



Universitatea *Transilvania* din Braşov

HABILITATION THESIS

**Applicability of Digitalization in Systems
Engineering**

Domain: Systems Engineering

**Author: Constantin SUCIU
Transilvania University of Braşov**

BRAŞOV, 2017

Acknowledgment

I would like to express my gratitude to all those from whom I have enriched my scientific, professional and life experience. I would like to thank to all those that accepted to jointly undertake paths that did not look predictable all the time.

I dedicate my modest career achievements to my family that proved incommensurable patience in all these years.

Contents

Part I	5
Part II	11
1. Introduction	13
2. Systems modeling	16
2.1 Induction machine with variable impedance.....	16
2.1.1 Electronically switched capacitor	17
2.1.2 Dynamic model of the induction motor with variable rotor impedance	18
2.1.2.1 Steady state performance of the induction motor with variable rotor impedance	22
2.1.2.2 Simulations & experimental results	25
2.1.3 Single phase induction machine with variable capacitor	30
2.1.3.1 Dynamic model of the single phase motor	32
2.1.3.2 Simulation & experimental results	35
2.2 Hemodynamics modeling.....	37
2.2.1 Fractional Flow Reserve.....	40
2.2.2 One dimensional blood flow models.....	41
2.2.3 Stenosis model	42
2.2.3 Outflow boundary bondition implementation	42
2.2.3.1 Implicit Lax-Wendroff	43
2.2.3.2 Explicit Lax-Wendroff	43
2.2.3.3 Implicit method of characteristics	44
2.2.4 Personalization of the multi-scale model of the coronary hemodynamics.....	44
2.2.4.1 Estimation of boundary bonditions at rest	44
2.2.4.2 Estimation of boundary bonditions at hyperemia	46
2.2.4.3 Autoregulation	47
2.2.5 Feedback control system	48
2.2.7 Alternative method in modeling arterial hemodynamics	49
2.2.8 Results - Model simulations and analysis	51
2.2.8.1 Simulation of outflow boundary condition implementations	51

2.2.8.2 Simulation of stenosis model.....	54
2.3 Synopsis.....	57
3. High performance computing of system models	58
3.1 Numerical solution of elliptic equations.....	59
3.1.1 Problem definition	60
3.1.2 Implementation of elliptic equations	60
3.1.3 Implementation results for elliptic equations	63
3.2 Acceleration of hemodynamic models	65
3.2.1 Methods for one-dimensional blood flow models.....	66
3.2.1.1 Boundary conditions	66
3.2.1.2 Numerical solution of the one-dimensional blood flow model	66
3.2.2 Parallelization of the numerical solution	70
3.2.2.1 Parallel Hybrid CPU-GPU (PHCG) algorithm.....	70
3.2.2.2 Parallel GPU Only (PGO) implementation.....	75
3.2.3 High performance simulation results.....	77
3.2.3.1 Comparison of parallel and sequential computing and with different numerical schemes	78
3.2.3.2 Comparison of the memory copy strategies for the PHCG algorithm.....	79
3.2.3.3 Comparison of the performance obtained with the SCO, MCO, PHCG and PGO algorithms.....	79
3.3 Synopsis	85
4. Decision support.....	86
4.1 Monitor and inform	86
4.1.1 HW description.....	86
4.1.2. Communication protocol.....	88
4.1.3 SW description	89
4.1.3.1 Central server software	89
4.1.3.2 PDU operating system	90
4.2 Constrained based production optimization using SOA.....	91
4.2.1 UA servers.....	94
4.2.2 Software services.....	95
4.2.2.1 Basic services	95
4.2.2.2 Complex Services	97
4.2.3 Optimization driven constraint satisfaction models	97
4.2.3.1 MIP algorithms	98
4.2.4 Performance tests.....	101
4.2.4.1 Basic service execution times.....	101
4.2.4.2 Reading and writing of boolean variables.....	102
4.2.4.3 Roundtrip test	102
4.2.4.4 Alarm test	103
4.2.4.5 UA Connection test.....	103
4.3 Synopsis.....	104

Part III	105
5. Academic and Research Career	107
5.1 Past Research and Academic Activities.....	107
5.2 Future Work	108
References.....	113

Part I

Rezumat

Digitalizarea reprezintă utilizarea *technologiilor digitale* cu scopul de a modifica modelul de afaceri/operare pentru furnizarea de venituri suplimentare organizațiilor și pentru identificarea de oportunități de generare de noi fluxuri de valoare. În același timp, digitalizarea urmărește *creșterea eficienței proceselor deja funcționale*.

Domeniile care sunt influențate de consumatorul final (de ex. media, comerțul) au fost influențate într-o măsură semnificativă datorită evoluției dispozitivelor mobile și a sistemelor de calcul bazate pe arhitecturi de tip many/multi-core. Sectoarele care sunt strict reglementate sau sensibile (de ex. utilități, sănătate) sunt încă într-o fază relativ de început în ceea ce înseamnă adoptarea digitalizării. Introducerea digitalizării poate fi obținută prin dezvoltarea pe următoarele paliere:

- a. Generarea de modele care să reprezinte procesul/sistemul cât mai aproape de realitate;
- b. Achiziția de date de la proces/sistem indiferent unde se află dispozitivele datorită evoluției rapide ale tehnologiilor de comunicație; stocare și analizarea unor volume masive de date în intervale de timp extrem de scurte;
- c. Accelerarea simulării de modele specifice/”personalizate” ale sistemelor sub observație folosind aspectele de la punctele a și b, precum, și a analizei datelor colectate; detecția de modele comportamentale;
- d. Generarea și gestionarea dinamică a comenzilor pentru îmbunătățirea către sistem supervizat în combinație cu simularea modelelor “personalizate” pentru a prezice efectul comenzilor aplicate (de ex. optimizarea consumurilor energetice într-o fabrică, mentenanță/operare predictivă sau prescriptivă);

Activitatea de cercetare prezentată în această lucrare se pliază în special pe palierele *a* și *c*, abordând și aspecte din palierele *b* și *d*.

O metodologie pentru modelarea mașinii de inducție cu impedanță rotorică modificată nerezistiv a fost dezvoltată pentru o evaluarea facilă a performanțelor unei mașini de inducție în cazul în care în circuitul rotoric este introdus un condensator a cărui valoare poate fi modificată în mod dinamic. Metodologia permite estimarea valorii condensatorului necesară pentru a obține performanțele dorite. Abordarea teoretică a arătat posibilitatea de a îmbunătăți factorul de putere, eficiența și diagrama cuplu-rotatie pe baza unui set de ecuații. Pentru validarea acestor modele din punct de vedere practic, a fost folosit un mecanism de emulare de efecte capacitive variabile care a fost dezvoltat anterior. Rezultatele experimentale pe un motor de inducție de putere mică au arătat că acestea sunt similare cu rezultatele simulărilor. Factorul de putere și eficiența se îmbunătățesc practic pentru orice cuplu de sarcină. Raportul turație-cuplu este îmbunătățit prin creșterea cuplului maxim dezvoltat. Performanțe optime la un cuplu de sarcină dată pot fi obținute numai prin varierea valorii condensatorului aplicat în circuitul rotoric. Analize echivalente au fost făcute pentru motorul de inducție monofazat.

Persoanele cu boli cardiovasculare sau care prezintă un risc ridicat (datorită prezentei unor factori de risc cum ar fi hipertensiunea, diabetul etc.) au nevoie de detecție timpurie și un management corespunzător al tratamentului. Dezvoltarea de modele cardiovasculare personalizate a avut o contribuție semnificativă în acest sens. Provocarea în ceea ce privește modelarea cu acuratețe a sistemului cardiovascular este dată de faptul că acesta este un circuit închis cu un grad înalt de interdependență între compartimentele vasculare individuale. Studiul curgerii locale a sângelui este deosebit de important din moment ce anumite patologii, cum ar fi îngrosarea locală a vaselor sau formarea de stenoze, sunt influențate de hemodinamica locală. În același timp, modificări locale cum ar fi lumenul vascular, pot conduce la redistribuiri globale a curgerii sângelui determinând mecanisme compensatorii care să asigure rate de curgere suficientă în zona distală a vasului afectat. Simularile 3D la scara completă a curgerii sângelui sunt extrem de solicitante din punct de vedere computațional și pot fi realizate doar pe un număr redus de vase. Influența reciprocă între hemodinamica globală și cea locală precum și cerințele de calcul

ridicate ale simulărilor 3D, au condus la conceptul de modelare multi-scalară a curgerii sângelui care a fost preluată și în activitățile prezentate în această lucrare. Doar zonele de interes din cadrul arborelui arterial - de ex. segmentele care sunt îngustate și prezintă depuneri - sunt simulate folosind modelul complet 3D în timp ce pentru celelalte segmente se folosesc modele de ordin redus (modele 1D pentru arterele largi și modele 0D pentru arterele mici și microvasculatură). Modelele de ordin redus produc rezultate viabile în ceea ce privește presiunea și rata de curgere, iau în considerare în mod corect efectul în zona distală a vaselor și al microvasculaturii, facilitând timpi de execuție de câteva ori mai mici decât în cazul simulărilor corespondente 3D. Modelele de curgere ale sângelui uni-dimensionale au fost investigate pe baza ecuațiilor Navier-Stokes folosind vâscoelasticitatea pereților vaselor, precum și condițiile de frontieră pe baza unor metode cum ar fi Lex-Wendrof, variantele implicite respectiv explicite. În vederea obținerii de modele personalizate cât mai apropiate de realitate este necesară determinarea condițiilor de frontieră realizând măsurători de presiune arterială în stare de repaus și de hiperemie. În combinație cu modelele vaselor stenozate, este posibilă estimarea valorii coeficientului FFR (Fractional Flow Reserve) ca element de evaluare non-invazivă a gradului de stenozare a vasului.

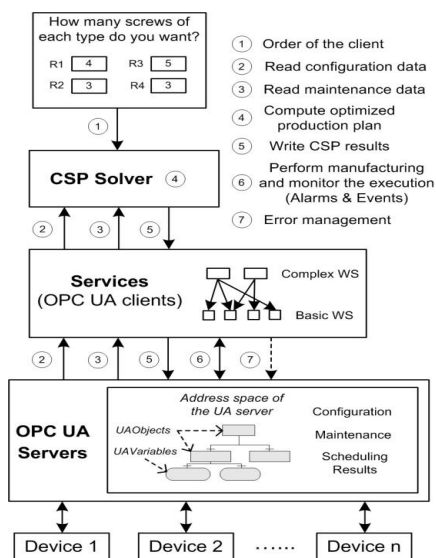
Aplicarea metodologiilor de calcul de înaltă performanță (HPC-high performance computing) a fost investigată pentru a obține accelerări ale simulării algoritmilor medicali. Deși modelele unidimensionale ale curgerii sângelui sunt în general cu două ordine de mărime mai rapide în comparație cu cel 3D, cerințele de a avea timpi de execuție foarte scurți rămân valide. Rezultatele simulărilor curgerii sângelui pe geometrii specifice pacientului în context clinic trebuie obținute într-un timp corespunzător nu doar din perspectiva riscului potențial de sănătate dar și în ceea ce înseamnă procesarea mai multor pacienți într-un anumit interval de timp. Este crucial să existe sincronizare între simularea cardiovasculară și starea specifică a pacientului. Procedura de "acordare" a modelului cu pacientul solicită rulări repetate pe aceeași geometrie având parametri diferiți (de ex. condițiile de frontieră de intrare, ieșire, etc.) până când măsurimile măsurate respectiv simulate se potrivesc. Acest fapt sporește timpul de execuție pentru o geometrie specifică a unui singur pacient. Prin urmare, activitățile de investigare au fost realizate în vederea de a beneficia de capacitățile de calcul sporite ale unor platforme cu GPU pentru a optimiza timpul de execuție a diferitelor tipuri de ecuații diferențiale ca și bază pentru modele mult mai complexe. Următorul pas a fost obținerea accelerării modelelor unidimensionale utilizând abordări cum ar fi: Parallel Hybrid CPU-GPU cu operațiuni compacte de copier (PHCGCC) sau simplu Parallel GPU (PGO). Acestea au fost aplicate și pentru modelul arterial al întregului corp compus din 51 de artere iar accelerarea pentru cele două abordări a fost comparată atât cu varianta pe CPU a unui singur fir de execuție cât și multi-fir de execuție. Simulările au fost realizate pe două scheme numerice de ordin al doilea diferite, utilizând modele elastice respectiv vâscoelastice ale peretului, condiții de frontieră Windkessel sau arbore structurat, ca exemple reprezentative ale fiziologiei non-periodice respectiv periodice a condițiilor de frontieră de ieșire. Simulările au arătat accelerări de câteva ordine de mărime comparativ cu abordarea secvențială.

Costurile pentru pregătirea activităților de producție reprezintă aproximativ o treime din totalul costurilor de producție. Dinamica dată de cererile consumatorilor, globalizarea, nevoia de a exista o corelare a producției cu prețurile fluctuante ale materiilor prime-precum și disponibilitatea acestora- exercită o presiune asupra companiilor manufacturiere în a găsi modalități inovatoare de a-și configura sistemul de producție pentru a fi capabili să satisfacă cerințele clienților la termenul solicitat, optimizând costurile de producție cu scopul maximizării profitului companiilor. Prin urmare, una dintre direcțiile de investigare a fost optimizarea fluxului de producție pentru a satisface într-un interval de timp minim solicitările dinamice de manufacturare ale unor game de diferite de produse prin metodologii automatizate. Această

optimizare este obținută pe baza unei arhitecturi ce poate fi folosită practic în orice fabrică fiind flexibilă și adaptivă, îndeplinind următoarele criterii din punct de vedere funcțional:

- Optimizarea procesului de producție prin calcularea planului optim de manufacturare;
- Utilizarea automată a castor planuri-fără intervenția operatorului;
- Dezvoltarea unei arhitecturi flexibile și reutilizabilă care scurtează timpii necesari mentenanței, instalării și configurării, oferind posibilitatea de a reacționa rapid la cerințele pieții;
- Tranziția lină de la configurațiile existente la noua abordare;

Arhitectura dezvoltată pentru a satisface cerințele menționate mai sus a constat din 3 nivele:



- OPC UA (OLE-Object Linking and Embedding- for Process Control Unified Architecture) – colectează date de la dispozitivele, senzorii și elementele de execuție de pe linia de producție, le modelează într-un mod standardizat și asigură comunicarea în timp real cu acestea;

- Nivelul de servicii software – elementul central al arhitecturii asigurând flexibilitatea și adaptabilitatea sistemului; serviciile software au o contribuție majoră în realizarea acestui nivel;

- Nivelul CSP (Constraint Satisfaction Problem) – se adresează optimizarea planurilor de producție folosind conceptele de programare a constrângerilor/dependențelor dintre diferitele elemente care influențează întreg procesul. Conceptul a fost implementat și validat pe o linie de producție flexibilă de laborator.

Activitățile de cercetare au fost susținute prin intermediul proiectelor finanțate public național și European iar rezultatele s-au concretizat în 37 articole la conferințe și în jurnale ISI, cu factor de impact cumulat de peste 80.

Part II

Scientific and Professional Achievements

1. Introduction

Digitalization is the use of *digital technologies* to change a business model and provide new revenue and value-producing opportunities; it is the process of moving to a *digital business*. **Digital business** is the creation of new business designs by blurring the digital and physical worlds.

Digitalization technologies are mainly formed by three technological pillars: Internet of Things (IoT), Analytics and Big Data. Support technologies such as hardware and software or security as equally important though they are only forming the basis for digitalization. Hardware and Software are the basic foundation of any computing system. Both of them are becoming a commodity as prices are going down while demand is on a steep rise. While less than a decade ago hardware development was targeting the unification of all components into standard modules, ready to be plugged into a new silicon device, today we are witnessing the diversification and customization efforts of hardware manufacturers. While standard components are still used, they only form the basic architecture of modern System on Chips (SoC's). Custom components such computational co-processors or AI accelerators are just a few of the major investment areas of hardware manufacturers. With mobile devices having a greater importance than ever, energy efficiency and communication channels are of crucial importance. Also, in order to help software developers, hardware has expanded into providing ease of access and computational support for more functions such as multimedia and security(see fig. 1.1).

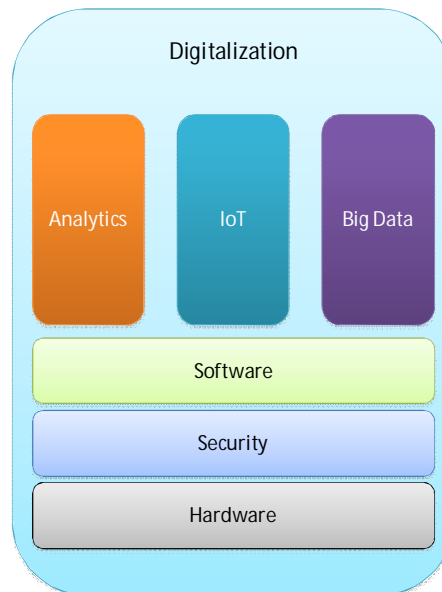


Fig. 1.1 – Basic concept of a digitalized process

In the context of digitalization, where huge amount of data is generated, stored, processed or exchanged between one or more partners, security plays a crucial role.

Big Data and Analytics are the naturally following technologies which form Digitalization. Big Data reflects the very large amounts of digital information which need to be transferred, stored and processed. Typical mechanisms which transfer and process data are unusable therefore new paradigms and technologies need to replace traditional stereotypes. Analytics is the discovery, interpretation, and communication of meaningful patterns in digital information.

Analytics over Big Data is the last chain in extracting usable patterns which can be further used in automating industrial processes such as manufacturing.

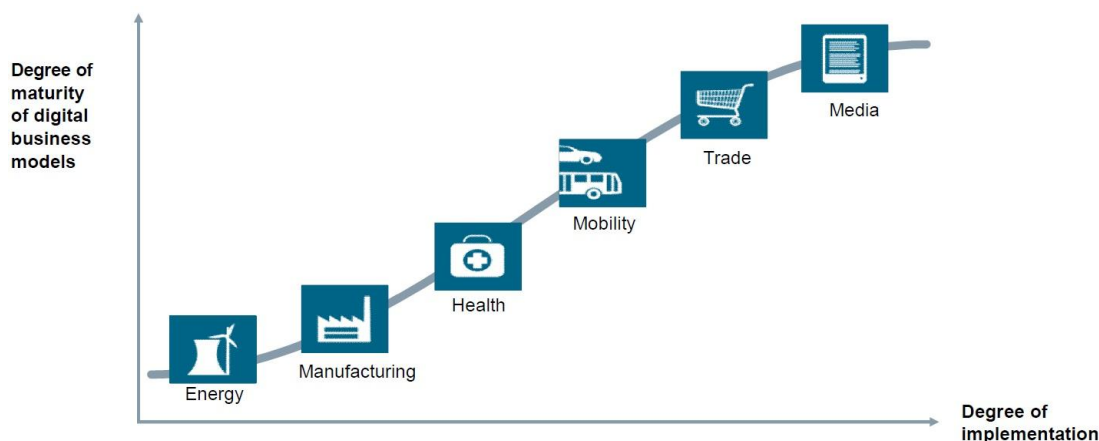


Fig. 1.2 - Digitalization on relevant domains

The studies (see fig. 1.2 – from a Gartner study) regarding the degree of digitalization penetration in various domains of activities show currently a stronger impact on end-user consumer related domains like media and trade where, thanks to fast evolution of mobile devices (e.g. smart-phones, tablets etc.) and/or thanks to high processing computing (e.g. multi- and many-core based systems), new approaches were enabled in retail business or stock markets.

The strong regulated or sensitive domains (e.g. utilities, healthcare) are at the start or on the bring to take off of the process that poses besides a challenge a good opportunity regarding innovative approaches due to the natural effects/goals of digitalization: increase of the production efficiency. This can be achieved by considering the following levels of development:

- a. Generation of models that represent the process/system as close to reality as possible;
- b. Acquisition of data from the process/system regardless where are located the sensor/acquisition devices as a result of fast evolution of communication technologies; storage and handling of paramount volume of data in short time intervals;
- c. Speed up the simulation of the customized models based on points a and b as well as analysing the collected data; detection of behavioral patterns in real time;
- d. Dynamically generate/manage commands for improving the supervised process/business (e.g. energy optimisation in a plant, predictive or prescriptive maintenance, etc.); use the customised models to simulate the effect of the commands on the process before applying- digital twin approach.

The work presented herein has been focused mainly on the levels a&c, touching also elements of levels b&d.

Chapter 2 focuses on the development models from 2 different domains:

- Modeling of induction motors with variable (rotor) impedance with the goal to evaluate the potential in performance improvement by the usage of dynamically emulated capacitive effect;
- Patient personalized 1D models for non-invasive assessment of stenoses for coronary vessels.

Chapter 3 tackles methodologies to accelerate the simulation of the developed models using many-core based platforms along with the optimization of the model algorithms.

Chapter 4 focuses on an early investigation regarding data collection from remote area as well as on a methodology for scheduling in a flexible and dynamic way the multiple-product manufacture line for achieving efficient utilization of the resources.

2. Systems modeling

The development of system models has become of significance relevance regardless of the domain of activity. In the case of industrial applications, availability of customised/adapted models support implementation of predictive or (potentially) prescriptive maintenance strategies with impact on efficient operations of the systems or on cost savings in regard to service operations. Personalised patient models have started having an impact in non-invasive diagnostics or on customised treatments based on simulation either of virtual surgery or of drug effects facilitated by big data knowledge base.

The current chapter presents contributions on system modelling on 2 directions:

- Induction machine in the context of improving operational performance;
- Patient specific models for coronary arteries.

2.1 Induction machine with variable impedance

The induction machine is widely used in industrial and domestic applications. The range of power varies between hundreds of watts to megawatts, depending upon the application:

- For industrial medium power applications a three phase squirrel-cage motor is preferred. These motors are relatively low cost and robust. In higher power applications, e.g. pumps and cranes, the wound rotor type of the induction machine is often used.
- For domestic applications, the single phase induction motor is a common choice. The single phase induction motor is essentially a two phase machine with one winding in series with a capacitor to give the required phase shift

Three phase induction motors are popular in industrial and in manufacturing processes as well as some propulsion applications. The wound rotor and the squirrel cage rotor are the main two categories of the induction motors. In some application wound rotor induction motor (WRIM) is more suitable than squirrel cage motor. The main advantage of these motors over the squirrel cage motors is their high starting torque with low startup current by the means of inserting an external resistance in series to the rotor circuit. Besides limiting the starting current, this resistance can also be used in controlling the motor speed. These advantages made the WRIM suitable for large power industrial application such as cranes, pumps, conveyors and hoists.

At present wound rotor motors still find applications in drives with high inertia loads like the pumping of water, oil and petroleum. In these applications, due to the high power levels, consideration of losses has been of paramount importance. Slip energy recovery and rotor impedance modification have been used for improving motors performance [2.2, 2.3].

There have been efforts to analyse and produce rotor control techniques that avoid double feeding and the associated disadvantages. The modification of rotor impedance through a capacitance instead of adding resistance has been one of the directions. The initial tests showed that the required capacitor values are significant and there is the need to be changed dynamically (e.g. depending on the load).

In the same time, single-phase induction machines with capacitor start, with/without capacitor run connections in the auxiliary winding are the mostly widely used electric machines in home appliances and in applications requiring less than $5kW$, consuming a significant percentage of electricity generated in the world [2.8]. Therefore, the attached/inserted capacitor is also used to enhance the performance of a single phase induction motor which has two phases designed to operate in quadrature. Quadrature fluxes are produced by connecting a capacitor in series to one of the windings. Appropriate choice of capacitance can optimise performance. The capacitor is important both at starting phase and in steady-state. The main problem is in the choice of the capacitor. The capacitor size must be carefully determined according to the terminal impedance and from the fact that two different values of the capacitor are required for starting and steady state running. The capacitor used in the steady state is between five and ten times as small as the capacitor used for starting. The difference between these two values results in the use of two different capacitors, the large value is placed in series with the auxiliary winding during starting and the small one is switched in the circuit when the machine reaches full speed. At the instant of switching in the small capacitor, the large capacitor is switched out of circuit.

There has been the need to propose HW implementations that allow the emulation of dynamic large capacitor values but as well mathematically models that reflect the machine-capacitor coupling reflecting the value of the capacitor to be used but also the performance evolution of the motor.

The author presents herein contributions to models used for mathematically describe the problems stated above for both induction motor with variable rotor impedance as well as for single phase machine.

2.1.1 Electronically switched capacitor

The author has contributed to the introduction of a novel methodology of emulating high value dynamically adjustable capacitors (see fig. 2.1)[2.4].

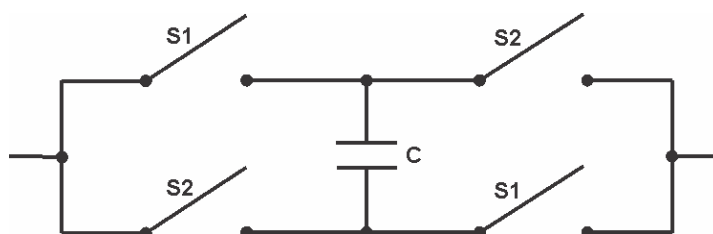


Fig. 2.1 - Emulated capacitor using H-Bridge with bi-directional switches

A capacitor is placed in the middle of H bridge that has bidirectional switches. The switches pair S1 is ON for interval t_1 while switches pair is on t_2 where:

$$T = t_1 + t_2$$

T represent the switching period and it is defined duty ratio d :

$$d = \frac{t_1}{T}$$

The value of the emulated capacitor between points a-b is:

$$C_e = \frac{C}{(2d - 1)^2}$$

having $0,5 \leq d \leq 1$.

The previously developed HW emulation method has been considered as enabler for modeling the induction machine with variable rotor impedance.

2.1.2 Dynamic model of the induction motor with variable rotor impedance

The emulated capacitor has been used in the modeling and analysis of the wound induction motor with modifying rotor impedance by the insertion on each phase of a dynamically adjustable high value capacitor.

Phasor theory and the space vector basics of the induction motor are given in reference [2.1].

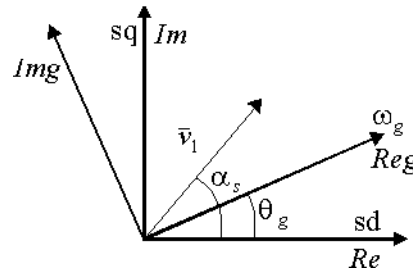


Fig. 2.2 – General reference frame

In the analysis, there is considered a general reference frame (see fig. 2.2) that is spinning with the angular speed ω_g ($\omega_g = \omega_g t$) with respect to stationary reference frame (sd-sq).

The mathematical model of the induction motor is presented in [2.1, 2.12] with the following assumptions: the stator and rotor are cylindrical with a smooth air gap with symmetrical three-phase windings displaced by 120° electrical degrees.

In this analysis the magnetic circuit is assumed to be infinitely permeable, the flux density is assumed to be radial in the air gap and the effect of iron losses and end-effects are neglected. The number of effective stator turns, N_{se} , and effective rotor turns N_{re} are equal. The stator winding voltages are:

$$\begin{aligned} u_{sa} &= R_s i_{sa} + \frac{dy_{sa}}{dt} \\ u_{sb} &= R_s i_{sb} + \frac{dy_{sb}}{dt} \\ u_{sc} &= R_s i_{sc} + \frac{dy_{sc}}{dt} \end{aligned} \quad (2.1)$$

where u_{sa}, u_{sb}, u_{sc} are the input stator voltages, i_{sa}, i_{sb}, i_{sc} are the stator currents, R_s is the stator resistance per phase, y_{sa}, y_{sb}, y_{sc} are the instantaneous values of the stator flux linkages of each of the three phases.

Starting from this system, it is possible to write the space phasor equation that characterises the stator circuit in a reference frame fixed to the stator (stator reference frame sd-sq):

$$\bar{u}_s = R_s \bar{i}_s + \frac{d\bar{y}_s}{dt} = R_s \bar{i}_s + L_s \frac{d\bar{i}_s}{dt} + L_m \frac{d\bar{i}_r}{dt} \quad (2.2)$$

where $\bar{u}_s, \bar{i}_s, \bar{y}_s, \bar{i}_r$ are the space phasors in the given reference frame of stator voltage, stator current, stator flux and rotor current; L_s and L_m are the stator self inductance per phase and three phase magnetising inductance respectively.

Similarly for the rotor circuit in a reference frame fixed to the rotor (rotor reference frame):

$$\bar{u}_r = R_r \bar{i}_r + \frac{d\bar{y}_r}{dt} = R_r \bar{i}_r + L_r \frac{d\bar{i}_r}{dt} + L_m \frac{d\bar{i}_s}{dt} \quad (2.3)$$

where R_r is the rotor resistance per phase, L_r is the rotor self inductance per phase $\bar{u}_r, \bar{i}_r, \bar{y}_r, \bar{i}_s$ are the space phasors in the given reference frame of rotor voltage, rotor current, rotor flux and stator current.

Equations (2.2) and (2.3) are written in the stator reference frame and in the rotor reference frame respectively. These equations are rewritten in a common reference frame and, in this way, it will be possible to describe the behavior of the motor. In the general reference frame:

$$\begin{aligned} \bar{u}_{sg} &= R_s \bar{i}_{sg} + L_s \frac{d\bar{i}_{sg}}{dt} + L_m \frac{d\bar{i}_{rg}}{dt} + j\omega_g (L_s \bar{i}_{sg} + L_m \bar{i}_{rg}) \\ \bar{u}_{rg} &= R_r \bar{i}_{rg} + L_r \frac{d\bar{i}_{rg}}{dt} + L_m \frac{d\bar{i}_{sg}}{dt} + j(\omega_g - \omega_r)(L_m \bar{i}_{sg} + L_r \bar{i}_{rg}) \end{aligned} \quad (2.4)$$

where ω_g is the angular velocity of the spinning general reference frame, ω_r is the angular speed of the rotor; all the space phasors are represented with respect to the general reference frame.

Also from [2.1], the system defined in equation (2.4) is completed with the mechanical equation of motion:

$$t_e - t_l = J \frac{d\omega_r}{dt} + D_w \omega_r \quad (2.5)$$

where t_e is the electromagnetic torque produced by motor (Nm), t_l is the load torque (Nm), J is the inertia of the motor (kgm^2) and D_w is damping constant -representing the dissipation of energy due to windage and friction.

Equations (2.4) and (2.5) along with torque expression:

$$t_e = \frac{3}{2} p \bar{y}_{sg} \cdot \bar{i}_{sg} = - \frac{3}{2} p \bar{y}_{rg} \cdot \bar{i}_{rg} \quad (2.6)$$

describes the behavior of the induction motor (where p is the number of pole pairs).

The following system is obtained if the equations from system (2.4) are split into the direct and quadrature components:

$$\begin{aligned} \frac{di_{sx}}{dt} &= \frac{-R_s L_r i_{sx} + \omega_r L_m^2 i_{sy} + R_r L_m i_{rx} + \omega_r L_m L_r i_{ry} + L_r u_{sx} - L_m u_{rx}}{LD} + \omega_g i_{sy} \\ \frac{di_{rx}}{dt} &= \frac{R_s L_m i_{sx} - \omega_r L_m L_s i_{sy} - R_r L_s i_{rx} - \omega_r L_s L_r i_{ry} - L_m u_{sx} + L_s u_{rx}}{LD} + \omega_g i_{ry} \\ \frac{di_{sy}}{dt} &= \frac{-\omega_r L_m^2 i_{sx} - L_r R_s i_{sy} - \omega_r L_r L_m i_{rx} + L_m R_r i_{ry} + L_r u_{sy} - L_m u_{ry}}{LD} - \omega_g i_{sx} \\ \frac{di_{ry}}{dt} &= \frac{\omega_r L_m L_s i_{sx} + L_m R_s i_{sy} + \omega_r L_r L_s i_{rx} - L_s R_r i_{ry} - L_m u_{sy} + L_s u_{ry}}{LD} - \omega_g i_{sx} \end{aligned} \quad (2.7)$$

where $LD = L_s L_r - L_m^2$.

Equation (2.6) can be rewritten as:

$$t_e = \frac{3}{2} p(L_m \bar{i}_{rg} + L_s \bar{i}_{sg})' \bar{i}_{sg} = \frac{3}{2} pL_m \bar{i}_{rg}' \bar{i}_{sg} = \frac{3}{2} pL_m (i_{rx} i_{sy} - i_{sx} i_{ry}) \quad (2.8)$$

Equation (2.8) can be, also, expressed as function of the modulus of both rotor current and stator current space vectors, and the displacement angle Z between these two vectors:

$$t_e = \frac{3}{2} pL_m |\bar{i}_{rg}| |\bar{i}_{sg}| \sin Z \quad (2.9)$$

Equation (2.5) is brought to the following form:

$$\frac{dw_r}{dt} = \frac{t_l - t_e - D_w w_r}{J} \quad (2.10)$$

Equations (2.7), (2.8) and (2.10) form a differential equation system that facilitates the simulation of the dynamic behavior of an induction.

Having above considerations and equations as basis, a set of differential equations that predicts the dynamic behavior of a motor with externally controlled rotor impedance has been developed [2.12] using the space vector model of the induction motor.

If a capacitor is introduced on each phase of the rotor circuit, each phase of the rotor circuit will have the following configuration:

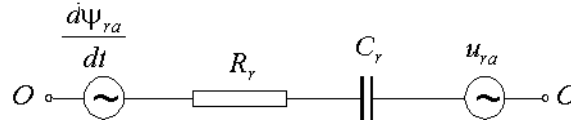


Fig. 2.3 – General configuration of the rotor circuit with modified impedance

The behavior on phase A is described by the following equations:

$$u_{ra} = R_r i_{ra} + \frac{dy_{ra}}{dt} + u_{ca}$$

$$\frac{du_{ca}}{dt} = \frac{i_{ra}}{C_r}$$

where u_{ra} is the input rotor voltage on phase a, i_{ra} is the rotor current on phase a, y_{ra} is the instantaneous values of the rotor flux linkages on phase a and u_{ca} is voltage across the capacitor.

Similar equations can be written for each phase of the rotor:

$$u_{rb} = R_r i_{rb} + \frac{dy_{rb}}{dt} + u_{cb}$$

$$\frac{du_{cb}}{dt} = \frac{i_{rb}}{C_r}$$

$$u_{rc} = R_r i_{rc} + \frac{dy_{rc}}{dt} + u_{cc}$$

$$\frac{du_{cc}}{dt} = \frac{i_{rc}}{C_r}$$

The space vector equations [2.12] can now be written in a reference frame fixed to the rotor, starting from the above equations:

$$\bar{u}_r = R_r \bar{i}_r + \frac{d\bar{y}_r}{dt} + \bar{u}_c$$

$$\frac{d\bar{u}_c}{dt} = \frac{\bar{i}_r}{C_r}$$

Combining with (2.3) results in:

$$\bar{u}_r = R_r \bar{i}_r + L_r \frac{d\bar{i}_r}{dt} + L_m \frac{d\bar{i}_s}{dt} + \bar{u}_c \quad (2.11)$$

$$\frac{d\bar{u}_c}{dt} = \frac{\bar{i}_r}{C_r} \quad (2.12)$$

The stator circuit equation is not affected by the insertion of capacitors. Considering a general reference frame for equations (2.2), (2.11), (2.12) and using the transformation rules from one reference system to another, results in:

$$\begin{aligned} \bar{u}_{sg} &= R_s \bar{i}_{sg} + L_s \frac{d\bar{i}_{sg}}{dt} + L_m \frac{d\bar{i}_{rg}}{dt} + j\omega_g (L_s \bar{i}_{sg} + L_m \bar{i}_{rg}) \\ \bar{u}_{rg} &= R_r \bar{i}_{rg} + L_r \frac{d\bar{i}_{rg}}{dt} + L_m \frac{d\bar{i}_{sg}}{dt} + j(\omega_g - \omega_r)(L_m \bar{i}_{sg} + L_r \bar{i}_{rg}) + \bar{u}_{cg} \\ \frac{d\bar{u}_{cg}}{dt} &= \frac{\bar{i}_{rg}}{C_r} - j(\omega_g - \omega_r)\bar{u}_{cg} \end{aligned} \quad (2.13)$$

This system in combination with equations (2.8) and (2.10) describes the behaviour of an induction motor with variable rotor impedance in a general reference frame.

The induction motor will be considered in the stator reference frame (this implies $\omega_g = 0$), with no voltage supplied into motor from the rotor side ($|\bar{u}_r| = 0$), with D_w neglected and a system similar to (2.7) is obtained by splitting the equations (2.13) into the direct and quadrature components:

$$\begin{aligned} \frac{di_{sx}}{dt} &= \frac{-R_s L_r i_{sx} + \omega_r L_m^2 i_{sy} + R_r L_m i_{rx} + \omega_r L_m L_r i_{ry} + L_r u_{sx} + L_m u_{cx}}{LD} \\ \frac{di_{rx}}{dt} &= \frac{R_s L_m i_{sx} - \omega_r L_m L_s i_{sy} - R_r L_s i_{rx} - \omega_r L_s L_r i_{ry} - L_m u_{sx} - L_s u_{cx}}{LD} \\ \frac{di_{sy}}{dt} &= \frac{-\omega_r L_m^2 i_{sx} - L_r R_s i_{sy} - \omega_r L_r L_m i_{rx} + L_m R_r i_{ry} + L_r u_{sy} + L_m u_{cy}}{LD} \\ \frac{di_{ry}}{dt} &= \frac{\omega_r L_m L_s i_{sx} + L_m R_s i_{sy} + \omega_r L_r L_s i_{rx} - L_s R_r i_{ry} - L_m u_{sy} - L_s u_{cy}}{LD} \end{aligned} \quad (2.14)$$

$$\frac{du_{cx}}{dt} = \frac{i_{rx}}{C_r} - w_r u_{cy}$$

$$\frac{du_{cy}}{dt} = \frac{i_{ry}}{C_r} + w_r u_{cx}$$

where $LD = L_s L_r - L_m^2$.

The expression of the torque equation remains the same and, thus, system (2.14) along with equations (2.8) and (2.10), offers the possibility to analyze the induction motor with modified rotor impedance in dynamic regime.

2.1.2.1 Steady state performance of the induction motor with variable rotor impedance

The equations (2.2) and (2.11) form the basis for the motor model in steady state operation [2.12]. This means $|\bar{i}_s|$, $|\bar{i}_r|$ and w_r are constant. Equation (2.2) is then modified to the form:

$$\bar{i}_s = \frac{\bar{u}_s - jw_1 L_m \bar{i}_r}{R_s + jw_1 L_s} \quad (2.15)$$

The expression is identical with the equation of the induction motor without capacitance.

Equation (2.11) is different from the equivalent equation of the induction motor in standard configuration given in (2.3), resulting in:

$$0 = R_r \bar{i}_r + jw_2 L_m \bar{i}_s + jw_2 L_r \bar{i}_r + \frac{\bar{i}_r}{jw_2 C_r} \quad (2.16)$$

where w_1 is the stator angular velocity and w_2 is the rotor angular velocity.

The mathematical combination of (2.15) and (2.16) gives:

$$0 = \left((R_r R_s + w_1 w_2 L_m^2 - w_1 w_2 L_s L_r + \frac{w_1 L_s}{w_2 C_r}) + j(w_2 L_s R_r + w_2 L_r R_s - \frac{R_s}{w_2 C_r}) \right) \bar{i}_r + jw_2 L_m \bar{u}_s$$

The modulus of the rotor vector current is obtained from:

$$\bar{i}_r = - \frac{jw_2 L_m \bar{u}_s}{(R_r R_s + w_1 w_2 L_m^2 - w_1 w_2 L_s L_r + \frac{w_1 L_s}{w_2 C_r}) + j(w_2 L_s R_r + w_2 L_r R_s - \frac{R_s}{w_2 C_r})}$$

and it results in:

$$|\bar{i}_r| = \frac{w_2 L_m |\bar{u}_s|}{\sqrt{A_m w_2^2 + B_m w_2 + C_m + \frac{D_m}{w_2^2}}} \quad (2.17)$$

where:

$$A_m = (w_1 L_m^2 - w_1 L_s L_r)^2 + L_r^2 R_s^2$$

$$B_m = 2w_1 R_r R_s L_m^2$$

$$C_m = (R_r R_s)^2 + (w_1 L_s R_r)^2 - 2 \frac{w_1^2 L_s^2 L_r}{C_r} + 2 \frac{w_1^2 L_m^2 L_s}{C_r} - \frac{2 L_r R_s^2}{C_r}$$

$$D_m = \frac{\frac{\partial}{\partial} w_1 L_s \ddot{\theta}^2}{\frac{\partial}{\partial} C_r \emptyset} + \frac{\frac{\partial}{\partial} R_s \ddot{\theta}^2}{\frac{\partial}{\partial} C_r \emptyset}$$

In (2.17) all variables except w_2 are known, the rotor angular velocity will be determined from the relation between the electromagnetic torque and the rotor current as follows:

$$t_e = \frac{3 p R_r |\bar{i}_r|^2}{2 w_2} \quad (2.18)$$

which results in a the fourth order polynomial in w_2 :

$$A_m w_2^4 + (B_m - E_m) w_2^3 + C_m w_2^2 + D_m = 0$$

where:

$$E_m = \frac{3 L_m^2 |\bar{u}_s|^2 p R_r}{2 t_e},$$

it is possible to calculate the modulus of the stator vector current:

$$|\bar{i}_s| = \frac{\sqrt{R_r^2 + w_2^2 \frac{\partial}{\partial} L_r - \frac{1}{w_2^2 C_r} \frac{\partial}{\partial} \ddot{\theta}^2}}{w_2 L_m} |\bar{i}_r| \quad (2.19)$$

Then using the expressions of the electromagnetic torque, the copper losses can be calculated using the formula:

$$P_{cu} = \frac{t_e}{p} w_2 + \frac{R_s}{R_r} \frac{R_r^2 + w_2^2 \frac{\partial}{\partial} L_r - \frac{1}{w_2^2 C_r} \frac{\partial}{\partial} \ddot{\theta}^2}{w_2 L_m} \quad (2.20)$$

It is possible to draw the torque-speed diagram using space vector approach knowing only the motor parameters. Equation (2.17) will be used for this purpose. The modulus of rotor current vector is introduced in (2.18):

$$t_e = \frac{3 p R_r w_2^3 L_m^2 |\bar{u}_s|^2}{2 A_m w_2^4 + B_m w_2^3 + C_m w_2^2 + D_m} \quad (2.21)$$

Similarly to the induction motor in normal configuration, the expression of rotor current space vector is deduced from (2.16):

$$\bar{i}_r = \frac{-j w_2 L_m \bar{i}_s}{R_r + j w_2 \frac{\partial}{\partial} L_r - \frac{1}{w_2^2 C_r} \frac{\partial}{\partial} \ddot{\theta}}$$

The last expression substituted in (2.15) gives:

$$\bar{i}_s = \frac{-\frac{\partial}{\partial t} u_s \frac{\partial}{\partial t} R_r + j w_2 \frac{\partial}{\partial t} L_r - \frac{1}{w_2^2 C_r} \frac{\partial^2}{\partial t^2}}{V_m + j W_m} \quad (2.22)$$

where:

$$V_m = R_s R_r + w_1 w_2 L_m^2 - w_1 w_2 L_s \frac{\partial}{\partial t} L_r - \frac{1}{w_2^2 C_r} \frac{\partial^2}{\partial t^2}$$

$$W_m = w_1 L_s R_r + w_2 R_s \frac{\partial}{\partial t} L_r - \frac{1}{w_2^2 C_r} \frac{\partial^2}{\partial t^2}$$

The input active power is given by:

$$P = \frac{3}{2} \frac{\left| \bar{u}_s \right|^2 \frac{\partial}{\partial t} R_r V_m + w_2 W_m \frac{\partial}{\partial t} L_r - \frac{1}{w_2^2 C_r} \frac{\partial^2}{\partial t^2}}{V_m^2 + W_m^2} \quad (2.23)$$

and the input reactive power is given by:

$$Q = \frac{3}{2} \frac{\left| \bar{u}_s \right|^2 \frac{\partial}{\partial t} R_r W_m - V_m w_2 \frac{\partial}{\partial t} L_r - \frac{1}{w_2^2 C_r} \frac{\partial^2}{\partial t^2}}{V_m^2 + W_m^2} \quad (2.24)$$

are deduced, starting from the expression of the apparent complex power \bar{S} .

The power factor $\cos j_1$ is calculated from the input active and reactive powers using the expression

$$\cos j_1 = \frac{P}{\sqrt{P^2 + Q^2}} \quad (2.25)$$

The mechanical power developed by the motor is expressed by:

$$P_M = t_e W_r \quad (2.26)$$

where W_r the rotor angular speed and can be calculated:

$$W_r = \frac{w_1 - w_2}{p} \quad (2.27)$$

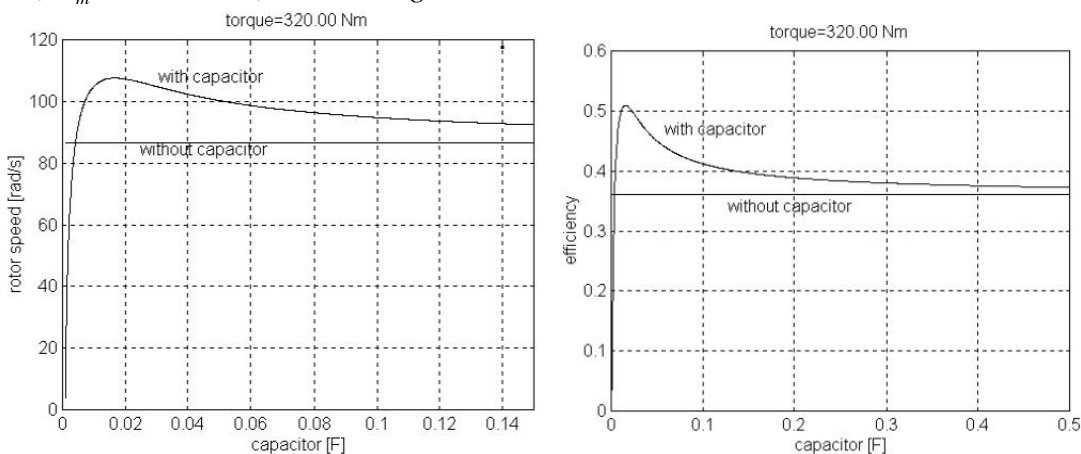
The motor efficiency may be expressed by the ratio between the mechanical power developed by the motor and the active input power (assuming P_M is equal with power transmitted to the motor shaft):

$$h = \frac{P_M}{P} \quad (2.28)$$

Thus, it is possible to calculate power factor and the motor efficiency using (2.25), respective (2.28).

2.1.2.2 Simulations & experimental results

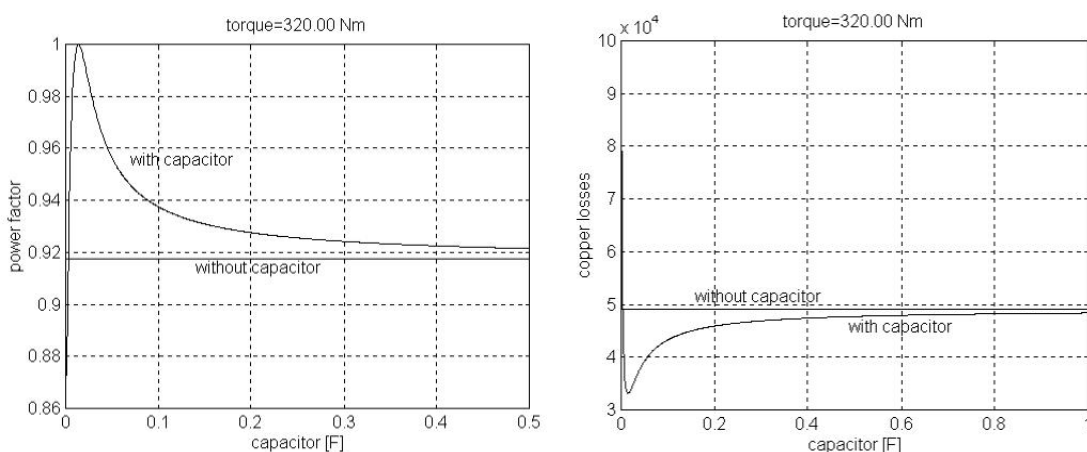
There has been considered for simulation purposes the motor with following parameters:
 Motor 1: 380V, 50Hz, 4 pole, wound motor 20KW, $R_s = 1.64\Omega, R_r = 1.45\Omega, L_{sl} = 3mH, L_{rl} = 3mH, L_m = 163.6mH, J = 0.203kgm^2$



Constant torque characteristics $T=320Nm$

a) Rotor speed/Capacitance

b) Efficiency/Capacitance



Constant torque characteristic $T=320Nm$

c) Power factor/Capacitance

d) Copper Losses/Capacitance

Fig. 2.4 – Motor1 (20KW)

By varying the capacitor value, the rotor speed may be controlled between 0 and the value of the speed that will be reached if the rotor circuit was short-circuited (graphs *a*). When the load torque is in the peak area (the peak torque of the short-circuited rotor configuration), higher rotor speeds may be achieved. If the inserted capacitor is too small, the motor may become unstable.

The efficiency and power factor may be improved if the capacitor is varied. The efficiency has significant improvements in the low and high torque areas. Around the normal operating point the improvement is not that significant. The power factor may be “pushed” towards almost unity in any load condition.

The copper losses for a given load can be controlled if a capacitor is inserted in the rotor circuit. A low capacitance value produces an increase of losses. Increasing the capacitor

decreases the losses and it is possible to reduce these to half of the losses of the short-circuited rotor configuration. A high value of capacitance has the same effects as in normal configuration. The percentage of copper losses reduction is significant for high and low loads. This might give appreciable energy saving in high power motors.

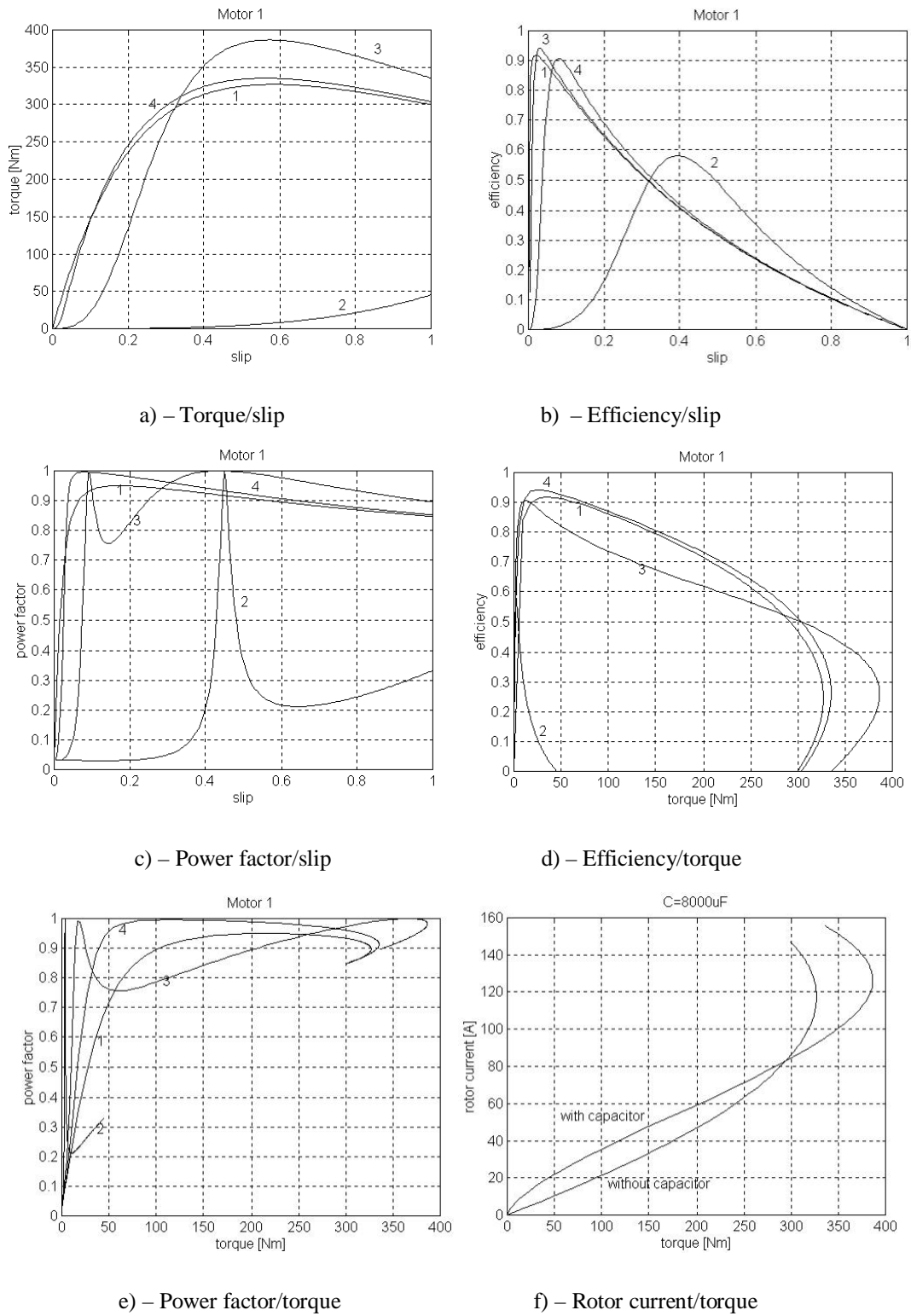


Fig. 2.5 - Motor1 (20KW)

Curve 1: without capacitor; Curve 2: with a capacitor of 300 μF;

Curve 3: with a capacitor of $8000\mu\text{F}$; Curve 4: with a capacitor of $70 \cdot 10^3\mu\text{F}$;

The simulation results can be grouped in 3 categories determined by the value of the used capacitor.

When the value of the capacitance used is very small, the motor is not operable, the starting torque being very small and, in general, the overall efficiency and power factor are smaller than in the case of the short-circuited rotor.

When the capacitor value is high/very high, the performance of the induction motors with externally modified rotor impedance do not differ significantly from the performances of the induction motors in a standard configuration.

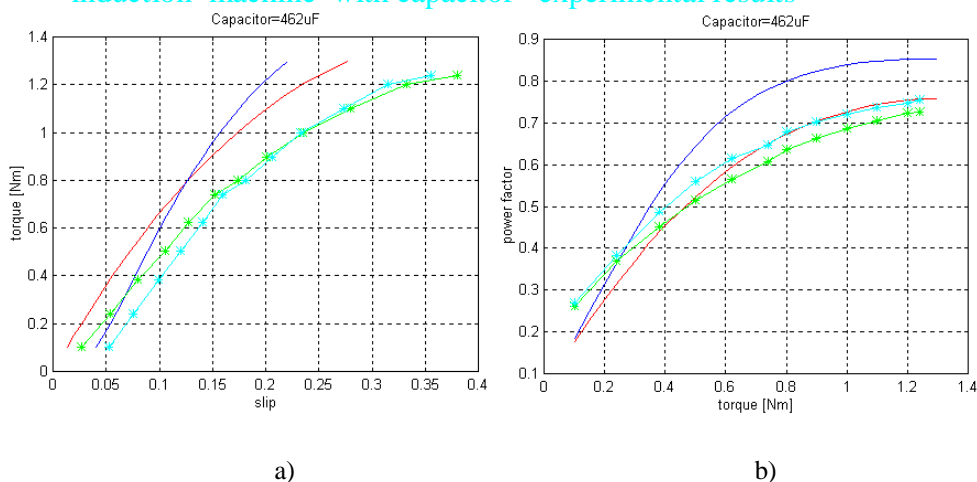
The curves 3 – simulated for intermediate capacitor value- show improvements in performance. It is noticeable the peak and starting torque are improved. The graphs *b* and *c* show that the power factor and efficiency as a function of slip do not have significant improvements – there are situations when they are even worse than those of the short-circuited rotor configuration. This suggests that the modification of rotor impedance is recommended in applications where the speed is not regulated within a narrow band.

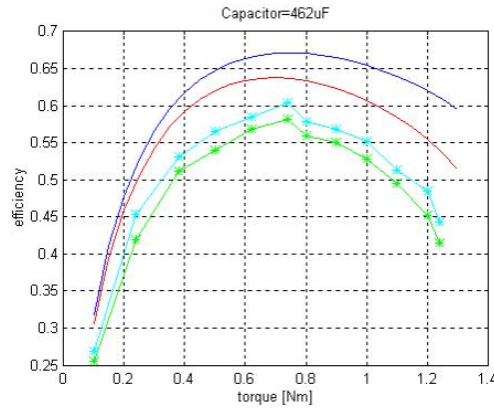
The power factor and efficiency representations from graphs *d*, *e* show that a constant improvement of these performance for the whole range of torque could be achieved by just varying the capacitor value in accordance to the load. The improvement of the performances, keeping the capacitor constant, are achieved for limited torque regions, for the rest of the loads the performances are worse than in short-circuited rotor configuration.

The experimental validation [2.12] of the developed models has been done on a low power wound induction motor (Motor 2) with the estimated parameters $R_s = 85\Omega$, $R_r = 115\Omega$, $L_m = 3.16\text{H}$, $L_{sl} + L_{rl} = 0.72\text{H}$. A set of two power analyzers was used to record the active and apparent power absorbed by the motor. An infrared speed transducer was used to measure the rotor speed. The motor set had a dynamometer with spring balance for monitoring the torque. The load torque on the motor was varied by current supplied to the dynamometer.

The results[2.12] for the test along with those obtained from simulation are presented in the figures 2.6-2.11. The following color code is used for all graphs presented in this chapter (* represents the experimental measurement points):

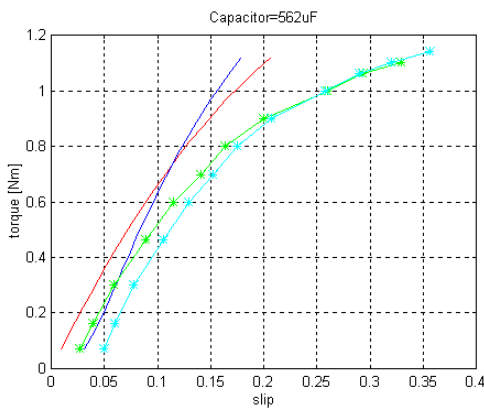
- induction machine without capacitor – simulated results
- induction machine with capacitor – simulated results
- induction machine without capacitor – experimental results
- induction machine with capacitor – experimental results



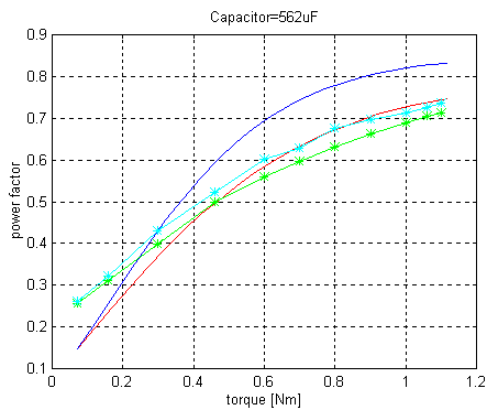


c)

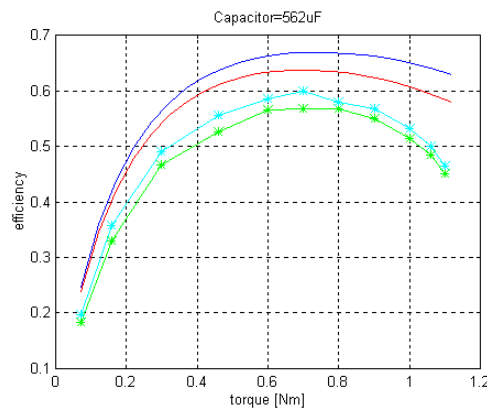
Fig. 2.6 – Motor 2 : a) Torque / slip, b) Power factor / torque, c) Efficiency / torque



a)



b)



c)

Fig. 2.7 - Motor2 a) Torque / slip, b) Power factor / torque, c) Efficiency / torque

The experimental results bear a close resemblance to the theoretically predicted effects of a capacitor in the rotor circuit. For example, the existence of the cross-over for the torque-slip curves is seen in both figures 2.6a and 2.7a. From the cross-over point onwards, the rotor speed is higher in the case of modified rotor than that of the short circuited rotor for the same torque.

Improvement in power factor and efficiency is obtained for the whole torque range; this gives an enhancement in the operation of the motor. It should be noted that the model used in simulations does not include iron and mechanical losses. The open rotor circuit test and the no

load test were undertaken to estimate these losses. The iron losses were estimated to be $13.8W$ and the mechanical losses (at a speed of $1455rpm$) were estimated to be $9.2W$. These values were added to the simulated value of the input power, to present the theoretical efficiency curves.

The second set of tests was performed to study variation of slip, power factor and efficiency as a function of the duty ratio with torque constant at values of

(i) $0.5Nm$

and

(ii) $0.9Nm$

The central capacitor of the H-bridge was $90nF$ for throughout this test.

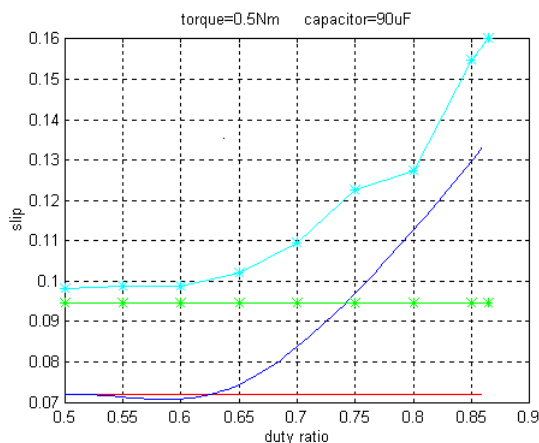


Fig. 2.8 – Motor2 Slip /duty ratio

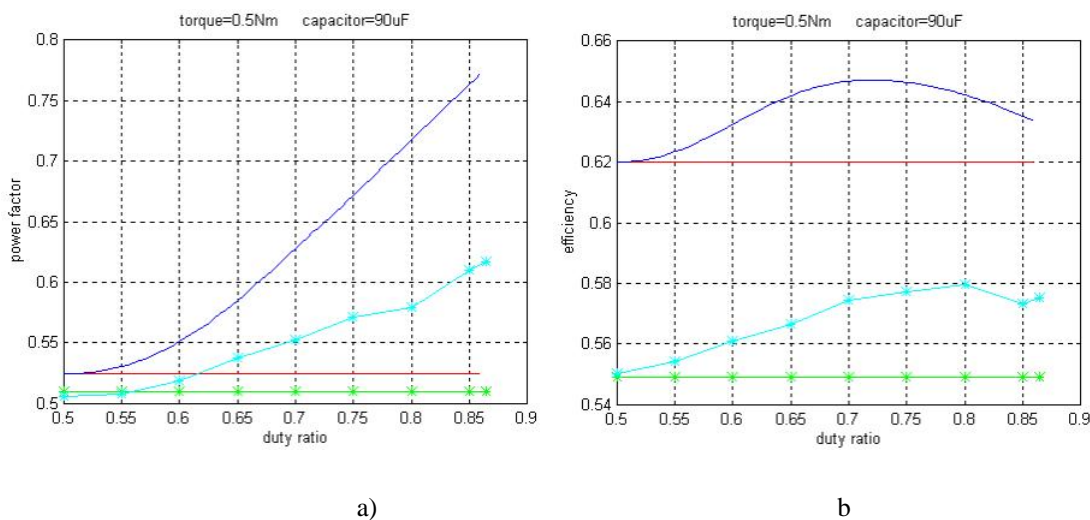


Fig. 2.9 - Motor2 a)Power Factor / duty ratio, b) Efficiency / duty ratio

The experimental and simulation results show the speed variation as a result of the modification of the duty ratio. Usually the insertion of a resistance in the rotor circuit produces a decrease of the rotor speed, an improvement of the power factor but the efficiency decreases. In the presented application the power factor is improved while the efficiencies are at least equal to or better than that for the short-circuited rotor. This, also, gives the possibility of energy-efficient speed control schemes. Also, it has to be noted that the initial efficiency of motor with switched capacitor system in $0.9Nm$ case is lower than in the short circuited configuration. This is

explained by the fact the increase of the load produces higher currents and the losses introduced by the switched capacitor system become more significant.

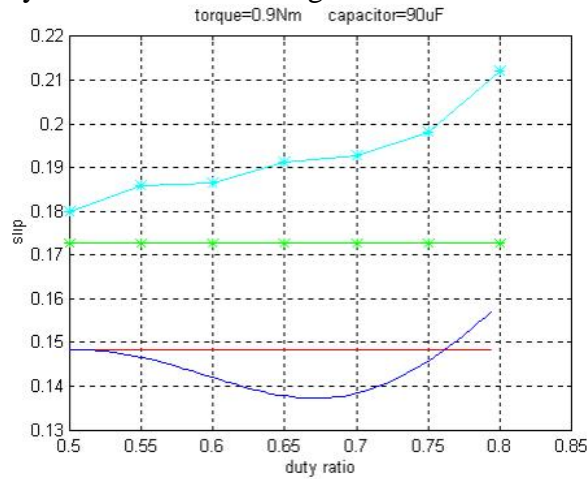


Fig. 2.10 - Motor2 Slip / duty Ratio

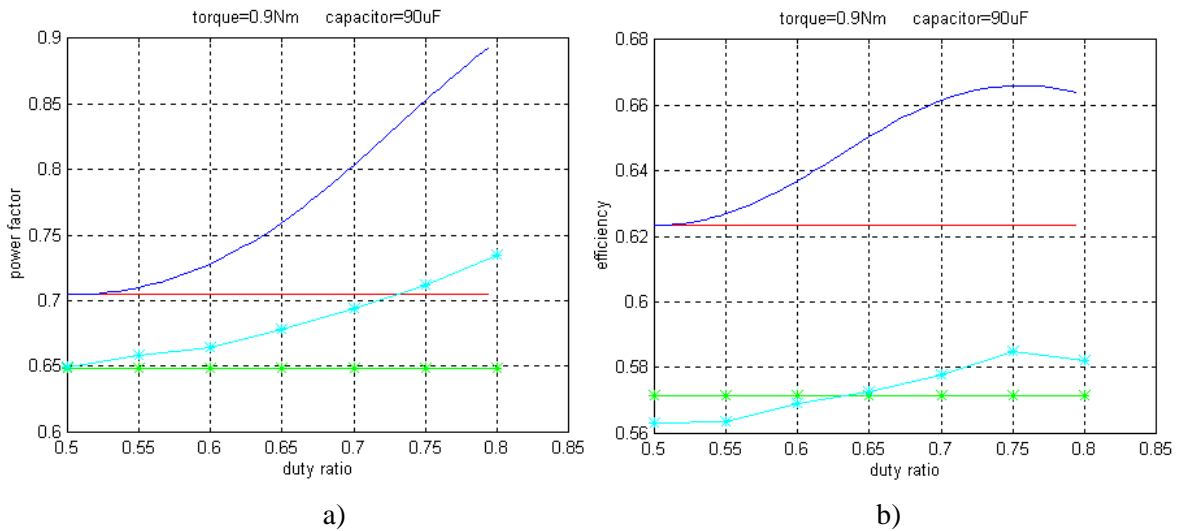


Fig. 2.11 – Motor2 a) Power Factor / duty Ratio, b) Efficiency / duty Ratio

The tests on the low power motor having high rotor resistance show moderate improvements. But as per theoretical analysis, these improvements are expected to be significant in the case of high power motors. However slip-ring drives are used in very high power applications. At these powers, a 1% increase in efficiency is very significant.

2.1.3 Single phase induction machine with variable capacitor

As already mentioned, the need for a capacitor that can dynamically adjust its value based on the process demands is required as well in the case of a single phase induction motor having such emulated capacitor instead of 2 capacitors (one for start, the other one for steady state regime) with the intention to optimise the motor performance.

Even in the case the capacitor is electronic emulated the value for different stationary points has to be calculated considering that the rotating field is circular.

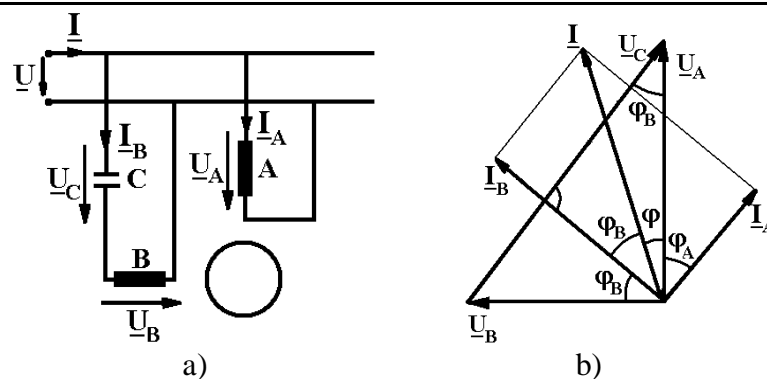


Fig. 2.12 - a) General diagram of single phase machine;

b) The phasor diagram of the single phase machine

The schematic of the single phase induction motor is presented in figure 2.12a.

The principal winding A is electric orthogonal with the auxiliary winding B containing the series capacitor. It results the phasor diagram presented in figure 2.12b[2.7]:

The conditions that the rotating field must be circular conduct to following relations:

$$U_A @p\sqrt{2}fN_A k_{BA} F \quad (2.29)$$

$$U_B @p\sqrt{2}fN_B k_{BB} F_h \quad (2.30)$$

where: U_A is voltage accross main winding, U_B is voltage accross auxiliary winding, U_C is voltage accross emulated capacitor; N_A , N_B represent the number of turns; k_{AA} , k_{BA} represent windings coefficients; F_h is rotating flux.

In this rotating field the voltage ratio is:

$$\frac{U_B}{U_A} = \frac{N_B k_{BB}}{N_A k_{BA}} = \frac{1}{k_E} \quad (2.31)$$

or in complex:

$$\underline{U}_B = j \frac{1}{k_E} \underline{U}_A \quad (2.32)$$

As the rotating field has been considered circular results:

$$\underline{q}_B = j \underline{q}_A \quad (2.33)$$

and

$$N_B k_{BB} \underline{I}_B = j N_A k_{BA} \underline{I}_A \quad (2.34)$$

or:

$$\underline{I}_B = j k_E \underline{I}_A \quad (2.35)$$

From fig. 2.12b results:

$$\operatorname{tg} j_B = \frac{U_B}{U_A} = \frac{1}{k_E} = c \quad (2.36)$$

and the relations:

$$j_A = j_B \quad (2.37)$$

$$j = \frac{\rho}{2} - 2j_B \quad (2.38)$$

It follows:

$$U_C = \frac{U_B}{\sin j_B} \quad (2.39)$$

and

$$I_B = \frac{U_C}{X_C} = \frac{U_B}{\sin j_B} \omega C = I \cos j_B \quad (2.40)$$

In this case the capacitor value C is:

$$C = \frac{I_B \sin j_B}{\omega U_B} \quad (2.41)$$

The active power from the grid is:

$$P_a = U_A I \cos j \quad (2.42)$$

and from (2.40) results:

$$C = \frac{I \cos j_B \sin j_B}{\omega U_B} = \frac{P_a \cos j_B \sin j_B}{U_A U_B \cos j} = \frac{\frac{1}{2} P_a \sin 2j_B}{\omega U_A U_B \cos j} \quad (2.43)$$

add from (2.31) and (2.35) results:

$$C = \frac{\frac{1}{2} P_a \cos j_B}{\omega U_A^2 \sin j_B} = \frac{P_a N_A k_{BA}}{2 \omega U_A^2 N_B k_{BB}} = \frac{P_{mec} k_E}{2 \omega U_A^2 h} \quad (2.44)$$

The capacitor C power is:

$$P_C = U_C I_B = \frac{U_B I_B}{\sin j_B} \quad (2.45)$$

and the apparent power from the grid:

$$P = U_A I = U_C \cos j_B I = \frac{U_B}{\sin j_B} \frac{\cos j_B I_B}{\cos j_B} = \frac{U_B I_B}{\sin j_B} \quad (2.46)$$

That means:

$$P_C = P \quad (2.47)$$

The active powers for the windings A and B are equal:

$$P_{Aa} = U_A I_A \cos j_A = P_{Ba} = U_B I_B \cos j_B = \frac{1}{2} P \quad (2.48)$$

2.1.3.1 Dynamic model of the single phase motor

The dynamic model of the motor is based on quadrature-axis theory, in the stator bound reference system [2.9]. The stator currents and voltages are considered as known based on measurements (e.g. using a data acquisition board). These allows to estimate the rotor and air

gap electrical data variations that cannot be directly measured, the electromagnetic torque variation and the angular speed of the rotor during the transient operation mode.

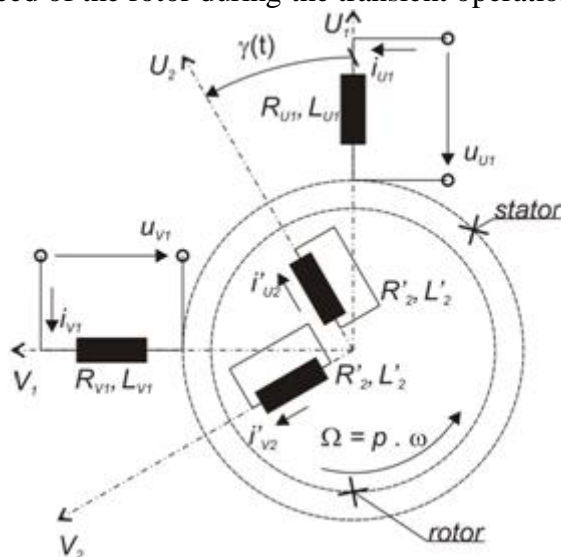


Fig. 2.13 – Two axis winding system

The quadrature-axis theory is used in a simplified hypothesis [2.10-2.11] considering the followings:

- space-distribution of the phase current ampere-turns is sinusoidal;
- the magnetic saturation and the magnetic forces in the ferromagnetic motor circuit are neglected;
- resistance and permeances along the stator winding differ;
- the axes of coordinates are bound to the anisotropic part of the machine, i.e. the stator.

The single/two phase induction motor dynamic model approach considers also the followings based on the representation from figure 2.13

- the two stator windings ratios are related with the same turns-number; the rotor binding ratio is related to the equivalent turns-number of the stator;
- the squirrel cage rotor is equivalent with a fixed two phase winding orientated after axis of coordinates.

The symbols indexed with 1 (e.g. R_{d1} – resistance of the main winding) are related to the stator and while those indexed to 2 are related related to the rotor. Indexes d and q are relating to the data bound at the axes (Od), (Oq) respectively.

It is noted ω_1 the angular frequency of the stator data and $\omega = \omega_1 / p$ the angular speed of the rotor. (ω - angular frequency of the machine; p - pair poles) The set of dynamic equations of the motor in matrix representation is as follows [2.5, 2.6]:

$$\begin{bmatrix} \dot{u}_{d1} \\ \dot{u}_{q1} \end{bmatrix} = \begin{bmatrix} R_{d1} & R_{q1} \end{bmatrix} \times \begin{bmatrix} \dot{e}_{d1} \\ \dot{e}_{q1} \end{bmatrix} + \frac{d}{dt} \times \begin{bmatrix} e_{d1} \\ e_{q1} \end{bmatrix} \quad (2.49)$$

$$\begin{bmatrix} \dot{u}_{d2} \\ \dot{u}_{q2} \end{bmatrix} = \begin{bmatrix} R_{d2} & R_{q2} \end{bmatrix} \times \begin{bmatrix} \dot{e}_{d2} \\ \dot{e}_{q2} \end{bmatrix} + \frac{d}{dt} \times \begin{bmatrix} e_{d2} \\ e_{q2} \end{bmatrix} + (\omega - \omega_1) \times \begin{bmatrix} e_{q2} \\ -e_{d2} \end{bmatrix} \quad (2.50)$$

$$\begin{bmatrix} \dot{Y}_{d1} \\ \dot{Y}_{q1} \end{bmatrix} = \begin{bmatrix} L_{d1} & L_{q1} \\ L_{hd} & L_{hq} \end{bmatrix} \begin{bmatrix} \dot{e}_{q1} \\ \dot{e}_{q2} \end{bmatrix} + \begin{bmatrix} L_{hd} & L_{hq} \\ L_{hd} & L_{hq} \end{bmatrix} \begin{bmatrix} \dot{e}_{d1} \\ \dot{e}_{d2} \end{bmatrix} \quad (2.51)$$

$$\begin{bmatrix} \dot{Y}'_{d2} \\ \dot{Y}'_{q2} \end{bmatrix} = \begin{bmatrix} L'_{d2} & L'_{q2} \\ L_{hd} & L_{hq} \end{bmatrix} \begin{bmatrix} \dot{e}'_{q2} \\ \dot{e}'_{q1} \end{bmatrix} + \begin{bmatrix} L_{hd} & L_{hq} \\ L_{hd} & L_{hq} \end{bmatrix} \begin{bmatrix} \dot{e}'_{d1} \\ \dot{e}'_{q1} \end{bmatrix} \quad (2.52)$$

$$M_{el} = p(Y_{d1} i_{q1} - Y_{q1} i_{d1}) = p(Y'_{q2} i'_{d2} - Y'_{d2} i'_{q2}) = M_L + \frac{J}{p} \frac{dw}{dt} \quad (2.53)$$

To perform an on-line compensation of the drive, the continuous-time dynamic model of the motor in state-space representation [2.11] must be transformed into a set of differential equations. A simple way to perform this task is to use the Euler-Cauchy method [2.10]. The following set of differential equations describing the dynamics of the drive is obtained.

Flux linkages differential equations:

$$Y_{d1}(k+1) = Y_{d1}(k) + Dt [u_{d1}(k) - R_{d1}(k) i_{d1}(k) + w_1 Y_{q1}(k)] \quad (2.54)$$

$$Y_{q1}(k+1) = Y_{q1}(k) + Dt [u_{q1}(k) - R_{q1}(k) i_{q1}(k) - w_1 Y_{qd}(k)] \quad (2.55)$$

$$Y'_{d2}(k+1) = Y'_{d2}(k) + Dt [-R'_{d2}(k) i'_{d2}(k) - (w(k) - w_1) Y'_{q2}(k)] \quad (2.56)$$

$$Y'_{q2}(k+1) = Y'_{q2}(k) + Dt [-R'_{q2}(k) i'_{q2}(k) + (w(k) - w_1) Y'_{d2}(k)] \quad (2.57)$$

The currents equations are in the matrix form:

$$[I] = [L]^{-1} [Y] \quad (2.58)$$

where: $[I]$ is the currents vector, $[L]$ is the inductances matrix and $[Y]$ is the flux linkages vector,

$$[I] = \begin{bmatrix} \dot{e}_{d1} \\ \dot{e}_{q1} \\ \dot{e}'_{d2} \\ \dot{e}'_{q2} \end{bmatrix} \quad [Y] = \begin{bmatrix} \dot{e}_{d1} \\ \dot{e}_{q1} \\ \dot{e}'_{d2} \\ \dot{e}'_{q2} \end{bmatrix} \quad (2.59)$$

and

$$[L] = \begin{bmatrix} L_{d1} & 0 & L_{hd} & 0 \\ 0 & L_{q1} & 0 & L_{hq} \\ L_{hd} & 0 & L'_{d2} & 0 \\ 0 & L_{hq} & 0 & L'_{q2} \end{bmatrix} \quad (2.60)$$

The angular speed equation:

$$w(k+1) = w(k) + \frac{pDt}{J} [p(Y_{d1} i_{q1}(k) - Y_{q1}(k) i_{d1}(k)) - M_w] \quad (2.61)$$

The series capacitor provides the phase shift of the auxiliary winding supply voltage therefore following relation has to be added:

$$u_{q1}(k) = u_{d1}(k) + \frac{1}{C} \sum_{i=0}^{k-1} i_{q1}(i) Dt(i) \quad (2.62)$$

The set of differential equations (2.54) – (2.62) may be used to determine the transients such as the drive's startup from zero speed.

2.1.3.2 Simulation & experimental results

For the simulations as well as for experiment measurements [2.5, 2.6], there was used a motor-Motor 3- with the following parameters:

- Main winding resistance $R_{d1} = 20,8W$,
- Auxiliary winding resistance $R_{q1} = 57,5W$,
- Main winding self inductance, axis Od, $L_{d1} = 0,358H$
- Auxiliary winding self inductance, axis Oq, $L_{q1} = 0,665H$
- Rotor self inductance, axis Od, $L'_{d2} = 0,523H$
- Rotor self inductance, axis Oq, $L'_{q2} = 1,099H$
- Stator-rotor mutual inductance, axis Od, $L_{hd} = 0,275H$
- Stator-rotor mutual inductance, axis Oq, $L_{hq} = 0,275H$

During the operation of the motor, it was considered a step change of the emulated capacitor as shown in figure:

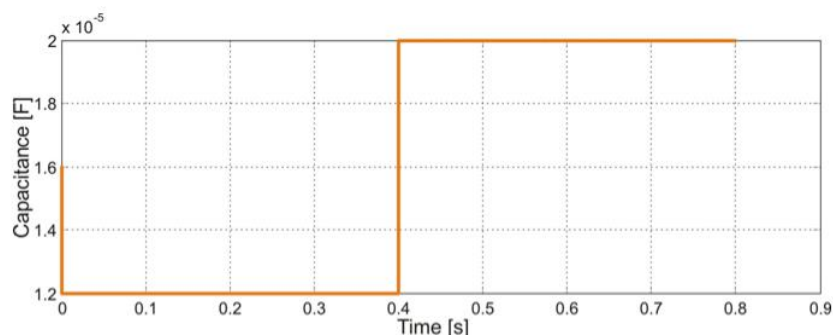


Fig. 2.14 - The step deviation of the electronically switched capacitance

A step deviation of the capacitance was imposed at $0.4s$ starting from the stationary operation. An increased value of capacitance results in an increased value of the voltage to the auxiliary winding terminals and subsequently an increased value of the corresponding current component.

The loci of the stator's flux linkage components before and during transients are presented in figure 2.15. As seen onto these plots the sudden change of the capacitance results in phase angle and amplitude modifications. Subsequently the amplitude of the electromagnetic torque modifies too. The amplitude of the instantaneous values of electromagnetic torque generally increases when the capacitance is increased.

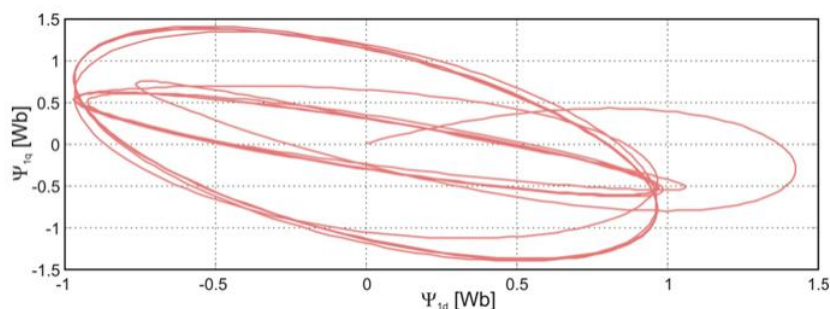
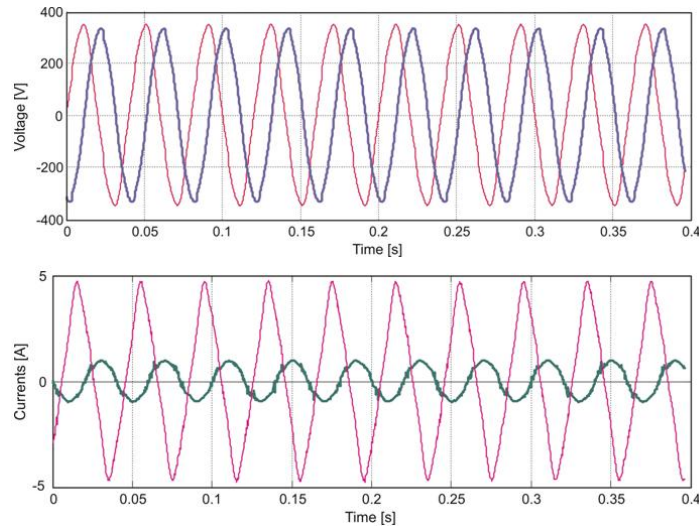
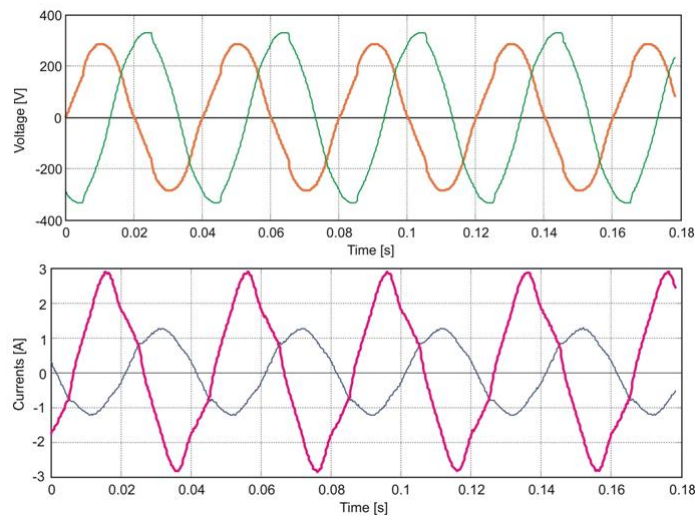


Fig. 2.15 - Loci of the stator's magnetic flux linkage components at the split capacitance sudden change

In purpose to predetermine by measurements the control law, experiments in stationary operating regime with different values of the split capacitance were performed. During measurements three emulated capacitors were used; the values of the capacitance were $36\mu F$, $16\mu F$ and $10\mu F$.

**Fig. 2.16** -Measured values of stator voltages (above) and currents (below) for a given split capacitance of $36\mu F$; thin lines – main winding variables, thick lines – auxiliary winding variables**Fig. 2.17** - a) Measured values of stator voltages (above) and currents (below) for a given split capacitance of $16\mu F$; thin lines – main winding variables, thick lines – auxiliary winding variables

In figures 2.16, 2.17a and 2.17b the plots of phase voltages and currents for given values of the split capacitance are presented. A decrease of the capacitance from $36\mu F$ to $10\mu F$ results in a large modification of amplitudes of the phase currents; also results a large modification of the ratio between the amplitude of these currents. During transients, the currents amplitude modifications take effect if the transient duration is larger than the mains frequency. This assumption is correct in many practical cases such as fans and washing machines.

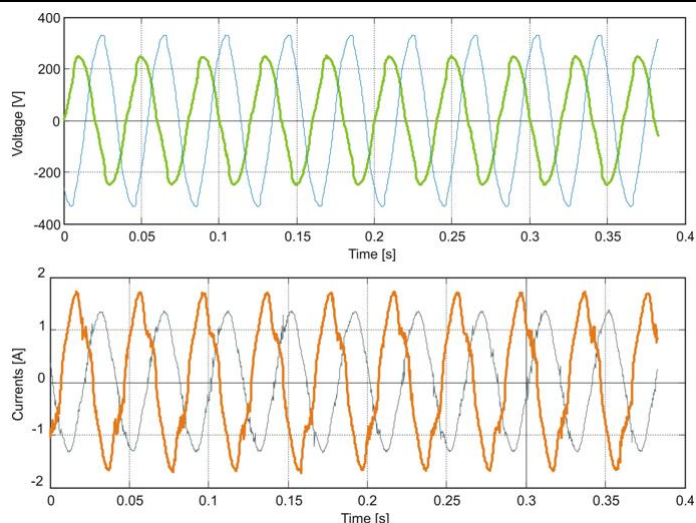


Fig. 2.17 - b) Measured values of stator voltages (above) and currents (below) for a given split capacitance of $10\mu F$; thin lines – main winding variables, thick lines – auxiliary winding variables

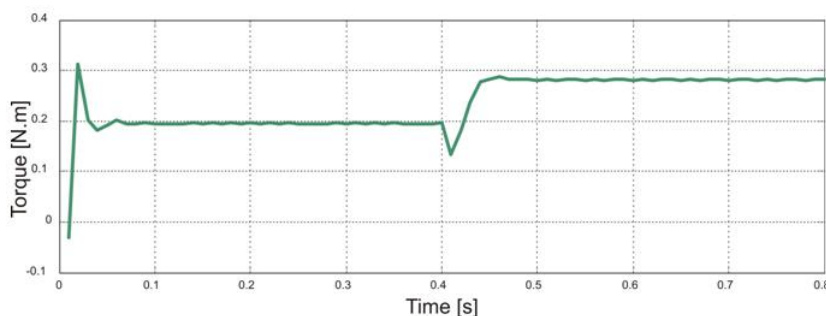


Fig. 2.18 - Torque response to step input variation of capacitance from steady state operation of the drive.

Excessively small values of the split capacitance results in a large difference between the phase currents. An optimum of currents amplitude ratio is achieved for $16\mu F$. The electronically switched capacitor allows on-line refinement. Simulation on computer have proven that for no load steady-state operation the optimum value of the capacitance is around $18\mu F$. However the exact value of the capacitance is to be determined on-line by the control system itself.

To determine the parameters of the suitable controller, the torque variation at sudden capacitance changes was simulated on a computer. The average values on a phase period of the electromagnetic torque were computed and plotted, Figure 2.18. The drive behaves as a first-order element with a time constant of 0.2 s and a gain of $3.5 \times 10^5\text{ [N}\cdot\text{m/F]}$.

2.2 Hemodynamics modeling

According to World Health Organisation, Cardiovascular diseases (CVD) are the number 1 cause of death globally: more people die annually from CVDs than from any other cause. An estimated 17.5 million people died from CVDs in 2012, representing 31% of all global deaths. Of these deaths, an estimated 7.4 million were due to coronary heart disease and 6.7 million were due to stroke. Heart attacks and strokes are usually acute events and are mainly caused by a blockage that prevents blood from flowing to the heart or brain. The most common reason for this is a build-up of fatty deposits on the inner walls of the blood vessels that supply the heart or brain.

People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes or already established disease) need early detection and management using counseling and medicines, as appropriate.

In spite of the significant improvements in medical imaging and other diagnostic modalities, the incidence of premature morbidity and mortality for CAD patients is still very high, the main reason being the lack of accurate in-vivo and in-vitro patient-specific estimates for diagnosis and progression of the disease. For example, in the case of coronary stenosis, accurate estimates of the anatomy (amount of narrowing blockage in the coronary) as seen in the diagnostic images, can vastly underestimate or overestimate the severity of the blockage. For a functional assessment of such a blockage, analysis of the subsequent disease progression, and assessment of the best intervention/surgical option for an individual patient, it is important to incorporate multi-faceted information from the hemodynamics and cellular mechanisms from multiple scales. Incorporating such multi-scale information in a complex computational model has been difficult in the past due to the high computational demands. Therefore, High Performance Computing platforms have enabled such simulations on as it is presented in the next chapter.

One of the major difficulties regarding the precise modeling of the human cardio-vascular system is the fact that it represents a closed circuit with a high degree of interdependence between the individual compartments. The blood flow characteristics in a certain segment of the system (the local hemodynamics) are closely related to the global dynamics of the system [2.13]. Studying the local blood flow is very important, since certain pathologies, like the local thickening of the blood vessel or the formation of a stenosis, are strongly influenced by the local hemodynamics.

On the other side certain local changes, like the modification of the vascular lumen, may lead to a global redistribution of blood flow, triggering some compensatory mechanism which assures a high enough flow rate in the distal part of the affected vessel. 3D or full-scale blood flow simulations are computationally very expensive and can only be performed for a reduced number of vessels [2.14]. Both the reciprocal influence between the systemic or global hemodynamics and the local one, and the high computational requirements of 3D simulations, have led to the concept of geometrical multi-scale modeling of blood flow, which is here applied in order to analyze the coronary circulation.

Thus, only the local regions of interest inside the coronary arterial tree, e.g. the segments which contain a narrowing and plaque deposits, are simulated using full 3D models, while the rest of the circulation is represented through reduced-order models (1D models for the large arteries and lumped models for the small arteries and microvasculature). Reduced-order models produce reliable results in terms of pressure and flow rate waveforms (1D models), they correctly take into account the effect of the distal vessels and of the microvasculature (lumped models) and lead to execution times which are more than two orders of magnitude smaller than the corresponding 3D simulations.

There is presented in figure 2.19 an overview of a multi-scale modeling approach. A heart model (lumped parameter heart model parameterized through patient-specific data) is coupled at the root of the aorta. The aorta and the large arteries are simulated through 1D blood flow models since these produce reliable results in terms of pressure and flow rate values and take into account wave propagation phenomena. The coronary microvascular beds are modeled through lumped or 0D models: the systemic beds are represented by regular windkessel elements, while coronary beds are represented by specialized lumped models which account for the influence of the myocardial contraction on the flow waveform [2.15].

For the coronary arterial tree, the large (epicardial) vessels are simulated through 1D models. The stenosis segments cannot be simulated using this type of models since there is a high variation in cross-sectional area and the shape of the stenosis influences the blood flow behavior

and especially the trans-stenotic pressure drop which plays a major role in the assessment of the functional importance of such a stenosis.

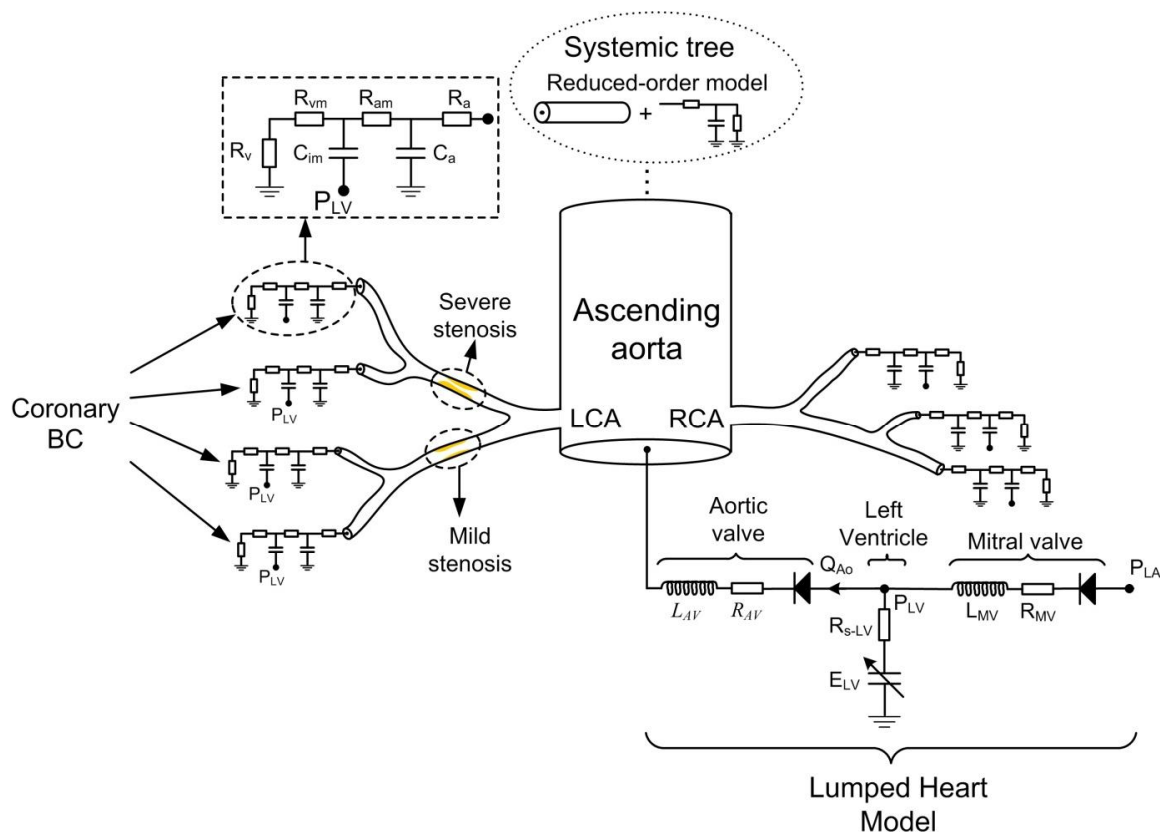


Fig. 2.19 - Reduced-order model of the coronary circulation

An important aspect for the clinical decision making is the modeling of the hyperemic state. Hyperemia is obtained either through intense exercise or by drugs that are administered either intravenously or intracoronary. Since measurements cannot be taken reliably during intense exercise, drug-induced hyperemia is preferred. Intravenous administration of vasodilators leads to a slight increase of heart rate and decrease in blood pressure [2.16]. For simulations, the effect of an intracoronary vasodilator can be extended infinitely and this alternative to obtain hyperemia does not influence heart rate and blood pressure [2.16]. The resistance and compliance of the systemic or coronary lumped models (for the normal rest state) is obtained by imposing a structured-tree outflow boundary condition [2.17]. These impedance values are then adapted for the patient-specific model by a parameter estimation process. The hyperemic state is modeled through a corresponding decrease in the microvascular resistances, as caused by the administration of intracoronary adenosine [2.18] (epicardial arteries are not influenced by vasodilators [2.19]) and leads to a 3 to 5 fold increase in the coronary flow.

The execution time is crucial and hence rigid wall 3D models were used and not 3D fluid-structure interaction (FSI) whose execution times are more than two times higher [2.20]. This aspect does not influence the overall results, since the elasticity in the stenosis region is not important, but in order to correctly represent the wave propagation phenomena inside the coronary tree, 0D interface models have to be included at the interfaces between the compliant 1D and rigid 3D models (these interface models concentrate the compliance of the 3D segments at its interfaces). One aspect which is very important in the coronary circulation, and which contributes to the big discordance between morphological and functional importance of a stenosis is the presence of collateral flow which can render a morphologically important stenosis

into a functionally insignificant one. Depending on the patient specific vessel morphology, the collateral flow can be modeled both through anastomotic large vessels (with 1D models) or through microvascular vessels which supply the affected region with blood (modeled through lumped elements as in Figure 2.19).

A major component of the blood flow model is a lumped heart model. Several models have been proposed that can determine the pressure and the flow in the different heart chambers. Several parameters like contractility, stroke volume, time-to-maximum, dead volume (V_0) or heart rate can be adapted in order to account for different states of the body and to personalize the model. A varying elastance model (equation 2.63) has been used, which is coupled to the aortic input through a lumped aortic valve model (Figure 2.19) and indirectly coupled to the specialized microvascular models of the coronary arterial tree through the left ventricular pressure:

$$E(t) = \frac{P_{LV}(t)}{V_{LV}(t) - V_0}. \quad (2.63)$$

A goal of the scientific activity has been in bringing contributions in the development of integrated cardiovascular models that could allow for a non-invasive assessment of the stenosis in blood vessels.

2.2.1 Fractional Flow Reserve

Fractional Flow Reserve (FFR) is an invasive methodology that „measures” the degree of stenosis in a suspected affected vessel. Measuring FFR is an invasive procedure: a catheter is inserted together with a pressure sensor from the femoral or radial artery through coronary arteries of the epicardium up to the stenosis area. The average values of the pressures before and after stenosis are measured. It is necessary to administrate a solution - e.g adenosine - that induces hyperemia to perform these measurements.

FFR is the ratio between maximum flow-hyperemia- in the artery affected by stenosis and hypothetical flow in the absence of stenosis:

$$FFR = \frac{Q_{\max}^S}{Q_{\max}^N} \quad (2.64)$$

Based on the equivalent Ohm law in the hydraulic domain the flow is the ratio between pressure and resistance, therefore:

$$FFR = \frac{Q_{\max}^S}{Q_{\max}^N} = \frac{(P_d - P_v)/R_{\max}^S}{(P_a - P_v)/R_{\max}^N} \quad (2.65)$$

where P_d is the average pressure measured before stenosis, P_a is the average aortic pressure, P_v is the average vein pressure, R_{\max}^S is the microvascular resistance in the presence of the stenosis in the hypermia state and R_{\max}^N is the hypothetical microvascular resistance without stenosis in the hypermia state, as well. Assuming [2.21] that the two resistances are equal and that the vein pressure is negligible compared to the two artery pressures, then:

$$FFR = \frac{Q_{\max}^S}{Q_{\max}^N} = \frac{P_d}{P_a} \quad (2.66)$$

The usage of FFR is done with the purpose to determine if the stenosis is significant from the functional point of view. The normal value of FFR is 1.00 while a critical “border” value is approx. 0.75-0.8. A smaller FFR value implies the implant of a stent for the respective patient.

FFR has been used to validate non-invasive algorithms of estimating the degree of stenosis.

2.2.2 One dimensional blood flow models

The one-dimensional blood flow model is derived from the three-dimensional Navier-Stokes equations based on a series of simplifying assumptions [2.22]. The governing equations ensuring mass and momentum conservation are as follows:

$$\frac{\partial A(x,t)}{\partial t} + \frac{\partial q(x,t)}{\partial x} = 0 \quad (2.67)$$

$$\frac{\partial q(x,t)}{\partial t} + \frac{\partial}{\partial x} \left(\frac{\alpha}{\rho} q^2(x,t) \right) + \frac{A(x,t)}{r} \frac{\partial p(x,t)}{\partial x} = K_R \frac{q(x,t)}{A(x,t)} \quad (2.68)$$

where x denotes the axial location and t denotes the time. $A(x,t)$ is the cross-sectional area, $p(x,t)$ the pressure, $q(x,t)$ the flow rate, and ρ is the density. Coefficients α and K_R account for the momentum-flux correction and viscous losses due to friction respectively. For a parabolic velocity profile, $K_R = -8\rho\nu$ and $\alpha = 4/3$, with ν being the kinematic viscosity of the fluid.

A state equation, which relates the pressure inside the vessel to the cross-sectional area, is used to close the system of equations. When the vessel wall is modeled as a pure elastic material, the following relationship holds:

$$p(x,t) = Y_{el}(A) + p_0 = \frac{4}{3} \frac{Eh}{r_0} \left(\frac{A}{A_0} \right)^{\frac{3}{2}} - \sqrt{\frac{A_0(x)}{A(x,t)}} \ddot{\phi} + p_0 \quad (2.69)$$

where E is the Young modulus, h is the wall thickness, r_0 is the initial radius corresponding to the initial pressure p_0 , and A_0 is the initial cross-sectional area. The elastic wall properties are estimated using a best fit to experimental data [2.17].

Alternatively, a viscoelastic wall model can also be used. To include viscoelasticity, the vessel wall is considered to be a Voigt-type material [2.23], for which the tensile stress depends on both the tensile strain and the time-derivative of the strain [2.24]:

$$p(x,t) = Y_{el}(A) + Y_v(A) + p_0 = \frac{4}{3} \frac{Eh}{r_0} \left(\frac{A}{A_0} \right)^{\frac{3}{2}} - \sqrt{\frac{A_0}{A(x,t)}} \frac{\ddot{\phi}}{\partial} + \frac{g_s}{A\sqrt{A}} \frac{\partial A}{\partial t} + p_0 \quad (2.70)$$

where g_s is the viscoelastic coefficient, defined by:

$$g_s = \frac{T_s \tan \Phi_s}{4\rho} \frac{hE}{1 - \sigma^2} \quad (2.71)$$

where T_s is the wave characteristic time (usually taken equal to the systolic period $\sim 0.24s$), Φ_s is the viscoelastic angle (10°), while σ is the Poisson ratio (the material is considered to be incompressible for $\sigma = 0.5$). As in the case of elastic modeling, the viscoelastic coefficient is considered to be non-uniform in space (i.e. $g_s = g_s(x)$).

The presence of the viscoelastic component in (2.70) introduces an additional term in the momentum conservation equation:

$$\frac{\partial q}{\partial t} + \frac{\partial}{\partial x} \left(\frac{\alpha}{\rho} q^2 \right) + \frac{A}{r} \frac{\partial Y_{el}}{\partial x} + \frac{A}{r} \frac{\partial Y_v}{\partial x} = K_R \frac{q}{A} \quad (2.72)$$

Spatial and temporal dependencies of the quantities have been omitted for notational clarity. At each bifurcation, the continuity of flow and total pressure is imposed,

$$q_p = \dot{\mathbf{a}}_i(q_d)_i \quad (2.73a)$$

$$p_p + \frac{1}{2} r \frac{q_p^2}{A_p^2} = (p_d)_i + \frac{1}{2} r \frac{(q_d)_i^2}{(A_d)_i^2} \quad (2.73b)$$

where subscript p refers to the parent, while subscript d refers to the daughter vessels.

The two main schemes for the numerical solution of 1D models are the method of characteristics and the two-step Lax-Wendroff method. As indicated in [2.25], the latter is a second order method while the former is only of first order. Therefore, the Lax-Wendroff method has been used for all interior points of the domain, and only the outflow point has been subject to different implementation types.

2.2.3 Stenosis model

The patient-specific coronary tree is coupled with stenosis segments(see figure 2.20). One of the assumptions made during the derivation of the reduced-order model is that the axial velocity is dominant and the radial components are negligible. This assumption holds well for normal, healthy vessels, but in case of sudden changes in lumen diameter, e.g. for a stenosis, the radial components can no longer be excluded. Much attention has been directed towards the local velocity fields, but for the overall functional assessment the trans-stenotic pressure drop is the most important. Previous works have included semi-empirical stenosis models in 1D blood flow models [2.35, 2.36] and have obtained good results compared to full-scale models.

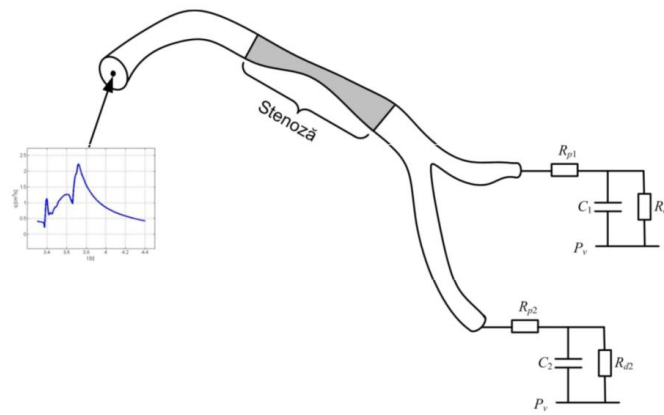


Fig. 2.20 – A stenosed vessel

The pressure drop is expressed as a sum of three terms (viscous term, turbulent or Bernoulli term and inertance term):

$$DP_s = \frac{rK_v}{2pr_0^3} q + \frac{rK_t}{2A_0^2} \frac{\partial A_0}{\partial A_s} - 1 \frac{\partial^2}{\partial t} |q| q + \frac{rK_u L_s}{A_0} \frac{\partial q}{\partial t} \quad (2.74)$$

where μ is the blood viscosity, L_s is the stenosis length, K_v , K_t and K_u are the viscous, turbulent and inertance coefficient respectively (quantities indexed with 0 refer to the normal vessel while s refers to the stenosis). The segments treated as stenosis segments are coupled to the regular segments by considering continuity of total pressure and of flow rate.

2.2.3 Outflow boundary condition implementation

There are three main types of boundary conditions, which are generally accepted for one-dimensional blood flow simulations. The first one is the resistance boundary condition (2.75), which considers a flow rate proportional to the pressure [2.26]. The drawbacks are that it is

difficult to choose the correct value for the peripheral resistance and the pressure and the flow are forced to be in phase.

$$p = q \times R \quad (2.75)$$

The second type of outflow boundary condition is the Windkessel boundary condition (2.76) (figure 2.21), which is a three-element model, consisting of two resistances and a compliance. The total resistance is equal to the value chosen for the resistance boundary condition and the ratio R_1/R_2 has been considered as 0.25 [2.26]. This model allows for a phase lag between pressure and flow rate and is much closer to the physiological data. Again the problem consists in choosing adequate values for the resistances and for the compliance.

$$\frac{\mathbb{P}p}{\mathbb{P}t} = R_1 \frac{\mathbb{P}q}{\mathbb{P}t} - \frac{p}{R_2 C_T} + \frac{q(R_1 + R_2)}{R_2 C_T} \quad (2.76)$$

The third type of outflow boundary condition is the structured tree outflow boundary condition [2.17]. This condition is derived by considering the vascular tree lying downwards from the outflow point as a structured binary tree.

There have been considered the first two types of conditions from the three described above. This choice has been made because one of the main aspects considered during the analysis is the execution time, and for the structured tree outflow condition, most of the time is spent on the summation of previous values (introduced by the convolution integral) and the type of implementation plays a less important role.

2.2.3.1 Implicit Lax-Wendroff

The first type of implementation is the implicit implementation based on the Lax-Wendroff method. In this case four equations are used. Two of them represent the numerical scheme at the outflow point and the other two equations are derived from the outflow boundary condition at half of the time step and at full time step. The resistance boundary condition at full time step is:

$$p^{n+1} = q^{n+1} \times R \quad (2.77)$$

and for the Windkessel boundary condition:

$$\frac{p^{n+1} - p^n}{\Delta t} = R_1 \frac{q^{n+1} - q^n}{\Delta t} - \frac{p^{n+1}}{R_2 C_T} + \frac{q^{n+1}(R_1 + R_2)}{R_2 C_T} \quad (2.78)$$

The outflow equations are called implicit because both pressure and flow rate are considered at the new time-step. The four equations are subsequently solved using Newton-Raphson's method for nonlinear equations.

2.2.3.2 Explicit Lax-Wendroff

The second type of implementation is the explicit version of the upper presented equations. Again the outflow boundary condition will be expressed both at half of the time step and at full time step. The resistance condition at full time step is:

$$p^n = q^{n+1} \times R \quad (2.79)$$

and for the Windkessel boundary condition:

$$\frac{p^n - p^{n-1}}{\Delta t} = R_1 \frac{q^{n+1} - q^n}{\Delta t} - \frac{p^n}{R_2 C_T} + \frac{q^n(R_1 + R_2)}{R_2 C_T} \quad (2.80)$$

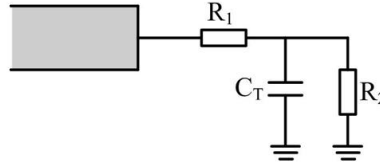


Fig. 2.21 - Windkessel outflow boundary condition.

The equations are called explicit, because only the flow rate is considered at the current time step, the pressure values are taken over from the previous time steps.

2.2.3.3 Implicit method of characteristics

The third type of implementation considered is the implicit method of characteristics. Along the characteristics, a simplified set of ordinary differential equations can be determined. At the outflow boundary, the discretization only involves the positive characteristic:

$$A_M - A_R + \frac{q_M - q_R}{-q_R / A_R + c_R} = H_R^+ Dt \quad (2.81)$$

where A represents the area, q the flow rate and H the viscous losses and the terms containing the derivatives with respect to the radius; while M and R represent the current grid point and the point on the characteristic at the previous time step respectively. The flow rate in (2.5-10) is substituted with the expression in (2.77) for the resistance outflow condition or with the expression in (2.78) for the Windkessel boundary condition. Since the pressure is only related to the area (see (2.69)), a non-linear equation will be obtained where the area at the outflow point is unknown and is determined using Newton-Raphson's method. The computation is much faster since there is only one unknown value instead of four.

2.2.4 Personalization of the multi-scale model of the coronary hemodynamics

The multi-scale reduced order model is used for performing the generic coronary hemodynamics simulations. To achieve a patient specific model simulation it is necessary, besides the acquisition of patient coronary geometry, to estimate and adapt the parameters of the model. These are measured in the hyperemia state of the patient. The main aspects that have to be personalized are inflow and outflow boundary conditions. There are 2 sequential steps which need to be performed:

- a set of parameters are estimated directly;
- an iterative automated optimisation algorithm to estimate hyperemia flow through the arteries affected by stenosis.

2.2.4.1 Estimation of boundary conditions at rest

Since the region of interest, namely the coronary vessel tree is part of the larger circulation system, the inlet and outlet boundary conditions should be chosen such that they adequately model the proximal and distal phenomenon of the patient's circulation. For the coronary outlets, several models have been proposed [2.27, 2.15] which take into account the effect of the myocardial contraction on the flow. These lumped models are usually composed of a set of resistances and compliances, which represent the microvascular beds. The compliance influences

the transient waveform, while the mean value is affected only by the resistance. Since the key diagnostic indexes (such as FFR and CFR) are based on average quantities over the cardiac cycle, the boundary condition estimation is limited to correctly determining the resistance values at each outlet, which is defined as the ratio of the pressure to the flow through that outlet.

Mean arterial pressure (MAP) is constant in healthy epicardial arteries and can be estimated by systolic, diastolic cuff blood pressures (SBP and DBP), and the heart rate [2.28]:

$$MAP = DBP + \frac{1}{3} + (HR \times 0.0012) \times (SBP - DBP) \quad (2.82)$$

Coronary flow depends on the oxygen demand of the heart and since oxygen extraction in the coronary capillaries is close to maximum levels even at rest state, the increased metabolic need can be satisfied only through an increased flow, hence coronary flow is proportional to the oxygen demand. It is difficult to quantify oxygen demand and consumption in the coronaries through non-invasive measurements. Several methods for estimating oxygen consumption from mechanical variables have been proposed in the past, with heart rate as a primary determinant of oxygen consumption. The second major determinant is pressure (pressure generation costs more oxygen than muscle shortening, i.e. flow). The most widely used index for estimating the myocardial oxygen consumption is the rate-pressure product [2.29], according to which,

$$q_{rest} = 8 \times [7 \times 10^{-4} (HR \times SBP)] - 0.4 \quad [ml/m/100g] \quad (2.83)$$

To determine the absolute value of the resting flow (Q_{rest}), resting perfusion is multiplied with the total myocardial mass. In normal hearts, it is generally assumed that the left ventricle represents two thirds of the total mass [2.30]:

$$Q_{rest} = q_{rest} \times 1.5 \times M_{LV} \quad (2.84)$$

Hence total coronary resistance can be computed as:

$$R_{cor} = MAP / Q_{rest} \quad (2.85)$$

M_{LV} is estimated from CTA images using image segmentation.

The next step is to appropriately distribute the total resistance to the various lumped models coupled to the outlet points of the vessel tree. To do this, Murray's law [2.31, 2.32] is used, which states that the energy required for blood flow and the energy needed to maintain the vasculature is assumed minimal and hence:

$$Q_i \sim k \times r_i^3 \quad (2.86)$$

where k is a constant and r is the radius of the vessel.

A value of 3 for the power coefficient in (2.86) has been suggested through the observed invariability of wall shear stress (rate) when flow rate varies substantially [2.44-2.46]. Next, the absolute resting flow, which is the sum of all outlet flows of the coronary vessels, may be written as:

$$Q_{rest} = \sum_{i=1}^n k \times r_i^3 = \sum_{i=1}^n Q_i \quad (2.87)$$

and the flow through a particular outlet is determined by:

$$\frac{Q_i}{Q_{rest}} = \frac{k \times r_i^3}{\prod_{j=1}^n k \times r_j^3} = \frac{r_i^3}{\prod_{j=1}^n r_j^3} \quad \text{and} \quad Q_i = \frac{Q_{rest} \times r_i^3}{\prod_{j=1}^n r_j^3} \quad (2.88)$$

Thus, the terminal resistances can now be determined by:

$$R_i = MAP/Q_i = MAP \times \prod_{j=1}^n r_j^3 / Q_{rest} \times r_i^3 \quad (2.89)$$

2.2.4.2 Estimation of boundary conditions at hyperemia

Intracoronary and intravenously drug-induced hyperemia leads to similar decreases in microvascular resistances [2.18]. The intravenous administration of adenosine leads to a slight increase of heart rate and decrease of blood pressure [2.16]. For a simulation the effect of intracoronary vasodilator can be extended infinitely and it minimally influences the heart rate and blood pressure [2.16]. Adenosine leads to an increase in coronary flow velocity of around 4.5 for normal, healthy subjects (with no coronary artery disease) [2.18].

A good starting point for estimating the normal rest flow rate of the diseased vessel, $(Q_r)_i$, is the generalization of Murray's law:

$$(Q_r)_i = k \times r_i^n \quad (2.90)$$

where r_i is the radius of the healthy part of the vessel. Then using a population average hyperemic-to-rest flow rate ratio [2.18], one can compute the normal hyperemic flow rate of the diseased vessel:

$$(Q_h)_i = rCBF \times (Q_r)_i \quad (2.91)$$

where $rCBF$ is the ratio of coronary blood flow at hyperemia to coronary blood flow at rest. The rest mean arterial pressure (MAP) can be computed from SBP , DBP and HR as described in equation (2.82).

The hyperemic mean arterial pressure, MAP_h , can then be estimated as follows:

$$MAP_h = MAP - \Delta MAP \quad (2.92)$$

where ΔMAP is a population average rest-to-hyperemic pressure difference, after the administration of intracoronary adenosine: 5-10mmHg [2.18]. Supposing an average venous pressure, P_v , of 5mmHg, it can be determined the total hyperemic microvascular resistance:

$$(R_{t-h})_i = \frac{MAP_h - P_v}{(Q_h)_i} \quad (2.93)$$

Since the vessel contains a stenosis, which introduces an additional resistance, the actual flow rate is smaller than the normal flow rate $(Q_h)_i$. To obtain the flow rate in the diseased state, a tuning procedure can be applied, which solves the following nonlinear equation:

$$\mathbf{f}((Q_h)_i) = (MAP_h)_{comp} - (MAP_h)_{ref} = 0 \quad (2.94)$$

where $(Q_h)_i$ is the tuned parameter, $(MAP_h)_{comp}$ refers to the computed mean arterial pressure of the coronary vessel and $(MAP_h)_{ref}$ refers to the reference value, computed with (2.92). As a result, the average flow rate applied at the inlet can be tuned until the desired average pressure is obtained.

Assuming a certain value for n (between 2.33 and 3.0), the only unknown for this method is the constant k in (1), which can be determined for example by minimizing the error between measured and computed FFR values once a large set of patient data is available.

For very large patient data sets, it is possible to estimate different values of k for different categories of vessels, and/or apply some data mining algorithms based on several characteristics (geometry, SBP, DBP, HR, etc.), for a hybrid data mining – CFD approach.

Since blood pressure decreases slightly during hyperemia, a 4.5-fold increase in flow does not mean a 4.5-fold decrease in coronary resistance. A total coronary resistance index can be computed (TCRI), which is equal to:

$$TCRI = \frac{MAP_{hyper}}{Q_{hyper}} \bigg/ \frac{MAP_{rest}}{Q_{rest}} = \frac{(R_{cor})_{hyper}}{(R_{cor})_{rest}} \quad (2.95)$$

A mean value of $TCRI = 0.22$ has been obtained during various studies. It increases from 0.22, for HR less than 75bpm, to 0.26, for a heart rate of 100bpm, and to 0.28 for a heart rate of 120bpm [2.33, 2.34]. Hence, the following relationship can be derived to obtain a HR corrected TCRI:

$$TCRI_{corr} = \begin{cases} 0.0016 \times HR + 0.1 & \text{for } HR \leq 100 \text{ bpm} \\ 0.001 \times HR + 0.16 & \text{for } HR > 100 \text{ bpm} \end{cases} \quad (2.96)$$

Finally, hyperemic microvascular resistances are computed:

$$(R_i)_{hyper} = (R_i)_{rest} \times TCRI \quad (2.97)$$

where $(R_i)_{rest}$ is the value from (2.89).

2.2.4.3 Autoregulation

The four different possible states for a coronary vessel are depicted in the figure below. The assumption in (2.90) is based on state (a) – healthy rest, but it is equally valid for state (b) – stenosed rest, since during rest state autoregulation maintains the flow rate at the same level as with a healthy vessel (unless the stenosis is something like 90%, case in which the autoregulation limit is reached). As a result, proximal to the stenosis, the pressure is the same, the flow rate is the same, and Murray's law can be applied without any restriction. Indeed the terminal resistance decreases because of the autoregulation aspect, but this does not influence the assumption in (2.90).

Next, state (c) – healthy hyperemia is used to determine the terminal hyperemic resistance, which is the same for both stenosed and healthy vessel.

Finally, having determined the hyperemic resistance, the tuning algorithm is employed to determine the flow rate and the distal pressure for state (d) – stenosed hyperemia.

Autoregulation practically reduces the microvascular resistance so as to obtain the same amount of flow through the stenosed vessel, hence even though distal pressure is smaller, flow at

rest will be the same. Of course flow will then not increase at hyperemia as it should, but the tuning algorithm accounts for this aspect.

Even for patients with very high grade stenoses, for which the autoregulation limit is reached, there will be a remodeling process so that the vessel size adapts to the amount of flow through it. Usually these patients though have rest angina and should for the moment be excluded from the study.

One quantity which is crucial though for this method is the healthy radius of the vessel, which has to be reliably estimated. If there is diffuse disease, it's difficult to estimate this value.

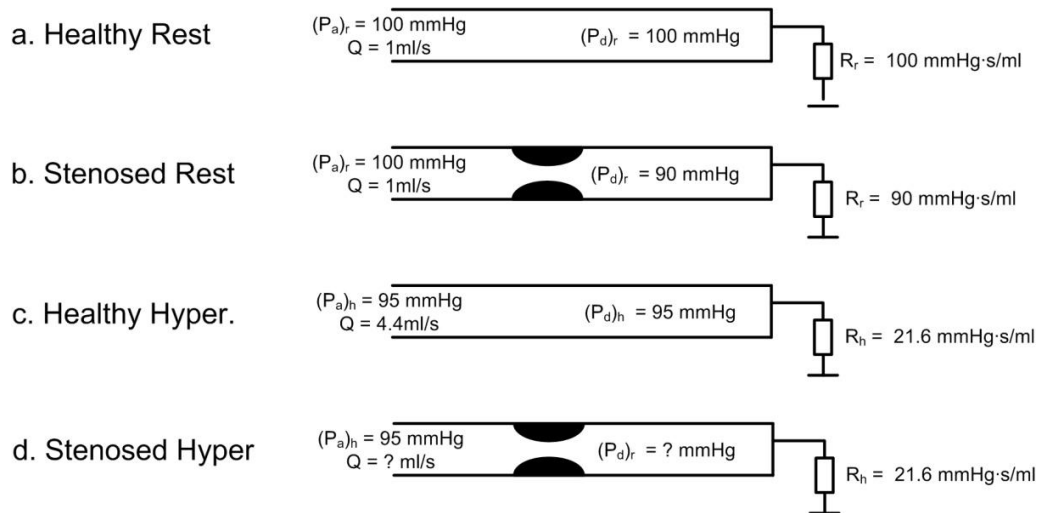


Fig. 2.22 – Coronary vessels states

2.2.5 Feedback control system

In order to accurately evaluate coronary diagnostic indexes, the goal of a CFD simulation is to obtain the same average pressure and flow rates inside the coronary arteries as the ones that would be obtained if the patient were in the rest/drug-induced intracoronary hyperemia [2.47]. Since the proposed method is based on parameters acquired during the rest state, it is important to first set-up the simulation for the rest state and then make the transition to the hyperemia.

The coronary resistances are determined as described by (2.82) to (2.89). As a result, if the simulated MAP matches the value determined through (2.82), the coronary flow will automatically match the estimated value. During intracoronary drug-induced hyperemia, MAP drops slightly due to the decrease in coronary resistances. To capture this aspect, the coronary tree is coupled to the aorta. This coupling also enables the use of a simplified heart model (varying elastance model) in order to provide the inlet boundary condition. If only the coronaries were modeled, then either time-varying flow or pressure would be needed at inflow, none of which is available non-invasively.

Coronary flow at rest represents around 4-5% of the total cardiac output [2.16]. Although the focus lies on the coronary circulation, the systemic resistances (coupled at the outlet of the aorta and of the other proximal vessels) are adapted so that the total coronary flow is around 4.5% of the cardiac output. Thus, the second reference variable is the coronary flow as a percent of the cardiac output. An accurate estimate for it during rest is important to obtain an accurate decrease in aortic pressure when performing simulation at hyperemia.

Figure 2.23 displays the feedback control system that is used for the rest state estimation. From a sensitivity analysis, there are various ways to change the cardiac output of the heart model, namely the time of maximum elastance, maximum contractility, dead volume, initial LV volume, systemic resistance or left atrial pressure (heart rate is given and cannot be changed).

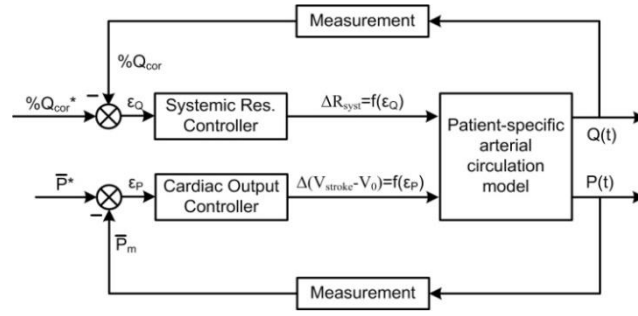


Fig. 2.23 - Multivariable feedback control system

The highest sensitivity is due to the difference between initial LV volume and dead volume. It was designed a PI controller for the systemic resistance, and a PID controller for the cardiac output. Note that the goal of the proposed method is to reach a steady-state which matches the patient-specific steady-state correctly, and not necessarily model the transient aspects of the control mechanism.

Once the simulation has converged and the values for the systemic resistances and for the difference between initial LV volume and dead volume have been determined, the patient-specific model is taken out of the control loop and the rest outflow coronary resistances are substituted by the hyperemic resistances determined as described in (2.97). Thus the MAP is allowed to drop slightly and the percentage represented by the coronary flow out of the total flow becomes much higher since coronary flow increases several times. The simulation is run again until convergence.

2.2.7 Alternative method in modeling arterial hemodynamics

The Lattice Boltzmann Method (LBM) has been introduced in the 80's, and has developed into an alternative powerful numerical solver for the Navier-Stokes (NS) equations for modeling fluid flow. Specifically, LBM has been used consistently in the last years in several blood flow applications (e.g. coronaries [2.37], aneurysms [2.38], abdominal aorta [2.39]). The LBM is a mesoscopic particle based method, which has its origin in the Lattice Gas Automata. It uses a simplified kinetic model of the essential physics of microscopic processes, such that the macroscopic properties of the system are governed by a certain set of equations. The equation of LBM is hyperbolic, and can be solved explicitly and efficiently on parallel computers [2.40].

In the presented context, the single relaxation time version of the equation is considered, based on the Bhatnagar-Gross-Krook (BGK) approximation, which assumes that the macroscopic quantities of the fluid are not influenced by most of the molecular collisions [2.48]:

$$\frac{\mathbb{D}f_i^c(x,t)}{\mathbb{D}t} + c_i \tilde{\mathbb{N}}f(x,t) = \frac{1}{t} (f_i^{eq}(x,t) - f_i(x,t)) \quad (2.98)$$

where f_i represents the probability distribution function along an axis c_i , t is a relaxation factor related to the fluid viscosity, x represents the position and t is the time. The discretization in space and time is performed with finite difference formulas. This is usually done in two steps:

$$f_i(x, t + Dt) = f_i(x, t) + \frac{Dt}{t} (f_i^{eq}(x, t) - f_i(x, t)) \quad (2.99)$$

and

$$f_i(x + c_i Dt, t + Dt) = f_i(x, t + Dt) \quad (2.100)$$

The equation (2.100) is known as the collision step, while (2.101) represents the streaming step. f_i^{eq} is called the equilibrium distribution and is given by the following formula:

$$f_i^{eq} = w_i r(x, t) \left[1 + \frac{c_k u}{c_s} + \frac{1}{2} \frac{c_k u^2}{c_s^2} - \frac{1}{2} \frac{u^2}{c_s^2} \frac{\partial c_k}{\partial x} \right] \quad (2.101)$$

where w_i is a weighting scalar, c_s is the lattice speed of sound, c_k is the direction vector, and \mathbf{u} is the fluid velocity. $r(\mathbf{x}, t)$ is a scalar field, commonly called density, which is related to the macroscopic fluid pressure as follows:

$$p(x, t) = \frac{r(x, t)}{3} \quad (2.102)$$

Once all f_i have been computed, the macroscopic quantities (velocity and density) can be determined:

$$\mathbf{u}(x, t) = \frac{1}{r(x, t)} \sum_{i=0}^n c_i f_i(x, t) \quad (2.103)$$

$$r(x, t) = \sum_{i=0}^n f_i(x, t) \quad (2.104)$$

The computational domain is similar to a regular grid used for finite difference algorithms [2.40]. The presented modeling focuses on 3D flow domains: the D3Q15 lattice structure is used as displayed in figure 2.24 for a single grid node. The weighting factors are: $w_i = 16/72$ for $i = 0$, $w_i = 8/72$ for $i = 1 \dots 6$, and $w_i = 1/72$ for $i = 7 \dots 14$.

The boundary conditions (inlet, outlet and wall) are crucial for any fluid flow computation. For the LBM, the macroscopic quantities (flow rate/pressure) can not be directly imposed at inlet and outlet. Instead, the known values of the macroscopic quantities are used for computing the unknown distribution functions near the boundary. For the inlet and outlet of the domain Zou-He [2.41] boundary conditions with known velocity were used. For the outlet homogeneous Neumann boundary condition was used. The arterial geometry has complex boundaries in patient-specific blood flow computations, and hence, for improving the accuracy of the results, advanced bounce-back boundary conditions based on interpolations were considered [2.42, 2.43].

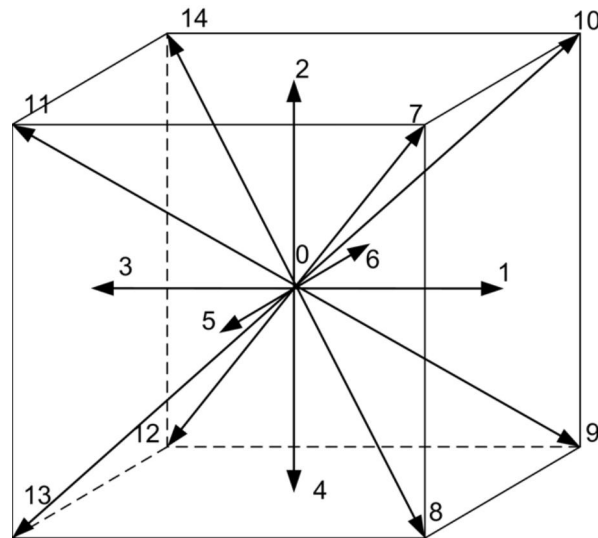


Fig. 2.24 - The D3Q15 lattice structure, first number in the notation is the space dimension, while the second one is the lattice links number

The solid walls are defined as an isosurface of a scalar field, commonly known as the level-set function.

2.2.8 Results - Model simulations and analysis

The generic flow presented in figure 2.25 has been employed in collecting the needed information for performing simulations and validation of the models used in diagnosing stenoses.

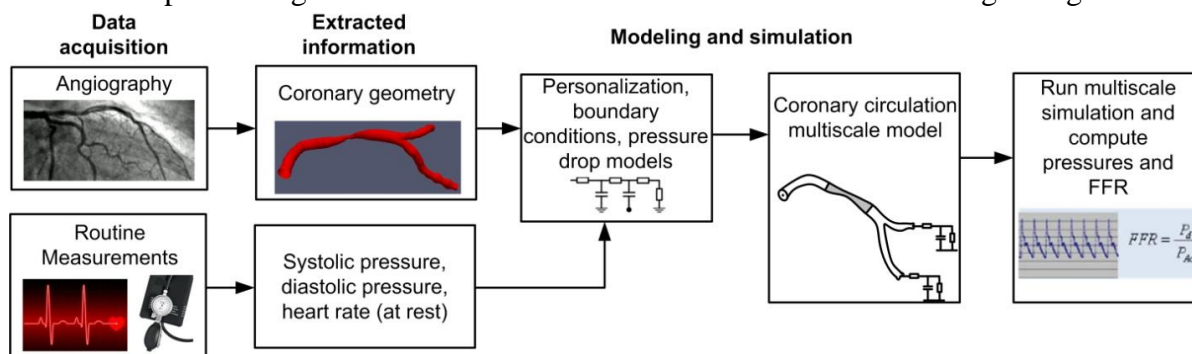


Fig. 2.25 – Generic flow from patient to personalised patient model simulation

During this flow, systolic and diastolic pressure values are acquired using cuff-based measurements and the ejection fraction and end diastolic volumes are estimated from the echocardiography exam performed at rest in a horizontal position. The Computer Tomograph Angiography images of the coronary vessels under study are acquired in 2D projections from 2 different exposure angles. Based on these images that are synchronized using echocardiography, the three-dimensional geometry of the coronary arteries is built and pre-processed, the personalized boundary conditions are estimated. The blood flow through the coronary geometry is simulated based on the above information and FFR is computed (see figure 2.26).

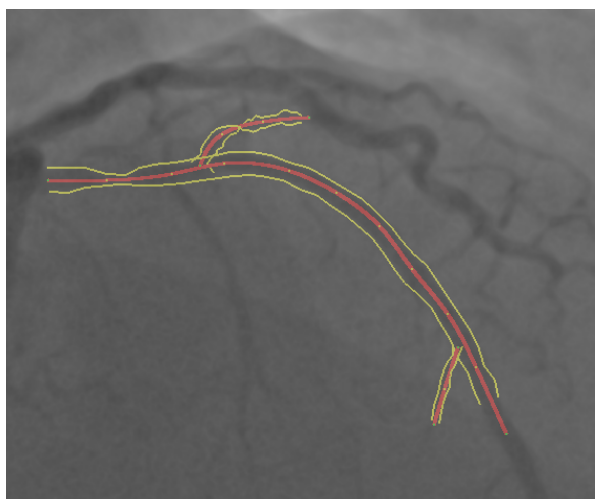


Fig.2.26 – Simultaneously segmentation of several coronary vessels

2.2.8.1 Simulation of outflow boundary condition implementations

Three different implementation approaches of outflow boundary condition have been described that have been tested on three different arteries. Table 2.1 displays the important data of the three vessels: a large one (which resembles the ascending aorta), a medium one (which

resembles the femoral artery) and a small one (which resembles the gastric artery). All of the three vessels have the same length. The table also displays the mean flow rate imposed at the inflow during the simulations. The flow waveform is pulsatile, it is determined through an analytical function [2.25] and has the same shape for all types of vessels (the shape does not exactly match the physiological one, especially for the smaller vessels, and has been chosen only for computational purposes). The resistance and compliance values have been chosen so as to provide reasonable results but they are not guaranteed to be exactly matched by the physiological ones, an aspect which is irrelevant for the scope of this work.

A total of 18 different simulations have been performed for three types of arteries, the two types of outflow boundary conditions and three types of implementations. Table 2.2 displays for the 18 test cases the total execution time, the time spent for the outflow boundary condition and the percentage of the time spent on the outflow computation. It can be seen from the results, some of the simulations have diverged, especially for the implicit Lax-Wendroff implementation of the boundary condition. The data show that the most costly implementation is by far the implicit LW version. The other two implementations are comparable from the execution time point of view, with the explicit LW scheme needing slightly less time than the method of characteristics (12% compared to 13-15%). Also the resistance boundary condition has lower execution times and when the artery becomes smaller, the execution time generally increases.

Table 2.3 displays the standard deviation of the flow rate and pressure values between the three implementation types. Looking first at the resistance boundary condition applied for large arteries, it can be concluded that the explicit LW method is better than the method of characteristics implementation (the standard deviations are smaller when compared to the implicit LW method which is taken as reference). For medium and small arteries, it can be seen that still the results are comparable, leading to higher deviations for the pressure values than for the flow rate values. Also for smaller arteries the errors are greater.

TABLE 2.1 ARTERY DATA

Artery Nr.	Type	R_{top} [cm]	R_{bot} [cm]	Q_{mean} [ml/s]	Length [cm]	R_1 [g/(cm ⁴ s)]	R_2 [g/(cm ⁴ s)]	C_T [(cm ⁴ s ²)/g]
1	Large	1.25	1.14	50	7.0	106	424	9.43e-4
2	Medium	0.43	0.37	5.7	7.0	848	3392	9.43e-6
3	Small	0.16	0.14	0.7	7.0	1696	6784	9.43e-7

TABLE 2.2 EXECUTION TIMES FOR THE THREE ARTERIES AND OUTFLOW BOUNDARY CONDITION IMPLEMENTATION TYPES

Outflow Boundary Condition	Artery	Implicit LW		Explicit LW		Implicit Method of Charact.	
		Total [s]	Outflow [s]	Total [s]	Outflow [s]	Total [s]	Outflow [s]
Resistance	Large	27.17	16.78 (61.7%)	11.59	1.37 (11.8%)	11.92	1.62 (13.6%)
	Medium	Div	Div	12.33	1.48 (12.0%)	12.84	1.68 (13.1%)
	Small	Div	Div	12.46	1.52 (12.2%)	13.12	1.98 (15.1%)
Windkessel	Large	19.76	9.25 (46.8%)	11.24	1.38 (12.3%)	11.83	1.52 (12.8%)
	Medium	Div	Div	11.66	1.42 (12.18)	12.33	1.71 (13.9%)
	Small	Div	Div	Div	Div	11.63	1.78 (15.3%)

TABLE 2.3 STANDARD DEVIATION OF FLOW AND PRESSURE VALUES RELATED TO THE IMPLICIT LAX-WENDROFF IMPLEMENTATION OF THE OUTFLOW BOUNDARY CONDITION

Outflow Boundary Condition	Artery	Comparison	Flow Rate [ml/s]	Pressure [mmHg]
Resistance	Large	Implicit LW vs. Explicit LW	1.39e-2	1.33e-3
	Large	Implicit LW vs. Method of Charact.	3.41e-2	2.13e-3
	Large	Explicit LW vs. Method of Charact.	2.30e-2	1.89e-3
	Medium	Explicit LW vs. Method of Charact.	1.83e-2	8.02e-2
	Small	Explicit LW vs. Method of Charact.	2.73e-2	6.52e-1
Windkessel	Large	Implicit LW vs. Explicit LW	1.44e-2	3.34e-4
	Large	Implicit LW vs. Method of Charact.	1.11	2.09e-2
	Large	Explicit LW vs. Method of Charact.	1.10	2.06e-2

For the Windkessel outflow boundary condition, for large arteries, one can see that the implicit and explicit LW implementations are comparable but there is an important deviation when comparing the two of them to the method of characteristics implementation, especially regarding the flow rate. For medium size arteries, the difference between the explicit LW and the method of characteristics implementations is smaller, allowing one to use the method of characteristics for small arteries with greater confidence (since this is the only non-diverging type of implementation for small arteries with Windkessel boundary conditions).

Three dimensional plots are presented, in order to see how the deviations displayed in table 2.3 are distributed in time and space. Only the plots of the pressure deviations are displayed since the patterns are very similar for the flow rate deviations. Starting with the resistance boundary condition, figure 2.27 shows that the main differences between the implicit and explicit LW outflow implementations lie at the beginning and during peak flow rate periods of time. Nevertheless the deviations are very small. Figure 2.28 shows a comparison between the explicit LW and the method of characteristics implementation for small vessels. The plot displays the greatest deviations at the outflow point. This shows that the method of characteristics leads to some oscillations near the outflow boundary, but these oscillations do not pollute the whole domain.

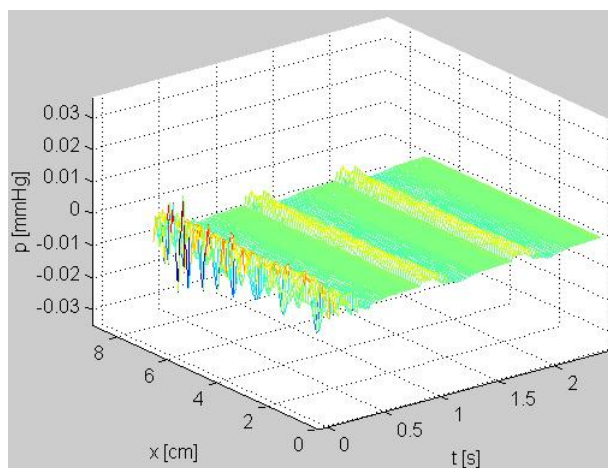


Fig. 2.27 - Pressure deviation between Implicit LW and Explicit LW implementations for large arteries and resistance boundary condition

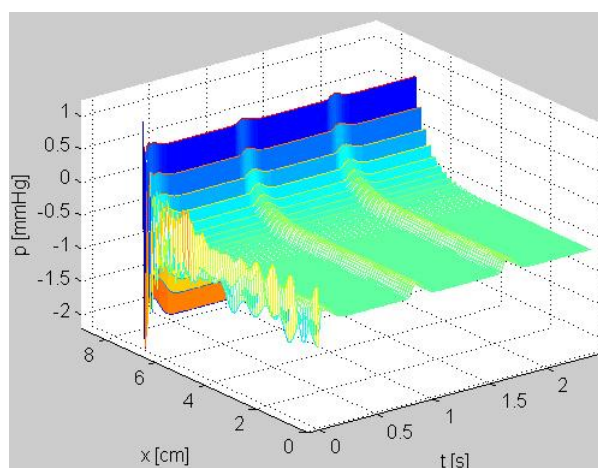


Fig. 2.28 - Pressure deviation between Explicit LW and Method of Characteristics implementations for small arteries and resistance boundary condition

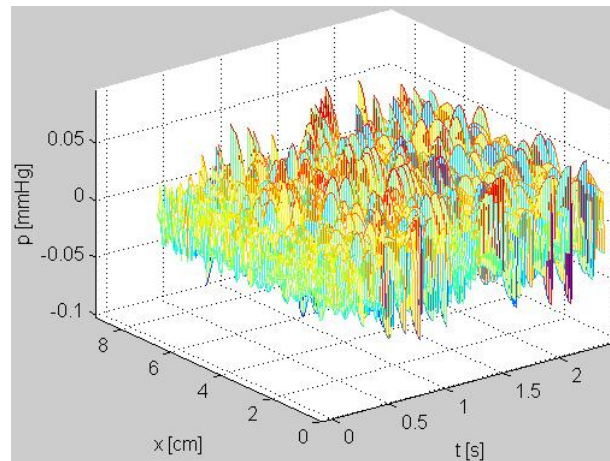


Fig. 2.29 - Pressure deviation between Implicit LW and Method of Characteristics implementations for large arteries and Windkessel boundary condition.

Regarding the Windkessel outflow boundary condition, figure 2.29 compares the implicit LW and the method of characteristics for large vessels and shows that the peak flow rate period has no influence and the deviations are evenly distributed.

A thorough evaluation of three main implementations types for outflow boundary conditions for one-dimensional blood flow models has been performed, namely: implicit Lax-Wendroff, explicit Lax-Wendroff and the implicit method of characteristics. The three versions have been applied for both resistance and Windkessel outflow boundary conditions. In terms of execution time, the implicit LW is the most expensive, while the other two methods are similar. The standard deviations displayed in table 2.3 have shown that all of the three implementations lead to good results, with slight deductions for the method of characteristics. Finally, the analysis of the time-space distribution of the deviations has shown that the highest deviations appear during the first steps and during the peak flow rate period. For small and medium arteries the method of characteristics introduces oscillations near the outflow boundary. These oscillations though do not pollute the rest of the domain.

It can be concluded that for large arteries the best method is the explicit LW implementation, since it is much faster than the implicit LW and more precise than the method of characteristics. For medium arteries, the best choice is again the explicit LW implementation, since the implicit version is divergent and the method of characteristics leads to similar execution but has lower accuracy. These conclusions are valid for both considered outflow boundary conditions. For small vessels and resistance boundary condition the best choice is still the explicit LW method (for the same reasons as for the medium arteries). For small vessels and Windkessel boundary condition, the method of characteristics has to be used, since the other implementation types lead to divergence. The results displayed in table 2.3 and in figures 2.28 and 2.29 show that the errors introduced through the lower accuracy of the method of characteristics are generally negligible and do not influence the results, especially inside the domain.

2.2.8.2 Simulation of stenosis model

The system of equations in the reduced-order arterial model are solved using a finite-difference approach and the two-step Lax-Wendroff method, with a grid-spacing of 0.1 cm and a time step of $2.5e-5\text{ s}$. The average computation time for each cardiac cycle was 54.3 seconds. The patient-specific coronary geometry is displayed in figure 2.30a. The coronary arterial tree has been simulated during rest, at hyperemia and during intense exercise. As specified, the coronary geometry has been coupled to a general model of the systemic circulation comprising 9 segments

(the coronary model contains 15 segments). There are two locations with significant narrowing of the vessel, hence two stenosis segments are included inside the model in the left coronary tree: a mild stenosis (Figure 2.30b) with 48% area reduction, and a mild to moderate stenosis (Figure 2.30c) with a 67% area reduction).

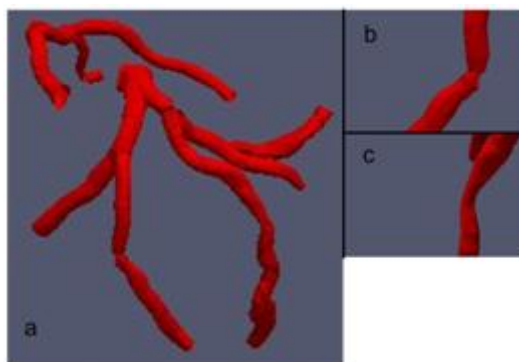


Fig. 2.30 - a) Patient-specific coronary tree, b) stenosis 1 (48% area reduction), c) stenosis 2 (67% area reduction)

Figure 2.31 displays a flow waveform comparison with waveforms reported in literature for the rest state (the waveforms are normalized since they were recorded at different locations and on different models). All three waveforms display the typical low systole and high diastole flow; diastolic decays are similar while the minor differences at systole can be explained through the different coronary models adopted. The simulation parameters and the results are tabulated in Table 2.4. There are several other parameters, which are adopted and which are independent of the state: dead volume of the heart ($V_0=10ml$), stroke volume ($V=120ml$), minimum elastance value ($E_{min}=0.08mmHg/ml$), aortic valve resistance ($R_{LV-art}=10.0 g/cm^4s$), aortic valve inertance ($L_{LV-art}=0.69 g/cm^4$). The results obtained for the rest state are within normal average values, coronary flow represents 4.28% of the total flow (4 - 5% is the average value).

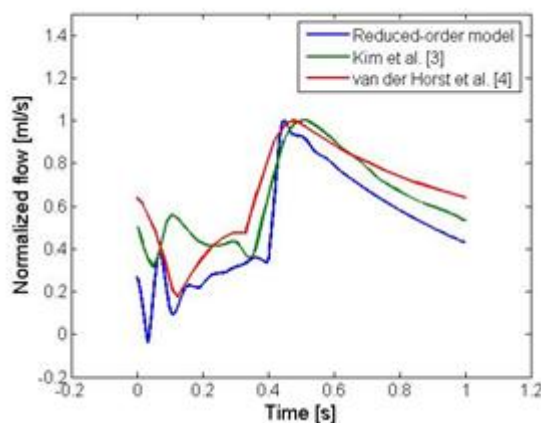


Fig. 2.31 - Coronary flow waveform comparison

TABLE 2.4 SIMULATION PARAMETERS AND RESULTS

State	E_{max} [mmHg/ml]	t_{max} [s]	T [s]	Pa [mmHg]	Cardiac output [ml/min]	Left coronary flow [ml/min]	Right coronary flow [ml/min]
Normal (Rest)	2.1	0.35	1	85.73	3754.6	102.44 (2.73%)	58.32 (1.55%)
Hyperemia	2.1	0.35	1	84.23	3788.9	350.13 (9.24%)	54.91 (1.45%)
Intense exercise	2.3	0.17	0.35	95.98	11395.8	437.23 (3.84%)	203.54 (1.78%)

For simulating drug-induced intracoronary hyperemia, only the lumped parameters of the left coronary tree are adapted. Average pressure was found to be almost identical, as reported by invasive measurements [2.16]. The slight decrease is caused by the decrease of the left coronary resistance. Cardiac output and right coronary flow are almost unchanged. Left coronary flow experiences a three-to-fourfold increase which is again within measured ranges of three-to-five. For the intense exercise state, the left coronary microvascular parameters are identical to the ones used during hyperemia while the other lumped models, namely the right coronary and systemic, are adapted correspondingly. Average aortic pressures increases by around 10mmHg , while the cardiac output triples. The simulation corresponds to a heart rate of around 171bpm . Coronary flow represents 5.62% of total flow. This increase compared to the rest value can be explained as follows: since oxygen extraction in the coronary capillaries is close to maximum levels even at rest state, the increased metabolic need can be satisfied only through an increased flow. On the other side, skeletal muscles can increase oxygen extraction and thus compensate the increased metabolic need not only through a rise in flow rate.

The average pressures distal and proximal to the stenosis at normal and hyperemia state are listed in Table 2.5. In order to investigate the effect of a more pronounced occlusion, the severity of the second stenosis has been virtually increased and the results are displayed in the last two columns. Average proximal pressures (P_a) are close to the aortic average pressure (Table 2.4) since the pressure loss along the large epicardial arteries is very small. Distal average pressures are close to the proximal pressures during rest state, even for the virtual severe stenosis. At hyperemia, the trans-stenotic pressure drop along the two stenoses of the patient-specific model is functionally insignificant. On the other side, the virtual severe stenosis introduces a functionally significant pressure drop. Figures 2.32 a, b show the time-varying pressures for the second stenosis (67% area reduction).

TABLE 2.5 STENOSIS DISTAL AND PROXIMAL AVERAGE PRESSURES [MMHG]

State	Stenoses 1 (48% AR)		Stenoses 2 (67% AR)		Stenoses 2' (84% AR)	
	P_a	P_d	P_a	P_d	P_a	P_d
At Rest	85.39	84.98	85.16	84.23	85.12	78.13
Hyperemia	83.31	81.37	83.09	78.57	82.93	58.16

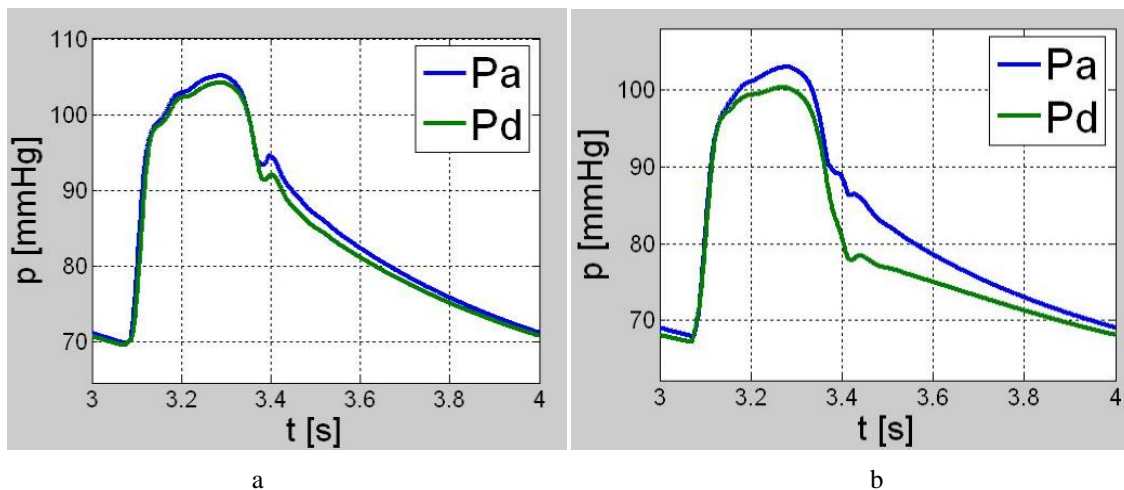


Fig. 2.32 - Proximal and distal time-varying pressures for stenoses 2, a) - at rest, b) – at hyperemia

Based on the reduced-order model for patient-specific coronary circulation which determines the distribution of time-varying and average flow and pressure in the coronary tree extracted from CTA images, three different patient-states have been simulated. In terms of clinical diagnosis and decision-making the most important one is the drug-induced intracoronary

hyperemia, since values of different indices such as FFR (Fractional Flow Reserve) may be estimated. In terms of computation time, the proposed reduced-order model is significantly faster (at least two orders of magnitude) when compared to the full-order models reported in the literature, thereby making it amenable in a clinical setting.

2.3 Synopsis

The chapter focused on the following contributions that were presented in ISI journals or ISI conference papers:

- the modeling of wound and single phase induction motors in the context of using a dynamically emulated high value capacitor on the rotor phases and instead of the 2 capacitor set-up, respectively; analysis of the performance evolution in both simulation and experimental contexts [2.5, 2.6, 2.12, 2.49];
- hemodynamic models for 1D blood flow with focus on boundary conditions applied on stenosis models [2.47, 2.50, 2.51, 2.52, 2.53, 2.57, 2.58-2.60];
- validation through simulation of the designed models [2.54-2.56].

The work has been sustained through public national and European funded projects:

- “Sistem integrat, suport decizional bazat pe fuziunea informatiilor multisenzoriale pentru supravegherea si predictia comportarii barajelor si amenajarilor hidrotehnice – FUZIBAR”
- “Studiul teoretic, experimental si optimizarea sistemului motor monofazat de inductie-condensator controlat”
- “Sisteme de reglare cu structura variabila, fara senzori mecanici (sensorless control), pentru controlul direct al cuplului si fluxului masinilor de c.a. , cu aplicatie in servo-sistemele cu miscare incrementală”
- “Sisteme deschise pentru controlul și instrumentarea proceselor”
- “MD PAEDIGREE – Model-Driven European Paediatric Digital Repository”,
- “HEART – High PERFORMANCE Computing of Personalized Cardio Component Models”

3. High performance computing of system models

The technological advance and globalised interaction between processes/players in practically all domains of activity (e.g. banking, healthcare, industry etc.) imposed development of SW decision making systems based on significant multidimensional datasets that have grown the processing complexity requiring high performance computing (HPC).

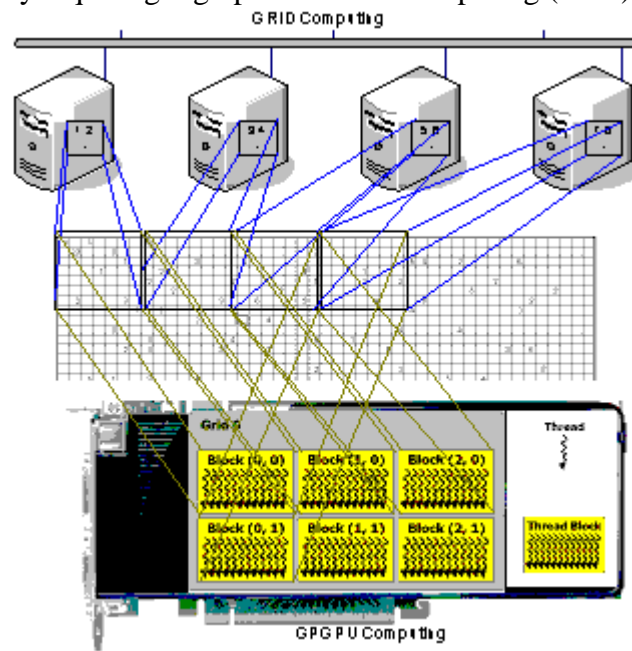


Fig. 3.1 - Grid versus cloud/ multi-CPU versus GPU computing

Similar demands arise from the constantly growing complexity of the models used in simulations & predictions from various domains.

High Performance Computing Platforms (see figure 3.1) are based on distributed or parallel computing in various flavors (e.g. grid computing, cloud). In principle, it can be referred to multi-CPU systems that either have distributed memory (each processor has its own memory) or shared memory (all processors have access to the same memory).

A competitive alternative to CPU based solutions has been raised in the last decade from Graphics Processing Unit (GPU) based implementations [3.5]. A GPU is a stream processor, specifically designed to perform a very large number of floating point operations (Flops) in parallel by using simultaneously multiple computational units. One of the most recent boards, [Tesla P100 whitepaper 2017], provides more than 21 TeraFlops of FloatingPoint performance, being optimized for deep learning applications. In the same time, it could generate cost savings up to 70% in HPC data centers due to its capabilities to replace up to 32 commodity CPU nodes for a variety of applications by using a single GPU-accelerated node. The possibility to perform tasks, which require multi-CPU/grid computing, faster and at lower costs using only one PC that has inside a (multi)GPU based board(s) attracted the attention of the researchers from different areas

N.Parashar[3.1] has shown optimization that can be achieved by parallelizing Gas Kinetic Methos-based flow solvers for performing fast and accurate direct numerical simulations of compressible turbulent flows on simple GPU based workstations. D. Ciresan[3.2] has used a fully parameterizable GPU implementation of a Deep Neural Network (DNN) that does not

require careful design of pre-wired feature extractors, rather learned in a supervised way, for a traffic sign recognition implementation achieving a further boost in recognition performance having in the same time, the system insensitive to variations in contrast and illumination by combining various DNNs trained on differently preprocessed data into a Multi-Column DNN. Le Grand [3.3] presented a new precision model for the acceleration of all-atom classical molecular dynamics simulations on graphics processing units that has a significantly increase in performance on modern GPU hardware without sacrificing numerical accuracy for both generalized Born implicit solvent simulations as well as explicit solvent simulations using the particle mesh Ewald (PME) algorithm for long-range electrostatics.

R. Shams [3.6] showed that image guided therapy systems that are increasingly important in clinical treatment and interventions rely on HPC hardware and highly parallelized software with cost advantages in GPU based equipment. The GPU use has been also involved in cardiac research in various topics such as heart modeling and simulation (especially for reducing the simulation times) or in real time display of dynamic 3D/4D images acquired from MRIs or CTs. Bernabeu [3.4] combined a state-of-the-art cardiac simulation software and grid computing is used to investigate the impact of the block of the HERG current on the ECG waveform using state-of-the-art 3D ventricular models of electrophysiology reducing the execution time of the simulations performed. Sato [3.7] introduces a way to accelerate simulations of the electrical waves propagation in cardiac tissues, comparing GPU and CPU implementations. The single-GPU based approach results to be 20, 40 times faster than the single-CPU configuration. Similar comments stand for clusters GPU respective CPU. R. Yu [3.8] presents an interactive simulation for cardiac intervention, including the detection of the collision between the catheter and the heart wall, based on a boundary representation. A GPU based implementation is used for cardiac modeling, visualization and interactive simulation to enhance the performance of the simulator for real-time and realistic cardiac intervention. Shen [3.9] uses a 3D heart model inside a torso model represented by 344 nodal points with 684 triangular meshes to be used for computer simulation of electrocardiogram (ECG). The ECG computing is done by simulating the excitation propagation through the heart model and applying the Poisson equation to the volume conductor. To speed up the overall simulation process the authors designed the algorithms in a parallelized way. As a result of running these on a GPU, the simulation was 2.74 times faster than in the case of serialized algorithms performed on a CPU.

The potential algorithms acceleration that can be achieved through GPU based implementations using parallelization techniques has been in focus to speed-up mathematical apparatus used for cardio models as well as the simulation of the models themselves. These aspects have been investigation topics in the last years.

3.1 Numerical solution of elliptic equations

Partial differential equations can be generally divided into [3.15]:

- hyperbolic,
- parabolic,
- elliptic.

Elliptic equations are a fundamental building block in fluid dynamics. Some of the most important examples are: steady heat conduction, diffusion process in viscous, turbulent flows, boundary layer flows or chemically reacting flows [3.16].

The solution methods for elliptic equations can be divided into [3.17]:

- direct methods - have the disadvantage that they consume more time and they are susceptible to round-off errors;

- iterative methods - are faster and round-off errors are corrected in subsequent iterations.

Hence iterative methods will be used for the solution of the elliptic equations.

Various iterative finite difference schemes have been developed for elliptic equations, like: Jacobi, Point Gauss-Seidel, Line Gauss-Seidel, Point Successive Over-Relaxation (PSOR), Line Successive Over-Relaxation (LSOR) and Alternating Direction Implicit (ADI) [3.18].

3.1.1 Problem definition

In order to evaluate the speed up, which can be obtained through the use of a GPU, the solution of a steady state heat conduction problem will be considered:

$$\frac{\nabla^2 T}{\Delta x^2} + \frac{\nabla^2 T}{\Delta y^2} = 0 \tag{3.1}$$

A five point finite difference scheme, which leads to the following discretized formula, was considered:

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} = 0 \tag{3.2}$$

In order to obtain a numerical scheme with a fast convergence rate, the point successive over relaxation method (PSOR) has been chosen. The numerical scheme obtained through the usage of this scheme has been explicit since it contains a single unknown value [3.14].

Starting from the above equations, the goal is to asses the speed-up which can be obtained through the use of a GPU.

The heat conduction problem will be solved on a simple, rectangular domain as the one in figure 3.2. The boundary conditions are known, and the boundaries have been chosen so as to coincide with the grid lines.

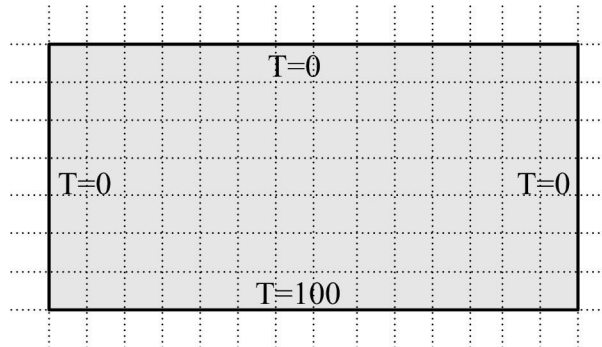


Fig. 3.2 - Rectangular domain of the problem and boundary conditions

3.1.2 Implementation of elliptic equations

The nodes are computed sequentially in a row major order (sometimes column major) on a CPU. Thus, every computation will be using two new values and two old values of the neighboring points. This solution cannot be implemented on a GPU because all of the computations have to be sequential.

The PSOR scheme can be adapted for parallel execution [3.40] by using a so called red-black or checkerboard scheme (figure 3.3). In order to be able to compute the values of the nodes in parallel, they are divided into two groups: a red and a black group [3.10]. The right hand side of fig. 3.3 shows that, when the red nodes are computed, all of the neighbors are black nodes and

vice-versa. This means that the computations can take place in two phases: first all red nodes will be computed in parallel and then all black nodes will be computed in parallel.

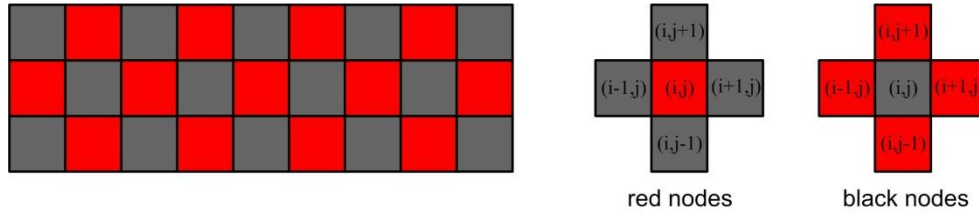


Fig. 3.3 - Red black PSOR memory stencils

The sequential equation will be rewritten for the two phases. The red nodes are those which fulfill the condition $(i+j)\%2==0$, and the black nodes are the nodes which fulfill the condition $(i+j)\%2==1$ (i is the row and j is the column).

The general work flow used to solve the problem is presented in figure 3.4. Initially, the host and device memory buffers are allocated. Then the host memory is initialized. Two buffers need to be allocated, one for the old values and one for the new ones [3.40].

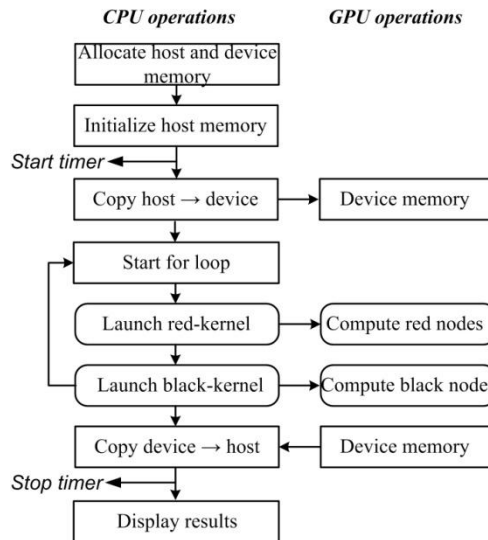


Fig. 3.4 - Work flow of the CPU-GPU version

After the host memory has been initialized, the timer is started in order to determine the exact execution time of the program (through CUDA events [3.11]). The next step is to copy the initial values of the temperature buffer to the device, into its global memory. Then a *for* loop is started, which will perform the computations of the grid nodes. Every iteration of the for loop computes one iteration of the grid values. Inside the loop, first a kernel calculating the red nodes is launched and then a kernel calculating the black nodes is launched. After the for loop has finished its execution, the results are copied back from the global memory to the device memory. Then the timer is stopped and the results are displayed. In order to perform a fair comparison of the execution times for the CPU and the CPU-GPU version, the execution time has to also take into account the memory copies.

The computation of all nodes of the same color can take place in parallel [3.40]. Every thread of the kernel grid computes the value of one node. Fig. 3.5 displays the GPU grid corresponding to the domain shown in figure 3.2. The x direction of the GPU grid coincides with the x direction

of the domain, but there is no specific link between the two directions. The grid displayed in figure 3.5 contains 201x101 nodes (a total of 20301 nodes). If every thread computes one value of the grid, it means that there will be 20301 threads in the kernel grid. A single block can contain up to 512 threads arranged in three dimensions. Since the domain of the problem is two dimensional, two dimensional blocks of 256 threads have been used (16 threads in each dimension). This means that there will be 13 blocks in the x direction and 7 in the y direction. In order to perform a more detailed analysis of the performances of the CPU and CPU-GPU versions, the heat conduction problem has been solved on three different grained grids. The grid displayed in figure 3.5 corresponds to the coarsest grid.

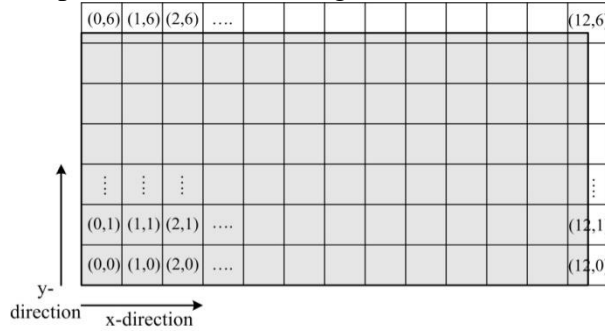


Fig. 3.5 - GPU grid corresponding to the domain of the problem

An important change which has been implemented in the CPU-GPU version is that the memory buffers have been padded on the right hand side of the domain so that most of the global memory reads and writes performed by the half-warps are sequential and aligned [3.12]. The detailed algorithm used to perform the global memory reads and writes requested by the half-warps of a block is described in [3.13].

Figure 3.6 displays the warp – half-warp organization of the threads of a block. All threads of a block are divided into groups of 32 threads called *warps*. All threads of the same warp execute the same instruction at one time (unless there is a branch divergence, case in which the executions of the branches will be serialized). This type of architecture is called SIMT (Single Instruction Multiple Thread), which means that the threads of a warp execute the same instructions but different warps may execute different instructions.

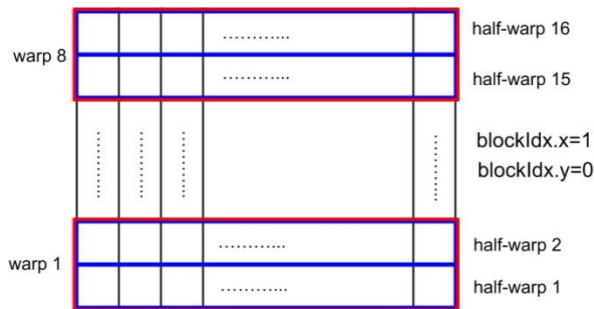


Fig. 3.6 - GPU grid corresponding to the domain of the problem

The threads of the blocks displayed in figure 3.6 will be organized into warps by linearizing their two dimensional structure in a row-major order. Hence consecutive sets of two rows will form the warps. According to the algorithm described in [3.13], the memory transactions (reads/writes) are issued for half-warps. The GPU can only issue transactions of 32, 64 or 128 bytes. If the memory locations requested by the threads of a half-warp are next to each other, then few transactions need to be issued (ideally a single one).

Figure 3.7 displays the global memory access pattern for padded and unpadded memory and

refers to the second half-warp of the block, identified by the built-in variables `blockIdx.x=1` and `blockIdx.y=0` (half-warp 2 in fig. 3.6). If the memory is unpadding, then the first row of the grid occupies the first 804 bytes. Then, on the second row, the first sixteen nodes occupy another 64 bytes. This means that the second half-warp of the above mentioned block has to read the locations 868-932. Fig. 3.7a displays the various segments of different sizes. In order to read these locations two transactions have to be issued: a 32 byte memory read and a 64 byte read.

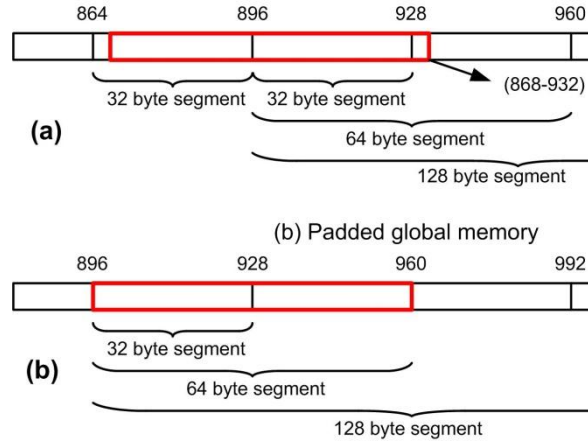


Fig. 3.7 - Unpadding access (a) vs. padded access (b)

If the memory is padded (fig. 3.7b), i.e. the memory is allocated for a number of locations which is a multiple of 16 (for 201 grid points 208 memory locations will be allocated), then the same half-warp will have to read the memory locations 896-960, which corresponds exactly to a 64 byte read. Hence a 32 byte read has been eliminated. Section four shows that this is a very important improvement since this problem is bandwidth limited and not compute limited.

This change leads to an increase of 3,5% in memory usage, but this disadvantage is easily compensated by the time gained through the reduced number of global memory reads and writes (the actual increase will vary from grid size to grid size).

Every kernel (red and black) will perform only a single iteration since the black nodes need to know the new values of the red nodes and there is no possibility of communication or synchronization between the blocks of a GPU grid. This is also the reason why the for loop has been kept on CPU side and not moved into the kernels.

Another important aspect is that there are two buffers for the temperature values at the grid nodes. One holds the old values and the other one the new values. In order to use only two buffers, their status is interchanged after each iteration of the for loop. This action is performed by swapping their positions in the parameter list of the kernel calls at each iteration.

3.1.3 Implementation results for elliptic equations

The tests have been conducted on a NVIDIA GTX260 GPU for assessing the performance improvements of the CPU-GPU version over the CPU version. The `nvcc` compiler reports a usage of 8 registers for both the red and the black kernel. This means that the occupancy of the multiprocessors will be one since four blocks of 256 threads will be able to run simultaneously on a multiprocessor (4 blocks=1024 threads, 8192 registers used of the available 16K)[3.40].

The performance of the two versions have been measured for three different grids: coarse grid (201x101 points, 91 blocks, initial occ. 0.84), medium grid (401x201 points, 338 blocks, initial occ. 3.13) and fine grid (801x401 points, 1326 blocks, initial occ. 12.27).

TABLE 3.1 COMPARISON PADDED VS. UNPADDED MEMORY

Kernel	Total	Execution time [μs]
Red Kernel unpadded	16274	107.02
Red Kernel padded	12456	78.95
Black Kernel unpadded	16131	104.74
Black Kernel padded	12309	77.32

TABLE 3.2 SPEED UP VALUES FOR THE THREE GRIDS

Iterations	Coarse Grid	Medium Grid	Fine Grid
100	0.396	2.03	5.36
300	0.852	3.93	11.21
500	1.117	4.89	11.68
1000	1.454	6.17	12.11
3000	1.827	7.51	12.49
5000	1.927	7.86	12.56
10000	2.009	8.17	12.64

TABLE 3.3 SPEED UP VALUES FOR THE THREE GRIDS

Kernel	Coarse Grid	Medium Grid	Fine Grid
Red kernel	10.48	26.09 (2.49)	78.95 (3.03)
Black kernel	10.26	26.00 (2.53)	77.32 (2.97)

Reference [3.12] states that the kernel grid should contain enough blocks in order to fully occupy the multiprocessors at the kernel launch. Since there are 27 multiprocessors inside the GTX260 GPU, there should be at least 108 blocks on the grid.

Before comparing the CPU and CPU-GPU versions, the improvements, which have been achieved by the usage of padded, instead of unpadded global memory arrays, are presented. Table 3.1 displays the results for the finest grid. In the case of the red kernel, the memory operations have been reduced by 23.46% and the execution time by 26,22%. In the case of the black kernel, the memory operations have been reduced by 23.69% and the execution time by 26,18%.

The comparisons performed hereinafter have used the kernels which access padded global memory. Figure 3.8 displays a comparison of the execution time on the finest grid.

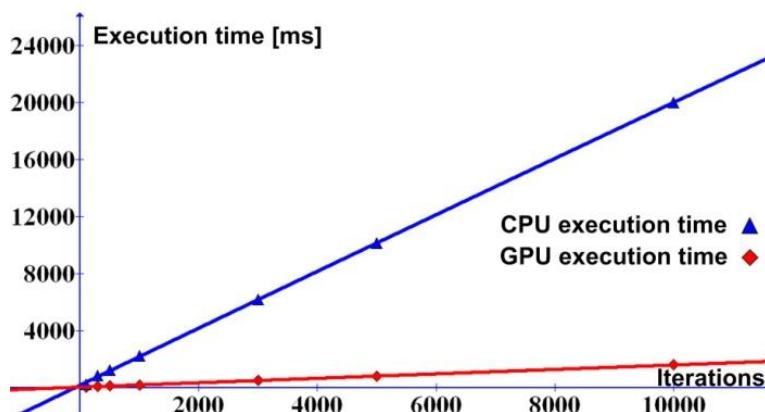


Fig. 3.8 - Performance comparison between CPU and GPU version

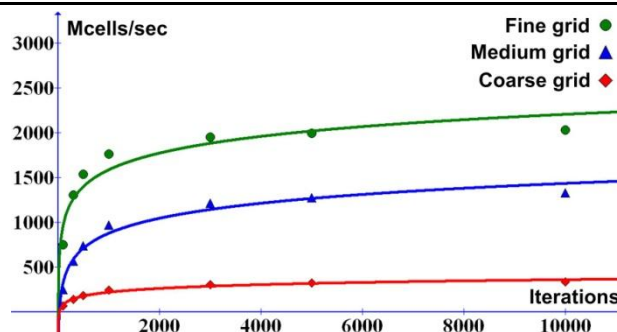


Fig. 3.9 - Performance comparison of the CPU-GPU version for the three grids

A significant improvement of the execution time can be achieved through the GPU version. Table 3.2 displays the exact speed-up values obtained for the three grids.

Table 3.3 displays the average execution times of the two kernels on three different grids. The values in brackets represent the ratio between the execution time on the current grid and the execution time on the next coarser grid (these values should be around four because by doubling the number of points on each dimension, the grid will contain four times more nodes). The values in the table though, show a smaller increase (especially when moving from the coarse grid to the medium grid), which clearly indicates that the GPU does not work at full capacity on coarser grids.

Figure 3.9 displays the number of Mcells computed per second for the three grids and for the different number of iterations. After performing some iterations, the values remain almost constant. The improvement in performance is greater when switching from the coarse grid to the medium grid than from the medium grid to the fine one, which can be mainly explained through the poor initial occupancy on the fine grid.

3.2 Acceleration of hemodynamic models

There has been a lot of interest in exploring high performance computing techniques for speeding up the algorithms in the medical domain, e.g. for three-dimensional blood flow models due to the extremely high computational requirements. Although one-dimensional blood flow models are generally at least two orders of magnitude faster, the requirement of short execution times is still valid. Thus, when blood flow is modeled in patient-specific geometries in a clinical setting, results are required in a timely manner not only to potentially treat the patient faster, but also to perform computations for more patients in a certain amount of time.

As described in chapter 2, it is crucial to match the patient-specific state in a hemodynamic computation. The tuning procedure requires repetitive runs on the same geometry, with different parameter values (e.g. for inlet, outlet or wall boundary conditions (BC)), until the computed and the measured quantities match. This increases the total execution time for a single patient-specific geometry.

In this section, the outcomes of the activities that focus on the GPU based acceleration of the one-dimensional blood flow model are presented based on two algorithms: a novel Parallel Hybrid CPU-GPU algorithm with Compact Copy operations (PHCGCC) and a Parallel GPU Only (PGO) algorithm. These have been applied on a full body arterial model composed of 51 arteries and the speed-up of the two approaches is evaluated compared to both single-threaded and multi-threaded CPU implementations. The computations have been performed using two different second order numerical schemes, with an elastic or viscoelastic wall model, and Windkessel or structured tree boundary conditions as representative examples of physiological non-periodic and respectively periodic outlet boundary conditions.

3.2.1 Methods for one-dimensional blood flow models

The one-dimensional blood flow model is derived from the three-dimensional Navier-Stokes equations based on a series of simplifying assumptions [2.22]. The corresponding equations 2.67-2.74 have been presented in section 2.2

3.2.1.1 Boundary conditions

To perform simulations of the models it is necessary to set the inlet and outlet boundary conditions.

One of the following inlet boundary conditions can be used depending on the availability of in-vivo measurements and the underlying assumptions used in the modeling, researchers typically use one of the following inlet boundary condition: (i) time-varying flow profile, (ii) a lumped model of the heart coupled at the inlet [3.32]), or (iii) a non-reflecting boundary condition like a forward running pressure wave [3.33], [3.34]. A time-varying velocity profile (or flow rate profile) can be consistently determined in a clinical setting, and is often part of the diagnostic workflow (2D/3D Phase-contrast MRI (Magnetic Resonance Imaging), Doppler ultrasound). The parameters of the lumped heart model can be computed based on non-invasively acquired flow rate and pressure values [3.35], while the third type of inlet boundary condition is generally not used in patient-specific computations.

Outlet boundary conditions may be classified as either periodic or non-periodic boundary conditions. Whereas periodic boundary conditions can only be used in steady-state computations (e.g. the patient state does not change from one heart cycle to the next – the same inlet flow rate profile is applied for each heart cycle) and require the flow information from the previous heart cycle, non-periodic boundary conditions do not have these restrictions (e.g. they can be used to model the transition from a rest state to an exercise state for a patient).

Two physiologically motivated boundary conditions are considered [3.41]:

1. The three-element windkessel model (WK), as a non-periodic boundary condition [3.36] as depicted by equation (2.76).

2. The structured tree model (ST) [2.17], as a periodic boundary condition. The structured tree is a binary, asymmetrical vascular tree computed individually for each outlet, composed of a varying number of vessel generations. It is terminated once the radius decreases below a preset minimum radius and its root impedance, $z(t)$, is computed recursively. The root impedance is applied at the outlet of the proximal domain through a convolution integral:

$$(x, t) = \int_{t-T}^t \dot{Q}(x, t) z(x, t - t) dt \quad (3.3)$$

where T is the period. To apply a periodic boundary condition, the flow history is stored and a multiply-sum scan operation is performed at each time-step, leading to considerably higher execution times than for a non-periodic boundary condition.

The Windkessel model could also be applied as a periodic boundary condition, and, as recently described, even the structured tree boundary condition can be applied as a non-periodic boundary condition [3.37].

3.2.1.2 Numerical solution of the one-dimensional blood flow model

When an elastic wall model is used, equations (2.67) - (2.69) represent a hyperbolic system of equations. When a viscoelastic wall model is used, the hyperbolic nature of the equations is

lost due to the additional term in the pressure-area relationship. The approaches for the numerical solution of the one-dimensional equations can be divided into two main categories[3.41]:

1. Methods which do not exploit the original hyperbolic nature of the equations: discontinuous finite element Galerkin method with stabilization terms [3.19]; implicit finite difference/spectral element method where the non-linear terms are solved iteratively at each time-step using the Newton method [3.20, 3.21] etc.;

2. Methods which exploit the hyperbolic nature of the equations in case an elastic wall law is used [2.17], [3.22] or which recover the original hyperbolic nature of the equations in case a viscoelastic wall law is used, by employing an operator-splitting scheme for the momentum equation. This method has been originally proposed in [2.22] for a single vessel and subsequently used in [3.23] and [3.24].

Implicit methods, although solvable with larger time-steps, are slower since they require the solution of a system of equations at each time step and, additionally, require the application of the Newton method for the non-linear terms [3.20]. On the other hand, the methods that exploit the hyperbolic nature of the equations are explicit. They are computationally faster, in spite of the time-step limitation imposed by the CFL condition (named after Courant, Friedrich and Lewy [3.25]).

In addition to the fact that the explicit methods are faster in a sequential implementation, their explicit nature also enables them to be parallelized, and thus be implemented on a cluster [3.26] or on a GPU.

The explicit methods are based either on a first order method (the method of characteristics), or on a second order method (two-step Lax Wendroff [2.17] or expansion in Taylor series [3.22]). Due to their higher accuracy, second-order methods are preferred and have been used for the current study. The method of characteristics is used at the inflow, bifurcation and outflow points.

3.2.1.2.1 Numerical solution of the elastic one-dimensional model

First, equations (2.67) and (2.68) are written in conservation form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{R}}{\partial x} = \mathbf{S} \quad (3.4)$$

$$\mathbf{U} = \begin{pmatrix} A \\ q \\ \delta \end{pmatrix}, \quad \mathbf{R} = \begin{pmatrix} R_1 \\ R_2 \\ \delta \end{pmatrix} = \begin{pmatrix} q \\ q^2/A + B \\ \delta \end{pmatrix}, \quad \mathbf{S} = \begin{pmatrix} S_1 \\ S_2 \\ \delta \end{pmatrix} = \begin{pmatrix} 0 \\ -K_R \frac{q}{A} + \frac{\partial B}{\partial r_0} \frac{dr_0}{dx} \\ \delta \end{pmatrix} \quad (3.5)$$

$$B(r_0(x), p(x,t)) = \frac{1}{r} \int_{p_0}^{p(x,t)} \sigma_{el}^{-1}(p') dp',$$

where \mathbf{U} is the vector of the unknown quantities, \mathbf{R} is the flux term, and \mathbf{S} is the right hand side (RHS).

The Lax-Wendroff (LW) scheme consists of two main steps:

Step 1. Computation of the half step-points: these values are computed between the grid points, hence there are only interior half-step values:

$$\mathbf{U}_j^{n+1/2} = \frac{\mathbf{U}_{j+1/2}^n + \mathbf{U}_{j-1/2}^n}{2} + \frac{Dt}{2} \frac{\partial}{\partial x} \left(\frac{\mathbf{R}_{j+1/2}^n - \mathbf{R}_{j-1/2}^n}{Dx} \right) + \frac{\mathbf{S}_{j+1/2}^n + \mathbf{S}_{j-1/2}^n}{2} \frac{\partial}{\partial t} \quad (3.6)$$

where $j = m \pm 1/2$ and m refers to the grid points;

Step 2. Computation of the full-step-points: this step uses values both from the previous time step and from the half-step points:

$$\mathbf{U}_m^{n+1} = \mathbf{U}_m^n - \frac{Dt}{Dx} (\mathbf{R}_{m+1/2}^{n+1/2} - \mathbf{R}_{m-1/2}^{n+1/2}) + \frac{Dt}{2} (\mathbf{S}_{m+1/2}^{n+1/2} + \mathbf{S}_{m-1/2}^{n+1/2}) (\mathbf{S}_{m+1/2}^{n+1/2} + \mathbf{S}_{m-1/2}^{n+1/2}) \quad (3.7)$$

The expansion in Taylor series (TS) scheme consists of a single step:

$$\frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{Dt} = \mathbf{S}^n - \frac{\partial \mathbf{R}^n}{\partial x} - \frac{Dt}{2} \frac{\partial}{\partial x} \left(\frac{\partial \mathbf{R}^n}{\partial x} \right) - \mathbf{S}_U^n \frac{\partial \mathbf{R}^n}{\partial x} - \mathbf{S}_U^n \mathbf{S}^n \frac{\partial}{\partial t} \quad (3.8)$$

where all the spatial derivatives are discretized using central difference schemes, and:

$$\mathbf{R}_U^n = \frac{\partial \mathbf{R}}{\partial \mathbf{U}}; \quad \mathbf{S}_U^n = \frac{\partial \mathbf{S}}{\partial \mathbf{U}} \quad (3.9)$$

Both numerical schemes require the apriori computation of the flux and RHS terms.

3.2.1.2.2 Numerical solution of the viscoelastic one-dimensional model

An operator splitting scheme is employed for the momentum equation in order to recover the hyperbolic nature of the equations. Thus, equation (2.72) is rewritten as:

$$\frac{\partial q}{\partial t} + \frac{\partial R_2}{\partial x} - \frac{A}{r} \frac{\partial}{\partial x} \left(\frac{\partial q}{\partial x} \right) = S_2 \quad (3.10)$$

The equation is no longer hyperbolic and cannot be cast into conservative form. The splitting scheme assumes that the contribution of the viscoelastic term is small compared to the contribution of the elastic term. The flow rate is considered to be composed of an elastic and a viscoelastic component ($q = q_e + q_v$), and (3.10) is split into two equations:

$$\frac{\partial q_e}{\partial t} + \frac{\partial R_2}{\partial x} = S_2 \quad (3.11)$$

$$\frac{\partial q_v}{\partial t} - \frac{A}{r} \frac{\partial}{\partial x} \left(\frac{\partial q}{\partial x} \right) = 0 \quad (3.12)$$

Consequently, the numerical solution at each step is composed of two sequential sub-steps:

Step 1. The system composed of equations (2.67) and (3.11) is solved, yielding the quantities: $A(x,t)$ and $q_e(x,t)$.

Step 2. Equation (3.12) is solved to obtain $q_v(x,t)$ and thus the total flow rate $q(x,t)$:

$$\frac{q_v^{n+1} - q_v^n}{Dt} - \frac{A^{n+1}}{r} \frac{\partial}{\partial x} \left(\frac{\partial (q_e^{n+1} + q_v^{n+1})}{\partial x} \right) = 0 \quad (3.13)$$

Equation (3.13) is discretized using a central difference scheme, leading to a tridiagonal system of equations, which can be readily solved using the Thomas algorithm in a sequential program. For the viscoelastic component of the flow, homogeneous Dirichlet boundary conditions are imposed at the boundaries of each vessel.

Both the LW and the TS schemes can be used to compute $A(x,t)$ and $q_e(x,t)$, but because the LW method is composed of two steps, it requires the computation of the viscoelastic correction term twice for each time step [3.41]. Since this would significantly increase the total execution time, only the TS scheme has been applied when a viscoelastic wall law was enforced. The various computational setups, for which different parallelization strategies have been adopted, are displayed in table 3.4. Both, a Single-threaded CPU only (SCO) algorithm and a Multi-threaded CPU only (MCO) algorithm, were considered for performance comparison. The MCO algorithm represents a parallel version of the SCO algorithm, implemented using openMP.

TABLE 3.4 COMPUTATIONAL SETUPS FOR WHICH THE SPEED-UP OBTAINED THROUGH GPU-BASED PARALLEL IMPLEMENTATIONS IS INVESTIGATED

Case	Numerical scheme	Wall law	Outlet BC
1	Lax-Wendroff	Elastic	Windkessel
2	Lax-Wendroff	Elastic	Structured tree
3	Taylor series	Elastic	Windkessel
4	Taylor series	Elastic	Structured tree
5	Taylor series	Viscoelastic	Windkessel
6	Taylor series	Viscoelastic	Structured tree

Table 3.5 displays the execution time of the different parts of the SCO algorithm, for cases 5 and 6 from table 3.4 (these two cases were chosen because they contain all computational steps and both types of outlet boundary conditions are considered). Execution times are obtained for equivalent model of the systemic and coronary arterial circulation described in chapter 2 and correspond to the computation for ten heart cycles. When the WK boundary condition is used, approx. 93% of the time is spent on the computation at the interior grid points and on the viscoelastic terms of the flow rate. Since the numerical solution for the interior grid points is explicit, this part can be efficiently parallelized on a manycore architecture (like the one of a GPU device). Though the computation of the viscoelastic terms employs a sequential algorithm, it can also be efficiently parallelized on a manycore architecture, as shown in [3.28]. Furthermore, the computation at the bifurcation and outflow grid points is also parallelizable, but due to the low number of grid points of these types, usually below 100 for an arterial tree, the implementation on a manycore architecture is not efficient. Other operations (initialization activities, writing results to files during the last heart cycle, etc.) account for 2.21% of the total execution time and are not parallelizable. As a result, operations which occupy 93.57% of the total execution time for case 5 are efficiently parallelizable. The difference in terms of execution time between case 5 and case 6 is primarily due to the outlet boundary condition, which requires a multiply-sum scan operation at each time step. Since this operation is efficiently parallelizable on a manycore architecture [3.27], the operations which occupy 95.54% of the total execution time for case 6 are efficiently parallelizable.

TABLE 3.5 EXECUTION TIME AND CORRESPONDING PERCENTAGE OF TOTAL EXECUTION TIME FOR THE COMPUTATIONAL STEPS OF THE NUMERICAL SOLUTION OF THE ONE-DIMENSIONAL BLOOD FLOW MODEL

Computational step	Case 5		Case 6	
	Time [s]	Perc. of total time [%]	Time [s]	Perc. of total time [%]
Interior grid points	357.12	46.12	357.67	30.32
Inflow grid point	0.08	0.01	0.14	0.01
Bifurcation grid points	27.66	3.57	27.61	2.34
Outflow grid points	4.96	0.64	401.72	34.06
Viscoelastic comp.	367.43	47.45	367.55	31.16
Other operations	17.12	2.21	24.87	2.11

For the MCO algorithm, the computation on the interior, bifurcation and outflow points, as well as the computation of the viscoelastic component of the flow rate, can be efficiently parallelized since the number of cores is much smaller for a multicore architecture than for a manycore architecture. This is achieved by associating different arterial segments and bifurcation points to distinct cores.

The results in Table 3.5 show that if the LW scheme were used for a viscoelastic wall law (case in which the viscoelastic correction term would be computed twice at each time step), the total execution would increase by 30-50%, depending on the computational setup.

The implementation of the numerical solution of the one-dimensional blood flow model is efficiently parallelizable on a manycore architecture (like the one of a GPU device), regardless of the computational setup.

3.2.2 Parallelization of the numerical solution

A parallel implementation of the one-dimensional blood flow model is presented in this section, based on a GPU device, programmed through CUDA [3.29].

There have been considered two different implementation approaches to efficiently parallelize the numerical scheme of the interior grid points of each vessel [3.41]:

1. A Parallel Hybrid CPU – GPU (PHCG) algorithm, whereas the unknown quantities at the interior points are computed on the GPU and the inflow/bifurcation/outflow points (called in the following junction points) are computed on the CPU. The advantage is that each device is used for computations for which it is best suited (CPU – sequential, GPU – parallel), but the disadvantage is that memory copies are required at each time step in order to interchange the values near the junction points;

2. A Parallel GPU Only (PGO) algorithm, whereas all grid points are computed on the GPU and the CPU is only used to initialize and to control the execution on the GPU. The advantage is that no memory copies between the CPU and the GPU are required, but the disadvantage is that less parallelizable operations need to be performed on the GPU.

4.2.2.1 Parallel Hybrid CPU-GPU (PHCG) algorithm

The general workflow for the implementation used in case an elastic wall law is displayed in fig. 3.9a.

The CPU is called host, while the GPU is called device. First, the arterial model is initialized (host memory is allocated for each grid point, initial radius, initial cross-sectional area, wall elasticity, derivatives of radius and wall elasticity are computed) and the device memory is allocated and initialized. Next, a while loop is started which advances the entire model in time for a given number of iterations and heart cycles. Inside the while loop, the host and device thread are executed in parallel until a synchronization barrier is reached. During the parallel activities, the CPU computes the new values at the junction points and the device performs the computations for the interior points (equations (3.6), (3.7) for the LW scheme and (3.8) for the TS scheme). Since the device operations are asynchronous, no special approach is required to achieve the task level parallelism between the CPU and the GPU. The computation of the junction points on the CPU is parallelized using openMP for all PHCG implementations. An acronym is displayed in figure 3.9 for each operation to easily match the execution times discussed in the next section with the operations (e.g. OTH stands for *Other operations*, which comprise several activities).

To compute the junction points, the host code requires the values at the grid points next to the junction points, from the previous time step. To compute the values at the grid points next to the

junction points, the device code requires the values at the junction points, also from the previous time step.

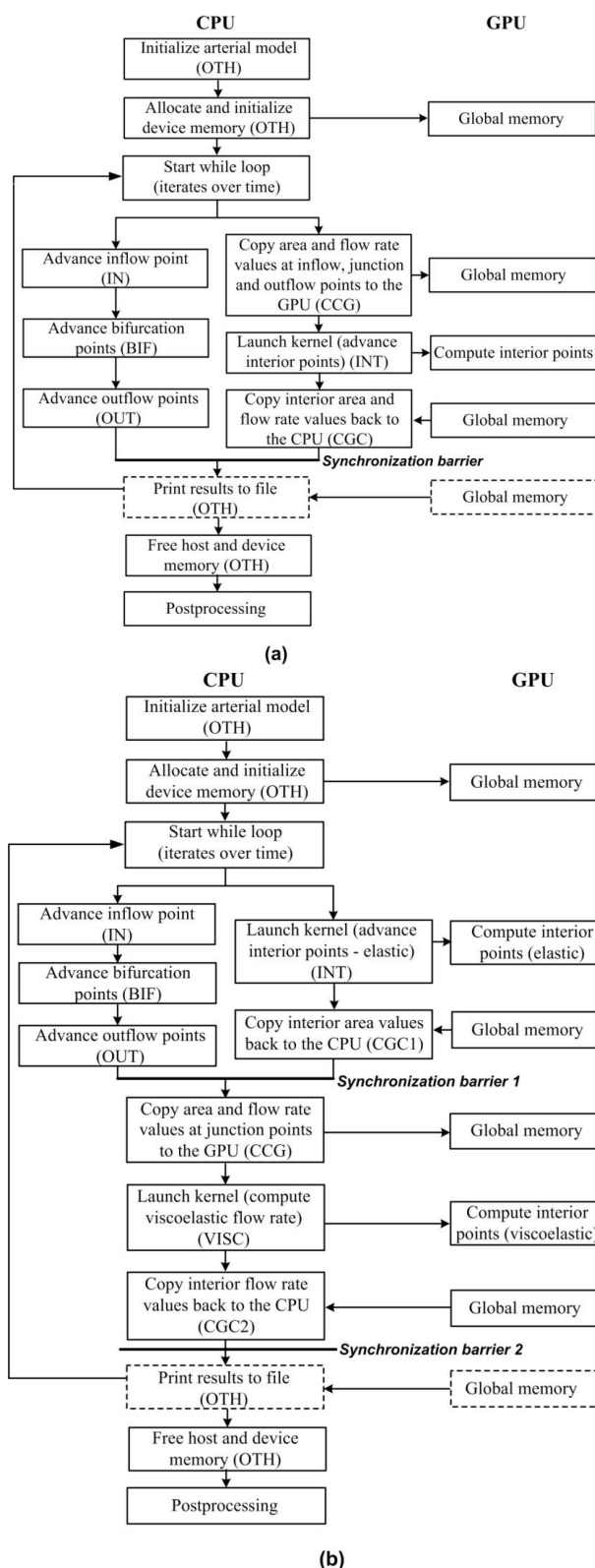


Fig. 3.9 - PHCG workflow in case (a) an elastic wall law or (b) a viscoelastic wall law is used. Junction points are solved on the CPU, while interior points are solved on the GPU. Memory copies are required at each iteration in order to exchange the values near and at the junction points

Hence, to exchange the values at or next to the next junction points, memory copy operations between the device and the host are performed at the beginning and the end of each iteration. A synchronization barrier is introduced after each iteration to ensure that the copy operations have finished. During the last cycle of the computation, after convergence has been reached, the results are saved to files for visualization or post processing. Since the number of iterations for each heart cycle is very high (18000 for a grid space of 0.1 cm), the results are saved only after a certain number of iterations (every 20 to 50 iterations).

To improve the execution time on the GPU, the kernel has been optimized. The specific goal has been to lower the global memory requirement. This approach is necessary to assure efficient kernel performance, even for smaller arterial trees, where parallelism is not pronounced. To reduce global memory operations, memory accesses are coalesced (the global memory accesses performed by threads of the same warp (group of 32 threads) are both sequential and aligned). To obtain aligned memory accesses all global memory arrays have been padded. Furthermore, to avoid redundant accesses performed by different threads, intermediate results are stored in the shared memory of the multiprocessor.

The execution configuration of the kernel which computes the interior grid points is organized as follows: each thread is responsible for one grid point, a block of threads is defined for each arterial segment and both the block and the thread grid are one-dimensional. The numerical solution of a one-dimensional arterial tree, as described in the previous section, is a domain decomposition approach. Hence, data is exchanged between two arterial segments only at the interfaces of the domains. Since for the PHCG algorithm the junction points are solved on the GPU and there is no communication and synchronization requirement between the thread blocks, the association between one thread block and one arterial segment is natural. Furthermore, since parallelism is limited (the number of interior grid points in an arterial tree is usually below 10000 when a grid space of 0.1 cm is used) and the computational intensity is high (the kernel which computes the interior grid points is limited by the instruction throughput – see section four), an approach for which one thread may compute the unknown quantities of several grid points has not been considered. An arterial segment is split into several domains if the hardware resources of a streaming multiprocessor were insufficient to run the corresponding thread block (the solution variables at the interfaces between the domains of the same arterial segment were determined by enforcing continuity of flow rate and total pressure).

An important aspect for the PHCG algorithm is the data transfer between host and device. Although, the amount of data to be transferred is low (only the values at or next to the junction points are exchanged), the total execution time required for these data transfers is high. This is due to the high number of copy operations and the fact that the locations to be copied are scattered throughout the memory arrays. Three different approaches, displayed in figure 3.10, are evaluated for decreasing the total execution time and have led to three different variants of the PHCG algorithm:

1. PHCG Copy Separately (PHCGCS): each location is copied separately, resulting in a total of 8 copy operations for each arterial segment at each iteration. Figure 3.10a displays the locations for which the values are exchanged at each iteration as well as the direction of the copy operations. The arrays displayed in this figure are generic and correspond to either the cross-sectional areas or the flow rates of a single blood vessel;

2. PHCG Copy All (PHCGCA): the entire arrays used for cross-sectional area and for flow rate are transferred at each iteration: 4 copy operations at each iteration (figure 3.10);

3. PHCG Copy Compact (PHCGCC): additional arrays are allocated for the locations which are copied: 4 copy operations at each iteration. Figure 3.10c displays the additional arrays which need to be allocated and the locations which read/write to these arrays. For an arterial network, the values of all dependent variables of one type (cross sectional area or flow rate) are stored in a single array.

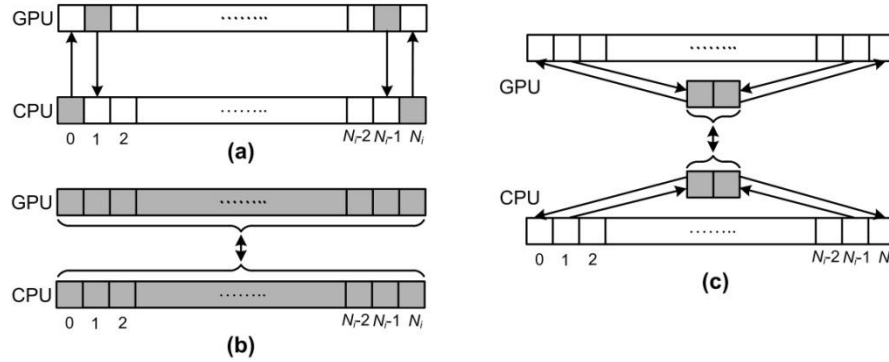


Fig. 3.10 - Host ↔ Device Memory copy variants: (a) separate copy operation for each location, (b) copy entire array, (c) copy compact additional arrays

The first two memory copy strategies were introduced previously[3.38], while the third one is developed during the current research activity and represents a combination of the first two strategies. The PHCGCS variant minimizes the amount of memory to be copied; the PHCGCA minimizes the number of copy operations, while the PHCGCC variant minimizes both aspects by trading kernel performance for data transfer performance (some threads of the kernel populate the additional arrays displayed in figure 3.10).

Figure 3.11 displays the kernel operations and the shared memory arrays used for the two previously described numerical schemes (LW and TS). Since neighboring threads access the same $q/A/R/S$ values, shared memory is used to avoid redundant global memory reads and redundant computations. The operations of the LW scheme (figure 3.11a) are based on equations (3.6) and (3.7) and require only four shared memory arrays (the shared memory is dynamically allocated and the size of the arrays is equal to the number of grid points of the longest vessel). The operations of the TS scheme (figure 3.11b) are based on equation (3.8) and use eleven shared memory arrays. The shared memory requirement is much higher for the TS scheme since: (i) the computations are performed in a single step (the arrays cannot be reused), and (ii) equation (3.8) uses the derivatives of R and S with respect to q and A (the quantities terminated with subscript i are computed by interpolation at locations between the grid points). If a viscoelastic wall law is enforced, the kernel displayed in figure 3.11b is used to compute the cross-sectional area values and the elastic component of the flow rate. The last operation of each kernel is displayed in a dashed rectangle since it is only performed for the PHCGCC variant of the PHCG algorithm. If the PHCGCC algorithm is used, the values corresponding to the last time step are read either from the regular arrays, or from the compact arrays displayed in figure 3.10c during the first operation of each of the two kernels displayed in figure 3.11. Synchronization barriers are used between the individual steps if, during the subsequent step, threads access values computed by other threads (these values are typically stored in the shared memory arrays). The synchronization barriers displayed in figure 3.11 are inserted at GPU thread block level (using `__syncthreads()`), while the synchronization barriers displayed in figure 3.9 are inserted at CPU level (using `cudaDeviceSynchronize()`).

The PHCG workflow introduced previously in [3.38], and reviewed in figure 3.9a cannot be used with a viscoelastic wall law. This is due to the additional steps required by the operator splitting scheme employed for this type of wall law. Consequently, a new workflow have been introduced, as illustrated in figure 3.9b. Two different kernels are used: one for the computation of the cross-sectional area and of the elastic flow rate: equations (2.67) and (3.11); and a second one for the computation of the viscoelastic flow rate according to equation (3.13).

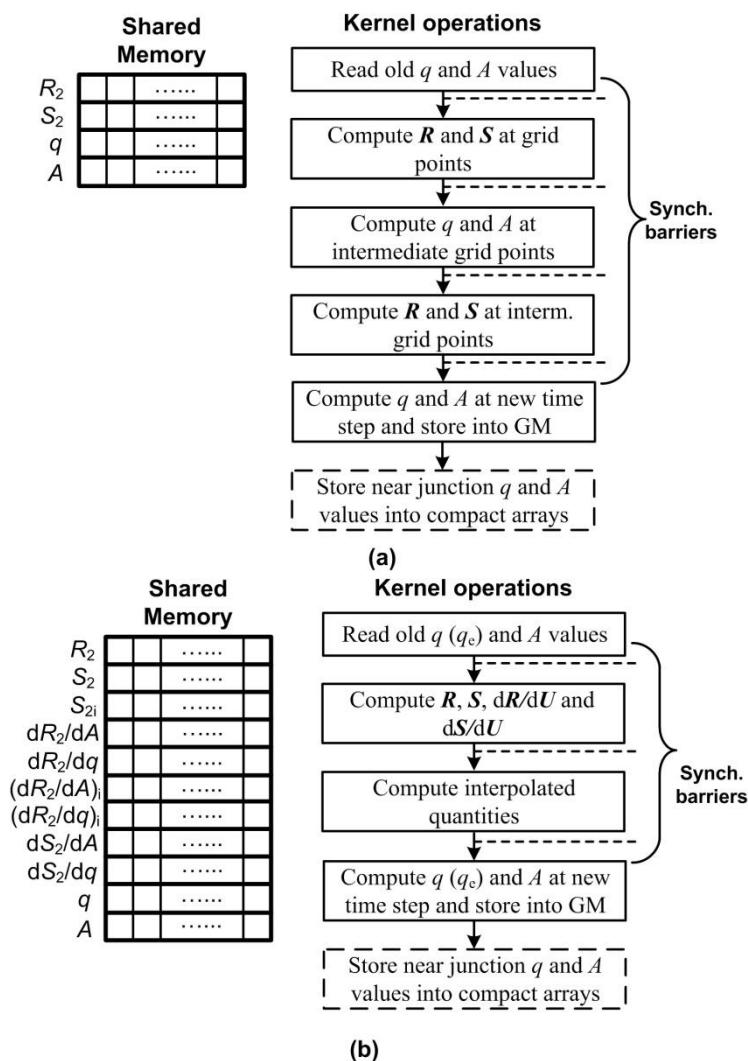


Fig. 3.11 - Kernel operations and shared memory arrays used for the computation of new values at the interior grid point using (a) the LW scheme [3.38], and (b) the TS scheme.

The execution configuration of the first kernel is the same as in the case of an elastic wall law. Host and device instructions are executed in parallel at the beginning of each iteration. After a first synchronization barrier, the values at or next to the junction points are interchanged in order to prepare the computation of the viscoelastic flow rate (in equation (3.13) the new values of the cross-sectional area and of the elastic flow rate at all grid points are required), followed by the computation of the viscoelastic and the total flow rate. An optimized CR (Cyclic Reduction) – PCR (Parallel Cyclic Reduction) algorithm [3.28] is employed to solve the tridiagonal system of equations on the device. Finally, the new flow rate values next to the junction points are copied back to the host and a second synchronization barrier is introduced at the end of the iteration.

An execution configuration with a number of blocks equal to the number of arterial segments is used for the kernel which computes the viscoelastic flow rate. The number of threads of each block is set equal to the smallest power of two value which is higher than the number of grid points in the longest arterial segment. This enables an efficient execution of the CR-PCR algorithm on the GPU.

Figure 3.12 displays the kernel and the shared memory arrays used for the computation of the viscoelastic component of the flow rate and of the total flow rate. First the tridiagonal system is set up (i.e. the coefficients of the three diagonals and of the RHS are computed). The CR-PCR

algorithm is composed of three main steps, two forward reduction (CR and PCR, respectively) and one backward substitution (CR) step. Next, the total flow rate is determined and the new flow rate values are stored in the compact arrays if the PHCGCC algorithm is used.

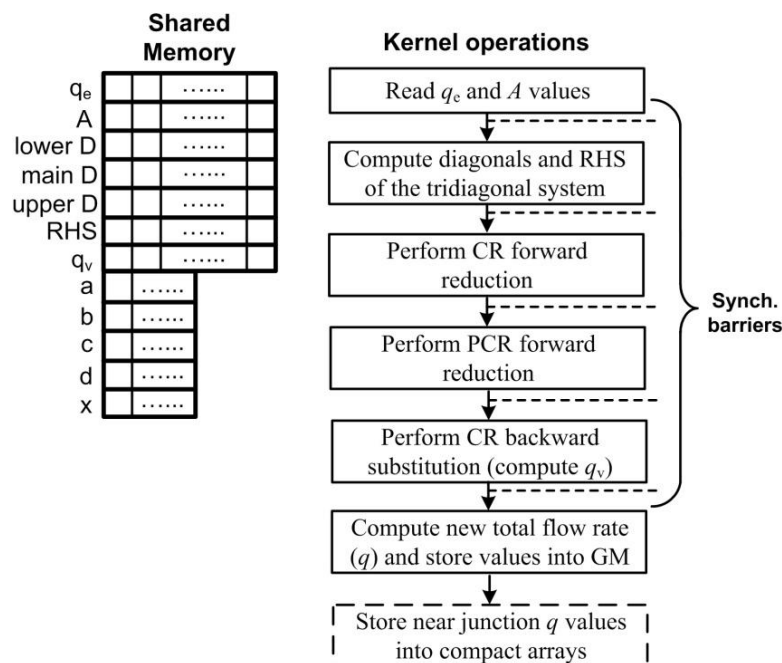


Fig. 3.12 - Kernel operations and shared memory arrays used for the computation of the viscoelastic component of the flow rate and of the total flow rate.

3.2.2.2 Parallel GPU Only (PGO) implementation

The necessity to perform copy operations at each iteration reduces significantly the overall performance of the PHCG algorithm [3.41]. Hence an implementation whereas all grid points are computed on the device, is introduced. This eliminates the memory copies (only the memory copies at the print iterations are required), but also forces the device to perform less parallelizable computations required for the junction points. Another disadvantage of the PGO algorithm, compared to the PHCG algorithm, is that since all operations are performed on the GPU, the task-level parallelism between CPU and GPU is lost. Figure 3.13a displays the workflow for the most complex case, namely when a viscoelastic wall law is used together with the ST boundary condition. A maximum of three kernels are executed at each iteration:

1. Computation of the convolution integral (a multiply-sum scan operation [3.30]): equation (3.3);
2. Computation of the new cross-sectional area and of the elastic flow rate: equations (2.67) and (3.11);
3. Computation of the viscoelastic flow rate: equation (3.13).

The execution configuration of the first kernel is organized as follows: the number of blocks is equal to the number of arterial segments and the number of threads is set to 512. Since the number of time steps per heart cycle (which varies between 8000 and 38000 for different grid space values) is much higher than the number of threads per block, first each thread performs multiple multiply-sum operations and stores the result in a static shared memory array (composed of 1024 double precision elements). Finally, the threads perform a scan operation for the shared memory array and store the result in the global memory. The execution configuration of the other two kernels is the same as the one described in the previous section.

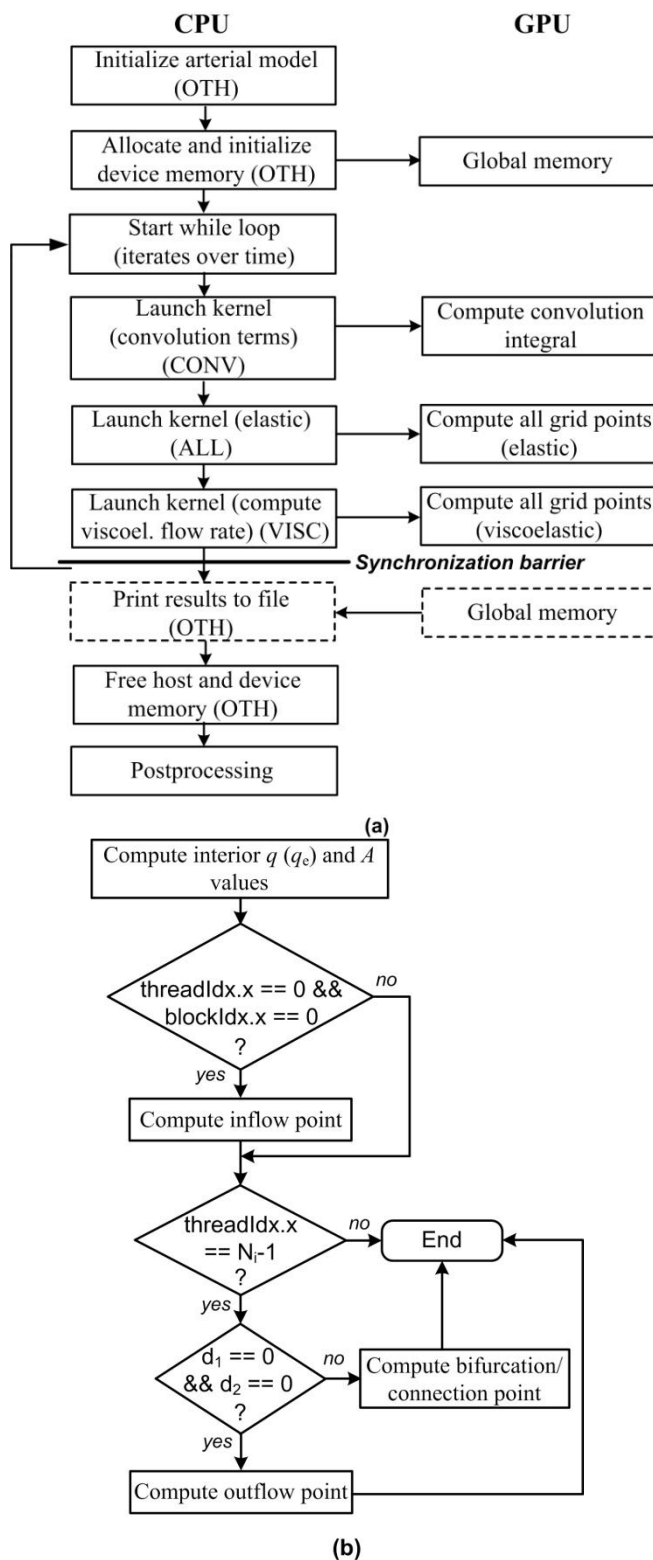


Fig. 3.13 - (a) Generic GPU workflow when a structured tree boundary condition is used and a viscoelastic wall law is enforced. All of the computations are performed inside GPU kernels and the CPU only coordinates the operations; (b) Kernel operations used for the computation of the new values at all grid points.

If the WK boundary condition is used, the first kernel is not called, and if an elastic wall law is used the third kernel is not called. An acronym is displayed in figure 3.13 for each operation to easily match the execution times discussed in the next section with the operations.

Figure 3.13b displays the kernel operations used to compute the new cross-sectional area and flow rate values at all grid points of a vessel segment, with a focus on the junction points. First, the interior points are computed as displayed in figure 3.11 (the individual operations have not been detailed). Next, the first thread of the first block solves the inlet point and the last thread of each block solves the outlet or the bifurcation/connection point (a connection point is also a junction point, which is introduced if an arterial segment is split into several domains). Thus, for the junction points, parallelism is only present at block level and not at thread level.

3.2.3 High performance simulation results

Blood was modeled as an incompressible Newtonian fluid with a density of $\rho = 1.055 \text{ g/cm}^3$ and a dynamic viscosity of $\mu = \nu \cdot \rho = 0.045 \text{ dynes/cm}^2\text{s}$ for all the computations.

To compare the performance of the different algorithms (SCO, MCO, three PHCG variants and PGO) [3.41], the arterial tree detailed in [3.39], and displayed in figure 3.14 was used. It is composed of 51 arteries. A time-varying flow rate profile was imposed at the inlet [2.17], and for the outlets, the WK and the ST boundary conditions were applied (the parameter values displayed in table 3.6 were used). The total resistance and the compliance values were set as in [3.27], and the minimum radius used for the generation of the structured tree was tuned ad-hoc so as to obtain a similar total resistance as for the WK outlet boundary condition (total resistance: $1.37 \cdot 10^3 \text{ dynes} \cdot \text{s/cm}^5$). This aspect allowed us to adequately compare the time-varying flow rate and pressure profiles obtained with the two types of physiologically motivated boundary conditions.

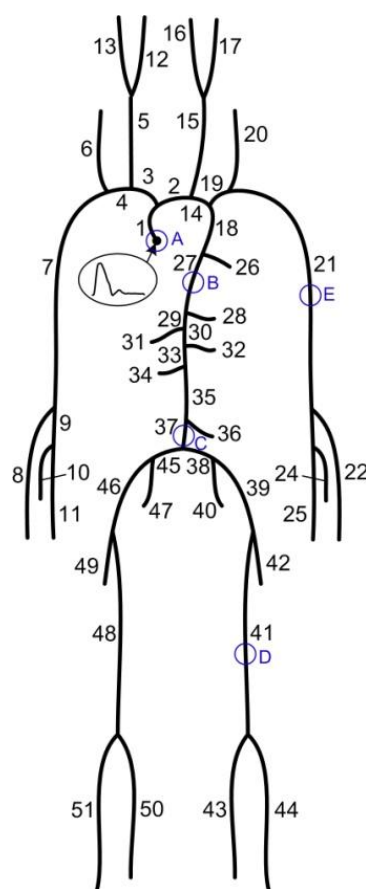


Fig. 3.14 - Representation of the 51 main arteries in the human arterial system; the artery numbers of the outlet segments correspond to those displayed in table 3.6

TABLE 3.6 PARAMETERS OF THE OUTLET VESSELS USED FOR THE WINDKESSEL BOUNDARY CONDITION (R_p , R_d , C) AND FOR THE STRUCTURED TREE BOUNDARY CONDITION (R_{\min}).

Art.Nr.	r_{top} [cm]	r_{bot} [cm]	Length [cm]	R_p [g/(cm ⁴ ·s)]	R_d [g/(cm ⁴ ·s)]	C [10 ⁻⁶ cm ⁴ ·s ² /g]	r_{\min} [cm]
6	0.188	0.183	14.8	8.693	28.007	58.7	0.00235
8	0.174	0.142	23.5	17.165	61.434	25.9	0.00182
10	0.091	0.091	7.9	59.782	238.61	6.6	0.0012
11	0.203	0.183	17.1	8.693	28.007	59.0	0.00235
12	0.177	0.083	17.7	76.989	316.51	4.8	0.0011
13	0.177	0.083	17.7	76.989	315.81	4.8	0.0011
16	0.177	0.083	17.7	76.989	316.51	4.8	0.0011
17	0.177	0.083	17.7	76.989	315.81	4.8	0.0011
20	0.188	0.186	14.8	8.339	28.360	58.7	0.0022
22	0.174	0.142	23.5	17.165	61.434	25.9	0.00182
24	0.091	0.091	7.9	59.782	238.61	6.6	0.0012
25	0.203	0.183	17.1	8.693	28.007	59.0	0.00235
26	0.20	0.15	8.0	14.755	51.844	28.3	0.00193
28	0.30	0.30	1.0	2.796	5.504	268.0	0.0039
31	0.435	0.435	5.9	1.313	11.486	431.0	0.00007
32	0.26	0.26	3.2	3.792	9.007	162.0	0.0033
34	0.26	0.26	3.2	3.792	9.007	162.0	0.0033
36	0.16	0.16	5.0	12.378	42.521	34.0	0.00205
40	0.20	0.20	5.0	6.955	21.144	92.6	0.00255
42	0.255	0.186	12.6	8.339	26.560	62.5	0.0024
43	0.247	0.141	32.1	17.506	62.694	30.0	0.00182
44	0.13	0.13	34.3	21.969	80.330	22.1	0.0017
47	0.20	0.20	5.0	6.955	21.144	92.6	0.00255
49	0.255	0.186	12.6	8.339	26.560	62.5	0.0024
50	0.247	0.141	32.1	17.506	62.694	30.0	0.00182
51	0.13	0.13	34.3	21.969	80.330	22.1	0.0017

An exponential factor equal to 2.7 is used for the ST boundary condition. The constants characterizing the asymmetry of the binary tree were set to 0.908 and 0.578, and the length-to-radius ratio was equal to 50. The elastic properties of the wall were set equal for both the proximal domain and for the structured trees. Together with the minimum radius at which the structured tree is terminated, these parameters determine the compliance of the boundary condition.

The single-threaded CPU algorithm (SCO) was executed on single Intel i7 CPU core with 3.4GHz, the multi-threaded CPU algorithm (MCO) was executed on an eight-core i7 processor, while for the parallel algorithms (PHCG, PGO) a NVIDIA GPU GTX680 (1536 cores on 8 streaming multiprocessors with 192 cores, 48KB of shared memory and 64K registers) was used (the GTX680 is based on the Kepler architecture). All computations were performed with double precision floating-point data structures, since single precision would affect the accuracy of the results, especially at the junction points where the method of characteristics is applied based on the Newton method.

3.2.3.1 Comparison of parallel and sequential computing and with different numerical schemes

Taking the results determined with the SCO algorithm as reference, the L_2 norms of the absolute differences between the reference numerical solution and the numerical solution obtained with the PHCG and PGO algorithms are computed. All L_2 norm results were smaller than 10^{-13} , i.e. close to the precision of the double-type value in computer data structures (both

numerical schemes, LW and TS, were used, but differences were only computed between results obtained with the same numerical scheme).

When the L_2 norm of the absolute differences between the numerical solution obtained with the LW scheme and the TS scheme were computed using the SCO algorithm, the norm results were in the order of $10^{-6}cm^2$ for the cross-sectional area and of $10^{-5}ml/s$ for the flow rate, showing that both numerical schemes lead to the practically same results [3.41,3.42].

3.2.3.2 Comparison of the memory copy strategies for the PHCG algorithm

The three memory copy strategies for the PHCG algorithm are options that have been evaluated in terms of performance. Table 3.7 displays the execution times of the GPU operations, corresponding to the computation of one heart cycle with an elastic wall, the LW scheme and WK outlet boundary conditions (this is the computational setup considered in [3.31]). For the PHCGCS algorithm, the kernel execution occupies only 2.7% of the total execution time on the GPU, making the application heavily PCI Express Bus limited. Although the amount of data to be transferred is higher, the PHCGCA algorithm represents an improvement, since the number of copy operations is reduced drastically. The best results are obtained with the PHCGCC algorithm, since the amount of data to be transferred is small as in the first case and the number of copy operations is reduced as in the second case. The only drawback is that some of the threads of the kernel need to populate the additional arrays displayed in figure 3.10c. This leads to an increase of 8.6% for the kernel execution time, but the increase is easily compensated by the time gained for the memory copies.

TABLE 3.7 EXECUTION TIMES [S] OF THE GPU OPERATIONS OBTAINED FOR THE COMPUTATION OF ONE HEART CYCLE WITH THE THREE VARIANTS OF THE PHCG ALGORITHM. THE RESULTS CORRESPOND TO A COMPUTATION WITH ELASTIC WALLS, THE LW SCHEME AND THE WK OUTLET BOUNDARY CONDITION. *COPY H→D* REFERS TO A COPY OPERATION BETWEEN THE HOST (CPU) AND THE DEVICE (GPU), WHILE *COPY D→H* REFERS TO A COPY OPERATION IN THE OPPOSITE DIRECTION

Operation	PHCGCS	PHCGCA	PHCGCC
Copy H→D	23.7	3.76	0.85
Kernel	1.86	1.86	2.02
Copy D→H	43.1	5.29	0.89

3.2.3.3 Comparison of the performance obtained with the SCO, MCO, PHCG and PGO algorithms

Table 3.8 summarizes the execution times measured for the six different computational setups displayed in table 3.4. The execution times correspond to ten heart cycles and the highest speed-up values are displayed in bold. The grid space has been set to $0.1cm$ and the time step to $5.55*10^{-5}s$. The values are based on literature data and on the CFL-restriction respectively.

The PHCGCA algorithm cannot be applied for all computational setups presented herein (e.g. the workflow from figure 3.10a cannot be applied for a viscoelastic wall law). The speed-up values in table 3.8 are computed based on the execution time of both the SCO and MCO algorithms.

The speed-up values vary between **5.26x** and **8.55x** compared to the SCO algorithm and between **1.84x** and **4.02x** compared to the MCO algorithm. As anticipated, the PHCGCC algorithm outperforms the PHCGCA algorithm for all cases for which the PHCGCA was applied.

TABLE 3.8 (a) Execution times and speed-ups obtained for the computation of ten heart cycles with the SCO, MCO, and PGO algorithms. The first four cases correspond to an elastic wall with either the Lax-Wendroff (LW) or the Taylor series (TS) scheme and with a Windkessel (WK) or structured tree (ST) boundary condition. The last two cases correspond to a viscoelastic wall law with the TS scheme and with a WK or ST boundary condition.

Case	Num. sch.	Wall law	Outlet BC	SCO [s]	MCO [s]	PGO		
						Time [s]	Speed-up	
							SCO	MCO
1	LW	Elastic	WK	273.4	81.13	101.98	2.68x	0.79x
2	LW	Elastic	ST	673.7	205.38	111.49	6.04x	1.84x
3	TS	Elastic	WK	396.3	119.37	105.10	3.77x	1.14x
4	TS	Elastic	ST	797.2	233.09	116.56	6.84x	2.00x
5	TS	Viscoel.	WK	774.4	384.43	235.14	3.29x	1.63x
6	TS	Viscoel.	ST	1179.6	501.08	241.45	4.89x	2.07x

TABLE 3.8 (B) EXECUTION TIMES AND SPEED-UPS OBTAINED FOR THE COMPUTATION OF TEN HEART CYCLES WITH THE PHCGC ALGORITHMS. THE FIRST FOUR CASES CORRESPOND TO AN ELASTIC WALL WITH EITHER THE LAX-WENDROFF (LW) OR THE TAYLOR SERIES (TS) SCHEME AND WITH A WINDKESSEL (WK) OR STRUCTURED TREE (ST) BOUNDARY CONDITION. THE LAST TWO CASES CORRESPOND TO A VISCOELASTIC WALL LAW WITH THE TS SCHEME AND WITH A WK OR ST BOUNDARY CONDITION

Case	Num.sch.	Wall law	OutletBC	PHCGCA			PHCGCC		
				Time [s]	Speed-up		Time [s]	Speed-up	
					SCO	MCO		SCO	MCO
1	LW	Elastic	WK	68.21	4.01x	1.19x	42.4	6.45x	1.91x
2	LW	Elastic	ST	182.34	3.69x	1.13x	149.92	4.49x	1.37x
3	TS	Elastic	WK	74.81	5.30x	1.59x	46.37	8.55x	2.57x
4	TS	Elastic	ST	187.27	4.26x	1.24x	151.90	5.25x	1.53x
5	TS	Viscol.	WK	-	-	-	95.64	8.10x	4.02x
6	TS	Viscol.	ST	-	-	-	224.38	5.26x	2.23x

For an elastic wall law, in case a WK boundary condition is used, the PHCGCC algorithm performs best, while in case the ST boundary condition is used, the PGO algorithm leads to the highest speed-up. For a viscoelastic wall law, the PHCGCC algorithm performs best, regardless of the type of outlet boundary condition. Execution times are higher with a ST boundary condition because of the time spent for the computation of the convolution integral in equation (3.3).

For an elastic wall law, with the PHCGCC algorithm, the execution times are comparable for the LW and the TS scheme (for both outlet boundary condition types), with slight advantages for the LW scheme. For the SCO and MCO algorithms, the LW scheme is superior to the TS scheme.

A detailed analysis of the results obtained with the best performing algorithms (PHCGCC and PGO) is presented below [3.41]. Figure 3.15 displays the percentage of the execution time occupied by each operation identified in the workflows in figure 3.9, for cases 1, 2, 5 and 6, computed with the PHCGCC algorithm. Regarding the computations with an elastic wall law, as is displayed in figure 3.9a, computations on the host and on the device are performed in parallel. The operations on the device require more time than the host operations if a WK boundary condition is used. Although the copy operations were optimized, they occupy almost half of the total time spent on the GPU. Besides, a considerable time is required for other operations, which include control instructions, data exchange operations between the host arrays and the arrays used for the copy operations, and print operations during the last cycle. If a ST boundary condition is used, the computation of the convolution integral in equation (3.3), performed on the host, occupies most of the execution time.

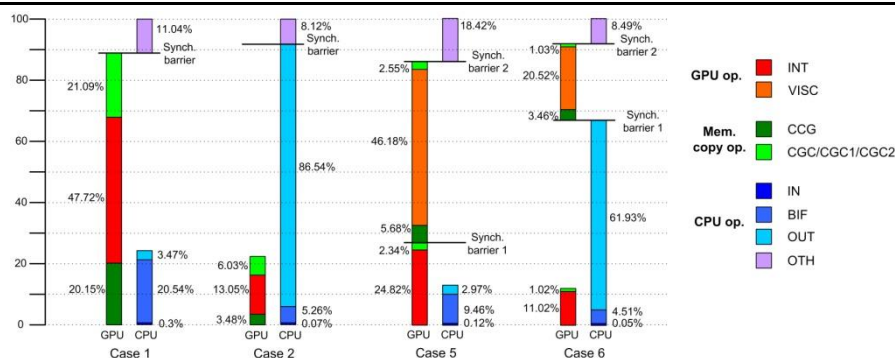


Fig. 3.15 - Detailed percentage values of the execution time occupied by the operations identified in the workflow in Fig. 3.9 for the PHCGCC algorithm, for an elastic wall law (cases 1 and 2) and a viscoelastic wall law (cases 5 and 6). Acronyms are detailed in figure 3.9

This is the primary reason behind the low speedup achieved with the PHCGCC algorithm and the ST boundary condition. In addition, it also explains the similar speed-up values obtained with the PHCGCC and PHCGCA algorithms.

Regarding the computations with a viscoelastic wall law, as is displayed in figure 3.9b, computations on the host and on the device are performed in parallel at the beginning of each iteration, but since the computation of the viscoelastic flow rate requires the values of the elastic flow rate and of the cross-sectional area at the junction points (from the current time step), during the second part of each iteration, only the device performs computations. As for the elastic wall law, in case a ST boundary condition is used, the computation of the convolution integral in equation (3.3), performed on the host, occupies most of the execution time.

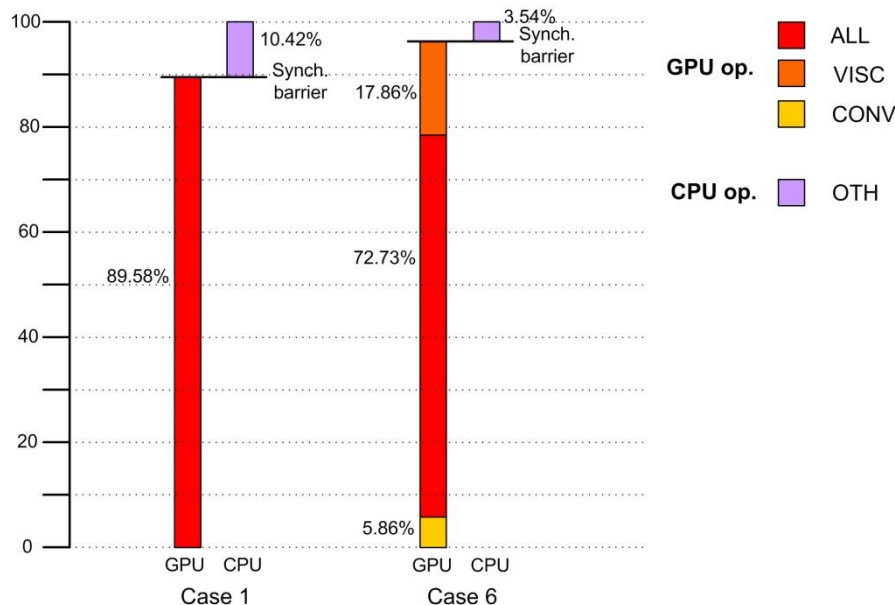


Fig. 3.16 - Detailed percentage values of the execution time occupied by the operations identified in the workflow in Fig. 3.13 for the PGO algorithm, for an elastic wall law (case 1) and a viscoelastic wall law (case 6). Acronyms are detailed in Fig. 3.13.

Figure 3.16 displays the percentage of the execution time occupied by each operation identified in the workflow in figure 3.13, for cases 1 and 6, computed with the PGO algorithm. In the first case a single kernel is used, while for case 6 also the convolution integral and the viscoelastic flow rate correction are computed. The computation of the interior and junction points require more execution time for case 6 than for case 1, since, on the one side the TS

scheme is used instead of the LW scheme, and on the other side additional operations are performed because of the viscoelastic wall law. Compared to case 6 in figure 3.15 (ST boundary condition), the execution time dedicated to the outflow points is reduced significantly since the operations are performed on the device, but because the computation of all grid points requires considerably more time, the total execution time for case 6 is higher with the PGO algorithm than with the PHCGCC algorithm.

Figure 3.17 displays a comparison of the number of heart cycles which can be computed per hour with different algorithms: the SCO and MCO algorithms, the previously introduced PHCGCA algorithm (applied only for non-periodic boundary conditions) and the best performing parallel algorithm for each computational setup as determined in the current study. The four different computational setups have been obtained by combining the different wall laws and outlet boundary conditions and by choosing the best performing numerical scheme (according to the results in table 3.8). The results show that the best performing GPU based algorithms considerably increase the number of heart cycles which can be computed per hour.

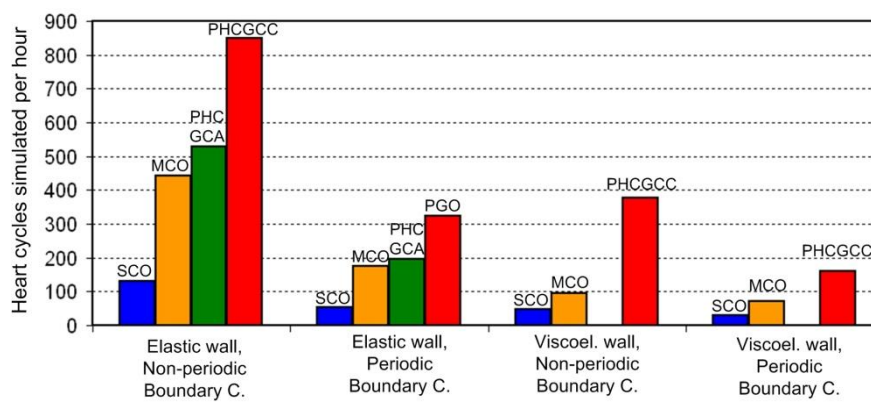


Fig. 3.17 - Heart cycles computed per hour for the SCO algorithm, the MCO algorithm and the best performing parallel algorithm for each computational setup

The effect of the simulation parameters on the speed-up factor can be analyzed based on the figure 3.18 the speed-up values obtained for different grid space values: $\Delta x = 0.25cm$ (5486 degrees of freedom (dofs), 8000 time steps per cycle), $\Delta x = 0.15cm$ (9144 dofs, 12500 time steps per cycle), $\Delta x = 0.1cm$ (13716 dofs, 18000 time steps per cycle), and $\Delta x = 0.05cm$ (27432 dofs, 37000 time steps per cycle).

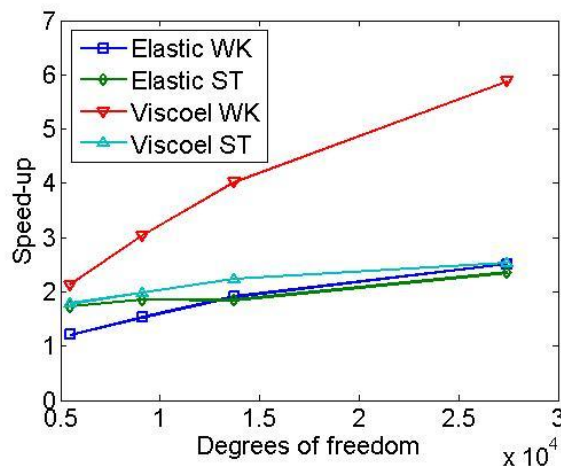


Fig.3.18 - Speed-up values obtained for different grid space configurations for the best performing parallel algorithm compared to the MCO algorithm

The displayed values represent the speed-up obtained by the best performing GPU based algorithms compared to the MCO algorithm. The time-step values are chosen to satisfy the CFL condition for each case, and both types of wall laws and outlet boundary conditions are considered. In each case, the numerical scheme and the parallel algorithm applied for the computation correspond to the best speed up value obtained for a grid space of 0.1 cm .

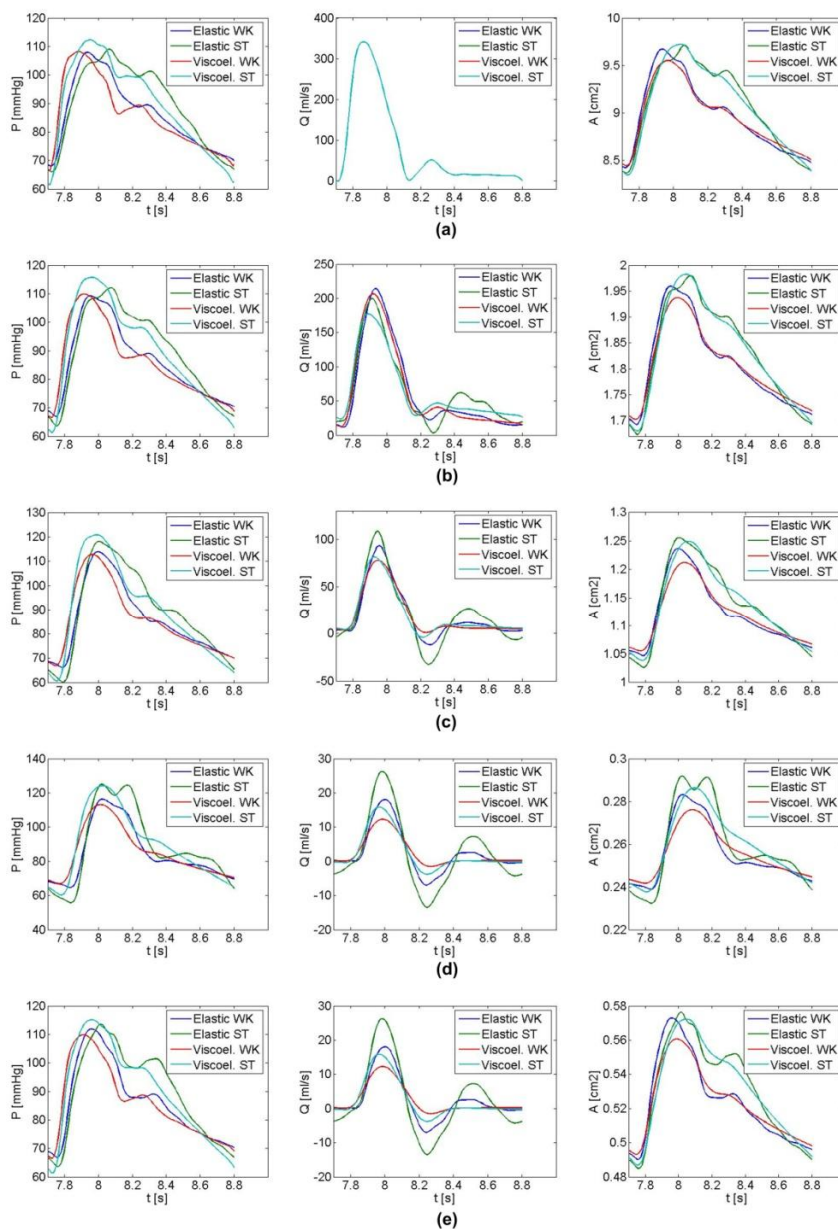


Fig. 3.19 - Time-varying pressure, flow rate and cross-sectional area at (a) aortic root, (b) descending aorta, (c) abdominal aorta, (d) femoral artery, and (e) subclavian artery (corresponding to locations A-E respectively in Fig. 3.14). Four plots are displayed in each figure, which have been obtained with either an elastic or viscoelastic wall and with a WK or ST boundary condition

Figure 3.18 displays an approximately linear increase of the speed-up value, indicating that the computational power of the GPU is not fully exploited for any of the computational configurations with a grid space higher than 0.05 cm . The increase is moderate for three of the four computational setups and more pronounced in case a viscoelastic wall law is used together with a WK boundary condition. This aspect is given by the fact that the implementation of the

viscoelastic wall law is more efficient for the PHCGCC algorithm compared to the MCO algorithm. On the other hand, when a viscoelastic wall law is used together with the ST boundary condition, most of the time is spent for computing the outlet grid points and the difference in execution time for the viscoelastic component becomes less important.

Figure 3.19 displays the time-varying pressure, flow rate and cross-sectional area at the five locations marked with a blue circle in figure 3.14.

Each figure contains four plots, which have been obtained with either an elastic or viscoelastic wall and with a WK or ST boundary condition. Since the total resistance introduced by either of the two types of boundary conditions is similar, the average quantities are approximately equal at all locations inside the arterial tree. Referring first to the computations with elastic walls, the pressure values obtained with the ST boundary condition decrease at a later time inside one heart cycle, indicating that the reflected wave arrives later (an aspect which is more pronounced for the proximal parts of the arterial tree). This can be explained as follows: the ST boundary condition simulates the propagation of the waves down to the arteriolar level where the reflections occur primarily, whereas the WK boundary condition, as a lumped model, is not able to capture the wave propagation phenomena in the distal part of the tree and introduces the reflections at the outlet points of the proximal arteries. As a result of the later arriving pressure waves, also the maximum pressure value is reached at a later moment in time. These aspects also lead to higher oscillations inside the flow rate waveforms which are displayed in the second column of figure 3.19. Finally, for the cross-sectional area, generally, the variation inside one heart cycle is higher with a ST boundary condition. For the elastic wall, the pressure and the cross-sectional area waveforms are in phase and a more pronounced variation of the area values is reflected by a higher pressure pulse. The higher pressure pulse obtained for the structure tree boundary condition indicates a lower total compliance than the one enforced through the WK boundary condition. It should be noticed that the compliance of the proximal part of the tree is identical in both cases and the difference in total compliance is given only by the outlet boundary conditions.

When a viscoelastic wall is used, the main difference is that the high-frequency oscillations in the waveforms are reduced. This can be observed in both the pressure and the flow rate waveforms and the phenomenon is more pronounced at the distal locations. These observations are consistent with results reported in literature [3.20]. The introduction of the viscoelastic wall does not change the overall behavior of the WK and ST boundary conditions, the observations mentioned above being still valid, as would be expected. Another important consequence of the introduction of the viscoelastic wall is the fact that pressure and area are no longer in phase, the peak cross-sectional area value being generally obtained at a later moment in time inside one heart cycle.

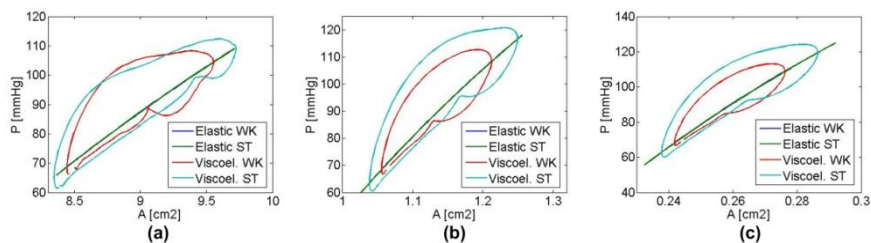


Fig. 3.20 - Pressure-area relationships at (a) aortic root, (b) abdominal aorta, and (c) femoral artery (corresponding to locations A, C and respectively D in Fig. 3.14). Four plots are displayed in each figure, which have been obtained with either an elastic or viscoelastic wall and with a WK or ST boundary condition

Furthermore, figure 3.20 displays the pressure-area relationships at three different locations. A hysteresis loop can be observed when a viscoelastic wall laws is used, as opposed to the linear

variation for an elastic wall law. The area of the hysteresis loop is proportional to the energy dissipation given by the viscoelastic properties of the wall.

3.3 Synopsis

The chapter focused on the following contributions that were presented in ISI journals or ISI conference papers:

- Methodologies to accelerate solving algorithms(e.g., differential equations) [3.46-3.51];
- GPU-based implementations of the one-dimensional blood flow model that considerably improve the execution time compared to previously reported parallel implementations [3.42-3.45, 3.49-3.51];
- Implementation of a novel memory copy strategy between host and device [3.41, 3.42].

The work has been sustained through public national and European funded projects:

- “MD PAEDIGREE – Model-Driven European Paediatric Digital Repository”,
- “HEART – High PErformance Computing of PersonAlized CaRdio Component Models”

4. Decision support

The significant evolution of wireless mobile communication technologies, especially by the end of 90s (e.g. release of 3G) allowed data transfer rates that enabled a different approach in regard to both industrial and consumer applications. This development facilitated wider usage of internet services and thus any device - even located in a remote/isolated area- that had a GPRS interface with/without an IP (internet protocol) address allocated, could communicate easily with central servers for transferring relevant volumes of data, get commands or interact with other similar devices. Internet of Things (IoT) has strongly developed thanks to this. Efforts have been invested that the devices to provide secure and interoperable access interfaces, to be predictable, to be easily “plugged in” a network of similar devices that together compose complex systems.

On the other side, internet and the various web technologies have become of significant relevance in the domains where devices are interconnected in different ways. The XML -based (Extensible Markup Language) web services paradigm - introduced relatively in synch with the evolution of 3G technology- that interconnects software solutions through a lightweight infrastructure, is based on a language and platform neutral connectivity [4.1].

In the context of this technological evolution, an early interest has been from the author side to investigate how to take advantage in the direction of collecting large volumes of data from remote areas, process and initiate decisions. Mainly, two areas were in focus:

- monitor and generate alerts regarding processes (e.g. environment, production lines, etc.);
- the optimization of manufacturing processes in production plants.

4.1 Monitor and inform

It has become important to get informed about the pollution levels anytime, especially when they exceed certain limits that are considered unsafe or that are stipulated by local laws.

Therefore, it has been the goal to develop a SW-HW system [4.2, 4.3] that could ensure:

- up-to-date information about the composition of the air based on user location;
- a practical interface, available to users from anywhere;
- pollution warnings via SMS and email for registered users;
- a structured way to view information about the evolution of pollution in certain locations.

The architectural approach was a client-server model, the clients being materialized by a generic PDU (*Pollution Detection Unit*) device and the server by a central unit which collects and stores the atmospheric data transmitted by the PDU devices, as it is represented in figure 4.1.

4.1.1 HW description

A PIC16F877A microcontroller is the main component of the pollution detection unit. The operating system that runs inside the chip coordinates the measurement process, the acquisition of the GPS coordinates and the data transmission to the central server. The microcontroller is mounted on a development board that provides an RS232 serial communication to the *Telit* GM862 GPRS modem and GPS receiver and a parallel connection to the sensors.

The following sensors have been used for the measurement process:

- carbon monoxide: TGS2442 (Figaro);

- carbon dioxide: TGS4160 (Figaro);
- nitrogen dioxide: NO23050 (Sensoric);
- sulfur dioxide: SO2BF (Sensoric);
- ozone: O33E1 (Sensoric).

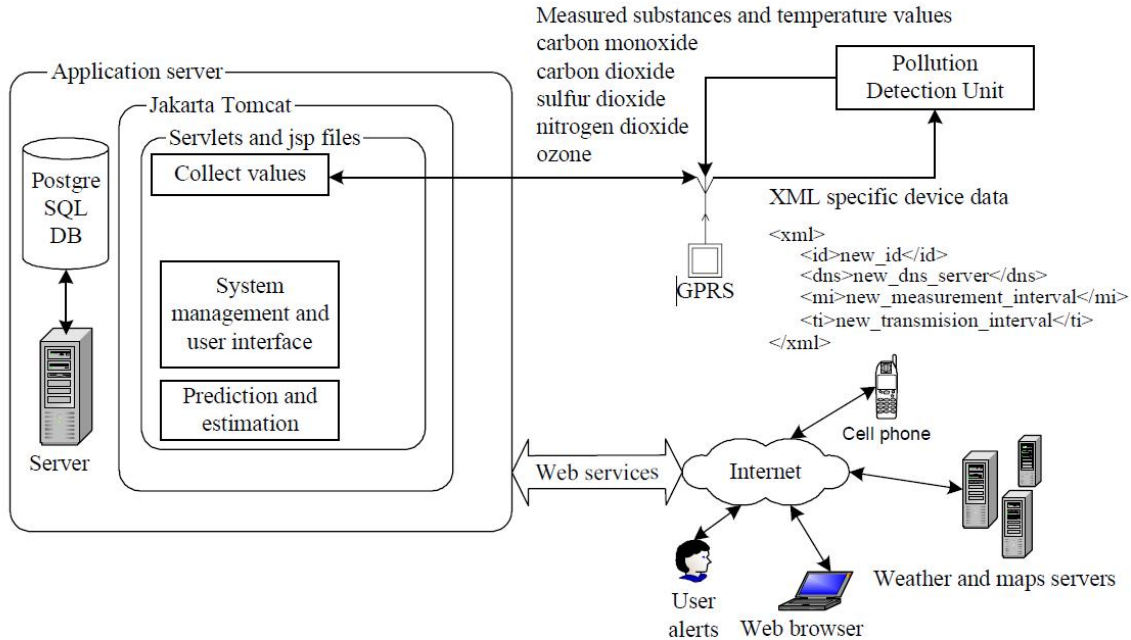


Fig. 4.1 - System Architecture

Operational amplifiers and transistors are used to ensure consistency of the signals provided by the sensors(see figure 4.2 – as example).

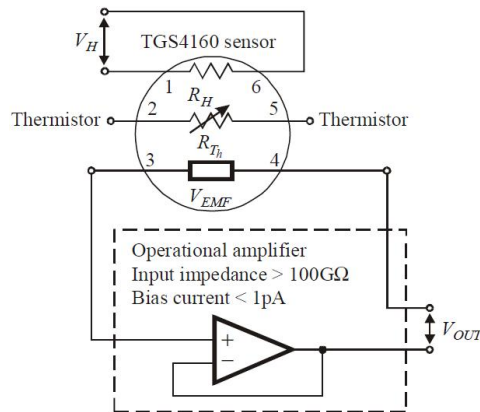


Fig. 4.2 - Electronic schematic for CO2 measurement

The actual measurement process of the substances present in the atmosphere takes place at a specific time interval m_i (2min). The data transfer to the central server is made according to another time interval t_i which can be set with regard to the variation of the monitored substances (e.g. 1). The calculation of the t_i interval is influenced by factors such as:

- the power needed by the system for a GPRS data transmission;
- the cost of the GPRS data transmission, relative relevant at the time when the system was developed;

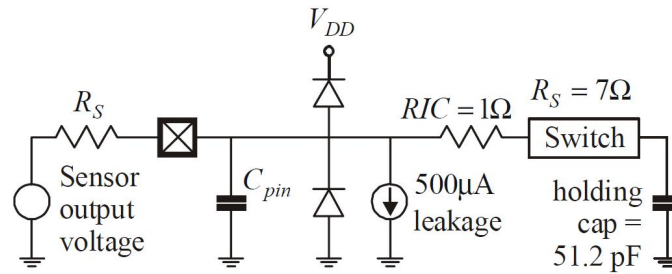


Fig. 4.3 - Electronic schematic for gas sensor

The OP (*Operating System*) of the PIC16F877A microcontroller sends to the server the mean value of the measurement made between two given time intervals t_i (for the initial configuration the mean value of the 6 measurements made during one hour). After this, the internal RAM memory allocated for the intermediate storage of the substances data is freed, thus, making memory space available for the next measurement set, the mean value being sent to the database server at time t_{i+1} . If the GSM network is not reachable, the data is not erased from the memory. New measurement data is stored until the internal and external memory is full. This data is sent to the server when the GSM network is available again.

4.1.2. Communication protocol

For having a correct functioning of the system, a PDU device contains inside his memory system variables like:

- unique identifier which corresponds to a primary key from the table that holds the information about the existing PDU devices,
- the DNS address of the central server
- the m_i and t_i intervals.

The measured data is sent to the central server using the GPRS data communication and the HTTP protocol. The PDU does not have an IP (Internet Protocol) address. The data is sent to the server by accessing the available POST or GET requests methods using the HTTP protocol. If the PDU needs to be reconfigured then the server will return to the PDU an ASCII (American Standard Code for Information Interchange) string in XML format which will be interpreted by the OP of the microcontroller.

The string format is:

```
<xml>
<id>new_id</id>
<dns>new_dns_server</dns>
<mi>new_measurement_interval</mi>
<ti>new_transmission_interval</ti>
</xml>
```

This XML file will be returned to the PDU only in the case of a reconfiguration of the device. The system is able to monitor the toxic pollutants from a remote/isolated area where the GSM network cannot be reached (national parks, etc.). The PDU devices could have been optionally equipped with GPS (Global Positioning System) receivers, the values of the measurements and the coordinates of the measurement place being stored inside the external memory accessed by the PIC16F877A microcontroller.

The m_i interval and the time when the PDU can make measurements depend on the amount of external memory, a larger memory providing to the device a longer period of functioning. If

there are many PDU devices needed for monitoring an isolated region and the cost of the GPS receivers is too high, then the GPS coordinates can be stored inside the database when the system is configured. When the PDU devices are taken inside the area where the GSM network can be reached the data stored in the external memory is sent to the central server.

4.1.3 SW description

The software running on the system consists of two separate modules:

- the central server software;
- the PDU operating system.

The management of the *Pollution Guard* system is made using a web interface provided by the central server, thus the system may be configured and upgraded from any place in the world only by using an internet connection and a web browser. Also, the user could register for air pollution warning on a certain region of interest.

The main function of the management interface is to handle the users logged into the system and the registered PDU devices.

4.1.3.1 Central server software

The server application is written in the Java language. The web server is accessed from the Internet using the *Apache Tomcat Software* [4.16]. *Apache Tomcat* has native support for the JSP-Servlet technology and to function, it needs the installation of a JDK (Java Development Kit) distribution.

The server code can be separated into three subsystems:

- the data acquisition and storage subsystems;
- the user alert subsystem;
- the user interface and administration subsystem.

The data acquisition process is implemented using the HTTP protocol. The transmission of measurement data from the PDU devices will trigger the execution of the Insert-Values servlet. This servlet is responsible of data integrity checking and to verify if the data comes from a valid PDU. If the received data is valid then the servlet will execute the procedure of inserting the data in the database.

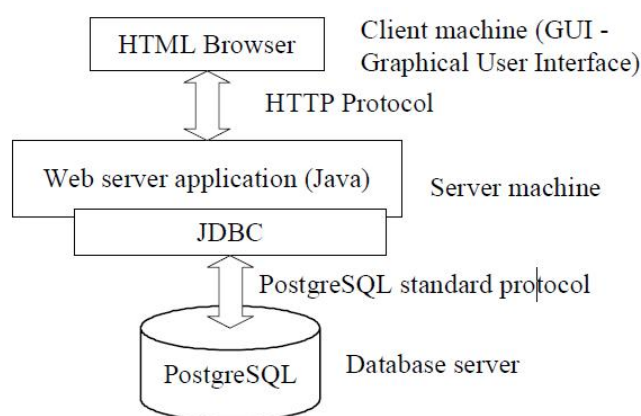


Fig. 4.4 – The architecture of the SW application

A *PostgreSQL* database has been used for storing the incoming data. The data access is made through a set of JDBC (Java Database Connectivity) functions, the interrogation mechanism are implemented using the SQL (Structured Query Language) language. In the JDBC context the

system architecture is represented in Fig. 4.5. The classes for implementing the database access are defined in the package `net.pollution.base.database`. Every table is read through the use of classes derived from the package mentioned above.

The user alert subsystem is linked to the data acquisition subsystem. Before data being stored, the measured values are compared to a threshold value. If this value is bigger than the threshold value a connection to a GPRS modem is made and a series of SMS and e-mail alert messages are sent to the subscribers and also a series of e-mail messages. If a value of one of the substances presented in the air exceeds a certain threshold value, then the GPRS data transmission takes place immediately. When this data will arrive at the central server it will be processed and if it is really an environmental problem then the server will automatically broadcast emergency SMS messages and e-mails to the subscribers.

The concentration value of toxic substances present in the air at a given moment of time is calculated using an AQI (AirQuality Index):

$$AQI = \frac{\text{pollutant concentration}}{\text{pollutant nominal concentration}} * 100 \quad (4.1)$$

This index is calculated only on the server side and displayed in automatically generated reports and charts. The AQI is shown on the pollution map using a color code:

- BLUE – very good (AQI ∈ [0...33]);
- GREEN – good (AQI ∈ [34...66]);
- PURPLE – fair (AQI ∈ [67...99]);
- RED – poor (AQI ∈ [100...149]);
- GRAY – very poor (AQI >150).

For sending SMS messages a *Telit GM862* GPRS modem was used. The user interface and administration subsystem are accessible through a web browser. The code of the interface is written using the JSP-Servlet technology. A snapshot of the user interface representing the distribution of atmospheric pollution over a period of time is presented in figure 4.5.

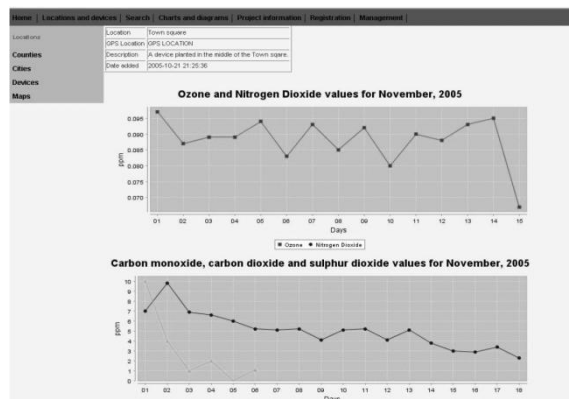


Fig. 4.5 – Application user interface

4.1.3.2 PDU operating system

The flow chart describing the functioning of the OS running inside the PIC16F877A microcontroller is presented in figure 4.6. For the measured data to be transferred to the server, the system makes use of the HTTP protocol. The configuration data needed for the connection is:

- the DNS (*Domain Name Server*) of the server to be contacted;
- the application level protocol: HTTP 1.0 (RFC1945 – Request For Comment).

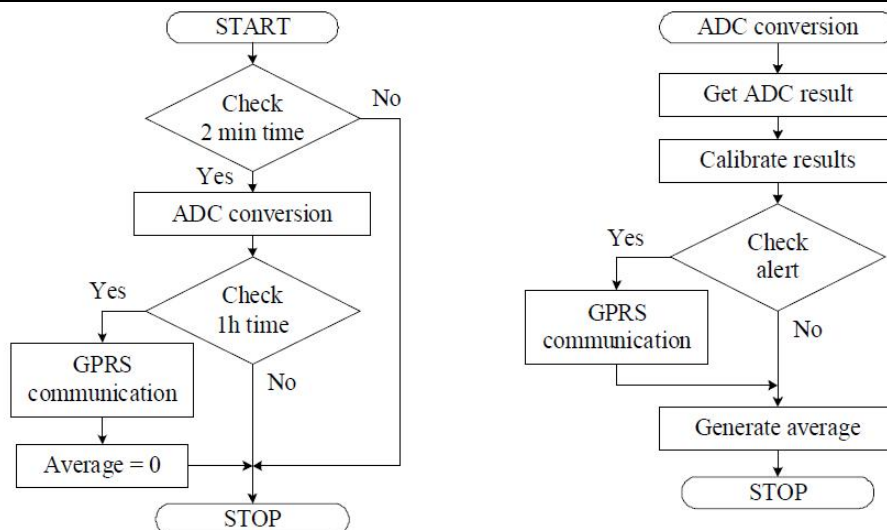


Fig. 4.6 – Diagrams of the OS components – timer interrupts and data transfer

Using the standard AT command set, mapped into the internal RAM of the PIC16F877A microcontroller, the ASCII strings passed to the *Telit* GM862 GPRS modem are:

```

AT+CGDCONT=1,"IP","internet.gprs","0.0.0
.0",0,0,0<cr>
AT#USERID="EASY GPRS"<cr>
AT#PASSW="EASY GPRS"<cr>
AT#SKTSET=0,80,82.78.144.155/servlet/Ins
ertVaues?id=2154&o=3.2&cm=2.1&cd=0.1&sd=
3.1&nd=2.4&t=29><measured_data<cr>
  
```

4.2 Constrained based production optimization using SOA

The costs for setting up production activities and installations are approximately one third from total manufacturing expenses [4.4]. The dynamic of the customer markets, globalization, the need to correlate with raw materials fluctuating prices as well as availability pressure the manufacturing companies to remodel its strategies based on innovation in such a way to find new approaches to configure distribution systems to deliver the desired customer service at the specified due date and at the lowest possible cost while maximizing the enterprise profit. The important aspects for the companies for facing this context, are adaptability and flexibility with focus on optimizing the manufacturing processes.

Therefore, an investigation direction has been to optimize production schedules, and, by automatically using the results of these plans and schedules for the manufacturing of the parts, to produce in a minimum time the required products. This optimization shall be achieved based on a general architecture that can be used, practically, in almost any factory, therefore, to provide as main characteristics: reusability, flexibility and adaptability. In the same time, Software Services represent an important way of fulfilling these requirements.

The solution that should cover the mentioned challenges, has to fulfill the following criteria from a functional point of view:

- optimization of manufacturing processes through the computation of optimal production plans;
- automated usage of the optimal production plans (without the intervention of a human operator);

- development of a flexible and reusable architecture, which shortens the maintenance, installation and setup times and consequently improves the ability to react to changes in the market demand;
- seamless transition from current practice to the novel approach that is presented in the following paragraphs.

The figure 4.7 presents the mapping between the functionalities that the solution based on the designed architecture shall fulfill and the technologies that can enable these requirements.

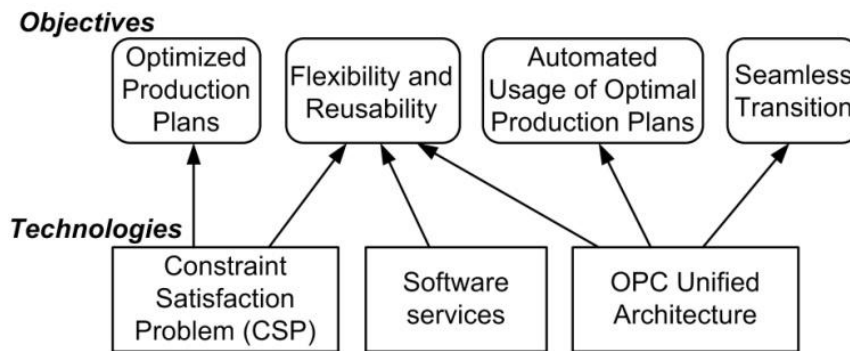


Fig. 4.7 - Mapping between technologies and objectives

Figure 4.8 displays the conceptual design of the architecture that is composed of three main layers:

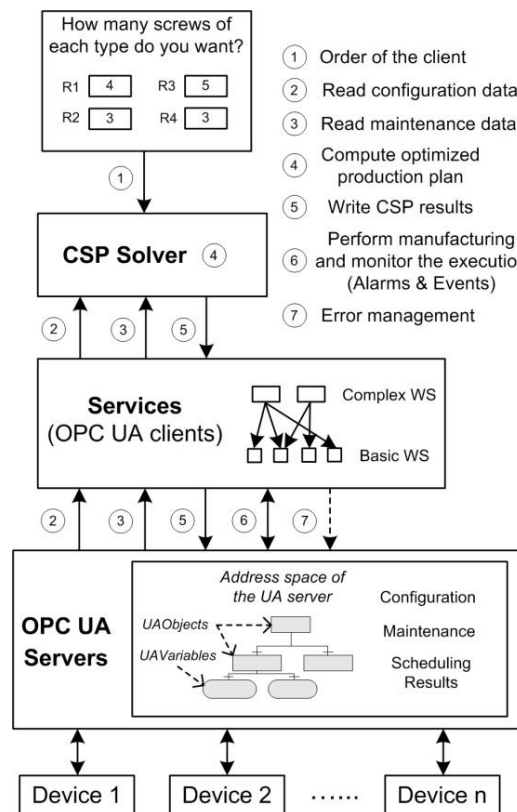


Fig. 4.8 - Main components of the architecture.

-
- The OPC UA servers collect data from all devices, sensors and actuators, model them in a standardized and unified way and assure real-time communication with the devices;
 - The second component is represented by several service layers, which, as backbone of the architecture, guarantee its flexibility and adaptability;
 - The third component of the architecture (Constraint Satisfaction Problem - CSP) solver) addresses its main goal, namely the optimization of production plans and schedules.

The basic idea is that a client places an order (step 1), then, a CSP solver computes an optimized manufacturing plan by using configuration and maintenance data stored inside the UA servers and read through services (steps 2, 3 and 4). Once the manufacturing plan is established, the execution of a complex service is started, which writes the solution to the UA server (and implicitly to the devices, again through services – step 5), and which monitors the execution of the order (step 6). If errors or alarms appear, the complex service performs specific error management activities (step 7). The information stored in the address space of the UA server is used to configure the CSP models, to organize data related to the maintenance of the equipments and to store the scheduling results, while these are used automatically by the manufacturing process.

The constraint satisfaction approach was employed for computing optimized production plans. The main reason for this is the fact that the modeling of a certain application, i.e. the definition of variables, domains and constraints, is completely decoupled from the solution of the model, allowing the approach of optimizing the production plans from an end-user perspective. CSP solvers are able to generate optimal schedules from different points of views, i.e. different optimality criterions, and, last but not least, there are several powerful open-source solvers available for both linear on non-linear models and for different implementation languages.

The architecture relies on the advantages brought by the *service* paradigm, but it keeps the device level unchanged and introduces as base component of the architecture a set of OPC UA servers between the devices and the services. Software services have been used to achieve flexibility. The process of isolating the functionality in a service as one “cell” of execution, for a certain processing step, is generating by default the support for reusability and the migration of the code into the software services allows one to build flexible applications. The main goal of the software services is to allow an easy communication between the OPC UA servers and the solvers used in order to determine the optimal solutions, and to completely decouple the two levels. Further, the flexibility and the reusability of the architecture is enhanced by the development and implementation characteristics of the CSP level (the CSP models automatically adapt to the current state of the devices and of the manufacturing stations) and of the OPC UA level (the address space of the UA servers can be efficiently generated and maintained through a set of specialized algorithms). In order to address the issue of unwanted delays at the software service level, two different strategies were tackled:

- three different frameworks were evaluated in order to adopt an implementation which leads to the smallest execution times,
- the time critical operations were removed from the service level through the introduction of the UA level.

The OPC Unified Architecture is employed to ensure the communication with the devices for automatically using the optimized production plans. In the same time, for covering the products based on classic OPC, a *special adapter software* solution is introduced for performing the migration to the OPC UA standard. One of the main reasons for the introduction of web services directly at the device level has been the fact that classic OPC was very rigid, platform and technology dependent [4.8]. These limitations, together with several others, have been eliminated through the new UA specification.

4.2.1 UA servers

OPC (OLE-Object Linking and Embedding- for Process Control) has been developed based on Microsoft technologies in 1990s to provide a standardized interface between high level SW applications and the field devices regardless who is the producer and which industrial interface/protocol is used at floor shop level. In this way, developers were able to easily create applications that could access through the consistent methods provided by the standard data from, e.g. PLCs (Programmable Logic Controllers) or DCS (Distributed Control Systems) devices as long as the manufacturer of the equipment makes available an OPC Server.

Since OPC is driven by the major companies in industrial automation, the choice to employ this specification for the standardized access of device data, is natural. The total OPC market has over 2,500 vendors providing over 15,000 OPC-enabled products, making thus the seamless transition to new standards extremely important (an aspect which has been completely ignored by the activities focused on web services).

Due to dependency of the classic/initial OPC on Microsoft technologies (not the only reason), a new standard has been developed in the last decade, OPC UA (Unified Architecture) that has aimed, among other goals, to allow multi-platform implementations (e.g. Java), scalability (from field devices to mainframes) or bring more security in line with technological changes. OPC UA provides all data (*current data, alarms and events* and *historical data*) in a unified address space as opposed to the classic OPC specifications, which were separated. OPC UA on the other hand also introduces an extensible meta model where each item has a type definition, much like in an object oriented environment.

The lowest level in the architecture depicted in figure 4.8 is represented by the OPC UA servers and the address space is the most important concept of the UA specification and all the other functional blocks should be realized upon it [4.7]. The basic units of the address space are the nodes and the references which connect the nodes. Every node has attributes which are determined by its type, however there are some attributes common to all nodes, e.g. nodeid. There is a base node, which is an abstract node type, and which can not be instantiated. All other nodes of the address space are derived from the base node: referencetype, object, objecttype, variable, variabletype, method, datatype and view. Through references, all the nodes of the address space are organized into a mesh. Fully functional UA servers contain thousands of nodes in order to provide clients all the information they need and this is why it is important to implement an efficient way of generating the address space, which also facilitates easy maintenance.

The architecture includes a special software adapter that allows the access to the already installed device base that is accessible through classic OPC.

The performance of the algorithms used to efficiently generate the address space has been tested for both the initial development and for the maintenance of the address spaces. Three tests have been designed for the initial development phase, which consisted in the generation of the address space for an OPC UA server, for three different applications.

Table 4.1 displays the results of the tests. The times shown in the table are measured in minutes and, because the number of nodes varies from one scenario to another, the normalized time which corresponds to an address space of 100 nodes was also computed. All three cases showed a significant improvement in terms of development time (a speed-up of around 2.2-2.3x). The same three applications have been used for the evaluation of the maintenance phase. During the tests, 10 UA nodes of the previously generated address spaces needed to be changed/removed and, although the improvements are not as significant as before, an important speed-up of around 1.4-1.6x has been obtained.

TABLE 4.1 SPEED-UP OF THE DEVELOPMENT TIME

Test scenario	Nr. of nodes	Time without algorithms	Time with algorithms	Normalized time with algorithms	Speed-up
Air conditioning	70	365	157	224	2.32
Transport system	100	495	217	217	2.28
Inspection cell	200	958	430	215	2.23

The automatic generation of the space address contributes to the improvement of the flexibility and reusability at OPC UA level due to the reduction of development, deployment and maintenance time.

4.2.2 Software services

Service Oriented Architecture has been present in the SW area since decades in various forms. Its philosophy has been well exploited in the context of web/network applications based on the concept of aggregating self-contained “boxes” (not relevant for the users how these services are implemented) that provides an expected functionality/business logic towards the other services with which it is interconnected. SOA based application integrate components that are distributed, separately-maintained and deployed.

At the same time, in the context of industrial applications, as a result of the SW technologies evolution, the classic OPC has become very rigid, platform and technology dependent, forcing, thus, the introduction of web services directly at the device [4.8].

The middle tier of the architecture (figure 4.8) exploits the idea of developing a SOA in connection with OPC UA(OPC UA specification itself has been developed around the corner poles of SOA), considering that UA defines 51 services as main mean of communication and abstraction between UA clients and UA servers (these services can be grouped into different categories: read and write data, subscribe for data changes and events, call methods, access history of data and events, find information in a complex address space, and modify the address space).

The SOA tier consists of two layers:

- basic services - perform a general set of operations and are integrated in any usage scenario of the architecture, providing additional abstraction from the artefacts lying beneath;
- complex services - interact with several basic services in order to complete their operations.

The fact that the services defined in the middle tier reside above OPC UA services leads to more flexibility and smaller reaction times when changes are needed.

4.2.2.1 Basic services

There are eight basic services:

- WriteVariableNodeService,
- ReadVariableNodeService,
- CallMethodService,
- ReadAlarm-StateService,
- ReadAlarmEventService,
- ReadRawDataService,
- ReadProcessedService,
- ReadAtTimeService.

All basic services (some of the complex ones, too) act as clients from the UA server point of view. Every service which simultaneously represents an UA client has to establish a connection to the UA server in order to be able to retrieve the data requested by the service client. To reduce the response time for a service client, a session approach is used, as displayed in figure 4.9: each service keeps itself track of the connections, which means that each service will contain a map of connections for the entities/users which use it. The execution time is reduced significantly if an UA connection can be reused (see performance tests section). In order to limit the amount of open connections, UA connections are automatically closed after a certain time of inactivity.

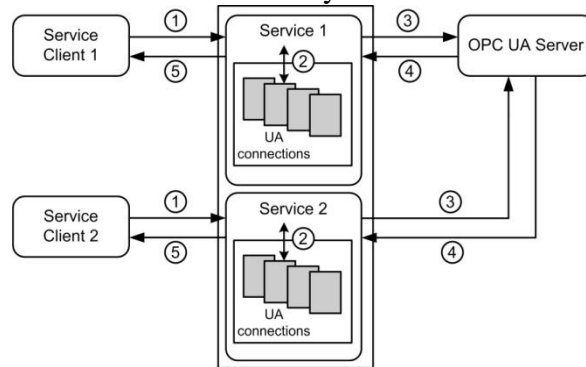


Fig. 4.9 - Session management

This approach can be easily programmed at service level (no actions need to be performed at service server level) and more specialized connections (particularized for each type of service) can be established. The increased number of connections does not represent a disadvantage, since the UA servers can manage high numbers of connections and operations without significant loss of performance.

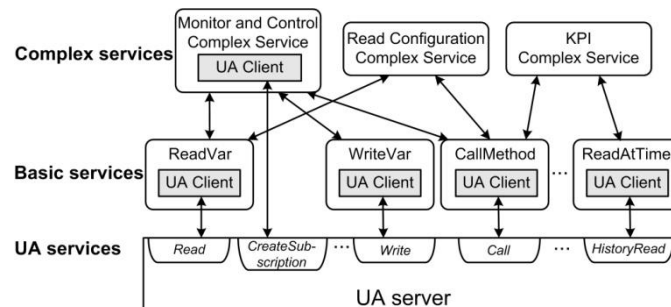


Fig. 4.10 – The mapping between different types of services

A clear distinction is necessary to be made between the basic services of the architecture and the OPC UA standard services (figure 4.10). The basic services are much more coarse-grained than the UA services; they are used to completely decouple the lower part of the architecture (the UA server and the devices modeled by the UA server) from the upper levels. The connections to the UA server are established and managed inside the basic services (as shown in figure 4.9), and hence, the clients of the basic services do not need to perform any OPC UA related operations (as shown in figure 4.10, the only exception to this rule is given by the complex services used to control and monitor the manufacturing). This aspect enhances the flexibility of the architecture since the lower and the upper level of the architecture are decoupled. The basic services, by acting as UA clients, in fact use the standard UA services in order to perform their operations.

The basic services can be completely reused when a new application is developed. Further, the basic services also completely decouple the UA server and the CSP level, thus increasing the

flexibility of the architecture (for an existing application, the CSP and the UA level can be changed independently)

4.2.2.2 Complex Services

The complex services can be divided into two groups:

- services which control and monitor the production
- services which provide general information regarding the manufacturing process.

These complex services interact with several basic services in order to complete their operations. Complex services from the first category receive solutions from CSP models and control and monitor manufacturing processes (these are usually longer running services).

Prior to determining optimized solutions, the CSP model calls a complex service in order to determine the current state of the physical devices, for setting the parameters of the model. Services can also be used to determine values of key performance indicators (KPI) at the enterprise level.

A specific problem of the first group of services is related to the monitoring of the alarms and events. Although these services call several basic services in order to fulfill the task, the basic services cannot be used to detect emergency states and alarms.

Therefore, when a complex service is called, first it establishes a connection to the UA server and it subscribes for the alarms and events which correspond to the current action performed by the complex service. Then, throughout its execution, it calls several basic services in order to control the manufacturing process according to the optimized solution proposed by the CSP model.

If an error occurs during the manufacturing process, either it stops the execution and notifies the engineer, or it calls a special service used to handle the error.

The complex services have to perform certain computations and actions. Since the complex services call several basic services, they act as service clients. In the same time, complex services are application specific.

4.2.3 Optimization driven constraint satisfaction models

The highest level of the architecture proposed herein is represented by the CSP models. Constraint programming is a paradigm aimed at solving combinatorial optimization problems.

The two main goals of the CSP domain are the formulation and the resolution of the combinatorial problems [4.9]. This is an effective way of solving several industrial problems such as scheduling, planning or design of timetables. The user has to only build the model of the problem, he is not interested in the way the problem is solved.

The problems solved through this approach are called *Constraint Satisfaction Problems* (CSP) and consist of:

- a set of variables: x_1, x_2, \dots, x_n ;
- a set of possible values for each variable: D_1, D_2, \dots, D_n ;
- a set of constraints which restrict either the values of a single variable (unary constraint) or the values which a set of variables can simultaneously take (binary constraints, ternary constraints etc.).

The solution of a CSP problem consists of a tuple $v = \{v_1, v_2, \dots, v_n\}$ specifying a value for each variable, values which satisfy all constraints.

Several frameworks are suitable to implement Constraint Programming [4.10]: ECLIPSE, CHOCO, KOALOG, ILOG SOLVER, ILOG SCHEDULER, ILOG OPL. A Java based open source solver could be a good option for research activities from license perspective as well as

integration with OPC UA [4.11]. Through the combined use of the solver and the OPC UA server, it is possible to use immediately and automatically the solutions of the problems without any human intervention [4.12]. Two important open-source Java solvers are Choco and JaCoP. The Choco solver has a better documentation and the code is easier to understand [4.13]. Also its API contains more constraints when compared to JaCoP, which has a rather minimalistic approach regarding the supported constraints, and it allows the use of askVariables which is useful for scheduling problems. The only disadvantage of Choco is that it requires more system resources and it has longer solving times. An important feature of Choco is the possibility to define an objective variable that is used to determine the best solution of all possible solutions of the problem. The solver will seek to find the values which either maximize or minimize the chosen objective variable, as specified by the user.

As the goal is to manufacture the ordered products as soon as possible and to perform all the necessary steps automatically without human intervention, an optimized execution plan is determined through the mixed integer programming (MIP) models. MIP models have been chosen because they provide an efficient mechanism for optimizing decisions which occur in complex systems, i.e. for scheduling and planning problems [4.9], whose main objectives are to allocate scarce resources to different activities over time. The MIP approach is a sub-branch of the CSP (Constraint Satisfaction Problems) paradigm and therefore any CSP solver can be used instead of the mixed integer models. One novelty [4.6] is the fact that the input data for the MIP models are represented inside the address space of the UA server, a fact which contributes to the flexibility of the architecture [4.15]. An important step in building a MIP model is to define a set of decision variables which represent choices and need to be optimized. Afterwards a set of restrictions are defined for these decision variables. Finally, an objective function, that is a linear function of the decision variables, is defined.

4.2.3.1 MIP algorithms

Running scenario - In order to explain elements of the architecture the following use case is described:

A factory uses a flexible assembly system (FAS) which is composed of 3 assembly stations ($i=1,2,3$), and a Loading/Unloading station. The FAS structure for the current example is presented in figure 4.11 and four types of products may be manufactured. For assembling of each product a certain sequence of tasks, composed of up to four different tasks of the total of six tasks which can be performed by the stations of the FAS, has to be followed (figure 4.12). A client orders four products (one of each type). The goal is to assemble all the products as soon as possible and to perform it automatically.

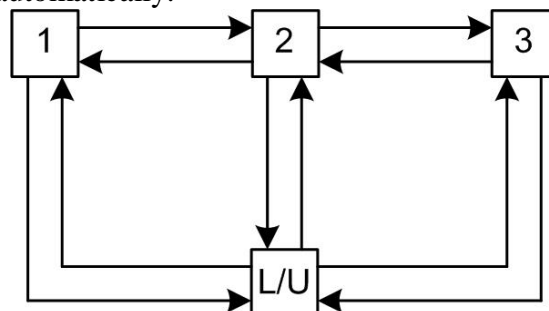


Fig. 4.11 - Flexible assembly system (FAS)

Thus, the example will be solved through the described architecture involving MIP server on CPS solver part. Two main aspects of this application are the loading (association of assembly tasks and component suppliers among the assembly stations with limited work space) and

scheduling (determines the sequencing as well as the timing of all tasks and of all products, by maximizing the system productivity) of the FAS.

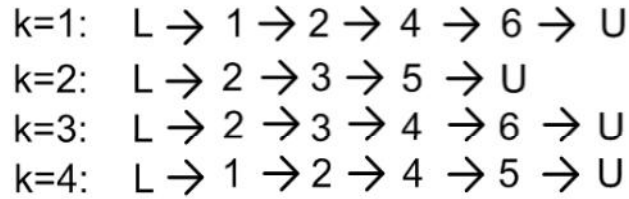


Fig.4.12 - Tasks corresponding to each type of product

The input parameters read from the address space are:

- a_{if} (work space required for assignment of task f to station i),
- b_i (total work space of station i),
- e_{ifk} (earliest end time in station i for task f of product k),
- q_{ifk} (assembly time on station i for task f of product k),
- I_f (subset of stations capable of performing task f),
- F_k (subset of tasks required for product k),
- Q (a large positive constant not less than the schedule length),
- R_k (a set of immediate predecessor/successor pairs of tasks (f,g) for product k such that task $f \in F_k$ must be processed immediately before task $g \in F_k$).

The following decision variables are defined:

- c_{ifk} (completion time on station i of task f of product k),
- x_{ifk} ($x_{ifk}=1$ if product k is assigned to station i to perform task f ; otherwise $x_{ifk}=0$),
- y_{ifkgl} ($y_{ifkgl}=1$, if on station i task f of product k precedes task g of product l ; otherwise $y_{ifkgl}=0$),
- y_{fkgil} ($y_{fkgil}=1$, if task f of product k precedes task g of product l when both tasks are assigned to the same station; otherwise $y_{fkgil}=0$),
- z_{if} ($z_{if}=1$ if task f is assigned to station $i \in I_f$; otherwise $z_{if}=0$)

where the used indices have the following significances:

- f (assembly task $f \in F$),
- i (assembly station, $i \in I = \{1, \dots, m\}$),
- k product, $k \in K = \{1, \dots, n\}$);

Regarding the implementation of the MIP models, two different approaches are presented in the context of the defined goal [4.6], applied on the use case:

- an integrated approach (a large monolithic MIP model is used to simultaneously take the loading and scheduling decisions) – implies longer execution periods having somehow better results;
- a hierarchical approach (two different MIP models are solved sequentially, the first one for the balancing of the workloads and the second one for the detailed scheduling of the tasks by using the results of the first model) – near optimum solutions in a shorter time than the previous model.

1. Model L1a

Model L1a is used for the loading of a flexible assembly system with single stations and alternative routing [4.6, 4.15]. The optimization criterion is represented by the minimization of the maximum workload (W). First, task assignment constraints have to be specified (each task is

assigned to at least one assembly station – more than one is allowed since alternative routing is admitted) and the total space required for the tasks assigned to each assembly station cannot exceed available finite work space available for each station. Therefore, the defined restrictions ensure these conditions:

$$\sum_{i \in I_f} z_{if} \leq 1; f \in F;$$

and

$$\sum_{i \in I_f} a_{if} z_{if} \leq b_i; i \in I.$$

The product assignment constraints are defined (each product is assigned to exactly one station in order to perform each task and each product has to be routed to the stations where the required tasks can be performed) through the following restrictions:

$$\sum_{i \in I_f} x_{ifk} = 1; k \in K; f \in F_k$$

and

$$\sum_{i \in I_f} x_{ifk} \leq z_{if}; k \in K; f \in F_k; i \in I_f$$

The total assembly time required to perform tasks for products assigned to a certain station cannot exceed the maximum workload to be minimized and, therefore, constraints which take into account the maximum workload allocated to each station are defined as:

$$\sum_{k \in K} \sum_{f \in F_k} q_{ifk} x_{ifk} \leq W; i \in I$$

For every model variable non-negativity and integrality conditions have to be considered.

2 Model L/SL1

Model S|L1 is used for the scheduling of a flexible assembly system with single stations and prefixed product assignments [4.6]. The optimization criterion is represented by the minimization of the maximum completion time of the parts (*cmax*). First, product non-interference constraints are defined, i.e. no two products assigned to the same station can be performed simultaneously and the restrictions are:

$$c_{ifk} + Q y_{fkg} \leq c_{igl} + q_{ifk}; k, l \in K; f \in F_k; g \in F_l; i \in I: k < l; x_{ifk}^L x_{igl}^L = 1$$

and

$$c_{igl} + Q(1 - y_{fkg}) \leq c_{ifk} + q_{ifk}; k, l \in K; f \in F_k; g \in F_l; i \in I: k < l; x_{ifk}^L x_{igl}^L = 1$$

These product non-interference constraints are defined only for such pairs of tasks *f* of product *k* and *g* of product *l* which are assigned to the same station *i*, i.e. for $x_{ifk}^L x_{igl}^L = 1$.

The second step is to define product completion constraints which means that the completion time of each task of a product assigned to some station cannot be less than its earliest completion time on that station ($c_{ifk} \leq e_{ifk}; k \in K; f \in F_k; i \in I_f: x_{ifk}^L = 1$) and that each task of each product cannot be started until its immediate predecessor task is completed:

$$c_{hgk} - q_{hgk} \leq c_{ifk}; k \in K, (f, g) \in R_k, i, h \in I: i^1 h, x_{ifk}^L x_{hgk}^L = 1$$

and successive tasks of each product assigned to the same station are performed contiguously:

$$c_{igk} - q_{igk} = c_{ifk}; k \in K, (f, g) \in R_k, i \in I: x_{ifk}^L x_{hgk}^L = 1$$

The last two constraints maintain for each product the precedence relations among its tasks.

The third step is to specify maximum completion time constraints which means that the schedule length is determined by the latest completion time among all products:

$$c_{ifk} \leq c_{\max}; k \in K; f \in F_k; i \in I_f: x_{ifk}^L = 1$$

and that the schedule length cannot be less than the maximum workload ($c_{\max} \geq W^L$, where W^L is the solution value of the corresponding loading problem L1a). The final step is to define variable non-negativity and integrality conditions.

The use case was implemented in a near real-life problem [4.6, 4.15], which does not contain all the extensive details, but whose complexity suffices in order to extensively test the functionality and utility of the architecture. The sequence and process duration for each machine is presented in figure 4.13. The number of each block is the part number. There are four parts each with 3-4 tasks. The task number is not displayed, but the order of the tasks for each part is the one shown in figure 4.13. The first model (L1a) was used for the loading of a FAS with single stations and alternative routing and has determined a maximum workload W of 17 with an execution time of 85.3seconds. The hierarchical approach leads to an execution time of 49.6seconds having the scheduling of a FAS with single stations and prefixed product assignments and determined a maximum workload of 22.

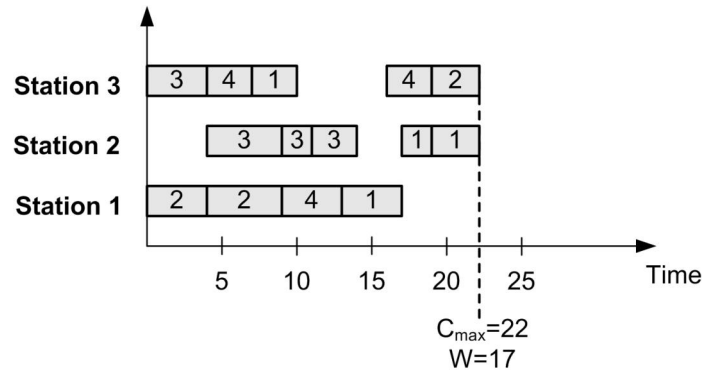


Fig. 4.13 – MIP results

The challenge in finding the optimum solution for a certain CSP model is that the execution time increases exponentially with increasing complexity of the model.

In general, when the CSP models are developed, a trade-off is made between the performance and the flexibility of these models. If the models are tightly tailored to the current setup of the application, then the execution time of the solver improves, but this approach becomes time-consuming when changes need to be performed. On the other side, if the models are more generic, execution time might increase, but the flexibility improves greatly.

4.2.4 Performance tests

Several tests have been performed [4.5] for analyzing the behavior of the proposed solution in various contexts and as well to have a comparative evaluation of different frame that could be used on the service component.

4.2.4.1 Basic service execution times

The first two tests have been designed in order to choose the most suitable service framework. The eight basic services have been implemented using all three service frameworks and table 4.2

displays the average execution times together with the standard deviations. The results are displayed only for the first four services. There is an important difference between the first call of a service for a certain user and subsequent calls, which is given by the session management approach. The values in the table are average values for the subsequent calls, whereas the number displayed in parenthesis is an average value for the first call.

TABLE 4.2 BASIC SERVICE EXECUTION TIMES

Basic service	Apache CXF [ms]	Jersey REST [ms]	Apache River (Jini) [ms]
WriteVariable	280.9 ± 10.5 (1102.8)	309.7 ± 7.59 (907.7)	10.1 ± 0.94 (1097.3)
ReadVariable	244.2 ± 2.83 (1259.3)	233.0 ± 3.36 (1312.1)	6.2 ± 0.41 (1142.8)
CallMethod	312.3 ± 12.7 (1167.6)	318.5 ± 23.9 (1078.3)	13.4 ± 0.56 (1112.5)
ReadAlarmState	245.4 ± 2.56 (1243.7)	235.9 ± 2.78 (1308.2)	6.6 ± 0.32 (1245.9)

The results indicate that Apache River services are considerably faster (more than one order of magnitude) than both web service frameworks, which are comparable to each other. In terms of the first service call, all three frameworks perform similarly since the execution time is given mainly by the time needed to connect to the UA server.

4.2.4.2 Reading and writing of boolean variables

A second service specific test has been performed in order to evaluate the performance of the read and write services for boolean variables, which are the most encountered ones in factory automation applications. All boolean variables are stored in groups of 16 inside *Int16* variables. The execution times are displayed in table 4.3, Apache River services being again much faster than the web service frameworks.

TABLE 4.3 READING AND WRITING OF BOOLEAN VARIABLES

Basic web service	Apache CXF [ms]	Jersey REST [ms]	Apache River (Jini) [ms]
Read 100 boolean variables	1059 ± 34	918 ± 32	37.8 ± 2.1
Write 100 boolean Variables	1217 ± 28	1340 ± 57	59.3 ± 3.2

Nevertheless, since the difference in execution time is significant in the favor of Apache River services, following tests and applications(4.2.3.3÷4.2.3.5) are implemented based on Apache River in order to assure shorter communication and reaction times.

4.2.4.3 Roundtrip test

This test has the role to evaluate the speed and responsiveness of the connection between the UA server and the controller devices. Since the current implementation of the architecture relies on the *special adapter software* solution, a classic OPC server enables the communication between the devices and the UA server. Figure 4.14a displays the roundtrip test: two integer variables (*Var1* and *Var2*, which are prefixed with a *c* inside the classic server and with a *p* inside the PLC) which are present in the address space of the UA and of the classic server and at PLC level, where the value of *Var1* is assigned to *Var2*. When a new value is assigned to *Var1*, the value will then be assigned, in this order, to the following variables: *cVar1*, *pVar1*, *pVar2*, *cVar2* and *Var2*. The average time needed for this roundtrip test was of $101.2 \pm 8.63ms$, which shows that the communication is very fast and changes at device level are immediately reflected

at the UA server level.

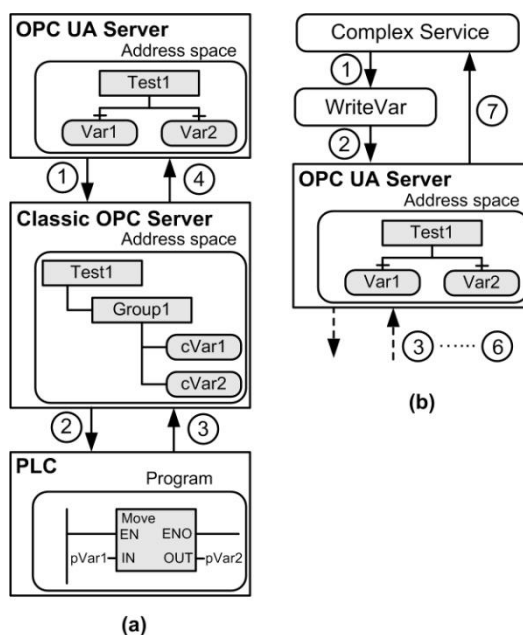


Fig. 4.14 - (a) Roundtrip test, (b) Alarm test

4.2.4.4 Alarm test

The alarm test builds on top of the roundtrip test and is designed for the services which perform the control and monitoring of the manufacturing process. The goal is to determine how fast a complex service is able to react to an alarm which is generated at device level. Figure 4.14b displays the test scenario (the structure underneath the UA server is the same as for the roundtrip test). First, the complex service subscribes to the variable *Var2*, then it calls the *WriteVariableNodeService* in order to write a new value to *Var1* (step 1), which then writes this new value in the address space (step 2). Afterwards, steps 3-6 are identical to steps 1-4 in figure 4.14a and, finally, at step 7 the complex service is notified about the change in the value of *Var2*. The average time needed for this alarm test was of $128.2 \pm 13.4ms$, showing that the architecture is extremely agile and the complex service can react fast to an error by calling a special service, which treats the error.

4.2.4.5 UA Connection test

The UA server can manage a high number of UA connections, and consequently it was considered a test case aimed to evaluate the influence on the execution time of an increased number of UA connections and operations, based on the following scenario: an OPC UA client has to perform 20 UA operations (read and write) in order to finalize a complex activity. The non-UA related operations of this complex activity require two seconds of processing time and up to 50 UA clients need to perform the complex activity at the same time. Table 4.4 displays the results of this test case for two different scenarios (the second column refers to the case when the complex activity requires UA operations, while the results in the third column have been obtained without performing the UA operations). All tests have been performed on a standard Intel Quad-core desktop machine.

The results show that, even when 50 UA connections are established to the same UA server and 1000 UA operations are performed, the additional time required for these operations is

below one second (715ms). The loss of performance with an increasing number of UA connections and operations is insignificant.

TABLE 4.4 UA CONNECTION TEST

Connections (Operations)	With OPC operations [ms]	Without OPC operations [ms]
1 (20)	2188	2003
5 (100)	2253	2007
10 (200)	2313	2015
20 (400)	2403	2044
50 (1000)	2880	2065

4.3 Synopsis

The chapter focused on the following contributions that were presented in ISI journals or ISI conference papers:

- Early investigation of using the advance in mobile communication technologies for monitoring remote and distributed processes [4.2-4.3];
- Exploration and definition of a flexible and unified architecture to [4.5, 4.6, 4.12, 4.15, 4.16]:
 - model, monitor and control manufacturing processes to achieve an optimized production schedule in accordance to market realities(client orders, efficiency in using the production lines, raw materials etc.);
 - provide abstraction, flexibility and reusability through the use of software services and the constraint satisfaction models;
 - easy add of new components (devices) in a manufacturing process through the algorithms used for the automatic generation of the address space and can be readily integrated at the CSP level through the generalized services.

The work has been sustained through public national and European funded projects:

- “Sistem integrat, suport decizional bazat pe fuziunea informatiilor multisenzoriale pentru supravegherea si predictia comportarii barajelor si amenajarilor hidrotehnice – FUZIBAR”
- “Sisteme de reglare cu structura variabila, fara senzori mecanici (sensorless control), pentru controlul direct al cuplului si fluxului masinilor de c.a. , cu aplicatie in servo-sistemele cu miscare incrementala”
- “Sisteme deschise pentru controlul și instrumentarea proceselor”

Part III

The evolution and development plans for career development

5. Academic and Research Career

5.1 Past Research and Academic Activities

My early research period was focused on the modeling and control of electrical drives-induction machines. The doctoral period at Nottingham Trent University had as goal of the thesis to improve the performance of induction motors as an use case derived from oil&gas industry. Therefore, I developed, implemented and validated a prototype that allowed increased efficiency in operation.

Involvement immediately after PhD in industrial projects related to telecommunication, energy or medical domains crystallized the directions to follow in the research side such as the need of customized models for the controlled/investigated systems, the importance of exploiting the continuous advance in telecommunications technologies by collecting and processing in real-time manner the paramount volumes of data collected from remote processes and to provide frames that support decision making in a flexible and dynamic way. As a consequence, the research activities have tackled topics such as personalised cardio-vascular models with focus on non-invasive diagnose of stenosis or models of induction motors with variable rotor impedance, on high performance computing of the developed models as well as on mechanisms for real time planning and scheduling of production lines in factories in accordance with the demands. Beside these, an achievement has been the contribution to the coagulation of a group that is working on the mentioned domains.

The research activities performed up to now have led to the publication of 57 research papers in journals and conferences. I am also author or co-author of six books, including one published at Springer. This work have received over 100 citations (excluding self-citations) and I have an h-index and i10-index of 9, according to google scholar (<https://scholar.google.com/citations?user=Sni114IAAAAJ&hl>). I have been reviewer for IEEE Transactions on Industrial Informatics.

The research work has been facilitated by public national and european funded projects. I was the director of the PNII partnership research project HEART- High PERformance Computing of PersonAlized CaRdio ComponentT Models- (over 3 Milion RONs budget) that had as goals to develop and integrate comprehensive, multi-scale and patient-specific computational models and it brought together 2 academic institutions, a clinic and an industrial partner. I was responsible from Transilvania University side for the EC FP7 project MD-Paedegree-Model-Driven European Paediatric Digital Repository-(over 300.000 euros budget for Transilvania University) that had as objectives to validate and to bring to maturity patient-specific computer-based predictive models of various paediatric diseases, to achieve high-level semantic interoperability- thus requiring standards enabling the clinical contents to be interpreted consistently across the different EHR regimes. I have been contributor in other 9 national or european projects.

The academic activity meant the involvement in teaching or contributing to the lectures or laboratories for subjects such as *Programmable Logic Controllers, Control Engineering, Artificial Intelligence, Industrial Computer Networks, Parallel and Distributing Processing*.

I have been the coordinator of the master program SAATI (Sisteme Avansate în Automatică și Tehnologii Informatice).

5.2 Future Work

There are evolution trends such as digitalization, globalization, urbanization or demographic change that are strongly influenced by social, economic or technology drivers (e.g. related to increasing level of life quality – average life expectation, no. retire vs. productive force) that will have significant impact on the following 5-7 years as is shown in the IEEE Computer Society 2022 Report, figure 5.1.

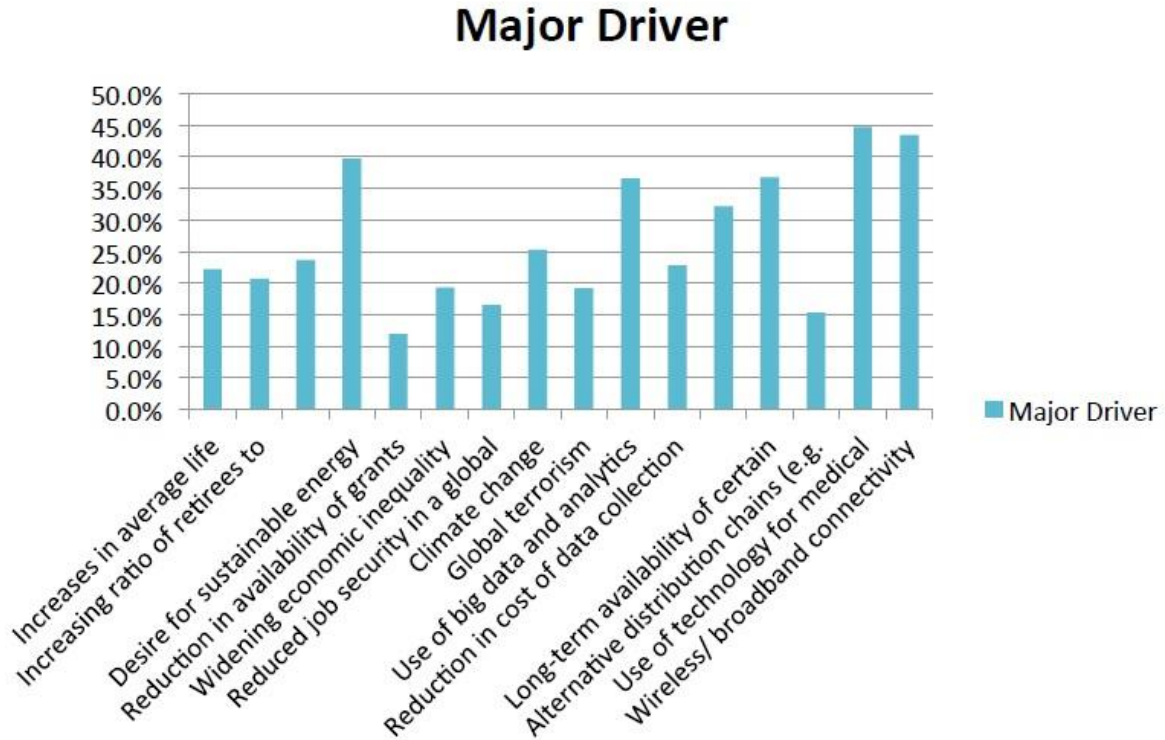


Fig. 5.1 - Major drivers.

These drivers would go along with technology disruptors elements (see fig. 5.2 - IEEE Computer Society 2022 Report). One example is robotization. As digitalization will penetrate all corners of manufacturing, robots will slowly take over more and more complex tasks from human actors, allowing them to focus on other, more creative and non-trivial tasks. Robotization will be a major disruptor of present day manufacturing paradigms such as mechanical industrialization was 100 years ago. Robots will outgrow, with the help of digitalization, the present day paradigm of “pick-and-place”. Interconnected robots, controlled by decentralized control logic, will rearrange automatically based on given tasks – tasks which are in a more human-natural language as compared to the current approach where individual programming is needed for each machine.

The use of 3D printing will also offer a tremendous leap forward in terms of efficiency and speed of manufacturing. 3D printing has the potential of replacing numerous, highly expensive and difficult to operate machinery which forms the basis of present day industrial production chains.

Currently, there are a few initiatives that take advantage of digitalization such as Industry 4.0 or EnergieWende.

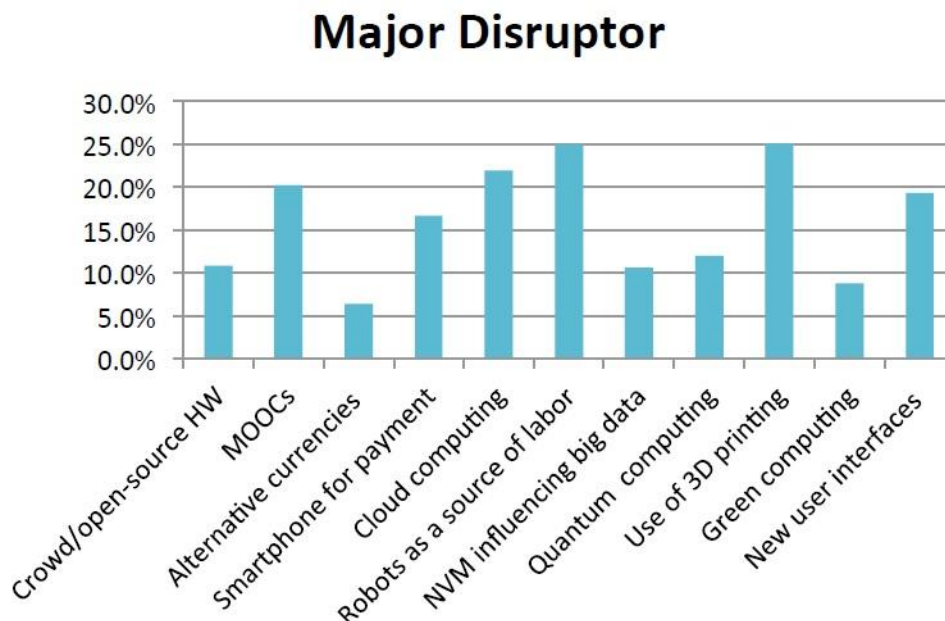


Fig. 5.2 - Major disruptor.

“Industry 4.0” is the so-called 4th industrial revolution. It reflects the present and future trends in industrial automation which, compared to the previous “industrialization wave”, not only targets machinery but data exchange in manufacturing technologies. The current generation automation devices are all connected to a central computer which performs the “heavy duty” operations and synchronizes all machines in order to automate manufacturing. The 4th wave of industrialization includes cyber-physical systems – it’s characterized by a deep, internet based device interconnection in which all machines are equipped with local intelligence and are able to share their experience and problems to other machines as well as human actors. Machines not only collaborate but are tightly connected to the processes they control and the end users. Using the Internet of Things, machines are able to leverage local and remote cognitive capabilities allowing them to perform operations previously impossible even to humans. In this context “Industry 4.0” will create smart factories, modular by design and capable to manufacture a wide range of products. In order to achieve such an amazing feat, the physical factory is represented by a virtual one in order to facilitate the decision processes and mitigate errors that might appear. “Industry 4.0” is a cross-domain endeavor reaching to all internal and external services of a company, bridging the gaps between all parties involved in the manufacturing process. Despite the great efforts in the last years, several open issues are still to be addressed to accelerate the network softwarization paradigm adoption.

The second major driver of digitalization is the “Energiewende” – Germany’s initial term of transitioning to a sustainable, low-carbon energy supply. The “Energiewende” states that a first step in have a low carbon footprint on the environment is to switch from traditional energy sources, such as coal, oil and nuclear to renewable sources such as wind, biomass or photovoltaics. This transition is slowly taking place for a few decades and Europe is the clear leader. Even more, the last years have pushed for even more changes to the classical energy system thus Europe has embarked on a continent-wide radical greenhouse gas reduction. In order to meet these new regulations energy efficiency is of crucial importance therefore our entire energy production, distribution and consumer chains will need replacement. Consumers demand will be reduced by providing more efficient electrical appliances. Electrical distribution networks will be streamlined and steadily replaced by smart grids which will leverage energy consumption, distribution and production in order to reduce losses and increase efficiency.

“Electricity storage” will be developed in order to facilitate storage of renewably generated electric energy in order to feed the grids even when no production is possible (e.g. during the night for photovoltaics).

Next generation 5G cellular networks are expected to meet extremely demanding requirements of mobile data traffic in manifold application scenarios, typical for digitalization, which integrates a great number of revolutionary technologies. Indeed, the explosive number of IoT smart devices, which populate our Smart Cities and Factories, introduces a growing number of connections and much need energy-efficient communications. Over-the-top (OTT) services, such as video streaming, augmented reality and Tactile Internet, require strict requirements in terms of bandwidth and latency. To face these manifold challenges, cellular networks are evolving towards new architecture models driven by Software-Defined Networking (SDN), Network Function Virtualisation (NFV) and Cloud paradigms.

As PwC depicts in their 2016 Global Industry 4.0 Survey, IoT, Analytics and Big Data are some of the core technologies which lie at the foundation of Digitalization.

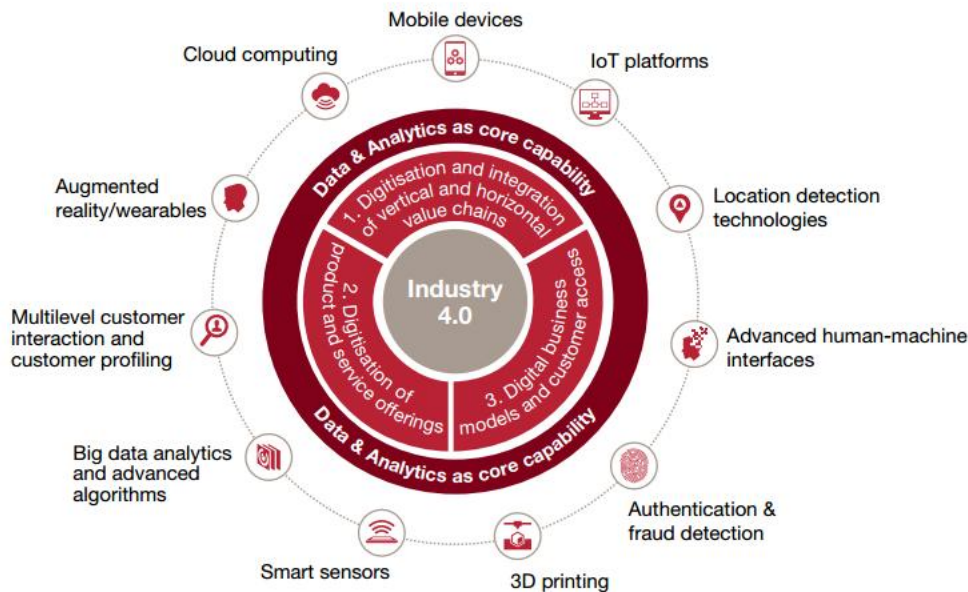


Fig. 5.3 - Integration of Industry 4.0 with various technologies.

These core technologies will enable a rich digital environment which bridges gaps in the physical world. The IoT dream of an interconnected world, from the smallest sensors to the biggest data center, is becoming a reality as we speak. The IoT has been a long-lasting research area which has turned in recent years into a profitable business. Smart “things” that is devices interconnected using IoT services, are forming the infrastructure of the information society which in turn is the foundation for Digitalization. IoT’s most valuable asset is the device agnostic approach to data generation, transfer and consumption. In this context digital information becomes part of the value chain and can thus be exploited in businesses which have a direct and measurable impact into productivity and competitiveness of industrial actors.

In the depicted context, the future investigation work of the author will be go along with the aggregated trends mentioned above in combination with the predicted major drivers and disruptors and it will be focused on 2 main pillars:

- exploitation of Enriched Data Models;
- extreme computing.

Increased intelligence available on the devices/machines/robots, high speed interconnectivity facilitated in the future by 5G technology will generate paramount volumes of data from which will be derived smart information resulted either from local or centralized processing.

The work will consider the processing of the data acquired from a process/devices using semantic based technologies and reasoning combined with data mining offers the possibility to enrich the data model of the respective process. These are the basis in concluding what happen on a specific process/device.

Further, exploiting these models using artificial intelligence, in particular neural networks, will be investigated to perform analytics that shall forecast, in corroboration with domain know-how, what will happen on complex systems (that generate Giga Bytes of data from hundred/thousands of sensors, like in the case of a gas turbine) resulting predictive models. These shall be the basis – involving deep learning – in investigating and developing methodologies and models that shall support activities/decision choices for prescribing what will be the outcome of the chosen decision.

The investigation of real-time monitoring and interaction among processes, devices and intelligent control units will target a closed loop mechanism that starting from fast acquisition of vast data volumes, analyzing, generating command options, evaluating/prescribing the effect of each possible series of commands over the process, in a so called digital twin as it is known.

To support the mechanisms described above will be necessary to investigate options in having extreme processing capabilities, ensuring resolution of described flow in real-time or in reasonable time intervals in respect to the inspected process. Besides the current interest on parallel and distributed processing, an additional direction will be to investigate option to employ quantum computing with the goal to translate already developed or future algorithms/mechanism on quantum computing philosophy.

Use cases that are critical either from operational point of view (e.g as costs) or impacts will be considered, such as:

- extending of patient specific modeling and prescriptive treatment (to evaluate which medication should be administrated related on big data analytics based on historical records of similar symptoms from other patients, what will be the effects and further steps that has to be performed for a positive outcome; or if surgery is the better option);
- prescriptive operations/maintenance for turbines (gas, steam, wind) that are expensive to replace in case of a serious damage and produce significant loses when they are not operational.

The academic activities will be corroborated with the research trends. As coordinator of the master program SAATI (Sisteme Avansate în Automatică și Tehnologii Informaticice) will be the focus to bring into curricula new topics in fields such as analytics, computing, security, cyber physical systems, and to correlate this curricula with the one from the bachelor line of study in such a way to cover in the pre-doctoral period from fundamental knowledge in the field of systems engineering to state of the art topics, as it has been done in the last years too.

References

- [2.1] Vas, P., *Vector Control of AC Machines*, Clarendon Press, Oxford, 1990.
- [2.2] Reinert, J., Parsley, M.J., *Controlling the Speed of an Induction Motor by Resonating the Rotor Circuit*, IEEE Transactions on Industry Applications, Vol. 31, No. 4, July/August 1995, pp.887-891.
- [2.3] Salama, H., Kansara, M., Holmes, P.G., Safar, Y., *Optimal Steady - State Performance of Induction Motor Drives*, OPTIM'96 Conference, pp. 1347-1360, Brasov, 13-15 May 1996.
- [2.4] Suci, C., Kansara, M., Holmes, P.G., Szabo, W., *Phase Advancing for Current in R-L Circuits Using Switched Capacitors*, Electronics Letters, Vol. 35, Issue: 16, August 1999, ISSN: 0013-5194, DOI: 10.1049/el:19990923.
- [2.5] Suci, C., Câmpeanu, R., Câmpeanu, A., Mărgineanu, I., Dănilă, A., *A Virtual instrumentation-based on-line Determination of a Single/two Phase Induction Motor Drive Characteristics at Coarse Start-up*, IEEE International Conference on Automation, Quality and Testing, Robotics, Vol. III, pp. 440-443, Cluj-Napoca, May 22-25, 2008, ISBN: 978-1-4244-2576-1.
- [2.6] Dănilă, A., Mărgineanu, I., Câmpeanu, R., Suci, C., Boian, I., *The Optimization of the Single/two Phase Induction Motor Start-up with Electronically Switched Capacitor*, IEEE International Conference on Automation, Quality and Testing, Robotics, Vol. III, pp. 450-453, Cluj-Napoca, May 22-25, 2008, ISBN:978-1-4244-2576-1.
- [2.7] Câmpeanu, R., Suci, C., Câmpeanu, A., *Electronic Controlled Capacitor for Single Phase Induction Motor*, ICATE Conference, Craiova, 2006.
- [2.8] Ojo, O., Omozusi, O., *Parameter Estimation of Single-phase Induction Machines*, Thirty-Sixth IAS Annual Meeting Conference Record of the 2001 IEEE Industry Applications Conference, 30 Sept.-4 Oct. 2001.
- [2.9] Bala, S., *Dynamics of Single/Two Phase Induction Motors*, University of Wisconsin - Madison, 2004.
- [2.10] Henneberger, G., *Electrical Machines II. Dynamic Behavior, Converter Supply and Control*, Aachen University, 2003.
- [2.11] Yeadon, H., Yeadon, A.W., *Handbook of Small Electric Motors*, New York: McGraw-Hill, 2001.
- [2.12] Suci, C., Kansara, M., Holmes, P., Szabo, W., *Performance Enhancement of An Induction Motor by Secondary Impedance control*, IEEE Transactions on Energy Conversion, Vol. 17, No. 2, pp. 211-217, June 2002.
- [2.13] Formaggia, L. Nobile, F., Quarteroni, A., Veneziani, A., *Multiscale Modeling of the Circulatory System: a Preliminary Analysis*, Computing and Visualizations in Science, Vol. 2, pp. 75-83, 1999.
- [2.14] Quarteroni, A., Veneziani, A., *Analysis of a Geometrical Multiscale Model based on the Coupling of ODE's and PDE's for Blood Flow Simulation*, SIAM Journal Multiscale Model. Sim., Vol. 1, pp. 173-195, 2003.
- [2.15] Mantero, S., Pietrabissa, R., Fumero, R., *The Coronary Bed and Its Role in the Cardiovascular System: A Review and An Introductory Single-branch Model*, Journal of Biomedical Engineering, Vol. 14, pp. 109-116, 1992.
- [2.16] Pijls, N.H., De Bruyne, B., *Coronary Pressure*, Series: Developments in Cardiovascular Medicine, Vol.195, 2nd ed., 2000.

- [2.17] Olufsen, M., Peskin, C., Kim, W.Y., Pedersen, E., *Numerical Simulation and Experimental Validation of Blood Flow in Arteries with Structured-Tree Outflow Conditions*, Annals of Biomedical Engineering, Vol. 28, pp. 1281–1299, 2000.
- [2.18] Wilson, R.F., Wyche, K., Christensen, B.V., Zimmer, S., Laxson, D.D., *Effects of Adenosine on Human Coronary Arterial Circulation*, Circulation, Vol. 82, pp. 1595-1606, 1990.
- [2.19] Hozumi, T., Yoshida, K., Ogata, Y., Akasaka, T., Asami, Y., Takagi, T., Morioka, S., *Noninvasive Assessment of Significant Left Anterior Descending Coronary Artery Stenosis by Coronary Flow Velocity Reserve With Transthoracic Color Doppler Echocardiography*, Circulation, Vol. 97, pp. 1557-1562, 1997.
- [2.20] Passerini, T., de Luca, M., Formaggia, L., Quarteroni, A., Veneziani, A., *A 3D/1D Geometrical Multiscale Model of Cerebral Vasculature*, Journal of Engineering Mathematics, Vol. 64, 2009, pp. 319–330.
- [2.21] Pijls, N.H. et al., *Measurement of Fractional Flow Reserve to Assess the Functional Severity of Coronary-Artery Stenoses*, The New England Journal of Medicine., Vol. 334, pp. 1703-1708, 1996.
- [2.22] Formaggia, L., Lamponi D., Quarteroni, A., *One Dimensional Models for Blood Flow in Arteries*, Journal of Engineering Mathematics, Vol. 47, pp. 251-276, 2003.
- [2.23] Fung, Y., *Biomechanics: Mechanical Properties of Living Tissues*, Springer: New York, 1993.
- [2.24] Malossi, C, Blanco, P, Deparis, S., *A Two-level Time Step Technique for the Partitioned Solution of One-dimensional Arterial Networks*, Computer Methods in Applied Mechanics and Engineering 2012; 237: pp. 212-226, DOI: 10.1016/j.cma.2012.05.017.
- [2.25] Olufsen, M.S., *Modeling the Arterial System with Reference to An Anesthesia Simulator*, PhD Thesis, Røksilde University, May 1998.
- [2.26] Sherwin, S.J., Frankel, V., Peiró, J., Parker, K., *One-dimensional Modelling of a Vascular Network in Space-time Variables*, Journal of Engineering Mathematics, Vol. 47, pp. 217-250, 2003.
- [2.27] Bruinsma, P., Arts, T., Dankelman, J., Spaan, J. A. E. *Model of the Coronary Circulation Based on Pressure Dependence of Coronary Resistance and Compliance*, Basic Research in Cardiology, Vol. 83, 1988, pp. 510-524.
- [2.28] Razminia, M., Trivedi, A., Molnar, J., Elbzour, M., Guerrero, M., Salem, Y., Ahmed, A., Khosla, S., Lubell, D. L., *Validation of a New Formula for Mean Arterial Pressure Calculation: The New Formula is Superior to the Standard Formula*, Catheterization and Cardiovascular Interventions, Vol. 63, pp. 419–425, 2004.
- [2.29] Anderson, H.V., Stokes, M., Leon, M., Abu-Halawa, S., Stuart, Y., Kirkeeide, R., *Coronary Artery Flow Velocity Is Related To Lumen Area and Regional Left Ventricular Mass*, Circulation, Vol. 102, pp. 48-54, 2000.
- [2.30] Jauhiainen, T., Jarvinen, V.M., Hekali P.E., *Evaluation of Methods for MR Imaging of Human Right Ventricular Heart Volumes and Mass*, Acta Radiologica, Vol. 43, pp. 587–592, 2002.
- [2.31] Murray, C. D., *The Physiological Principle of Minimum Work: I. The Vascular System and the Cost of Blood Volume*, Proc. of the National Academy of Sciences of the USA, Vol. 12, pp. 207–214, 1926.
- [2.32] Murray, C. D., *The Physiological Principle of Minimum Work: II. Oxygen Exchange in Capillaries*, Proc. of the National Academy of Sciences of the USA, Vol. 12, pp. 299–304, 1926.
- [2.33] McGinn, A.L., White, C.W., Wilson, R. F., *Interstudy Variability of Coronary Flow Reserve. Influence of Heart Rate, Arterial Pressure, and Ventricular Preload*, Circulation, Vol. 81, pp. 1319-1330, 1990.

- [2.34] Rossen J.D., Winniford M.D., *Effect of Increases in Heart Rate and Arterial Pressure on Coronary Flow Reserve in Humans*, Journal of the American College of Cardiology, Vol. 21, pp. 343-348, 1993.
- [2.35] Steele, B., Wan, J., Ku, J., Hughes, T., Taylor, C., *In Vivo Validation of a One-dimensional Finite-element Method for Predicting Blood Flow in Cardiovascular Bypass Grafts*, IEEE Transactions on Biomedical Engineering, Vol. 50, pp. 649–656, 2003.
- [2.36] Bessems, D., *On the Propagation of Pressure and Flow Waves Through the Patient-Specific Arterial System*, PhD Thesis, Eindhoven, 2007.
- [2.37] Melchionna, S., Bernaschi, M., Succi, S., Kaxiras, E., Rybicki, F.J., Mitsouras, D. et al., *Hydrokinetic Approach to Large-scale Cardiovascular Blood Flow*, Computer Physics Communications, Vol. 181, pp. 462-72, 2010.
- [2.38] Bernsdorf, J., Wang, D., *Non-Newtonian Blood Flow Simulation in Cerebral Aneurysms*, Computers & Mathematics with Applications, Vol. 58 pp. 1024-1029, 2009.
- [2.39] Artoli, A.M., Hoekstra, A.G., Slood, P.M.A., *Mesoscopic Simulations of Systolic Flow in the Human Abdominal Aorta*, Journal of Biomechanics, Vol. 39, pp. 873-874, 2006.
- [2.40] Succi, S., *The Lattice Boltzmann Equation - For Fluid Dynamics and Beyond*, New York: Oxford University Press, 2001.
- [2.41] Zou, Q., He, Z., *On Pressure and Velocity Boundary Conditions for the Lattice Boltzmann BGK Model*, Physics of Fluids, Vol. 9, pp. 1591-1598, 1997.
- [2.42] Bouzidi, M., Firdaouss, M., Lallemand, P., *Momentum Transfer of a Boltzmann-Lattice Fluid with Boundaries*, Physics of Fluids, Vol. 13, pp. 452-459, 2001.
- [2.43] Bailey, P., Myre, J., Walsh, S.D.C., Lilja, D.J., Saar, M.O., *Accelerating Lattice Boltzmann Fluid Flow Simulations Using Graphics Processors*, IEEE International Conference on Parallel Processing, Vienna, Austria, pp. 550-557, Sept. 2009.
- [2.44] Hutchins, G.M., Miner, M.M., Boitnott, J.K., *Vessel Caliber and Branch-Angle of Human Coronary Artery Branch-Points*, Circulation Research, Vol. 38, pp. 572–576, 1976.
- [2.45] Kamiya, A., Togawa, T., *Adaptive Regulation of Wall Shear Stress to Flow Change in the Canine Carotid Artery*, American Journal of Physiology, Vol. 239, pp. 14–21, 1980.
- [2.46] Zarins, C. K., Zatina M. A., Giddens D. P., Ku D. N., Glagov S., *Shear Stress Regulation of Artery Lumen Diameter in Experimental Atherogenesis*, Journal of Vascular Surgery, Vol. 5, pp. 413–20, 1987.
- [2.47] Sharma, P., Itu, L., Xudong, Z., Kamen, A., Bernhardt, D., Suciu, C., Comaniciu, D., *A Framework for Personalization of Coronary flow Computations During Rest and Hyperemia*, 34th Annual International Conference of the IEEE EMBS, San Diego, California, USA, pp. 6665 - 6668, 28 August - 1 September, 2012.
- [2.48] Niță, C., Itu, L.M., Suciu, C., *GPU Accelerated Blood Flow Computation Using the Lattice Boltzmann Method*, IEEE High Performance Extreme Computing Conference - HPEC, 2013, Boston, USA, pp. 1-6, Electronic ISBN: 978-1-4799-1365-7.
- [2.49] Dănilă, A., Mărgineanu, I., Câmpeanu, R., Suciu, C., *Experimental Validation of the Dynamic Model of a Single/two Phase Induction Motor*, 11th International Conference on Optimization of Electrical and Electronic Equipment - OPTIM 2008, pp. 3-8, ISBN 978-1-4244-1544-1.
- [2.50] Itu, L.M., Suciu, C., Postelnicu, A., Moldoveanu, F., *Analysis of Outflow Boundary Condition Implementations for 1D Blood Flow Models*, E-Health and Bioengineering Conference – EHB 2011, ISBN 978-1-4577-0292-1.
- [2.51] Itu, L., Sharma, P., Mihalef, V., Kamen, A., Suciu, C., Comaniciu, D., *A Patient-Specific Reduced-order Model for Coronary Circulation*, 9th IEEE International Symposium on Biomedical Imaging – ISBI 2012, pp. 832 - 835, ISBN 978-1-4577-1857-1.
- [2.52] Itu, L. M., Sharma, P., Zheng, X., Mihalef, V., Kamen, A., Suciu, C., *Patient-Specific Modeling and Hemodynamic Simulation in Healthy and Diseased Coronary Arteries*, ASME

- 2012 Summer Bioengineering Conference - SBC 2012, Fajardo, Puerto Rico, June 20-23, 2012, ISBN 978-0-7918-4480-9.
- [2.53] Itu, L., Sharma, P., Kamen, A., Suci, C., Comaniciu, D., *A Novel Coupling Algorithm for Computing Blood Flow in Viscoelastic Arterial Models*, 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society – EMBC 2013, pp. 727 - 730, ISSN: 1557-170X.
- [2.54] Calmac, L., Niculescu, R., Bădilă, E., Weiss, E., Zamfir, D., Itu, L.M., Lazar, L., Carp, M., Itu, A., Suci, C., Passerini, T., Sharma, S., Georgescu, B., Comaniciu, D., *Image-Based Computation of Instantaneous Wave-free Ratio from Routine Coronary Angiography - Initial Validation by Invasively Measured Coronary Pressures*, Journal of the American College of Cardiology, Vol. 66, October 2015, pp. B17-B18, ISSN: 0735-1097.
- [2.55] Itu, L.M., Passerini, T., Calmac, L., Niculescu, R., Bădilă, E., Weiss, E., Zamfir, D., Peneș, D., Lazar, L., Carp, M., Itu, A., Suci, C., Sharma, S., Georgescu, B., Comaniciu, D., *Image-Based Computation of Instantaneous Wave-free Ratio from Routine Coronary Angiography - Evaluation of a Hybrid Decision Making Strategy with FFR*, Journal of the American College of Cardiology, Vol. 67, April 2016, ISSN: 0735-1097.
- [2.56] Calmac, L., Niculescu, R., Bădilă, E., Weiss, E., Zamfir, D., Peneș, D., Itu, L.M., Lazar, L., Carp, M., Itu, A., Suci, C., Passerini, T., Sharma, S., Georgescu, B., Comaniciu, D., *A Data-driven Approach Combining Image-based Anatomical Features and Resting State Measurements for the Functional Assessment of Coronary Artery Disease*, Journal of the American College of Cardiology, Vol. 68, November 2016, pp. B212-B213, ISSN: 0735-1097.
- [2.57] Itu, L. M., Sharma, P., Passerini T., Kamen, A., D., Suci, C., Comaniciu, D., *A Parameter Estimation Framework for Patient-specific Hemodynamic Computations*, Journal of Computational Physics, Vol. 281, Jan, 2015, pp. 316-333, ISSN 0021-9991.
- [2.58] Itu, L. M., Suci, C., *A Method for Modeling Surrounding Tissue Support and its Global Effects on Arterial Hemodynamics*, IEEE International Conference on Biomedical and Health Informatics – BHI 2014, Valencia, Spain, June 1-4, 2014, pp. 1-4, ISSN: 2168-2194.
- [2.59] Itu, L.M., Suci, C., *An External Tissue Support Model for the Arterial Wall Based on in Vivo Data*, IEEE International Symposium on Medical Measurements and Applications – MeMeA 2014, Lisbon, Portugal, June 11-12, 2014, pp. 1-6, ISBN: 978-1-4799-2922-1.
- [2.60] Itu, L.M., Sharma, P., Georgescu, B., Kamen, A., D., Suci, C., Comaniciu, D. *Model Based Non-invasive Estimation of PV Loop from Echocardiography*, 36th Annual Inter. Conf. of the IEEE Engineering in Medicine & Biology Society - EMBC 2014, Chicago, USA, August 26-30, 2014, pp. 6774-6777, ISSN: 1094-687X.
- [3.1] Parashar, N., Srinivasan, B., Sinha, S., Agarwal, M., *GPU-accelerated Direct Numerical Simulations of Decaying Compressible Turbulence Employing a GKM-based Solver*, International Journal for Numerical Methods in Fluids, Vol. 83 Issue 10, 2017, pp. 737-754, ISSN 1097-0363.
- [3.2] Cireșan, D., Meier, U., Masci, J., Schmidhuber, J, *Multi-column Deep Neural Network for Traffic Sign Classification*, Neural Networks, Vol. 32, pp. 333-338, August 2012, Elsevier, DOI: 10.1016/j.neunet.2012.02.023.
- [3.3] Le Grand, S., Gotz, A.W., Walker, R.C., *SPFP: Speed Without Compromise-A mixed Precision Model for GPU Accelerated Molecular Dynamics Simulations*, Computer Physics Communications, Vol. 184, Issue: 2, pp. 374-380, Elsevier, DOI: 10.1016/j.cpc.2012.09.02.
- [3.4] Bernabeu, M.O., Corrias, A., Pitt-Francis, J., Rodriguez, B., Bethwaite, B., Enticott, C., Garic, S., Peachey, T., Tan, J., Abramson, D., Gavaghan, D., *Grid Computing Simulations of Ion Channel Block Effects on the ECG Using 3D Anatomically-based Models*, 36th Annual Computers in Cardiology Conference – CinC 2009, pp. 213 – 216.

- [3.5] Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C., *GPU Computing*, Proceedings of IEEE, Vol. 96, Issue 5, pp. 879-899, 2008, DOI: 10.1109/JPROC.2008.917757
- [3.6] Shams, R., Sadeghi, P., Kennedy, R.A., Hartley, R.I., *A Survey of Medical Image Registration on Multicore and the GPU*, IEEE Signal Processing Magazine, March 2010, pp. 50-60, ISSN: 1053-5888.
- [3.7] Sato, D., Xie, Y., Weiss, J.N., Qu, Z., Garfinkel, A., Sanderson, A. R., *Acceleration of Cardiac Tissue Simulation with Graphic Processing Units*, Springer Medical and Biological Engineering and Computing Journal, Vol. 47, No. 9, September 2009, pp. 1011–1015, doi:10.1007/s11517-009-0514-4.
- [3.8] Yu, R., Zhang, S., Chiang, P., Cai, Y., Zheng, J., *Real-Time and Realistic Simulation for Cardiac Intervention with GPU*, Second International Conference on Computer Modeling and Simulation - ICCMS '10, Sanya, China, January 2010, pp. 68-72, ISBN: 978-1-4244-5642-0.
- [3.9] Shen, W., Wei, D., Xu, W., Zhu, X., Yuan S., *GPU-Based Parallelization for Computer Simulation of Electrocardiogram*, Ninth IEEE International Conference on Computer and Information Technology - CIT '09, Xiamen, China, October 2009, pp. 280 – 284, ISBN: 978-0-7695-3836-5.
- [3.10] Bruaset, A.M., Tveito, A., *Numerical Solution of Partial Differential Equations on Parallel Computers*, Springer, New York, USA, 2006.
- [3.11] Chen, G., Li, G., Pei, S., Wu, B., *High Performance Computing Via a GPU*, 1st International Conference on Information Science and Engineering, pp. 238-241, 2009.
- [3.12] NVIDIA Corporation. *CUDA, Compute Unified Device Architecture Programming Guide*, v3.1, 2010.
- [3.13] NVIDIA Corporation. *CUDA, Compute Unified Device Architecture Best Practices Guide*, v3.1, 2010.
- [3.14] Jin, Q., Thomas, D.B., Luk, W., *Exploring Reconfigurable Architectures for Explicit Finite Difference Option Pricing Models*, International Conference on Field Programmable Logic and Applications, Prague, Czech Republic, August 2009, pp. 73-78.
- [3.15] Wendt, J.F., *Computational Fluid Dynamics: An Introduction*, 3rd Edition, Springer, Berlin, Germania, 2009.
- [3.16] Chung, T.J., *Computational Fluid Dynamics*, Cambridge University Press, Cambridge, UK, 2002.
- [3.17] Blazek, J., *Computational Fluid Dynamics: Principles and Applications*, 2nd Edition, Elsevier, London, UK, 2007.
- [3.18] Hoffmann, K.A., Chiang, S.T., *Computational Fluid Dynamics, Vol. 1*, Engineering Education System, Wichita, USA, 1998.
- [3.19] Raghu, R., Vignon-Clementel, I., Figueroa, C.A., Taylor, C.A., *Comparative Study of Viscoelastic Arterial Wall Models in Nonlinear One-dimensional Finite Element Simulations of Blood Flow*, Journal of Biomechanical Engineering 2011; Vol. 133(8):081003; DOI: 10.1115/1.4004532.
- [3.20] Reymond, P., Bohraus, Y., Perren, F., Lazeyras, F., Stergiopulos N., *Validation of a Patient-specific One-dimensional Model of the Systemic Arterial Tree*, American Journal of Physiology - Heart and Circulatory Physiology 2011; Vol. 301 (?), pp. 1173-1182, DOI: 10.1152/ajpheart.00821.2010.
- [3.21] Bessems, D., Giannopapa, C., Rutten, M., van de Vosse, F., *Experimental Validation of a Time-domain-based Wave Propagation Model of Blood Flow in Viscoelastic Vessels*, Journal of Biomechanics 2008, Vol. 41, pp. 284-291, DOI: 10.1016/j.jbiomech.2007.09.014.
- [3.22] Mynard, J.P., Nithiarasu, P., *A 1D Arterial Blood Flow Model Incorporating Ventricular Pressure, Aortic Valve and Regional Coronary Flow Using the Locally Conservative Galerkin*

- (*LCG*) Method, Communications in Numerical Methods in Engineering 2008, Vol. 24, pp. 367-417, DOI: 10.1002/cnm.1117.
- [3.23] Passerini, T., *Computational Hemodynamics of the Cerebral Circulation: Multiscale Modeling from the Circle of Willis to Cerebral Aneurysms*, PhD Thesis, Politecnico di Milano, Italy, 2009.
- [3.24] Alastruey, J., Khir, A., Matthys, K., Segers, P., Sherwin, S., Verdonck, P., Parker, K., Peiro, J., *Pulse Wave Propagation in a Model Human Arterial Network: Assessment of 1-D Visco-elastic Simulations Against in Vitro Measurements*, Journal of Biomechanics; Vol. 44, 2011, pp. 2250-2258, DOI: 10.1016/j.jbiomech.2011.05.041.
- [3.25] Courant, R., Friedrichs, K., Lewy, H., *Über die partiellen Differenzgleichungen der mathematischen Physik*. Mathematische Annalen, 1928; Vol. 100, pp. 32-74.
- [3.26] Kumar, R., Quateroni, A., Formaggia, L., Lamponi, D., *On Parallel Computation of Blood Flow in Human Arterial Network Based on 1-D Modelling*, Computing, Vol.71, 2003, pp. 321-351, DOI: 10.1007/s00607-003-0025-3.
- [3.27] Bessems, D., *On the Propagation of Pressure and Flow Waves Through the Patient-Specific Arterial System*, PhD Thesis, Technical University of Eindhoven, Netherlands, 2008.
- [3.28] Zhang, Y, Cohen, J, Owens, J.D., *Fast Tridiagonal Solvers on the GPU*, 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Bangalore, India, 2010, pp. 127-136.
- [3.29] Kirk, D., Hwu, W.M., *Programming Massively Parallel Processors: A Hands-on Approach*, Elsevier: London, 2010.
- [3.30] Sengupta, S., Harris, M., Garland, M., *Efficient Parallel Scan Algorithms for GPUs*, NVIDIA Technical Report NVR-2008-003, 2008.
- [3.31] Itu, L., Sharma, P., Mihalef, V., Kamen, A., Suci, C., Comaniciu, D., *A Patient-Specific Reduced-order Model for Coronary Circulation*, IEEE International Symposium on Biomedical Imaging, Barcelona Spain, 2012, pp. 832-835, DOI: 10.1109/ISBI.2012.6235677.
- [3.32] Formaggia, L., Lamponi, D., Tuveri, M., Veneziani, A., *Numerical Modeling of 1D Arterial Networks Coupled with a Lumped Parameters Description of the Heart*, Computer Methods in Biomechanics and Biomedical Engineering, Vol. 9, 2006, pp. 273-88, DOI: 10.1080/10255840600857767.
- [3.33] Mynard, J.P., Davidson, M.R., Penny, D.J., Smolich, J.J., *A Simple, Versatile Valve Model for Use in Lumped Parameter and One-dimensional Cardiovascular Models*, International Journal for Numerical Methods in Biomedical Engineering, Vol. 28, 2012, pp. 626-641, DOI: 10.1002/cnm.1466.
- [3.34] Willemet, M., Lacroix, V., Marchandise, E., *Inlet Boundary Conditions for Blood Flow Simulations in Truncated Arterial Networks*, Journal of Biomechanics, Vol. 44, 2011, pp. 897-903, DOI: 10.1016/j.jbiomech.2010.11.036.
- [3.35] Senzaki, H., Chen, C.H., Kass D.A., *Valvular Heart Disease/heart Failure/hypertension: Single-beat Estimation of End-systolic Pressure-volume Relation in Humans: A new Method with the Potential for Noninvasive Application*, Circulation, Vol. 94, 1996, pp. 2497-2506.
- [3.36] Westerhof, N., Elzinga, G., Sipkema, P. *An Artificial Arterial System for Pumping Hearts*, Journal of Applied Physiology, Vol. 31, 1971, pp. 776-781.
- [3.37] Cousins, W., Gremaud, P.A., Tartakovsky, D.M., *A New Physiological Boundary Condition for Hemodynamics*, SIAM Journal on Applied Mathematics, Vol. 73, pp. 1203-1223, 2012.
- [3.38] Itu, L.M., Sharma, P., Kamen, A., Suci, C., Moldoveanu, F., Postelnicu, A., *GPU Accelerated Simulation of the Human Arterial Circulation*, 13th International Conference on Optimization of Electrical and Electronic Equipment, Brasov, Romania, 2012, pp. 1478-1485, DOI: 10.1109/OPTIM.2012.6231764.

- [3.39] Stergiopoulos, N., Young, D., Rogge, T.R., *Computer Simulation of Arterial Flow with Applications to Arterial and Aortic Stenosis*, Journal of Biomechanics, Vol. 25, 1992, pp. 1477-1488, DOI: 10.1016/0021-9290(92)90060-E.
- [3.40] Itu, L.M., Suciu, C., Moldoveanu, F., Postelnicu, A., *GPU Accelerated Simulation of Elliptic Partial Differential Equations*, 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 15-17 September 2011, Prague, Czech Republic, pp. 238-242, ISBN 978-1-4577-1426-9.
- [3.41] Itu, L.M., Sharma, P., Suciu, C., *Patient-specific Hemodynamic Computations: Application to Personalized Diagnosis of Cardiovascular Pathologies*, Springer 2017, ISBN 978-3-319-56852-2.
- [3.42] Itu, L.M., Sharma, P., Kamen, A., D., Suciu, C., Comaniciu, D., *Graphics Processing Unit Accelerated One-Dimensional Blood Flow Computation in the Human Arterial Tree*, International Journal on Numerical Methods in Biomedical Engineering, Vol. 29, December 2013, pp. 1428 – 1455, ISSN: 2040-7947.
- [3.43] Itu, L.M., Sharma, P., Suciu, C., Moldoveanu, F., Comaniciu, D., *Personalized Blood Flow Computations: A Hierarchical Parameter Estimation Framework for Tuning Boundary Conditions*, International Journal on Numerical Methods in Biomedical Engineering, Vol. 33, March 2017, pp. e02803, ISSN: 2040-7947.
- [3.44] Itu, L.M., Sharma P., Kamen, A., Suciu, C., Postelnicu, A., Moldoveanu, F., *GPU Accelerated Simulation of the Human Arterial Circulation*, 13th International Conference on Optimization of Electrical and Electronic Equipment – OPTIM 2012, Braşov, Romania, May 24-26, 2012, pp. 1478-1485, ISSN: 1842-0133.
- [3.45] Sharma, P., Itu, L. M., Zheng, X., Kamen, A., Bernhardt, D., Suciu, C., Comaniciu, D., *A Framework for Personalization of Coronary Flow Computations During Rest and Hyperemia*, 34th Annual International Conference of the IEEE Engineering in Medicine & Biology Society - EMBC 2012, San Diego, California, USA, Aug. 28-Sept. 1, 2012, pp. 6665 - 6668, ISSN: 1557-170X.
- [3.46] Vizitiu, A., Itu, L.M., Niţă, C., Suciu, C., *Optimized Three-Dimensional Stencil Computation on Fermi and Kepler GPUs*, 18th IEEE High Performance Extreme Computing Conference, Waltham, MA, USA, Sept. 9-11, 2014, pp. 78-83, ISBN: 978-1-4799-6232-7.
- [3.47] Stroia, I., Itu, L., Niţă, C., Lazăr, L., Suciu, C., *GPU Accelerated Geometric Multigrid Method: Comparison with Preconditioned Conjugate Gradient*, 19th IEEE High Performance Extreme Computing Conference, Waltham, MA, USA, Sept. 15-17, 2015, pp. 1-6, ISBN 978-1-4673-9287-7.
- [3.48] Iacob, A., Itu, L.M., Sasu, L., Moldoveanu, F., Suciu, C., *GPU Accelerated Information Retrieval Using Bloom Filters*, 19th International Conference on System Theory, Control and Computing – ICSTCC 2015, Cheile Grădiştei - Fundata, Romania, October 14÷16, 2015, pp. 872÷876, ISBN: 978-1-4799-8481-7.
- [3.49] Stroia, I., Itu, L., Niţă, C., Lazăr, L., Suciu, C., *GPU Accelerated Geometric Multigrid Method - Performance Comparison on Recent NVIDIA Architectures*, 19th Inter. Conf. on System Theory, Control and Computing - ICSTCC 2015, Sinaia, Romania, October 14-16, 2015, pp. 175-179, ISBN: 978-1-4799-8482-4.
- [3.50] Vizitiu, A., Itu, L., Joyseeree, R., Depeursinge, A., Muller, H., Suciu, C., *GPU-Accelerated Texture Analysis Using Steerable Riesz Wavelets*, 24th Euromirco International Conference on Parallel, Distributed, and Network-Based Processing – PDP 2016, Heraklion Crete, Greece, February 17-19, 2016, pp. 56-61, ISSN: 2377-5750.
- [3.51] Niţă, C., Stroia, I., Itu, L.M., Suciu, C., Mihalef, V., Datar, M., Rapaka, S., Sharma, P., *GPU Accelerated, Robust Method for Voxelization of Solid Objects*, 20th IEEE High Performance Extreme Computing Conference, Waltham, MA, USA, Sept. 13-15, 2016, pp. 50-55, ISBN: 978-1-5090-3526-7.

- [4.1] Jammes, F., Smit, H., *Service Oriented Paradigms in Industrial Automation*, IEEE Transaction on Industrial Informatics, Vol. 1, February 2005, pp. 62-70, DOI: 10.1109/TII.2005.844419.
- [4.2] Suci, C., Moldoveanu, F., Câmpeanu, R., Baci, I., Grigorescu, S.M., Cârstea, B., Voinea, V., *GPRS Based System for Atmospheric Pollution Monitoring and Warning*, IEEE International Conference on Automation, Quality and Testing, Robotics, Cluj-Napoca, Vol. :2, 2006, ISBN 1-4244-0360-X.
- [4.3] Moldoveanu, F., Suci, C., Baci, I., Grigorescu, S.M., Cârstea, B., Voinea, V., *Microcontroller Based SCADA System for Air Pollution Monitoring and Warning*, 10th International Conference on Optimization of Electrical and Electronic Equipment - OPTIM, Vol. III: Industrial Automation and Control, pp. 185-190, May 18-19, 2006, Braşov, Romania, ISBN 978-973-635-705-3.
- [4.4] Gilart-Iglesias, V., Macia-Perez, F., Marcos-Jorquera, D., Mora-Gimeno, F.J., *Industrial Machines as a Service: Modelling Industrial Machinery Processes*, 5th IEEE Inter. Conf. on Industrial Informatics, pp. 737-742, June 2007 DOI:10.1109/INDIN.2007.4384865.
- [4.5] Gîrbea, A., Suci, C., Nechifor, S., Sisak, F., *Design and Implementation of a Service-Oriented Architecture for the Optimization of Industrial Applications*, IEEE Transactions on Industrial Informatics, Vol. 10, Issue 1, February 2014, pp. 185-196, DOI: 10.1109/TII.2013.2253112.
- [4.6] Gîrbea, A., Suci, C., Sisak, F., *An Innovative and Flexible Architecture for Industrial Automation*, 13th International Conference on Optimization of Electrical and Electronic Equipment - OPTIM, 24-26 May 2012, pp. 1085-1092, ISBN: 978-1-4673-1650-7.
- [4.7] The OPC Foundation, *OPC UA Specification: Part 1 – 11*, <http://opcfoundation.org/>, accessed August 2011.
- [4.8] Filho, F.Sd.L., da Fonseca, A.L.T.B., de Souza, A.J., Couto, F.A., dos Santos, R.P.R., Guedes, L.A., *Industrial Processes Supervision Using Service Oriented Architecture*, 32nd IEEE Conference on Industrial Electronics, Paris, November 2006, pp. 4552-56.
- [4.9] Baptiste, P., Le Pape, C., Nuijten, W., *Constraint-Based Optimization and Approximation for Job-Shop Scheduling*, SIGMAN Workshop on Intelligent Manufacturing Systems, Montreal, Canada, August 20-25, 1995, pp. 21-27.
- [4.10] Hentenryck, P., *The OPL Optimization Programming Language*, Cambridge.MIT Press, 1999.
- [4.11] Mahnke, W., Leitner, S.H., Damm,M., *OPC Unified Architecture*, Berlin, Springer Press, 2009.
- [4.12] Gîrbea, A., Suci, C., Sisak, F.: *Design and Implementation of a Fully Automated Planner-Scheduler Constraint Satisfaction Problem*, 6th IEEE International Symposium on Applied Computational Intelligence and Informatics, Timişoara, Romania, May 19-21, 2011, pp. 54-59.
- [4.13] Benavides, D., Segura, S., Trinidad, P., Ruiz-Cortes, A., *Using Java CSP Solvers in the Automated Analyses of Feature Models*, Generative and Transformational Techniques in Software Engineering, Braga, Portugal, July 4-8, 2005, pp. 399-408.
- [4.14] Zweben, M., Fox, M., *Intelligent Scheduling*, Burlington, Morgan Kaufman, 1994.
- [4.15] Gîrbea, A., Suci, C., Sisak, F., *Constraint Based Approach for Optimized Planning-Scheduling Problems*, Bulletin of the Transilvania University of Brasov, Series I: Engineering Sciences, 2011, Vol. 4, Issue 2, pp. 123-130.
- [4.16] Târnaucă, B., Puiu, D., Comnac, V., Suci, C., *Modeling a Flexible Manufacturing System Using Reconfigurable Finite Capacity Petri Nets*, 13th International Conference on Optimization of Electrical and Electronic Equipment - OPTIM, Vol. 1-5, 24-26 May 2012, pp. 1079-1084, ISBN: 978-1-4673-1650-7.