Laurea Magistrale in Ingegneria Matematica

Progetto per il corso di
Programmazione Avanzata per il Calcolo Scientifico
Professor Luca Formaggia

Riduzione gerarchica di modello per problemi ADR su domini a sezione circolare

Progetto di:
Guzzetti Sofia
Matr. 782674

Lupo Pasini Massimiliano
Matr. 782283

Anno Accademico 2012-2013
Contents

1 Description of the method 3
2 Educated Basis 8
3 Implementation 11
   3.1 BasisAbstract 11
4 ModalSpace 16
5 HiModAssembler 21
6 Numerical results 24
References 33
Introduction

A lot of engineering topics are solved modeling them with partial differential problems. Anyway it usually happens that the complexity of the phenomenon to be analyzed translates likewise into a very intricate mathematical modeling of the problem to be solved too. Thus it is strictly important to develop proper algorithms that enable us to achieve a reliable solution with a managed expense in terms of computational cost and resources that have to be spent to get them. The complexity of the problem might be associated with the shape of the domain where the event is set rather than the considerably high number of coefficients to be considered in the equations that characterize the model. In the worst cases these two aspects raise at the same time. Today a lot of problems concerning scientific matters and engineering can’t be solved due to the shortage of proper devices to be able to sustain the amount of computation which is necessary. It is strictly associated with the evolution of numerical analysis which, in the recent years, had the goal to solve analytical problems that are characterized by a huge number of variables, pushing forward the frontiers created by calculators’ limited ability.

Indeed this work takes place in the field of model reduction algorithms which nowadays are used very widely to solve high dimensional partial differential problems in order to find a good compromise between solution error and computational costs which are required by the implementation of the method itself.

Nowadays there are different methods which try to address the problem induced by curse of dimensionality. One of them is the POD (Proper Orthogonal Decomposition) whose aim is to represent high-dimensional problems in a reduced space so to express the solution of the problem as a linear combination of base functions that are properly selected by the algorithm.

Another way of proceeding to reduce the complexity of the problems consists in adopting different kind of discretization along different dimensions of the domain. That’s the case of Hierarchical Model Reduction (Hi-Mod). As it is shown in [EPV08], [PEV], [PEV10] and [PZ13] the spirit of this method is to detect if there is one direction which is more important than others to describe the phenomenon. If so, then along this way a Finite Element discretization is applied, instead along all the others the discretization is done by pseudo-spectral methods. The result is a linear system whose stiffness matrix is much easier to treat from a numerical point of view rather than using Finite Elements along all the dimensions of the domain.

The goal of our work is to use Hierarchical Model Reduction to solve 3D partial differential problems that are defined on a domain whose geometry is not a parallelepiped, but for instance a cylinder with a circular base. Furthermore we will consider different boundary conditions to be coupled to the partial differential equation to fully determine the problem to be solved.
1 Description of the method

Let us consider for example an advection-diffusion-reaction problem set on a cylindric domain:
\[
\begin{align*}
-\mu \Delta u + \mathbf{b} \cdot \nabla u + \sigma u &= f & \text{in } \Omega \\
u &= u_{in} & \text{on } \Gamma_{in} \\
\frac{\partial u}{\partial x} &= 0 & \text{on } \Gamma_{out} \\
u &= 0 & \text{on } \Gamma_{side}
\end{align*}
\]

(1)

\[\Omega = \{(x, \rho, \theta) : 0 < x < a, 0 < \rho < R, 0 < \rho < 2\pi\},\]
\[\Gamma_{in} \cap \Gamma_{out} \cap \Gamma_{side} = \emptyset, \Gamma_{in} \cup \Gamma_{out} \cup \Gamma_{side} = \partial \Omega.\]

The coefficients of (1) are set so that well-posedness of its variational form is guaranteed by Lax-Milgram Theorem.

To develop the problem in the weak form it is necessary to introduce a proper functional setting. For this we appeal to the Sobolev Space:
\[H^1_0(\Omega) = \{v \in H^1(\Omega) : v |_{\Gamma} = 0\}\]

where \(\partial \Omega\) is the boundary of the set \(\Omega\) and \(H^1(\Omega)\) is defined as:
\[H^1(\Omega) = \{v \in L^2(\Omega) : \nabla v \in [L^2(\Omega)]^d\}.\]

\(\nabla(\cdot)\) denotes the gradient operator and \(d\) is the number of dimensions of \(\mathbb{R}^d\) so that \(\Omega \subset \mathbb{R}^d\). In our case \(d = 3\).

Thanks to the properties of Banach space \(L^2(\Omega)\) which is also an Hilbert space, it descends that \(H^1_0(\Omega)\) is an Hilbert space too, as all its subsets. Thus it is possible to introduce the dual to \(H^1(\Omega)\) whose symbol is \([H^1(\Omega)]^*\) and for sake of simplicity we will refer to it with the abbreviated representation \(H^{-1}(\Omega)\).

We use the following convention:
• \( \Gamma_D \) = portion of frontier \( \partial \Omega \) with Dirichlet boundary conditions (thus \( \Gamma_{in} \subset \Gamma_D \))
• \( \Gamma_N \) = portion of frontier \( \partial \Omega \) with Neumann boundary conditions
• \( \Gamma_R \) = portion of frontier \( \partial \Omega \) with Robin boundary conditions

The weak form of the problem descends as follows:

\[
\int_\Omega (-\mu \Delta u + b \cdot \nabla u + \sigma u - f) v d\omega = 0 \quad \forall v \in H^1_{0, \Gamma_D}(\Omega). \tag{2}
\]

Integrating by parts we obtain:

\[
\int_\Omega \left[ \mu \nabla u \cdot \nabla v + (b \cdot \nabla u)v + \sigma uv \right] d\omega - \int_{\partial \Omega \setminus \Gamma_D} \mu \frac{\partial u}{\partial n} v - \int_\Omega f v d\omega = 0 \quad \forall v \in H^1_{0, \Gamma_D}(\Omega). \tag{3}
\]

Introducing function \( g \in H^1(\Omega) \) defined as follows:

\[
geq_{|\Gamma_D} = u_{|\Gamma_D},
\]

we can modify the previous weak form so that: \( u = \tilde{u} + g \)

\[
\int_\Omega \left[ \mu \nabla \tilde{u} \cdot \nabla v + (b \cdot \nabla \tilde{u})v + \sigma \tilde{u}v \right] d\omega - \int_{\partial \Omega \setminus \Gamma_D} \mu \frac{\partial \tilde{u}}{\partial n} v = \int_\Omega f v d\omega \quad \forall v \in H^1_{0, \Gamma_D}(\Omega).
\]

The last form of the variational problem enables us to use Lax-Milgram theorem. In fact introducing a bilinear form:

\[
a(\cdot, \cdot) : H^1_{0, \Gamma_D} \times H^1_{0, \Gamma_D} \to \mathbb{R}
\]

and a functional \( F \in [H^1_{0, \Gamma_D}]^* : F : H^1_{0, \Gamma_D} \to \mathbb{R} \)

we can define them as follows:

\[
a(\tilde{u}, v) = \int_\Omega \left[ \mu \nabla \tilde{u} \cdot \nabla v + (b \cdot \nabla \tilde{u})v + \sigma \tilde{u}v \right] d\omega - \int_{\partial \Omega \setminus \Gamma_D} \mu \frac{\partial \tilde{u}}{\partial n} v
\]

\[
F(v) = \int_\Omega f v d\omega
\]

Therefore the variational form of the problem turns into:

\[
\text{find } \tilde{u} \in H^1_{0, \Gamma_D} \text{ so that } a(\tilde{u}, v) = F(v) \quad \forall v \in H^1_{0, \Gamma_D}. \tag{5}
\]

Bilinear form \( a(\cdot, \cdot) \) and and functional \( F(\cdot) \) satisfies all the hypothesis that are required by Lax-Milgram theorem, so one and only one solution to (5) is guaranteed.

For our purposes domain \( \Omega \) will be an open right circular cylinder with constants transversal sections which has the property to be defined as a tensor product:

\[
\Omega = [a, b] \times \Sigma_R,
\]
is the range for variable $x$ along the horizontal direction and $\Sigma_R$ is a generic circular section.
For sake of simplicity we will refer to the radius of circular section with $R$ and to the length of the horizontal dimension of the cylinder with $L$.

If $R \ll L$, then phenomenon is characterized by a privileged direction (mainstream) which is detected by variable $x$, instead what happens in the transversal section $\Sigma_R$ can be synthetized in proper coefficients in view of a model reduction.

In fact the particular structure of the domain gives us the chance to solve the partial differential problem using a simplified method that cut down the complexity of the algorithm to be adopted.

In order to do this we introduce an orthogonal base $\{\varphi_i\}_{i \in \mathbb{N}}$ for $H^1(\Sigma)$ that needs to be orthonormal respect to $L^2(\Sigma)$ (weighted) scalar product. Thus solution to 5 can be expressed in the following way:

$$u(x, \rho, \theta) = \sum_{i=0}^{\infty} u_i(x) \varphi_i(\rho, \theta).$$

We can notice that the tensorial structure of the domain translates into a tensorial structure of the solution itself. This way of proceeding is at the basis of our discretization method. In detail we will use different discretizing approaches along the mainstream and in the transversal section.

As concerns the vertical fiber the approximation consists in truncating the number of Fourier coefficients that are used to build the value of the solution along the transversal section:

$$u(x, \rho, \theta) \approx \sum_{i=0}^{m} u_i(x) \varphi_i(\rho, \theta), \quad m \in \mathbb{N}.$$ 

Setting 5 in this functional framework and integrating along the transversal section of the domain we introduce a set of proper coefficients for the unidimensional equation along the mainstream. To do this we multiplied both the member of the equation with $\rho$ in order to neglect the singularity of the elliptic operator in cylindric reference system for $\rho = 0$.

$$-\mu \int_0^L \int_0^{2\pi} \int_0^R \left( \frac{\partial^2 u}{\partial \rho^2} + \frac{1}{\rho} \frac{\partial u}{\partial \rho} + \frac{1}{\rho^2} \frac{\partial^2 u}{\partial \theta^2} + \frac{\partial^2 u}{\partial x^2} \right) v \rho^2 d\rho d\theta dx$$

$$+ \int_0^L \int_0^{2\pi} \int_0^R \left( \beta_x \frac{\partial u}{\partial x} + \beta_\rho \frac{\partial u}{\partial \rho} + \beta_\theta \frac{1}{\rho} \frac{\partial u}{\partial \theta} \right) v \rho^2 d\rho d\theta dx$$

$$+ \int_0^L \int_0^{2\pi} \int_0^R \sigma uv \rho^2 d\rho d\theta dx = \int_0^L \int_0^{2\pi} \int_0^R f v \rho^2 d\rho d\theta dx$$

Applying variable separation we obtain:

$$u(x, \rho, \theta) = \sum_{k=1}^{M} u_k(x) \varphi_k(\rho, \theta)$$

$$\varphi_k(\rho, \theta) = \frac{1}{R\sqrt{2\pi}} \eta_k(\rho) \xi(\theta)$$

and defining $z_j$ as the $j$-th Finite Element basis function the integrals above can be written as the following:
\( \left( I \right) \)

\[- \mu \int_0^{2\pi} \int_0^R \frac{\partial^2 u}{\partial \rho^2} \varphi_j \rho^2 d\rho d\theta = \frac{\mu}{R} \left[ 2 \int_0^1 \tilde{\eta}_k \hat{\eta}_j \rho d\rho \int_0^1 \hat{\xi}_k \hat{\xi}_j d\theta \right. \]

\[+ \int_0^1 \tilde{\eta}_k \hat{\eta}_j \rho^2 d\rho \int_0^1 \hat{\xi}_k \hat{\xi}_j d\theta \]

\[+ \frac{\chi}{R} \tilde{\eta}(1) \hat{\eta}(1) \int_0^1 \xi_k \xi_j d\theta \]

\( \left( II \right) \)

\[- \mu \int_0^{2\pi} \int_0^R \frac{\partial \varphi_j}{\partial \rho} \rho d\rho d\theta = - \mu \int_0^1 \tilde{\eta}_k \hat{\eta}_j \rho d\rho \int_0^1 \hat{\xi}_k \hat{\xi}_j d\theta \]

\( \left( III \right) \)

\[- \mu \int_0^{2\pi} \int_0^R \frac{\partial^2 \varphi_j}{\partial \theta^2} \rho d\rho d\theta = - \mu \int_0^1 \tilde{\eta}_k \hat{\eta}_j \rho d\rho \int_0^1 \hat{\xi}_k \hat{\xi}_j d\theta \]

\[+ \frac{\mu}{R(2\pi)^2} \left[ \tilde{\xi}_k(1) \hat{\xi}_j(1) - \tilde{\xi}_k(0) \hat{\xi}_j(0) \right] \int_0^1 \tilde{\eta}_k \hat{\eta}_j d\rho \]

\( \left( IV \right) \)

\[- \mu \int_0^L \int_0^{2\pi} \int_0^R \frac{\partial^2 \varphi_k}{\partial x^2} \varphi_j \rho^2 d\rho d\theta dx = \frac{R \mu}{L} \left[ \int_0^1 \dot{u}_k(0) \dot{z}_j(0) - \int_0^1 \dot{u}_k(1) \dot{z}_j(1) + \int_0^1 \dot{u}_k(\dot{z}_j)^{\prime} dx \right] \int_0^1 \hat{\xi}_k \hat{\xi}_j d\theta \]

\( \left( V \right) \)

\[\int_0^{2\pi} \int_0^R \beta \frac{\partial \varphi_k}{\partial \rho} \varphi_j \rho^2 d\rho d\theta = \beta \int_0^1 \tilde{\eta}_k \hat{\eta}_j \rho^2 d\rho \int_0^1 \hat{\xi}_k \hat{\xi}_j d\theta \]

\( \left( VI \right) \)

\[\int_0^{2\pi} \int_0^R \beta \frac{\partial \varphi_k}{\partial \theta} \varphi_j \rho^2 d\rho d\theta = \beta \int_0^1 \tilde{\eta}_k \hat{\eta}_j \rho d\rho \int_0^1 \hat{\xi}_k \hat{\xi}_j d\theta \]

\( \left( VII \right) \)

\[\int_0^L \int_0^{2\pi} \int_0^R \beta \frac{\partial u}{\partial x} \varphi_j \rho^2 d\rho d\theta dx = \beta \int_0^1 \tilde{\eta}_k \hat{\eta}_j \rho^2 d\rho \int_0^1 \hat{\xi}_k \hat{\xi}_j d\theta \]

\( \left( VIII \right) \)

\[\int_0^{2\pi} \int_0^R \sigma \varphi_k \varphi_j \rho^2 d\rho d\theta = \sigma R \int_0^1 \tilde{\eta}_k \hat{\eta}_j \rho^2 d\rho \int_0^1 \hat{\xi}_k \hat{\xi}_j d\theta \]
This enables us to read a set of one-dimensional ADR problems, one for each couple \((j,k)\) of modal functions.

In the cases that we will consider, the coefficients of the equation are constant and this makes possible to additionally apply separation of variables upon \((\rho, \theta)\).
2 Educated Basis

The most common choice for the modal basis is the Fourier Basis, which allows us to write any function belonging to the Sobolev space $L^2((0,1))$ as a series of sines and cosines, weighted on proper coefficients. As the sine function vanishes in $2k\pi$, $\forall k \in \mathbb{N}$, we are able to pointwise impose boundary conditions just in case of Dirichlet.

We would like to enforce the hierarchical model by working out an educated modal basis which keeps memory of the specific boundary conditions of the problem we want to solve. In this way we can set essentially Neumann and Robin boundary conditions, as well as Dirichlet ones. We can do that by solving an eigenvalue problem associated to Laplace operator on the circular section, setting on the circumference the same conditions of the global problem. As the domain is characterized by a specific geometry, it is reasonable to express the basis functions (and consequently the Laplace operator) in terms of polar coordinates. If we consider a circle $\Sigma_R$ with radius $R$ as a domain, the problem is then expressed as

\[
\begin{align*}
\frac{\partial^2 \varphi}{\partial \rho^2} + \frac{1}{\rho} \frac{\partial \varphi}{\partial \rho} + \frac{1}{\rho^2} \frac{\partial^2 \varphi}{\partial \theta^2} &= \lambda \varphi \quad \text{on } \Sigma_R \\
+BC &\quad \text{on } \partial \Sigma_R
\end{align*}
\]

For sake of simplicity, we write the equation on a unit circle as a reference domain through the map $\hat{\rho} = \frac{\rho}{R}$: \footnote{\textcolor{red}{We will make use of the symbol }$\hat{\rho}$\textcolor{red}{ for any quantity defined on the reference domain.}}

\[
\begin{align*}
\frac{\partial^2 \hat{\varphi}}{\partial \hat{\rho}^2} + \frac{1}{\hat{\rho}} \frac{\partial \hat{\varphi}}{\partial \hat{\rho}} + \frac{1}{\hat{\rho}^2} \frac{\partial^2 \hat{\varphi}}{\partial \theta^2} &= R^2 \lambda \hat{\varphi} \quad \text{on } \Sigma_1 \\
+BC &\quad \text{on } \partial \Sigma_1
\end{align*}
\]

We resort to separation of variables and express $\hat{\varphi}$ as $\hat{\varphi}(\hat{\rho}, \theta) = \hat{\eta}(\hat{\rho}) \xi(\theta)$, $\hat{\rho} \in [0, 1], \theta \in [0, 2\pi]$, which leads us to the following couple of mutually independent eigenvalues problems [Str86]:

\[
\begin{align*}
-\xi''(\theta) &= n^2 \xi(\theta) \quad \text{in } [0, 2\pi] \\
\xi(0) &= \xi(2\pi) \\
+\hat{\eta}''(\hat{\rho}) + \hat{\rho} \hat{\eta}'(\hat{\rho}) + (\hat{\lambda} \hat{\rho} - n^2) \hat{\eta}(\hat{\rho}) &= 0 \quad \text{in } [0, 1]
\end{align*}
\]

where $n$ is an arbitrary real constant and $\hat{\lambda} = R^2 \lambda$ is the eigenvalue on the reference domain.

We must note that, compared with the akin eigenvalues problems of the case of the cylinder with rectangular section [AB13a], here problem the former problem does not keep trace of the eigenvalue, which instead characterizes the latter equation, known as Bessel’s Equation. As a consequence, it is not possible to obtain the two-dimensional eigenvalues from the one-dimensional ones: we have to compute straightforward the formers, which turn out to be the roots of the function associated to the condition on contour.

The periodic request for $\xi$ forces $n$ to be integral and the real solution in $\theta$ is

\[
\xi(\theta) = c_1 \sin(n\theta) + c_2 \cos(n\theta).
\]

If we choose $c_1 = c_2 = c$ such that (9) has unit norm in $L^2((0,1))$, we get $c = \frac{1}{\sqrt{2\pi}}$. 

\[
\begin{equation}
\xi(\theta) = \frac{1}{\sqrt{2\pi}} \left( \sin(n\theta) + \cos(n\theta) \right).
\end{equation}
\]
Provided that \( n \in \mathbb{N} \), the general solution of Bessel’s Equation is a linear combination of Bessel’s functions of first and second type \((J_n, Y_n)\) respectively in the form

\[
\hat{\eta}(\hat{\rho}) = c_1 J_n(\sqrt{\hat{\lambda}} \hat{\rho}) + c_2 Y_n(\sqrt{\hat{\lambda}} \hat{\rho}),
\]

where

\[
J_n(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m! \Gamma(m+n+1)} \left( \frac{x}{2} \right)^{2m+n}
\]

\[
Y_n(x) = \lim_{\nu \to n} \frac{J_\nu(x) \cos(\nu \pi) - J_{-\nu}(x)}{\sin(\nu \pi)}
\]

and \( \Gamma(\cdot) \) represents the Gamma function.

As the function of second type has a singularity in the origin, we ask for \( c_2 \) to vanish and for \( c_1 \) to normalize \( J_n \), such that \( c_1 = \frac{1}{\| J_n \|^2_{L^2((0,1))}} \).

We can finally write the \( k \)-th function of the modal basis as

\[
\hat{\varphi}_m(\hat{\rho}, \theta) = \frac{1}{\sqrt{2 \pi \| J_n \|^2_{L^2((0,1))}}} \left( \sin(n \theta) + \cos(n \theta) \right) J_n(\sqrt{\hat{\lambda}_m} \hat{\rho}),
\]

and the orthogonality of the basis with respect to the \( \rho \)-weighted scalar product is guaranteed from theory. As the modal basis is required to be normalized on the physical domain too, a rescale is needed so that we get

\[
\varphi_m(\rho, \theta) = \frac{1}{\sqrt{2 \pi R \| J_n \|^2_{L^2((0,1))}}} \left( \sin(n \theta) + \cos(n \theta) \right) J_n(\sqrt{\lambda_m} \rho).
\]

In this way we obtain

\[
\| \varphi_m(\rho, \theta) \|^2_{L^2((0,1))} = \int_0^{2\pi} \int_0^R \eta^2(\rho) \xi^2(\theta) \rho d\rho d\theta
\]

\[
= \frac{1}{2 \pi R^2 \| J_n \|^2_{L^2((0,1))}} \int_0^{2\pi} \int_0^1 \left( \sin(n \theta) + \cos(n \theta) \right)^2 J_n^2(\sqrt{\lambda} \hat{\rho}) R \hat{\rho} R d\hat{\rho} d\theta
\]

\[
= \frac{1}{2 \pi} \int_0^{2\pi} \left( \sin(n \theta) + \cos(n \theta) \right)^2 d\theta \frac{1}{\| J_n \|^2_{L^2((0,1))}} \int_0^1 J_n^2(\sqrt{\lambda} \hat{\rho}) \hat{\rho} d\hat{\rho}
\]

\[
= 1.
\]

In order to determine the values of the frequencies \( \{ \lambda_k \} \), \( k \in \mathbb{N} \), we introduce the boundary conditions upon the Bessel Function.

**Dirichlet BC**

We ask the eigenfunction to vanish in \( \rho = R \), i.e. \( \hat{\rho} = 1 \):

\[
J(\sqrt{\lambda} R) = J(\sqrt{\lambda}) = 0.
\]

\footnote{It is known from literature the relation \( \| J_n \|^2_{L^2((0,1))} = \int_0^1 J_n(\sqrt{\lambda} \hat{\rho}) \hat{\rho} d\hat{\rho} = \frac{1}{2} \{ J_n(\sqrt{\lambda}) + (1 - \frac{n^2}{\lambda}) J_n(\sqrt{\lambda})^2 \} \).
}
The roots of the Bessel function turn out to be the eigenvalues of the problem in the reference domain.

In this case the norm of the Bessel function is given by the following simple relation:

$$\|J_n\|_{L^2((0,1))}^2 = \int_0^1 J_n^2(\sqrt{\hat{\lambda}} \hat{\rho}) \hat{\rho} d\hat{\rho} = \frac{1}{2} J_n^2(\sqrt{\hat{\lambda}}).$$

(14)

**Neumann BC**

The first derivative vanishes on the circumference:

$$\sqrt{\hat{\lambda}} J'(\sqrt{\hat{\lambda}} R) = \sqrt{\hat{\lambda}} R J(\sqrt{\hat{\lambda}}) = 0$$

(15)

and the norm is computed as

$$\|J_n\|_{L^2((0,1))}^2 = \frac{1}{2\sqrt{\hat{\lambda}}} (\hat{\lambda} - n^2) J^2(\sqrt{\hat{\lambda}}).$$

(16)

**Robin BC**

The radial basis function is required to satisfy the following condition:

$$J'(\sqrt{\hat{\lambda}} R) + \frac{\chi}{\mu} J(\sqrt{\hat{\lambda}} R) = 0$$

(17)

with $\mu$ and $\chi$ real positive parameters. The norm assumes the form:

$$\|J_n\|_{L^2((0,1))}^2 = \frac{1}{2\sqrt{\hat{\lambda}}} (\hat{\lambda} + \left(\frac{\chi}{\mu}\right)^2 R^2 - n^2) J^2(\sqrt{\hat{\lambda}}).$$

(18)
The code has been developed in LifeV and is located in branch 20131017_HiMod2. We made use of `make` for the compilation and `git` for managing the scripts. Vectors and matrices have been handled with specific tools in LifeV, which are based on Trilinos, while the assembling of the sub-problems has been achieved through Expression Templates (ETA). The code is serial, in fact a parallel execution would need proper structures able to deal with the pattern of the system matrix.

The evaluation of the Bessel functions is already implemented in the `Navier_Stokes` package, while the algorithm to find the roots of the function itself and its derivatives has been developed on the base of a `fortran` code by Jin and Zhang [ZJ96].

The domain of the computation is a cylinder of length $L_x$ with circular section of radius $R$. We want to solve a steady ADR problem with generic Dirichlet inflow conditions and homogeneous Neumann conditions in outflow. We assume homogeneous conditions on the lateral wall and that the coefficients of the problem are constant. Moreover we suppose that it is possible to treat the two-dimensional problem on the section through the splitting of the radial and angular coordinates.

Our code is an extension of the branch named 20130507_HiMod, which solves the same problem on a cylinder with rectangular section. The geometric nature of the domain induces us to use a cylindric reference system, which implies a radical change in the structure of the problem. As we have seen before, in fact, the information about the two-dimensional problem is contained in only one 1D problem as a whole, while the other one determines the periodicity of the solution.

We tried to keep the pre-existent code untouched as much as possible and to keep a similar framework in the implementation of the new classes. For this purpose we made large use of template classes and inheritance.

The code is organized in three different levels which refer to the progressive dimension of the problem we want to solve. `BasisAbstract` deals with the one-dimensional problem associated to the search of the main frequencies of the problem. The former `Basis1DAbstract` and the latest `Basis2DAbstract` inherit from it. They both handle boundary conditions.

`ModalSpaceRectangular` and `ModalSpaceCircular`, whose father-class is `ModalSpaceAbstract`, assemble the two one-dimensional problems into the one defined on the section of the cylinder. They build the real modal basis by choosing the proper frequencies among those computed by a `Basis1D` or a `Basis2D`, compute the Fourier coefficients and the numeric integrals for the weak formulation of the problem.

The final link which leads to the three-dimensional problem is performed by `HiModAssembler`, which is a template class such that the shape of the section is given as a parameter. It builds the matrix and the known term of the equation.

### 3.1 BasisAbstract

`BasisAbstract` represents a generic interface for one-dimensional problems. It branches into two derived classes, one for each type of section. As these present many differences due to the geometry of the domain, the father-class provides just few methods to set the coefficients associated to the boundary conditions. In turn `Basis1D` and `Basis2D` provide an interface for the real implementation of conditions on contour. In particular, `Basis2D` supplies some methods for the computation of the modal basis and forces the computation of the eigenvalues to the derived class.

```cpp
class BasisAbstract
```
```cpp
public:

    BasisAbstract() {}
    virtual ~BasisAbstract() {};

    virtual void setMu ( Real const& /*mu*/ ) {}
    virtual void setChi ( Real const& /*chi*/ ) {};
    virtual Real chi () const = 0;

protected:

};

class Basis2DAbstract : public BasisAbstract
{
    public:

    Basis2DAbstract() : M_normTheta(1.) , M_Theta(M_PI*2) , M_icurrent(0) {};
    virtual ~Basis2DAbstract() {};

    void setRho( const Real& Rho )
    {
        M_Rho = Rho;
    };

    void setTheta( const Real& Theta )
    {
        M_Theta = Theta;
    };

    virtual void setMu ( Real const& /*mu*/ ) {};
    virtual void setChi ( Real const& /*chi*/ ) {};

    void setNumberModes ( const Uint& m )
    {
        M_mtot = m;
    }

    virtual Real chi () const = 0;
```
virtual EigenContainer eigenValues () const
{
    return M_eigenvalues;
};

virtual void computeEigenvalues () = 0;

void
evaluateBasis ( MBMatrix_type& phirho,
                MBMatrix_type& dphirho,
                MBMatrix_type& phitheta,
                MBMatrix_type& dphitheta,
                const EigenContainer& eigenvalues,
                const QuadratureRule* quadrulero,
                const QuadratureRule* quadruletheta ) const;

Real
evalSinglePoint ( const Real& eigen, const UInt index, const
MBVector_type& p ) const;

protected:

Real M_Rho;
Real M_Theta;
UInt M_icurrent;
UInt M_mtot;
EigenContainer M_eigenvalues;
Real M_normTheta;
MBVector_type M_normRIJ;

If we assume that on the lateral wall of the cylinder just one type of request may be enhanced, only
the three plain types of BCs are possible, without any combination. Accordingly, the derived classes
are named EducatedBasisD, N, R, while the first two objects compute the roots of the boundary
function straightforward through a specific algorithm, the last one employs a functor. From the
plot of the contour function and the akin Bessel function, it is easy to see that the roots of the
former lies always between the roots of the latter. In this way it is possible to apply the Bisection
method in a loop, in order to get the desired number of eigenvalues.

Although we could not split the two-dimensional eigenvalues into two one-dimensional ones,
like in the case of rectangular section, the parameter \(n\) which appears in the eigenvalues problem
introduces a further degree of freedom in the choice of the order of the modal function.

We know from the Spectral Theorem that the main contribution comes from the eigenfunctions
associated to the least eigenvalues. We have thus implemented an algorithm which selects the
eigenvalues in increasing order among the roots of the Bessel functions of different orders. Note
that in the rectangular case this operation is worked out by ModalBasis, since the two rare one-
dimensional eigenvalues are independent one from another and need to be combined at a higher
level. In this case, instead, the eigenvalues computed by the EducatedBasis concern directly the
problem on the section, so it is reasonable to compute the entire basis here.

The eigenvalues are computed by computeEigenvalues in the following way: the least one is the first root associated to the Bessel function of order zero. According to the nature of these functions, at each stage of the search the next bigger eigenvalue may be the next root of the function of the same order, or the next root of the function of higher order. At each step we insert these two candidates in a ordered set in such a way that the first element of the set is the least of all the candidates inserted so far. The abstract class then provides the methods evaluateBasis and evalSinglePoint to evaluate the modal basis on quadrature points and in any point for any boundary condition, since the information is condensed into the eigenvalues.
Figure 2: EducatedBasis classes hierarchy
4 ModalSpace

ModalSpaceAbstract provides the methods to compute the lumped coefficients which take into account all the information coming from the section when solving the one-dimensional problem. It computes also Fourier coefficients, useful in the assembling of the problem and in the postprocessing of the solution.

```cpp
class ModalSpaceAbstract {

public:
    ModalSpaceAbstract(UINT m): M_mtot(m) {}

    virtual ~ModalSpaceAbstract() {};

    virtual void evaluateBasis() = 0;

    virtual std::vector<Real> fourierCoefficients (const function_Type& g) const = 0;

    virtual Real fourierCoeffPointWise (const Real& x, const function_Type& f, const UINT& k) const = 0;

    virtual Real compute_R11 (const UINT& j, const UINT& k,
                              const function_Type& mu, const Real& x) const = 0;

    virtual Real compute_R10 (const UINT& j, const UINT& k,
                              const function_Type& beta, const Real& x) const = 0;

    virtual void showMe() const = 0;

    UINT mtot() const

protected:

    virtual void eigensProvider() = 0;

   UINT M_mtot;
};
```

ModalSpaceRectangular and ModalSpaceCircular derive from the abstract class. Differently from the case of the rectangle, in case of the circle the two-dimensional eigenvalues are computed directly.
by the proper \textit{EducatedBasis}, so the method \texttt{eigensProvider} only recalls the eigenvalues from the basis it points to and assign them to those of the modal basis. In the same way, \texttt{evaluateBasis} extracts the values of the modal functions previously computed by the proper educated basis, instantiated through the method \texttt{addSliceBC}.

The other members of the class are mainly functions which carry out the numeric computation of the integrals used to build the lumped parameters in polar coordinates. Note that the equation has a singularity in $\rho = 0$, so we multiply both the members for $\rho$. In this way when we integrate the weight simplifies the singularity. The integrals are computed as described here:

- $\text{compute1\_PhiPhi} \int_0^{2\pi} \int_0^R \varphi_i \varphi_k \rho \, dp \, d\theta$
- $\text{compute2\_PhiPhi} \int_0^{2\pi} \int_0^R \varphi_i \varphi_k \rho^2 \, dp \, d\theta$
- $\text{compute1\_DrhoPhiPhi} \int_0^{2\pi} \int_0^R \frac{\partial \varphi_i}{\partial \rho} \frac{\partial \varphi_k}{\partial \rho} \rho \, dp \, d\theta$
- $\text{compute2\_DrhoPhiPhi} \int_0^{2\pi} \int_0^R \frac{\partial \varphi_i}{\partial \rho} \frac{\partial \varphi_k}{\partial \rho} \rho^2 \, dp \, d\theta$
- $\text{compute1\_DthetaPhiPhi} \int_0^{2\pi} \int_0^R \frac{\partial \varphi_i}{\partial \theta} \frac{\partial \varphi_k}{\partial \theta} \rho \, dp \, d\theta$
- $\text{compute0\_DthetaPhiDthetaPhi} \int_0^{2\pi} \int_0^R \frac{\partial \varphi_i}{\partial \theta} \frac{\partial \varphi_k}{\partial \theta} \, dp \, d\theta$
- $\text{compute\_Phi} \int_0^{2\pi} \int_0^R \varphi_k \rho^2 \, dp \, d\theta$
- $\text{compute\_rho\_PhiPhi} \int_0^R \varphi_i \varphi_k \rho \, dp$
- $\text{compute\_theta\_PhiPhi} \int_0^{2\pi} \varphi_i \varphi_k \, d\theta$

The lumped coefficients of the weak formulation are computed in functions \texttt{compute\_R11}, \texttt{compute\_R00} and \texttt{compute\_R10}. 

17
class ModalSpaceCircular: public ModalSpaceAbstract
{

public:

ModalSpaceCircular
( const Real& Rho, const Real& Theta, const Uint& M, const QuadratureRule* quadrho = &quadRuleLobSeg64pt, const QuadratureRule* quadtheta = &quadRuleLobSeg64pt ) :
ModalSpaceAbstract( M ), M_quadruleRho ( quadrho ), M_quadruleTheta ( quadtheta ), M_Rho ( Rho ), M_Theta ( Theta ) {};

virtual ~ModalSpaceCircular();

void addSliceBC ( const std::string& BC, const Real& mu = 1., const Real& Chi = 1. );

void evaluateBasis ();

virtual std::vector<Real> fourierCoefficients (const function_Type& g) const;

virtual Real compute1_PhiPhi (const Uint& j, const Uint& k) const;
virtual Real compute2_PhiPhi (const Uint& j, const Uint& k) const;
virtual Real compute1_DrhoPhiPhi( const Uint& j, const Uint& k ) const;
virtual Real compute2_DrhoPhiPhi( const Uint& j, const Uint& k ) const;
virtual Real compute1_DrhoPhiDrhoPhi (const Uint& j, const Uint& k) const;

virtual Real compute2_DrhoPhiDrhoPhi (const Uint& j, const Uint& k) const;
virtual Real compute1_DthetaPhiPhi (const Uint& j, const Uint& k) const;
virtual Real compute0_DthetaPhiDthetaPhi (const Uint& j, const Uint& k) const;

virtual Real compute1_Phi (const Uint& k) const;
virtual Real compute_rho_PhiPhi (const Uint& j, const Uint& k) const;
virtual Real compute_theta_PhiPhi (const Uint& j, const Uint& k) const;
virtual Real compute_R11 (const Uint& j, const Uint& k, const function_Type& mu, const Real& x) const;
virtual Real compute_R10 (const Uint& j, const Uint& k, const function_Type& beta, const Real& x) const;
virtual Real compute_R00 (const Uint& j, const Uint& k, const function_Type& mu, const function_Type& beta, const Real& x) const;
virtual void showMe() const;

Real Rho() const;
Real Theta() const;

virtual Real phirho (const UInt& j, const UInt& n) const;
virtual Real dphirho (const UInt& j, const UInt& n) const;
virtual Real phitheta (const UInt& j, const UInt& n) const;
virtual Real dphitheta (constUInt& j, const UInt& n) const;

EigenMap2D eigenvalues (UInt j) const;
const QuadratureRule& qrRho() const;
const QuadratureRule& qrTheta() const;
const basis2d_ptrType gbRhoTheta() const;

protected:

virtual void eigensProvider();
Basis2DAbstract::EigenContainer M_eigenvalues;
const QuadratureRule* M_quadruleRho;
const QuadratureRule* M_quadruleTheta;
basis2d_ptrType M_genbasisRhoTheta;
Real M_Rho;
Real M_Theta;
MBMatrix_type M_phirho;
MBMatrix_type M_phitheta;
MBMatrix_type M_dphirho;
MBMatrix_type M_dphitheta;

};
Figure 3: Modal Basis classes hierarchy.
5 HiModAssembler

HiModAssembler is a template class that is just declared but not defined. It is used to set the way to differentiate its specializations for the geometry of the transversal section: rectangular or cylindric. The parameters of this class template are four: the type of the Finite Elements mesh which is used on the mainstream, the type of the matrix which is used to assemble the 3D problem, the type of vector which is used for the discretization of the force term and the last one is a string that is used to set the geometric form of the section. Thus the string can be initialized with two values: Rectangular or Circular.

```cpp
class sectionGeometry { Rectangular, Circular };
```

Setting the string equal to Rectangular we obtain the class that has been implemented by Aletti and Bortolossi for their project. Instead setting it to Circular we obtain the specialization that has been implemented by us.

```cpp
template< typename mesh_type, typename matrix_type, typename vector_type, int N>
class HiModAssembler;
```

We now get down to the Circular specialization. Its main methods are:

- **addADRProblem**: it builds the matrix of the problem,
- **addrhs**: it builds the vector which is associated to the known term,
- **addDirichletBC_In**: it is used to impose inlet Dirichlet, boundary conditions by penalization of both the matrix and force vector of the system,
- **evaluateBase3DGrid**: it gets an $m \cdot N_h$ vector that contains Fourier coefficients at every Finite Elements node and it builds a vector that contains evaluation of the function in all the points that discretize the domain.

All the methods that we have listed above are characterized by different specializations, due to the nature of the input parameters. As a matter of fact they get equations parameter that can be used as real values, rather than vectors or functions.

```cpp
template< typename mesh_type, typename matrix_type, typename vector_type >
class HiModAssembler< mesh_type, matrix_type, vector_type, 1 >
{
    public:

    HiModAssembler( const fespace_ptrType& fespace, const modalbasis_ptrType& modalbasis, commPtr_Type& Comm );

    void addADRProblem( const matrix_ptrType& systemMatrix, const Real& mu, const TreDvector_type& beta, const Real& sigma );

    void addADRProblem( const matrix_ptrType& systemMatrix,
```

21
const function_Type& mu,
const function_Type& beta ,
const function_Type& sigma );

void interpolate( const function_Type& f,  
const vector_ptrType& f_interpolated );

void addrhs( const vector_ptrType& rhs ,  
const Real& f );

void addrhs( const vector_ptrType& rhs ,  
const vector_ptrType& f_interpolated );

void addrhs_HiPrec( const vector_ptrType& rhs ,  
const function_Type& f );

void addDirichletBC_In( const matrix_ptrType& systemMatrix ,  
const vector_ptrType& rhs ,
const function_Type& g );

void interp_Coefficients( const vector_ptrType& r11 ,  
const vector_ptrType& r10 ,
const vector_ptrType& r00 ,
const UInt& j ,
const UInt& k ,
const function_Type& mu ,
const function_Type& beta ,
const function_Type& sigma );

modalbasis_ptrType modalspace() const ;

fespace_ptrType fespace() const ;

vector_type evaluateBase3DGrid( const vector_type& fun );

vector_type evaluateBase3DGrid( const function_Type& fun );

Real normL2( const vector_type& fun );

Real evaluateHiModFunc( const vector_ptrType& rhs , const Real& x , const Real& y ,
const Real& z ,
const GetPot& dfile , std::string prefix , std::string content = "solution");

void exporterStructuredVTK( const UInt& nx , const UInt& ny , const UInt& nz ,
const vector_ptrType& fun , const GetPot& dfile , std::string prefix , std::string content = "solution");
void exporterFunctionVTK (const UInt& nx, const UInt& ny, const UInt& nz, 
const function_Type& f, const GetPot& dfile, std::string prefix, std::string
content = "function");

private:

    modalbasis_ptrType     M_modalbasis;
    etfespace_ptrType      M_etfespace;
    fespace_ptrType        M_fespace;
};
6 Numerical results

In this section we show you some numerical results we achieved and to visualize them we have used ParaView software. At first we put the attention to a particular case for which solution is known a priori, so that we can compare the trend of numerical solution with the analytical one, varying the numerical setting of the mesh and of the model reduction too. Then we considered more complex problems for which analytical solution can’t be expressed in closed form to get nearer to physical phenomenons.

Problems of the first type are enclosed in himod/tutorials and the are detached in different subdirectories whose names have the structure kindofproblem_rect or kindofproblem_circ to distinguish the problems that are set on a cylinder with a rectangular section rather a circular one. Cases that have been studied by us are enclosed in the second type of subdirectories.

Problems of the second kind are enclosed in himod/testsuite and also in this case subdirectories are differentiated by the end of their name kindofproblem_rect or kindofproblem_circ to distinguish the problems that are set on a cylinder with a rectangular section rather a circular one. Cases that have been studied by us are enclosed in the second type of subdirectories.

Parameter setting is handled by user with parser GetPot so that no direct operations are needed on the code itself.

Theoretic support

As concerns the convergence error rate it is known an a priori error estimation that can be found in [PZ13]

**Theorem 1** Let us consider a domain $\Omega = \Omega_1 \otimes \Omega_2$.

Provided a function $u(x) \in V$, $x = (x_1, x_2) \in \Omega \subset \mathbb{R}^n$, $x_1 \in \mathbb{R}^{n-1}$, $x_2 \in \mathbb{R}$ so that $V \subset H^1(\Omega)$ and considering a set of basis functions $\{\varphi_i(x_2)\}_{i=1}^{\infty} \in H^1(\Omega_2)$ so that:

$$u_N(x) = \sum_{i=0}^{N} u_i(x_1)\varphi_i(x_2) \quad \forall u \in V$$

it can be stated what follows:

$$\|u - u_N\|_{L^2(\Omega)} \leq C(N + 1)^{-m}\|D^{m}_{x_2}u\|_{L^2(\Omega)},$$

$$\|u - u_N\|_{H^1(\Omega)} \leq C(N + 1)^{1-m}\|D^{m}_{x_2}u\|_{L^2(\Omega)},$$

where $m$ is a generic integer number, $C \in \mathbb{R}$ is a constant and $N \in \mathbb{N}$ is the index where the series expansion is truncated.

Applying a Finite Element discretization on $\Omega_1$ this brings to a further theorem that can be applied to our algorithm.

For this we introduce $V_1^h$ which is a finite-dimensional discretization of $H^1(\Omega_1)$ by Finite Element with $h$ as a discretization step and $r$ as the degree of the Lagrange polynomial functions. For sake of simplicity we define $V_2^N$ as the subspace of $H^1(\Omega_2)$ which is detected by $\{\varphi_i\}_{i=1}^{N}$.
Theorem 2 Provided a function $u^h_N(x) \in V^1_h \otimes V^2_N$,

$$u^h_N(x) = \sum_{i=0}^{N} u_i(x_1) \varphi_i(x_2)$$

it can be stated what follows:

$$\|u - u_N\|_{L^2(\Omega)} \leq C \left[ (N + 1)^{-m} + h^{2r} \right] \left[ \|D^r_{x_1} u\|_{L^2(\Omega)} + \|D^m_{x_2} u\|_{L^2(\Omega)} \right].$$

Adopting this result to our case it means that the error in $L^2$-norm of the numerical solution computed by Hi-Mod respect to the exact one is expected to scale as $(\frac{1}{m})^2$, while the error in $H^1$-norm is expected to scale as $(\frac{1}{m})$ because elliptic regularity results guarantee that a function $u \in H^1(\Omega)$, if $\Omega$ is convex, belongs to $H^2(\Omega)$ too.

The Finite Element step has been set to be small enough so that we can neglect its contribution to the global error over the domain. Therefore the model error is the dominant one while mesh error is much smaller and doesn’t affect the order of the global error.

For all the tests we did we set the mesh along the mainstream with 100 Finite Elements and on the transversal section we used 50 modal functions.

Case test1

We have focused on a particular problems whose formulation is as follows:

$$\begin{align*}
-\mu \Delta u + \mathbf{b} \cdot \nabla u + \sigma u &= f(x, \rho, \theta) & \text{in } \Omega \\
u &= 1 - \rho^2 & \text{on } \Gamma_{in} \\
u' &= 0 & \text{on } \Gamma_{out} \\
u &= 0 & \partial \Omega \setminus (\Gamma_{in} \cup \Gamma_{out})
\end{align*}$$

\begin{equation}
\Omega = \{(x, \rho, \theta) : 0 < x < 5, \ 0 < \rho < 1, \ 0 < \theta < 2\pi\},
\end{equation}

$$\Gamma_{in} = \{x = 0, \ 0 < \rho < 5, \ 0 < \theta < 2\pi\}.$$  

We set $\mu = 1$, $\beta_x = 3$, $\beta_\rho = 1$, $\beta_\theta = 1$. In order to have a solution that matches all the boundary conditions we set the exact solution in this way:

$$u_{es}(x, \rho, \theta) = (1 - \rho^2)(5 - x)^2.$$  

As a consequence the force term that we use to figure out numerical solution is:

$$f(x, \rho, \theta) = \mu(2\rho^2 - 2 + 4(5 - x)^2) - 2b_\rho \rho(5 - x)^2 - b_x(1 - \rho^2)(10 - 2x)$$  

Therefore to apply Hierarchical Model Reduction we used modal basis functions that are null on the whole side of the cylinder. The trend of the solution is characterized by a parabolic profile at the inlet section and the maximum value is reached along the axis of the cylinder. Moving towards the outlet section the parabolic profile on the transversal section is expected to decrease its peak progressively due to the presence of the diffusive term.
In the figures of the following page it is represented solution to (20) setting different number of modes along the transversal section. It can be noticed the fact that advection along the direction which is determined by the radius and the other one which is detected by the angle are deadened by diffusion. Instead something different from traditional Poisson problem is noticed for advection along the mainstream.

An improvement of the solution for an increase of the number of modes that are used along the transversal section of the domain is noticeable both from solution’s plot and from convergence rate too.

In our experiments we just computed $L^2$-norm for the error and results are shown in the following pages. You can notice that the empirical curve of the error is characterized by a linear order. It may have a sense because we are using bi-dimensional modal basis functions, while the theoretic results are presented using unidimensional Fourier series expansion. Anyway there are not well-known results about this probability yet. We make appeal to future works on this subject to explore this topic.

In conclusion we present here a comparison between the errors that are related to 3D Finite Elements and Hierarchical Model Reduction setting the number of degrees of freedom the are used on the transversal sections. It can be noticed that the curve which is associated with Hi-Mod stays below the one that is related to Finite Elements and its slope is bigger too. This means that Hierarchical Model Reduction is a proper method to tackle the typical trade-off between expense of computational cost (expressed by the number of degrees of freedom) and accuracy for the solution that is provided.
Figure 4: Solution to Case test 1 using HiMod

Figure 5: Solution to Case test 1 using HiMod
Figure 6: Solution to Case test 1 using FEM method

Figure 7: Solution to Case test 1 using FEM method
Figure 8: Error committed by Hi-Mod for different values of number of modes

Figure 9: Error committed by Hi-Mod and FEM against DOF
Case test 2

In this case test we aimed to analyze a problem which is nearer to real phenomenons. In detail we wanted to model the advection of a medication by a blood in a vein due to the presence of a stent on the wall. We modeled this object and the medication that it progressively releases using e localized force term next to the wall of the domain.

Of course this complex and unsteady phenomenon is not exactly what we are going to reproduce, it is just an hint to what motivated us in setting the parameter of the equation in a certain way rather than another.

Thus the formulation of the problem we have taken into account is the next one:

\[
\begin{cases}
-\mu \Delta u + \mathbf{b} \cdot \nabla u + \sigma u = f & \text{in } \Omega \\
u = 1 - \rho^2 & \text{on } \Gamma_{in} \\
u = 0 & \partial\Omega \setminus \Gamma_{in}
\end{cases}
\]

\[
\Omega = \{(x, \rho, \theta) : 0 < x < 5, \ 0 < \rho < 1, \ 0 < \theta < 2\pi\},
\]
\[
\Gamma_{in} = \{x = 0, \ 0 < \rho < 5, \ 0 < \theta < 2\pi\}.
\]

We set \( \mu = 1, \beta_x = 5, \beta_\rho = 0, \beta_\theta = 0 \). The force term is set to model the presence of an high concentration of medication near the wall, this:

\[
f(x, \rho, \theta) = 1_{[0.7, L_x]}(x) \cdot 1_{[0.7, 1]}(\rho)
\]

The solution which is provided by Hi-Mod is represented in the next page, near the full solution to the same problem that was calculated using 3D Finite Element method. and it can be noticed that the solutions coming from both the methods are very similar. To enforce this sentence we inserted a scheme where we plot the \( L^2 \)-norm distance between the full solution and the reduced one. It still happens that, increasing the number of modal basis functions to be used, the error is rapidly cut down.
Figure 10: Force term for Case test 2 using Hi-Mod method

Figure 11: Force term on a slice for Case test 2 using FEM method
Figure 12: Solution to Case test 2 using Hi-Mod method

Figure 13: Solution to Case test 2 using Hi-Mod method
References


