2008-07-28

The Development of a Computational Model of Thrombosis in Hemodialysis Catheters

Daniel J. Lattin
Brigham Young University - Provo

THE DEVELOPMENT OF A COMPUTATIONAL MODEL OF THROMBOSIS IN HEMODIALYSIS CATHETERS

by

Daniel J. Lattin

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Chemical Engineering

Brigham Young University

December 2008

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Daniel J. Lattin

This dissertation has been read by each member of the following graduate committee and by a majority vote has been found satisfactory.

Date	Kenneth A. Solen, Chair
Date	Thomas A. Knotts
Date	Sivaprasad Sukavaneshvar

ABSTRACT

THE DEVELOPMENT OF A COMPUTATIONAL MODEL OF THROMBOSIS IN HEMODIALYSIS CATHETERS

Daniel J. Lattin

Department of Chemical Engineering

Master of Science

Thromboembolism (TE) significantly limits the usefulness and safety of blood-contacting devices such as hemodialysis catheters. Computer simulation of TE can provide understanding of the process and can facilitate the design of more effective devices. Previous work conducted at BYU successfully modeled TE in a simple, two-dimensional flow cell design by adding quantitative TE code to a commercial computational fluid dynamics (CFD) package, Fluent. This two-dimensional model predicted thrombus initiation and growth and adjusted flow to accommodate thrombus geometries, but was limited by computational power and unsophisticated meshing techniques. To build upon this work, and take advantage of BYU's new supercomputing system and improvements in automatic meshing software, development of a three-dimensional computational model of thrombosis in three hemodialysis catheters designs was undertaken.

Development of the computer model was beset with challenges associated with limitations in both software and hardware, but those challenges were ultimately overcome as both software and hardware evolved. Eventually, the previous C-based Fluent model was ported to the Fortran-based STAR-CD model successfully. A computer geometry of a blood flow cell matching the geometry used with the previous two-dimensional model was created, and results for that geometry using the new computer compared favorably with the results from the previous model. Catheter geometries were created using computer-aided design (CAD) software and were meshed using auto-meshing software. CFD analysis identified potentially-troublesome flow regimes in the catheter designs that coincided with thrombotic regimes observed in preliminary experiments using those same catheter designs. The TE model is now ready for application to the catheter geometries and for rigorous testing (e.g., grid-independence, in-depth comparison with quantitative experiments, etc.).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 - INTRODUCTION

While much has been done to improve their functionality, blood-contacting devices are limited in their use by thrombosis and thromboembolism. Thrombosis is the aggregation of blood platelets onto the device surface, creating a growth known as a thrombus. Thrombi affect the flow of the blood and can shear off into the blood stream in a phenomenon known as thromboembolism. These free-flowing emboli present a severe risk to patients using medical devices as they can occlude downstream blood vessels, causing serious health complications or even death. It is difficult to experimentally assess device designs for thrombosis, as it requires an expensive procedure involving extensive animal testing. Despite rigorous testing to improve designs, however, it is generally acknowledged that thromboembolism regularly occurs while using blood-contacting devices.[1]

Hemodialysis catheters, which provide access to the blood of patients undergoing hemodialysis, are among those devices known to exhibit thromboembolism. In order to minimize the risk associated with thromboembolism, catheters are inserted in the venous side of the vascular system. In this way, the body is able to naturally filter the blood of emboli via the pulmonary system. However, dialysis patients and others using catheters often already have severely compromised or weakened immune systems, increasing the risk of complications due to thromboembolism. Because of these restrictions, patients using catheters must be closely monitored for thrombosis, with those most susceptible often needing alterations to their treatment plan, such as systemic anticoagulation and/or anti-platelet therapy.[2] Additionally, thrombolytic procedures are used to break apart or destroy thrombus formations that have built up over time in catheters.[3]

An important tool in the effort to minimize thromboembolism in medical devices is computational modeling. Because the formation of thrombi is influenced by flow patterns, computational fluid dynamic (CFD) software can be used as a foundation upon which to develop a predictive model of thromboembolism. Goodman et al. developed such a model within a commercial CFD software package, Fluent, and successfully compared thromboembolism predicted for a simple flow cell with experimental thromboembolism in such a flow cell.[4] However, extension of the model to geometries with more clinical relevance was prevented at that time by the need for faster supercomputers (with the supercomputers then available, computation times already extended over several days) and the ability to create computational meshes for more complex geometries (at that time, Fluent required structured meshing which was accomplished by the user manually generating all mesh points, as is discussed below).

While Goodman's work was an important step towards a comprehensive model, there have been significant technological improvements since his research, with both an increase in the computational power of BYU's supercomputers as well as enhanced meshing capabilities. At the onset of this research, it was therefore desired to take advantage of these improvements and develop a three-dimensional computational model of thrombosis in three hemodialysis catheter designs: Mahurkar, Decathlon, and Ash. As the research progressed, the majority of time was spent overcoming obstacles discussed in this thesis, and the full realization of this goal was not possible. The purpose of this research, therefore, was to lay a foundation upon which future researchers can build in applying the thrombosis model to the catheter geometries.

CHAPTER 2 - BACKGROUND

2.1 - Blood-Material Interactions

In the design of all medical devices an important consideration is the blood-material interactions associated with the device. Whenever an artificial material comes in contact with blood, the surface acquires a layer of adsorbed blood proteins.[5] These proteins then further interact with the cells and molecules in the blood, especially platelets, and dangerous thrombus formations can grow. As a thrombus increases in size, the shear force exerted on the thrombus can eventually exceed the adhesive force keeping it attached to the surface, causing the growth to embolize. Figure 2.1 depicts this process.

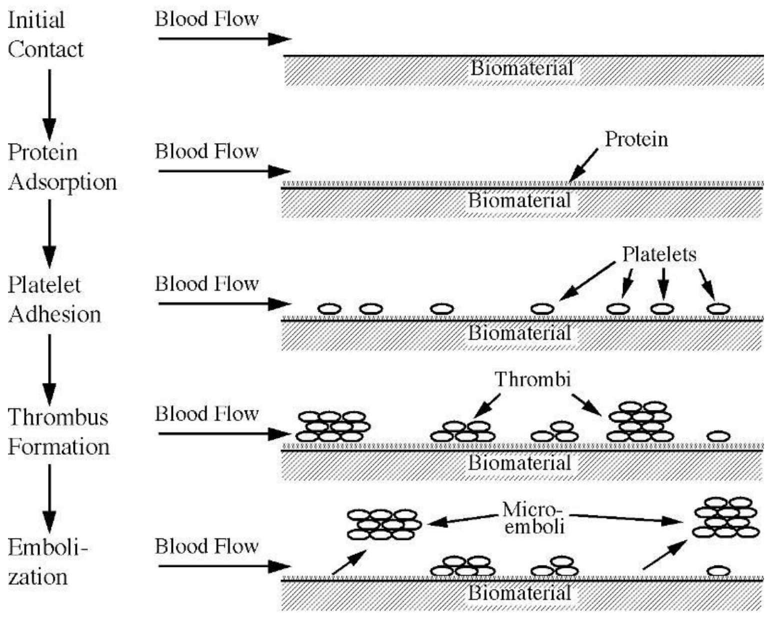

Figure 2.1. Process of thrombosis and embolization.

Platelets are non-nucleated cells comprising only about 0.23% of total blood volume, and normally function in the body to form an initial plug to repair damaged or lacerated blood vessels.[6] Platelets can become activated, either by shear forces or by agonists in the blood, causing them to be more adhesive. In the case of acute thrombosis, thrombi are largely composed of both activated and unactivated platelets.[7]

Flow conditions greatly influence the formation of thrombi for a variety of reasons. The rate of transport of cells and proteins to the surface, both by diffusion and convection, is governed by flow rate. Additionally, shear rates influence the initial attachment of platelets to artificial surfaces as well as the rate of activation of platelets.[8] Unfortunately, according to Hanson, "the relationships between material surface properties… and the design of blood-contacting cardiovascular devices are not well understood… because the dynamics of protein-cell-surface interactions under flow are extraordinarily complex."[9]

2.2 - Computational Modeling

Because of the complexity of blood-material interactions, as well as their dependency on flow conditions, computational fluid dynamics can be used as a platform for modeling blood-material behavior. In recent years, CFD has taken on a greater role in the design of artificial organs and medical devices.[10] CFD can quickly demonstrate the effects of design alterations on blood flow, along with blood-material interactions added to the CFD model, thus significantly reducing the costs, time, and risks involved in developing new medical device designs. While such computational simulations can be of great benefit to the design stages, it is still important to validate the predicted results.

Although several CFD software packages exist that predict flow patterns with great success, STAR-CD was chosen for several reasons. Its user-friendly graphical user interface (GUI) allows for complex geometries to be modeled easily, and its automated meshing software provides for sophisticated, unstructured meshing of those geometries. Additionally, STAR-CD has excellent post-processing tools that can be used after the solution is found to graphically display the solution. A summary of utilities available in the STAR-CD package is found in Table 2.1.

Table 2.1. Summary of utilities available in STAR-CD package.

Name	Function
STAR	Calculation of the CFD solution.
Pro-STAR	GUI-based pre-processing. Prepares model for calculation in STAR.
Pro-am	Automated meshing.
STAR-Design	Basic CAD. Includes a simple CFD solver.

CFD uses numerical techniques to governing mass- and momentum-conservation equations. For general compressible and incompressible fluid flow, STAR-CD solves total mass (Eq. 2.1), momentum (Eq. 2.2), and species mass (Eq. 2.3) conservation equations simultaneously using a finite-volume approach. The equations used, expressed in Cartesian tensor notation, are as follows: [11]

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho \cdot u_j\right) = s_m \tag{2.1}$$

$$\frac{\partial \rho \cdot u_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho \cdot u_j \cdot u_{ij} - \tau_{ij}\right) = -\frac{\partial p}{\partial x_i} + s_i \tag{2.2}$$

$$\frac{\partial}{\partial t}\left(\rho \cdot Y_k\right) + \frac{\partial}{\partial x_j}\left(\rho \cdot u_j \cdot Y_k\right) + F_{k,j} = s_k \tag{2.3}$$

where,

t = time
x_i = Cartesian coordinate
u_i = absolute fluid velocity component in direction x_i
p = piezometric pressure = $p_s - \rho_o g_m x_m$ where p_s is static pressure, ρ_o is reference density, g_m are gravitational acceleration components and the x_m are coordinates relative to a datum where ρ_o is defined
ρ = density
τ_{ij} = stress tensor components
s_m = mass source
s_i = momentum source components
s_k = mass source of constituent k
Y_k = mass fraction of constituent k
$F_{k,j}$ = Diffusional flux component.

There are two solution methods utilized by STAR-CD to simplify these differential equations into algebraic relationships. The first is the Pressure Implicit with Splitting of Operators (PISO) algorithm and the second is the Semi-Implicit Method for Pressure Linked Operators (SIMPLE). Both algorithms employ forms of the predictor-corrector strategy, in which the flow equations are decoupled from each other by predicting the next value and then refined in an ensuing corrector stage through an iterative process. The SIMPLE algorithm differs from the PISO in that it employs only one corrector stage.[12]

Solving these algorithms is made possible by dividing the flow geometry into many individual points, or nodes. These nodes form a computational mesh used by the CFD software in predicting such things as velocity fields, species transport by diffusion and convection, and shear forces. In order to obtain an accurate solution, the nodes in the computational mesh must be sufficiently close to each other to justify approximating the differential relationship algebraically. One of the significant obstacles in computational hemodynamic analyses is the generation of an accurate computational mesh.[13]

There are two types of computational meshes: structured and unstructured. A structured mesh is one in which equally sized, regular hexahedrons are used. These are often created by the user specifying the location of each node in the mesh, and can be both tedious and time-consuming to create. Unstructured meshes consist of irregular tetrahedrons, hexahedrons, or both. Unstructured meshes allow for greater flexibility in modeling complex geometries. They are typically created using automated meshing software. Additionally, meshing software can create meshes that have variable cell sizes, to give accurate resolution in regions of importance while minimizing the total number of cells required. These variable-sized cells consist of regular hexahedrons.

2.3 - Previous CFD Models

There are not many examples in the literature of computational thrombosis models. However, an early attempt to use CFD to model the complex blood-material interactions was made by Sorenson et al., who developed a comprehensive two-dimensional model for platelet deposition in flowing blood. Their study included both shear-induced and agonist platelet activation, platelet transport, platelet-platelet interactions, and platelet-surface interactions. While these efforts represented an important step forward in the modeling of thromboembolism, it was limited to initial conditions and did not model thrombus growth over time.[14]

Other researchers have taken a CFD-only approach to predicting thrombosis in hemodialysis catheters. One such example is Mareels et al. They developed a three-dimensional CFD model of a dual-lumen Niagara catheter in a simulated venous environment. Several performance parameters, such as pressure drop and shear rates, were analyzed.[15] In a follow-up study, Mareels validated the CFD with Particle Image

Velocimetry by comparing simulated and measured velocities and shear strains.[16] Using a steady-state CFD calculation, these studies identified regions of high shear stress and/or stagnation, areas where thrombosis is likely to occur. Studies such as these have become fairly common, but are limited to a CFD analysis only, and do not incorporate thrombosis into the computational model nor dynamically change the flow patterns and solutions based on the presence of thrombi.

Goodman et al. were able to develop a computational model for thrombosis in a simple flow-cell geometry. Like Sorenson, Goodman effectively incorporated shear- and agonist-induced platelet activation, species transport, and both platelet-platelet and platelet-surface interactions. However, he was also able to predict the growth of thrombi over time and to adjust the flow patterns to accommodate the presence of the thrombi. He also incorporated the embolization of single adherent platelets based on experimental measurements. His predictions were validated by dynamic experimental measurements of thrombus initiation, growth, and embolization. Interactive changes in a transient problem represent one of those most difficult challenges in modeling, and Goodman accomplished this by increasing the viscosity of computational grid cells by several orders of magnitude when they became a thrombus. This, however, increases the number of computations required to converge to an accurate solution. Because of this, and due to existing hardware limitations, the model was restricted to simple two-dimensional geometries rather than those of clinically-relevant devices. Additionally, the meshing capabilities then available limited the model to a regular, structured mesh of uniform cell size.[4]

CHAPTER 3 - OBJECTIVES

As mentioned in the Introduction, the original aim of this research was to build upon Goodman’s previous two-dimensional flow-cell model and develop a thrombosis model in clinically-relevant, three-dimensional hemodialysis catheters. Goodman’s model was developed using the CFD software Fluent. At the time, Fluent was limited in its meshing capabilities. Another CFD program, CD-adapco’s STAR-CD, advertised vast improvements in automated meshing as well as post-processing capabilities (animation of transient results, etc.). These promised improvements, coupled with concerns over the availability of Fluent licenses, prompted the decision to develop the catheter thrombosis model using STAR-CD. However, as research progressed, the majority of time developing the model was spent overcoming limitations associated with the STAR-CD software (see Chapter 6). Research was impeded to such a degree that full application of the thrombosis model to catheter geometries was prevented. As a result, the research objectives were modified in such a way as to prepare a framework for future researchers to easily apply the thrombosis model to the catheters. Specifically, the research objectives were:

1) Creating three-dimensional meshes of the catheters using computer-aided design (CAD) and automated meshing software that can be used by future researchers.
2) Porting Goodman’s C-based Fluent model into a Fortran-based STAR-CD model.
3) Overcoming issues related to pushing the limits of the software.

4) Determining the feasibility of three-dimensional computations with current hardware limitations.

5) Performing a preliminary comparison of CFD and experimental results in the hemodialysis catheters. These results could be used by future researchers as an early benchmark for the performance of the thrombosis model.

CHAPTER 4 - CREATING THE MESH

4.1 - Designing the Geometry

The first step in creating the computational mesh is designing the geometry of the catheter. The three different designs selected are the Mahurkar, Decathlon, and Ash hemodialysis catheters. These three catheters were chosen because they have aspects to their geometry that are representative of a range of catheter designs. The Mahurkar design has a large central intake port with small side ports that help prevent obstruction. The Decathlon catheter has a split-tip design, while the Ash design has a tapered outlet tip and side holes for the inlet. CAD software, such as CD-adapco's STAR-Design and Autodesk's AutoCAD, was used to create geometric representations of all three catheters (See Figures 4.1, 4.2, and 4.3, respectively). These programs allow for the complex geometries found in catheters to be created with relative ease.

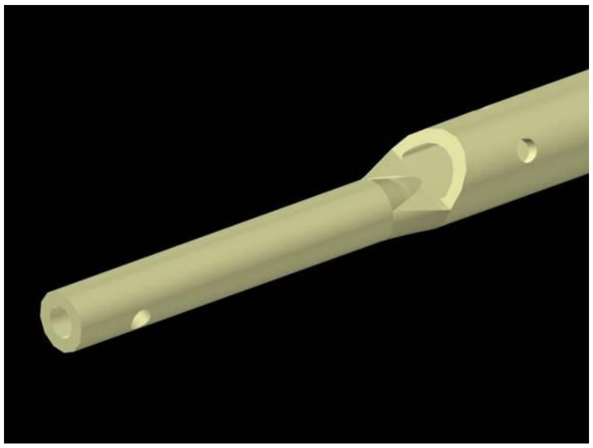
Figure 4.1. Insertion region for the Mahurkar catheter design.

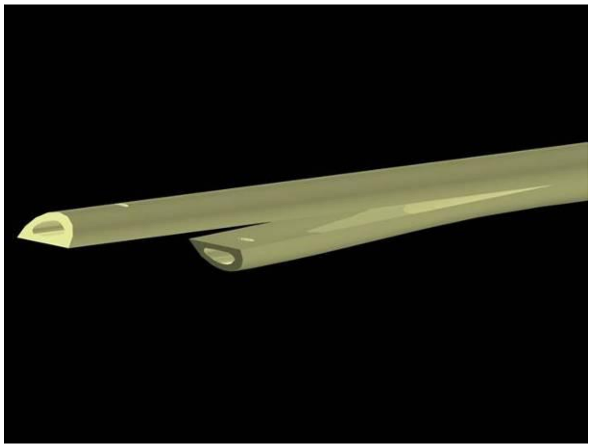

Figure 4.2. Insertion region for Decathlon (split lumen) catheter design.

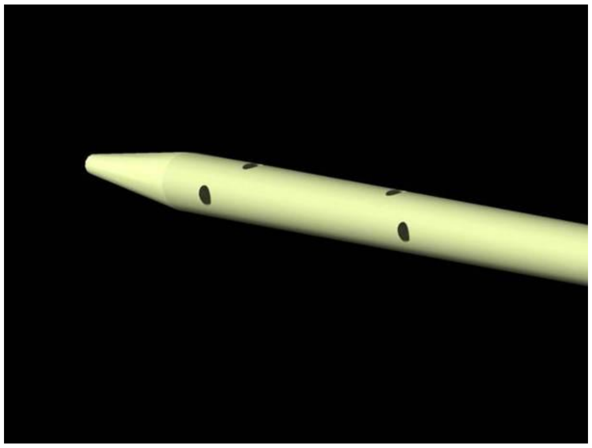

Figure 4.3. Insertion region for Ash (side hole) catheter design.

4.2 - Mesh Generation

Once the geometry was created, it was necessary to represent the flow regions of the catheter with a computational mesh. This was done with STAR-CD's automatic meshing utility, pro-am. By importing the geometries previously created, pro-am is able to generate computational meshes of variable cell size. As described above, Goodman was restricted to a uniform grid cell size. One of the significant advantages of current technology is that the grid size can be varied within a mesh, so that the edge length of a cell is smaller near regions of interest, such as a catheter wall. In the interior core of the model, edge lengths can be increased. In this way, the total number of cells required to mesh a specific geometry can be significantly reduced, in turn reducing the computational time and memory required for a solution.

The edge length required for grid-independence in Goodman's two-dimensional model was 0.5 microns, which, for his simple flow cell geometry, translated to approximately 2.3 million cells.[4] Uniform meshes with edge lengths of approximately 150 microns, or approximately 3.5 million cells, have been created for the catheter meshes for the purposes of CFD-only calculations. However, it is anticipated that a much smaller edge length, closer to 0.5 microns, would be required for the three-dimensional catheter thrombosis models. With a uniform mesh, this would require hundreds of millions of cells, if not more. The current supercomputing technology would not be able to handle creating the computational mesh, let alone calculating the solution. Therefore, creating variable-sized computational meshes using pro-am is necessary to developing the thrombosis model. Further discussion of the feasibility of generating these meshes is addressed in Chapter 7.

CHAPTER 5 - PORTING THE MODEL INTO STAR-CD

5.1 - Defining Basic Model Properties

The first step in porting the model was to create a model using STAR-CD's pre-processing utility, pro-STAR. Pro-STAR provides a user-friendly GUI for establishing model parameters. Since the flow characteristics would be changing as thrombus developed, the model was defined to be transient. A transient solution iterates at each time step to solve the various conservation equations, rather than converging to a single, steady-state value. The fluid properties of blood, such as density, viscosity, etc., were then defined. Scalar species were then created for activated and unactivated platelets, as well as the various agonist species involved in platelet activation. The initial mass fraction, molecular weight, and effective diffusivity for each scalar species is listed in Table 5.1. Boundary and initial conditions were also defined for the scalar species and fluid flow.

Table 5.1. Species parameters.[4]

	Initial Mass Fraction	Molecular Weight (g/mol)	Thermal Diffusivity, D_b (cm^2/s)
Unactivated Platelet (Pl)	0.0006	$1.28x10^{12}$	$1.58x10^{-9}$
Activated Platelet (aPl)	0.000006	$1.28x10^{12}$	$1.58x10^{-9}$
Adenosine Diphosphate (ADP)	–	424.4	$2.57x10^{-6}$
Thromboxane (TxA_2)	–	352.5	$2.14x10^{-6}$
Prothrombin (pT)	0.000075	72000	$3.32x10^{-7}$
Thrombin (T)	–	36600	$4.16x10^{-7}$
Antithrombin III (aT)	0.000167	62000	$3.49x10^{-7}$

5.2 - Creating User Subroutines

Simply defining the problem parameters in pro-STAR would have allowed STAR-CD to solve only the CFD portion of the model. In order to include thrombosis, it was necessary to include user subroutines. STAR-CD allows users to create subroutines that can dynamically modify problem parameters as it solves. Goodman similarly used Fluent macros in order to accomplish this. He used macros that defined the inlet profile, viscosity, source terms, surface reaction kinetics, and adjusted certain variables. All of his programming was in the C language. However, STAR-CD uses the FORTRAN programming language. Also, the data structure used by STAR-CD is very different from that used in Fluent. This necessitated porting the code he developed into the new language and data format. To do this, seven user subroutines were created. Table 5.2 contains a list of the subroutines and a description of their function. The full code of each subroutine is included in the appendix.

Table 5.2. Description of user subroutines.

Name	Function
initfi.f	Initializes bulk fluid with parabolic flow profile.
bcdefi.f	Establishes parabolic flow profile at inlet boundary.
diffus.f	Creates shear rate dependence for diffusivity of larger species.
sorsca.f	Calculates source terms for scalar species.
scalfn.f	Calculates the maximum shear rate in each cell at each time step.
vismol.f	Increases viscosity of a "thrombus" cell.
posdat.f	Calculates platelet surface fluxes and thereby determines whether a cell has become a thrombus.

5.2.1 - Surface Reactions

The first significant hurdle faced in porting the code was the handling of surface reactions. Platelet-surface interactions are an important aspect to thrombosis. Fluent had built-in surface reactions, but STAR-CD does not, making it necessary to determine a strategy for including these important blood-material interactions. It was determined to include these reactions as part of the source terms in the computational cells next to walls. This was done by calculating the mass flux to or from the wall due to the reaction and then converting it into a volumetric source term using the ratio of the area of a cell face to the total cell volume for a cell at the wall. A summary of the surface reaction terms, based on Sorenson's work, is given in Table 5.3.[4, 14]

Table 5.3. Surface reaction terms.

Species	Term
Pl (platelet/$m^2 \cdot s$)	$-S \cdot k_{ps}[Pl]_{surface} + F_{emb} \cdot M_u$
aPl (platelet/$m^2 \cdot s$)	$-S \cdot k_{as}[aPl]_{surface} - M_a/M_\infty \cdot k_t \cdot [aPl]$
ADP (mol/$m^2 \cdot s$)	–
TxA_2 (mol/$m^2 \cdot s$)	$M_a(x,t) \cdot s_t$
pT (mol/$m^2 \cdot s$)	$-(\varphi_{rt} \cdot M_u + \varphi_{at} \cdot M_a) \cdot [pT]$
T (mol/$m^2 \cdot s$)	$(\varphi_{rt} \cdot M_u + \varphi_{at} \cdot M_a) \cdot [pT]$
aT (mol/$m^2 \cdot s$)	–

In these reaction terms,

S =	fraction of surface available for platelet-surface adhesion
F_{emb} =	fraction of surface-adherent platelets that embolize during each time step
M_u =	local concentration of unactivated platelets on the surface
M_a =	local concentration of activated platelets on the surface
M_∞ =	local concentration of activated platelets on the surface (6.0×10^{10} plt/m^2)
k_{ps} =	rate constant for unactivated platelets adhering to the surface (2.5×10^{-6} m/s)
k_{as} =	rate constant for activated platelets adhering to the surface (3.5×10^{-3} m/s)
k_t =	rate constant for platelets adhering to other platelets (3.5×10^{-3} m/s)
s_t =	rate of synthesis of ADP by platelets (9.5×10^{-12} nmol/plt·s)
φ_{at} =	rate of thrombin generation from prothrombin on activated platelets (6.50×10^{-10} U/plt·s·μM pT)
φ_{rt} =	rate of thrombin generation from prothrombin on unactivated platelets (3.69×10^{-9} U/plt·s·μM pT)

5.2.2 - Source Terms

As platelets become activated, they are more likely to adhere to biomaterials. This activation can take place either by shear activation, as discussed below, or through interaction with agonist species. These interactions can be represented through source terms, or bulk reaction terms. Like the surface reaction terms, they are based on Sorenson's work, and are given below in Table 5.4.[4, 14]

Table 5.4. Scalar source terms.

Species	Term
Pl (platelet/mL·s)	$-k_{pa}[\mathrm{Pl}]$
aPl (platelet/mL·s)	$+k_{pa}[\mathrm{Pl}]$
ADP (mol/m^2·s)	$+\lambda \cdot k_{pa}[\mathrm{Pl}]$
TxA$_2$ (mol/m^2·s)	$+s_t[\mathrm{aPl}]-k_i[\mathrm{TxA_2}]$
pT (mol/m^2·s)	$-[\mathrm{pT}](\varphi_{at}[\mathrm{aPl}]+\varphi_{rt}[\mathrm{Pl}])$
T (mol/m^2·s)	$[\mathrm{pT}](\varphi_{at}[\mathrm{aPl}]+\varphi_{rt}[\mathrm{Pl}])-\Gamma[\mathrm{T}]$
aT (mol/m^2·s)	$-\ \Gamma[\mathrm{T}]$

In these equations,

k_{pa} = rate constant for platelet activation (see below)
λ = amount of ADP released per activated platelet ($2.48\mathrm{x}10^{-8}$ nmol/plt)
Γ = rate constant for heparin-catalyzed inhibition of thrombin by antithrombin III, from Griffith's template model:[4, 17]

$$\Gamma = \frac{k_1 \cdot [H] \cdot [aT]}{\alpha \cdot K_{aT} \cdot K_T + \alpha \cdot K_{aT} \cdot [T] + \alpha \cdot K_T \cdot [aT] + [aT] \cdot [T]} \tag{5.1}$$

where,

k_1 = first-order rate constant ($13.33\ \mathrm{s}^{-1}$)
[H] = concentration of heparin (1.0 U/mL)
α = factor simulating the change in affinity of heparin for aT when it is bound to T or for T when it is bound to aT (set to 1.0)
K_{aT} = dissociation constant for the heparin/aT complex (0.100 μM)
K_T = dissociation constant for the heparin/T complex (0.035 μM)

Sorenson developed k_{pa} as a linear rate equation with an activation threshold:[4, 14]

$$k_{pa} = \begin{cases} \dfrac{0}{t_{act}}, \Omega < 1.0 \\ \dfrac{\Omega}{t_{act}}, \Omega \geq 1.0 \end{cases} \tag{5.2}$$

where Ω was the activation function:[4, 14]

$$\Omega = \sum_{j=1}^{n_a} w_j \cdot \frac{a_j}{a_{j,crit}} \tag{5.3}$$

In these equations,

n_a = total number of agonists in the model
a_j = concentration of the j^{th} agonist
$a_{j,crit}$ = threshold concentration of that agonist for platelet activation (see below)
w_j = agonist-specific weight to mimic the differential effects of strong and weak agonists on the activation reaction (see below)
t_{act} = characteristic time constant for platelet activation (1.0 sec.)

In this way, platelet activation occurs only once critical levels have been reached for multiple agonists. The critical concentrations and agonist-specific weights for those species involved in platelet activation is given in Table 5.5.[4]

Table 5.5. Agonist concentrations needed for platelet activation.

Species	Conc. Needed for Activation	Weight (w_j)
ADP	2.00 μM	1*
TxA_2	0.6 μM	3.3
T	0.01 U/mL	220

*Weights were calculated by dividing the concentration required for activation of the least stimulatory species (ADP) by the concentration required for activation of the species in question.

5.2.3 - Shear Activation of Platelets

Platelets can also be activated through the shear forces of the blood. This takes place in a time-dependent manner, once unactivated platelets have adhered to a surface or to a thrombus. Equation 5.4 shows the relationship between the shear stress, τ, and the time required for platelet activation, as described by Hellum.[4, 18]

$$t = 4.0 \times 10^{-6} \cdot \tau^{-2.3} \tag{5.4}$$

If agonist-induced activation had not occurred by this time, the platelets were considered activated. This model is based solely on local shear stress on surfaces. However, recent research by Nobili et al. indicates that platelets in the bulk fluid can be activated by lower shear stresses than indicated by Hellums if those stresses are applied over time and in certain patterns.[19] Future development of the model may benefit from examining how much the catheter results change by adding this method of shear activation to the model.

5.2.4 - Shear-Dependence of Diffusivity

The diffusivities listed above in Table 5.1 are thermal, or Brownian, diffusivities. As such, they do not take into account the diffusion-enhancing effect of red cell motion in flowing blood. This motion creates an effective diffusivity for the platelets, as well as the large scalar species (pT, T, aT). For these species, equation 5.5 was applied, following Wootton et al.[4, 20]

$$D_e = D_b + \alpha \cdot \gamma \tag{5.5}$$

where,

D_e = effective diffusivity
D_b = thermal (Brownian) diffusivity
α = scaling factor ($6.0\text{x}10^{-10}$ cm^2)
γ = maximum local shear rate

5.2.5 - Viscosity of a Thrombus

In order to simulate the creation of a thrombus, Goodman increased the viscosity of a “thrombus” computational cell by a factor of approximately 25,000. This effectively created a solid region. Such a large step in viscosity is difficult for the CFD solver to account for, increasing the number of iterations required to converge each time step as well as necessitating a smaller edge length. In order to reduce the solution time, the STAR-CD model introduced a step in viscosity of only 250 times that of the blood. This smaller step in viscosity may also be more representative of thrombus behavior, as thrombus is living tissue, and as such may allow for very limited flow through it. While determining the “effective viscosity” of a thrombus is outside the scope of this project, future research might benefit from an investigation of the effect of varying levels of thrombus viscosity on the model results, as well as attempting to experimentally determine thrombus viscosity.

5.3 - Comparison with Goodman’s Results

In order to determine the validity of porting the model from Fluent to STAR-CD, it was necessary to compare the model’s results with those obtained by Goodman. While it was initially thought that this would be an easy step in the process, significant challenges were faced in developing the model in STAR-CD (see Chapter 6). The majority of time spent on this project was dedicated to addressing issues created by pushing the limits of STAR-CD. However, a two-dimensional mesh of the simple flow cell, with a uniform 0.5 micron cell edge length, was created. The same time step size (0.05 seconds) and run time (30 seconds) used by Goodman were also applied to the model, and a solution calculated. Using the PISO algorithm, a stable solution was never obtained. However, by implementing the SIMPLE algorithm, it was possible for the solution to converge.

The flow cell geometry consists of a cylindrical flow cell 1 mm in length (ID = 580 μm) with an abrupt contraction (ID = 280 μm), also 1 mm in length, followed by an expansion to the original diameter. Thrombus growth generally occurs in two main regions, the leading and trailing edges of the contraction.[4] The STAR-CD model replicates the Fluent model favorably. Table 5.6 shows the area occupied by thrombus in both of these regions, as calculated by the Fluent and STAR-CD models.

Table 5.6. Summary of calculated thrombus areas.

	Fluent Model		STAR-CD Model	
	Leading Edge	Trailing Edge	Leading Edge	Trailing Edge
Area (μm^2)	52.96	51.97	121.2	152.8

While the STAR-CD model generates more thrombus than the Fluent model, it is still on the same order of magnitude. A possible explanation for the discrepancy between the two models is there were a few mathematical errors found in Goodman's model. These errors, while minor, have been corrected in the STAR-CD model, and could account for the differences in thrombus growth. Also, the differences in surface reaction terms and viscosity, as discussed above, may slightly influenced the solution. As seen in Figures 5.1 and 5.2, while there are some differences in the amount and shape of the thrombus growth, the growth occurs in the same location in both models.

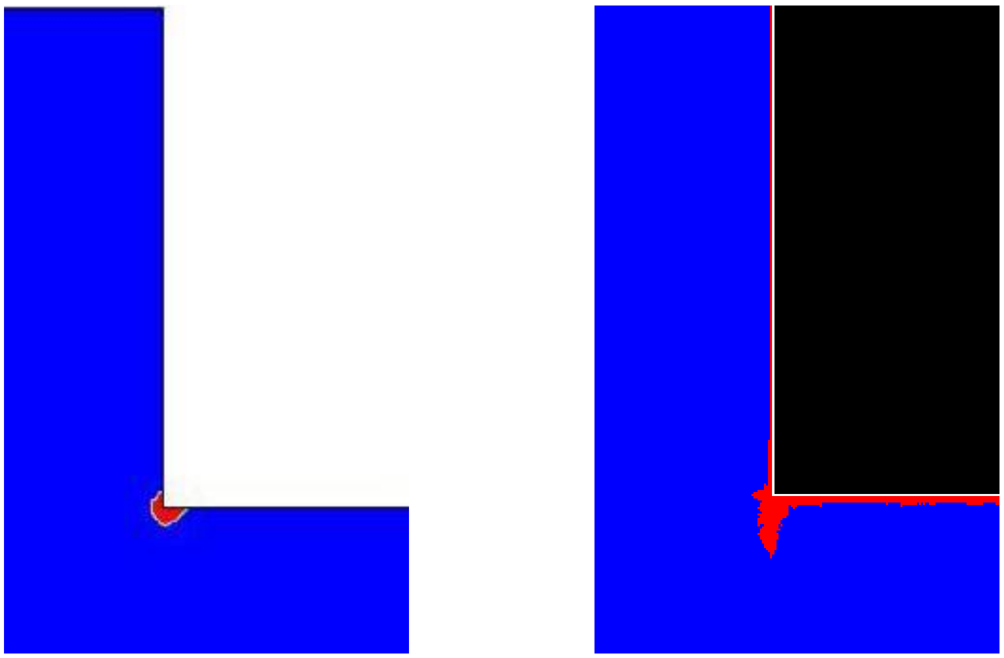

Figure 5.1. Thrombus growth on leading edge of contraction. The Fluent model is on the left and the STAR-CD model is on the right. (Run time = 30 sec., time step = 0.05 sec, grid edge length = 0.5 µm)

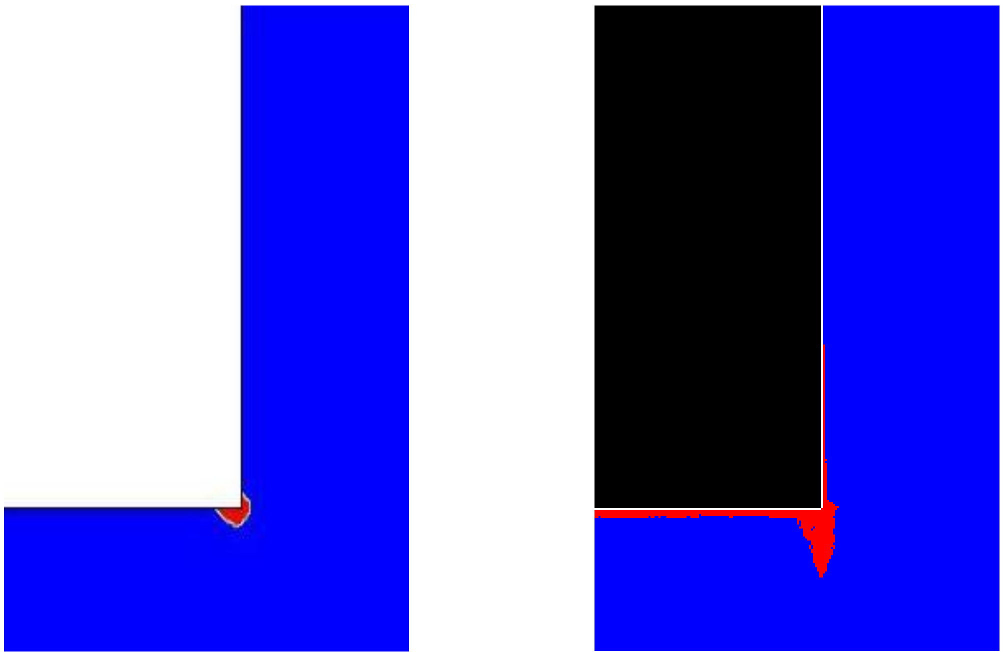

Figure 5.2. Thrombus growth on trailing edge of contraction. The Fluent model is on the left and the STAR-CD model is on the right. (Run time = 30 sec., time step = 0.05 sec, grid edge length = 0.5 µm)

CHAPTER 6 - OVERCOMING CHALLENGES OF PUSHING THE SOFTWARE LIMITS

CD-adapco advertises the feature of Star-CD in which the user can add user-defined subroutines to supplement the CFD calculations, as described above. However, interactions with CD-adapco have given the impression that the company focuses most of their energy on building as many packaged configurations as possible, customizable through the GUI. In this way the company attempts to minimize the need for user-created subroutines. As a result, documentation and functionality associated with the use of custom subroutines is incomplete. Furthermore, obtaining information about such matters from the customer-support arm of the company is extremely slow and problematic.

6.1 - User-Defined Variables

One example illustrating the challenge of pushing the software was illustrated in the problem of handling the various user-defined variables Goodman used in his model. Besides the concentrations of various agonist species, an additional set of variables permitted the identification and tracking of computational cells that had reached certain conditions, such as whether the cell had become thrombus or neighbored thrombus, or whether the platelet activation time had been reached in a particular cell. Goodman accomplished this using built-in Fluent “User-Defined Memory Allocations.” Initially personnel at CD-adapco recommended using dynamically-sized arrays to store these variables in STAR-CD. However, it was discovered after significant testing (and months of time) that the handling of memory for the arrays by STAR-CD was flawed and prevented successful execution of the user subroutine code by this method.

Eventually, after numerous and persistent inquiries to CD-adapco, finally resulting in personnel at the company testing the user subroutines, the company acknowledged that the code was not performing adequately using their recommended strategy (although the cause for the malfunction was never identified). At that point, they recommended that we create passive scalar species in the pro-STAR GUI for each necessary variable. These passive species are not solved for in the transport equations, and thus do not influence the solution during each time step. However, they are used at the end of each time step to record thrombus location and growth, and to then modify the flow regime in the ensuing time step through the user subroutines. Eventually, this second strategy successfully allowed us to accomplish this necessary task.

6.2 - STAR-CD Internal Variables

While developing the code for the user subroutines, it was often necessary to use variables, functions, and procedures internal to the STAR-CD proprietary code in order to perform tasks such as looping through computational cells, identifying neighbor cells, etc. These items were often mentioned in the user subroutine documentation without any description of their functionality. Full documentation of these variables and procedures was either difficult to locate within the directory structure of STAR-CD or non-existent. The majority of the time spent developing the model was spent determining how to use these various tools in order to accomplish necessary tasks, often through a tedious, trial-

and-error approach. Documentation of the knowledge gained about the various internal variables and functions is included in the appendix of this thesis.

6.3 - Upgrading Versions of STAR-CD

When development of the model was initially undertaken, the most current version of STAR-CD was version 3.20. As model development continued and challenges presented themselves, solutions were often not available using the (then) current version of the software. What seemed to be problems with the user subroutines or model parameters were often problems inherent to the STAR-CD coding that were fixed in later versions. It became clear that many of the features advertised by CD-adapco were not fully functional in the latest version, or did not perform at a level necessary for successful solution of the thrombosis model. Development was often stalled until a newer version of the software was released. Additionally, the current version of STAR-CD, version 4.06, required porting the user coding from FORTRAN 77 to FORTRAN 90. This necessitated a significant re-write of essential user subroutines, further extending the project's time.

6.4 - Summary of Software Challenges and Solutions

The above-mentioned cases are but a few of numerous time-consuming obstacles encountered in the development of this model. Each obstacle had at least one, often all, of the following characteristics:

- Inadequate documentation.
- Very slow response from the customer-support arm of CD-adapco.
- Failure of the remedy/procedure provided by CD-adapco to work.

Unfortunately, these technological challenges associated with the software typically were severe enough that the model could not solve to successful completion. Much, if not the majority, of the time spent developing the model was spent in addressing these issues rather than debugging and trouble-shooting the user subroutine code. The current thrombosis model is able to run and return results for the two-dimensional flow cell geometry. Therefore, the software challenges that have prevented further development of the model have been overcome. From this point forward, it should be possible for future researchers to apply the model to three-dimensional catheter geometries.

CHAPTER 7 - DETERMINING FEASIBILITY OF THREE-DIMENSIONAL COMPUTATIONS

While there has been significant improvement in the computational capabilities of the supercomputer hardware since Goodman's work, there is still a limit to how much data can be processed in a given amount of time. This limit is a function of both the processor speeds and available memory on the supercomputing system. As the number of nodes in a computational mesh increases, it requires more computing power both to generate the mesh and to solve the CFD calculations. Because the simple flow-cell used in Goodman's model had radial symmetry, he was able to represent the flow-cell with a two-dimensional geometry. Although this significantly reduced the amount of computation time necessary, it still required several days for the supercomputer system to complete the calculations.

Since most, if not all, clinically relevant devices do not have convenient symmetry, it is necessary to represent these devices using three-dimensional meshes to accurately predict thrombosis. Obviously, meshes in three dimensions require many more nodes to represent the geometry. The number of computational cells required to achieve appropriate resolution, often tens of millions or more, may limit the system's ability to complete the solution in a reasonable amount of time. In order to create an accurate model of thrombosis in catheter designs, it must first be determined if the current hardware technology is capable of performing the necessary tasks.

7.1 - Generating Computational Meshes

Because STAR-CD can be parallelized between dozens of processors on the supercomputing system, sharing both available memory and processor power, solving the model is not the limiting step in the process. Instead, it is the actual generation of the computational mesh that taxes the system's resources. The current version of STAR-CD does not permit automated meshing to be parallelized. While the process is not computationally intensive, it is very memory-intensive. To determine the anticipated amount of memory required for a catheter model, a series of tests were conducted on a simple three-dimensional flow cell model. The dimensions of the flow cell were the same as the flow cell used in Goodman's model, except it was half of the length. Pro-am was used to automatically generate a mesh with variable edge lengths. Table 7.1 shows the memory required to mesh the geometry and number of cells generated as the minimum edge length decreases.

Table 7.1. Memory requirements for automated meshing.

Minimum Edge Length (microns)	Number of Cells (millions)	Memory Required (GB)
15	0.089	0.136
10	0.215	0.216
7.5	0.419	0.314
5	1.081	0.619
2.5	4.574	2.291
1	30.622	14.149

As the edge length decreases, both the number of cells and required memory increase according to a power law relationship of the form. There was not a system available to generate a mesh on this test model with a minimum edge length of 0.5

microns. However, by fitting the data to a power law equation of the form $y = a \cdot x^b$, where y is the required memory, x is the edge length, and a and b are both constants, a and b are found to be 12.156 and -1.7454, respectively ($R^2 = 0.9912$). According to this relationship, the memory required to generate a 0.5 micron mesh would be 40.7 GB.

It is unknown whether computational meshes for catheter geometries will require the same resolution as the meshes used by Goodman. If the same minimum edge length of 0.5 microns is required, it is anticipated that generating a computational mesh for catheter geometries would require perhaps hundreds of gigabytes. If, however, the process of automated meshing in pro-am can be parallelized, the memory requirements could be shared over several nodes. Representatives from CD-adapco have said that such a parallelization process is currently being developed. This could address the needs of creating the required computational meshes. Currently, however, this is still a limitation in the technology.

7.2 - Defining Problem Parameters and Calculating a Solution

Once a mesh has been generated, it is necessary to input the various problem parameters into pro-STAR, as described previously in Chapter 5. When a mesh has been created with millions of computational cells, it requires an extensive amount of memory to use the pro-STAR GUI. However, without the GUI, it is difficult to accurately assign boundaries and input other parameters. Even if the meshing process is parallelized, pre-processing with the GUI is not. This is potentially a limitation in developing the thrombosis model in catheters. In order to prepare the model in pro-STAR, it will be necessary to use a workstation with large amounts of available memory. A workstation with 32 GB of memory has been obtained for this purpose, but it is still undetermined whether this amount of memory will be sufficient for catheter meshes with high resolutions.

The actual computation of a solution is not hampered by available memory, but is instead dependent on processing power. Because this aspect of STAR-CD can be run in parallel mode, it is possible to assign dozens of processors to calculate the solution. A computational mesh of the simple flow cell geometry with approximately 783,000 cells was created and an analysis of the time required to complete computations using the PISO algorithm was conducted. Table 7.2 shows the results of this analysis. As can be seen, computational times using 64 processors for this mesh were very reasonable, about 2.25 hours to complete 400 time steps. For fewer processors, the calculations timed out after 24 hours. The solution using 16 processors diverged before finishing calculations.

Table 7.2. Analysis of computational time for 783,000 cell mesh.

Number of Processors	Elapsed Time (hr)	Completed Steps
64	2.25	400
32	4.51	400
16	4.30	380
12	6.08	400
8	9.06	400
4	15.03	400
2	24.01	359
1	24.00	246

Calculations (implementing the SIMPLE algorithm) on the two-dimensional flow cell mesh containing 2.3 million cells and using 64 processors took approximately 12 hours to calculate all 600 time steps. This would imply that for the catheter geometries, several days would be necessary to complete the calculations. This would not be an unreasonable amount of time and should not be considered a limiting factor in the overall process.

7.3 - Feasibility with Current Hardware Limitations

The most significant hardware limitation is available memory. While this does not affect the actual computation of the solution, it does affect pro-am's automated meshing capabilities as well as defining the problem parameters with pro-STAR's GUI. Although meshes have been created for each catheter design for the purpose of CFD-only analysis (discussed below in Chapter 8), because CFD does not require the same grid resolution as the thrombosis model, these meshes were fairly crude, with a uniform edge length of only approximately 150 μm. As a starting point, it is recommended the thrombosis model be applied to these meshes. However, a grid-independence study should be conducted to ensure that the results are valid. In order to minimize the total number of cells, the edge lengths of the meshes in the grid-independence studies should be variable-sized. Even so, it is unknown whether the current memory available will be sufficient to obtain the resolution desired near the walls. In order to further reduce the total number of computational cells and thereby the memory required for generating the meshes and defining the problem parameters, the catheter geometries should be cut across planes of symmetry. In this way, future researchers can maximize the feasibility of applying the model to the catheters given current memory limitations.

CHAPTER 8 - PERFORMING A PRELIMINARY COMPARISON

At the onset of this research, the primary objective was to apply the computational thrombosis model to the catheter geometries and obtain results that could be validated by *in vitro* testing. However, significant time was lost addressing the challenges that arose in developing the STAR-CD model, as discussed in Chapters 6 and 7, and it became necessary to adjust the objectives in such a way as to lay a foundation that future researchers can use in applying the computational model to the catheter geometries. While it is left to future researchers to apply the STAR-CD model to the catheter designs, it was decided to perform a preliminary comparison between a CFD-only analysis of the catheters and *in vitro* experiments. In this preliminary comparison, no quantitative analyses were conducted, but could be anticipated once the computational model has been applied to the catheter geometries.

8.1 - CFD-Only Results

A CFD-only analysis on each of the catheter geometries was performed. These analyses were not intended to be an exhaustive study of hemodynamics in the catheters, but instead a preliminary look at potential locations of thrombus growth in the computational model. Therefore, memory-intensive grid resolution analyses were not conducted. However, by examining the flow patterns, regions of high shear were identified by determining where fluid flow was forced near the walls of the catheters.

These are the areas that should promote thrombus growth in the computational model. For the purposes of this analysis, only the inlet regions of the catheters were modeled.

Figure 8.1 shows the flow pattern for the Ash catheter. The proximal inlet port (left) is first exposed to the blood flow, and has a higher flow rate through it, inducing higher shear rates than seen in the distal inlet port. Because of the sharp change in direction of the flow, as well as the high velocities, it is anticipated that thrombus would grow in several locations, as indicated in the figure.

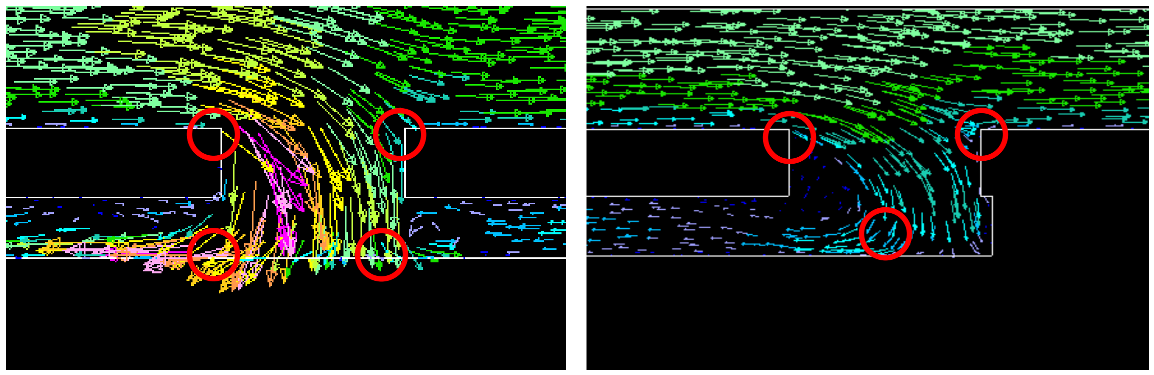

Figure 8.1. Flow pattern for proximal (left) and distal (right) inlet ports of Ash catheter. Regions of potential thrombus growth are highlighted in red.

The flow pattern for the Decathlon and Mahurkar catheters are similar to each other and are shown in Figures 8.2 and 8.3, respectively. The Decathlon catheter has been modeled with the split tips slightly separated from each other. Both of these models have large main inlet ports, with smaller side ports nearby. The Decathlon side ports are directly across from each other, causing turbulent mixing of the inlet streams, whereas the inlet flow through the side port in the Mahurkar design is direct against the opposing wall. Those areas where thrombus growth is anticipated are marked in the figures.

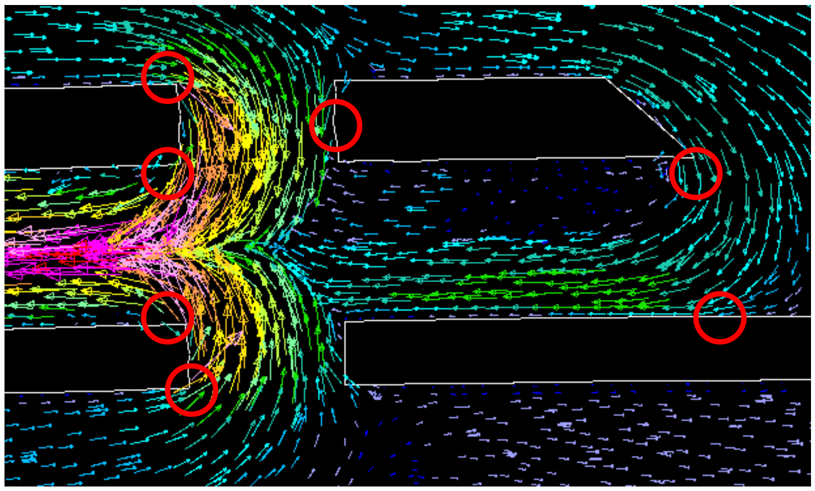

Figure 8.2. Flow pattern for inlet region of Decathlon catheter. Regions of potential thrombus growth are highlighted in red.

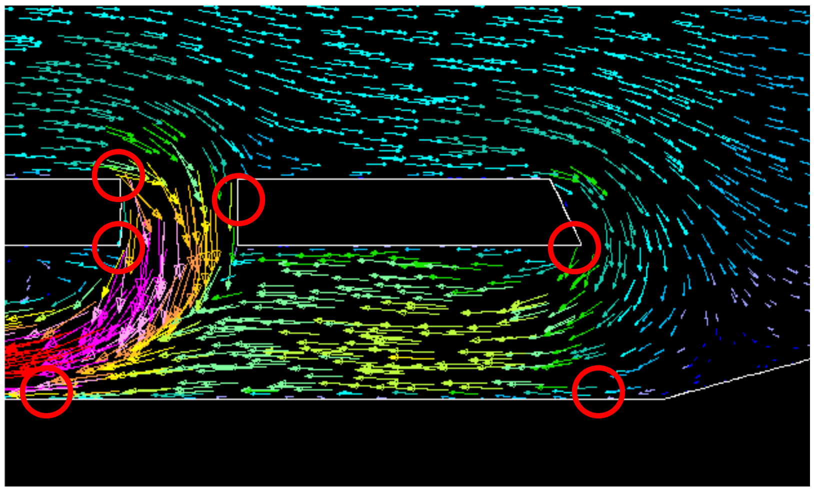

Figure 8.3. Flow pattern for inlet region of Mahurkar catheter. Regions of potential thrombus growth are highlighted in red.

8.2 - Preliminary Experimental Results

While a complete and rigorous experimental validation of the computational model is outside the scope of this project, a preliminary analysis of the various catheter designs was conducted at the Medical Device Evaluation Center (MDEC) in Salt Lake City, UT. Because the computational model should account for changes in geometry as well as predict the growth of thrombus over time, two preliminary experimental analyses were conducted. The first was a comparison of thrombus growth location between the three catheter designs. The second analysis was that of the time course of thrombus development in the Mahurkar catheter only.

8.2.1 - Blood Collection and Experimental Apparatus

Fresh bovine blood was obtained from an abattoir following a cardiac puncture and collected into collapsible reservoirs containing the anticoagulant heparin (final concentration 2 U/ml). The blood was then filtered through a 40-μm filter to remove particulates. Platelet reactivity, which is an essential component of thrombosis, is unstable in blood stored for longer than 8 hours. Therefore, experiments were conducted within 8 hours of blood collection to ensure that the platelets and the clotting system were active and relatively stable.

The test circuit consisted of 25 mm I.D. PVC tubing (outer loop) into which the catheter was inserted and sealed. The 25 mm ID PVC tubing was connected to 12.5 mm ID PVC tubing, whose ends were submerged into the blood reservoir maintained at 37˚C. The inflow and outflow ports of the catheters were connected to 6.4-mm I.D. tubing, which served as the inner (‘dialysis’) loop. Blood flow in the outer loop (2 L/min for acute catheters and 2.5 L/min for chronic catheters) was achieved with the help of a Cobe-Stockert roller pump, and blood flow in the inner loop (300 ml/min) was established with the help of a Masterflex peristaltic pump (See Figure 8.4). A pressure gauge was placed in the inner loop between the inlet lumen of the catheter and the pump. This gauge helped establish the degree to which the inlet ports of the catheter were occluded by thrombus. As thrombus formation in the ports increased, there was a corresponding decrease in flow area, thus increasing the pressure drop through the catheter system. When a thrombus embolized, the flow area increased and the pressure drop was restored to the pre-thrombus value.

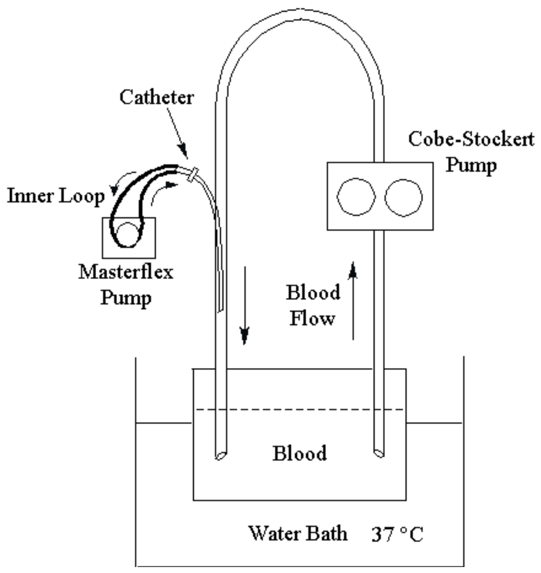

Figure 8.4. Schematic of experimental setup for preliminary testing.

8.2.2 - Comparison of Catheter Designs

The three catheter designs were run simultaneously for a period of forty-five minutes and the pressure drop recorded every five minutes. Table 8.1 shows the pressure measurements over the course of the experiment. The Mahurkar and Ash designs both experienced a pressure pattern consistent with thrombus growth (i.e. steady increase in pressure drop through the catheters) followed by embolization (i.e. sudden decrease in pressure drop), while the Decathlon design experienced only a steady increase in pressure drop.

Table 8.1. Pressure results from experimental catheter comparison.

	Pressure Drop (mm Hg)		
Time (min)	Decathlon	Mahurkar	Ash
0	205	123	187
5	267	277	199
10	271	214	176
15	280	200	150
20	278	167	136
25	307	108	123
30	373	110	111
35	368*	90	106
40	421	94	106
45	500	94	106

*Transient spikes to 500

After forty-five minutes of run time, the catheters were rinsed in saline and bisected, in order to visually examine any remaining thrombi. As can be seen in Figure 8.5, each of the geometries had significant thrombus growth in the inlet and outlet ports. The Decathlon (split tip) geometry also experienced thrombosis in the region between the split tips as well as the flat exterior surface of the outlet branch. The Ash (side hole) geometry experienced dramatic thrombosis in the side holes, but also had a layer of thrombus growth along the entire exterior surface. Because this was a preliminary analysis, the exact reason for increased surface growth on this geometry compared to the other geometries is unknown. However, one explanation might be found in the research conducted by Nobili et al.[19] As platelets in the bulk are forced through the occluded side holes, they experience a repeated history of significant shear forces, becoming activated and increasing the affinity for the catheter surface. Alternatively, the occluded side holes contain activated platelets, which in turn activate nearby platelets through agonist release, propogating the thrombus growth across the surface. Further experimentation is necessary to test these hypotheses.

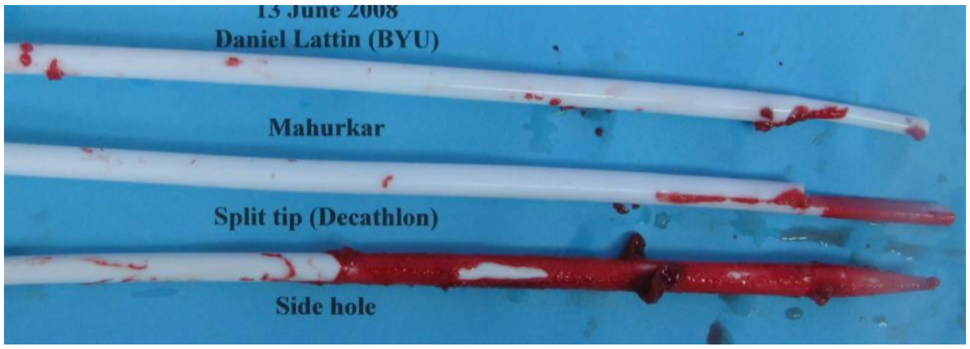

Figure 8.5. Preliminary experimental results before bisection. (Run time = 40 min.)

In Figure 8.6, the interior of the catheter insertion regions can be seen. Again, thrombosis was observed in the inlet and outlet ports of each design. The Decathlon design also had growth along the interior corners of the inlet branch. As on the exterior, significant growth appeared along the entire interior surface of the Ash catheter design. It is likely that some of the thrombus embolized during the course of the experiment. Therefore, it is difficult to use these runs as an absolute measure of thrombus location. However, those locations that contain thrombus are good candidates for thrombus growth in the computational model.

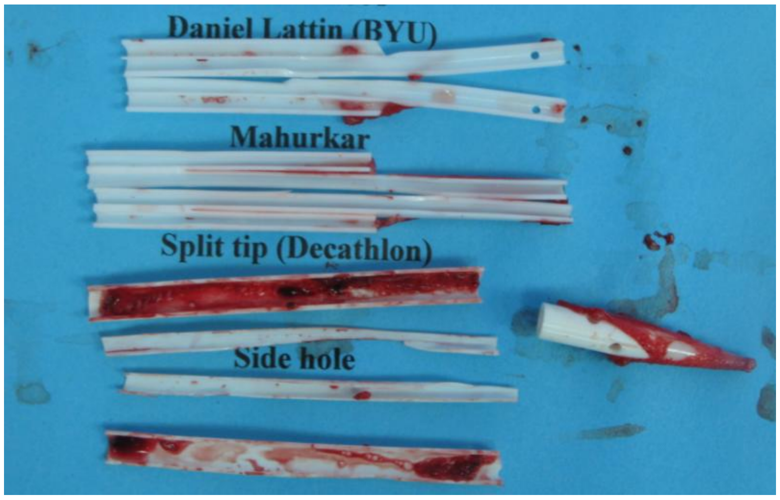

Figure 8.6. Preliminary experimental results after bisection. (Run time = 40 min.)

8.2.3 - Time Course of Thrombus Development

It was also desirable to determine how the thrombus developed over time. To accomplish this, blood was run through two Mahurkar catheters. One of the catheters was allowed to run for forty minutes. The other catheter was removed from the experimental system after 12 minutes for qualitative analysis, and then reinserted into the experimental system for an additional 15 minutes. Both catheters were then removed from the system, rinsed with saline and visually inspected for thrombus formations.

Figure 8.7 shows the catheter that was removed from the system after 12 minutes. Thrombus is clearly visible in the main inlet port, whereas the two side ports are still clear. The same catheter is shown again in Figure 8.8. It is clear that the main port and side ports have nearly completely occluded. In Figure 8.9, the second catheter is shown after running uninterrupted for 40 minutes. It shows the same pattern of growth as that of the first catheter after only 12 minutes. This clearly demonstrates the variability of blood response, even within the same donor batch. On first glance, it also alludes to a pattern of thrombus growth over time, beginning with occlusion of the main port, followed by the occlusion of the side ports. However, as seen in Figure 8.3, the CFD analysis shows significantly higher velocities and shear rates in the side ports, indicating thrombus growth is more likely in that region. A possibly explanation is that thrombus initially grows in the side ports, but due to the high shear forces present in that location, quickly embolizes. The main port experiences lower shear force and thrombus growth is permitted to slowly develop there without much embolization. This growth continues until it eventually occludes the side ports as well.

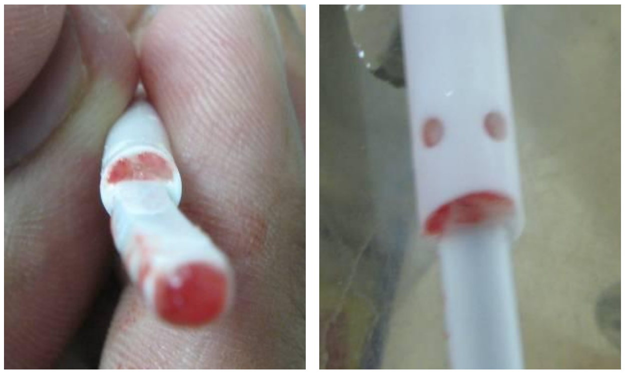

Figure 8.7. Early thrombus growth in first Mahurkar catheter. The occlusion of the main port can clearly be seen in the left-hand image, whereas the side ports are still clear of thrombus in the right-hand image. (Run time = 12 min.)

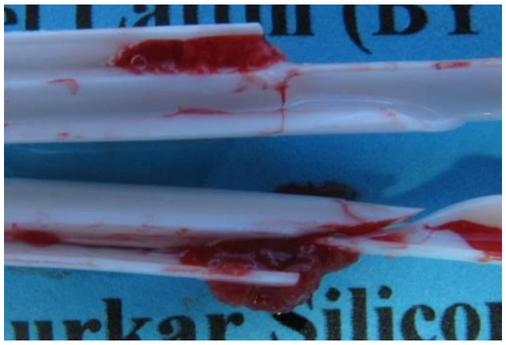

Figure 8.8. Late thrombus growth in first Mahurkar catheter. Near-total occlusion of both the main and side inlet ports is visible. (Run time = 27 min.)

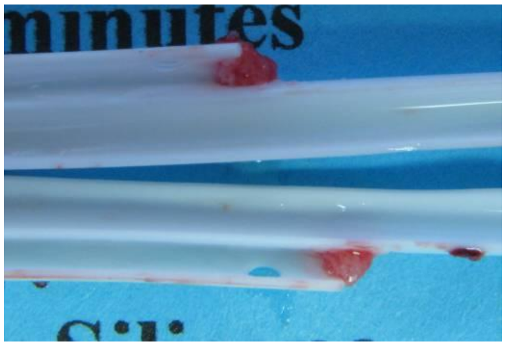

Figure 8.9. Late thrombus growth in second Mahurkar catheter. Thrombus has only occluded the main inlet port. (Run time = 40 min.)

8.2.4 - Conclusions from Preliminary Comparison

While these results are preliminary, they help to give a qualitative perspective on where and how thrombosis should develop when the computational model is applied to the catheter designs. In this way future researchers have a benchmark by which to judge early catheter computational runs. In the CFD-only analyses of the catheters, there are clearly potential areas of high shear. The blood is forced to reverse direction as it travels from the blood vessel, through the inlet ports, and into the catheter. This increases platelet transport to the surface at the sharp corners of the ports, as well as in the recirculation zones just inside the catheter inlet branch. As the computational model incorporates shear-activation of platelets and platelet transport, it would be expected that thrombosis should occur in those regions.

The CFD analysis agrees nicely with the preliminary experimental results, as thrombosis occurred in the inlet and outlet ports of all three designs, even to the extent of plugging the ports. The Ash design grew thrombus along the interior and exterior surfaces. However, further repetitions of experimental runs should be conducted, to determine if those are consistent regions of thrombus growth. The time course experiments of the Mahurkar catheter give a basis for how thrombus should develop over time in that design. While a cursory inspection of the results might indicate that thrombus growth occurs first in the main port, it does not take into account the possible embolization of thrombi that forms in the side ports. It is likely that rapid growth occurs in the side ports but embolizes quickly, followed by slower, more stable growth in the main inlet port. As the current thrombosis model does not include embolization, this would be an important improvement to include in future research.

CHAPTER 9 - CONCLUSIONS AND RECOMMENDATIONS

9.1 - Conclusions

This research initially undertook to take advantage of BYU's new supercomputing system and improvements in automatic meshing software by developing a three-dimensional computational model of thrombosis in three hemodialysis catheters designs. While limitations both in the computational fluid dynamic software and in the hardware prevented the full realization of this goal, significant groundwork has been laid for future researchers to accomplish this task. A significant amount of time was dedicated to addressing the challenges presented by both of these limitations. However, catheter geometries have been designed using CAD software and the automatic meshing process defined. Goodman's C-based Fluent model has been ported to a Fortran-based STAR-CD model. The software limitations that impeded development have been addressed and Goodman's results have been reproduced. The current hardware limitations as well as expected hardware needs for three-dimensional automated meshing and computation have been explored and possible solutions presented. A preliminary comparison of a CFD-only analysis and qualitative *in vitro* experiments for the three catheter designs found that anticipated regions of thrombus growth coincided favorably. Based upon this research, future researchers should be able to apply the thrombosis model to catheter designs and validate the findings experimentally.

9.2 - Recommendations

Significant groundwork has been laid for the future development of the thrombosis model. There are a few improvements that could be made to the model itself. Currently, the model does not allow for the embolization of thrombi. Future directions for the model should include modification of the user subroutines in such a way that this phenomenon is represented. The model may also benefit from inclusion of shear-induced activation of platelets in the bulk, as investigated by Nobili et al., as well as a study of the effect of thrombus viscosity on thrombus growth. Even without these improvements included, though, it is recommended that the current model be applied to the catheter geometries.

The first step in applying the model will be generating a fairly crude mesh using pro-am. In order to minimize the number of computational cells, the catheter geometries should be cut along planes of symmetry. A high memory (32 GB) workstation has been obtained for this purpose, as well as preparing the model once a mesh has been generated. However, in order to produce an accurate model, a grid-independence analysis should then be conducted with meshes of variable-sized edge lengths. Incrementally smaller edge lengths should be generated until the solution obtained is independent of the grid resolution. However, as the edge length decreases, the required memory increases. Therefore, even with the recently obtained workstation, meshing of the catheter geometries to a suitable resolution may require an increase in available memory, or the release of a parallelized automatic meshing utility by CD-adapco.

Once accurate results are obtained from the STAR-CD model in the catheters, it is also recommended that an in-depth statistical comparison between the computational and experimental results be performed. Because the computational model predicts both the size of thrombus as well as thrombus growth over time, a time-dependent experimental model should be created. By bisecting the various catheters at regular time intervals, the location and size of thrombus as a function of time can be determined qualitatively. Quantitative results can also be obtained by radiolabeling platelets in the blood used in the catheters, so that the mass of thrombus can be assessed via radioactive counting. In this way, a statistical comparison of the computational and experimental models can be conducted. This will be an important verification that the computational model accurately predicts thrombosis in each of the three catheter designs.

REFERENCES

1. Branger, B.; Garreau, M.; Baudin, G.; Gris, J., Biocompatability of blood tubings. *International Journal of Artificial Organs* **1990,** 13, 697-703.

2. Zellweger, M.; Bouchard, J.; Raymond-Carrier, S.; Laforest-Renald, A.; Querin, S.; Madore, F., Systemic Anticoagulation and Prevention of Hemodialysis Catheter Malfunction. *ASAIO Journal* **2005,** 51, (4), 360-365.

3. Gray, R. J., Percutaneous Intervention for Permanent Hemodialysis Access: A Review. *SO - Journal of Vascular & Interventional Radiology* **1997,** 8, (3), 313-327.

4. Goodman, P. D.; Barlow, E. T.; Crapo, P. M.; Mohammad, S. F.; Solen, K. A., Computational Model of Device-Induced Thrombosis and Thromboembolism. *Annals of Biomedical Engineering* **2005,** 33, (6), 780-797.

5. Oeveren, W. V., Blood flow dynamics and surface interactions. In *Surfaces and interfaces for biomaterials*, Vadgama, P., Ed. Woodhead Publishing Limited: Cambridge, 2005; pp 414-446.

6. Hanson, S. R., Blood-Material Interactions. In *Encyclopedia of Biomaterials and Biomedical Engineering*, Wnek, G. E.; Bowlin, G. L., Eds. Marcel Dekker, Inc.: New York, NY, 2004; Vol. 1, pp 144-154.

7. Goel, M. S.; Diamond, S. L., Thrombosis. In *Encyclopedia of Biomaterials and Biomedical Engineering*, Wnek, G. E.; Bowlin, G. L., Eds. Marcel Dekker, Inc.: New York, NY, 2004; Vol. 2, pp 1458-1466.

8. Turitto, V. T.; Hall, C. L., Mechanical factors affecting hemostasis and thrombosis. *Thrombosis Research* **1998,** 92, (6, Supplement 2), S25-S31.

9. Hanson, S. R.; Ratner, B. D., Evaluation of Blood-Materials Interactions. In *Biomaterials Science*, 2 ed.; Ratner, B. D.; Hoffman, A. S.; Schoen, F. J.; Lemons, J. E., Eds. Elsevier Academic Press: London, 2004; pp 367-379.

10. Verdonck, P., The Role of Computational Fluid Dynamics for Artificial Organ Design. *Artificial Organs* **2002,** 26, (7), 569-570.

11. Methodology: STAR-CD Version 4.06. In CD-adapco: 2007; pp 1.1,1.6.

12. Methodology: STAR-CD Version 4.06. In CD-adapco: 2007; pp 7.1-6.

13. Prakash, S.; Ethier, C. R., Requirements for Mesh Resolution in 3D Computational Hemodynamics. *Journal of Biomechanical Engineering* **2001,** 123, (2), 134-144.

14. Sorensen, E. N.; Burgreen, G. W.; Wagner, W. R.; Antaki, J. F., Computational Simulation of Platelet Deposition and Activation: I. Model Development and Properties. *Annals of Biomedical Engineering* **1999,** 27, (4), 436-448.

15. Mareels, G.; De Wachter, D. S.; Verdonck, P. R., Computational Fluid Dynamics-Analysis of the Niagara Hemodialysis Catheter in a Right Heart Model. *Artificial Organs* **2004,** 28, (7), 639-648.

16. Mareels, G.; Kaminsky, R.; Eloot, S.; Verdonck, P. R., Particle Image Velocimetry-Validated, Computational Fluid Dynamics-Based Design to Reduce Shear Stress and Residence Time in Central Venous Hemodialysis Catheters. *ASAIO Journal* **2007,** 53, (4), 438-446.

17. Griffith, M. J., The heparin-enhanced antithrombin III/thrombin reaction is saturable with respect to both thrombin and antithrombin III. *J. Biol. Chem.* **1982,** 257, (23), 13899-13302.

18. Hellums, J., 1993 Whitaker lecture: Biorheology in thrombosis research. *Annals of Biomedical Engineering* **1994,** 22, (5), 445-455.

19. Nobili, M.; Sheriff, J.; Morbiducci, U.; Redaelli, A.; Bluestein, D., Platelet Activation Due to Hemodynamic Shear Stresses: Damage Accumulation Model and Comparison to In Vitro Measurements. *ASAIO Journal* **2008,** 54, (1), 64-72.

20. Wootton, D.; Markou, C.; Hanson, S.; Ku, D., A Mechanistic Model of Acute Platelet Accumulation in Thrombogenic Stenoses. *Annals of Biomedical Engineering* **2001,** 29, (4), 321-329.

APPENDIX A: PAUL GOODMAN'S CODE OVERVIEW

Introduction
Paul Goodman's thrombosis model was done in the Fluent CFD application and programmed in the C language.

Important Concepts
Hellum's equation: Relationship between exposure time and shear stress for predicting the length of time needed to activate *surface-adherent* platelets
Sorenson's model: Platelet model Paul based his model on. Incorporates platelet activation based on achieving threshold of agonist concentrations, as well as surface deposition based on surface availability.
Single platelet embolization: One of the important and unique aspects to Paul's model is he incorporated single platelet embolization (as well as thrombus embolization).
User-Defined Memory Allocations (C_UDMI): User-defined variables specific to cell. Summary contained on pg. 2 of Annotated Code (AC). Enabled Paul to calculate all of the necessary terms for the thrombosis model.
Thrombus Growth: Thrombus growth is comprised entirely of platelets, both activated and unactivated. It is measured by the mass flux of platelets to both wall surfaces and other thrombus cells, then converted to a volume fraction of thrombus in the cell.

Summary of Functions
DEFINE_ADJUST: Loops through cells and performs the following tasks in each cell:

- Sums area of all faces in the cell (pg. 5)
- Calculates cumulative thrombus growth in the cell(pg. 5)
  - Adds "delthu" and "deltha," the change in activated and unactivated thrombus fraction.
- Calculates shear rate in the cell (pg. 6)
  - Examines all shear rates (du/dx, du/dy, du/dz, dv/dx…) and identifies maximum value.
- Updates activation time of the cell that is on wall or in thrombus (pg. 6 & 8)
  - Follows Hellum's equation (dissertation pg. 73)
- Identifies and adjusts "cell type" (pg. 6-9)
- Calculates and updates M, Ma, S, X for the cell (pg. 7-8)
  - Based on flux of resting (unactivated) platelets and activated platelets to wall, as well as single platelet embolization (dissertation pg. 63).
- Pages 10-15 are his attempts at performing embolization. I haven't looked too closely at this.

DEFINE_PROPERTY: Fixes the viscosity of thrombus cells at a very high value (essentially making it solid).

DEFINE_PROFILE: Gives a laminar inlet profile (specific to flow-cell)

DEFINE_SOURCE: Defines source terms for each scalar species

- Uses "grid id" of cell to determine the source term
- For platelets:
    - Includes Sorenson's "activation threshold" model (pg. 16-17, dissertation pg. 59)
    - Incorporates platelets adhering to activating platelets attached to wall (uses M, etc.)
    - Builds "thrombus" from both unactivated and activated platelets as a vol% of cell.
- Other species:
    - Follows Source Terms from pg. 59 of dissertation, although also has different terms if it is in a thrombus. It's a little convoluted, because some terms are multiplied together, etc.

DEFINE_DIFFUSIVITY: Defines the diffusivity of the various species

- Larger species are dependent on shear rate. His routine is incomplete as given in the annotated code.

DEFINE_SR_RATE: Defines the reaction rate for platelet adhesion, thrombin, thromboxane A2. Also calculates the values here that will alter M, etc. (such as single platelet embolization).

- Seems to duplicate some of the code from the platelet sources. I think, but could be wrong, that the surface reaction adds to (or takes away from) a surface concentration, whereas the source terms are for the bulk concentration in the cell.

APPENDIX B: AUTOMESHING IN PRO-STAR 4.06

Introduction

- This will start in STAR-Design.
- In order to use STAR-Design and Automeshing on the supercomputers, you will need to know how to "source" between v3.26 and v4.04 (STAR-Design is only available with v3.26).
  - Depending on your shell configuration, the command to source to v3.26 is either of the following two commands:
    - "source /fslapps/starcd_3.26/etc/setstar"
    - ". /fslapps/starcd_3.26/etc/setstar"
  - For v4.06, it is one of the following two commands:
    - "source /fslapps/starcd_4.06/etc/setstar"
    - ". /fslapps/starcd_4.06/etc/setstar"
- The over-all process can be summarized by:
  - Model the geometry in STAR-Design
  - Create a surface mesh in pro-surf
  - Generate a volume mesh using pro-am

STAR-Design: This is where you will create the geometry. Alternatively, you could create it in a different CAD program and import it here.

- Source to v3.26
- Open STAR-Design by entering the command "stardesign gl" in the Linux command line
- Create the desired geometry
- Save the geometry by selecting "File→Save As…" (not a necessary step, but useful if you ever want to open the geometry again).
- Select "File→Export…" from the main menu bar.
  - Select a "File type" of "pro-Surf Restart Files (*.ezs,*.EZS)"
  - Name the file
  - Click on "Save"
- Exit STAR-Design

pro-surf: This is where you will create the surface mesh and the meshing database file

- Source to v4.06
- Open pro-surf by entering the command "prosurf gl" in the Linux command line.
- Select "File→Read pro-surf Restart File" from the menu bar
  - In the ensuing window, click on the icon that looks like a folder
  - Select the appropriate file and click "Open"

    - Click on “OK”
- In the upper-right corner of the screen, under “Entity Display” you can toggle whether to show the model’s “Curves” or “Surfaces.” It is useful to have the “Surface” option selected.
- In the “pro-surf NavCenter” click on “Surface Groups.” It is often useful to organize the various surface into the future boundary regions of the model.
    - Click on “Create New Group”
        - Give it a “Name” (I usually name it the same as the boundary region it will be, i.e. “Inlet” or “Walls”)
        - Click OK
    - Under “Select Surfaces” (making sure that the “Surface” option is selected as described above), click on the icon that looks like an arrow.
        - In the main viewing area, click on the surfaces you want associated with the group. If you make a mistake, you can clear the surfaces from the list by clicking on the icon that looks like an “X”
    - Once all of the surfaces associated with that group have been selected, click on “Add Selected.”
    - Click on “Create a New Group” to repeat the process for each group
- Click on “Grid” and then “Generate”
    - You can either “Use Default Values” or “Use Custom Values” for the edge length scale. If you choose to use custom values, they can not be smaller than the “Point” and “Curve” tolerances. You can change these tolerances Under “Cad Repair→Set Tolerance” in the pro-surf NavCenter.
    - Click on “OK” to generate the surface mesh
- Click on “Output→Write Database”
    - Enter the name of the “Database File.” It is a good idea to give it the same name you will give the pro-am “Case Name” (see below)
    - Input the “Database ID Number.” I usually leave it at the default of “2.”
    - If you created groups previously that corresponded to boundary regions, Select “By Group” from the “Save Cells to Tables:” drop-down option
    - Click on “OK”
- Exit pro-surf. Feel free to first save the pro-surf restart file.

pro-am: This is where you will generate the volume mesh. Feel free to play around with the various options and learn how to use it. You can reference the “Meshing User Guide” for explanations of all of the various options. I will describe what I normally do.

- Start pro-am by entering the command “proam gl” in the Linux command line
- Give the file a “Case Name”
- Select “Advanced→Automesh…” from the main menu bar
    - Set “Mesh Type” to “Trimmed” and “Prism Layers” to “Yes”.
        - A trimmed mesh starts with smaller cells near the wall and larger cells in the core. This minimizes the number of cells necessary. Alternatively you could create a tetrahedral or polyhedral mesh, but these don’t have the capability to vary cell size.
        - Prism layers are designed to be used in heat transfer situations and a few, flat cells to be created immediately next to the wall. I don’t really use it

for that, but it also allows you to identify the boundary regions that correspond to the groups created in pro-surf.

- <u>Input Settings</u>
  - Click on the "Database" button and select "Load." This will load whichever Database ID is in the text field immediately to the right of the button. This should correspond to the "Database ID Number" input above (defaulted to "2").
  - Leave the "Input Size Option" set to "Model Units". This means all length sizes you input will be relative the units you specify for the model, i.e. if one model unit length is equal to 1 mm, then an edge length of 0.05 will be 0.05 mm, or 50 microns.
  - You can input "Volume Sources". I have never really used this option, but it might be very valuable. It allows you to insert previously defined geometries (cylinders, spheres, bricks, or cones) that have a specific edge length. This might allow you to create larger cells of a specific length in a defined region, rather than allowing AutoMesh to dynamically size the cells in that region.
- <u>Trimmed Cell Mesh Settings</u>
  - Set "Surface Size" to "Min and Target"
  - Input the "Minimum Size", "Target Size", and "Max Cell Size." By varying these values, you can vary how dense the final mesh is. I usually start with a "Target Size" and then modify the other two correspondingly.
  - You can specify certain "Boundary Values" for specific boundaries.
    - I haven't really used this very much, but it may be possible to use this to minimize the number of cells used at the inlet and outlets.
  - Click on "Template Settings".
    - Set "Template Mesh Type" to "Hexahedra"
    - Set "Template Mesh Growth Type" to "Two Level". I have found this works better for having a thicker region of small cells near the walls.
    - Set the "Core Mesh Growth Rate". I usually set it to "Slow" or "Medium"
    - I haven't played much with the "Surface Mesh Growth". I usually leave it at "0"
    - Specify the "Small Cell Cutoff Size". I usually set this to a value near my "Target Size", but that might not be the best way to do it.
    - Set the "Small Cell Growth Rate". I usually use "Very Slow". The slower the growth rate, the more small cells in the mesh.
    - Click "Close"
- <u>Prism Layer Settings</u>
  - Specify the "Prism Layer Thickness". I usually set it to my "Minimum Size" set above. This is the thickness of all the prism layers combined.
  - Specify the "Number of Prism Layers". I usually only have 1 or 2.
  - The "Stretch Factor" is the thickness of each prism layer relative to the preceding prism layer.
  - Click on "Boundary Settings". This is where you can associate the previously created groups with a corresponding boundary type.

        - Specify a “Description”, such as “Inlet 1”, etc.
        - Set the “Boundary Type(s)”.
            - By clicking on “Tools→Cell...” from the main menu bar, you can view all of the cell types. If you created groups in pro-surf and specified to save the cells to tables “By Group”, they will be listed here by group. The pro-surf groups usually begin number with 11, and are in the same order created in pro-surf (the first user-created group, which is usually group #2 is pro-surf is cell type 11 in pro-am).
        - Make sure “Cell/Group” is set to “Cell Type”.
        - Set the appropriate “Boundary Type”
        - Click on “Close”
    - Click on “Start Meshing”. If you would like to submit the job as a batch job on the supercomputers (for memory purposes), follow the instructions below:
        - Once the “Status” finishes reading “AutoMesh – Initializing”, click on “Abort”
        - Close the AutoMesh panel
        - Save the model (“File→Save Model”)
        - Quit pro-am.
        - A directory called “automesh_batch” should have been generated. Go into this directory by typing “cd automesh_batch” into the Linux command line.
        - Edit the file “mktrun.sh” by typing “vi mktrun.sh” in the command line.
            - Enter into insert mode by typing “i”
            - Above the line starting “$PROSTAR/bin…”, insert a line containing the appropriate source command for v4.06 (see above).
            - Exit out of insert mode by typing “Esc”
            - Save and exit by typing “:x”
        - You can now submit mktrun.sh to the PBS scheduler by using the “qsub” command, as described in the document “Running a Solution”. It should look something like this:
            - “qsub –q quad –l nodes=1:ppn=8,walltime=3:00:00 mktrun.sh”
        - In this case, I specifed using the 8 processor “quad” queue. This is a high memory (24 GB RAM) node. You must reserve all 8 processors on the node to access all of the memory. Unfortunately, it will be phased out in the near future (possibly end of summer 2008). Fortunately, it should be replaced with different high memory nodes.

You need approximately 1 GB of memory for every million cells being generated using Trimmed Cell meshing. On a normal node (not using the “–q quad” option), there are only 8 GB available. If you attempt to create a mesh that requires more memory, it will essentially shut down the node. This is not good, and the supercomputing people really don’t like when we do that.

APPENDIX C: SETTING UP A PRO-STAR 4.06 MODEL

Introduction

- This assumes the mesh has already been generated.
- At each step, it is important to press "Apply" in order to actually save the changes. It will usually prompt you to do this if you attempt to move away from the screen without doing this.
- The output for most commands appear in the pro-STAR output screen just below t he plotting area. This is a good place to look for explanation of errors, etc.
- Sometimes an error will come up, especially when dealing with boundaries and gr ids, instructing the user that there is "insufficient scratch space." Simply follow th e instructions by typing "memory" in the pro-STAR command line, followed by t he parameter indicated and a sufficient size (it usually gives a minimum required size in the pro-STAR standard output window, located beneath the plotting area).
- Before doing any further set-up, you must enable user subroutines.
  - From the main pro-STAR toolbar, select "File→System Command…"
  - In the ensuing window, type the command "ufiles"
  - Close the "System Command" window
- If you have previously created a problem input file, simply load it by selecting "Fi le→Read Coded Input File…" and then selecting the appropriate file from the ens uing window. You will still need to go through the problem parameters to make s ure they loaded correctly, as well as check that the boundaries are defined appropr iately both under "Locate Boundaries" and "Define Boundaries" (see below).
- If you don't have a previously created problem input file, the following outline sh ould mirror the structure found in the pro-STAR Model Guide:

Analysis Features: Establishes the general properties for the model: steady state vs. transient, reacting flow, etc.

- Set the "Time Domain to "Transient" and click "Apply"
  - This should activate the "Time Relationships" submenu in the Guide
    - Time should be specified in "Seconds"
- All other categories should remain the default value:
  - Free Surface: Off
  - Cavitation: Off
  - Multi-Phase Treatment: None
  - Rotating Reference Frame Status: Off
  - Reacting Flow: None—Our model has flow at the surfaces, but this option is f or bulk flow reactions only.
  - Aeroacoustic Analysis: Off

        - Liquid Films: Off
        - Stress Analysis: Off

Geometry: Not used, unless you would like to create geometric entities by hand. Not recommended, except perhaps to create splines. Splines allow you to track pressure drops, etc. along a specific path.

Grids: Allows the user, to import, create, check, refine grids (computational meshes)

- Check Grids
    - Select "All", then click "Apply" to run an integrity check on the mesh.
    - If there are errors, you may just want to create a new mesh. I have not had mu ch success using the "Fix Grids" utilities.

Locate Boundaries: Used to identify and create boundaries.

- Create Boundaries. This is an essential step. Usually grids created with the AutoM esh utility identify all boundaries as walls. The best way to change the boundaries is as follows:
    - Make sure the "Regions" tab is selected.
    - Select the "Boundary Region" that you wish to create by clicking on the first o pen space in the list. If you wish to modify a region, click on the number for t hat boundary region.
        - In order to be compatible with the input file (described later), I have usuall y selected Region 1 as Walls, Region 2 as Inlet, and Region 3 as Outlet. T his setup may have to be modified, depending on the geometry being mod eled.
    - Select the "Type" of boundary from the drop-down menu
    - Give the boundary a meaningful "Name" and click "Define"
    - If the boundary regions are not already organized according to corresponding groups, do the following:
        - Make sure the region you desire to identify as the boundary is visible in th e plotting area and the recently defined boundary is highlighted in the "Bo undary Regions" list.
        - Select "Create by Picking Surface based on Edges" from the "Create/Modi fy Boundary Action List" then click "Apply."
        - Right-click on the face of the desired boundary region in the plotting area.
        - Click "Define" again
    - If the boundaries are organized according to region and you would like to assi gn them to a different boundary:
        - Click on the number of the region you wish to modify.
        - Create a new boundary set from that region, by clicking on the boundary s et toolbar (Looks like "**B►**" in the lower left-hand side of the screen), and clicking "New→Region Current."
        - Click on the number of the region you wish to assign them to
        - Click on "Change Curent Boundary Set" from the "Create/Modify Bounda ry Action List" then click "Apply."

Validate Boundary Locations: Can be used to make sure there are no illegally placed boundaries

Thermophysical Models and Properties: This is where all of the properties of blood and various scalars are defined.

- Thermal Options: We aren't modeling any heat transfer.
- Gravity: Make sure that the "Direction of Gravity Force" is correct, according to t he indicated Coordinate System.
- Liquids and Gases
  - Molecular Properties
    - Select "Define user material" from the drop-down menu
    - Name the material "Blood"
    - Give it a "Constant" density of 1056 kg/m3
    - Indicate a "User" molecular viscosity and give it a default value of 0.0035 kg/ms
    - Specify the molecular weight as 28.964 kg/kmol
    - Click "Apply" and "User Define." This second button creates the skeleton molecular viscosity user subroutine, "vismol.f"
  - Turbulence Models: Turn "Turbulence" off.
  - Thermal Models: Turn "Temperature Calculation" off.
  - Fluid Initialization: This initializes the bulk fluid conditions. Could select "Us er" to define a user subroutine that gives a laminar profile throughout the mod el. For simplicity's sake, I have just given it a "Constant" velocity in the axial direction (for the flow cell model, 0.0466 m/s in the W direction).
  - Monitoring and Reference: In the "Reference Data" area, input both a referenc e temperature and a reference pressure. All pressure and temperature data are actually calculated as deviations from these reference values. Select a "Pressur e Cell Number" and "Monitoring Cell Number" using the following process. The Monitoring Cell is the cell used to output data throughout the solution pro cess.
    - I have found it helpful to first set the plot to a "Geometry" plot using a "Cl ipped Hidden" plot type in order to select an interior cell.
    - Click on the button with an image of a computer mouse. Select the desired cell from the plotting area.
    - I usually use a cell near the entrance as the Pressure Reference Cell and a cell where thrombus growth is expected as the Monitoring Cell
- Additional Scalars: This is where both the active and passive scalars are input. A ctive scalars are the agonist species whose concentrations are solved for. The pass ive scalars are what is used in place of Fluent's User Defined Memory Allocation s. They keep track of variables used to simulate thrombus growth.
  - Molecular Properties—Make sure to click "Apply" after entering each scalar
    - Names and molecular weights of scalar species 1-7 (in ascending order): P l—1.28e12, aPl—1.28e12, ADP—424.4, TxA2—352.5, pT—72000, T—3 6600, aT—62000
      - Set "Influence" to "Active" and set the density and molecular viscosity to be the same as blood (1056 and 0.0035, respectively).
    - Names of scalar species 8-22 (in ascending order): nBldSt, ThrFr, actT, Sh rRt, newThr, NFcBnd, NFcThr, xM, xMact, Xact, Sp, PlSrc, aPlSrc, pulse, NPlsFl
  - Binary Properties
    - Set the "Material Mass Diffusivity" to "User" then click "User Define." T

his will activate the subroutine “diffus.f”

    - Sometimes, in order to click “Apply”, you must first select a scalar species from the list.
  - Scalar Initialization: This sets the initial value for each scalar species.
    - To set a particular scalar species’ value, click on the desired “Scalar Numb er” of the species in the list in the lower half of the screen.
    - Make sure the “Values” drop-down list is set to “Constant” and e.0nter the value in the “Initial Mass Fraction” text area. Make sure to click “Apply” after each species.
      - Initial Values for Active species: Pl—0.0006; aPl—6e-6; ADP—0; Tx A2—0; pT—7.5e-5; T—0; aT—0.000167.
      - Initial Values for Passive species: Set all to 0
- Porosity: We don’t use any porosity features
- Sources:
  - For the seven Active species, under “Source Term,” set the drop-down menu t o “Scalar,” the “State” to “On” and “Define Source” to “User Coding.” This w ill create the subroutine for generating sources, “sorsca.f”
  - You navigate between scalar species using the small arrow buttons next to the “Scalar #.” Make sure to click “Apply” after changing each species.
  - None of the passive species should have a source term.

Define Boundaries

- Define Boundary Regions: This allows you to set the conditions for each individu al boundary.
  - Click on the “Wall” boundary region(s). Make sure the “Slip” Wall Parameter is set to “No” and there are no velocities set
  - Click on the “Inlet” boundary region(s). Set the “User Option” to “User” then click “Apply” and “User Define.” This will allow you to set up a parabolic pro file at the inlet in subroutine “bcdefi.f”
  - Click on the “Outlet” boundary region(s). Depending on the geometry, you m ay need to specify a “Condition” of “Split,” and then input the fraction of “Flo w Split” associated with that outlet.
- Scalar Boundaries: Sets the boundary condition for the scalar species.
  - All of the scalar species for the “Wall” boundary region should have a conditi on of “Zero Flux”
  - All of the Active scalar species should have a “Boundary Condition” of “Cons tant” with the “Value” set to the initial mass fraction specified above.

Analysis Controls

- Solution Method: This is where you can specify the solution method, residual tole rance, corrector stages, pressure under-relaxation factor, etc. Can be a useful place to fine-tune obtaining an accurate solution.
  - Set “Solution Algorithm” to “SIMPLE”
  - Increase “Maximum Number of Outer Iterations” to “1000”
- Primary Variables: Depending on the solution method, can choose which variable s to solve for, as well as relaxation factors, etc.
- Additional Scalars: Specify which scalars to solve for and how to solve for them.
  - Scalars 1-7 (active)

        - Should have the “Solve for Scalar” box checked
        - “Solution Method” should be set to “Transport”
    - Scalars 8-22 (passive)
        - All except ShrRt (scalar 11) should have “Solution Method” set to “Off.” ShrRt should have “Solution Method” set to “User” and the click on the “User Define” button. This creates the “scalfn.f” user subroutine.
        - All except nBldSt, ShrRt, NFcThr (scalars 8, 11, and 14) should have “Sol ve for Scalar” unchecked. Those three should have it checked (although n BldSt and NFcThr should still have “Solution Method” turned “Off.” By c hecking the box, I think (although I could be wrong, since there isn’t any r eal documentation about this) that it makes it available for writing a new v alue in posdat.f (see below). Even if it is not checked, it is still available fo r reading the current value in posdat.f
- Analysis Output: Specify which data to output to the post-processing file
    - Click on the “Post” tab.
        - Make sure the “Write Solution file” option is checked
        - Under “Additional Output Data”, check the box next to “User Subroutine” then click “Apply” and “User Define”. This creates the subroutine “posda t.f” which will contain most of the thrombus calculations.
    - Click on the “Transient” tab.
        - Make sure the “Output file format” is “ccmt” and check the “Define outpu t period” box.
            - Specify a “Starting at time” (usually 0 sec) and an “Output interval” (t he ccmt files are huge—several GB—so try to use the greatest frequen cy that is still useful, depending on whether you are more interested in the initial time steps or the final solution).
        - Under “Data to Write,” check the “Cell” box
        - For each variable in the list you desire to have in the post processing file, s elect the variable, check the “Post” box, then click “Apply.” I usually have the velocity, pressure, molecular viscosity, nBldSt, and ThrFr variables tu rned on.
    - Switches and Real Constants: These are a bunch of secret CD-adapco variable s. The only way to know how to use them is to contact them. Ridiculous. We did manage to obtain a list of definitions for v3.22
        - Make sure Real Constant “C11” is set to one (“1”) under the “Real Consta nts” tab. This makes all of the velocity derivatives available to both sorsca. f and scalfn.f

Analysis Preparation

- Model Validation: Make sure to Validate “All”
- Run Time Controls: This is where you set the time steps and length of time (mode l time) the solution will run for.
    - Set “Run time control” to “Run for” in the drop-down menu
    - Input the desired amount of run time in the “Time (sec)” field and click “Appl y”.
    - Set “Time Step Method” to “Constant”
    - Set the “Period start time” to 0

    - Input the size of the time step in “Time step for period” and then click “Set.” Paul found time-step independence at 0.05 second intervals.
- Run Analysis Interactively: This allows for creating the problem and geometry fil es. These two files are necessary for running a solution.
    - Make sure the “Model Units” are correct (for the flow cell, millimeters). The “Scale Factor” should update automatically.
    - Set the “Precision” to “Double.” This slows the calculation slightly, but is nec essary for an accurate calculation, since our agonist concentrations are so low.
    - Make sure “Write Geometry File” is set to “Yes” (unless you have previously created the geometry file. This is done by selecting “File→Save Geometry…” from the main pro-STAR toolbar, and then clicking “Apply” in the ensuing w indow. Make sure to close the window. You can also create a problem file by selecting “File→Save Problem…” and then clicking on “Save.”)
    - Click on “Create Batch.” Wait for the message window to appear and click “O K”. This will create the problem file, as well as generate a script file in the mo del’s directory that can be run in order to submit a job in batch mode. By click ing “Start,” pro-STAR will start solving the problem interactively. This is typi cally not practical for the meshes we have, unless they are very crude.

Saving the Model

- Make sure to save the model by selecting “File→Save Model” or “File→Save Mo del As…” from the main pro-STAR toolbar.

You can also save a “problem setup file” that will save all of the parameters you have set and allow you to read it into another model. This way, you only have to set up the model once, and then tweak it (especially the boundary definitions) for each individual model. To do this, select “File→Save As Coded Format…” from the main pro-STAR toolbar.

APPENDIX D: RUNNING A BATCH SOLUTION IN STAR-CD

Introduction

- This assumes that the geometry and problem files have already been created.
- This also assumes that the "Create Batch" option was executed in pro-STAR, so that the "star_inter.sh" file has already been created.
- If you have already run a solution in the same directory, in order to run the "star_inter.sh" script (see below), you must first delete the ".reg" files previously generated.

Creating and setting up the batch script file

- In the directory containing the model files, execute "star_inter.sh" by typing in the Linux command line "./star_inter.sh"
- The previous step will have generated a new script file, "batch.sh" This is what will be submitted to the supercomputer. First you will need to modify it slightly.
  - Edit "batch.sh" by typing "vi batch.sh" in the Linux command line.
  - VI is not a very user-friendly word processing program. Be careful what you type, as you may execute an undesired command. In order to edit the file, type the letter "i"
  - The line starting "star –case=" should have the option "-mpi=mpich " input just prior to the variable "$PNP_JOBNODES". Make sure there is still a space between them, i.e. "-mpi=mpich $PNP_JOBNODES". This makes sure the mpich option is used for parallel processing. The default option is hpmpi, but I have found that mpich has a lower failure rate on our supercomputing system.
  - Exit editing mode by hitting the "Esc" button and save batch.sh by typing ":x"
  - If you want to exit without saving any changes, type ":q!"
- I have found that, especially with the larger meshes, the jobs have a 25-50% failure rate. For this reason, I usually submit the three or four identical jobs. I do this by first executing the above steps, then copying all of the files into three or four different directories. To do this:
  - First make the directories using the "mkdir" command, i.e. "mkdir dir1 dir2 dir3"
  - Copy the contents of the original directory using "cp –r [orig_dir]/* [destination]"
  - Go into each directory and execute the qsub command (see below).

Submitting the job to the supercomputers

- Once “batch.sh” has been created and modified, you can submit it to the supercomputers, using the “qsub” command. Instructions are found on the supercomputing website. An example is shown below:
    - Type “qsub –l nodes=2:ppn=4,walltime=8:00:00 batch.sh”
        - The letter l option lets you specify the number of nodes, processors per node (ppn), and walltime. These three values follow the “-l” option. The first two are separated by a colon and the second two are separated by a comma. The walltime is the amount of time it will remain on the supercomputers once it has started running and is given in the format hh:mm:ss. It is better to over-estimate slightly, so that the queue system doesn’t kick the job off before it has finished. It will automatically remove itself once it has solved completely.

Monitoring the job

- Once the job has been submitted, you can monitor it a couple of ways.
    - You can monitor the supercomputers to see if it is still running by typing “showq –u [username]” in the Linux command line. This will show any jobs you have submitted, whether they are running, waiting, or blocked (I have found that blocked jobs will still eventually make it to the queue).
    - You can monitor the output of the job (which time step it is on, etc.) by reading the “batch.sh.o[job#]” file that is automatically created by STAR-CD once the job starts running. To view the whole file type “less [filename]” in the command line. To view just the end of the file (and thus the last time step), type “tail [filename] in the Linux command line. You can quantify how many lines to see with the “tail” command by adding a “-n [# of lines]” option.
    - You can determine the nodes used by the job by typing the command “checkjob [job#]” The job number is displayed by the showq command.

APPENDIX E: POST-PROCESSING IN PRO-STAR 4.06

Introduction

- This assumes that calculations have already been run on the given model and the model file has been loaded.

Post Processing: Click on this folder in the pro-STAR Model Guide to start

- Load Data
    - Click on the "Files(s)" tab
        - Under "Analysis" select "Transient"
        - Make sure the "Post File" is set to [model name].ccmt
        - Click on "Add File"
        - Click on the name of the file that appears in the "Transient" list.
        - Click on "Open Transient File"
        - Click on the desired time step in the subsequent "Time Step" list.
        - Click on "Store Time"
    - Click on the "Data" tab
        - The "Data Type" should be set to "Cell"
        - Select a "Vector Data" and/or a "Scalar Data" that you would like to view for that time step. I usually select the "Velocity Components UVW" from the vector data and/or the "ThrFr" Scalar Data (although sometimes I look at "nBldSt" or "Molecular Viscosity").
        - Click on the "Get Data" button.
- Create Plots: This is where you graphically display the data for the time step. Play around with the various types of plots available. Below is what I usually do:
    - Select the "Section/Clipped" tab
        - Set the "Plot Type" to "Clipped"
        - Set the "Option" to correspond to the type of data you loaded
        - Set the "Plot" to "Single Plane"
        - Pick which normal plane to plot (I usually have the X & Z normals set to 0 and the Y normal set to 1, with each point set to 0).
        - Click on "Apply" then "Plot to Screen"

Panels: Panels can be used to animate transient results in pro-STAR. They are created by selecting from the pro-STAR menu bar "Panels→Define Panel…" After giving the panel an appropriate name in the available text field, click on "New". This will add newly created panel to the list. Select it from the list and click on "Open".

Once the panel is open, there is a bank of available, blank buttons. Select one of the buttons, enter a name for it in the "Button Name & Definition" text field, then input the desired pro-STAR coding in the large field below.

There is not a lot of documentation about pro-STAR coding, and I have not worked out all of the kinks in v4.06. Page 9-46 of the User Manual has some information for transient animation (albeit, not very helpful). Other resources include the “Day 3: Transient Analysis Tutorial” from the custom user training (more helpful, but limited in scope) and the Star-CD New User Training Manual, Exercise 3.4: “Advanced Post Processing Using Macros and Panels.” Here is some sample code for animating transient results on the screen (Use the pro-STAR Help for details on each command):

```
TRLOAD,,
!This line loads the transient results
POPT,BOTH PLTY,SECT
!Sets the Plot Option (In this case both contour and vector) and Plot Type (in this
        case a section plot)
CSET,NEWS,FLUID
!Defines a new cell set of the fluid cells
*DEFI,NOEX
!Begins the loop
STOR,NEXT
!Stores the next time step in the transient data
GETC,ALL,CONC,9
!Gets the specific data to be plotted (for this case, all of the velocity vectors and
        the concentration of scalar 9)
CSCALE,14,USER,0,1.0 !CSCALE,14,AUTO
!These are two ways to set the color scale. I find it useful for my work to use a
        user scale that doesn't change with time like the auto scale
CPLOT
!Plots to the screen
*END *LOOP,1.0,100,1.0
!Ends the loop and sets the start, end, and increment values for the loop
```

APPENDIX F: USER SUBROUTINES IN PRO-STAR 4.06

Introduction

- Uses Fortran 90
- In order for these files to be included in the solution calculations, they must be pre sent in the "ufile" directory.
- The following is a list of all the subroutines and their functions. Following the list is the complete code for each subroutine:

initfi.f

- Called once in each cell before any time steps have started
- Used to give an initial parabolic profile to the bulk fluid.

bcdefi.f

- Defines the boundary condition for inlet boundaries.
- Called in each cell during the solution phase of the time step.
- Currently used to set a parabolic profile at the inlet.
- Returns several variables, including U,V,W, the three velocity vectors.
- Would need to be changed or not included based on the geometry.

diffus.f

- Sets the diffusivity of each scalar species as a function of shear rate.
- Called in each cell during the solution phase of the time step.
- Returns the variable DIF.

scalfn.f

- Calculates the maximum shear rate in the cell.
- Called in each cell during the solution phase of the time step.
- Returns the variable PHI.

sorsca.f

- Sets the volumetric source terms for the active scalar species (agonists).
- Called in each cell during the solution phase of the time step.
- Returns the variables S1P & S2P, where the form is "Source = S1P – S2P*SCAL AR(IS)." SCALAR(IS) is the mass fraction of scalar species IS in the current cell. We actually set S2P = 0, so that we can specify a constant source term each time step.
- Based on Paul's code. I tried to annotate it fairly heavily.
- There is a variable "AtoV" which converts surface flux terms to volumetric sourc e terms. This needs to be updated, depending on the shape of the cell used (hexah edral vs. tetrahedral).

vismol.f

- Sets the molecular viscosity of the computational cell.
- Called in each cell during the solution phase of the time step.
- Based on the variable “ThrFr” (thrombus fraction).
- Returns the variable VISM.

posdat.f

- Enables the user to output (and input) data.
- Called once (not once in each cell) at the beginning and end of each time step.
- Contains the majority of our “thrombus growth” calculations.
- At the beginning of the *first* time step, loops through all cells and initializes the “b lood status” or grid ID.
- At the beginning of *each* time step, loops through all cells and:
  - Calculates surface coverage terms (M, Mact, S, Xact)
  - Calculates the change in thrombus and updates thrombus fraction.
- At the end of each time step, loops through all cells and updates the “blood status” or grid ID of cells that become thrombus (or are neighbors to a new thrombus).
- There is a variable “AtoV” which converts surface flux terms to volumetric sourc e terms. This needs to be updated, depending on the shape of the cell used (hexah edral vs. tetrahedral).
- The surface reactions included in Paul Goodman’s codes are calculated here, save d, and used in sorsca.f

```
C***********************************************************************
      SUBROUTINE INITFI(U,V,W,PR,TE,ED,T,SCALAR,XVF2)
C     Initialise fields
C***********************************************************************
C--------------------------------------------------------------------------*
C     STAR-CD VERSION 4.06.000
C--------------------------------------------------------------------------*
      INCLUDE 'comdb.inc'
      COMMON/USR001/INTFLG(100)
      DIMENSION SCALAR(*)
      INCLUDE 'usrdat.inc'
      EQUIVALENCE( UDAT12(001), ICTID )
      EQUIVALENCE( UDAT03(001), CON )
      EQUIVALENCE( UDAT04(001), CP )
      EQUIVALENCE( UDAT04(002), DEN )
      EQUIVALENCE( UDAT04(062), VISM )
      EQUIVALENCE( UDAT04(063), VIST )
      EQUIVALENCE( UDAT04(067), X )
      EQUIVALENCE( UDAT04(068), Y )
      EQUIVALENCE( UDAT04(069), Z )
C---------------------------------------------------------------------------
C
C    This subroutine enables the user to initialise the following
C    dependent variables: U,V,W,PR,TE,ED,T,SCALAR,XVF2 to arbitrary
C    values.
C
C    ** Parameters to be returned to STAR: U,V,W,PR,TE,ED,T,
C                                          SCALAR,XVF2
C
C    The returned velocity components (U, V and W) are in the
C    coordinate system requested for initialisation.
C
C---------------------------------------------------------------------------

      Rmax = 2.9E-4
      Vavg = 0.0466
      Rlocal = (X**2.0+Y**2.0)**0.5
      Vmax = 2.0*Vavg
      W = Vmax*(1.0-(Rlocal/Rmax)**2)


      RETURN
      END


C***********************************************************************
      SUBROUTINE BCDEFI(SCALAR,U,V,W,TE,ED,T,DEN,TURINT,RSU,V2P,F2P)
C     Boundary conditions at inlets
C***********************************************************************
C--------------------------------------------------------------------------*
C     STAR-CD VERSION 4.06.000
C--------------------------------------------------------------------------*
      INCLUDE 'comdb.inc'
      COMMON/USR001/INTFLG(100)
      DIMENSION SCALAR(*)
      DIMENSION RSU(6)
      LOGICAL TURINT
      INCLUDE 'usrdat.inc'
      DIMENSION SCALC(50)
      EQUIVALENCE( UDAT12(001), ICTID )
      EQUIVALENCE( UDAT04(002), DENC )
      EQUIVALENCE( UDAT04(003), EDC )
      EQUIVALENCE( UDAT02(005), PR )
      EQUIVALENCE( UDAT04(005), PRC )
      EQUIVALENCE( UDAT04(009), SCALC(01) )
      EQUIVALENCE( UDAT04(007), TC )
      EQUIVALENCE( UDAT04(008), TEC )
      EQUIVALENCE( UDAT04(059), UC )
      EQUIVALENCE( UDAT04(060), VC )
      EQUIVALENCE( UDAT04(061), WC )
      EQUIVALENCE( UDAT04(064), UCL )
      EQUIVALENCE( UDAT04(065), VCL )
      EQUIVALENCE( UDAT04(066), WCL )
      EQUIVALENCE( UDAT02(070), X )
      EQUIVALENCE( UDAT02(071), Y )
      EQUIVALENCE( UDAT02(072), Z )
C-----------------------------------------------------------------------
C
C    This subroutine enables the user to specify INLET boundary
C    conditions for U,V,W,TE,ED,T,(V22,F22 for V2F model) and SCALAR.
C
C
C    ** Parameters to be returned to STAR: U,V,W,TE,ED,T,
C                                          SCALAR, DEN, TURINT
C
C    NB U,V and W are in the local coordinate-system of the
C    inlet boundary.
C
C-----------------------------------------------------------------------

C---- This subroutine gives the inlet fluid a parabolic profile.
C     It will be required to update this subroutine with different
C     geometries, flow conditions, and coordinate systems.
      Rmax = 2.9E-4
      Vavg = 0.0466
      Rlocal = (X**2.0+Y**2.0)**0.5
      Vmax = 2.0*Vavg
      W = Vmax*(1.0-(Rlocal/Rmax)**2)
      RETURN
      END


C***********************************************************************
      SUBROUTINE DIFFUS(DIF)
C     Diffusivity
C***********************************************************************
C----------------------------------------------------------------------*
C     STAR-CD VERSION 4.06.000
C----------------------------------------------------------------------*
      INCLUDE 'comdb.inc'
      COMMON/USR001/INTFLG(100)
      INCLUDE 'usrdat.inc'
      DIMENSION SCALAR(50), DENSC(50), VISSC(50)
      EQUIVALENCE( UDAT05(001), IDR )
      EQUIVALENCE( UDAT05(002), IDRT )
      EQUIVALENCE( UDAT06(004), TD )
      EQUIVALENCE( UDAT12(001), ICTID )
      EQUIVALENCE( UDAT10(001), DENSC(01) )
      EQUIVALENCE( UDAT10(051), VISSC(01) )
      EQUIVALENCE( UDAT11(001), CP )
      EQUIVALENCE( UDAT11(002), DEN )
      EQUIVALENCE( UDAT11(003), ED )
      EQUIVALENCE( UDAT11(006), P )
      EQUIVALENCE( UDAT11(007), T )
      EQUIVALENCE( UDAT11(008), TE )
      EQUIVALENCE( UDAT11(009), SCALAR(01) )
      EQUIVALENCE( UDAT11(016), SCALAR(08) )
      EQUIVALENCE( UDAT11(019), SCALAR(11) )
      EQUIVALENCE( UDAT11(059), U )
      EQUIVALENCE( UDAT11(060), V )
      EQUIVALENCE( UDAT11(061), W )
      EQUIVALENCE( UDAT11(062), VISM )
      EQUIVALENCE( UDAT11(063), VIST )
      EQUIVALENCE( UDAT11(067), X )
      EQUIVALENCE( UDAT11(068), Y )
      EQUIVALENCE( UDAT11(069), Z )
      EQUIVALENCE( UDAT09(001), IS )
      EQUIVALENCE( UDAT11(073), CON )
C-----------------------------------------------------------------------
C
C    This subroutine enables the user to specify the molecular
C    diffusivity (DIF) of species IS as an arbitrary function.
C    STAR calls this subroutine for cells and boundaries.
C
C    ** Parameter to be returned to STAR:  DIF
C
C    NOTE: This subroutine also defines the molecular diffusivity
C          in the Schmidt number and the Sherwood number used in
C          droplet mass transfer calculations. If this molecular
C          diffusivity is different to that used in the species
C          transport equations, then a condition is needed such as
C
C             IF(IDR.GT.0) THEN
C               DIF= value for droplet mass transfer calculation
C             ELSE
C               DIF= value for species transport equation
C             ENDIF
C
C-----------------------------------------------------------------------
C
      nBldSt = SCALAR(08)
      ShrRt = SCALAR(11)

C-- Calculate Diffusivity
C   For larger molecules, diffusivity is a function of the shear rate
C   For smaller molecules, it is constant
C     Unactivated & Activated Platelets
      IF (IS.EQ.1.OR.IS.EQ.2) THEN
        DIF = 6.02E-14*ShrRt + 1.58E-13
C     ADP -- neglecting the shear rate term due to small size
      ELSEIF (IS.EQ.3) THEN
```

```
        DIF = 2.57E-10
C     Thromboxane A2 -- neglecting the shear rate term due to small
C     size
      ELSEIF (IS.EQ.4) THEN
        DIF = 2.14E-10
C     Prothrombin
      ELSEIF (IS.EQ.5) THEN
        DIF = 6.02E-14*ShrRt + 3.32E-11
C     Thrombin
      ELSEIF (IS.EQ.6) THEN
        DIF = 6.02E-14*ShrRt + 4.16E-11
C     Antithrombin III
      ELSEIF (IS.EQ.7) THEN
        DIF = 6.02E-14*ShrRt + 3.49E-11
      ENDIF

      RETURN
      END


C*********************************************************************
      SUBROUTINE SCALFN(PHI)
C     Species-scalar function
C*********************************************************************
C--------------------------------------------------------------------*
C     STAR-CD VERSION 4.06.000
C--------------------------------------------------------------------*
      INCLUDE 'comdb.inc'
      COMMON/USR001/INTFLG(100)
      INCLUDE 'usrdat.inc'
      DIMENSION SCALAR(50), HFORM(50)
      EQUIVALENCE( UDAT12(001), ICTID )
      EQUIVALENCE( UDAT03(001), CON )
      EQUIVALENCE( UDAT03(009), DUDX )
      EQUIVALENCE( UDAT03(010), DVDX )
      EQUIVALENCE( UDAT03(011), DWDX )
      EQUIVALENCE( UDAT03(012), DUDY )
      EQUIVALENCE( UDAT03(013), DVDY )
      EQUIVALENCE( UDAT03(014), DWDY )
      EQUIVALENCE( UDAT03(015), DUDZ )
      EQUIVALENCE( UDAT03(016), DVDZ )
      EQUIVALENCE( UDAT03(017), DWDZ )
      EQUIVALENCE( UDAT04(001), CP )
      EQUIVALENCE( UDAT04(002), DEN )
      EQUIVALENCE( UDAT04(003), ED )
      EQUIVALENCE( UDAT04(006), P )
      EQUIVALENCE( UDAT04(008), TE )
      EQUIVALENCE( UDAT04(009), SCALAR(01) )
      EQUIVALENCE( UDAT04(059), U )
      EQUIVALENCE( UDAT04(060), V )
      EQUIVALENCE( UDAT04(061), W )
      EQUIVALENCE( UDAT04(062), VISM )
      EQUIVALENCE( UDAT04(063), VIST )
      EQUIVALENCE( UDAT04(007), T )
      EQUIVALENCE( UDAT04(067), X )
      EQUIVALENCE( UDAT04(068), Y )
      EQUIVALENCE( UDAT04(069), Z )
      EQUIVALENCE( UDAT09(001), IS )
      EQUIVALENCE( UDAT10(101), HFORM(01) )
C-----------------------------------------------------------------------
C
C    This subroutine enables the user to specify SCALAR(IS) in an
C    arbitrary manner, instead of solving the corresponding transport
C    equation.
C
C    ** Parameter to be returned to STAR: PHI
C
C-----------------------------------------------------------------------
C---- Re-define the mass fractions of the various scalars with helpful
C     names:
      REAL ShrRt

C---- Calculate the Maximum Shear Rate from dU/dX, etc. for each cell
C     (used in diffus.f,posdat.f)
      IF (IS.EQ.11) THEN
      ADUDX = ABS(DUDX)
      ADUDY = ABS(DUDY)
      ADUDZ = ABS(DUDZ)
      ADVDX = ABS(DVDX)
      ADVDY = ABS(DVDY)
      ADVDZ = ABS(DVDZ)
      ADWDX = ABS(DWDX)
      ADWDY = ABS(DWDY)
      ADWDZ = ABS(DWDZ)
      ShrRt = MAX(ADUDX,ADVDX,ADWDX,ADUDY,
     &ADVDY,ADWDY,ADUDZ,ADVDZ,ADWDZ)
      PHI = ShrRt
      ENDIF

      RETURN
      END


C***********************************************************************
      SUBROUTINE SORSCA(S1P,S2P)
C     Source-term for scalar species
C***********************************************************************
C----------------------------------------------------------------------*
C     STAR-CD VERSION 4.06.000
C----------------------------------------------------------------------*
      INCLUDE 'comdb.inc'
      COMMON/USR001/INTFLG(100)
      INCLUDE 'usrdat.inc'
      DIMENSION SCALAR(50)
      EQUIVALENCE( UDAT12(001), ICTID )
      EQUIVALENCE( UDAT03(001), CON )
      EQUIVALENCE( UDAT03(002), TAU )
      EQUIVALENCE( UDAT03(009), DUDX )
      EQUIVALENCE( UDAT03(010), DVDX )
      EQUIVALENCE( UDAT03(011), DWDX )
      EQUIVALENCE( UDAT03(012), DUDY )
      EQUIVALENCE( UDAT03(013), DVDY )
      EQUIVALENCE( UDAT03(014), DWDY )
      EQUIVALENCE( UDAT03(015), DUDZ )
      EQUIVALENCE( UDAT03(016), DVDZ )
      EQUIVALENCE( UDAT03(017), DWDZ )
      EQUIVALENCE( UDAT03(019), VOLP )
      EQUIVALENCE( UDAT04(001), CP )
      EQUIVALENCE( UDAT04(002), DEN )
      EQUIVALENCE( UDAT04(003), ED )
      EQUIVALENCE( UDAT04(004), HP )
      EQUIVALENCE( UDAT04(006), P )
      EQUIVALENCE( UDAT04(008), TE )
      EQUIVALENCE( UDAT04(009), SCALAR(01) )
      EQUIVALENCE( UDAT04(010), SCALAR(02) )
      EQUIVALENCE( UDAT04(011), SCALAR(03) )
      EQUIVALENCE( UDAT04(012), SCALAR(04) )
      EQUIVALENCE( UDAT04(013), SCALAR(05) )
      EQUIVALENCE( UDAT04(014), SCALAR(06) )
      EQUIVALENCE( UDAT04(015), SCALAR(07) )
      EQUIVALENCE( UDAT04(016), SCALAR(08) )
      EQUIVALENCE( UDAT04(017), SCALAR(09) )
      EQUIVALENCE( UDAT04(018), SCALAR(10) )
      EQUIVALENCE( UDAT04(019), SCALAR(11) )
      EQUIVALENCE( UDAT04(020), SCALAR(12) )
      EQUIVALENCE( UDAT04(021), SCALAR(13) )
      EQUIVALENCE( UDAT04(022), SCALAR(14) )
      EQUIVALENCE( UDAT04(023), SCALAR(15) )
      EQUIVALENCE( UDAT04(024), SCALAR(16) )
      EQUIVALENCE( UDAT04(025), SCALAR(17) )
      EQUIVALENCE( UDAT04(026), SCALAR(18) )
      EQUIVALENCE( UDAT04(027), SCALAR(19) )
      EQUIVALENCE( UDAT04(028), SCALAR(20) )
      EQUIVALENCE( UDAT04(029), SCALAR(21) )
      EQUIVALENCE( UDAT04(030), SCALAR(22) )
      EQUIVALENCE( UDAT04(059), U )
      EQUIVALENCE( UDAT04(060), V )
      EQUIVALENCE( UDAT04(061), W )
      EQUIVALENCE( UDAT04(062), VISM )
      EQUIVALENCE( UDAT04(063), VIST )
      EQUIVALENCE( UDAT04(007), T )
      EQUIVALENCE( UDAT04(067), X )
      EQUIVALENCE( UDAT04(068), Y )
      EQUIVALENCE( UDAT04(069), Z )
      EQUIVALENCE( UDAT09(001), IS )
C----------------------------------------------------------------------
C
C    This subroutine enables the user to specify source terms (per unit
C    volume) for species in linearized form:
C
C        Source = S1P-S2P*SCALAR(IS), (kg/sm3)
C
C    in an arbitrary manner.
C
C    If the species is to be fixed to a given value SCI, then the
C    following may be used:
C
C        S1P=GREAT*SCI
C        S2P=GREAT,
C**********
```

```
C          Must ACTUALLY use the GREAT term in these equations for it
to work!!!
C**********
C    where SCI can be a constant or an arbitrary function of the
C    parameters in parameter list.
C-----------------------------------------------------------------------
C     Sample coding 1: Fix the mass concentration of scalar 1 in fluid
2
C                     to a constant value SCALAR(1)=.75
C
C      IF(IS.EQ.1.AND.IMAT.EQ.2) THEN
C        S1P=GREAT*.75
C        S2P=GREAT
C      ENDIF
C
C-----------------------------------------------------------------------
C    ** Parameters to be returned to STAR:  S1P,S2P
C
C-----------------------------------------------------------------------
C     For Blood Status (nBldSt):
C     0 = interior fluid
C     1 = wall fluid
C     2 = wall thrombus
C     3 = interior thrombus
C     4 = interior fluid with thrombus neighbor
C     5 = wall fluid with thrombus neighbor
C
C     M = concentration of platelets on the surface (Pl/m2)
C     Mact = concentration of activated platelets on the surface
C     (Pl/m2)
C     Sp = 1-M/Minfinity = fraction of surface not covered by platelets
C     Xact = fraction of activated platelets on surface

C---- Re-define the mass fractions of the various scalars with helpful
C     names:
      yPl = SCALAR(01)
      yaPl = SCALAR(02)
      yADP = SCALAR(03)
      yTxA2 = SCALAR(04)
      ypT = SCALAR(05)
      yT = SCALAR(06)
      yaT = SCALAR(07)
      nBldSt = SCALAR(08)
      ThrFr = SCALAR(09)
      actT = SCALAR(10)
      ShrRt = SCALAR(11)
      newThr = SCALAR(12)
      NFcBnd = SCALAR(13)
      NFcThr = SCALAR(14)
      xM = SCALAR(15)
      xMact = SCALAR(16)
      Xact = SCALAR(17)
      Sp = SCALAR(18)
      PlSrc = SCALAR(19)
      aPlSrc = SCALAR(20)
      pulse = SCALAR(21)
      nPlsFl = SCALAR(22)
```

```
C---- Calculate the Ratio of Volume of the Cell / Area of a Face
C     Assumes a regular tetrahedral mesh -- even if not exact, good
C     enough approx.
C     Used to Convert Fluxes (mass/area*time) to source terms
C     (mass/volume*time)
C     VOLP comes from usrdat.inc (see nom.inc)
C     --This one is for tetrahedral meshes
c      AtoV = 0.555*VOLP**(1./3.)
C     --The one is for flat cells near boundary
c      AtoV = 4.0E-6
C     --This one is for hexahedral meshes
      AtoV = VOLP**(1./3.)


C---- Initialize the source terms (S1P & S2P) to 0.0
      S1P = 0.0
      S2P = 0.0

C---- Define the basic source term
C     From pgs. 59-60 of Paul's dissertation:
C     omega = sum (w.j *(a.j/a.j,crit)
C     if omega < 1.0, k.pa =0
C     else k.pa = omega/t.act [call k.pa ratec]
C
C     the platelet source term "Pl_src" below = k.ap * [Pl]

      tact = 1.0
      ratec = 0.0
      omega = (1.0E06)*(1.244*yADP + 16.48*yTxA2 + 62.79*yT)
      IF (omega.GE.1) THEN
      ratec = omega/tact
      ENDIF
      Pl_src = ratec*yPl*1056.

C---- Unactivated Platelet
      IF (IS.EQ.1) THEN
C     In cells other than thrombus... see posdat.f for calculation of
C     PlSrc
      S1P = -Pl_src + PlSrc
C       In cell that is wall thrombus (2) or interior thrombus (3)
      IF (nBldSt.EQ.2.OR.nBldSt.EQ.3) S1P = 0.0

C---- Activated Platelet
      ELSE IF (IS.EQ.2) THEN
C     In cells that are not thrombus... see posdat.f for calculation of
C     aPlSrc
      S1P = Pl_src + aPlSrc

C       In cell that is wall thrombus (2) or interior thrombus (3)
      IF (nBldSt.EQ.2.OR.nBldSt.EQ.3) S1P = 0.0

C---- ADP (see pg. 21 of AC and pg. 59-61 of D) -- Same no matter where
C     source(ADP) = +lambda*kpa*[Pl]
C       lambda = 0.0046 kg_ADP/kg_platelet-- pg. 21 of AC
      ELSE IF (IS.EQ.3) THEN
```

```
C     ADP released from agonist-induced activating platelets
C        = lambda*kpa*[Pl], lambda = 2.4*10^-8 nmol ADP/platelet
C         (in kg ADP/m3*s), = lambda*(mol/nmol)*(platelet/kg
C          platelet)*(kg ADP/mol ADP)
C       (mol/nmol) =  10^-9, platelet/kg = 1/2.12E-15, kg/mol = 0.4244
      S1P = 2.4E-8*1.0E-9*(1.0/2.12E-15)*0.4244*Pl_src
C     ADP released from shear-induced activating platelets
C        = lambda*(M-Ma) when Hellum's activation time is reached, M-Ma
C          is platelets that activate
C         (in kg ADP/m3*s), = lambda*(mol/nmol)*(kg ADP/mol
C          ADP)*(Area/Volume)*(1/time step)*(M-Ma)
C      (mol/nmol) =  10^-9, kg/mol = 0.4244, time step = DT, Area/Volume
C       = 1/AtoV, M-Ma = pulse
      IF (nPlsFl.EQ.1) THEN
        S1P = S1P + pulse*2.4E-8*1.0E-9*0.4244*(NFcBnd/AtoV)/DT
      ENDIF
C     IF(MOD(IPSTAR,500).EQ.0.0)  write(*,*) IS, IPSTAR, S1P

C---- Thromboxane A2 (see pg. 21 of AC and pgs. 59,61 of D)
C     source(TxA2) = +st[aPl]-ki[TxA2]  ---> need to convert to units
C     kg TxA2/m3*s
C                = st(mol TxA2/platelet*s)*(kg TxA2/mol
C                  TxA2)*(platelet/mass)*(mass blood/volume)*(mass
C                  platelet/mass blood)
C                  - ki*(mass blood/volume)*(mass TxA2/mass blood)
C                  st = 9.51*10^-21 mol TxA2/platelet*s, kg TxA2/mol
C                 TxA2 = 0.3525 kg/mol, platelet/mass = 1/2.212*10^-15,
C                  mass blood/vol = 1056. kg/m3, mass platelet/mass
C                  blood = yaPl
C                  ki = 0.0161*s^-1, mass TxA2/mass blood = 1056.
      ELSE IF (IS.EQ.4) THEN
      S1P = 9.51E-21*0.3525*(1./2.12E-15)*1056.*yaPl
     &- 0.0161*1056.*yTxA2
C       In wall fluid (1) or wall fluid with thrombus neighbor (5),
C       Add the boundary terms to the bulk terms, i.e. S1P = source +
C       boundary term,
C       Multiply boundary term by number of faces on a boundary
C       (NFcBnd), and
C       Divide boundary term by Volume/Area ratio (AtoV) to convert
C       from flux to volumetric source
      IF (nBldSt.EQ.1.OR.nBldSt.EQ.5) THEN
C     kg TxA2/m2*s = Ma*st*(kg TxA2/mol TxA2)
C                 st = 9.5*10^-21 mol TxA2/platelet*s, Ma = activated
C                 platelet/m2, kg TxA2/mol TxA2 = 0.3525 kg/mol
        S1P = S1P + (xM*Xact*9.5E-21*0.3525)*NFcBnd/AtoV
C       In cell that is wall thrombus (2) or interior thrombus (3)
C     source(TxA2) mol/mL*s = st[aPl], [aPl] = [platelet/mL]
C                kg/m3*s = st*(vol pl/vol thrombus)*(platelet/vol
C                platelet)*(kg TxA2/mol TxA2)*(10^6 ml/m3)
      ELSE IF (nBldSt.EQ.2.OR.nBldSt.EQ.3) THEN
        S1P = 9.5E-21*0.3525*(1./2.011E-12)*1.0E6*0.9
      ENDIF
C     IF(MOD(IPSTAR,500).EQ.0.0)  write(*,*) IS,IPSTAR,S1P

C---- Prothrombin (see pg. 21 of AC and pgs. 59,61-62 of D)
      ELSE IF (IS.EQ.5) THEN
C     source(pT) mol pT/m3*s = -Beta*[pT](Fi_at[aPl]+Fi_rt[Pl])
```

```
C           Fi_at (units*m3/plt*s*mol pT) = 3.69*10^-6, Fi_rt
C                 (units*m3/plt*s*mol pT) = 6.5*10^-7
C                  Beta = 9.11*10^-12 mol pT/unit,
C           "        kg pT/m3*s = -Beta*(mass pT/mol pT)*(mass pT/mass
C                    blood)*(mass blood/vol)*(mol pT/mass pT)
C                       *[Fi_at*(mass aPl/mass blood)*(mass
C                       blood/vol)*(platelet/mass platelet) +
C                         Fi_rt*(mass Pl/mass blood)*(mass
C                       blood/vol)*(platelet/mass platelet)]
C                         mass pT/mass blood = ypT, mass blood/vol =
C                         1056. kg/m3, mass aPl/mass blood = yaPl,
C                         mass Pl/mass blood = yPl, platelet/mass
C                         platelet = 1./2.12*10^-15 kg^-1,
      S1P = -9.11E-12*1056.*ypT*(3.69E-06*yaPl + 6.5E-07*yPl)*
     &1056.*(1./2.12E-15)
C       In wall fluid (1) or wall fluid with thrombus neighbor (5),
C       Add the boundary terms to the bulk terms, i.e. S1P = source +
C       boundary term,
C       Multiply boundary term by number of faces on a boundary
C       (NFcBnd), and
C       Divide boundary term by Volume/Area ratio (AtoV) to convert
C       from flux to volumetric source
      IF (nBldSt.EQ.1.OR.nBldSt.EQ.5) THEN
C       boundary source (kg/m2*s) ~= -Fi_at*Ma*[pT]
C                      Fi_at = (3.69*10^-9 units/plt*s*microM
C                       pT)*(9.11*10^-6 micromol pT/unit)(1 m3/1000*L)
C                       = 3.36*10^-17 m3/plt*s
C                      Ma = activated platelet/m2, [pT] = (mass pT/mass
C                       blood)*(mass blood/vol) = ypT*1056. kg pT/m3
C                      Must then convert to volumetric source term
        S1P = S1P - (xMact*ypT*3.36E-17*1056.)*NFcBnd/AtoV
C       In cell that is wall thrombus (2) or interior thrombus (3)
      ELSE IF (nBldSt.EQ.2.OR.nBldSt.EQ.3) THEN
C       source(pT) kg pT/m3*s = same as above, except using assumption
C       yaPl = 1.0 and yPl = 0.0 in thrombus
C                 -9.11E-12*1056.*3.69E-6*1056.*(1/2.12E-15) = 1.77E04
        S1P = -1.77E04*ypT*Xact
      ENDIF
C     IF(MOD(IPSTAR,500).EQ.0.0)  write(*,*) IS,IPSTAR,S1P

C---- Thrombin in Bulk (see pg. 22 of AC and pgs. 59,61-62 of D)
      ELSE IF (IS.EQ.6) THEN
C       source(T) = [pT](fi_at[aPl]+Fi_rt[Pl])-Gamma[T]
C     tgen (kg/m3*s) = Beta*[pT](Fi_at[aPl]+Fi_rt[Pl])*(mass
C     thrombin/mass prothrombin)
C                  = (see prothrombin source for variable definitions),
C                     mass thrombin/mass prothrombin = 0.508
C
      tgen = 4.792E09*ypT*(3.69E-06*yaPl+6.5E-07*yPl)*0.508
C     tloss = Gamma*[T]
C          where Gamma =
C     k1t*[H]*[aT]/(alpha*Kat*Kt+alpha*Kat*[T]+alpha*Kt*[aT]+[aT]*[T]
C               k1t = 13.33 s^-1, [H]=2.08*10^-7 kgmol/m3, alpha = 1.0,
C               Kat = 10^-7 kgmol/m3, Kt=3.6*10^-8 kgmol/m3
C                k1t*[H]*[aT]*[T] (kg T/m3*s) = k1t*[H]*(mass
C     blood/vol)*(mass aT/mass blood) = k1T*[H]*1056. kg/m3*yT*[aT]
C                [aT] (kgmol/m3) = (mass aT/mass blood)*(mass
C     blood/vol)*(mol aT/mass aT) = yaT*1056. kg/m3*1/62000 kgmol/kg
      tloss = (49.8*yaT*yT)/(3.6E-09 + 2.89E-03*yT
     &+ 6.12E-04*yaT + 491.*yaT*yT)
      S1P = tgen - tloss
C       In wall fluid (1) or wall fluid with thrombus neighbor (5),
      IF (nBldSt.EQ.1.OR.nBldSt.EQ.5) THEN
C       boundary source (kg/m2*s) ~= Fi_at*Ma*[pT]
C                      Fi_at = (3.69*10^-9 units/plt*s*microM
C pT)*(9.11*10^-6 micromol pT/unit)(1 m3/1000*L) = 3.36*10^-17 m3/plt*s
C                      Ma = activated platelet/m2, [pT] = (mass pT/mass
C     blood)*(mass blood/vol) = ypT*1056. kg/m3
C                      Must then convert to volumetric source term
        S1P = S1P + (xMact*ypT*3.36E-17*1056.)*NFcBnd/AtoV
C       In cell that is wall thrombus (2) or interior thrombus (3)
      ELSE IF (nBldSt.EQ.2.OR.nBldSt.EQ.3) THEN
C       tgen = Positive value of the Prothrombin source, but multiplied
C       by 0.508 (MW_t/MW_pt)
        tgen = 8.973E03*ypT*Xact
C       tloss = same as tloss above (Gamma*[T])
        tloss = (49.8*yaT*yT)/(3.6E-09 + 2.895E-03*yT
     &+ 6.12E-04*yaT + 491.*yaT*yT)
        S1P = tgen - tloss
      ENDIF
C     IF(MOD(IPSTAR,500).EQ.0.0)  write(*,*) IS,IPSTAR,S1P

C---- antiThrombin III (see pg. 22 of AC and pgs. 59,61-62 of D) -
C  Same no matter where
C     source(aT) = -Gamma*[T]    (84.077 =
C     0.047*1056*mass_antithrombinIII/mass_thrombin)
C          where Gamma =
C     k1t*[H]*[aT]/(alpha*Kat*Kt+alpha*Kat*[T]+alpha*Kt*[aT]+[aT]*[T]
C               k1t = 13.33 s^-1, [H]=2.08*10^-7 kgmol/m3, alpha = 1.0,
C               Kat = 10^-7 kgmol/m3, Kt=3.6*10^-8 kgmol/m3
C                k1t*[H]*[aT]*[T] (kg aT/m3*s) = k1t*[H]*(mass
C     blood/vol)*(mass aT/mass blood) = k1T*[H]*1056. kg/m3*yaT*[T]
C                [T] (kgmol/m3) = (mass T/mass blood)*(mass
C           blood/vol)*(mol T/mass T) = yT*1056. kg/m3*1/36600 kgmol/kg
      ELSE IF (IS.EQ.7) THEN
      S1P = (-84.6*yaT*yT)/(3.6E-09 + 2.89E-03*yT
     &+ 6.12E-04*yaT + 491.*yaT*yT)
C     IF(MOD(IPSTAR,500).EQ.0.0)  write(*,*) IS, IPSTAR, S1P

      ENDIF

      RETURN
      END


C***********************************************************************
      SUBROUTINE VISMOL(VISM)
C     Viscosity (molecular)
C***********************************************************************
C----------------------------------------------------------------------*
C     STAR-CD VERSION 4.06.000
C----------------------------------------------------------------------*
      INCLUDE 'comdb.inc'
      COMMON/USR001/INTFLG(100)
      INCLUDE 'usrdat.inc'
      DIMENSION SCALAR(50)
      EQUIVALENCE( UDAT12(001), ICTID )
      EQUIVALENCE( UDAT03(009), DUDX )
      EQUIVALENCE( UDAT03(010), DVDX )
      EQUIVALENCE( UDAT03(011), DWDX )
      EQUIVALENCE( UDAT03(012), DUDY )
      EQUIVALENCE( UDAT03(013), DVDY )
      EQUIVALENCE( UDAT03(014), DWDY )
      EQUIVALENCE( UDAT03(015), DUDZ )
      EQUIVALENCE( UDAT03(016), DVDZ )
      EQUIVALENCE( UDAT03(017), DWDZ )
      EQUIVALENCE( UDAT03(018), SECINV )
      EQUIVALENCE( UDAT11(001), CP )
      EQUIVALENCE( UDAT11(002), DEN )
      EQUIVALENCE( UDAT11(006), P )
      EQUIVALENCE( UDAT11(007), T )
      EQUIVALENCE( UDAT11(009), SCALAR(01) )
      EQUIVALENCE( UDAT11(017), SCALAR(09) )
      EQUIVALENCE( UDAT11(059), U )
      EQUIVALENCE( UDAT11(060), V )
      EQUIVALENCE( UDAT11(061), W )
      EQUIVALENCE( UDAT11(067), X )
      EQUIVALENCE( UDAT11(068), Y )
      EQUIVALENCE( UDAT11(069), Z )
C-----------------------------------------------------------------------
C
C    This subroutine enables the user to specify the molecular
C    viscosity
C    (VISM) in an arbitrary manner for boundaries and cells.
C
C      IFLUTYP - (Free surface only) Light/Heavy fluid flag
C                IFLUTYP=1 for light fluid
C                IFLUTYP=2 for heavy fluid
C
C    ** Parameter to be returned to STAR: VISM
C
C-----------------------------------------------------------------------
C
C    Sample coding: To calculate viscosity from the 'power law'
C                   constitutive relation for Non-Newtonian flow (as in
C                   the standard STAR coding)
C
C      EM=0.001
C      EN=2.
C      VISM=EM*(ABS(SECINV))**(0.5*(EN-1.))
C-----------------------------------------------------------------------


C---- Change the viscosity of a cell depending on whether or not it is
C     a thrombus
      ThrFr = SCALAR(09)

      IF (ThrFr.GE.0.9999) THEN
        VISM = 1.0
      ENDIF
      RETURN
      END


c**********************************************************************
      subroutine posdat(level)
c     Post-process data
C**********************************************************************
C---------------------------------------------------------------------*
C     STAR-CD VERSION 4.06.000
C---------------------------------------------------------------------*
      USE allmod
      IMPLICIT NONE
      INCLUDE 'std.inc'
c
c     Argument variables
      INTEGER level
C----------------------------------------------------------------------
C
C     This subroutine enables the user to output data and is called
C     at the beginning and at the end of each iteration/time step,
C     i.e.
C          if (level.eq.1) then
c.>          called at the beginning of iteration/time step
C          else if (level.eq.2) then
c.>          called at the end of iteration/time step
C          end if
C          Any user code which is not enclosed in the IF condition will
C          be executed for both calls
C
C     Note: 1. File units available to the users for opening their own
C              files are from 84 to 89. Users may write to unit 6 or 60
C              if they want to see their output on the terminal or
C              the run file.
C           2. All variables passed to this routine use STAR cell
C               numbering, which is different from pro-STAR cell numbers.
C              pro-STAR cell number can be obtained from a STAR cell
C              number ICSTAR by ICPROSTAR=ICLMAP(ICSTAR)
C----------------------------------------------------------------------
C
C     Sample coding: (a) To write values of U-velocity component,
absolute
C                        pressure and temperature at 5 specified points
C                        to a file at each time step (for plotting).
C
C                    (b) To calculate average temperature at all walls
C                        and in cell next to walls in domain 1.
C
C                    (c) To calculate and print mass averaged
concentra-
C                        tion of SC1 at the end of the run into the
C                        run file.
C----------------------------------------------------------------------
C     For Blood Status (nBldSt):
C       0 = interior fluid
C       1 = wall fluid
C       2 = wall thrombus
```

```
C       3 = interior thrombus
C       4 = interior fluid with thrombus neighbor
C       5 = wall fluid with thrombus neighbor
C
C       M = concentration of platelets on the surface (Pl/m2)
C       Mact = concentration of activated platelets on the surface
C       (Pl/m2)
C       Sp = 1-M/Minfinity = fraction of surface not covered by
platelets
C       Xact = fraction of activated platelets on surface
C
C Dimension variables to be used
      REAL(ra) AtoV,xMinf
      REAL(ra) rpadh,rpadhm,apadh,apadhm,upemb,upembm
      REAL(ra) deltha,delthu,uthromb,athromb

      INTEGER iyPl,iyaPl,inBldSt,iThrFr,iactT,iShrRt,inewThr,iNFcBnd,
     &iNFcThr,ixM,ixMact,iXact,iSp,iPlSrc,iaPlSrc,ipulse,inPlsFl

      INTEGER ir,nset,IS,IC,nd,NBRCLL,fset,j,jf

      iyPl = 1
      iyaPl = 2
      inBldSt = 8
      iThrFr = 9
      iactT = 10
      iShrRt = 11
      inewThr = 12
      iNFcBnd = 13
      iNFcThr = 14
      ixM = 15
      ixMact = 16
      iXact = 17
      iSp = 18
      iPlSrc = 19
      iaPlSrc = 20
      ipulse = 21
      inPlsFl = 22

C Stuff to do at the beginning of the time step (Level = 1)
      IF (level.EQ.1) THEN
C In the first time step, set-up the Blood Status for cells next to a
C wall.
C Make sure that this stuff is only done once
      IF (INTFLG(20).EQ.0) THEN
        INTFLG(20) = 1
        INTFLG(21) = -1
C       Set the number of wall boundary faces on a cell

        do ir=0,regi_no
            if (regi(ir)%type.eq.WALL) then
            call FsetBoun(fs,0,ND_ALL,NSD_ALL,ir,ISIDE_ALL)
            do nset=1,fs%no
            do IS=fs%ns(nset), fs%ne(nset)
              IC = lfc(1,IS)
              c(IC,inBldSt) = 1.0
              c(IC,iNFcBnd) = c(IC,iNFcBnd) + 1.0
            enddo
            enddo
          endif
        enddo
      ENDIF
      IF (INTFLG(21).NE.ITER) THEN
C---- Loop through all of the cells to do calculations at the beginning
C     of the time step
        DO nd=1,doma_no
        if (doma(nd)%mattyp.eq.FLUID) then
          call cset(cs,0,nd,NSD_ALL,INTERNAL)
          do nset=1,cs%no
            do IC=cs%ns(nset),cs%ne(nset)
        delthu = 0.0
        deltha = 0.0
        c(IC,iPlSrc) = 0.0
        c(IC,iaPlSrc) = 0.0
C---- Calculate the activation time for each cell, if it hasn't already
C     been calculated
C     Only calculate it in cells next to the wall, or next to uthrombs
C     Uses Hellum's equation (see pg. 73 of dissertation) where 0.035
C     is viscosity of blood in poise.
          IF (c(IC,inBldSt).EQ.1.0.OR.c(IC,inBldSt).EQ.4.0) THEN
          IF (c(IC,iShrRt).GT.0.0.AND.c(IC,iactT).EQ.0.0) THEN
            c(IC,iactT) = TIME + 4.0E6*(c(IC,iShrRt)*0.035)**(-2.3)
          ENDIF
        ENDIF

C---- Calculate the "pulse" of activated platelets when activation time
C     is reached and set the flag that it has been reached
        IF (TIME.GT.c(IC,iactT).AND.c(IC,iactT).GT.0.0) THEN
          IF (c(IC,inPlsFl).EQ.0.0) THEN
            c(IC,ipulse) = c(IC,ixM) - c(IC,ixMact)
            c(IC,inPlsFl) = 1.0
          ENDIF
          c(IC,ixMact) = c(IC,ixM)
          c(IC,iXact) = 1.0
        ENDIF

C---- Calculate the Ratio of Volume of the Cell / Area of a Face
C     Assumes a regular tetrahedral mesh -- even if not exact, good
C     enough approx.
C     Used to Convert Fluxes (mass/area*time) to source terms
C     (mass/volume*time)
C     VOLP comes from usrdat.inc (see nom.inc)
C     --This one is for tetrahedral meshes
c       AtoV = 0.555*VOL(IC)**(1./3.)
C     --This one is for flat cells near boundary
c       AtoV = 4.0E-6
C     --This one is for hexahedral meshes
        AtoV = VOL(IC)**(1./3.)

C---- Define and check the xMinf, Sp, xM, xMact,Xact terms
C       xMinf = maximum platelet surface coverage (platelets/m2)
        xMinf = 6.0E10

C---- Calculate source terms for Pl, aPl--store as scalars for use in
```

```
C     sorsca.f
C---- If cell is next to a wall
        IF (c(IC,inBldSt).EQ.1.0.OR.c(IC,inBldSt).EQ.5.0) THEN
          uthromb = 0.0
          athromb = 0.0
          upembm = 0.0
          upemb = 0.0
          rpadhm = 0.0
          rpadh = 0.0
          apadhm = 0.0
          apadh = 0.0
C---- Define the upemb, rpadhm, apadhm, thromb terms
C     Calculate flux of resting platelets embolizing FROM the wall
C     upemb (plt/m2*s) = (frac. rp embolized/time)*(conc. of resting
C     platelet on surface)
C     frac. rp embolized/time = (1-Femb)/(120 s) -- see
C     D, p. 77, and AC, pg. 23
C                    conc. of resting platelet on surface = M*(1-Xa)
C     upembm (kg/m2*s) = upemb(plt/m2*s)*(mass platelet/platelet)
C                    mass platelet/platelet = 2.12*10^-15 kg/plt
          upemb = ((1-EXP(-0.000309*c(IC,iShrRt)))/120.0)*c(IC,ixM)*
     &(1-c(IC,iXact))
          upembm = upemb*2.12E-15
C       Calculate the flux of resting platelets adhering TO the wall
C     rpadhm (kg/m2*s) = -S*krs*[Pl]
C        S (plt/m2), krs = 0.0000025 m/s, [Pl] (kg/m3) =
C     (mass Pl/mass blood)*(mass blood/vol blood)
C        mass Pl/mass blood = yPl, mass blood/vol blood = 1056 kg/m3,
C       rpadh (plt/m2*s) = rpadhm*(platelet/mass platelet)
C                      platelet/mass platelet = 1/2.12*10^-15 kg^-1
          rpadhm = c(IC,iSp)*0.0000025*c(IC,iyPl)*1056.
          rpadh = rpadhm*(1.0/2.12E-15)
C     Calculate the flux of active platelets adhering TO the wall
C     apadhm (kg/m2*s) = -S*kas*[aPl]
C        S (pl/m2), kas = 0.0035 m/s, [aPl] (kg/m3) = (kg platelet/kg
C           blood)*(kg blood/vol)
C        kg platelet/kg blood = yaPl, kg blood/vol = 1056. kg/m3
C     apadh (plt/m2*s) = apadhm*(platelet/mass platelet)
C           platelet/mass platelet = 1.0 platelet/2.12E-15 kg,
          apadhm = 0.0035*1056.*c(IC,iyaPl)*c(IC,iSp)
          apadh = apadhm*(1.0/2.12E-15)
C       Calculate the mass of unactivated platelets adhering to
C       activated platelets on the surface (i.e. forming uthrombs)
C     uthromb (kg/m2*s) = -Ma/Minf*kt*(mass platelet/mass
C     blood)*(mass blood/volume)
C          kt = 3.5E-3 m/s, mass platelet/mass blood = yaPl, mass
C           blood/volume = 1056 kg/m3
          uthromb = c(IC,ixMact)/xMinf*3.5E-3*1056.*c(IC,iyPl)
C     Calculate the mass of activated platelets adhering to activated
C     platelets on the surface
C     athromb (kg/m2*s) = -Ma/Minf*kt*(mass platelet/mass
C     blood)*(mass blood/volume)
C       kt = 3.5E-3 m/s, mass platelet/mass blood = yaPl, mass
C       blood/volume = 1056 kg/m3
          athromb = c(IC,ixMact)/xMinf*3.5E-3*1056.*c(IC,iyaPl)
C       Calculate the source terms used in sorsca.f (i.e. S1P)
C       Convert to a volumetric source from flux by multiplying by
```

```
C       number of faces on a boundary (NFcBnd),
C       and dividing by Volume/Face Area ratio (AtoV)
          c(IC,iPlSrc) = (upembm - rpadhm - uthromb)*
     &c(IC,iNFcBnd)/AtoV
          c(IC,iaPlSrc) = -(apadhm + athromb)*c(IC,iNFcBnd)/AtoV
C       Calculate the change in volumetric fraction of the cell
C       occupied by uthrombs
C       0.00472 comes from vol/mass of platelet: 10.2 fL and 2.12 pg
C       0.8 comes from assumption of 20% void volume in a uthrombs
          delthu = delthu + uthromb*c(IC,iNFcBnd)/AtoV*0.00472/0.8
          deltha = deltha + athromb*c(IC,iNFcBnd)/AtoV*0.00472/0.8
        ENDIF
C       In interior fluid with uthrombs neighbor (4) or wall with
C       uthrombs neighbor (5),
C       Add the uthrombs contribution to the bulk terms, i.e. S1P =
C       source + uthrombs term,
C       Multiply uthrombs term by number of faces on a uthrombs
C       (NFcThr)
        IF (c(IC,inBldSt).EQ.4.0.OR.c(IC,inBldSt).EQ.5.0) THEN
          uthromb = 0.0
          athromb = 0.0
C       Calculate the mass of unactivated platelets adhering to
C       activated platelets on the uthrombs
C       thromb (kg/m2*s) = -Xact*kt*(mass platelet/mass blood)*(mass
C       blood/volume)
C         kt = 3.5E-3 m/s, mass platelet/mass blood = yaPl, mass
C         blood/volume = 1056 kg/m3
C       --Paul uses Xact here instead of Ma/Minf (see above).... why?
C       Convert to a volumetric source from flux by multiplying by
C       number of faces on a uthrombs (NFcThr),
C         and dividing by Volume/Face Area ratio (AtoV)
          uthromb = c(IC,iXact)*0.0035*1056.*c(IC,iyPl)
C       athrombs growth (kg/m^3) = Xact*kt*yPl*density of
C       blood*area/volume
          athromb = c(IC,iXact)*0.0035*1056.*c(IC,iyaPl)
C       Re-Calculate the source terms used in sorsca.f (i.e. S1P)
          c(IC,iPlSrc) = c(IC,iPlSrc) - uthromb*c(IC,iNFcThr)/AtoV
          c(IC,iaPlSrc) = c(IC,iaPlSrc) - athromb*c(IC,iNFcThr)/AtoV
C       Calculate the change in volumetric fraction of the cell
C       occupied by uthrombs
C       0.00472 comes from vol/mass of platelet: 10.2 fL and 2.12 pg
C       0.8 comes from assumption of 20% void volume in a uthrombs
          delthu = delthu + uthromb*c(IC,iNFcThr)/AtoV*0.00472/0.8
          deltha = deltha + athromb*c(IC,iNFcThr)/AtoV*0.00472/0.8

        ENDIF

C---- Adjust the ThrFr, deltha, delthu terms
        c(IC,iThrFr) = c(IC,iThrFr) + deltha + delthu
        IF (c(IC,iThrFr).GE.0.9999) c(IC,iThrFr) = 1.0

        c(IC,ixM) = c(IC,ixM) + (rpadh + apadh - upemb)*DT
        IF (c(IC,ixM).GT.xMinf) c(IC,ixM) = xMinf

        c(IC,iSp) = 1.0-(c(IC,ixM)/xMinf)
        IF (c(IC,iSp).LT.0.0) c(IC,iSp) = 0.0
        c(IC,ixMact) = c(IC,ixMact) + apadh*DT
        IF (c(IC,ixMact).GT.c(IC,ixM)) c(IC,ixMact) = c(IC,ixM)


            enddo
            enddo
          endif
        END DO
        INTFLG(21) = ITER
      ENDIF

C---- If at the end of the time step
      ELSE IF(level.EQ.2) THEN

      DO nd=1,doma_no
        if (doma(nd)%mattyp.eq.FLUID) then
          call cset(cs,0,nd,NSD_ALL,INTERNAL)
          do nset=1,cs%no
            do IC=cs%ns(nset),cs%ne(nset)

C Change the pulse flag for when activation time has been reached.
            IF (c(IC,inPlsFl).EQ.1.0) THEN
              c(IC,inPlsFl) = 2.0
              c(IC,ipulse) = 0.0
            ENDIF

C Looking at all cells that are thrombus (i.e. have a high viscosity),
C and it's a new thrombus
            IF (c(IC,iThrFr).GE.0.9999.AND.c(IC,inewThr).EQ.0.0) THEN

C Changing Blood Status of cells that became thrombus
        IF (c(IC,inBldSt).EQ.0.0.OR.c(IC,inBldSt).EQ.4.0) THEN
                c(IC,inBldSt)=3.0
        ELSE IF (c(IC,inBldSt).EQ.1.0.OR.c(IC,inBldSt).EQ.5.0) THEN
                c(IC,inBldSt)=2.0
        ENDIF

c-----FIND NEIGHBOR CELLS AND IDENTIFY AS SUCH
              do jf=1,c2f(IC)%No
                IS = c2f(IC)%List(jf)
                IF(IS.GT.0) THEN
                  do j=1,2
                  IF(lfc(j,IS).NE.IC) THEN
                    NBRCLL = LFC(j,IS)
                    c(NBRCLL,iNFcThr) = c(NBRCLL,iNFcThr) + 1.0
                    IF (c(NBRCLL,inBldSt).EQ.0.0) THEN
                      c(NBRCLL,inBldSt) = 4.0
                    ELSE IF (c(NBRCLL,inBldSt).EQ.1.0) THEN
                      c(NBRCLL,inBldSt) = 5.0
                    ENDIF
                  ENDIF
                  enddo
                ENDIF
              enddo

              c(IC,inewThr) = 1.0
            ENDIF

            enddo
          enddo
        endif
      ENDDO

      ENDIF

      RETURN
      END
```