User's Manual for the Function `mfem_fselib`. A Function for Computing the Finite Element Matrices

Carlos Vilas Fernández

February 4, 2009

1 Introduction

The Finite Element Method (FEM) is a technique for computing the numerical solution of a partial differential equation of the form:

$$da\frac{\partial z}{\partial t} + \overrightarrow{\nabla}\cdot(\overrightarrow{v}z) = \overrightarrow{\nabla}\cdot\left(\kappa(z)\overrightarrow{\nabla}z\right) + f(z) \tag{1}$$

with boundary conditions (BC)

$$\overrightarrow{n}\cdot\left(\kappa(z)\overrightarrow{\nabla}z\right) + qz = g\Big|_{\mathscr{B}}. \tag{2}$$

where $\overrightarrow{\nabla}$ is the gradient operator and $\overrightarrow{n}$ is a unitary vector pointing outwards the spatial domain. It is important to remark that BC (2), typically known as Robin BC, can represent other type of BC. For instance, by setting $q = 0$ and $g = 0$ the classical Neumann BC are recovered, this is $\overrightarrow{n}\cdot\overrightarrow{\nabla}z = 0\Big|_{\mathscr{B}}$. For obtaining Dirichlet BC a large value of q is chosen, for instance $q = 1000$, in such case g will be chosen as $g = qz^*$ where z^* is the value of the field in the boundary. Since q and g are very large, the term $\overrightarrow{n}\cdot\left(\kappa(z)\overrightarrow{\nabla}z\right)$ can be neglected as compared with them, and thus Eqn (2) can be approximated by $qz = g$ which is equivalent to $z = z^*$.

In the FEM, the spatial domain is discretized into a number of finite elements. Through this discretization, the original PDE (1) with BC (2) is approximated by a number of ordinary differential equations (ODE). It is not the aim of this report to give an introduction to the FEM, the reader interested in a deeper insight in the FEM is referred to the literature (Reddy, 1993; Zienkiewicz et al., 2005; Pozrikidis, 2005; García, 2008; Vilas, 2008). The system of ODE can be expressed in matrix form as:

$$da\mathcal{M}\mathcal{M}\frac{dZ}{dt} + \left(\kappa\mathcal{D}\mathcal{M} + v\mathcal{C}\mathcal{M} + q\mathcal{B}\mathcal{M}\right)Z = \mathcal{F} + \mathcal{G} \tag{3}$$

where $\mathcal{MM}$, $\mathcal{DM}$, $\mathcal{CM}$, $\mathcal{BM}$ are, respectively, the mass, diffusion, convection and homogeneous boundary matrices. Z is the discretized version of the field z as indicated in Figure 1. Multiplying Eqn (3) by $\mathcal{MM}^{-1}$ and rearranging the terms one has that:

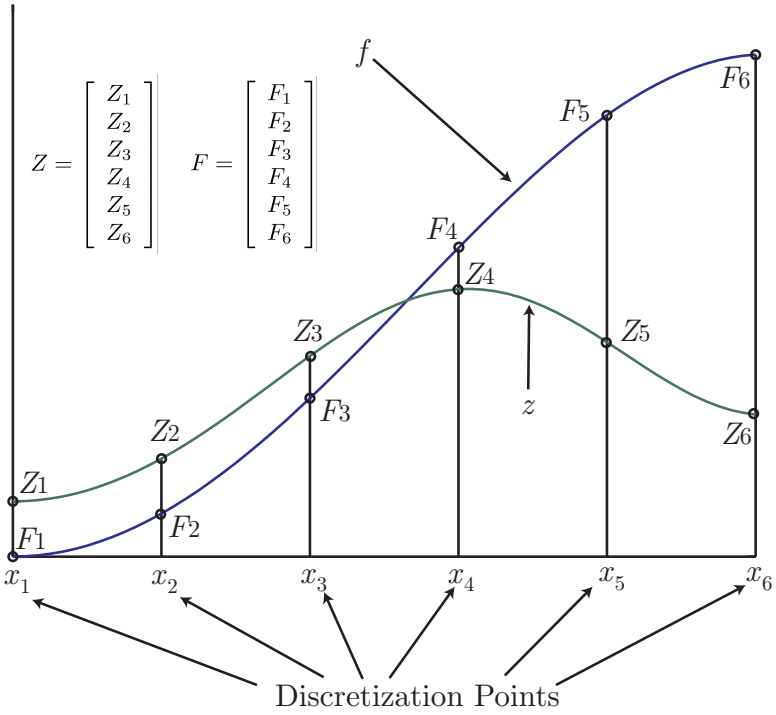

Figure 1: Discretization of field z and function f.

$$da\frac{dZ}{dt} = -\mathcal{MM}^{-1}\left(\kappa\mathcal{DM} + v\mathcal{CM} + q\mathcal{BM}\right)Z + \mathcal{MM}^{-1}\mathcal{F} + \mathcal{MM}^{-1}\mathcal{G} \tag{4}$$

It should be remarked that $F = \mathcal{MM}^{-1}\mathcal{F}$ corresponds to the value of the discretized version of the nonlinear function f as indicated in Figure 1. Finally, $\mathcal{G}$ is a vector with all the elements zero except in the boundary points (in 1D problems these correspond with the first and last points) in which it takes the value of g in Eqn (2). In the same way, matrix $\mathcal{BM}$ is a matrix of zeros except in the boundary (for 1D problems $\mathcal{BM}(1,1)$ and $\mathcal{BM}(N,N)$). In these points it can be:

- 1 for Robin BC
- 0 for Neumann BC
- 10^5 for Dirichlet BC[1].

[1]It should be highlighted that the only requirement on the value of $\mathcal{BM}$ at a Dirichlet boundary condition is that it has to be large. In the matlab function it has been fixed to 10^5 but other $(500, 1000, 2300, 1e6, ...)$ could be employed.

Another advantage of the FEM matrices is that they can be employed for approximating spatial derivatives and integrals by algebraic operations as indicated in Table 1.

	Continuous	**Discrete**
1	$\int_{\mathscr{V}} g(\boldsymbol{\xi}) f(\boldsymbol{\xi}) d\boldsymbol{\xi}$	$\mathcal{G}^T \mathcal{D} \mathcal{A} \mathcal{F}$
2	$\int_{\mathscr{V}} g(\boldsymbol{\xi}) \frac{\partial f(\boldsymbol{\xi})}{\partial \xi_i} \, \mathrm{d}\boldsymbol{\xi}$	$\mathcal{G}^T \mathcal{B} \mathcal{E}_i \mathcal{F}$
3	$\int_{\mathscr{V}} g(\boldsymbol{\xi}) \Delta f(\boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\xi}$	$-\mathcal{G}^T \left(\mathcal{C} + \mathcal{Q} \right) \mathcal{F}$
4	$\frac{\partial}{\partial \xi_i}$	$\mathcal{D} \mathcal{A}^{-1} \mathcal{B} \mathcal{E}_i$
5	$\Delta = \frac{\partial^2}{\partial \xi_1^2} + \frac{\partial^2}{\partial \xi_2^2} + \frac{\partial^2}{\partial \xi_3^2}$	$-\mathcal{D} \mathcal{A}^{-1} \left(\mathcal{C} + \mathcal{Q} \right)$

Table 1: Relationships between the continuous spatial derivatives and integrals and their discrete counterparts using the FEM matrices. Vectors $\mathcal{G}$ and $\mathcal{F}$ are the FEM discretized versions of continuous functions $g(\boldsymbol{\xi})$ and $f(\boldsymbol{\xi})$.

The objective of section 2 is to explain how to use the `mfem_fselib` function for obtaining these matrices.

1.1 The Proper Orthogonal Decomposition

The main inconvenience of classical techniques like the FEM or the FD is that, in general, they result into a large set of ODEs. This is specially critical when dealing with real time tasks as optimization and control. During the last decades a number of techniques for the model reduction of PDEs systems have arosen, among which, one of the most popular is the proper orthogonal decomposition (POD) (Sirovich, 1987; Berkooz et al., 1993).

In dissipative systems, the dynamics evolve to a low dimensional subspace and remain in it in the future. This property will allow us to approximate the dynamics of the system by computing only the dynamics in the low dimensional subspace. The basis functions of such subspace can be computed in many different forms. In the case of the POD, the procedure is as follows:

- Obtain a set of measurements or simulation data representative of the behaviour of the system.
- Compute the kernel $\mathcal{K}(\xi, \xi')$ of the integral equation

$$\phi_i(\xi) = \lambda_i \int_{\mathscr{V}} \mathcal{K}(\xi, \xi') \phi_i(\xi') d\xi', \tag{5}$$

For a discrete set of data (Z_i), the kernel is computed as:

$$\mathcal{K} = \frac{1}{\ell} \sum_{i=1}^{\ell} Z_i Z_i^T. \tag{6}$$

where ℓ number of elements in the data set.

- Solve the integral equation (5).

It must be pointed out that for large values of N, solving Eqn (5) can be computationally involved. In order to avoid this problem, a useful alternative, proposed in Sirovich (1987) and known as the *method of snapshots* or *strobes*, is briefly discussed. In this method, each eigenfunction is expressed in terms of the original data as:

$$\phi_j = \sum_{i=1}^{\ell} w_i^j Z_i, \tag{7}$$

where w_i^j are the weights to be computed. To this purpose, the following matrix is defined

$$\mathcal{M}_{ij} = \frac{1}{\ell} \langle Z_i, Z_j \rangle_{\mathcal{V}}. \tag{8}$$

Introducing Eqns (6) and (7) in the eigenvalue problem (5), results into:

$$\mathcal{M}\mathcal{W}_j = \lambda_j \mathcal{W}_j, \tag{9}$$

where the eigenvectors $\mathcal{W}_j$ have as element the weights in equation (7) so that $\mathcal{W}_j = [w_1^j, w_2^j, ..., w_\ell^j]^T$.

A more detailed description, including references, about this technique can be obtained in the thesis (Vilas, 2008; García, 2008).

In the section 3 the function `pod_computation` will be employed for computing the basis functions which will define the low dimensional subespace.

2 The Function `mfem_fselib`

This function was constructed from the free library `FSELIB` (see Pozrikidis (2005) for details). The original library consists of a number of Matlab© functions which allow us to carry out different steps of the FEM: from spatial domain discretization (using different techniques) to the computation of the FEM matrices (using different boundary conditions, basis functions,...). The `mfem_fselib` presented in this report is limited to the 1D case using first and

second order polynomials[2] as basis functions and the user is responsible for defining the spatial domain discretization. Nevertheless, as an advantage, this function is more transparent to the user than the FSELIB library. This will allow the user to obtain the FEM matrices and to deal with the boundary conditions in a more systematic way. The way of calling the function is

```
[fem_im] = mfem_fselib (xe, bc_x0, bc_xL,fem_om)
```

where the input and output parameters are

```
Input parameters:
xe:       Coordinates of the spatial discretization points.
bc_x0:    Type of boundary conditions at the beginning of the domain.
bc_xL:    Type of boundary conditions at the end of the domain.
fem_om:   This variable indicates the FEM matrices that the user wants to
          compute. The matrix is indicated by means of a string: 'MM' for
          the mass matrix, 'DM' for the diffusion matrix, 'CM' for the
          convection matrix and 'BM' for the boundary matrix. The last
          element of this variable indicates the type of basis functions
          employed: 'lin' for linear and 'quad' for quadratic. In the case
          of linear basis functions the last element can be omitted.

The boundary type can be 'dir' (Dirichlet boundary conditions), 'neu'
(Neumann boundary conditions) or 'rob' (Robin boundary conditions)

Output parameters:
fem_im:   The output parameters are the matrices indicated in fem_om.
          They must be sorted following the same sequence used in fem_om.
          If quadratic elements are chosen, it may result convenient to
          recover the mesh with the interior points. This is done by
          adding a new output variable.
```

[2]For constructing these new discretization points are considered. These points are located at the middle of each finite element.

2.1 A simple example

Compute the mass and the boundary matrices, using linear basis functions, with Dirichlet boundary conditions in the first point and Neumann in the last point. The spatial domain goes from 0 to π and 21 spatial discretization points are required. In Matlab© this is carried out by:

```
xe               = linspace(0,pi,21);  % Spatial discretization
[mass, boundary] = mfem_fselib (xe, 'dir', 'neu','MM','BM');
```

This is equivalent to

```
xe               = linspace(0,pi,21);  % Spatial discretization
[mass, boundary] = mfem_fselib (xe, 'dir', 'neu','MM','BM','lin');
```

When quadratic basis functions are chosen, a new input parameter is introduced, so that:

```
xe                   = linspace(0,pi,21);  % Spatial discretization
[mass, boundary,xex] = mfem_fselib (xe, 'dir', 'neu','MM','BM', 'quad');
```

Note that an extra output parameter (`xex`) is obtained. `xex` contains the extended mesh with the interior points. For details about this, the reader is referred to the bibliography (Reddy, 1993; Zienkiewicz et al., 2005; Pozrikidis, 2005).

3 The Function `pod_computation`

This function will compute the basis functions using the POD technique from a set of experimental or simulation data. The discretization of the data must coincide with the discretization of the finite element method. The way of calling the function is

```
[pods] = pod_computation(V , MM , mth , ener)
```

where the input and output parameters are

```
Input variables:
V:    Set of simulation or experimental data. Each column corresponds with
      a snapshot at a given time
MM:   Mass matrix obtained from the FEM. The FEM discretization must
      coincide with the discretization for V
```

```
mth:  Method used for computing the basis functions. 'd' is the direct
      method, 'i' is the indirect method. The indirect method results
      more efficient if the number of discretization points is much larger
      than the number of measurements.
ener: Energy captured by the PODs.

Output variables:
pods: This is a structure where pods.phi are the basis functions and
      pods.lambda the eigenvalues
```

It is important to note that the rows of the data matrix V are related to the FEM spatial discretization while the data at different times are stored in the columns.

4 Examples

In this section some problems will be employed to illustrate the use of the `mfem_fselib` and `pod_computation` functions.

4.1 A simple diffusion problem: FEM solution

Use the `mfem_fselib` function to compute the numerical solution of equation:

$$\frac{\partial z}{\partial t} = \kappa \frac{\partial^2 z}{\partial x^2}; \qquad \kappa = 0.1 \tag{10}$$

with boundary conditions

$$z(0,t) = 3, \qquad \frac{\partial z(L,t)}{\partial x} = 0. \tag{11}$$

These are Dirichlet in the first point and Neumann homogeneous in the last point. The spatial domain is $\mathscr{V} = \{x/x \in [0,\pi]\}$. The initial conditions are chosen as $z(x,0) = -x^2 + 2\pi x + 3$ and the number of discretization points will be $N = 21$. Since no convection is included in the formulation, the FEM matrices to be computed are: $\mathcal{MM}$, $\mathcal{DM}$ and $\mathcal{BM}$ which is done by:

```
L       = pi;                  % Length of the spatial domain
ndisc   = 21;                  % Number of discretization points
xe      = linspace(0,L,ndisc); % Spatial discretization
```

```
% Call to the function which computes the FEM matrices
[MM, DM, BM] = mfem_fselib (xe, 'dir', 'neu','MM','DM','BM');
inv_MM       = inv(MM);
```

Note that $\mathcal{BM}(1,1) = 10^5$ (Dirichlet boundary conditions) and $\mathcal{BM}(N,N) = 0$ (Neumann homogeneous boundary conditions). Now we have to define the vector $\mathcal{G}$. Since the boundary conditions are constant, this vector can be defined in the main program. In the first point we have Dirichlet BC so, according to the explanation given in the introduction, the value of g in this point must be $g = qz^*$, thus $\mathcal{G}(1) = 3\mathcal{BM}(1,1)$. In the last point we have $g = 0$. The rest of the code is standard and it is included in `Ex1.m` and `ode_ex1.m`.

When quadratic basis functions are considered, the code is modified so that

```
% Call to the function which computes the FEM matrices
[MM, DM, BM,xe] = mfem_fselib (xe, 'dir', 'neu','MM','DM','BM','quad');
ndisc = length(xe);
inv_MM          = inv(MM);
```

Note that now the number of discretization points (`ndisc`) is recomputed for taking into account the interior points. The rest of the code is the same as in the linear case and it is included in `Ex1_quad.m`.

4.2 A simple diffusion problem: POD solution

The example considered here is the same as that of the previous section. As mentioned in section 3, the first step to obtain the basis functions in the POD method is to generate a set of snapshots (measurements of the real system or simulation data) representative of the behaviour of the system. In this case, the snapshots were obtained by means of simulation data taken each 1 unit of time till 150 and they were saved in the file `data_simulation_Ex1.mat`. Thus, in the code we first load the snapshots:

```
load data_simulation_Ex1
```

After this, we compute the FEM matrices as indicated before and compute the basis functions by typing:

```
[pods]  = pod_computation(Z , MM , 'd' , 99.99998);
Phi     = pods.phi;
neig    = size(Phi , 2);
```

It is easy to see, by means of the parameter `neig`, that 3 basis functions are enough to capture the 99.99998% of the energy. The next step is to project Eqn (10) over the basis functions, this is:

$$\int_{\mathscr{V}} \Phi \frac{\partial z}{\partial t} d\xi = \int_{\mathscr{V}} \kappa \Phi \frac{\partial^2 z}{\partial x^2} d\xi$$

which, taking into account the boundary conditions, the orthonormality of basis functions and the relationships of Table 1 can be rewritten in discrete form as:

$$m_t = -\Phi^T M M M M^{-1} (\kappa D M + B M) \Phi m = -\Phi^T (\kappa D M + B M) \Phi m$$

with initial conditions:

$$m_0 = \Phi^T M M z_0$$

So in the Matlab code, we can define a new matrix `Sp_oper` and a new vector `G` of the form:

```
Sp_oper = - Phi'*(k*DM + BM)*Phi;
G       = Phi'*G;
```

so that the system of ODEs remains:

```
dm = Sp_oper*m + G;
```

Note that now we are solving 3 ODEs instead 21 and the error between the FEM and the POD is lower than 0.4% for all $t > 0$. If we consider the same example but capturing the 99.99% of the energy, the number of basis functions will be two. In this case the error at $t = 1$ will be around 3% and for $t > 10$ decreases to values lower than 1%.

The complete code is included in the matlab scripts `Ex1_POD.m` and `ode_Ex1_POD.m`.

4.3 Reaction-Diffusion-Convection problem: FEM solution

In this example reaction and convection terms are considered. The model equations are

$$\frac{\partial z}{\partial t} = \kappa \frac{\partial^2 z}{\partial x^2} - v \frac{\partial z}{\partial x} + f(z); \qquad f(z) = z - z^3 \tag{12}$$

where $\kappa = 0.1$ and $v = 0.01$, with the usual Danckwerts boundary conditions

$$\vec{\mathbf{n}} \cdot \kappa \vec{\boldsymbol{\nabla}} z \Big|_{x=0} = v(z_{in} - z|_{x=0}) \iff \kappa \left. \frac{\partial z}{\partial x} \right|_{x=0} = v(z|_{x=0} - z_{in}) \tag{13}$$

$$\vec{\mathbf{n}} \cdot \vec{\boldsymbol{\nabla}} z \Big|_{x=L} = 0, \tag{14}$$

with z_{in} being the concentration in the inlet stream ($z_{in} = 100\,|sin(5t)|$). The spatial domain is $\mathscr{V} = \{x/x \in [0, \pi]\}$. The initial conditions are $z(x, 0) = (-x^2 + 2\pi x + 2\pi)0.25$. Regarding the spatial discretization, in this example, a variable element size will be employed. In this regard, from $x = 0$ to $x = L/3$, 11 elements points will be employed. In the rest of the domain 10 elements will be employed. This discretization can be carried out, for instance, as follows:

```
L       = pi;                  % Length of the spatial domain
ndisc   = 21;                  % Number of discretization points
n1      = 11;                  % First part of the reactor
n2      = ndisc - n1 + 1;      % Second part of the reactor
xe1     = linspace(0,L/3,n1);  % Spatial discretization in the first part
xe2     = linspace(L/3,L,n2);  % Spatial discretization in the first part
xe      = [xe1 xe2(2:end)];    % Total spatial discretization
```

In this case, the FEM matrices are computed as follows:

```
% Call to the function which computes the FEM matrices
[MM, DM, CM, BM] = mfem_fselib (xe, 'rob', 'neu','MM','DM', 'CM', 'BM');
inv_MM = inv(MM);     % This will be employed later
G = zeros(ndisc , 1); % For the boundary conditions
```

Note that, in this example, the boundary conditions are not constant so they cannot be defined in the main program. The dimensions of matrix $\mathcal{G}$ can be defined in the main file as shown above but it has to be filled in the function where the ODEs as indicated below:

```
function dz = ode_ex2(t, z, Sp_oper, G, inv_MM, v)


% Boundary conditions
zin   = 100*abs(sin(5*t)); % Inlet concentration
G(1)  = v*zin;             % Boundary condition in the first point
GG    = inv_MM*G;
```

`GG` in this code corresponds to $\mathcal{M}\mathcal{M}^{-1}\mathcal{G}$ in Eqn (4). The rest of the code is included in `Ex2.m` and `ode_ex2.m`.

When quadratic basis functions are considered, the code is modified so that

```
[MM, DM, CM, BM, xe] = mfem_fselib (xe, 'rob', 'neu','MM','DM', 'CM',...
                                    'BM', 'quad');
ndisc                = length(xe);
```

The rest of the code is the same as in the linear case and it is included in `Ex2_quad.m`.

4.4 Reaction-Diffusion-Convection problem: POD solution

As in the previous case, the first step is to obtain the basis functions from the snapshots which were previously generated and saved in the file `data_simulation_Ex2`:

```
load data_simulation_Ex2
```

Now we define the spatial domain and the discretization and compute the FEM matrices. After these steps, the basis functions are computed as follows:

```
[pods]  = pod_computation(Z , MM , 'd' , 99.9999);
Phi     = pods.phi;
neig    = size(Phi , 2);
proj_op = Phi'*MM;          % Projection operator
```

In this case a 99.9999% of the energy implies using 5 basis functions. The next step is to project the spatial operator and the initial conditions.

```
Sp_oper = - Phi'*(k*DM + v*BM + v*CM)*Phi;


% Initial conditions
z0      = (-xe.^2 + 2*pi*xe + 2*pi)/4 ;
m0      = proj_op*z0;
```

In the script where the ODEs are defined we need to project the boundary conditions and the nonlinear term[3]:

```
 % Recovery the original field
z    = Phi*m;


% Boundary conditions
```

[3]Note that $\Phi^T MMMM^{-1}G = \Phi^T G$

```
zin  = 100*abs(sin(5*t));  % Inlet concentration
G(1) = v*zin;              % Boundary condition in the first point
GG   = Phi'*G;             % Projection of the BC


% Nonlinear term
f    = z - z.^3;
f    = proj_op*f;          % Projection of the nonlinear function
```

With 5 basis functions the error is always lower than 0.5%. The complete code is included in the matlab scripts `Ex2_POD.m` and `ode_Ex2_POD.m`.

4.5 The Burgers equation: FEM solution

The well known Burgers equation presents the following form:

$$\frac{\partial z}{\partial t} = \frac{\partial(-0.5z^2)}{\partial x} + \mu\frac{\partial^2 z}{\partial x^2} = -z\frac{\partial z}{\partial x} + \mu\frac{\partial^2 z}{\partial x^2} \tag{15}$$

The spatial domain is $\mathscr{V} = \{x/x \in [0,1]\}$ and Dirichlet boundary conditions are considered at both sides of the domain. The value for the boundary and initial conditions is obtained through the analytical solution given by:

$$z = \frac{0.1ea + 0.5eb + ec}{ea + eb + ec},$$

with:

$$ea = \exp\left(\frac{0.05}{\mu}(x - 0.5 + 4.95t)\right); \quad eb = \exp\left(\frac{0.25}{\mu}(x - 0.5 + 0.75 * t)\right).$$

$$ec = \exp\left(\frac{0.5}{\mu}(x - 0.375)\right).$$

These equations are included into a Matlab© function included in the toolbox and named `burgers_exact`. The number of discretization points will be $N = 201$:

```
%... spatial grid
x0 = 0.0;
xL = 1.0;
n  = 201;
dx = (xL-x0)/(n-1);
x  = [x0 : dx : xL]';
```

The FEM matrices are computed as follows:

```
[MM, DM, CM, BM]   = mfem_fselib(x, 'dir', 'dir', 'MM', 'DM', 'CM', 'BM');
inv_MM             = inv(MM);
G                  = zeros(n , 1); % For the boundary conditions
Lapl_op            = -inv_MM*DM;   % Laplacian operator
```

Note that $\mathcal{BM}(1,1) = \mathcal{BM}(N,N) = 10^5$, since Dirichlet boundary conditions are considered at both sides. In this case we decide to compute together the homogeneous part of the boundary -this is, the term qz in Eqn (2)- and the inhomogeneous part -this is, the term g in Eqn (2)-. Thus in the code where the ODEs are defined we have to add:

```
%... boundary conditions at z = 0
zin    = burgers_exact(x0,t);
G(1)   = BM(1,1)*(z(1) - zin);
%... boundary conditions at z = zL
zout   = burgers_exact(xL,t);
G(n)   = BM(n,n)*(z(n) - zout);
GG     = -inv_MM*G;
```

The main difference with respect to the other examples of this manual is the form of the convective term. For defining it, we use the relations of Table 1 and compute the gradient operator as:

```
Grad_op = inv_MM*CM;   % Gradient operator
```

The convective term is contructed, in the script were the ODEs are defined, as:

```
fx    = Grad_op*(0.5*z.^2);
```

The rest of the code is standard and it is included in the scripts `burgers_main_FEM.m` and `burgers_pde_FEM.m`.

When quadratic basis functions are considered, the code is modified so that

```
% Call to the mfem_fselib function for computing the FEM matrices
[MM, DM, CM, BM,x] = mfem_fselib(x, 'dir', 'dir', 'MM', 'DM', 'CM',...
 'BM', 'quad');
n                  = length(x);
```

The rest of the code is the same as in the linear case and it is included in the matlab script `burgers_main_FEM_quad.m`.

4.6 The Burgers equation: POD solution

The procedure is the same than in the above examples. First we obtain the basis functions from the snapshots taken by means of a FEM simulation. In this case the time interval between two consecutive measurements was 0.001 units of time and the simulation data were save in a matlab file `data_simulation_burgers.mat`.

```
load data_simulation_burgers
```

After defining the spatial grid, we compute the FEM matrices and the basis functions.

```
%... call to the mfem_fselib function for computing the FEM matrices
[MM, DM, CM, BM]   = mfem_fselib(x, 'dir', 'dir', 'MM', 'DM', 'CM', 'BM');
inv_MM             = inv(MM);


% Computation of the basis functions
[pods]  = pod_computation(Z , MM , 'd' , 99.9999);
Phi     = pods.phi;
neig    = size(Phi , 2);
proj_op = Phi'*MM;          % Projection operator
```

In this problem, the only difference with respect to the others is the convection term. The projection of this term is carried out in the same manner as if it was a nonlinear term:

```
fx     = Grad_op*(0.5*z.^2);
pfx    = proj_op*fx;        % Projection of the convection term
```

It must be remarked that due to the special form of the convection term and the boundary conditions, this problem dissipates energy at a very low rate. This fact can be seen by comparing the relation between eigenvalues:

$$\frac{\lambda_1}{\lambda_2} \approx 20, \quad \frac{\lambda_2}{\lambda_3} \approx 3.5, \quad \frac{\lambda_3}{\lambda_4} \approx 2.5,$$

while in other problems which dissipate much more energy, the relation is much bigger. For instance in the Reaction-Diffusion-Convection problem presented in this manual, the relation

is:

$$\frac{\lambda_1}{\lambda_2} \approx 100, \qquad \frac{\lambda_2}{\lambda_3} \approx 41, \qquad \frac{\lambda_3}{\lambda_4} \approx 2.8,$$

This fact translates into a large number of basis functions to be employed and even using such a number, some oscillations appear in the ROM which lead to error.

References

Berkooz, G., Holmes, P., and Lumley, L. (1993). The Proper Orthogonal Decomposition in the analysis of turbulent flows. *Ann. Rev. Fluid Mech.*, 25:539–575.

García, M. R. (2008). *Identification and Real Time Optimisation in the Food Processing and Biotechnology Industries*. PhD thesis, University of Vigo, Spain. Available online at http://digital.csic.es/handle/10261/4662.

Pozrikidis, C. (2005). *Introduction to Finite and Spectral Element Methods using Matlab*. Chapman & Hall/CRC.

Reddy, J. N. (1993). *An Introduction to the Finite Element Method*. McGraw-Hill, 2nd edition.

Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. Part I: Coherent structures. *Quaterly of Appl. Math.*, 45(3):561–571.

Vilas, C. (2008). *Modelling, Simulation and Robust Control of Distributed Processes: Application to Chemical and Biological Systems*. PhD thesis, University of Vigo, Spain. Available online at http://digital.csic.es/handle/10261/4236.

Zienkiewicz, O. C., Taylor, R. L., and Zhu, J. Z. (2005). *The Finite Element Method: Its Basis & Fundamentals*. Elsevier, Amsterdam, 6^{th} edition.