprenda le azioni dell'indice stesso. Per superare quest'ostacolo vi sono varie soluzioni, una consiste nell'acquistare un portafoglio titoli che rappresenti perfettamente l'indice sottostante al *future* (*full replication*), oppure si possono acquistare i titoli maggiormente rappresentativi dell'indice, in modo da avere in portafoglio i titoli che più influenzano le variazioni dell'indice (ottimizzazione di portafoglio); si può anche calcolare il rapporto di copertura, cioè il numero di contratti *future* da vendere per realizzare al meglio la copertura in relazione alle caratteristiche del proprio portafoglio.

Il rapporto di copertura è dato dalla seguente formula:

$$R_c = (CTV_p * \beta_p) / CTV_f$$

Il β è il coefficiente che misura la sensibilità del titolo alle variazioni del mercato. La sua formula è:

$$\beta = Cov (R_i , R_m) / Var (R_m)$$

Dove:

R_i: rendimento del titolo

R_m: rendimento del portafoglio di mercato

Cov: covarianza

Var: varianza

Facendo un esempio: un operatore possiede un portafoglio azionario pari ad un controvalore di 200.000 €, con un β medio di portafoglio pari a 0,75, prevedendo un calo delle quotazioni sul mercato, vuole attuare un'operazione di copertura mediante l'utilizzo di contratti *future*, dovrà quindi determinare quanti

contratti vendere; ipotizzando una quotazione dell'indice pari a 21500, si otterrà il numero di contratti con il seguente calcolo:

$$R_c = (200000 * 0{,}75) / 21500 = 6{,}978 \ \ contratti\ future$$

Per tutelare il suo investimento azionario, da eventuali ribassi, l'operatore dovrà vendere sette contratti future (acquistare sette contratti Fib30 *short*); quando l'operatore riterrà che la tendenza a ribasso sarà finita, chiuderà la propria posizione in *future*, acquistando sette contratti *future* (acquistando sette contratti Fib30 *long*).

Ipotizzando quindi un ribasso nel nostro esempio, avremo che il controvalore del portafoglio azionario dell'operatore ammonterà a 185000 ed la quotazione del *future* sarà pari a 19500; il guadagno dovuto alla posizione di copertura, che andrà a coprire la perdita subita sul portafoglio titoli, sarà dato da:

Posizione in titoli:

185000 – 20000 = - 15000

Posizione in *future*:

21500 * 7 * 5 = 752.500 Vendita del Fib30

19500 * 7 * 5 = 682.500 Acquisto del Fib30

Tot: 752.500 – 682500 = 70000

Risultato da strategia di copertura:

70000 – 15000 = 55000 Guadagno

In tale esempio l'operazione di copertura si è addirittura chiusa con un guadagno per l'operatore, questo si è verificato per due motivi uno legato al fatto che il rapporto di copertura risultava essere un po' inferiore al valore sette (numero di contratti *future* acquistati), il secondo legato al rapporto che lega l'indice con il portafoglio che l'operatore possiede.

L'inconveniente in cui si può incorrere utilizzando questa tecnica è dovuto al ottenimento di un rapporto di copertura come numero non intero, o comunque non prossimo all'intero; in questo modo si attuerà una strategia che non permetterà una perfetta copertura del proprio portafoglio. Per ovviare a tale problema si deve costituire un portafoglio che sia raffrontabile con l'indice, in modo da ottenere, in caso di necessità, una copertura adeguata sfruttando i contratti *future*.

Strategie di speculazione: vengono utilizzati i contratti *future* per fare speculazioni borsistiche, poiché questi garantiscono margini di guadagno notevolmente superiori a quelli ottenibili sul mercato dei valori mobiliari; il rovescio della medaglia è che i rischi che si sostengono acquistando/vendendo contratti *future*, con queste finalità, sono decisamente maggiori rispetto a quelli a cui si va in contro acquistando titoli.

I maggiori rendimenti derivano da diversi fattori:

- Meccanismo del margine iniziale, che genera un effetto leva.
- Minori costi di transazione presenti sul mercato *future* rispetto a quello a pronti.
- Notevole flessibilità dello strumento; una qualsiasi strategia di *trading* che sfrutta *future*, può essere ribaltata velocemente a seconda delle mutazioni delle condizioni di mercato. Infatti da una posizione *long* si può passare ad una *short* con facilità, cosa questa non vera nel caso dei valori mobiliari i quali non possono essere venduti allo scoperto facilmente.

CAPITOLO 9

Le modifiche apportate a SUM

Lo scopo della tesi è stato quello di introdurre uno strumento derivato nella simulazione di mercato, ed un operatore che potesse agire contemporaneamente sul mercato del derivato e sul mercato del sottostante mantenendoli in equilibrio, allineando i loro corsi. L'idea di fondo è stata quella di verificare se introducendo il *future* sull'indice di borsa questo, attraverso l'azione di un arbitraggista, rimanesse allineato con il suo valore teorico e quindi con il suo sottostante. Inoltre si era inizialmente pensato di sperimentare tale situazione con la convinzione che l'operato dell'arbitraggista non fosse sufficiente a riallineare i due mercati, spinti soprattutto da quanto rilevato nella realtà italiana; infatti l'idea sostenuta da alcuni operatori del settore (in Italia), è che il mantenimento della coerenza tra le quotazioni del *future* rispetto al sottostante è giustificata non tanto dalle operazioni di arbitraggio che, a loro vedere, oltre ad essere rare, non sono quasi mai eseguite da specialisti allocati sul nostro territorio nazionale ma da operatori stranieri, bensì dalla diffusa consapevolezza tra la maggioranza degli operatori del fatto che la quotazione del *future* debba essere collegata a quella del suo sottostante e che pertanto le quotazioni dell'uno non possano scostarsi di tanto da quelle dell'altro. Inoltre a parere degli operatori contattati, grosse operazioni di arbitraggio desterebbero l'attenzione della Consob e per tale motivo non vengono eseguite. Sono invece considerati possibili fulminei arbitraggi sul Fib30 l'ultimo giorno di contrattazione, nel quale per definizione, il prezzo del Fib30 ed il suo sottostante devono avere lo stesso valore; tale forma di operatività è resa anche possibile dal

fatto che gli aggiustamenti di prezzo del Fib30 non sono di un tick solo, ma di cinque alla volta.

La fase di sperimentazione ha poi smentito quest'idea di fondo, in quanto i risultati emersi da sperimentazione di mercato simulato in presenza di operatori arbitraggisti, sono stati di perfetto allineamento del *future* con il suo sottostante.

Le modifiche che sono state necessarie al fine di poter implementare l'agente arbitraggista ed il *future* sull'indice, sono state numerose ed hanno toccato tutte le parti del programma. L'elenco dei file che sono stati modificati e dei file che sono stati creati, è presente in appendice.

9.1.L'introduzione del book multiplo

Una delle prime modifiche che si sono rivelate necessarie in SUM è stato l'introdurre la possibilità di trattare sul mercato più titoli. Questo nella versione precedente non era possibile in quanto era presente un unico titolo sul mercato e tutti gli agenti indirizzavano le loro azioni su di esso.

L'introduzione di più titoli oltre che elemento necessario per creare un indice, è stato anche un elemento che è servito a rendere i fenomeni delle bolle speculative e dei *crash* meno frequenti e meno consistenti; ciò non è comunque servito ad eliminarne la presenza. Il fatto che le bolle ed i *crash* diminuissero per frequenza e per consistenza era stato previsto, in quanto si era pensato che essendoci più possibilità di scelta di investimento, l'attenzione degli agenti, si sarebbe orientata su tutti i titoli e ciò avrebbe contribuito a determinare una minore pressione della domanda e dell'offerta di titoli.

La struttura del book presente nella precedente versione di SUM non è stata toccata, ma è semplicemente stata duplicata, dando la possibilità di generare più titoli nello stesso mercato, che vengono tutti trattati in book aventi le stesse caratteristiche. Parlando in termini un più tecnici, si è presa la "*classe book"* e si

sono create delle "*instance"* di questa; attualmente al posto della classe *"theBook"*, presente nelle precedenti versioni, si sono introdotte le classi *"aBook"* e *"oneBook"*, utilizzate dai vari agenti a seconda del loro "profilo caratteriale". Quelle tipologie di agenti che richiedono di seguire un book nello specifico e di mantenere memoria di tale utilizzo, fanno capo alla classe *"oneBook"*, mentre quegli agenti che non necessitano di avere memoria del book su cui operano utilizzeranno la classe *"aBook"*. L'utente può attraverso la funzione interfaccia che *"Swarm"* permette di creare, attraverso la classe "*Observer",* decidere all'inizio della simulazione quanti titoli trattare, inserendo un numero nella casella *"BookNumber"*.

Questa modifica concettualmente, relativamente semplice ha però creato non pochi problemi relativi a tutti gli elementi che ad essa sono collegati; per comprendere meglio è sufficiente pensare che il cuore di SUM è il book, modificando esso si è dovuti andare a modificare tutti gli altri elementi del programma, *"Observer"*, *"Model"*, *"makefile"* in più parti ed inoltre tutti gli agenti già presenti in SUM. Si è dovuto modificare il calcolo della ricchezza degli agenti il quale era impostato su un sistema che ne permetteva la valutazione basandosi sulla richiesta del prezzo medio del titolo (l'unico presente nelle precedenti versioni), il *"meanPrice",* tuttavia adesso in SUM ci sono tanti *"meanPrice"* quanti book nel mercato; così è stato introdotto un metodo per calcolare la ricchezza che facesse prima la richiesta del titolo che l'agente ha acquistato/venduto, usando la matrice *bookArrayIndex* (in realtà in SUM viene richiesto il book su cui è stato chiuso il contratto, o meglio, l'indirizzo di questo book) e poi del relativo *"meanPrice",* che a questo punto viene utilizzato per considerare la posizione della ricchezza dell'agente. Per alcuni agenti questo non è stato possibile, poiché facenti uso sempre di un determinato book, allora si è costruito un sistema solo per il calcolo della ricchezza di queste tipologie di agenti.

Anche l'elemento che definisce la sequenza temporale di tutti gli eventi che si verificano in SUM, definito *"scheduler"*, e facente parte della classe

"Model", è stato modificato. E' stato necessario far sì che nella successione temporale delle azioni fossero ricompresse tutte le richieste di compravendita relative a tutti i book in quel momento attivi nel mercato; questa modifica ha sostanzialmente richiesto la necessità di far sì che fossero assegnati dei corretti indirizzi per le richieste di informazione rappresentanti le cosiddette chiamate delle variabili.

9.2.Le modifiche agli altri agenti del mercato

Tutti gli agenti precedentemente creati in SUM hanno richiesto delle modifiche per renderli compatibili con il sistema del "*Multi Book*". Il problema fondamentale che si è dovuto risolvere è stato quello di attuare degli opportuni aggiustamenti del codice esistente, al fine di permettere che tutte le tipologie di agenti avessero la possibilità di operare su più book, in modo da orientare le proprie scelte di compravendita sull'intero mercato.

Per alcuni agenti, a causa delle caratteristiche che presentano, è stato sufficiente far sì che prima di operare si chiedessero su quale titolo farlo e successivamente tutti i passaggi informatici relativi al loro comportamento sono stati lasciati inalterati. La scelta del book su cui operare viene presa secondo un processo casuale, successivamente, le relative operazioni sono quelle caratterizzanti la tipologia di agente. E' questo il caso del *RandomAgent*, dello *stopLossAgent*, del *marketImitatingAgent*. Stessa situazione vale per un operatore che si basa su previsioni generate da un apposito soggetto generatore di previsioni, che potrebbe trovare riscontro nella realtà con una Società di rating che, emettendo giudizi su determinati titoli, ne influenza l'andamento; l'agente in questione è l'*aNNForecastAppAgent*, al quale, a differenza di quanto pensato inizialmente in fase di modifica, non viene assegnato un book su cui operare;

questo perché il suo previsore (la Società di rating della realtà) forma previsioni sempre su uno stesso titolo, però si è deciso di dare a tale previsore la possibilità di fare previsioni su tutti i titoli del mercato, compreso il *future*. Seguendo questa idea si è assegnato un soggetto previsore per ogni book, mentre gli agenti *aNNForecastAppAgent* possono decidere, secondo un processo casuale, quale book utilizzare per le loro operazioni e per l'inserimento della loro posizione nel book e dei relativi prezzi utilizzeranno i consigli del previsore. Questo ha giustificazione in quanto non vi è nessun motivo per cui gli agenti che operano sulla base di previsioni siano vincolati all'operare su un solo book, infatti, nella realtà, si è liberi, ad esempio, di osservare i giudizi espressi da una società di rating relativi a più titoli ed operare su di essi senza prediligerne uno in particolare.

Altri agenti, invece presentando delle caratteristiche più complesse hanno richiesto piccole modifiche della loro struttura, le quali hanno comportato l'imposizione dell'esecuzione delle loro operazioni sempre su un unico book; questo poiché a causa delle loro caratteristiche peculiari, il loro comportamento è strettamente legato ad azioni svolte nei confronti di un book ben preciso. E' il caso del *BPCTAgent*, in tutte le sue tipologie, il quale segue le previsioni di una rete neurale ed opera relativamente a queste; essendo le previsioni svolte di volta in volta l'agente sarà costretto ad operare sempre sullo stesso book. Tuttavia per non creare delle asimmetrie nell'andamento dei vari titoli facendo operare il *BPCTagent* su un book solo, si è impostato un comando che assegna ad ogni book un *BPCTagent*, purché questi siano in numero multiplo dei book. Anche lo *stopLossAgent*, in una delle due tipologie possibili, quella che ha memoria del passato, presenta caratteristiche che lo legano ad un book nello specifico ed allora anche per tale agente si è assegnato un book predefinito.

9.3. L'introduzione del future

Svolte tutte queste modifiche si è potuto procedere con la creazione del primo dei due elementi necessari per la serie di esperimenti che sono l'obiettivo della tesi, il *future* sull'indice di borsa.

Prima di tutto si è fatta una distinzione in SUM tra i titoli veri e propri ed il *future*; tutte e due le tipologie di contratti sono trattate sulla stessa tipologia di classe, il book; è necessario porre una distinzione che si rivela utile quando si implementa il calcolo dell'indice, infatti, essendo tutti i book uguali si è dovuto trovare un modo per differenziare il book del *future* dagli altri book, per evitare che il calcolatore dell'indice comprendesse nel calcolo anche il future. Allora si è assegnato l'ultimo book come fisso per le contrattazioni sul *future*; facendo in questo modo il *future* è sempre trattato nell'ultimo book e così si è trovato un modo semplice per far sì che il *future* non sia erroneamente compreso nel calcolo dell'indice; infatti, così facendo, esso viene sistematicamente escluso dal calcolo.

Si è così creato un metodo che calcolasse la variabile indice sui titoli presenti in SUM, denominata *IndexValue*, calcolata dal metodo *IndexCalculator*; tale variabile opera il calcolo di un valor medio. Infatti, prende i prezzi dei titoli in un dato istante, ne fa la somma e poi li divide per il numero di titoli. Questa scelta semplificata di calcolo dell'indice fa capo a tutte le motivazioni citate sopra relative alle semplificazioni che in parte sono necessarie ed in parte sono volute, ma comunque sono tutte giustificabili. Molti indici azionari, tra cui quello relativo ai trenta titoli a maggior capitalizzazione della borsa italiana, il Mib30, assegnano un peso ad ogni titolo che li compone, questo sulla base di coefficienti che si originano sulla base del livello di capitalizzazione di ogni titolo considerato. In SUM non vi è un valore di capitalizzazione dei titoli, perché non vi è nemmeno la determinazione di un flottante, quindi per ora non ha senso inserire dei pesi da assegnare ai titoli per rendere questi maggiormente incidenti sul valore dell'indice. Altra caratteristica del Mib30, ma non solo di questo, è la

ponderazione della media per i prezzi; anche questo in SUM non avviene, poiché ci si è rifatti alla costruzione grezza di un indice, che non prevede la ponderazione per i prezzi. Relativamente agli indici grezzi si possono citare i due indici più famosi e più vecchi della storia di borsa il DJIA (*Dow Jones Industrial Average*) ed il DJTA (*Dow Jones Transportation Average*), anch'essi costruiti senza ponderazione per i prezzi.

Infine, calcolato l'indice è possibile costruire il *future* che abbia come sottostante l'indice stesso e che possa essere trattato liberamente in SUM. Per ottenere questo si fa semplicemente uso della classe book; calcolato il numero di book, l'ultimo in ordine cronologico, rappresenta quello del *future* il suo valore e quello dato dall'incontro della domanda e dell'offerta ed il suo valore teorico viene calcolato a parte, secondo una metodologia che sotto andrà descritta.

Tutti gli agenti operanti in SUM possono acquistare/vendere il *future*, nessuno escluso; tale decisione è stata presa per il fatto che l'utilizzo degli strumenti derivati, di qualsiasi tipo essi siano, si è notevolmente diffuso tra gli investitori nella realtà, per tale motivo anche in SUM ogni investitore può utilizzare come strumento per le proprie operazioni il *future*. Anche gli *avatarAgent*, particolare tipologia di agenti le cui decisioni sono prese da umani che attraverso gli *avatar*,[15] possono interagire ed operare in SUM, hanno la possibilità di operare sul *future*. Tuttavia il fatto che tutti gli agenti possano operare sul future, comporta che le quotazioni di questo siano libere di oscillare sulla base della domanda e dell'offerta; questo si verifica in SUM, perché nessun agente (a parte l'*arbitrageurAgent*), ha cognizione di cosa sia un future e quindi opera su esso come se fosse un semplice titolo. I prezzi inseriti dagli agenti nel book del *future*, sono quindi incoerenti rispetto alla relazione che lega *future* e suo sottostante; soltanto l'*arbitrageurAgent* ha cognizione di ciò e proprio grazie

[15] Il termine "avatar", informaticamente ha il significato di segno grafico rappresentante un utente in gruppi di discussione su internet o in uno spazio virtuale. Il termine ha origine dalla mitologia induista ed il suo vero significato è "incarnazione della divinità". Qui viene utilizzato per impersonare in SUM un utente che interagisca con il modello operandovi, al pari di ogni altro agente virtuale.

a tale cognizione riesce, operando su entrambi i mercati, a far sì che siano allineati;tale aspetto verrà analizzato più approfonditamente nel capitolo dedicato agli esperimenti.

9.3.1.La determinazione del prezzo del future in SUM

Nella nostra simulazione di mercato la determinazione del prezzo teorico del *future* sull'indice, non è funzione, come invece accade nella realtà, di più variabili, quali sono, il prezzo del sottostante, il tasso di interesse, il tempo rimanente alla scadenza del contratto ed i dividendi stimati relativi ai titoli dell'indice sottostante; esso è funzione unicamente del valore e dell'andamento dell'attività sottostante, che nel nostro caso è l'indice sui titoli presenti sul mercato simulato.

Ora, tutto ciò non pecca di realismo dal punto di vista della dinamica dei prezzi, secondo gli autori,[16] per una serie di considerazioni che ora saranno chiarite.

È indubbio che la scelta operata dagli autori sia una semplificazione della realtà, ma tale semplificazione è voluta, poiché si è cercato di scegliere un modo di rappresentare la realtà che non comportasse troppe modifiche contemporaneamente, le quali avrebbero inevitabilmente minato la buona riuscita dell'esperimento, in quanto avrebbero richiesto numerose e complesse modifiche di quella che è la struttura attuale del progetto SUM. Inoltre, come modus operandi si sta cercando, con la supervisione del prof. Terna, ideatore del progetto SUM, di fissare e testare a lungo ogni modifica, partendo dal presupposto che ogni cosa va prima fatta nel modo più semplice e poi successivamente si può pensare di renderla più complessa; tutto questo per avere delle basi solide e stabili (informaticamente) su cui attuare le modifiche che

[16] Andrea Vanara ed Alessandro Cappellini

verranno introdotte in futuro e che renderanno il modello sempre più aderente alla realtà e fonte di sempre nuove conferme, spiegazioni, spunti, ecc. utili a comprendere meglio la realtà che ci circonda, ad indirizzare lo studio dei ricercatori. Detto questo ben si comprende come l'introdurre troppi elementi nuovi in SUM avrebbe potuto comportare dei problemi attuativi notevoli, nonché l'impossibilità di comprendere cosa nel mercato simulato avviene, facendo un passo alla volta; seguendo la condotta opposta a quella qui abbracciata, si sarebbe sicuramente creato un sistema più aderente alla realtà (forse sarebbe meglio dire all'apparenza della realtà), ma per ciò che riguarda i risultati sarebbe sicuramente stata difficile l'interpretazione. Si sarebbero infatti ottenuti risultati facenti capo all'azione di più variabili di "nuovo inserimento" e non si sarebbe capito a quale delle tante modifiche imputarli. Con ciò si vuole dire che si può incrementare di quanto si vuole la complessità del modello, ma volendolo utilizzare per comprendere i fenomeni visti nella realtà è necessario fare un passo alla volta e fare sperimentazione ogni volta che si implementano nuovi elementi nel modello.

9.3.2. La formazione del prezzo teorico del future in SUM

Il modello SUM come detto più volte, è una semplificazione dei meccanismi della realtà, questo perché l'interesse dell'esperimento sta nel cogliere l'essenza delle caratteristiche che animano i fenomeni studiati e non nel considerare particolari considerati irrilevanti (almeno per ora); per tale motivo la determinazione del prezzo teorico del *future* in SUM è decisamente semplificata e quindi non fa eccezione a quanto appena detto. Tuttavia le semplificazioni fatte, che ora saranno elencate, sono giustificabili e pertanto hanno la loro ragion d'essere.

In primo luogo in SUM non si è ancora voluto inserire la variabile tasso d'interesse, questo perché la disponibilità liquida e di titoli è illimitata per ogni

agente, cioè ogni agente nel dover scegliere se comprare o vendere gli strumenti finanziari disponibili (attualmente titoli e *future* sull'indice) non deve prima analizzare il proprio portafoglio titoli o la propria disponibilità liquida, ma può liberamente decidere senza vincoli patrimoniali. Tutto ciò comporta che un operatore arbitraggista, investendo in un contratto *future* piuttosto che direttamente sul sottostante, non si troverà nel caso di operazione *riverse cash and carry* a dover investire in titoli privi di rischio (ad es. Bot) il controvalore del contratto future decurtato della somma relativa ai margini di garanzia versati alla Cassa di Compensazione e Garanzia e nemmeno l'operatore arbitraggista, in caso di operazione *cash and carry*, si troverà a dover chiedere denaro a prestito per l'acquisto dei titoli sottostanti e quindi a dover corrispondere per tale prestito il relativo tasso d'interesse.

Non sono inoltre presenti i margini di garanzia e quindi nemmeno i margini di mantenimento, giornalieri ed infragiornalieri. Così, in SUM non vi è un rischio di controparte, rappresentato dalla possibile insolvenza di una delle parti contrattuali, per quanto spiegato sopra. Per tale motivo non vi è nessuna possibilità, che un operatore esponendosi con un contratto *future* si ritrovi a scadenza del contratto una controparte inadempiente. Si è così deciso che l'acquisto del *future* riguardi il valore totale e non una parte di esso inizialmente, seguita da varie liquidazioni giornaliere od infragiornaliere, e poi il rimanente a scadenza come accade nella realtà.

Inoltre il meccanismo dei margini di garanzia e gli aggiornamenti che ad essi si fanno sulla base delle liquidazioni giornaliere di tutti i contratti sul *future*, avrebbero senso solo se fosse inserita nel progetto SUM la Cassa di Compensazione e Garanzia che attualmente manca, poiché non ve n'è la necessità. Infatti, non vi è nessun motivo che vi sia un ente garante della liquidità del mercato e dell'eliminazione del rischio di controparte. Questo perché in SUM non vi sono limitazioni di portafoglio per gli agenti e nemmeno di liquidità, ciò significa che ogni agente può comprare/vendere quante azioni vuole e quindi il

contratto *future* non racchiuderà in sé il rischio di inadempienza della controparte.

Nemmeno è considerata la variabile dividendi, poiché non vi sono società nello scenario di SUM, in quanto esso è un "semplice" mercato, un luogo di contrattazioni per adesso scollegato da un contesto sociale. Quindi non vi è la presentazione dei bilanci agli investitori delle società che quotano i titoli sul mercato SUM e nemmeno vi è la presentazione di stime di utile e di eventuali dividendi da distribuirsi. Inoltre, a ben vedere, tra le variabili che concorrono alla formazione del prezzo teorico del *future* nella realtà, i dividendi sono una variabile che si presta ad interpretazioni soggettive da parte degli operatori, poiché nello stimare i dividendi ci si basa su dati di bilancio passati e si cerca di dare una stima di quello che sarà il futuro andamento della società, andamento la cui determinazione è influenzata da una moltitudine di variabili, esogene ed endogene, che mutano continuamente; tra tutte le variabili mancanti in SUM, è quella che probabilmente richiederebbe la presenza di molti elementi, che ancora non ci sono, per poter essere implementata.

Altra semplificazione risultata necessaria è la quotazione di un'unica scadenza del contratto *future*, invece che di tre come nella realtà. Questo per non appesantire la struttura di calcolo del programma e per non fare troppe modifiche tutte insieme, senza dare ad esse un ragionevole periodo di assestamento e di sperimentazione. Facendo ciò nemmeno si è considerato il trascorrere dei giorni per il contratto, poiché ciò avrebbe creato dei problemi per l'agente arbitraggista. Infatti, quest'ultimo si sarebbe trovato impossibilitato a svolgere le sue operazioni di arbitraggio nei giorni di transizione tra un contratto e l'altro, giorni nei quali un contratto scade ma l'altro non c'è ancora poiché prenderà vita dal giorno successivo; in tale condizioni si avrebbe avuto un giorno in cui il future sarebbe stato libero di oscillare a suo piacimento senza che nessuno ne controllasse l'allineamento con il suo sottostante. Così la scadenza trattata del *future* è una sola ed in ogni giorno di contrattazione si quota un nuovo *future* sull'indice che scade il giorno stesso (alla fine della giornata di borsa), oppure è

equivalente pensare che il contratto abbia origine oggi e scadenza a tre mesi ed ogni giorno la scadenza sia sempre considerata a tre mesi. Questo comporta che nemmeno la variabile tempo influisca sull'andamento del prezzo teorico del *future*. Il non considerare il fattore tempo permette anche di non dover inserire un'ulteriore modifica del progetto originario e cioè la continuità del portafoglio titoli e della posizione di liquidità da un giorno all'altro.

Inoltre il contratto future è valutato di pari importo rispetto al valore dell'indice, questo significa che se l'indice vale 5, il contratto future, purché vi sia qualcuno che lo acquista/vende per tale prezzo, vale 5; questo discorda con la realtà in quanto il controvalore dei contratti future sugli indici (*Indexed Future*), viene determinato moltiplicando il valore dell'indice per una certa cifra espressa in termini monetari; il risultato di tale calcolo, produce contratti per importi significativamente alti. Tale tipologia di calcolo è stata adottata dagli enti responsabili del controllo sui mercati finanziari al fine di scoraggiare gli investitori più piccoli, considerati meno esperti, dall'utilizzare gli strumenti derivati, che si caratterizzano per un notevole profilo di rischio.

Fatte queste premesse si comprende come a concorrere alla formazione del prezzo teorico del *future* sia soltanto il prezzo del sottostante e cioè l'indice di riferimento; infatti, nella realtà per definizione, il prezzo teorico del *future* deve coincidere con il valore del sottostante nel giorno di scadenza del contratto *future*. Le semplificazioni fatte, comunque preferibili al fine di snellire la mole di calcoli della simulazione, quindi, non concorrono secondo gli autori a rendere il modello privo di realismo, visto le giustificazioni portate sopra e visto che lo scopo che l'esperimento SUM si prefigge è quello di cogliere da un punto di vista aggregato la dinamica della formazione dei prezzi in un mercato di borsa e il comportamento degli agenti che operano in esso.

9.4. L'introduzione dell'agente arbitraggista

Creati l'indice ed il *future* si è passati alla creazione dell'agente arbitraggista, denominato *arbitrageurAgent*, il quale, facendo determinati valutazioni, si avvale dello strumento indice per prendere decisioni operative sul contratto *future* e sui titoli sottostanti allo stesso contratto derivato.

Come nella realtà per l'arbitraggista, l'*arbitrageurAgent* cerca di operare individuando momentanee discrepanze di allineamento tra quotazione del *future* e valore teorico di questo; attuando poi opportune operazioni di segno opposto su mercati differenti, in questo caso mercato a pronti e mercato a termine, lucra profitti privi di rischio.

In SUM i passaggi operativi che vengono svolti dal programma per impostare l'operatività dell'*arbitrageurAgent*, non sono molto differenti da quanto si è appena detto a proposito degli arbitraggisti nella realtà e questo sarà subito dimostrato.

Quando lo "*scheduler"* interpella l'*arbitrageurAgent* perché è il suo turno di operare, questo importerà la variabile *indexValue*, la confronterà con il prezzo a cui è stato concluso l'ultimo contratto sul book del *future*, se non c'è nessuno scostamento, non opera e passa il turno, invece se individua uno scostamento prende in considerazione la possibilità di arbitraggio, vincolata al verificarsi di altre condizioni ottimali. Supposto che vi sia lo scostamento, l'*arbitrageurAgent* deve prima decidere quale tipologia di operazione di arbitraggio effettuare; allora valuterà che se il *future* è sottovalutato rispetto all'*indexValue* dovrà porsi come acquirente di *future* e venditore di titoli, mentre qualora il *future* fosse sopravvalutato rispetto al suo valore teorico egli dovrà porsi come acquirente di titoli e venditore di *future*. Successivamente all'individuazione della tipologia di arbitraggio egli dovrà verificare che altre condizioni sussistano, ed in particolare, dovrà verificare di poter rientrare dei costi dell'operazione e quindi che vi sia adeguata controparte. Tali due valutazione vengono svolte in un solo momento,

in quanto l'*arbitrageurAgent* avendo individuato quale tipologia di arbitraggio eseguire, va a controllare che le sue controparti esistano nel book. Se la controparte esiste l'operazione viene inserita e mandata "simultaneamente"[17] in esecuzione, così l'operazione di arbitraggio è conclusa; qualora vi siano più *arbitrageurAgent*, tale ciclo di calcolo sarà ripetuto per ognuno di essi.

Il controllo del fatto che l'operazione di arbitraggio garantisca un guadagno per l'arbitraggista, garantendogli quindi la copertura dei costi, è fatto secondo due modalità: in una l'arbitraggista controlla che il disallineamento tra *future* e valore teorico di questo (quindi la variazione tra i due valori) sia strettamente maggiore ad un valore percentuale, nell'altra valuta che il disallineamento sia strettamente maggiore di un valore fisso. L'indicazione sia del valore percentuale, sia del valore fisso possono essere modificate dall'utente stesso, il quale può innanzitutto scegliere la tipologia di costi da assegnare alle operazioni e poi ne può scegliere l'entità, modificando a proprio piacimento il valore da assegnare, in modo da ipotizzare varie condizioni di mercato possibili. Inoltre nella realtà, a sentire quanto detto dagli operatori, vi sono entrambe le tipologie di costi, tuttavia non si è riusciti a comprendere quale delle due prevalga sull'altra e quindi si è deciso di inserire tutte e due le possibilità. In futuro quando si sarà individuata con più precisione la condizione di tali costi si potrà inserire definitivamente una delle due possibilità e se fosse necessario, si inserirà una versione che comprenda una parte di costi in termini percentuali ed una parte di costi in termini fissi, con la dovuta proporzione tra le due.

L'*arbitrageurAgent* ha la possibilità di operare ogni volta dopo l'operazione di un qualsiasi altro agente, compreso l'*avatarAgent*, questo perché potenzialmente ogni operazione di un agente può in teoria portare fuori allineamento il *future* con il suo valore teorico; tale impostazione deriva dal fatto che nella realtà l'arbitraggista opera in qualsiasi momento le condizioni lo

[17] Ovviamente da un punto di vista informatico non è simultanea l'operazione, perché viene svolta un'azione alla volta; però da un punto di vista dell'operatività di mercato si verifica che l'arbitraggista inserisca contemporaneamente i due ordini.

permettano, infatti non vi è giustificazione concettuale che spinga un arbitraggista a non operare quando se ne presentano le opportunità. Per meglio comprendere la struttura delle chiamate ad operare degli arbitraggisti si veda lo schema sotto, che indica in sequenza temporale come le chiamate dello "*scheduler"* avvengano:

- Un agente qualsiasi (tranne l'arbitraggista).
- Un arbitraggista.
- Un avatar.
- Un arbitraggista.
- Un agente qualsiasi (tranne l'arbitraggista).
- ……

Una forzatura invece poco giustificabile, se non asserendo la necessità di semplificare la struttura dell'operatività dell'arbitraggista in SUM, per meglio comprendere l'esito delle modifiche, è il fatto che quando la posizione di arbitraggio è stata individuata e l'*arbitrageurAgent* ha fatto le valutazione che hanno confermato la possibilità di eseguire una delle due tipologie di arbitraggio, questa sicuramente va in porto, procurando all'*arbitrageurAgent* il guadagno sperato. Ciò si discosta dalla realtà nella quale dall'individuazione della possibilità di arbitraggio alla completa realizzazione di questa con l'esecuzione delle operazioni su entrambi i mercati, trascorrono preziosi secondi, che possono mutare le condizioni inizialmente valutate e garantire un perdita sicura invece di un guadagno sicuro.

CAPITOLO 10

Esperimenti

Come detto precedentemente lo scopo che ci si è posti è quello di verificare in sede sperimentale se l'operato di agenti arbitraggisti sia sufficiente a mantenere coerenza tra quotazione del *future* sull'indice e l'indice sottostante.

Per la verifica di tale aspetto sono stati eseguiti vari esperimenti, ognuno dei quali mirante ad ottenere determinate conferme, tutte comunque finalizzate a poter offrire una verifica di quanto esposto sopra.

Il lavoro di sperimentazione è articolato in due sessioni di esperimenti: una, quella iniziale, vuole valutare come il mercato si comporti quando in esso viene trattato un future sull'indice, ma nessun operatore abbia cognizione di ciò che esso rappresenti; la seconda e più importante, si prefigge di valutare la bontà dell'operato dell'arbitraggista ai fini dell'equilibrio tra il mercato a pronti ed il mercato a termine. Nella seconda sessione di sperimentazioni sono state fatte più prove volte a valutare se l'azione dell'*arbitrageurAgent* fosse sufficiente a riequilibrare le quotazioni indipendentemente dalle caratteristiche degli operatori che agivano sul mercato. Gli esperimenti sono in totale sei, due dei quali sono svolti per verificare se senza la presenza dell'arbitraggista, vi sia allineamento tra *future* e suo valore teorico, i rimanenti quattro, sono svolti per verificare l'operato dell'arbitarggista in varie condizioni di mercato. Nei vari esperimenti si è anche tenuto in considerazione di impostare in alcuni casi, popolazioni di agenti imitatori di mercato, in altri, popolazioni formate per la maggior parte da agenti casuali (*randomAgent*); questo perché gli agenti imitatori di mercato, rispetto a quanto fanno gli agenti casuali, tendono ad incrementare il formarsi delle bolle speculative. Impostando le popolazioni secondo diversi criteri, si sono analizzate varie condizioni di mercato.

10.1 Il mercato con il future, ma senza arbitraggista

10.1.1 Risultati attesi

Ciò che ci si aspetta da tale fase di sperimentazione è che il *future* sia libero di oscillare a suo piacimento, sulla base della spinta della domanda e dell'offerta. Ci si aspetta questo risultato in quanto il *future* sarà trattato come un qualsiasi titolo dagli agenti che operano sul mercato e quindi non vi è nessun motivo che esso mantenga un allineamento con quello che dovrebbe essere il suo valore teorico (rappresentato in SUM dal valore dell'indice sottostante). L'*arbitrageurAgent* non è in questa fase presente nelle contrattazioni, quindi non partecipa alla formazione dei prezzi nel modello; mentre tutti gli altri operatori del mercato simulato potranno operare liberamente sul *future*.

Non vi sarebbe nessuna giustificazione plausibile al fatto che il *future* fosse allineato con il suo sottostante, salvo il fatto di errori nella strutturazione del codice di programmazione; se si verificasse infatti che il *future* fosse allineato con il suo sottostante anche senza la presenza di operatori arbitraggisti, vorrebbe dire che si sono inserite delle righe di comando nel programma che inducono l'operatività degli agenti al mantenimento dell'allineamento.

Nel caso di SUM, nella attuale fase di programmazione si è tenuto ben distinto il metodo di calcolo *indexCalculator*, dal codice di tutti gli altri agenti; questo garantisce che non vi sia nessuna forma di interazione tra i vari oggetti (in questo caso solo gli agenti) se non con i metodi che essi utilizzano. L'unico agente che importa la variabile che rappresenta il valore teorico del future, rappresentata dal valore *indexValue* e calcolata dal metodo *indexCalculator* è l'*arbitrageurAgent*, il quale la usa per le sue valutazioni operative. Quindi si può affermare che l'unico agente che abbia cognizione di ciò che rappresenta il *future* è l'arbitraggista.

10.1.2 Gli esperimenti senza arbitraggista

Questa sessione di esperimenti prevede di verificare due casi di mercato per valutare se effettivamente, senza l'operato dell'arbitraggista, la quotazione del future sia indipendente da quello che è l'andamento del suo sottostante indicato dal valore i*ndexValue*.

Il primo esperimento vede l'impiego delle tipologie di agenti che meno abbiano un effetto trainante sulle dinamiche di prezzo di SUM; in particolare saranno esclusi da tale esperimento gli agenti fortemente imitatori di mercato.

Le caratteristiche principali dell'esperimento sono:

BookNumber: 4

ArbitrageurAgent: 0

RandomAgent: 300

BPCTAgent: 1 per tipologia

StopAtDay: 300

In questa sede viene riportata la versione dell'esperimento relativa a 300 giorni di contrattazione, per motivi espositivi in quanto la distinzione tra l'andamento del *future* e del suo sottostante è meglio visibile nel grafico riportante meno giorni di contrattazione. Tale esperimento è comunque già significativo per i risultati ottenuti, in quanto, sono state svolte altre prove che prevedevano più giorni di contrattazione, ma i risultati sono stati i medesimi.

I risultati emersi fanno ben vedere come non vi sia nessuna forma di allineamento tra i due corsi, infatti il *future* (descritto con il color grigio nel grafico) si muove indipendentemente dalla pista descritta dal valore dell'indice sottostante, che in SUM ne rappresenta il valore teorico (descritto con il color ocra nel grafico).

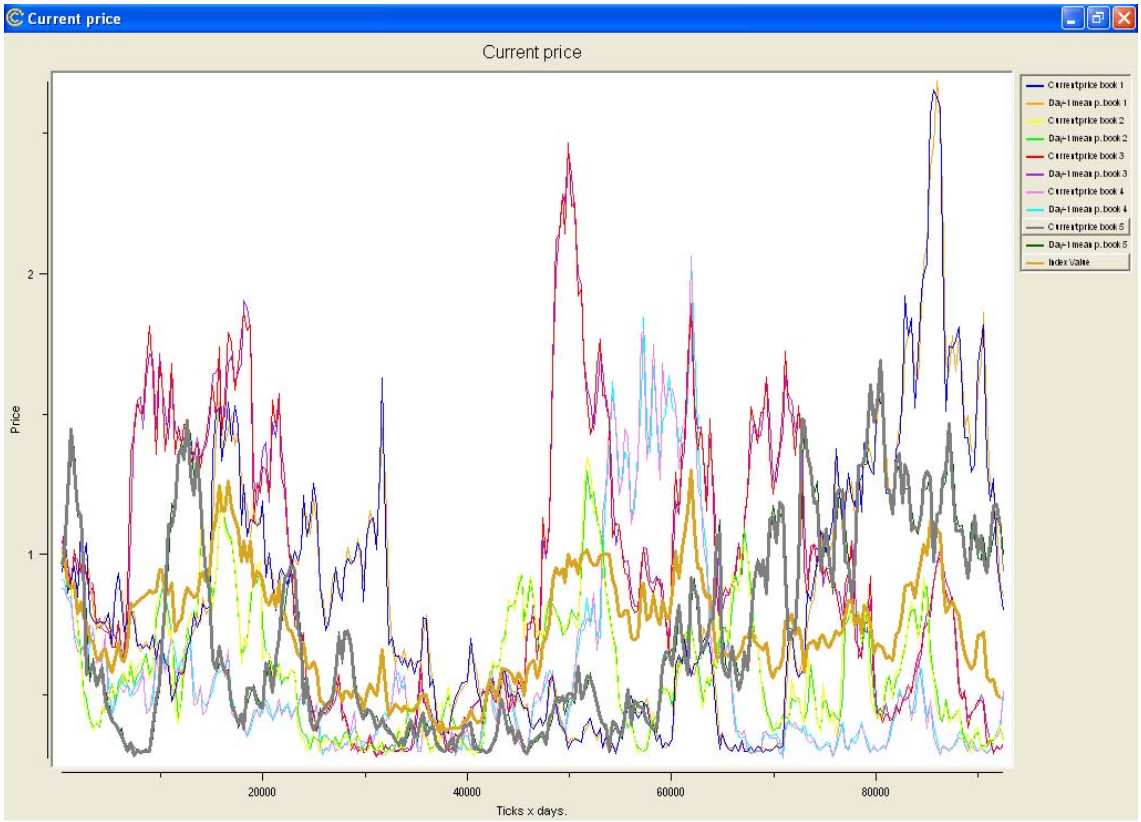

Grafico Esperimento 1

Il secondo esperimento prevede sempre un mercato senza la presenza dell'agente arbitraggista, però a differenza del primo esperimento, qui vengono impiegati tutti gli agenti imitatori di mercato ed in più l'agente che segue le previsioni calcolate dall'agente neurale.

Le caratteristiche dell'esperimento (senza arbitraggista) sono le seguenti:

BookNumber: 4

ArbitrageurNumber: 0

RandomAgent: 400

MarketImitatingAgent: 10

LocallyImitatingAgent: 10

StopLossAgent: 10

ANNForecastAppAgent: 10

StopAtDay: 300

Anche in questo caso si riporta una versione dell'esperimento per soli 300 giorni perché ugualmente rappresentativa della situazione che si viene a formare. Essendo gli imitatori di mercato decisamente destabilizzanti di mercato rispetto alle altre tipologie di agenti, si è pensato di incrementare il numero di agenti *random* per evitare che si formassero bolle e *crash* eccessivi, che avrebbero impedito la comoda visualizzazione del grafico.

Anche in quest'esperimento si vede bene come l'andamento del *future* (rappresentato in grigio nel grafico) sia indipendente da quello del suo indice sottostante (rappresentato in ocra nel grafico).

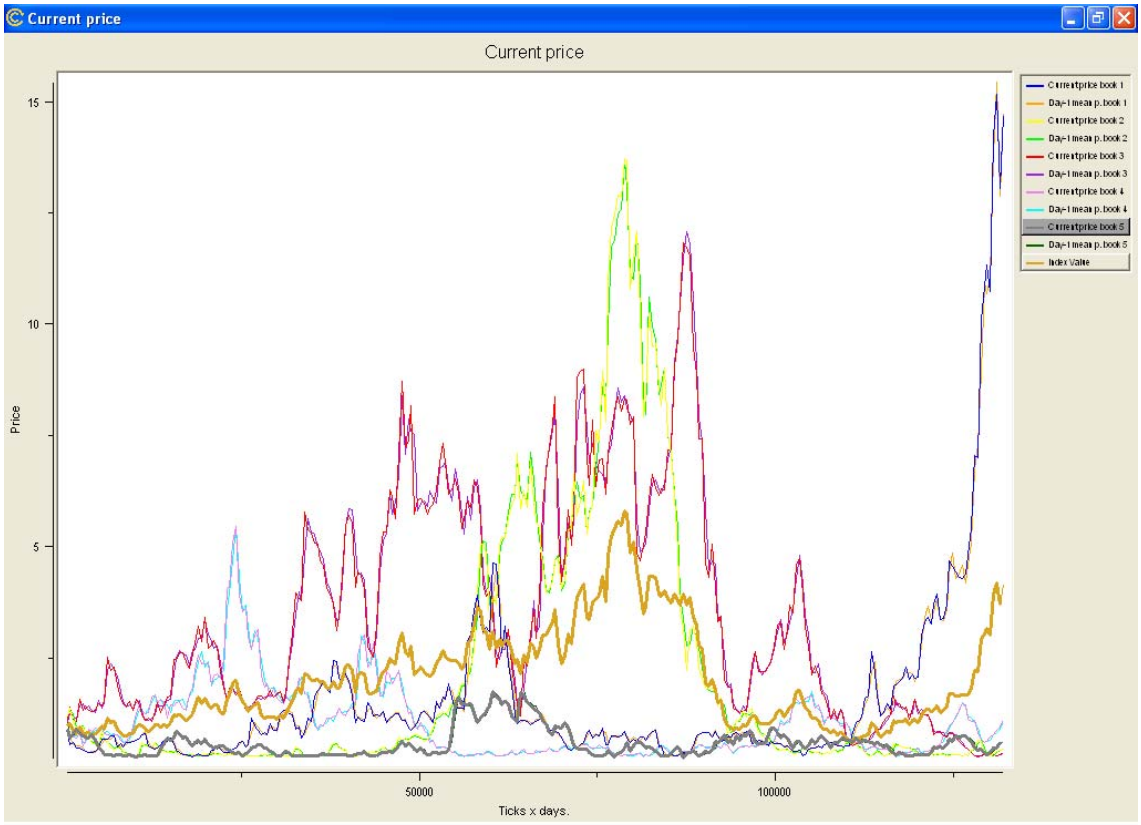

Grafico Esperimento 2

10.2 Il mercato con il future e con l'arbitraggista

10.2.1 Risultati attesi

Da questa sessione di esperimenti ci si aspetta di vedere emergere quanto descritto dagli operatori di mercato, e cioè che l'operato di arbitraggisti non sia sufficiente ad allineare i corsi del *future* e del suo sottostante.

Bisogna tuttavia dire che in linea teorica si è guardato a tale ipotesi pervasi da dubbio, in quanto non ci si spiegava come fosse possibile che gli operatori di mercato nel loro complesso avessero cognizione di quanto un *future* sull'indice rappresenti. Infatti, ci si è chiesti come una buona parte di operatori di mercato, quella meno esperta, potesse avere le informazioni relative a quello che è realmente un *future* e soprattutto alla relazione (matematica) che lo lega al suo sottostante. Infatti, un conto è conoscere alcune delle caratteristiche del *future* come strumento derivato, un altro conto è però avere le conoscenze necessarie a capire realmente il perché della quotazione del *future*; è indubbio che venga da pensare che tali conoscenze siano riservate ad operatori esperti che hanno un costante contatto con il mercato. Tuttavia sui mercati finanziari operano una moltitudine di attori che non posseggono le conoscenze qui citate e in alcuni casi pur possedendo determinate conoscenze la loro modalità operativa non si cura di raffrontare la quotazione del *future* con il suo valore teorico.

10.2.1. Gli esperimenti con l'arbitraggista

Gli esperimenti con la presenza dell'agente arbitraggista si svolgono su due filoni differenti e sono quattro in totale. Il primo filone (terzo e quarto

esperimento) di esperimenti si occupa di verificare l'operatività dell'arbitraggista quando i costi dell'operazione di acquisto/vendita del *future* siano calcolati in termini percentuali rispetto al controvalore rappresentato dal prezzo del *future*. Il secondo filone (quinto e sesto esperimento) di esperimenti si occupa invece del caso in cui l'operatore arbitraggista debba sostenere costi di tipo per acquistare/vendere il *future*. Tale distinzione dipende dal fatto che pur avendo parlato con operatori di mercato, che in merito ai costi hanno individuato sia costi di tipo fisso sia costi di tipo variabile, non sì è riuscito a capire quale delle due tipologie fosse prevalente rispetto all'altra e nemmeno si è individuato un rapporto tra le due che potesse essere mantenuto in una considerazione di costi misti. Si è quindi pensato di fare sperimentazione con entrambe le versioni di costo, anche in considerazione del fatto che l'interesse era quello di verificare se l'allineamento fosse mantenuto dall'operato di arbitraggisti e non di analizzare, sulla base dei costi da sostenere, quante volte gli stessi arbitraggisti fossero stati in grado di operare. Il terzo ed il quarto esperimento considerano una popolazione di agenti non imitativi di mercato, mentre il quinto ed il sesto considerano popolazioni di agenti decisamente imitative di mercato.

Il costo dell'operazione è fisso quindi nel terzo e nel quarto esperimento, la scelta in termini di entità del costo fisso è stata fissata per un valore non troppo elevato, in modo da garantire una certa aderenza con la realtà, nella quale i costi relativi ad operazioni di acquisto/vendita di *future* sono contenuti; si è fissato un valore pari 0,05.

Le caratteristiche del terzo esperimento sono le seguenti:

BookNumber: 3
RandomAgentNumber: 400
ArbitrageurAgentNumber: 1
ArbitrageurOperatingInervalFixed: 1
BPCTAgentNumber: 1 per tipologia

StopAtDay: 300

Ciò che emerge palesemente, è che il *future* (rappresentato in lilla nel grafico) sia allineato con il suo valore teorico (rappresentato in grigio nel grafico) e che, nonostante non vi siano agenti imitatori di mercato, si sia formata una bolla speculativa.

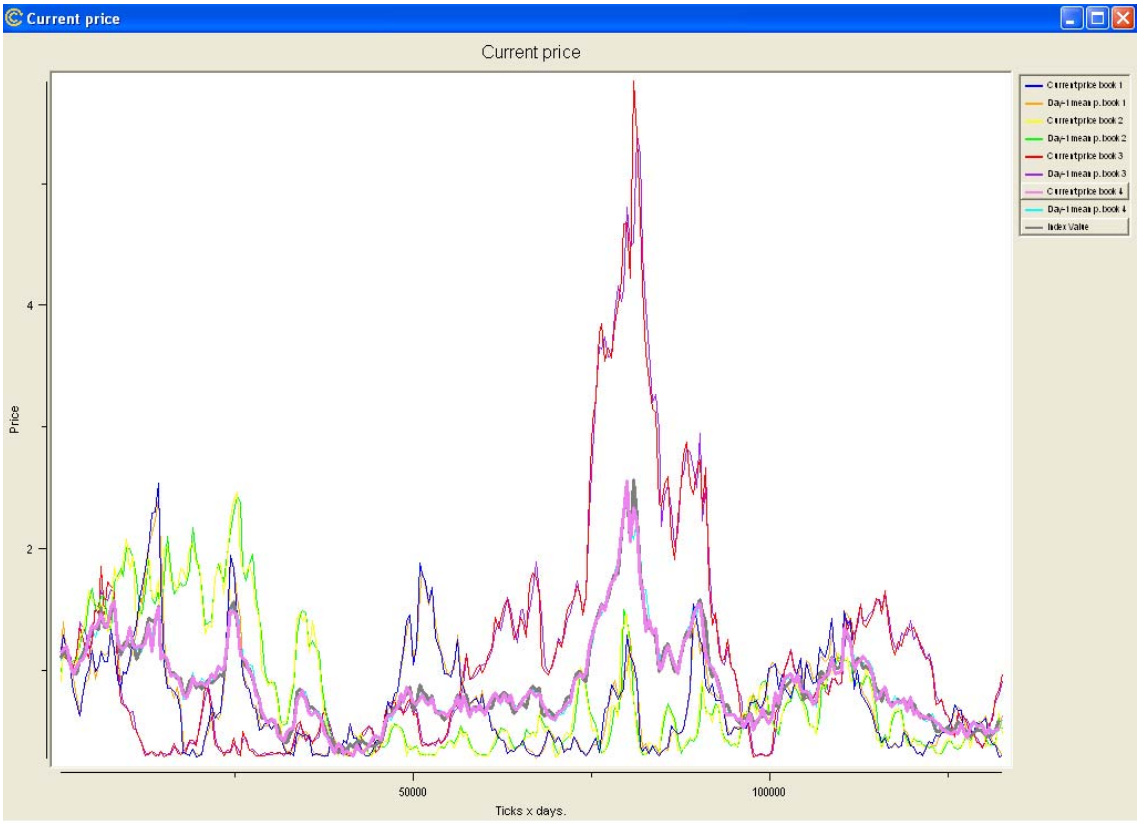

Grafico Esperimento 3

Le caratteristiche del quarto esperimento sono le seguenti:

BookNumber: 3
ArbitrgeurAgentNumber: 2

ArbitrageurOperatingIntervalFixed: 1

RandomAgentNumber: 400

MarketImitatingAgentNumber: 10

LocallyImitatingAgentNumber: 10

StoppLossAgentNumber: 10

ANNForecastAgentNumber: 10

StopAtDayNumber: 300

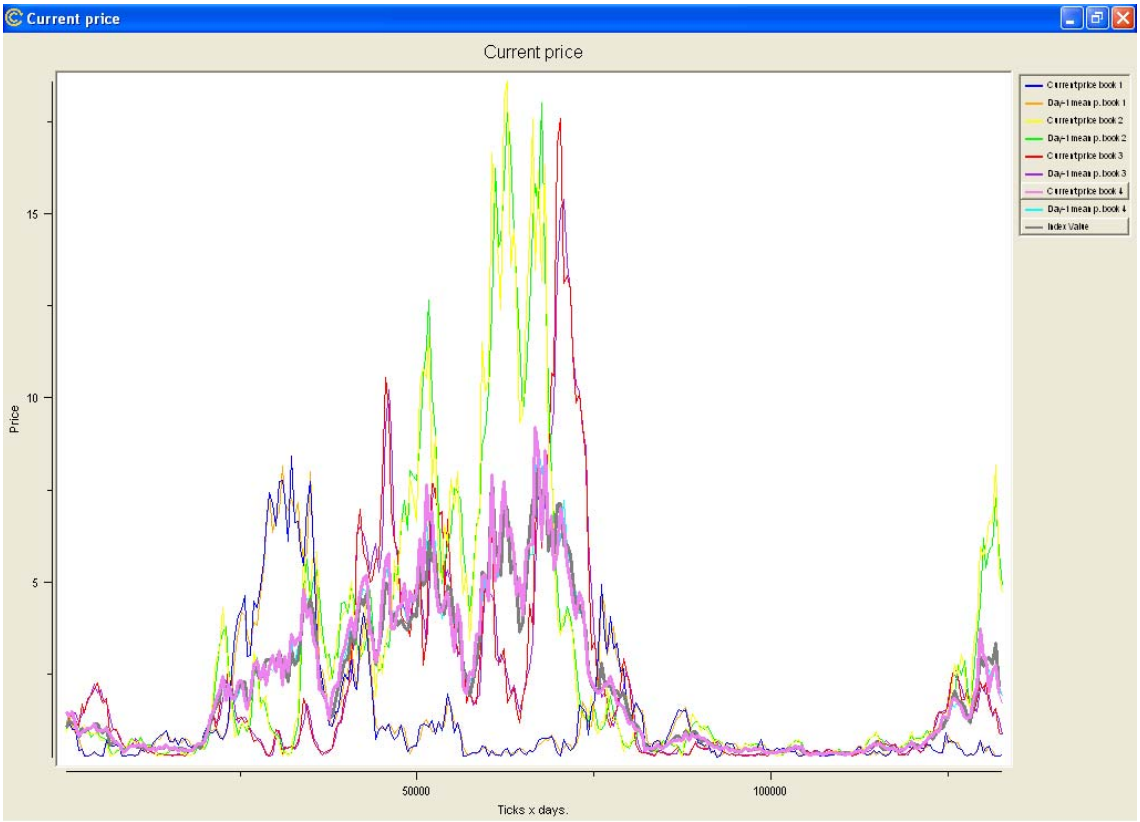

Grafico Esperimento 4

Dal grafico emerge come l'indice (rappresentato in grigio nel grafico) ed il *future* (rappresentato in lilla nel grafico), siano perfettamente sovrapposti, segno questo che indica il fatto che l'operato dell'arbitraggista è sufficiente ad allineare

i due mercati. In quest'esperimento, vista la presenza di molti agenti imitativi di mercato, si sono introdotti due operatori arbitraggisti, al fine di contenere in parte la formazione di bolle e *crash*. Tuttavia, nonostante si sia introdotto tale accorgimento, una bolla speculativa si è formata ed ha portato il prezzo di quasi tutti i titoli a crescere notevolmente rispetto al proprio valore iniziale.

Il quinto ed il sesto esperimento considerano invece l'operatività dell'arbitarggista a costi percentuali; la percentuale è stata fissata per un importo pari a 0,03%.

Le caratteristiche del quinto esperimento sono le seguenti:

BookNumber: 3

ArbitrageurAgentNumber: 1

ArbitrageurOperatingInterval: 0,03

RandomAgentNumber: 300

BPCTAgentNumber: 1 per tipologia

In quest'esperimento, non essendoci agenti imitatori di mercato, si è inserito un solo agente arbitraggista, in quanto si presume che la sua azione sia sufficiente a contenere (in parte) il formarsi di bolle speculative e di *crash*.

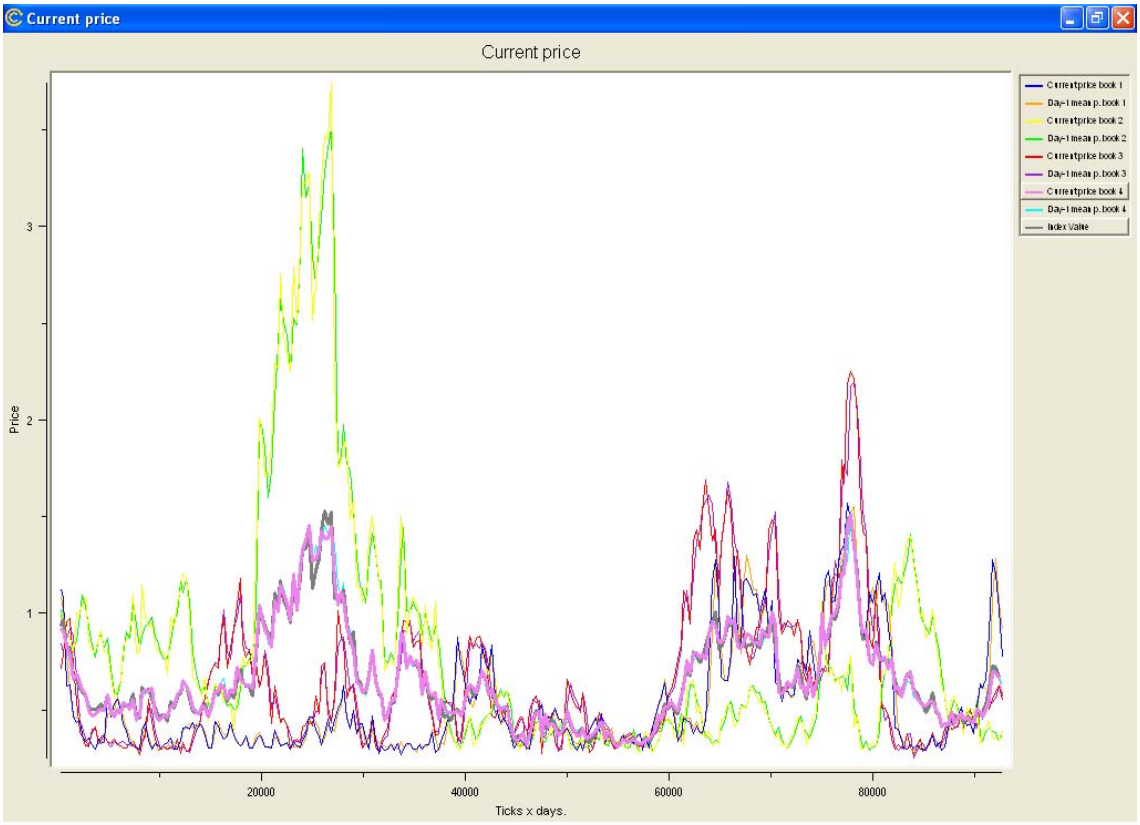

Grafico Esperimento 5

Come nei precedenti due esperimenti, il *future* (rappresentato in lilla nel grafico) è allineato con il suo sottostante (rappresentato in grigio nel grafico). Quindi l'arbitraggista riesce a mantenere i due mercati in equilibrio anche quando i costi delle sue operazioni sono fissi.

Il sesto esperimento presenta le seguenti caratteristiche:

BookNumber: 3

ArbitrageurAgentNumber: 2

ArbitrageurOperatingInterval: 0,03

RandomAgentNumber: 400

MarketImitatingAgentNumber: 10

LocallyImitatingAgentNumber: 10

StoppLossAgentNumber: 10

ANNForecastAgentNumber: 10

StopAtDayNumber: 300

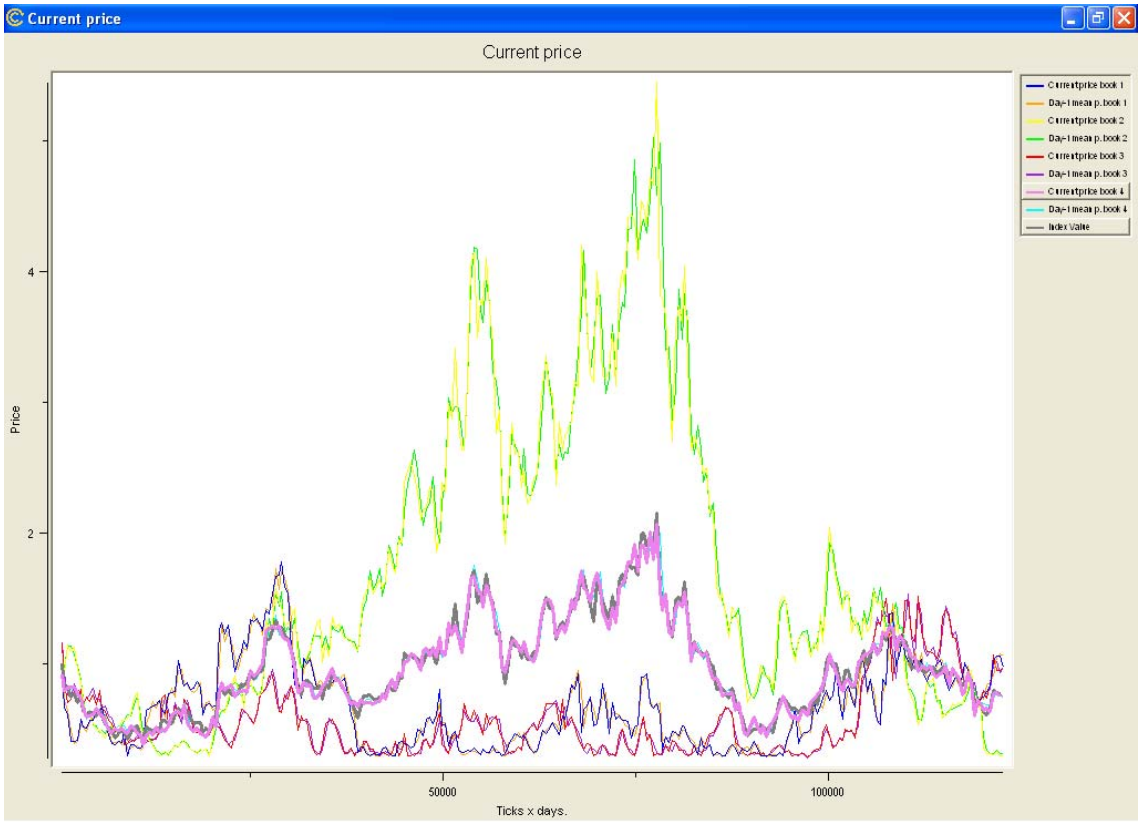

Grafico Esperimento 6

Anche in questo caso si sono inseriti due arbitraggisti per compensare in parte l'azione destabilizzante a livello di mercato degli agenti imitatori. La situazione intermini di equilibrio tra mercato a pronti e mercato a termine è mantenuta anche in questo esperimento.

10.2.3 Risultati Ottenuti

Anche in questo caso, considerando l'operatività dell'arbitraggista con costi percentuali e con costi fissi, rispetto alla quotazione del contratto *future*, si ottiene che l'allineamento del mercato a termine con quello del mercato a pronti è garantito. In entrambi gli esperimenti ciò si verifica, sia in quello in cui il mercato è popolato da agenti non imitativi, sia in quello in cui il mercato è popolato da agenti imitatiti del comportamento degli altri.

10.3 L'aspetto cognitivo relativo al future come caratteristica degli altri agenti

A questo punto l'idea, sostenuta dagli operatori di mercato circa la presenza di cognitività da parte degli operatori stessi nei confronti di ciò che rappresenta il *future*, potrebbe essere inserita nella simulazione e testata con vari esperimenti, non più per far sì che il *future* si allinei con la sua valutazione teorica, bensì per cercare di verificare quali siano le conseguenze sull'allineamento dei due mercati e sull'operatività degli arbitraggisti. Infatti si potrebbe ben verificare che, essendo diffusa la cognitività relativamente al future tra gli operatori, la loro azione fosse già sufficiente a far sì che l'andamento di *future* e sottostante fosse mantenuto. In tali condizioni le possibilità di operare dell'arbitraggista si ridurrebbero notevolmente; questo indicherebbe che l'operato degli agenti nel loro complesso manterrebbe, in linea di massima, l'allineamento tra quotazione del *future* e suo valore teorico, è ciò comporterebbe una minore attività degli arbitraggisti in quanto si ridurrebbero le possibilità di ottenere dei guadagni da tali operazioni.

10.4. Conclusioni

In definitiva da tutti gli esperimenti fatti, dei quali in questa sede si offre solo una parte rappresentativa, è emerso palesemente che le operazioni svolte da un agente che sfrutti le possibilità di arbitraggio, tra due mercati che trattino merci i cui valori sono relazionati tra loro, sono sufficienti a mantenere la coerenza in termini di prezzo che i loro legami rappresentano. Quindi è evidente che la convinzione di alcuni operatori circa l'impossibilità delle operazioni di arbitraggio di garantire da sole l'andamento coerente tra sottostante e derivato non è dimostrata nella realtà simulata e poiché la realtà simulata replica i legami, le strutture, le regole della realtà che si vuole analizzare, il ragionamento appena fatto può essere esteso ai mercati finanziari reali, ed a tale proposito si può dire che in essi l'equilibrio e garantito da operazioni di arbitraggio. Ben evidente è anche il fatto che senza operazioni di arbitraggio e senza cognizione da parte degli investitori in merito al concetto di *future* sull'indice, la coerenza tra i due mercati non è per nulla mantenuta, in quanto le quotazioni dei rispettivi strumenti finanziari seguono due differenti andamenti, indipendenti l'uno dall'altro.

Quindi, si può affermare che i risultati ottenuti in questa fase di sperimentazione del modello SUM, hanno fatto chiarezza su un argomento di difficile trattazione, che, grazie alle simulazioni ad agenti, si è riusciti ad interpretare con maggior facilità di quanto non si sarebbe potuto fare altrimenti.

Oltre al risultato appena esposto, che è lo scopo della tesi, è emerso un altro importante risultato, che tuttavia necessita di essere ancora analizzato specificamente, ed è il fatto che l'operato di agenti arbitraggisti limiti il formarsi delle bolle speculative e dei relativi crash, che si formano soprattutto con la presenza di agenti imitatori del comportamento degli altri. Infatti, come è anche possibile vedere dagli esperimenti esposti sopra, in presenza di agenti imitatori di mercato le bolle speculative ed i *crash* tendono a formarsi più facilmente, come è intuibile. L'operato degli arbitraggisti tende a contenere tali anomalie, in quanto,

quando si forma una bolla speculativa su un titolo, la conseguenza di tale evento si ripercuote nei confronti del prezzo teorico del *future* che sale, creando opportunità all'ingresso dell'arbitraggista. In questa condizione l'arbitraggista trovando il sottostante sopravvalutato rispetto al valore del *future*, opererà acquistando il future e vendendo il sottostante; proprio la posizione assunta dall'arbitraggista nei confronti del sottostante sarà contenitiva nei confronti della bolla speculativa, in quanto egli si porrà come controparte a quegli agenti che stanno formando la bolla (mettendosi in vendita) e tenderà a ridurre lo squilibrio che si è generato sul book, nel lato della domanda, limitando la bolla speculativa. Identica, ma speculare è la situazione che si crea in caso di *crash*; l'effetto delle operazioni, sarà sempre quello di limitare le anomalie di mercato, ma in questo caso queste anomalie saranno i *crash*, i quali squilibrano il book di contrattazione dal lato dell'offerta.

Tale effetto limitativo di bolle e crash servirà probabilmente a contenere gli andamenti dei titoli quando ad operare nel mercato vi siano anche gli *eventAgent*, cioè gli agenti sensibili al verificarsi di eventi esogeni; infatti tali operatori creano consistenti bolle e *crash*.

Appendice

// **ModelSwarm.m**

```
#import "ModelSwarm.h"

@implementation ModelSwarm

+ createBegin: aZone
{
  ModelSwarm *obj;
  id <ProbeMap> probeMap;

  // in createBegin, we set up the simulation parameters

  // first, call our superclass createBegin - the return value is the
  // allocated Swarm object.

  obj = [super createBegin: aZone];

  // now fill in simulation parameters with default values.

  // agentForge step 5
  obj-> bookNumber                          =  3;
  obj-> arbitrageurAgentNumber              =  1;
  obj-> arbitrageurOperatingInterval        =  0.1;
  obj-> arbitrageurOperatingIntervalFixed    =  1;
  obj-> arbitrageurGain                      =  1;
  obj-> randomAgentNumber                    =  300;
  obj-> marketImitatingAgentNumber           =  0;
  obj-> locallyImitatingAgentNumber           =  0;
  obj-> eventsAgentNumber                    =  0;
  obj-> stopLossAgentNumber                  =  0;
  obj-> avatarAgentNumber                    =  2;
  obj-> aNNForecastAppAgentNumber            =  0;
  obj-> bPCTAgentAEO_EP_0_Number             =  0;
  obj-> bPCTAgentAEO_EP_1_Number             =  0;
  obj-> bPCTAgentAEO_EP_2_Number             =  0;
  obj-> bPCTAgentAEO_EP_3_Number             =  0;
  obj-> bPCTAgentBEO_EP_0_Number             =  0;
  obj-> bPCTAgentBEO_EP_1_Number             =  0;
  obj-> bPCTAgentBEO_EP_2_Number             =  0;
  obj-> bPCTAgentBEO_EP_3_Number             =  0;

  obj->bPCTAgentANumber = obj->bPCTAgentAEO_EP_0_Number + obj-
>bPCTAgentAEO_EP_1_Number
```

```
                + obj->bPCTAgentAEO_EP_2_Number + obj-
>bPCTAgentAEO_EP_3_Number;
  obj->bPCTAgentBNumber = obj->bPCTAgentBEO_EP_0_Number + obj-
>bPCTAgentBEO_EP_1_Number
                + obj->bPCTAgentBEO_EP_2_Number + obj-
>bPCTAgentBEO_EP_3_Number;

  obj-> agentNumber = obj-> randomAgentNumber + obj->
marketImitatingAgentNumber +
                obj-> locallyImitatingAgentNumber + obj->stopLossAgentNumber +
                obj-> aNNForecastAppAgentNumber + obj->bPCTAgentANumber +
                obj -> bPCTAgentBNumber + obj -> avatarAgentNumber +
                obj-> eventsAgentNumber + obj-> arbitrageurAgentNumber ;
  // repeat this sum below, in BuildObjects method

  obj-> dayNumber                          = 0;
  obj-> asymmetricBuySellProb              = 0.9;
  obj-> minCorrectingCoeff                 = 0.9;
  obj-> maxCorrectingCoeff                 = 1.1;
  obj-> asymmetricRange                    = 0.0;
  obj-> agentProbToActBeforeOpening        = 0.05;
  obj-> floorP                             = 0.3;
  obj-> agentProbToActBelowFloorP          = 0.5;
  obj-> maxOrderQuantity                   = 1; // max order number per agent

  // agentForge step 6
  // used by imitating agents
  obj-> meanPriceHistoryLength             = 200;
  obj-> priceVolumesHistoryLength          = 200;
  obj-> quantityVolumesHistoryLength       = 200;
  obj-> localHistoryLength                 = 20;
  // used by stop loss agent
  obj-> maxLossRate                        = 0.10;
  obj-> stopLossInterval                   = 2;
  obj-> checkingIfShortOrLong              = 1;
  // used by forecasting agent
  obj-> dataWindowLength                   = 30;
  obj-> nAheadForecasting                  = 10;
  // used by forecasting agent
  obj-> forecastingTrainingSetLength       = 100;
  obj->
 epochNumberInEachForecastingTrainingCycle = 100;
  obj-> learningProcessEveryNDays          = 10;
  obj-> cleanForecastingANNEveryMgtemNDays = 50;
  // used by agents applying ANN forecast
  obj-> aNNInactivityRange                 = 0.02;
  obj-> aNNForecastAppAgentActDailyProb    = 0.1;
  // used by agent of BPCT type
```

```
obj->epochNumberInEachBPCTTrainingCycle    = 100;
obj->agentAEO_EPDelta                      = 0.1;
obj->agentBEO_EPDelta                      = 10;

obj-> printing                  = 0; // if 1 many objects print
                                    // data on the terminal
                                    // window; if 2 only
                                    // forecastingAgent prints;
                                    // forecastinAgent uses also 3;
                                    // 4 is used in BasicSumAgent;
                                    // 5 in ANNForecastAppAgent;

obj-> delay                     = 0; // if delay>0 the random agent execute a for cicle to
delay the                       execution of the simualtion for n seconds, where n is the
number inticated

// build a customized probe map. Without a probe map, the default
// is to show all variables and messages. Here we choose to
// customize the format of the probe to give a nicer interface.

probeMap = [EmptyProbeMap createBegin: aZone];
[probeMap setProbedClass: [self class]];
probeMap = [probeMap createEnd];

[probeMap addProbe: [probeLibrary
 getProbeForVariable: "bookNumber"                          inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "arbitrageurAgentNumber"              inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "arbitrageurOperatingInterval"        inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "arbitrageurOperatingIntervalFixed" inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "arbitrageurGain"                     inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "randomAgentNumber"                   inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "avatarAgentNumber"                   inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "marketImitatingAgentNumber"          inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "locallyImitatingAgentNumber"         inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "stopLossAgentNumber"                 inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "aNNForecastAppAgentNumber"           inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "bPCTAgentAEO_EP_0_Number"            inClass: [self class]]];
```

```
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "bPCTAgentAEO_EP_1_Number"    inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "bPCTAgentAEO_EP_2_Number"    inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "bPCTAgentAEO_EP_3_Number"    inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "bPCTAgentBEO_EP_0_Number"    inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "bPCTAgentBEO_EP_1_Number"    inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "bPCTAgentBEO_EP_2_Number"    inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "bPCTAgentBEO_EP_3_Number"    inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "agentNumber"                 inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "asymmetricBuySellProb"          inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "minCorrectingCoeff"             inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "maxCorrectingCoeff"             inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "asymmetricRange"                inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "agentProbToActBeforeOpening"    inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "floorP"                         inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "agentProbToActBelowFloorP"      inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "maxOrderQuantity"               inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "meanPriceHistoryLength"         inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "priceVolumesHistoryLength"       inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "quantityVolumesHistoryLength"

        inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "localHistoryLength"              inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "maxLossRate"                     inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "stopLossInterval"                inClass: [self class]]];
[probeMap addProbe: [probeLibrary
 getProbeForVariable: "checkingIfShortOrLong"           inClass: [self class]]];
```

```
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "dataWindowLength"                    inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "nAheadForecasting"                   inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable:"forecastingTrainingSetLength"         inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable:"epochNumberInEachForecastingTrainingCycle"
                                                              inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "learningProcessEveryNDays"           inClass: [self
class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "cleanForecastingANNEveryMgtemNDays"
                                                              inClass: [self
class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "aNNInactivityRange"                  inClass: [self
class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "aNNForecastAppAgentActDailyProb"
                                                              inClass: [self
class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "epochNumberInEachBPCTTrainingCycle"
                                                              inClass: [self
class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "agentAEO_EPDelta"                    inClass: [self
class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "agentBEO_EPDelta"                    inClass: [self
class]]];

  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "printing"                            inClass: [self
class]]];

  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "delay"                               inClass: [self
class]]];

  [probeMap addProbe: [probeLibrary
   getProbeForMessage:  "openProbeTo:"                        inClass: [self
class]]];


  // Now install our custom probeMap into the probeLibrary.
```

```
  [probeLibrary setProbeMap: probeMap For: [self class]];

  return obj;
}

- createEnd
{
  return [super createEnd];
}


- buildObjects
{
  int i;
  float otherAgents;

  // agentForge step 7
  RandomAgent * anAgent1;
  MarketImitatingAgent * anAgent2;
  LocallyImitatingAgent * anAgent3;
  StopLossAgent * anAgent4;
  ANNForecastAppAgent * anAgent5;
  BPCTAgentA * anAgent6;
  BPCTAgentB * anAgent7;
  AvatarAgent * anAgent8;
  ArbitrageurAgent * anAgent9;

  BPCTAgentAInterface * anInterface6;
  BPCTAgentBInterface * anInterface7;

  BPCTDataWarehouse * aDataWarehouse;

  // BPCT specific
  char * bPCTMinmaxFileNameA      ="minmaxA.data",  // mandatory, one for each
                                    // BPCT agent type
      * bPCTMinmaxFileNameB      ="minmaxB.data",

      * bPCTInitValuesFileNameA  ="initA.val",     // this file is used only
                                    // with internal data
                                    // generation (CT scheme)
                                    // but anyway the existence
                                    // of the file is not
                                    // mandatory
      * bPCTInitValuesFileNameB  ="initB.val",
      * unusedFile               ="";
  // BPCT end
```

```
// creating tools to generate ad hoc distributions to be used
// in SimpleANN and BPCT, to avoid interferences with random
// sequences used in determining agent behavior

// agentForge step 7r

unsigned int mySeed, mySeed2, mySeed3;
// used by forecastingAgent
mySeed = 223776;
myGenerator = [MT19937gen create: self setStateFromSeed: mySeed];
        // MT19937gen is the generator internally used for default call
        // to Dbl and Int uniform distributions
myUniformDblRand = [UniformDoubleDist create: self
                                    setGenerator: myGenerator];
myUniformIntRand = [UniformIntegerDist create: self
                                    setGenerator: myGenerator];
// used by BPCTAgentA
mySeed2 = 112233;
myGenerator2 = [MT19937gen create: self setStateFromSeed: mySeed2];
        // MT19937gen is the generator internally used for default call
        // to Dbl and Int uniform distributions
myUniformDblRand2 = [UniformDoubleDist create: self
                                    setGenerator: myGenerator2];
myUniformIntRand2 = [UniformIntegerDist create: self
                                    setGenerator: myGenerator2];

// used by BPCTAgentB
mySeed3 = 333112;
myGenerator3 = [MT19937gen create: self setStateFromSeed: mySeed3];
        // MT19937gen is the generator internally used for default call
        // to Dbl and Int uniform distributions
myUniformDblRand3 = [UniformDoubleDist create: self
                                    setGenerator: myGenerator3];
myUniformIntRand3 = [UniformIntegerDist create: self
                                    setGenerator: myGenerator3];

// agentForge step 5b
// this is an operating second definition of the followin sums (see above in
// obj-> agentNumber assignement

bPCTAgentANumber = bPCTAgentAEO_EP_0_Number +
bPCTAgentAEO_EP_1_Number
            + bPCTAgentAEO_EP_2_Number + bPCTAgentAEO_EP_3_Number;

bPCTAgentBNumber = bPCTAgentBEO_EP_0_Number +
bPCTAgentBEO_EP_1_Number
            + bPCTAgentBEO_EP_2_Number + bPCTAgentBEO_EP_3_Number;
```

```
  agentNumber = randomAgentNumber + marketImitatingAgentNumber +
              locallyImitatingAgentNumber + stopLossAgentNumber +

aNNForecastAppAgentNumber+bPCTAgentANumber+bPCTAgentBNumber+
              avatarAgentNumber+eventsAgentNumber+arbitrageurAgentNumber;

  // if dataWindowLength+nAheadForecasting > meanPriceHistoryLength
  // a matrix error arises
  if(dataWindowLength+2*nAheadForecasting>meanPriceHistoryLength)
  {
    printf("The sum of dataWindowLength plus 2*nAheadForecasting is\n"
           "greater than meanPriceHistoryLength; this is a nonsense.\n"
           "See the comment 'NB about lagged values and return indexes' in\n"
           "ForecastingAgent.m\n");
    exit(0);
  }

  // randomRuleMaster
  randomRuleMaster=[RandomRuleMaster createBegin: self];
  [randomRuleMaster setAgentProbToActBeforeOpening:
                              agentProbToActBeforeOpening];
  [randomRuleMaster setMinCorrectingCoeff: minCorrectingCoeff];
  [randomRuleMaster setMaxCorrectingCoeff: maxCorrectingCoeff];
  [randomRuleMaster setAsymmetricRange: asymmetricRange];
  [randomRuleMaster setFloorP: floorP
             andAgentProbToActBelowFloorP: agentProbToActBelowFloorP];
  randomRuleMaster=[randomRuleMaster createEnd];

  // stopLossRuleMaster
  stopLossRuleMaster=[StopLossRuleMaster createBegin: self];
  [stopLossRuleMaster setAgentProbToActBeforeOpening:
                              agentProbToActBeforeOpening];
  [stopLossRuleMaster setMinCorrectingCoeff: minCorrectingCoeff];
  [stopLossRuleMaster setMaxCorrectingCoeff: maxCorrectingCoeff];
  [stopLossRuleMaster setAsymmetricRange: asymmetricRange];
  [stopLossRuleMaster setFloorP: floorP
             andAgentProbToActBelowFloorP: agentProbToActBelowFloorP];
  stopLossRuleMaster=[stopLossRuleMaster createEnd];

  listShuffler=[ListShuffler createBegin: self];
  listShuffler=[listShuffler createEnd];

  bookList=[List create: self];
  arbitrageurAgentList=[List create: self];
  // agentForge step 5bb
  agentList=[List create: self];
  randomAgentList=[List create: self];
  avatarAgentList=[List create: self];
```

```
marketImitatingAgentList=[List create: self];
locallyImitatingAgentList=[List create: self];
eventsAgentList=[List create: self];
stopLossAgentList=[List create: self];
aNNForecastAppAgentList=[List create: self];
bPCTAgentAList=[List create: self];
bPCTAgentBList=[List create: self];
forecastingAgentList=[List create: self];

agentArray = [Array create: self];
if (agentNumber==0) {printf("Nonsense: agentNumber cannot be 0");exit(0);}
[agentArray setCount: agentNumber];
agentArrayIndex = [agentArray begin: self];

// agentForge step 5bbb

if (bPCTAgentANumber!=0)
{
 bPCTAgentAArray = [Array create: self];
 [bPCTAgentAArray setCount: bPCTAgentANumber];
 bPCTAgentAArrayIndex = [bPCTAgentAArray begin: self];
}

if (bPCTAgentBNumber!=0)
{
 bPCTAgentBArray = [Array create: self];
 [bPCTAgentBArray setCount: bPCTAgentBNumber];
 bPCTAgentBArrayIndex = [bPCTAgentBArray begin: self];
}


if (bookNumber<1){printf("Nonsense: You need almost ONE book (internally set to
1)\n");
                        bookNumber=1;
                        }

bookNumber++; // add one book for index

if (arbitrageurAgentNumber>=1){
            futureBook=bookNumber; // the futureBook is the last one
             }

bookArray = [Array create: self];
[bookArray setCount: bookNumber];
bookArrayIndex = [bookArray begin: self];

forecastingAgentArray = [Array create: self];
[forecastingAgentArray setCount: bookNumber];
```

```
forecastingAgentArrayIndex = [forecastingAgentArray begin: self];

maxOrderQuantity2=2*agentNumber*bookNumber;


for (i=1;i<=bookNumber;i++)
{
 aBook = [Book createBegin: self];
 [aBook setNumber: i];
 [aBook setAgentArrayIndex: agentArrayIndex];
 [aBook setNAheadForecasting: nAheadForecasting];
 [aBook setAgentNumber: agentNumber];           // to build the matrixes
 [aBook setMaxOrderQuantity: maxOrderQuantity2];  // "    "
  if (meanPriceHistoryLength<2){printf("The length of the history of mean "
                                "prices cannot be <2 (internally "
                                "set to 2).\n");
                          meanPriceHistoryLength=2;
                          }
 [aBook setMeanPriceHistoryLength: meanPriceHistoryLength];
  if (priceVolumesHistoryLength<2){printf("The length of the history of Volumes "
                                "cannot be <2 (internally "
                                "set to 2).\n");
                          priceVolumesHistoryLength=2;
                          }
 [aBook setPriceVolumesHistoryLength: priceVolumesHistoryLength];
  if (quantityVolumesHistoryLength<2){printf("The length of the history of Volumes "
                                "cannot be <2 (internally "
                                "set to 2).\n");
                          quantityVolumesHistoryLength=2;
                          }
 [aBook setQuantityVolumesHistoryLength: quantityVolumesHistoryLength];
  if (localHistoryLength<1){printf    ("The length of the local history"
                                "cannot be <1       (internally "
                                "set to 1).\n");
                          localHistoryLength=1;
                          }
 [aBook setLocalHistoryLength: localHistoryLength];
 [aBook setPrinting: printing];
 aBook = [aBook createEnd];

 [bookList addLast: aBook];
 [bookArrayIndex next];
 [bookArrayIndex put: aBook];

}


// IndexCalculator
```

```
indexCalculator=[IndexCalculator createBegin: self];
[indexCalculator setBookArrayIndex: bookArrayIndex];
[indexCalculator setBookNumber: bookNumber];
[indexCalculator setPrinting: printing];
indexCalculator=[indexCalculator createEnd];



// a few checks
  if (asymmetricBuySellProb<0.5){printf("The asymmetricBuySellProb "
                                "cannot be < 0.5 (internally "
                                "set to 0.5).\n");
                              asymmetricBuySellProb=0.5;
                              }

  if (stopLossInterval>meanPriceHistoryLength)
                              {printf("stopLossInterval "
                                "cannot be > meanPriceHistoryLength"
                                "\n(internally "
                                "set to meanPriceHistoryLength).\n");
                              stopLossInterval=meanPriceHistoryLength;
                              }







// randomAgent
for (i=1;i<=randomAgentNumber;i++)
{
anAgent1=[RandomAgent createBegin: self];
[anAgent1 setNumber: i];
[anAgent1 setMaxOrderQuantity: maxOrderQuantity];
[anAgent1 setMaxOrderQuantity2: maxOrderQuantity2];
[anAgent1 setBookArrayIndex: bookArrayIndex];
[anAgent1 setBookNumber: bookNumber];
[anAgent1 setRuleMaster: randomRuleMaster];
[anAgent1 setPrinting: printing];
[anAgent1 setDelay: delay];
anAgent1=[anAgent1 createEnd];

[agentList addLast: anAgent1];
[randomAgentList addLast: anAgent1];

[agentArrayIndex next];
```

```
  [agentArrayIndex put: anAgent1];
  }

  // marketImitatingAgent
  for (i=randomAgentNumber+1;i<=randomAgentNumber +
marketImitatingAgentNumber;i++)
  {
  anAgent2=[MarketImitatingAgent createBegin: self];
  [anAgent2 setNumber: i];
  [anAgent2 setAsymmetricBuySellProb: asymmetricBuySellProb];
  [anAgent2 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent2 setMaxOrderQuantity2: maxOrderQuantity2];
  [anAgent2 setBookArrayIndex: bookArrayIndex];
  [anAgent2 setBookNumber: bookNumber];
  [anAgent2 setRuleMaster: randomRuleMaster];
  [anAgent2 setPrinting: printing];
  anAgent2=[anAgent2 createEnd];

  [agentList addLast: anAgent2];
  [marketImitatingAgentList addLast: anAgent2];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent2];
  }

  // locallyImitatingAgent
  for (i=randomAgentNumber+marketImitatingAgentNumber+1;

i<=randomAgentNumber+marketImitatingAgentNumber+locallyImitatingAgentNumbe
r;i++)
  {
  anAgent3=[LocallyImitatingAgent createBegin: self];
  [anAgent3 setNumber: i];
  [anAgent3 setAsymmetricBuySellProb: asymmetricBuySellProb];
  [anAgent3 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent3 setMaxOrderQuantity2: maxOrderQuantity2];
  [anAgent3 setBookArrayIndex: bookArrayIndex];
  [anAgent3 setBookNumber: bookNumber];
  [anAgent3 setRuleMaster: randomRuleMaster];
  [anAgent3 setPrinting: printing];
  anAgent3=[anAgent3 createEnd];

  [agentList addLast: anAgent3];
  [locallyImitatingAgentList addLast: anAgent3];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent3];
  }
```

```
  // stopLossAgent
  for
(i=randomAgentNumber+marketImitatingAgentNumber+locallyImitatingAgentNumber
+1;

i<=randomAgentNumber+marketImitatingAgentNumber+locallyImitatingAgentNumbe
r+
        stopLossAgentNumber;i++)
  {
  anAgent4=[StopLossAgent createBegin: self];
  [anAgent4 setNumber: i];
  [anAgent4 setAsymmetricBuySellProb: asymmetricBuySellProb];
  [anAgent4 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent4 setMaxOrderQuantity2: maxOrderQuantity2];
  [anAgent4 setStopLossInterval: stopLossInterval];
  [anAgent4 setMaxLossRate: maxLossRate
          andCheckingIfShortOrLong: checkingIfShortOrLong];

  int randomBookNumber;
  randomBookNumber=[uniformIntRand getIntegerWithMin: 1 withMax:
bookNumber];
  Book * oneBook;
  [bookArrayIndex setOffset: randomBookNumber-1];
  oneBook=[bookArrayIndex get];
  [anAgent4 setBook: oneBook];
//  [anAgent4 setBookArrayIndex: bookArrayIndex];
//  [anAgent4 setBookNumber: bookNumber];
  [anAgent4 setRuleMaster: stopLossRuleMaster];
  [anAgent4 setPrinting: printing];
  anAgent4=[anAgent4 createEnd];

  [agentList addLast: anAgent4];
  [stopLossAgentList addLast: anAgent4];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent4];
  }

  // we must create here a lot of objects before anAgent5, because we have to
  // pass it the address of ForecastingAgent, needing the other objects created
  // immediately here

  // we create an instance of MatrixMult, VectorTransFunc
  // and of RuleMaster-RuleMaker

  matrixMult = [MatrixMult createBegin: self];
  matrixMult = [matrixMult createEnd];
```

```
transFunc = [TransFunc createBegin: self];
transFunc = [transFunc createEnd];

vectorTransFunc = [VectorTransFunc createBegin: self];
vectorTransFunc = [vectorTransFunc setTransFunc: transFunc];
vectorTransFunc = [vectorTransFunc createEnd];

// creating the simpleANNRuleMaker to evolve the ANN owned by the
// forecastingAgent
simpleANNRuleMaker=[SimpleANNRuleMaker createBegin: self];
[simpleANNRuleMaker setMyUniformIntRand: myUniformIntRand];
[simpleANNRuleMaker setMatrixMult: matrixMult];
[simpleANNRuleMaker setVectorTransFunc: vectorTransFunc];
simpleANNRuleMaker=[simpleANNRuleMaker createEnd];

// creating the simpleANNRuleMaster to apply the ANN owned by the
// forecastingAgent
simpleANNRuleMaster=[SimpleANNRuleMaster createBegin: self];
[simpleANNRuleMaster setSimpleANNRuleMaker: simpleANNRuleMaker];
[simpleANNRuleMaster setMatrixMult: matrixMult];
[simpleANNRuleMaster setVectorTransFunc: vectorTransFunc];
simpleANNRuleMaster=[simpleANNRuleMaster createEnd];

// creating the forecastingAgent
  if (cleanForecastingANNEveryMgtemNDays<learningProcessEveryNDays ||
     cleanForecastingANNEveryMgtemNDays%learningProcessEveryNDays !=0)
           {printf("cleanForecastingANNEveryMgtemNDays must be\n"
                 "must be greater than or equal and multiple of\n"
                 "learningProcessEveryNDays.\n");
                exit(0);
                      }

forecastingAgentArray = [Array create: self];
[forecastingAgentArray setCount: bookNumber];
forecastingAgentArrayIndex = [forecastingAgentArray begin: self];


for (i=1;i<=bookNumber;i++)
{



 Book * oneBook;
 [bookArrayIndex setOffset: i-1];
 oneBook=[bookArrayIndex get];
```

```
  aForecastingAgent=[ForecastingAgent createBegin: self];
  [aForecastingAgent setNumber: i];
  [aForecastingAgent setBook: oneBook];
  [aForecastingAgent setDataWindowLength: dataWindowLength
              andNAheadForecasting: nAheadForecasting
              andForecastingTrainingSetLength: forecastingTrainingSetLength
              andEpochNumberInEachForecastingTrainingCycle:
                epochNumberInEachForecastingTrainingCycle
              andLearningProcessEveryNDays: learningProcessEveryNDays
              andCleanForecastingANNEveryMgtemNDays:
                            cleanForecastingANNEveryMgtemNDays
              andForecastHistoryLength: meanPriceHistoryLength];
  [aForecastingAgent setModelSwarmAddress: self];
  [aForecastingAgent setSimpleANNRuleMasterAddress: simpleANNRuleMaster];
  [aForecastingAgent setMyUniformDblRand: myUniformDblRand]; // ad hoc distr.
  [aForecastingAgent setPrinting: printing];
  aForecastingAgent=[aForecastingAgent createEnd];

  [forecastingAgentList addLast: aForecastingAgent];
  [forecastingAgentArrayIndex next];
  [forecastingAgentArrayIndex put: aForecastingAgent];

   ForecastingAgent * oneForecastingAgent;
   [forecastingAgentArrayIndex setOffset: i-1];
   oneForecastingAgent=[forecastingAgentArrayIndex get];
/*
  quota = [Quota createBegin: self];
  [quota setNumber: i];
 // [quota setBookArrayIndex: bookArrayIndex];
 // [quota setBookNumber: bookNumber];
  [quota setBook: oneBook];
 // [quota setForecastingAgentArrayIndex: forecastingAgentIndex];
  [quota setForecastingAgent: oneForecastingAgent];
  [quota setCleanForecastingANNEveryMgtemNDays:
                        cleanForecastingANNEveryMgtemNDays];
  quota = [quota createEnd];
*/
  }

  // aNNForecastAppAgent
  for
(i=randomAgentNumber+marketImitatingAgentNumber+locallyImitatingAgentNumber
+
        stopLossAgentNumber + 1;

i<=randomAgentNumber+marketImitatingAgentNumber+locallyImitatingAgentNumbe
r+
```

```
        stopLossAgentNumber+aNNForecastAppAgentNumber;i++)
  {
  anAgent5=[ANNForecastAppAgent createBegin: self];
  [anAgent5 setNumber: i];
  [anAgent5 setAsymmetricBuySellProb: asymmetricBuySellProb];
  [anAgent5 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent5 setMaxOrderQuantity2: maxOrderQuantity2];
 // [anAgent5 setBook: theBook];
  [anAgent5 setBookArrayIndex: bookArrayIndex];
  [anAgent5 setBookNumber: bookNumber];

  [anAgent5 setForecastingAgentArrayIndex: forecastingAgentArrayIndex];

  [anAgent5 setANNInactivityRange: aNNInactivityRange
          andANNForecastAppAgentActDailyProb:
aNNForecastAppAgentActDailyProb];
  [anAgent5 setMyUniformDblRand: myUniformDblRand]; // ad hoc distr.
  [anAgent5 setRuleMaster: randomRuleMaster];
  //[anAgent5 setForecastingAgent: forecastingAgent];
  [anAgent5 setPrinting: printing];
  anAgent5=[anAgent5 createEnd];

  [agentList addLast: anAgent5];
  [aNNForecastAppAgentList addLast: anAgent5];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent5];
  }

  // BPCT

  // agentForge step 7b
     bPCTAgentAInputNodeNumber  = 7;
     bPCTAgentAHiddenNodeNumber = 5;
     bPCTAgentAOutputNodeNumber = 3;

     bPCTAgentBInputNodeNumber  = 8;
     bPCTAgentBHiddenNodeNumber = 6;
     bPCTAgentBOutputNodeNumber = 5;

     bPCTPatternNumberInVerificationSet = -1;
     bPCTPatternNumberInTrainingSet     = -10;
     bPCTAgentsAreDisplayingData        =  0;
     usingRandomOrderInBPCTLearning     =  1;
     longTermLearningInBPCT_OnlyWithCompleteTrainingSet = 1;
     useOutputsAsTargetsInBPCT_RelearningScheme = 0;
     bPCTWeightRange = 0.3;
     bPCTEps         = 0.6;
```

```
    bPCTAlpha       = 0.9;

  [ObjectLoader load: self fromFileNamed: "bp.setup"];

  otherAgents=randomAgentNumber+marketImitatingAgentNumber+
            locallyImitatingAgentNumber+
            stopLossAgentNumber+aNNForecastAppAgentNumber;

  // bPCTAgent A
  // the various rule Master/Maker are private, to keep independent the random
  // distributions
  bPCTRuleMakerA = [BPCTRuleMaker createBegin: self];
  [bPCTRuleMakerA setMyUniformIntRand: myUniformIntRand2]; // ad hoc distr.
  [bPCTRuleMakerA setMatrixMult: matrixMult];
  [bPCTRuleMakerA setVectorTransFunc: vectorTransFunc];
  bPCTRuleMakerA = [bPCTRuleMakerA createEnd];

  bPCTRuleMasterA = [BPCTRuleMaster createBegin: self];
  [bPCTRuleMasterA setRuleMaker: bPCTRuleMakerA];
  [bPCTRuleMasterA setMatrixMult: matrixMult];
  [bPCTRuleMasterA setVectorTransFunc: vectorTransFunc];
  bPCTRuleMasterA = [bPCTRuleMasterA createEnd];

  bPCTPriceRuleMasterA = [BPCTPriceRuleMaster createBegin: self];
  [bPCTPriceRuleMasterA setMyUniformDblRand: myUniformDblRand2]; // ad hoc
distr.
  [bPCTPriceRuleMasterA setAgentProbToActBeforeOpening:
                        agentProbToActBeforeOpening];
  [bPCTPriceRuleMasterA setMinCorrectingCoeff: minCorrectingCoeff];
  [bPCTPriceRuleMasterA setMaxCorrectingCoeff: maxCorrectingCoeff];
  [bPCTPriceRuleMasterA setAsymmetricRange: asymmetricRange];
  bPCTPriceRuleMasterA = [bPCTPriceRuleMasterA createEnd];

  for (i=otherAgents + 1;
      i<=otherAgents + bPCTAgentANumber;i++)
  {
  // we set a random book for the agent

  int randomBookNumber;
  randomBookNumber=[uniformIntRand getIntegerWithMin: 1 withMax:
bookNumber];
  Book * oneBook;
  [bookArrayIndex setOffset: randomBookNumber-1];
  oneBook=[bookArrayIndex get];

  // first we create the datawarehouse where BPCTagent technical data are stored
  aDataWarehouse= [BPCTDataWarehouse createBegin: self];
  [aDataWarehouse setMinmaxRowToBeModifiedFromInt: 9
```

```
                    usingGenericIntVariableAddress: &maxOrderQuantity];
  [aDataWarehouse setVerificationFileName: unusedFile // never used here
              andTrainingFileName:     unusedFile // never used here
              andMinmaxName: bPCTMinmaxFileNameA
              andInitValuesFileName: bPCTInitValuesFileNameA];
  [aDataWarehouse setInputNodeNumber:  bPCTAgentAInputNodeNumber
              andHiddenNodeNumber: bPCTAgentAHiddenNodeNumber
              andOutputNodeNumber: bPCTAgentAOutputNodeNumber
         andPatternNumberInVerificationSet: bPCTPatternNumberInVerificationSet
         andPatternNumberInTrainingSet:     bPCTPatternNumberInTrainingSet
         andEpochNumberInEachTrainingCycle:
                              epochNumberInEachBPCTTrainingCycle];
  [aDataWarehouse setBackPropagationParametersWeightRange: bPCTWeightRange
              eps: bPCTEps alpha: bPCTAlpha
              andWithOrderInLearning: usingRandomOrderInBPCTLearning
              andLongTermLearningInCT:
              longTermLearningInBPCT_OnlyWithCompleteTrainingSet
              andUseOutputsAsTargetsInCT:
              useOutputsAsTargetsInBPCT_RelearningScheme];
  [aDataWarehouse setMyUniformDblRand: myUniformDblRand2]; // ad hoc dist.

  aDataWarehouse=[aDataWarehouse createEnd];

  // then we create an interface for our agent, to simplifly its links
  // with the observer, if any, but mainly as a help in CT building

  anInterface6 = [BPCTAgentAInterface createBegin: self];
  if
(i<=otherAgents+bPCTAgentAEO_EP_0_Number+bPCTAgentAEO_EP_1_Number
                +bPCTAgentAEO_EP_2_Number+bPCTAgentAEO_EP_3_Number)
    [anInterface6 setUseEO_EP: 3]; // EO_EP 3
  if
(i<=otherAgents+bPCTAgentAEO_EP_0_Number+bPCTAgentAEO_EP_1_Number
                +bPCTAgentAEO_EP_2_Number)
    [anInterface6 setUseEO_EP: 2]; // EO_EP 2
  if
(i<=otherAgents+bPCTAgentAEO_EP_0_Number+bPCTAgentAEO_EP_1_Number)
    [anInterface6 setUseEO_EP: 1]; // EO_EP 1
  if (i<=otherAgents+bPCTAgentAEO_EP_0_Number)
  [anInterface6 setUseEO_EP: 0]; // no EO_EP use
  [anInterface6 setEO_EPDelta: agentAEO_EPDelta];
  [anInterface6 setMyUniformIntRand: myUniformIntRand2]; // ad hoc distr.
  [anInterface6 setAgentNumber: i];
  [anInterface6 setDataWarehouse: aDataWarehouse];
  [anInterface6 setBook: oneBook];
  anInterface6 = [anInterface6 createEnd];
  [anInterface6 initialize]; // NB. after createEnd
```

```
  anAgent6 = [BPCTAgentA createBegin: self];
  [anAgent6 setNumber: i andSetReadWeightsFromFile: 0]; // it never reads weights
                                        // from a file
  [anAgent6 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent6 setMaxOrderQuantity2: maxOrderQuantity2];
  [anAgent6 setDataWarehouse: aDataWarehouse];
  [anAgent6 setInterface: anInterface6];        // double declaration for the parent
  [anAgent6 setSpecificInterface: anInterface6]; // and for the inheriting class
  [anAgent6 setRuleMaster: bPCTRuleMasterA];
  [anAgent6 setPriceRuleMaster: bPCTPriceRuleMasterA];
  [anAgent6 setDisplayDataWhileRunning: bPCTAgentsAreDisplayingData];
  [anAgent6 setBook: oneBook]; // used by accounting method act2
  [anAgent6 setPrinting: printing];

  anAgent6 = [anAgent6 createEnd];


  [agentList addLast: anAgent6];
  [bPCTAgentAList addLast: anAgent6];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent6];

  [bPCTAgentAArrayIndex next];
  [bPCTAgentAArrayIndex put: anAgent6];

  }

  otherAgents+=bPCTAgentANumber;

  // bPCTAgent B
  // the various rule Master/Maker are private, to keep independent the random
  // distributions
  bPCTRuleMakerB = [BPCTRuleMaker createBegin: self];
  [bPCTRuleMakerB setMyUniformIntRand: myUniformIntRand3]; // ad hoc distr.
  [bPCTRuleMakerB setMatrixMult: matrixMult];
  [bPCTRuleMakerB setVectorTransFunc: vectorTransFunc];
  bPCTRuleMakerB = [bPCTRuleMakerB createEnd];

  bPCTRuleMasterB = [BPCTRuleMaster createBegin: self];
  [bPCTRuleMasterB setRuleMaker: bPCTRuleMakerB];
  [bPCTRuleMasterB setMatrixMult: matrixMult];
  [bPCTRuleMasterB setVectorTransFunc: vectorTransFunc];
  bPCTRuleMasterB = [bPCTRuleMasterB createEnd];

  bPCTPriceRuleMasterB = [BPCTPriceRuleMaster createBegin: self];
  [bPCTPriceRuleMasterB setMyUniformDblRand: myUniformDblRand3]; // ad hoc
distr.
```

```
  [bPCTPriceRuleMasterB setAgentProbToActBeforeOpening:
                        agentProbToActBeforeOpening];
  [bPCTPriceRuleMasterB setMinCorrectingCoeff: minCorrectingCoeff];
  [bPCTPriceRuleMasterB setMaxCorrectingCoeff: maxCorrectingCoeff];
  [bPCTPriceRuleMasterB setAsymmetricRange: asymmetricRange];
  bPCTPriceRuleMasterB = [bPCTPriceRuleMasterB createEnd];

  for (i=otherAgents + 1;
      i<=otherAgents + bPCTAgentBNumber;i++)
  {
  // first we create the datawarehouse where BPCTagent technical data are stored
  aDataWarehouse= [BPCTDataWarehouse createBegin: self];
  [aDataWarehouse setMinmaxRowToBeModifiedFromInt: 12
                  usingGenericIntVariableAddress: &maxOrderQuantity];
  [aDataWarehouse setVerificationFileName: unusedFile // never used here
              andTrainingFileName:     unusedFile // never used here
              andMinmaxName: bPCTMinmaxFileNameB
              andInitValuesFileName: bPCTInitValuesFileNameB];
  [aDataWarehouse setInputNodeNumber:  bPCTAgentBInputNodeNumber
              andHiddenNodeNumber: bPCTAgentBHiddenNodeNumber
              andOutputNodeNumber: bPCTAgentBOutputNodeNumber
         andPatternNumberInVerificationSet: bPCTPatternNumberInVerificationSet
         andPatternNumberInTrainingSet:     bPCTPatternNumberInTrainingSet
         andEpochNumberInEachTrainingCycle:
                           epochNumberInEachBPCTTrainingCycle];
  [aDataWarehouse setBackPropagationParametersWeightRange: bPCTWeightRange
              eps: bPCTEps alpha: bPCTAlpha
              andWithOrderInLearning: usingRandomOrderInBPCTLearning
              andLongTermLearningInCT:
              longTermLearningInBPCT_OnlyWithCompleteTrainingSet
              andUseOutputsAsTargetsInCT:
              useOutputsAsTargetsInBPCT_RelearningScheme];
  [aDataWarehouse setMyUniformDblRand: myUniformDblRand3]; // ad hoc dist.

  aDataWarehouse=[aDataWarehouse createEnd];

  // then we create an interface for our agent, to simplifly its links
  // with the observer, if any, but mainly as a help in CT building

  anInterface7 = [BPCTAgentBInterface createBegin: self];
  if
(i<=otherAgents+bPCTAgentBEO_EP_0_Number+bPCTAgentBEO_EP_1_Number
              +bPCTAgentBEO_EP_2_Number+bPCTAgentBEO_EP_3_Number)
    [anInterface7 setUseEO_EP: 3]; // EO_EP 3
  if
(i<=otherAgents+bPCTAgentBEO_EP_0_Number+bPCTAgentBEO_EP_1_Number
              +bPCTAgentBEO_EP_2_Number)
    [anInterface7 setUseEO_EP: 2]; // EO_EP 2
```

```
  if
(i<=otherAgents+bPCTAgentBEO_EP_0_Number+bPCTAgentBEO_EP_1_Number)
    [anInterface7 setUseEO_EP: 1]; // EO_EP 1
  if (i<=otherAgents+bPCTAgentBEO_EP_0_Number)
    [anInterface7 setUseEO_EP: 0]; // no EO_EP use
  [anInterface7 setEO_EPDelta: agentBEO_EPDelta];
  [anInterface7 setMyUniformIntRand: myUniformIntRand3]; // ad hoc distr.
  [anInterface7 setAgentNumber: i];
  [anInterface7 setDataWarehouse: aDataWarehouse];
 // [anInterface7 setBook: theBook];
 // [anInterface7 setForecastingAgent: forecastingAgent];
  anInterface7 = [anInterface7 createEnd];
  [anInterface7 initialize]; // NB. after createEnd

  anAgent7 = [BPCTAgentB createBegin: self];
  [anAgent7 setNumber: i andSetReadWeightsFromFile: 0]; // it never reads weights
                                        // from a file
  [anAgent7 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent7 setMaxOrderQuantity2: maxOrderQuantity2];
  [anAgent7 setDataWarehouse: aDataWarehouse];
  [anAgent7 setInterface: anInterface7];        // double declaration for the parent
  [anAgent7 setSpecificInterface: anInterface7]; // and for the inheriting class
  [anAgent7 setRuleMaster: bPCTRuleMasterB];
  [anAgent7 setPriceRuleMaster: bPCTPriceRuleMasterB];
  [anAgent7 setDisplayDataWhileRunning: bPCTAgentsAreDisplayingData];
//  [anAgent7 setBook: theBook]; // used by accounting method act2
  [anAgent7 setPrinting: printing];

  anAgent7 = [anAgent7 createEnd];


  [agentList addLast: anAgent7];
  [bPCTAgentBList addLast: anAgent7];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent7];

  [bPCTAgentBArrayIndex next];
  [bPCTAgentBArrayIndex put: anAgent7];

  }

  // AvatarAgent
  otherAgents = otherAgents + bPCTAgentBNumber;

  for (i=otherAgents + 1;
    i<=otherAgents + avatarAgentNumber;i++)
  {
```

```
anAgent8 = [AvatarAgent createBegin: self];
[anAgent8 setNumber: i];
[anAgent8 setBookArrayIndex: bookArrayIndex];
[anAgent8 setBookNumber: bookNumber];
[anAgent8 setPrinting: printing];
[anAgent8 setMaxOrderQuantity: maxOrderQuantity];
[anAgent8 setMaxOrderQuantity2: maxOrderQuantity2];
anAgent8=[anAgent8 createEnd];

[agentList addLast: anAgent8];
[avatarAgentList addLast: anAgent8];

[agentArrayIndex next];
[agentArrayIndex put: anAgent8];
}

// ArbitrageurAgent
otherAgents = otherAgents + avatarAgentNumber;
 for (i= otherAgents + 1;i<=otherAgents + arbitrageurAgentNumber;i++)
{
anAgent9 = [ArbitrageurAgent createBegin: self];
[anAgent9 setNumber: i];
[anAgent9 setBookArrayIndex: bookArrayIndex];
[anAgent9 setBookNumber: bookNumber];
[anAgent9 setArbitrageurOperatingInterval: arbitrageurOperatingInterval];
[anAgent9 setArbitrageurGain: arbitrageurGain];
[anAgent9 setArbitrageurOperatingIntervalFixed: arbitrageurOperatingIntervalFixed];
[anAgent9 setPrinting: printing];
[anAgent9 setMaxOrderQuantity: maxOrderQuantity];
[anAgent9 setMaxOrderQuantity2: maxOrderQuantity2];
[anAgent9 setIndexCalculator: indexCalculator];
anAgent9=[anAgent9 createEnd];

[agentList addLast: anAgent9];
[arbitrageurAgentList addLast: anAgent9];

[agentArrayIndex next];
[agentArrayIndex put: anAgent9];
}


// agentForge step 8




// current agent
```

```
  // see the comment in CurrentAgent.m to understand this trick
  theCurrentAgent=[CurrentAgent createBegin: self];
  [theCurrentAgent setAgentList: agentList];
  theCurrentAgent=[theCurrentAgent createEnd];

  return self;
}

- buildActions
{
  int i;
  // we create the list of simulation actions

  modelActions1 = [ActionGroup create: self];
  [modelActions1 createActionTo: self message: M(setDayStateBeforeOpening)];
  [modelActions1 createActionTo: self message: M(increaseCurrentDayNumber)];
  [modelActions1 createActionForEach: bookList message: M(setClean)];
  [modelActions1 createActionForEach: forecastingAgentList message: M(step)];
//  [modelActions1 createActionTo: quota message: M(step)];
  // this shuffling action is not relevant if agent are equal instances of
  // the same class, but it is useful to avoid biased situations when agents
  // are not homegeneous
  [modelActions1 createActionTo: listShuffler
                        message: M(shuffleWholeList:) : agentList];
  // acting before opening
  [modelActions1 createActionForEach: agentList message: M(act0)];

  // acting in the market
  //status=2;
  modelActions2 = [ActionGroup create: self];
  [modelActions2 createActionTo: self message: M(setDayStateOpening)];
  [modelActions2 createActionTo: theCurrentAgent message: M(act1)];
  if (arbitrageurAgentNumber>0)[modelActions2 createActionForEach:
arbitrageurAgentList message: M(act1)];
  if (avatarAgentNumber>0)[modelActions2 createActionForEach: avatarAgentList
message: M(act1)];
  if (arbitrageurAgentNumber>0)[modelActions2 createActionForEach:
arbitrageurAgentList message: M(act1)];

  // accounting ...
  //status=1;
  modelActions3 = [ActionGroup create: self];
  [modelActions3 createActionTo: self message: M(setDayStateBeforeOpening)];
  [modelActions3 createActionForEach: bookList message: M(setMeanPrice)];
  [modelActions3 createActionForEach: agentList message: M(act2)];
  [modelActions3 createActionTo: indexCalculator message: M(ohlc)];
  // then we create a schedule that executes the modelActions.
```

```
  modelSchedule = [Schedule createBegin: self];
  // we use here agentNumber steps in each cycle, while the observer uses
  // a low display frequency (e.g. 1) to show the price of each step-tick
  // we can also use a high d.f. (e.g.1000 with 100 agents) to run a faster
  // simulation
  [modelSchedule setRepeatInterval: agentNumber];
  modelSchedule = [modelSchedule createEnd];

  [modelSchedule at: 0 createAction: modelActions1];
  for (i=0;i<agentNumber;i++){
          [modelSchedule at: i createAction: modelActions2];
  }
  [modelSchedule at: agentNumber-1 createAction: modelActions3];

  return self;
}

- activateIn: swarmContext
{
  // here, we activate the swarm in the context passed in
  // then we activate our schedule in ourselves

  [super activateIn: swarmContext];

  [modelSchedule activateIn: self];


  return [self getSwarmActivity];
}

- increaseCurrentDayNumber
{
  dayNumber++;
  if(printing==1)printf("Day number #%5d\n",dayNumber);
          FILE * pFile;
          pFile = fopen ("Day.dat","w");
          if (pFile == NULL) {
            perror("cannot open output file model");
            exit(1);
          }
          fprintf(pFile,"%5d\n", dayNumber);
          fclose(pFile);
  return self;
}

- (int) getCurrentDay
{
  return dayNumber;
```

```
}

  // agentForge step 9
- getAgentList{return agentList;}
- getRandomAgentList{return randomAgentList;}
- getArbitrageurAgentList{return arbitrageurAgentList;}

- getAgentArrayIndex{return agentArrayIndex;}
- getBookArrayIndex{return bookArrayIndex;}

- (int) getBPCTAgentANumber{return bPCTAgentANumber;}
- (int) getBPCTAgentBNumber{return bPCTAgentBNumber;}

- getAvatarAgentList{return avatarAgentList;}
- getMarketImitatingAgentList{return marketImitatingAgentList;}
- getLocallyImitatingAgentList{return locallyImitatingAgentList;}
- getStopLossAgentList{return stopLossAgentList;}
- getANNForecastAppAgentList{return aNNForecastAppAgentList;}
- getBPCTAgentAList{return bPCTAgentAList;}
- getBPCTAgentBList{return bPCTAgentBList;}

- getBPCTAgentAArrayIndex{return bPCTAgentAArrayIndex;}
- getBPCTAgentBArrayIndex{return bPCTAgentBArrayIndex;}


// - getForecastingAgentArrayIndex{return forecastingArrayIndex;}

- getIndexCalculator{return indexCalculator;}

- getForecastingAgent: (int) n
{
  ForecastingAgent * oneForecastingAgent;
  [forecastingAgentArrayIndex setOffset: n-1];
  oneForecastingAgent=[forecastingAgentArrayIndex get];
  return oneForecastingAgent;
}

- getBook: (int) n
{
  Book * oneBook;
  [bookArrayIndex setOffset: n-1];
  oneBook=[bookArrayIndex get];
  return oneBook;
}


- setDayStateBeforeOpening
{
```

```
  FILE * pFile;
  pFile = fopen ("daystate.dat","w");
  if (pFile == NULL) {
          perror("cannot open output file model");
          exit(1);
          }
  fprintf(pFile,"Before Opening");
  fclose(pFile);
  return self;
}

- setDayStateOpening
{
  FILE * pFile;
  pFile = fopen ("daystate.dat","w");
  if (pFile == NULL) {
          perror("cannot open output file model");
          exit(1);
          }
  fprintf(pFile,"Opening");
  fclose(pFile);
  return self;
}



- (int) getBookNumber
{
  return bookNumber;
}

- openProbeTo: (int) n
{
  BasicSumAgent * anAgent;

  if (n<1 || n>agentNumber) return self;

  [agentArrayIndex setOffset: n-1];
  anAgent=[agentArrayIndex get];
  [anAgent getProbe];

  return anAgent;
}


@end
```

// **ModelSwarm.h**

```
#import <objectbase/Swarm.h>
#import <simtools.h>      // necessary to invoke ObjectLoader
#import <objectbase.h>    // needed by <ProbeMap> in ModelSwarm.m
#import <activity.h>
#import <collections.h>

  // agentForge step 2

#import "AvatarAgent.h"

#import "BasicSumAgent.h"
#import "RandomAgent.h"
#import "CurrentAgent.h"
#import "MarketImitatingAgent.h"
#import "LocallyImitatingAgent.h"
#import "StopLossAgent.h"
#import "ANNForecastAppAgent.h"
#import "ForecastingAgent.h"
#import "BPCTAgentA.h"
#import "BPCTAgentAInterface.h" // related to the agents used in our simulation
#import "BPCTAgentB.h"
#import "BPCTAgentBInterface.h" // related to the agents used in our simulation
#import "BPCTPriceRuleMaster.h"

#import "ArbitrageurAgent.h"
#import "IndexCalculator.h"

#import "Book.h"
#import "RandomRuleMaster.h"
#import "StopLossRuleMaster.h"
#import "SimpleANNRuleMaster.h"
#import "SimpleANNRuleMaker.h"
#import "MatrixMult.h"
#import "VectorTransFunc.h"
#import "TransFunc.h"
#import "Quota.h"
#import <random.h> // to generate ad hoc private distributions to be used
                   // in SimpleANN and BPCT, to avoid interferences with random
                   // sequences used in determining agent behavior
#import "BPCTRuleMaster.h"
#import "BPCTRuleMaker.h"
#import "BPCTDataWarehouse.h"

@interface ModelSwarm: Swarm
{
```

```
  int bookNumber,futureBook, dayNumber, maxOrderQuantity,maxOrderQuantity2,
priceVolumesHistoryLength,
     quantityVolumesHistoryLength, meanPriceHistoryLength,
      localHistoryLength, stopLossInterval, checkingIfShortOrLong, printing, delay,
      dataWindowLength, nAheadForecasting, forecastingTrainingSetLength,
      epochNumberInEachForecastingTrainingCycle, learningProcessEveryNDays,
      cleanForecastingANNEveryMgtemNDays, arbitrageur, status,
arbitrageurAgentNumber, arbitrageurGain,
      arbitrageurOperatingIntervalFixed;
  float asymmetricBuySellProb, agentProbToActBeforeOpening,
        minCorrectingCoeff, maxCorrectingCoeff,
        asymmetricRange, floorP, agentProbToActBelowFloorP,
        maxLossRate, arbitrageurOperatingInterval ,
        aNNInactivityRange, aNNForecastAppAgentActDailyProb;

  // agentForge step 3
  int   agentNumber, randomAgentNumber, marketImitatingAgentNumber,
avatarAgentNumber,
    locallyImitatingAgentNumber, stopLossAgentNumber, eventsAgentNumber   ,
    aNNForecastAppAgentNumber, bPCTAgentANumber,
    bPCTAgentAEO_EP_0_Number, bPCTAgentAEO_EP_1_Number,
    bPCTAgentAEO_EP_2_Number, bPCTAgentAEO_EP_3_Number,
                        bPCTAgentBNumber,
    bPCTAgentBEO_EP_0_Number, bPCTAgentBEO_EP_1_Number,
    bPCTAgentBEO_EP_2_Number, bPCTAgentBEO_EP_3_Number;
  id <List> bookList, agentList, randomAgentList, marketImitatingAgentList,
           locallyImitatingAgentList, stopLossAgentList, aNNForecastAppAgentList,
           bPCTAgentAList, bPCTAgentBList, avatarAgentList, eventsAgentList,
           forecastingAgentList,arbitrageurAgentList;

  id <Array> bPCTAgentAArray;
  id <Index> bPCTAgentAArrayIndex;

  id <Array> bPCTAgentBArray;
  id <Index> bPCTAgentBArrayIndex;

  id <Array> agentArray;
  id <Index> agentArrayIndex;
  id <Array> forecastingAgentArray;
  id <Index> forecastingAgentArrayIndex;
  id <Array> bookArray;
  id <Index> bookArrayIndex;
  id <ListShuffler> listShuffler;

  id <ActionGroup> modelActions1, modelActions2, modelActions3;
  id <Schedule> modelSchedule;

  // agentForge step 3r
```

```
// see above about private distributions
id <SimpleRandomGenerator> myGenerator;
id <UniformDoubleDist> myUniformDblRand;
id <UniformIntegerDist> myUniformIntRand;
id <SimpleRandomGenerator> myGenerator2;
id <UniformDoubleDist> myUniformDblRand2;
id <UniformIntegerDist> myUniformIntRand2;
id <SimpleRandomGenerator> myGenerator3;
id <UniformDoubleDist> myUniformDblRand3;
id <UniformIntegerDist> myUniformIntRand3;

Book * aBook;

RandomRuleMaster * randomRuleMaster;
IndexCalculator * indexCalculator;

CurrentAgent * theCurrentAgent;
ForecastingAgent * aForecastingAgent;
StopLossRuleMaster * stopLossRuleMaster;
SimpleANNRuleMaster * simpleANNRuleMaster;
SimpleANNRuleMaker * simpleANNRuleMaker;
MatrixMult * matrixMult;
VectorTransFunc * vectorTransFunc;
TransFunc * transFunc;
Quota * quota;


// BPCT

// agentForge step 3b
int epochNumberInEachBPCTTrainingCycle;
float agentAEO_EPDelta, agentBEO_EPDelta;

// we are creating independent RuleMaster/Maker for each type of
// BPCT agent, to have separated random distributions also in weight
// generation
BPCTRuleMaster * bPCTRuleMasterA, * bPCTRuleMasterB;
BPCTRuleMaker * bPCTRuleMakerA, * bPCTRuleMakerB;
BPCTPriceRuleMaster * bPCTPriceRuleMasterA, * bPCTPriceRuleMasterB;

// agentForge step 3c
int bPCTAgentAInputNodeNumber, bPCTAgentAHiddenNodeNumber,
   bPCTAgentAOutputNodeNumber,
   bPCTAgentBInputNodeNumber, bPCTAgentBHiddenNodeNumber,
   bPCTAgentBOutputNodeNumber,

   bPCTPatternNumberInVerificationSet,
```

```
    bPCTPatternNumberInTrainingSet, bPCTAgentsAreDisplayingData,
    usingRandomOrderInBPCTLearning,
    longTermLearningInBPCT_OnlyWithCompleteTrainingSet,
    useOutputsAsTargetsInBPCT_RelearningScheme;
  float bPCTWeightRange, bPCTEps, bPCTAlpha;

}


+ createBegin: aZone;
- createEnd;
- buildObjects;
- buildActions;
- activateIn: swarmContext;

- increaseCurrentDayNumber;
- (int) getCurrentDay;
- getAgentArrayIndex;
// - getForecastAgentArrayIndex;
- getBookArrayIndex;
- getBPCTAgentAArrayIndex;
- getBPCTAgentBArrayIndex;
- getBook: (int) n;
- (int) getBookNumber;

  // agentForge step 4
- (int) getBPCTAgentANumber;
- getBPCTAgentAArrayIndex;
- (int) getBPCTAgentBNumber;
- getBPCTAgentBArrayIndex;
- getAgentList;

- getRandomAgentList;
- getArbitrageurAgentList;
- getMarketImitatingAgentList;
- getLocallyImitatingAgentList;
- getStopLossAgentList;
- getANNForecastAppAgentList;
- getBPCTAgentAList;
- getBPCTAgentBList;
- getAvatarAgentList;
- getForecastingAgent: (int) n;
- getIndexCalculator;
- openProbeTo: (int) ag;
- setDayStateBeforeOpening;
- setDayStateOpening;

@end
```

// **ObserverSwarm.m**

```
#import "ObserverSwarm.h"
#import <activity.h>
#import <simtoolsgui.h>

@implementation ObserverSwarm

+ createBegin: aZone
{
  ObserverSwarm *obj;
  id <ProbeMap> probeMap;

  // Superclass createBegin to allocate ourselves.

  obj = [super createBegin: aZone];

  // Fill in the relevant parameters.

  obj->displayFrequency                 = 303;

  obj->stopAtDayNumber                  = 500; // to stop the program (if != 0)
  obj->displayPreviousDayMean           = 1;
  obj->showPriceGraph                   = 1;
  obj->savePriceData                    = 1;
  obj->showBookGraph                    = 0;
  obj->saveBookData                     = 0;
  obj->showVolumesGraph                 = 0;
  obj->saveVolumesData                  = 0;
  obj->showAgentWealthGraph             = 1;
  obj->saveAgentWealthData              = 0;
  obj->showForecastingAgentGraph        = 0;
  obj->saveForecastingData              = 0;

  //agentForge step 11b
  obj->numberOfTheBPCTAgentA_ToBeObservedDirectly = 1;
  obj->saveBPCTAgentAData = 1;
  obj->numberOfTheBPCTAgentB_ToBeObservedDirectly = 1;
  obj->saveBPCTAgentBData = 1;

  // to build a customized probe map
  // without a probe map, the default is to show all variables and messages
  // here we choose to customize the appearance of the probe, to give a nicer
  // interface

  probeMap = [EmptyProbeMap createBegin: aZone];
  [probeMap setProbedClass: [self class]];
  probeMap = [probeMap createEnd];
```

```
// add variables to be probed

[probeMap addProbe: [probeLibrary getProbeForVariable:
        "displayFrequency"          inClass: [self class]]];
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "stopAtDayNumber"           inClass: [self class]]];

[probeMap addProbe: [probeLibrary getProbeForVariable:
        "displayPreviousDayMean"    inClass: [self class]]];
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "showPriceGraph"            inClass: [self class]]];
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "savePriceData"             inClass: [self class]]];
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "showBookGraph"             inClass: [self class]]];
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "saveBookData"              inClass: [self class]]];
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "showVolumesGraph"          inClass: [self class]]];
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "saveVolumesData"           inClass: [self class]]];

[probeMap addProbe: [probeLibrary getProbeForVariable:
        "showAgentWealthGraph"      inClass: [self class]]];
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "saveAgentWealthData"       inClass: [self class]]];

[probeMap addProbe: [probeLibrary getProbeForVariable:
        "showForecastingAgentGraph"  inClass: [self class]]];
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "saveForecastingData"       inClass: [self class]]];

//agentForge step 11c
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "numberOfTheBPCTAgentA_ToBeObservedDirectly"
                            inClass: [self class]]];
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "saveBPCTAgentAData"        inClass: [self class]]];

[probeMap addProbe: [probeLibrary getProbeForVariable:
        "numberOfTheBPCTAgentB_ToBeObservedDirectly"
                            inClass: [self class]]];
[probeMap addProbe: [probeLibrary getProbeForVariable:
        "saveBPCTAgentBData"        inClass: [self class]]];

// set our custom probeMap into the probeLibrary as the default probe for
// the ObserverSwarm class
```

```
  [probeLibrary setProbeMap: probeMap For: [self class]];

  return obj;
}

- createEnd
{
  return [super createEnd];
}


- buildObjects
{
  int i;
  [super buildObjects];

  modelSwarm = [ModelSwarm create: self];

  // to create probe objects on the model and the observer (self, here)
  // ARCHIVED to allow the "Save" button to operate

  CREATE_ARCHIVED_PROBE_DISPLAY (modelSwarm);
  CREATE_ARCHIVED_PROBE_DISPLAY (self);

  // we pause here to allow the parameters to be changed.

  [controlPanel setStateStopped];
  //2002-10-01 for a continous running decomment the line below, and comment the line
above
  //[controlPanel setStateRunning];


  // The system will wait until the user hits "Start" or "Next"
  // on the control panel

  [modelSwarm buildObjects];

  // checking the consistence of the displayFrequency with the agentNumber:
  // for display reasons it is necessary to update the graphic widgets
  // after all agent actions (mainly the BPCT ones, that calculate their
  // targets at the end of a day and their inputs at the beginning of the day)
  // so the displays are updated at displayFrequency-1
  // if displayFrequency=1, in the agentNumber steps of a day we have the
  // new BPCT input and output updated at the first step and the BPCT target
  // updated at the last
  // if we adopt a displayFrequency multiple of angentNumber we have the
  // apparent coincidence of these values (##)
```

```
  if(displayFrequency !=1 && displayFrequency %
    [[modelSwarm getAgentList] getCount] != 0)
    {
     printf("displayFrequency must be 1 or multiple of agentNumber.\n");
     exit(0);
    }

//van  theBook=[modelSwarm getBook];
//van  forecastingAgent=[modelSwarm getForecastingAgent];

indexCalculator=[modelSwarm getIndexCalculator];
  // Observer display objects.

  // The current price graph
  priceGraph = [EZGraph createBegin: self];
  SET_WINDOW_GEOMETRY_RECORD_NAME (priceGraph); // to allow "Save"

  if(savePriceData==1)[priceGraph setFileOutput: (BOOL) 1];
                  // to send the data also to a file
  if(showPriceGraph==0)[priceGraph setGraphics: (BOOL) 0];

  [priceGraph setTitle: "Current price"];
  [priceGraph setAxisLabelsX: "Ticks x days." Y: "Price"];
  priceGraph = [priceGraph createEnd];
  // that above is the old way of creating an EZgraph; now a unique
  // methos exists

  // pay attention: the lines below cannot be placed before createEnd

bookNumber=[modelSwarm getBookNumber];

for (i=1;i<=bookNumber;i++)
{
  oneBook=[modelSwarm getBook: i];
  char bookname [30];
  sprintf(bookname,"Current price book %d",i);
  [priceGraph createSequence: bookname withFeedFrom: oneBook
                                andSelector: M(getPrice)];
  if(displayPreviousDayMean==1){
  char bookname2 [30];
  sprintf(bookname2,"Day-1 mean p. book %d",i);
  [priceGraph createSequence: bookname2 withFeedFrom: oneBook
                                andSelector: M(getMeanPrice)];
}}

  [priceGraph createSequence: "Index Value" withFeedFrom: indexCalculator
                                andSelector: M(getIndexValue)];
```

```
  // The book graph


  bookGraph = [EZGraph createBegin: self];
  SET_WINDOW_GEOMETRY_RECORD_NAME (bookGraph); // to allow "Save"

  if(saveBookData==1)[bookGraph setFileOutput: (BOOL) 1];
                // to send the data also to a file
  if(showBookGraph==0)[bookGraph setGraphics: (BOOL) 0];

  [bookGraph setTitle: "Book log"];
  [bookGraph setAxisLabelsX: "Ticks x days." Y: "Sell and Buy Orders in Log."];
  bookGraph = [bookGraph createEnd];
  // that above is the old way of creating an EZgraph; now a unique
  // methos exists

  // pay attention: the lines below cannot be placed before createEnd

 for (i=1;i<=bookNumber;i++)
 {
  oneBook=[modelSwarm getBook: i];
  char bookname [30];
  sprintf(bookname,"Sell orders book %d",i);

  [bookGraph createSequence: bookname withFeedFrom: oneBook
                              andSelector: M(getSellOrderNumber)];
  char bookname2 [30];
  sprintf(bookname2,"Buy orders book %d",i);
  [bookGraph createSequence: bookname2  withFeedFrom: oneBook
                              andSelector: M(getBuyOrderNumber)];
 }


  // The Volumes graph


  volumesGraph = [EZGraph createBegin: self];
  SET_WINDOW_GEOMETRY_RECORD_NAME (volumesGraph); // to allow
"Save"

  if(saveVolumesData==1)[volumesGraph setFileOutput: (BOOL) 1];
                // to send the data also to a file
  if(showVolumesGraph==0)[volumesGraph setGraphics: (BOOL) 0];

  [volumesGraph setTitle: "Volumes log"];
  [volumesGraph setAxisLabelsX: "Ticks x days." Y: "Volumes in Log."];
  volumesGraph = [volumesGraph createEnd];
```

```
  // that above is the old way of creating an EZgraph; now a unique
  // methos exists

  // pay attention: the lines below cannot be placed before createEnd


for (i=1;i<=bookNumber;i++)
{
  oneBook=[modelSwarm getBook: i];
  char bookname [30];
  sprintf(bookname,"Price Volumes book %d",i);

  [volumesGraph createSequence: bookname withFeedFrom: oneBook
                               andSelector: M(getPriceVolumes)];
  char bookname2 [30];
  sprintf(bookname2,"Qty Volumes book %d",i);
  [volumesGraph createSequence: bookname2  withFeedFrom: oneBook
                               andSelector: M(getQuantityVolumes)];
}



  // The agent's wealth graph


  agentWealthGraph = [EZGraph createBegin: self];
  SET_WINDOW_GEOMETRY_RECORD_NAME (agentWealthGraph); // to allow
"Save"

  if(showAgentWealthGraph==0)[agentWealthGraph setGraphics: (BOOL) 0];

  if(saveAgentWealthData==1)[agentWealthGraph setFileOutput: (BOOL) 1];
                  // to send the data also to a file

  [agentWealthGraph setTitle: "Agent's wealth"];
  [agentWealthGraph setAxisLabelsX: "Ticks x days."
                                 Y: "Wealth."];
  agentWealthGraph = [agentWealthGraph createEnd];
  // that above is the old way of creating an EZgraph; now a unique
  // methos exists

  // pay attention: the lines below cannot be placed before createEnd

  [agentWealthGraph createMinSequence: "MinWealth (all)"
                withFeedFrom: [modelSwarm getAgentList]
                andSelector: M(getWealthAtMeanDailyPrice)];
  [agentWealthGraph createAverageSequence: "MeanWealth (all)"
                withFeedFrom: [modelSwarm getAgentList]
```

```
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (all)"
            withFeedFrom: [modelSwarm getAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (r.)"
            withFeedFrom: [modelSwarm getRandomAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (r.)"
            withFeedFrom: [modelSwarm getRandomAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (r.)"
            withFeedFrom: [modelSwarm getRandomAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
            //nuovo
[agentWealthGraph createMinSequence: "MinWealth (av.)"
            withFeedFrom: [modelSwarm getAvatarAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (av.)"
            withFeedFrom: [modelSwarm getAvatarAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (av.)"
            withFeedFrom: [modelSwarm getAvatarAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
            //fine nuovo

[agentWealthGraph createMinSequence: "MinWealth (arb.)"
            withFeedFrom: [modelSwarm getArbitrageurAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (arb.)"
            withFeedFrom: [modelSwarm getArbitrageurAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (arb.)"
            withFeedFrom: [modelSwarm getArbitrageurAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
            //fine nuovo


[agentWealthGraph createMinSequence: "MinWealth (m.i.)"
            withFeedFrom: [modelSwarm getMarketImitatingAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (m.i.)"
            withFeedFrom: [modelSwarm getMarketImitatingAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (m.i.)"
            withFeedFrom: [modelSwarm getMarketImitatingAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (l.i.)"
            withFeedFrom: [modelSwarm getLocallyImitatingAgentList]
```

```
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (l.i.)"
            withFeedFrom: [modelSwarm getLocallyImitatingAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (l.i.)"
            withFeedFrom: [modelSwarm getLocallyImitatingAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (s.l.)"
            withFeedFrom: [modelSwarm getStopLossAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (s.l.)"
            withFeedFrom: [modelSwarm getStopLossAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (s.l.)"
            withFeedFrom: [modelSwarm getStopLossAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (ann)"
            withFeedFrom: [modelSwarm getANNForecastAppAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (ann)"
            withFeedFrom: [modelSwarm getANNForecastAppAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (ann)"
            withFeedFrom: [modelSwarm getANNForecastAppAgentList]
            andSelector: M(getWealthAtMeanDailyPrice)];

[agentWealthGraph createMinSequence: "MinWealth (bPCTA)"
            withFeedFrom: [modelSwarm getBPCTAgentAList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (bPCTA)"
            withFeedFrom: [modelSwarm getBPCTAgentAList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (bPCTA)"
            withFeedFrom: [modelSwarm getBPCTAgentAList]
            andSelector: M(getWealthAtMeanDailyPrice)];

[agentWealthGraph createMinSequence: "MinWealth (bPCTB)"
            withFeedFrom: [modelSwarm getBPCTAgentBList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (bPCTB)"
            withFeedFrom: [modelSwarm getBPCTAgentBList]
            andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (bPCTB)"
            withFeedFrom: [modelSwarm getBPCTAgentBList]
            andSelector: M(getWealthAtMeanDailyPrice)];
```

```
  //agentForge step 11d (before this point)

  // The forecastingAgent graph


  forecastingAgentGraph = [EZGraph createBegin: self];
  SET_WINDOW_GEOMETRY_RECORD_NAME (forecastingAgentGraph); // to
allow "Save"

  if(saveForecastingData==1)[forecastingAgentGraph setFileOutput: (BOOL) 1];
                                // to send the data also to a file
  if(showForecastingAgentGraph==0)[forecastingAgentGraph setGraphics: (BOOL) 0];

  [forecastingAgentGraph setTitle: "Forecasting agent log"];
  [forecastingAgentGraph setAxisLabelsX:
                  "Ticks x days." Y: "Estimates and errors."];
  forecastingAgentGraph = [forecastingAgentGraph createEnd];
  // that above is the old way of creating an EZgraph; now a unique
  // methos exists

  // pay attention: the lines below cannot be placed before createEnd

 for (i=1;i<=bookNumber;i++)
 {
  oneBook=[modelSwarm getBook: i];
  oneForecastingAgent=[modelSwarm getForecastingAgent: i];
  char bookname [30];
  sprintf(bookname,"Day-1 mean p. (index) book %d",i);

  [forecastingAgentGraph createSequence:  bookname
                              withFeedFrom: oneBook
                          andSelector: M(getMeanPriceIndex)];
  char bookname1 [30];
  sprintf(bookname1,"Day-1 estimate (index) book %d",i);

  [forecastingAgentGraph createSequence: bookname1
                              withFeedFrom: oneForecastingAgent
                          andSelector: M(getPastEstimateIndex)];
  char bookname2 [30];
  sprintf(bookname2,"Back propagation error book %d",i);

  [forecastingAgentGraph createSequence: bookname2
          withFeedFrom: oneForecastingAgent
          andSelector: M(getBackPropagationErrorInTrainingSet)];
  char bookname3 [30];
  sprintf(bookname3,"Proportional error book %d",i);

  [forecastingAgentGraph createSequence: bookname3
```

```
        withFeedFrom: oneForecastingAgent
        andSelector: M(getProportionalErrorInTrainingSet)];
}



 // --- BPCT ---

 // agent A

 if([modelSwarm getBPCTAgentANumber] == 0)
                numberOfTheBPCTAgentA_ToBeObservedDirectly=0;

 // this may be 0 if we do not have any agent (above) or if we choose to
 // have no display
 if (numberOfTheBPCTAgentA_ToBeObservedDirectly > 0)
                        // the value comes from probe
 {
 // to identify the address of the chosen agent

 [[modelSwarm getBPCTAgentAArrayIndex] setOffset:
                        numberOfTheBPCTAgentA_ToBeObservedDirectly-1];
 agentA = [[modelSwarm getBPCTAgentAArrayIndex] get];
 agentAInterface = [agentA getInterface];

  // agent A graph
  bPCTAgentA_Graph = [EZGraph createBegin: [self getZone]];
  SET_WINDOW_GEOMETRY_RECORD_NAME (bPCTAgentA_Graph); // to
allow "Save"
  [bPCTAgentA_Graph setTitle: "BPCTAgent A data"];
  [bPCTAgentA_Graph setAxisLabelsX:
   "Ticks x days" Y: "Value"];
  bPCTAgentA_Graph = [bPCTAgentA_Graph createEnd];

  if(saveBPCTAgentAData==1)[bPCTAgentA_Graph setFileOutput: (BOOL) 1];
                        // to send the data also to a file

  [bPCTAgentA_Graph createSequence: "meanPrice1_A"
                        withFeedFrom: agentAInterface
                        andSelector: M(getMeanPrice1)];

  [bPCTAgentA_Graph createSequence: "liquidityQuantity_out_A"
                        withFeedFrom: agentAInterface
                        andSelector: M(getLiquidityQuantity_out)];
  [bPCTAgentA_Graph createSequence: "liquidityQuantity_target_A"
                        withFeedFrom: agentAInterface
                        andSelector: M(getLiquidityQuantity_target)];
  [bPCTAgentA_Graph createSequence: "shareQuantity_out_A"
```

```
                        withFeedFrom: agentAInterface
                        andSelector: M(getShareQuantity_out)];
  [bPCTAgentA_Graph createSequence: "shareQuantity_target_A"
                        withFeedFrom: agentAInterface
                        andSelector: M(getShareQuantity_target)];
  [bPCTAgentA_Graph createSequence: "buySell_out_A"
                        withFeedFrom: agentAInterface
                        andSelector: M(getBuySell_out)];
  [bPCTAgentA_Graph createSequence: "buySell_target_A"
                        withFeedFrom: agentAInterface
                        andSelector: M(getBuySell_target)];

  }

  // agent B

  if([modelSwarm getBPCTAgentBNumber] == 0)
                  numberOfTheBPCTAgentB_ToBeObservedDirectly=0;

  // this may be 0 if we do not have any agent (above) or if we choose to
  // have no display
  if (numberOfTheBPCTAgentB_ToBeObservedDirectly > 0)
                        // the value comes from probe
  {
  // to identify the address of the chosen agent

  [[modelSwarm getBPCTAgentBArrayIndex] setOffset:
                        numberOfTheBPCTAgentB_ToBeObservedDirectly-1];
  agentB = [[modelSwarm getBPCTAgentBArrayIndex] get];
  agentBInterface = [agentB getInterface];

   // agent B graph
   bPCTAgentB_Graph = [EZGraph createBegin: [self getZone]];
   SET_WINDOW_GEOMETRY_RECORD_NAME (bPCTAgentB_Graph); // to allow
"Save"
   [bPCTAgentB_Graph setTitle: "BPCTAgent B data"];
   [bPCTAgentB_Graph setAxisLabelsX:
    "Ticks x days" Y: "Value"];
   bPCTAgentB_Graph = [bPCTAgentB_Graph createEnd];

   if(saveBPCTAgentBData==1)[bPCTAgentB_Graph setFileOutput: (BOOL) 1];
                        // to send the data also to a file

   [bPCTAgentB_Graph createSequence: "meanPrice1_B"
                        withFeedFrom: agentBInterface
                        andSelector: M(getMeanPrice1)];

   [bPCTAgentB_Graph createSequence: "liquidityQuantity_out_B"
```

```
                withFeedFrom: agentBInterface
                andSelector: M(getLiquidityQuantity_out)];
  [bPCTAgentB_Graph createSequence: "liquidityQuantity_target_B"
                withFeedFrom: agentBInterface
                andSelector: M(getLiquidityQuantity_target)];
  [bPCTAgentB_Graph createSequence: "shareQuantity_out_B"
                withFeedFrom: agentBInterface
                andSelector: M(getShareQuantity_out)];
  [bPCTAgentB_Graph createSequence: "shareQuantity_target_B"
                withFeedFrom: agentBInterface
                andSelector: M(getShareQuantity_target)];
  [bPCTAgentB_Graph createSequence: "closingPriceWealth_out_B"
                withFeedFrom: agentBInterface
                andSelector: M(getClosingPriceWealth_out)];
  [bPCTAgentB_Graph createSequence: "closingPriceWealth_target_B"
                withFeedFrom: agentBInterface
                andSelector: M(getClosingPriceWealth_target)];
  [bPCTAgentB_Graph createSequence: "forecastedPriceWealth_out_B"
                withFeedFrom: agentBInterface
                andSelector: M(getForecastedPriceWealth_out)];
  [bPCTAgentB_Graph createSequence: "forecastedPriceWealth_target_B"
                withFeedFrom: agentBInterface
                andSelector: M(getForecastedPriceWealth_target)];
 [bPCTAgentB_Graph createSequence: "buySell_out_B"
                withFeedFrom: agentBInterface
                andSelector: M(getBuySell_out)];
  [bPCTAgentB_Graph createSequence: "buySell_target_B"
                withFeedFrom: agentBInterface
                andSelector: M(getBuySell_target)];

 }


 // agentForge step 11 (before this point, add ...)

 // to check (pressing only once the next or the start button) the initial
 //settings, uncomment the following line
 //  [controlPanel setStateStopped];

 return self;
}

- buildActions
{
 [super buildActions];

 // model schedule.
```

```
[modelSwarm buildActions];

//agentForge step 11e
bPCTPatternNumberInVerificationSet=0;
if(agentA != nil)bPCTPatternNumberInVerificationSet=
                    [agentA getBPCTPatternNumberInVerificationSet];
if(agentB != nil)bPCTPatternNumberInVerificationSet=
                    [agentB getBPCTPatternNumberInVerificationSet];
// this variable is used below for a consistence check, as we need
// here to have a verification set on length 1 (the same valued is used also
// by the other BPCT agent; we have to repeat the assign operation in case
// of missing categories of agents)

// ActionGroup for Observer display

displayActions = [ActionGroup create: self];

// to update probes
[displayActions createActionTo: probeDisplayManager message: M(update)];
[displayActions createActionTo: actionCache message: M(doTkEvents)];

[displayActions createActionTo: priceGraph message: M(step)];
 if (showBookGraph==1)
 [displayActions createActionTo: bookGraph  message: M(step)];
 if (showVolumesGraph==1)
 [displayActions createActionTo: volumesGraph  message: M(step)];
if (showAgentWealthGraph==1)
[displayActions createActionTo: agentWealthGraph  message: M(step)];
 if (showForecastingAgentGraph==1)
 [displayActions createActionTo: forecastingAgentGraph  message: M(step)];

[displayActions createActionTo: self message: M(checkToStop)];

// Display schedule. Note that the repeat interval is set by our
// own Swarm data structure. The display is frequently the slowest part of a
// simulation, when it is redraw less frequently things are faster

displaySchedule = [Schedule createBegin: self];
// we can use here a display frequency lower (e.g.1) than the number of steps
// in the model (step number = agentNumber) to display the prices of each
// step-tick
[displaySchedule setRepeatInterval: displayFrequency];
displaySchedule = [displaySchedule createEnd];

[displaySchedule at: displayFrequency-1 createAction: displayActions];
                                // see ## comment above

//agentForge step 11f
```

```
  // BPCT

  if (numberOfTheBPCTAgentA_ToBeObservedDirectly > 0)
  // the following if condition is related to this CT use of the program, to
  // avoid nonsens
   if (bPCTPatternNumberInVerificationSet == -1)
     [displaySchedule at: displayFrequency-1  // see ## comment above
                    createActionTo: bPCTAgentA_Graph message: M(step)];

  if (numberOfTheBPCTAgentB_ToBeObservedDirectly > 0)
  // the following if condition is related to this CT use of the program, to
  // avoid nonsens
   if (bPCTPatternNumberInVerificationSet == -1)
     [displaySchedule at: displayFrequency-1  // see ## comment above
                    createActionTo: bPCTAgentB_Graph message: M(step)];

  return self;
}

- activateIn: swarmContext
{
// activateIn: - to activate the schedules to make them ready to run.

  [super activateIn: swarmContext];

  [modelSwarm activateIn: self];

  [displaySchedule activateIn: self];

  return [self getSwarmActivity];
}

  // to check for the stopping conditions
- checkToStop
{

  // if stopAtDayNumber is left to 0, the program will never stop
  // this stop occurs after the first tick of the time in modelSwarm,
  // but this fact does not affect the results in the observerSwarm
  // being the stop performed before the graph update

  if (stopAtDayNumber !=0 &&
     stopAtDayNumber <= [modelSwarm getCurrentDay])
   {
     printf("Stopping at day number %4d\n",
                          [modelSwarm getCurrentDay]);

     // we can restart by pressing "Start" or "Next", but the simulation
```

```
    // will run for displayFrequency ticks and then it will stop again

    [controlPanel setStateStopped];
    // 2002-10-01 for a continous running decomment the line below, and comment the
line above
    // [controlPanel setStateQuit];

  }else{
        printf("Day number %4d is finished\n",
                        [modelSwarm getCurrentDay]);
  }

  return self;
}


@end
```

// **ObserverSwarm.h**

```
#import "ModelSwarm.h"
#import "Book.h"
#import "ForecastingAgent.h"
#import "IndexCalculator.h"

  //agentForge step 10b
#import "BPCTAgentA.h"
#import "BPCTAgentAInterface.h"
#import "BPCTAgentB.h"
#import "BPCTAgentBInterface.h"

#import <simtoolsgui/GUISwarm.h>
#import <analysis.h> // to use EZgraph


@interface ObserverSwarm: GUISwarm
{
  int displayFrequency;       // freq. of update in the
                              // Observer widgets
  int showPriceGraph, showBookGraph, showVolumesGraph, showAgentWealthGraph,
showForecastingAgentGraph,
      displayPreviousDayMean, savePriceData, saveForecastingData,
    saveBookData,saveVolumesData,saveAgentWealthData, bookNumber,

  //agentForge step 10c
      numberOfTheBPCTAgentA_ToBeObservedDirectly,
      saveBPCTAgentAData,
      numberOfTheBPCTAgentB_ToBeObservedDirectly,
      saveBPCTAgentBData,

      bPCTPatternNumberInVerificationSet;

  id displayActions;          // schedule data structures
  id displaySchedule;

  // agentForge step 10
  id <EZGraph> priceGraph, volumesGraph, bookGraph, agentWealth,
agentWealthGraph,
              forecastingAgentGraph, bPCTAgentA_Graph, bPCTAgentB_Graph;

  int stopAtDayNumber;

  ModelSwarm *modelSwarm;             // the Swarm containing our model
  Book * oneBook;
  IndexCalculator * indexCalculator;
  ForecastingAgent * oneForecastingAgent;
```

```
  //agentForge step 10d
  BPCTAgentA * agentA;
  BPCTAgentAInterface * agentAInterface;
  BPCTAgentB * agentB;
  BPCTAgentBInterface * agentBInterface;

}

+ createBegin: aZone;
- createEnd;
- buildObjects;
- buildActions;
- activateIn: swarmContext;

- checkToStop;

@end
```

// **Book.m**

```
#import "Book.h"
#include <time.h>
#import "BasicSumAgent.h" // this is placed here to avoid a circular call

  // functions

void message(id <Index> agentArrayIndex, int n, float p, int bn)
{
   BasicSumAgent * anAgent;
   [agentArrayIndex setOffset: n-1];
   anAgent=[agentArrayIndex get];
   [anAgent setConfirmationOfExecutedPrice: p inBook: bn];
   return;
}

void setLocal(Matrix2 * loc, float p)
{
   int i;
   if (p==0)return;
   for (i=[loc getRows]-2;i>=0;i--)[loc R:i+1 C:0 setFrom: [loc R:i C:0]];
   [loc R:0 C:0 setFrom: p];
   return;
}

int getLocal(Matrix2 * loc)
{
   int i, tot;
   tot=0;
   for (i=0;i<[loc getRows];i++)
      {
      if([loc R:i C:0]>0)tot++;
      if([loc R:i C:0]<0)tot--;
      }
   return tot;
}

@implementation Book

  // we set the book number
- setNumber: (int) n
{
  theBookNumber = n;
  return self;
}

- setAgentArrayIndex: i
```

```
{
  agentArrayIndex = i;
  return self;
}

- setAgentNumber: (int) n
{
  agentNumber=n;
  return self;
}

- setMaxOrderQuantity: (int) m
{
  maxOrderQuantity=m;
  return self;
}

- setMeanPriceHistoryLength: (int) l
{
  meanPriceHistoryLength=l;
  return self;
}

- setPriceVolumesHistoryLength: (int) l
{
  priceVolumesHistoryLength=l;
  return self;
}

- setQuantityVolumesHistoryLength: (int) l
{
  quantityVolumesHistoryLength=l;
  return self;
}

- setLocalHistoryLength: (int) l
{
  localHistoryLength=l;
  return self;
}

- setPrinting: (int) p
{
  printing=p;
  return self;
}

- createEnd
```

```
{
  int i;
  [super createEnd];

  executedPrice=1; // this is the starting price; it seems to be
                   // not relevant at all for the behavior of the model
  meanPrice=1;
  previousClosingPrice=executedPrice;
  currentMeanPrice=0;
  count=0;
  quantityVolumes=0;
  priceVolumes=0;
  openPrice=0.0;
  lowPrice=1000000000.0;
  highPrice=0.0;

  // the book works on the basis of two matrixes containing sell
  // order in increasing order or buy order in decreasing order
  // (in col 1 we have the orders; in col 2 the number of the agent
  // placing the order)

  // if an order obtains an immediate matching, it is not filed

  // the worse situation is that of having all the order on one side of
  // the market and all the agents ordering; so the rows of the two
  // matrixes must be equal to the number of the agents

  sellOrderStorehouse=[Matrix2 createBegin: [self getZone]];
  [sellOrderStorehouse setDimensionRows: agentNumber*maxOrderQuantity
                       Cols: 2 Code: 1];
  sellOrderStorehouse=[sellOrderStorehouse createEnd];

  buyOrderStorehouse=[Matrix2 createBegin: [self getZone]];
  [buyOrderStorehouse  setDimensionRows: agentNumber*maxOrderQuantity
                       Cols: 2 Code: 2];
  buyOrderStorehouse=[buyOrderStorehouse   createEnd];

  meanPriceHistory=[Matrix2 createBegin: [self getZone]];
  [meanPriceHistory setDimensionRows: meanPriceHistoryLength
                        Cols: 1 Code: 3];
  meanPriceHistory=[meanPriceHistory createEnd];
  // mean prices will be stored by rows; now we fill all the r. with
  // the starting mean price
  for (i=0;i<=meanPriceHistoryLength-1;i++)
     [meanPriceHistory R:i C:0 setFrom: meanPrice];

  // the same for volumes
  priceVolumesHistory=[Matrix2 createBegin: [self getZone]];
```

```
  [priceVolumesHistory setDimensionRows: priceVolumesHistoryLength
                        Cols: 1 Code: 5];
  priceVolumesHistory=[priceVolumesHistory createEnd];

  for (i=0;i<=priceVolumesHistoryLength-1;i++)
     [priceVolumesHistory R:i C:0 setFrom: priceVolumes];


  quantityVolumesHistory=[Matrix2 createBegin: [self getZone]];
  [quantityVolumesHistory setDimensionRows: quantityVolumesHistoryLength
                        Cols: 1 Code: 6];
  quantityVolumesHistory=[quantityVolumesHistory createEnd];

  for (i=0;i<=quantityVolumesHistoryLength-1;i++)
     [quantityVolumesHistory R:i C:0 setFrom: quantityVolumes];


  localHistory=[Matrix2 createBegin: [self getZone]];
  [localHistory setDimensionRows: localHistoryLength
                        Cols: 1 Code: 4];
  localHistory=[localHistory createEnd];
  // local actions will be stored by rows; we fill all the r. with
  // 0, i.e. 'no action', automatically, as a byproduct of the
  // setDimensionRows:Cols: method

  return self;
}

 // at the end of each day
- setMeanPrice{
 if (count>0) meanPrice=currentMeanPrice/count; // otherwise we keep
                                    // previous value
 return self;
}

- setNAheadForecasting: (int) na
{
 nAheadForecasting = na;
 return self;
}

 // at the beginning of each day
- setClean{
 int i;
 sellOrderNumber=0; buyOrderNumber=0;

 // meanPriceHistory in row 0 contains the t-1 meanPrice;
 //                  in row 1 contains the t-2 meanPrice;
```

```
  //                etc.
  for (i=meanPriceHistoryLength-2;i>=0;i--)
                [meanPriceHistory R:i+1 C:0 setFrom:
                                [meanPriceHistory R:i C:0]];
  [meanPriceHistory R:0 C:0 setFrom: meanPrice];

  for (i=priceVolumesHistoryLength-2;i>=0;i--)
                [priceVolumesHistory R:i+1 C:0 setFrom:
                                [priceVolumesHistory R:i C:0]];
  [priceVolumesHistory R:0 C:0 setFrom: priceVolumes];

  for (i=quantityVolumesHistoryLength-2;i>=0;i--)
                [quantityVolumesHistory R:i+1 C:0 setFrom:
                                [quantityVolumesHistory R:i C:0]];
  [quantityVolumesHistory R:0 C:0 setFrom: quantityVolumes];

  previousClosingPrice=executedPrice; // the last one of 'yesterday'
  currentMeanPrice=0;
  count=0;
  quantityVolumes=0;
  priceVolumes=0;
  openPrice=0.0;
  lowPrice=1000000000.0;
  highPrice=0.0;
  return self;
}

  // receiving an order before opening from an agent
- setOrderBeforeOpeningFromAgent: (int) n atPrice: (float) p
{
   int number;
   number = n;
   price = p;

   if(printing==1)
   printf(
   "The book #%3d received a before opening order from agent #%3d at price %7.4f\n",
         theBookNumber, number, price);

   // local history
   setLocal(localHistory, price);

   // if price==0 no action required, but sending a 0.0 message to the agent
   if(price==0) message(agentArrayIndex, number, 0.0, theBookNumber);

   // the agent is selling at min price '-price'
   if(price<0)      {
                 message(agentArrayIndex, number, 0.0, theBookNumber);
```

```
                // filing the sell order, in increasing order
                sellOrderNumber++;
                [sellOrderStorehouse fileIncreasingP: -price
                                andN: (float) number
                        usingAsNumberOfRows: sellOrderNumber];
                if(printing==1)
                [sellOrderStorehouse printNRows: sellOrderNumber];
                [sellOrderStorehouse printNRowsFileS: sellOrderNumber ofBook:
theBookNumber];
                }

   // the agent is buying at max price 'price'
   if(price>0)      {
                message(agentArrayIndex, number, 0.0, theBookNumber);
                // filing the buy order, in decreasing order
                buyOrderNumber++;
                [buyOrderStorehouse fileDecreasingP: price
                                andN: (float) number
                        usingAsNumberOfRows: buyOrderNumber];
                if(printing==1)
                [buyOrderStorehouse printNRows: buyOrderNumber];
                [buyOrderStorehouse printNRowsFileB: buyOrderNumber ofBook:
theBookNumber];
                }

   return self;
}

  // receiving an order when the market is open
- setOrderFromAgent: (int) n atPrice: (float) p
{

  time_t tiempo;
  char cad[80];
  struct tm *tmPtr;
  int number;
  tiempo = time(NULL);
  tmPtr = localtime(&tiempo);
  strftime( cad, 80, "%d-%m-%Y %H:%M:%S", tmPtr );

  char  webdataFileName [40];
    sprintf(webdataFileName,"bookdata%d.dat",theBookNumber);
  char  webdataFileName2 [40];
    sprintf(webdataFileName2,"lastSingle%d.dat",theBookNumber);


// test da eliminare
//printf("%s\n", webdataFileName);
```

```
number = n;
price = p;

if(printing==1)
printf("The book  #%3d received an order from agent #%3d at price %7.4f\n",
       theBookNumber, number, price);

// local history
setLocal(localHistory, price);

// if price==0 no action required, but sending a 0.0 message to the agent
if(price==0) message(agentArrayIndex, number, 0.0, theBookNumber);

// the agent is selling at min price '-price'
if(price<0) {if (buyOrderNumber>0 &&
            [buyOrderStorehouse R: 0 C: 0] >= -price)
              {executedPrice=[buyOrderStorehouse R: 0 C: 0];
               currentMeanPrice+=executedPrice;
               count++;
               quantityVolumes++;
                                        priceVolumes+=executedPrice;
               //AHIA
               FILE * pFile;

               pFile = fopen (webdataFileName,"a");
               if (pFile == NULL) {
                     perror("cannot open output file1");
                           exit(1);
                     }
                     fprintf(pFile,"%s %9.4f\n",cad,executedPrice);
                     fclose(pFile);
                     //AHIA2
               //AHIA
               FILE * ppFile;

               ppFile = fopen (webdataFileName2,"w");
               if (ppFile == NULL) {
                     perror("cannot open output file2");
                           exit(1);
                     }
                     fprintf(ppFile,"%7.4f\n",executedPrice);
                     fclose(ppFile);
                     //AHIA2
               message(agentArrayIndex, number, -executedPrice, theBookNumber);
               message(agentArrayIndex,(int)[buyOrderStorehouse R: 0 C: 1],
                                        executedPrice, theBookNumber);
               buyOrderNumber--;
               [buyOrderStorehouse shiftRowsDown: buyOrderNumber];
```

```
                if(printing==1)
                [buyOrderStorehouse printNRows: buyOrderNumber];
                [buyOrderStorehouse printNRowsFileB: buyOrderNumber ofBook:
theBookNumber];

                if (openPrice==0.0) openPrice=executedPrice;
                //if (lowPrice==1000000000.0) lowPrice=executedPrice;
                //if (highPrice==0.0) highPrice=executedPrice;
                if (lowPrice>executedPrice) lowPrice=executedPrice;
                if (highPrice<executedPrice) highPrice=executedPrice;

               }

            else {
                message(agentArrayIndex, number, 0.0, theBookNumber);
                // filing the sell order, in increasing order
                sellOrderNumber++;
                [sellOrderStorehouse fileIncreasingP: -price
                                        andN: (float) number
                              usingAsNumberOfRows: sellOrderNumber];
                if(printing==1)
                [sellOrderStorehouse printNRows: sellOrderNumber];
                [sellOrderStorehouse printNRowsFileS: sellOrderNumber ofBook:
theBookNumber];
                }
            }

  // the agent is buying at max price 'price'
  if(price>0) {if (sellOrderNumber>0 &&
               [sellOrderStorehouse R: 0 C: 0] <= price)
                {executedPrice=[sellOrderStorehouse R: 0 C: 0];
                currentMeanPrice+=executedPrice;
                count++;
                quantityVolumes++;
                                              priceVolumes+=executedPrice;
                //AHIA
                FILE * pFile;

                pFile = fopen (webdataFileName,"a");
                if (pFile == NULL) {
                      perror("cannot open output file4");
                            exit(1);
                       }
                       fprintf(pFile,"%s %9.4f\n",cad,executedPrice);
                       fclose(pFile);
                       //AHIA2
                //AHIA
                FILE * ppFile;
```

```
                ppFile = fopen (webdataFileName2,"w");
                if (ppFile == NULL) {
                      perror("cannot open output file5");
                            exit(1);
                       }
                      fprintf(ppFile,"%7.4f\n",executedPrice);
                      fclose(ppFile);
                      //AHIA2

                message(agentArrayIndex, number, executedPrice, theBookNumber);
                message(agentArrayIndex,(int)[sellOrderStorehouse R: 0 C: 1],
                                        -executedPrice, theBookNumber);
                sellOrderNumber--;

                [sellOrderStorehouse shiftRowsDown: sellOrderNumber];

                if(printing==1)
                [sellOrderStorehouse printNRows: sellOrderNumber];
                [sellOrderStorehouse printNRowsFileS: sellOrderNumber ofBook:
theBookNumber];

               }

           else {
                message(agentArrayIndex, number, 0.0, theBookNumber);
                // filing the buy order, in decreasing order
                buyOrderNumber++;
                [buyOrderStorehouse fileDecreasingP: price
                                    andN: (float) number
                           usingAsNumberOfRows: buyOrderNumber];
                if(printing==1)
                [buyOrderStorehouse printNRows: buyOrderNumber];
                [buyOrderStorehouse printNRowsFileB: buyOrderNumber ofBook:
theBookNumber];

                if (openPrice==0.0) openPrice=executedPrice;
                //if (lowPrice==NULL) lowPrice=executedPrice;
                //if (highPrice==NULL) highPrice=executedPrice;
                if (lowPrice>executedPrice) lowPrice=executedPrice;
                if (highPrice<executedPrice) highPrice=executedPrice;
               }

           }

  return self;
}
```

```
- (float) getPrice
{
  return executedPrice;
}

- (float) getMeanPrice
{
  return meanPrice;
}

- (float) getLaggedMeanPrice: (int) lag
{
  return [meanPriceHistory R: lag-1 C: 0];
}

- (float) getMeanPriceIndex
{
  return meanPrice/[meanPriceHistory R: nAheadForecasting C: 0];
}

- (int) getLocalHistory
{
  return getLocal(localHistory);
}

- (float) getSellOrderNumber
{
  return (float) sellOrderNumber;
}

- (float) getBuyOrderNumber
{
  return (float) buyOrderNumber;
}

- (float) getPreviousClosingPrice
{
  return previousClosingPrice;
}

// Nuovo

- (float) getAskPrice
{
  return [buyOrderStorehouse R: 0 C: 0];
}

- (float) getBidPrice
```

```
{
  return [sellOrderStorehouse R: 0 C: 0];
}

- (int) getQuantityVolumes
{
  return (int) quantityVolumes;
}

- (float) getPriceVolumes
{
  return (float) priceVolumes;
}

/*- (float) getFirstBuyOrder
{
  float firstBuyOrder;
  firstBuyOrder=[buyOrderStorehouse R: 0 C: 0];
  return firstBuyOrder;
}

- (float) getFirstSellOrder
{
  float firstSellOrder;
  firstSellOrder=[sellOrderStorehouse R: 0 C: 0];
  return firstSellOrder;
}*/

- (float) getOpenPrice
{
  return openPrice;
}


- (float) getLowPrice
{
  return lowPrice;
}


- (float) getHighPrice
{
  return highPrice;
}

// fine nuovo

@end
```

// **Book.h**

```
#import <objectbase/SwarmObject.h>
#import <collections.h>
#import "Matrix2.h"

@interface Book: SwarmObject
{

  id <Index> agentArrayIndex;
  Matrix2 * sellOrderStorehouse, * buyOrderStorehouse, * meanPriceHistory,
         * localHistory,* priceVolumesHistory,* quantityVolumesHistory;
  int theBookNumber,sellOrderNumber, buyOrderNumber, maxOrderQuantity,
     agentNumber, printing, meanPriceHistoryLength, localHistoryLength,
     nAheadForecasting, quantityVolumes,priceVolumesHistoryLength,
    quantityVolumesHistoryLength;
  float price, executedPrice, meanPrice, currentMeanPrice,
       previousClosingPrice, priceVolumes, openPrice, highPrice,
       lowPrice;
  int count;

}

- createEnd;

- setAgentArrayIndex: i;
- setAgentNumber: (int) n;
- setMaxOrderQuantity: (int) m;
- setMeanPriceHistoryLength: (int) l;
- setPriceVolumesHistoryLength: (int) l;
- setQuantityVolumesHistoryLength: (int) l;
- setLocalHistoryLength: (int) l;
- setPrinting: (int) p;
- setNumber: (int) n;
- setMeanPrice;

- setClean;
- setOrderBeforeOpeningFromAgent: (int) n atPrice: (float) p;
- setOrderFromAgent: (int) n atPrice: (float) p;
- setNAheadForecasting: (int) na;

- (float) getPrice;
- (float) getPreviousClosingPrice;
- (float) getMeanPrice;
- (float) getLaggedMeanPrice: (int) lag; // the methods
                                  // 'getLaggedMeanPrice: 1' and
                                  // 'getMeanPrice give the same
                                  // result
```

```
- (float) getMeanPriceIndex;
- (int) getLocalHistory; // the method returns
                    //                  a positive int if locally
                    // see localHistoryLength) the # of buying decisions
                    // is greater than the # of selling decisions
                    //                  a negative int in the
                    // opposite case
- (float) getSellOrderNumber;
- (float) getBuyOrderNumber;
- (float) getAskPrice;
- (float) getBidPrice;

- (float) getLowPrice;
- (float) getHighPrice;
- (float) getOpenPrice;

// Volumes of each day

- (int) getQuantityVolumes;
- (float) getPriceVolumes;

//- (float) getFirstBuyOrder;
//- (float) getFirstSellOrder;

@end
```

// **BasicSumAgent.m**

```
#import "BasicSumAgent.h"
#import "Book.h"  // this is placed here to avoid a circular call


@implementation BasicSumAgent

  // we set the agent number
- setNumber: (int) n
{
  number = n;
  return self;
}

  // we set the Book number
- setBookNumber: (int) m
{
  bookNumber = m;
  return self;
}

  // we set the Book
- setBook: b
{
  oneBook=b;
  return self;
}

- setAsymmetricBuySellProb: (float) p
{
  asymmetricBuySellProb=p;
  return self;
}

  // the address of the book of the market
- setBookArrayIndex: i
{
  bookArrayIndex = i;
  return self;
}

/*van - setForecastingAgentArrayIndex: i
{
  forecastingAgentArrayIndex = i;
  return self;
}
van*/
```

```
  // how many orders are we placing at each time?
- setMaxOrderQuantity: (int) m
{
  maxOrderQuantity=m;
  return self;
}

  // max number of orders for each agent (humans, arbitrageur, ecc.)
- setMaxOrderQuantity2: (int) m
{
  maxOrderQuantity2=m;
  return self;
}


  // this is a toggle: print / don't print
- setPrinting: (int) p
{
  printing=p;
  return self;
}

  // this is a toggle: delay / don't delay (for random agent)
- setDelay: (int) d
{
  delay=d;
  return self;
}

- createEnd
{
   [super createEnd];

   // initialize
   executedPriceCount=0;

   // keeping data for accounting reasons
   executedPrices=[Matrix2 createBegin: [self getZone]];
   [executedPrices setDimensionRows: maxOrderQuantity2 Cols: 2
                                    Code: 1000+number];
   executedPrices=[executedPrices createEnd];

   shareQuantity=0;
   shareValueAtMeanDailyPrice=0;
   liquidityQuantity=0;
   agentWealthAtMeanDailyPrice=0;
   return self;
```

```
}


- (float) getPrice2: (int) n
{
    Book * aBook;
    [bookArrayIndex setOffset: n-1];
    aBook=[bookArrayIndex get];
    price = [aBook  getPrice];
    return price;
}

- (float) getMeanPrice: (int) n
{
    Book * aBook;
    [bookArrayIndex setOffset: n-1];
    aBook=[bookArrayIndex get];
    price = [aBook  getMeanPrice];
    return price;
}

- setOrderBeforeOpeningFromAgent2: (int) n ofAgent: (int) m atPrice: (float) p
{
    number = m;
    price = p;
    Book * aBook;
    [bookArrayIndex setOffset: n-1];
    aBook=[bookArrayIndex get];
    [aBook setOrderBeforeOpeningFromAgent: number atPrice: price];
    return self;
}

- setOrderFromAgent2: (int) n ofAgent: (int) m atPrice: (float) p
{
    number = m;
    price = p;
    Book * aBook;
    [bookArrayIndex setOffset: n-1];
    aBook=[bookArrayIndex get];
    [aBook setOrderFromAgent: number atPrice: price];
    return self;
}


- act0  // defined for compatibility reasons with current agent
{
  return self;
}
```

```
- act1  // defined for compatibility reasons with current agent
{
 return self;
}


- setConfirmationOfExecutedPrice: (float) p inBook: (int) m
{
 [executedPrices R: executedPriceCount C: 0 setFrom: p];
 [executedPrices R: executedPriceCount C: 1 setFrom: m];
 if(p!=0)++executedPriceCount;
 return self;
}

 // accounting
- act2
{
 float mP, q;
 int i,k;
 shareValueAtMeanDailyPrice=0;
 // accounting operation and display of
 // a message about the operation eventually done: if price is not zero
 // we have been buyer (executedPrice>0) or seller (executedPrice<0)
  if(printing==4)
  for(i=0;i<executedPriceCount;i++)
  printf("I'm agent #%3d and the book number %3f told me: %7.4f\n",
                          number,[executedPrices R:i C:1],[executedPrices R:i C:0]);
 mP=0;

 for (k=1;k<=bookNumber;k++){
  //float mP;
  mP=[self getMeanPrice:k];
  // quantity of shares
  q=0;

  for (i=0;i<executedPriceCount;i++)
  {
  if([executedPrices R:i C:1]==k) {
        if([executedPrices R:i C:0]>0)q++;
        if([executedPrices R:i C:0]<0)q--;
  }}

  shareQuantityVector[k] += q;
  shareQuantity += q;

  shareValueAtMeanDailyPrice += shareQuantityVector[k]*mP;
                          // meanPrice at the end of each "day"
```

```
  }

  // we have CT agent for each book, so every agent operates only on a single book.
  meanOperatingPrice=0; // used in CT
  for (i=0;i<executedPriceCount;i++)
    {
     liquidityQuantity  -= [executedPrices R:i C:0];
     meanOperatingPrice += fabs([executedPrices R:i C:0]);
    }

  if(executedPriceCount != 0) meanOperatingPrice/=executedPriceCount;
  else meanOperatingPrice=mP;

  if(printing==4)
  printf("I'm agent #%3d and the meanOperatingPrice is: %7.4f\n\n",
                              number,meanOperatingPrice);


  agentWealthAtMeanDailyPrice=shareValueAtMeanDailyPrice+liquidityQuantity;
  return self;
}

- getProbe
{
 CREATE_ARCHIVED_PROBE_DISPLAY (self);
 return self;
}

- (float) getWealthAtMeanDailyPrice
{
 return agentWealthAtMeanDailyPrice;
}

- (int) getRandomBookNumber //: (int) randomBookNumber
{
 int randomBookNumber;
 randomBookNumber=[uniformIntRand getIntegerWithMin: 1 withMax:
bookNumber];
 return randomBookNumber;
}


@end
```

// **BasicSumAgent.h**

```
// This is the root Agent in Sum; all operating agents inherit from it,
// but ForecastingAgent; also CTAgent must inherit from it, to have
// accounting capality

#import <objectbase/SwarmObject.h>
#import <random.h>
#import <math.h> // to use fabs etc.
#import "Book.h"
#import <simtoolsgui/GUISwarm.h> // to use
CREATE_ARCHIVED_PROBE_DISPLAY
#import <simtoolsgui.h>          // ""
#import "Matrix2.h"

@interface BasicSumAgent: SwarmObject
{

  int number, maxOrderQuantity, maxOrderQuantity2, iMax, executedPriceCount,
  printing, delay, actingBeforeOpening, actingAtTheTime, bookNumber;
  // we can use at maximum 1000 books, if you want to use more, you need to change
the number in the brackets
  float shareQuantityVector[1000];
  float price,
        asymmetricBuySellProb, buySellSwitch,
        shareQuantity, shareValueAtMeanDailyPrice,
        liquidityQuantity, agentWealthAtMeanDailyPrice,
        meanOperatingPrice;
  Matrix2 * executedPrices;
  Book * oneBook;

   Book * theBook;// da eliminare a fine modifiche
  id <Index> bookArrayIndex;

}

- createEnd;

- setNumber: (int) n;

- setAsymmetricBuySellProb: (float) p;
- setMaxOrderQuantity: (int) m;
- setMaxOrderQuantity2: (int) m;
//- setForecastingAgentArrayIndex: i;
- setBookArrayIndex: i;
- setBookNumber: (int) m;

- (float) getMeanPrice: (int) n;
```

```
- setBook: b;
- setPrinting: (int) p;
- setDelay: (int) d;

- setConfirmationOfExecutedPrice: (float) p inBook: (int) m;


- act0; // defined for compatibilty reasons with currentAgent
- act1;  // "" ""
- act2; // accounting at the end of a day
- getProbe;
- (float) getWealthAtMeanDailyPrice;
- (int) getRandomBookNumber;

- (float) getPrice2: (int) n;
- setOrderBeforeOpeningFromAgent2: (int) n ofAgent: (int) m atPrice: (float) p;
- setOrderFromAgent2: (int) n ofAgent: (int) m atPrice: (float) p;

/*
- (float) getPrice(id <Index> bookArrayIndex, int n);
- (void) setOrderBeforeOpeningFromAgent(id <Index> bookArrayIndex, int n, int
number, float price);
- (void) setOrderFromAgent(id <Index> bookArrayIndex, int n, int number, float price);
*/
/*- (float) getPrice;
- (void) setOrderBeforeOpeningFromAgent;
- (void) setOrderFromAgent;*/
@end
```

// **ArbitrageurAgent.m**

```
#import "ArbitrageurAgent.h"
#import "Book.h"
#import "IndexCalculator.h"

@implementation ArbitrageurAgent

- setArbitrageurOperatingInterval: (float) i
{
  arbitrageurOperatingInterval=i;
  return self;
}

- setArbitrageurGain: (int) i
{
  arbitrageurGain=i;
  return self;
}

- setArbitrageurOperatingIntervalFixed: (int) i
{
  arbitrageurOperatingIntervalFixed=i;
  return self;
}

- setIndexCalculator: t
{
  indexCalculator = t;
  return self;
}


- act0
{
    int i;
    // clearing executedPrices
    executedPriceCount=0;
    for (i=0;i<maxOrderQuantity2;i++) [executedPrices R:i C:0 setFrom:0];
   return self;
}

 // Acting
- act1
{
   int i;
   Book * futureBook;
```

```
  [bookArrayIndex setOffset: bookNumber-1];
  futureBook=[bookArrayIndex get];

  indexValue=[indexCalculator  getIndexValue];

  realFuturePrice= [self  getPrice2: bookNumber];

  //indexSpread=(((IndexValue-realFuturePrice)/IndexValue)+((IndexValue-
realFuturePrice)/realFuturePrice))/2;

  //choose short or long position on future and shares

  if(indexValue>realFuturePrice) {buySellSwitchFuture=1; buySellSwitchShares=-1; }
  else if(indexValue<realFuturePrice) {buySellSwitchFuture=-1;
buySellSwitchShares=1; }
  else {buySellSwitchFuture=0; buySellSwitchShares=0;
       if(printing==1)printf("Arbitrageur doesn't play: there isn't no  spread \n");
      }


  //check counterpart of shares

  if (buySellSwitchShares!=0){
  for (i=1;i<bookNumber;i++)
   {
    Book * aBook;
    [bookArrayIndex setOffset: i-1];
    aBook=[bookArrayIndex get];
    if (buySellSwitchShares==-1){
      if ([aBook getBuyOrderNumber]==0.0){buySellSwitchShares=0;
        if(printing==1)printf("Arbitrageur doesn't play: there isn't counterpart for selling
book %3d.\n",i);
        }
    }else if(buySellSwitchShares==1){
      if ([aBook getSellOrderNumber]==0.0){buySellSwitchShares=0;
        if(printing==1)printf("Arbitrageur doesn't play: there isn't counterpart for buying
book %3d.\n",i);
        }
    }else{buySellSwitchShares=0;}

   }}

   //check counterpart of index

    if (buySellSwitchFuture==-1){
      if ([futureBook getBuyOrderNumber]==0.0){buySellSwitchFuture=0;
        if(printing==1)printf("Arbitrageur doesn't play: there isn't counterpart for selling
Index.\n");
```

```
        }
    }else if(buySellSwitchFuture==1){
      if ([futureBook getSellOrderNumber]==0.0){buySellSwitchFuture=0;
        if(printing==1)printf("Arbitrageur doesn't play: there isn't counterpart for buying
Index.\n");
        }
    }else{buySellSwitchFuture=0;}


// we calculate if arbitrating actions produce a sure gain for arbitrageur agent

   sumOfPrice=0.0;

// here we use a fixed gain (arbitrageurOperatingIntervalFixed==1)

   if ((arbitrageurGain==1)&&(arbitrageurOperatingIntervalFixed==1)) {
    if ((buySellSwitchFuture==1)&&(buySellSwitchShares==-1)){


        if((indexValue -
realFuturePrice)>arbitrageurOperatingInterval){buySellSwitchFuture=1;
buySellSwitchShares=-1; }
        else {buySellSwitchFuture=0; buySellSwitchShares=0;
                if(printing==1)printf("Arbitrageur doesn't play: there isn't no  spread\n");
        }

    }else if((buySellSwitchFuture==-1)&&(buySellSwitchShares==1)){

        if((realFuturePrice -
indexValue)>arbitrageurOperatingInterval){buySellSwitchFuture=-1;
buySellSwitchShares=1; }
        else {buySellSwitchFuture=0; buySellSwitchShares=0;
                if(printing==1)printf("Arbitrageur doesn't play: there isn't no  spread\n");
        }

    }else{buySellSwitchFuture=0;}
   }

// here we use a gain expressed as a ratio of future to index
(arbitrageurOperatingIntervalFixed==0)

  if ((arbitrageurGain==1)&&(arbitrageurOperatingIntervalFixed==0)) {
    if ((buySellSwitchFuture==1)&&(buySellSwitchShares==-1)){
        for (i=1;i<bookNumber;i++)
            {
             Book * aBook;
             [bookArrayIndex setOffset: i-1];
             aBook=[bookArrayIndex get];
```

```
            sumOfPrice+=[aBook  getAskPrice];
          }
        newIndexValue=sumOfPrice/(bookNumber-1);
        price=[futureBook  getBidPrice];
        futureSpread=(((newIndexValue-price)/newIndexValue)+((newIndexValue-
price)/price))/2;

        if(futureSpread>=arbitrageurOperatingInterval) {buySellSwitchFuture=1;
buySellSwitchShares=-1; }
        else {buySellSwitchFuture=0; buySellSwitchShares=0;
            if(printing==1)printf("Arbitrageur doesn't play: there isn't no such spread
(%5.3f %5.3f).\n",futureSpread, indexSpread);
            }

    }else if((buySellSwitchFuture==-1)&&(buySellSwitchShares==1)){
        for (i=1;i<bookNumber;i++)
            {
             Book * aBook;
             [bookArrayIndex setOffset: i-1];
             aBook=[bookArrayIndex get];
           sumOfPrice+=[aBook  getBidPrice];
          }
        newIndexValue=sumOfPrice/(bookNumber-1);
        price=[futureBook  getAskPrice];
        futureSpread=(((newIndexValue-price)/newIndexValue)+((newIndexValue-
price)/price))/2;

        if(futureSpread<=-1*arbitrageurOperatingInterval) {buySellSwitchFuture=-1;
buySellSwitchShares=1; }
        else {buySellSwitchFuture=0; buySellSwitchShares=0;
            if(printing==1)printf("Arbitrageur doesn't play: there isn't no such spread
(%5.3f %5.3f).\n",futureSpread, indexSpread);
            }

    }else{buySellSwitchFuture=0;}
  }


   //get the prices and insert orders

    if ((buySellSwitchFuture==1)&&(buySellSwitchShares==-1)){
        for (i=1;i<bookNumber;i++)
            {
             Book * aBook;
             [bookArrayIndex setOffset: i-1];
             aBook=[bookArrayIndex get];
          price=-1*[aBook  getAskPrice];
          [self setOrderFromAgent2: i ofAgent: number atPrice:  price];
```

```
          }
        price=[futureBook  getBidPrice];
        [self setOrderFromAgent2: bookNumber ofAgent: number atPrice:  price];
    }else if((buySellSwitchFuture==-1)&&(buySellSwitchShares==1)){
        for (i=1;i<bookNumber;i++)
            {
              Book * aBook;
              [bookArrayIndex setOffset: i-1];
              aBook=[bookArrayIndex get];
           price=[aBook  getBidPrice];
           [self setOrderFromAgent2: i ofAgent: number atPrice:  price];
          }

        price=-1*[futureBook  getAskPrice];
        [self setOrderFromAgent2: bookNumber ofAgent: number atPrice:  price];
    }else{buySellSwitchFuture=0;}


  return self;
}

@end
```

// **ArbitrageurAgent.h**

// ArbitrageurAgent

```
#import "BasicSumAgent.h"
#import "IndexCalculator.h"

@interface ArbitrageurAgent: BasicSumAgent
{
  int buySellSwitchFuture, buySellSwitchShares, arbitrageurGain,
arbitrageurOperatingIntervalFixed;
  float arbitrageurOperatingInterval,  indexValue, realFuturePrice, indexSpread,
        newIndexValue, futureSpread, sumOfPrice;

  IndexCalculator * indexCalculator;
  //Book * aBook;
}

- setArbitrageurOperatingInterval: (float) i;
- setArbitrageurOperatingIntervalFixed: (int) i;
- setArbitrageurGain: (int) i;
- setIndexCalculator: t;

- act0;
- act1;  // placing buy or sell orders

@end
```

Bibliografia

[1] PARISI D. (2001), *Simulazioni - La realtà rifatta nel computer*, Il Mulino.

[2] SHILLER R. J. (2000), *Euforia irrazionale - Analisi dei boom di borsa*, Il Mulino.

[3] GALIMBERTI F. (2002), *Economia e pazzia - Crisi finanziarie di ieri e di oggi*, Editore Laterza.

[4] LEBARON B. (June 2002), *Building a Santa Fe Artificial Stock Market*, Brandeis University.

[5] PIA P. (1997), *Il mercato azionario italiano*, G. Giappichelli Editore.

[6] HULL J.C. (2000), *Opzioni Futures e altri derivati*, Prentice-Hall International.

[7] FABRIZI P.L., FORESTIERI G., MOTTURA P. (1997), *Gli strumenti finanziari*, EGEA.

[8] CHANCELLOR E. (2000), *Un mondo di bolle - La speculazione finanziaria dalle origini alla "new economy"*, Carocci.

[9] LARKING D., WILSON G. (1995), *Object-Oriented Programming and the Objective-C Language*, Next Developer's Library, Redwood City.

[10] LUNA F., STEFANSSON B. (2000), *Economic Simulation in Swarm: Agent-Based Modelling and Object Oriented Programming*, Kluwer Academic Publishers, Boston/London.

[11] AXELROD R. (1997), *Advancing the Art of Simulation in the Social Sciences*, Conte R, Hegselmann R. & Terna P., Simulating Social Phenomena, Berlin.