Hands-on OpenFOAM tutorial on uncertainty quantification

Robert Sawko

STFC, IBM Research

7th of July 2016, Warwick University

Outline

Burgers' equation

We will consider a 1D Burgers' equation on cyclic domain:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \quad x \in [0, 2\pi], \quad t \geq 0 \tag{1}$$

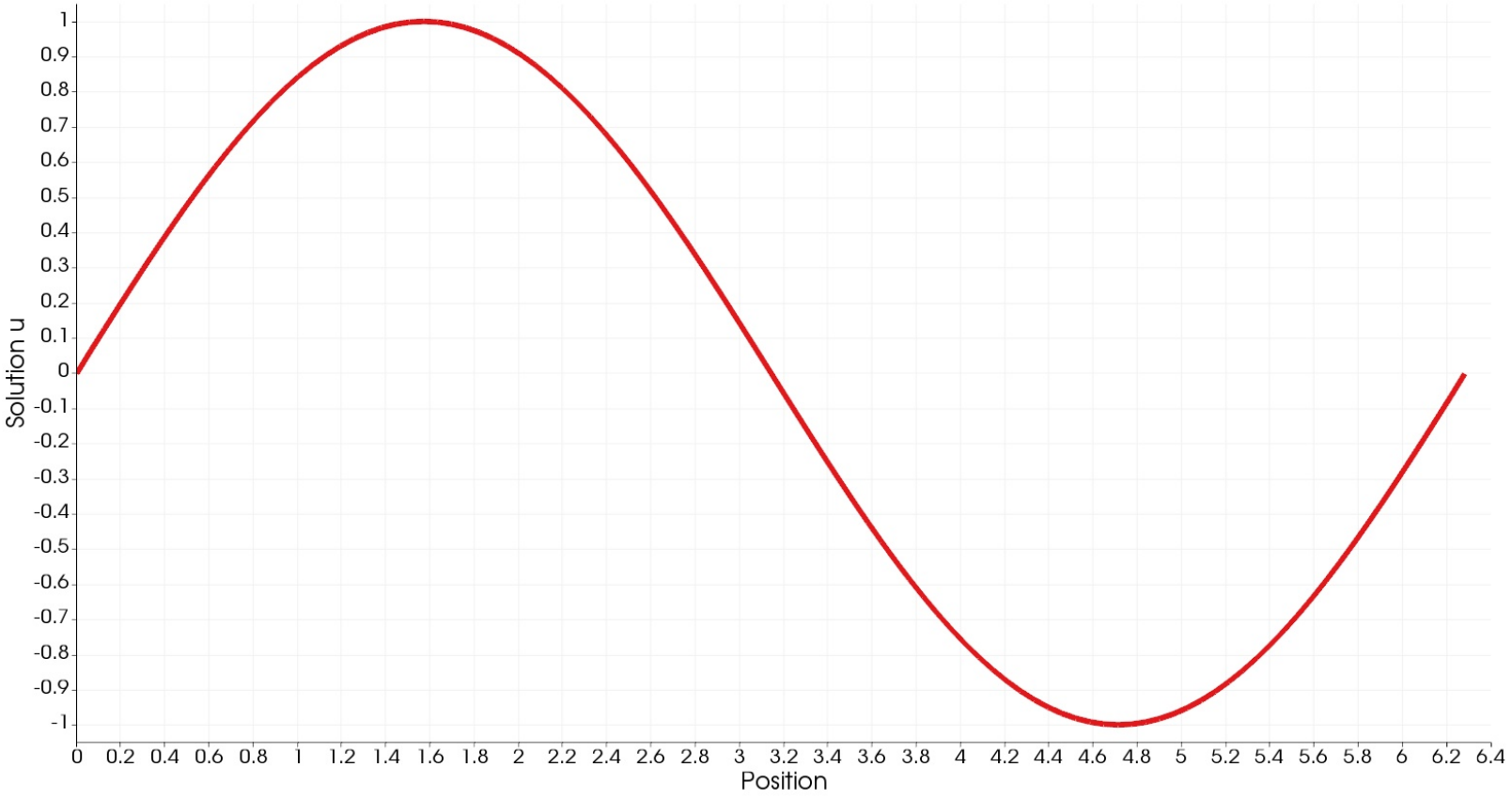
$$u(x, 0) = u_0(x) \tag{2}$$

$$u(2\pi, t) = u(0, t) \tag{3}$$

Some interesting properties

- Simple non-linear model problem for fluids.
- Develops discontinuous solutions from smooth conditions.

Example solution



For a sinusoidal initial condition the solution forms a shock.

Uncertainty quantification and Burgers' equation

We will assume that the source of uncertainty in our solution is due to initial condition:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \quad x \in [0, 2\pi], \quad t \geq 0 \tag{4}$$

$$u(x, 0; \omega) = u_0(x; \omega) \tag{5}$$

$$u(2\pi, t; \omega) = u(0, t; \omega) \tag{6}$$

where ω is an element from a sample space. We are interested in the vanishing viscosity limit. This is a simplification of a problem considered by Heryrim Cho(2014)[1].

[1] Heyrim Cho, Daniele Venturi, and George E Karniadakis. "Statistical analysis and simulation of random shocks in stochastic Burgers' equation". In: Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 470.2171 (2014). ISSN: 1364-5021. DOI: 10.1098/rspa.2014.0080.

Stochastic Galerkin method

Generalized polynomial chaos (gPC) stochastic Galerkin method can be formulated as follows[2]:

1. Choose orthogonal polynomial basis and assume the solution u can be approximated by

$$\hat{u}_N(x,t,Z) = \sum_{i=0}^{N} \hat{v}_i(x,t)\Phi_i(Z), \tag{7}$$

2. plug this approximation into your original equation and perform a projection onto the set of basis function

$$\mathbb{E}\left[\frac{\partial \hat{u}_N}{\partial t}\Phi_k\right] + \mathbb{E}\left[\hat{u}_N\frac{\partial \hat{u}_N}{\partial x}\Phi_k\right] = \nu\mathbb{E}\left[\frac{\partial^2 \hat{u}_N}{\partial x^2}\Phi_k\right], \tag{8}$$

   for $k = 0, \ldots, N$.

3. Use the orthogonality $\mathbb{E}\left[\Phi_i\Phi_j\right] = \gamma_i\delta_{ij}$.

[2] D. Xiu. Numerical Methods for Stochastic Computations: A Spectral Method Approach. Princeton University Press, 2010.

Stochastic Galerkin method - continued

The final form for the set of equations for the coefficients in gPC expansion is:

$$\frac{\partial \hat{v}_k}{\partial t} + \frac{1}{\gamma_k} \sum_{i=0}^{N} \sum_{j=0}^{N} \hat{v}_i \frac{\partial \hat{v}_j}{\partial x} e_{ijk} = \nu \frac{\partial^2 \hat{v}_k}{\partial x^2}, \quad k = 0, \cdots, N, \tag{9}$$

where $e_{ijk} = \mathbb{E}\left[\Phi_i \Phi_j \Phi_k\right]$ and $\gamma_k = \mathbb{E}\left[\Phi_k^2\right]$.

$N = 2$ in orthonormal basis

$$
\begin{cases}
\dfrac{\partial \hat{v}_0}{\partial t} + \dfrac{1}{2}\left(\dfrac{\partial \hat{v}_0^2}{\partial x} + \dfrac{\partial \hat{v}_1^2}{\partial x} + \dfrac{\partial \hat{v}_2^2}{\partial x}\right) = \nu \dfrac{\partial^2 \hat{v}_0}{\partial x^2}, \\
\dfrac{\partial \hat{v}_1}{\partial t} + \dfrac{\partial}{\partial x}\left(\hat{v}_0 \hat{v}_1\right) + \sqrt{2}\dfrac{\partial}{\partial x}\left(\hat{v}_1 \hat{v}_2\right) = \nu \dfrac{\partial^2 \hat{v}_1}{\partial x^2}, \\
\dfrac{\partial \hat{v}_2}{\partial t} + \dfrac{\partial}{\partial x}\left(\hat{v}_0 \hat{v}_2\right) + \dfrac{\sqrt{2}}{2}\dfrac{\partial \hat{v}_1^2}{\partial x} + \dfrac{\sqrt{3}}{2}\dfrac{\partial}{\partial x}\left(\hat{v}_1 \hat{v}_3\right) + \sqrt{2}\dfrac{\partial \hat{v}_2^2}{\partial x} \\
\qquad + \sqrt{3}\dfrac{\partial}{\partial x}\left(\hat{v}_2 \hat{v}_4\right) + \dfrac{3\sqrt{3}}{2}\hat{v}_3 \dfrac{\partial \hat{v}_3^2}{\partial x} + 2\sqrt{2}\dfrac{\partial \hat{v}_4^2}{\partial x} = \nu \dfrac{\partial^2 \hat{v}_2}{\partial x^2},
\end{cases}
\tag{10}
$$

Back to our problem

For simplicity we will consider a translation of the sin:

$$u_0(x;\omega) = \mathsf{sin}(x) + X(\omega) \tag{11}$$

and X has a normal distribution. For such a scenario Hermite polynomials, are the orthogonal basis.

Overview for today

Objective

Extend `burgersFoam` to a gPC Burgers' spectral Galerkin method.

Steps:

- Run `burgersFoam` with sinusoidal initialisation.
- Introduce UQ input file with polynomial order.
- Introduce and read new fields.
- Reimplement the solution to solve a system of equations.
- Reimplement the equations to introduce triplets.
- (Optional) library for different polynomials.

Outline

Outline

Preliminaries

1. Open the command line and type

```
echo $PATH
```

2. Type:

```
OF21 #To load OpenFOAM 2.1 environment
```

3. See how PATH has changed

```
echo $PATH #Observe how it has changed
```

4. Preview new environmental variables

```
echo $FOAM_#HIT TAB!
```

5. Some useful variables

```
echo $FOAM_INST_DIR #Where OpenFOAM is installed
echo $FOAM_RUN #Where you run your cases
echo $FOAM_TUTORIALS #Where the tutorials are
echo $FOAM_APPBIN #Where all the applications are
echo $FOAM_USER_APPBIN #User compiled applications
```

Check out the solver

- use `run` alias,
- in your `$FOAM_RUN` clone the repository,

```
git clone https://github.com/robertsawko/pdesoft2016-burgers-uq.git
```

- checkout `start` tag,

```
cd pdesoft2016-burgers-uq
git checkout start
```

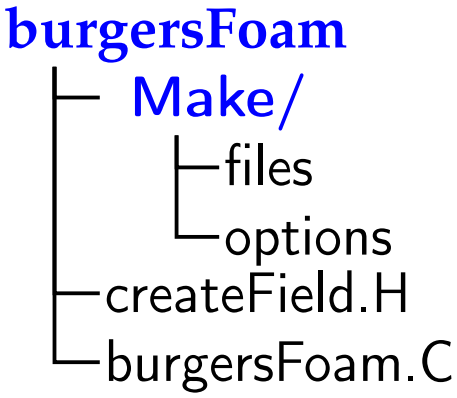
- `ls` the top-level structure

```
ls
burgersFoam  case
```

Reviewing and building the solver

```
createFields.H burgersFoam.C Make
```

```
ls Make
files options
```



Files in OF case directory.

From OF user guide.

Meaning of files

Make/files	name of the application and the name of the file with `main`.
Make/options	additional includes for `wmake`.
.C .H	source files

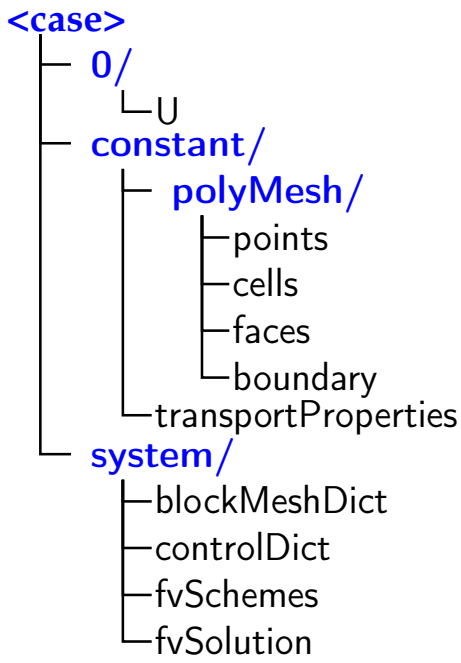
Reviewing and running the case

```
ls
0  constant  system
```

```
ls *
0:
U

constant:
polyMesh  transportProperties

system:
controlDict  fvSchemes  fvSolution
```



Meaning of directories

constant	physical models, physical constant, geometry (`constant/polyMesh`)
0	initial and boundary conditions
system	numerical set up, simulation controls, term discretisation

More information in OF user guide.

Testing the basic solver

- build the solver,

```
cd burgersFoam
wmake
```

- prepare the case,

```
cd ../case
blockMeshDict
funkySetFields -time 0
```

- run the case,

```
burgersFoam | tee log.burgersFoam
```

- post-process with paraFoam,

Introducing new input file

- in `createFields.H` copy `transportProperties` section and change name:

```
IOdictionary UQProperties
(
    IOobject
    (
        "UQProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ,
        IOobject::NO_WRITE
    )
);
```

- in `createFields.H` add read of a label (like int)

```
label order
(
    readLabel(UQProperties.lookup("order")) //needs readLabel!
);
```

- in `case` add new file in `constant/`.

Introducing new fields

- in `createFields.H` introduce new fields as `PtrList`

```
PtrList<volVectorField> Uhat(order + 1);
forAll(Uhat, i)
{
    Info<< "Reading field Uhat" << i <<endl;
    Uhat.set(i, new volVectorField(
        IOobject
        (
            "Uhat" + std::to_string(i),
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh));
}
```

- in `case` add new new fields in `0`.

Introducing new fields

If you're not C++11!

- in `gPCBurgersFoam.H` include `sstream` header,
- in `createFields.H` introduce new fields as `PtrList`

```
PtrList<volVectorField> Uhat(order + 1);
forAll(Uhat, i)
{
    std::stringstream variable_name;
    variable_name << "Uhat" << i;
    Uhat.set(i, new volVectorField(
        IOobject
        (
            variable_name.str(),
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh));
}
```

- in `case` add new new fields in `0`.

What about the flux?

- in `createFields.H` add your own fluxes

```
PtrList<volVectorField> Uhat(order + 1);
PtrList<surfaceScalarField> phihat(order + 1);
forAll(Uhat, i)
{
    ...
    phihat.set(i, new surfaceScalarField(
        IOobject
        (
            "phihat" + std::to_string(i),
            runTime.timeName(),
            mesh,
            IOobject::READ_IF_PRESENT,
            IOobject::AUTO_WRITE
        ),
        linearInterpolate(Uhat[i]) & mesh.Sf()));
}
```

- remove reference to `U` and `phi` variables,

Introducing new fields to case setup

- in `case` change `fvSchemes` and `fvSolution` (use regular expressions!),

```
divSchemes
{
    default          none;
    "div\(phihat.*,Uhat.*\)"      Gauss linear;
}

laplacianSchemes
{
    default          none;
    "laplacian\(nu,Uhat.*\)" Gauss linear orthogonal;
}
```

- test the program,

Solution

- implement a sequential solution of the system of equations,

$$\begin{cases} \dfrac{\partial \hat{v}_0}{\partial t} + \dfrac{1}{2}\dfrac{\partial \hat{v}_0^2}{\partial x} = \nu \dfrac{\partial^2 \hat{v}_0}{\partial x^2}, \\ \dfrac{\partial \hat{v}_1}{\partial t} + \dfrac{1}{2}\dfrac{\partial \hat{v}_1^2}{\partial x} = \nu \dfrac{\partial^2 \hat{v}_1}{\partial x^2}, \\ \dfrac{\partial \hat{v}_2}{\partial t} + \dfrac{1}{2}\dfrac{\partial \hat{v}_2^2}{\partial x} = \nu \dfrac{\partial^2 \hat{v}_2}{\partial x^2}, \end{cases} \tag{12}$$

```
forAll(Uhat, i)
{
    solve
    (
        fvm::ddt(Uhat[i])
      + fvm::div(phihat[i], Uhat[i])
      - fvm::laplacian(nu, Uhat[i])
    );
    phihat[i] = linearInterpolate(Uhat[i]) & mesh.Sf();
}
```

Initial condition

- in `case` change the files in `0/` directory

$$\hat{v}_0(x) = \sin(x), \quad \hat{v}_1(x) = 1, \quad \hat{v}_2(x) = 0 \tag{13}$$

Triplets

$$\frac{\partial \hat{v}_k}{\partial t} + \frac{1}{\gamma_k} \sum_{i=0}^{N} \sum_{j=0}^{N} \hat{v}_i \frac{\partial \hat{v}_j}{\partial x} e_{ijk} = \nu \frac{\partial^2 \hat{v}_k}{\partial x^2},$$

where $e_{ijk} = \mathbb{E}\left[\Phi_i \Phi_j \Phi_k\right]$ and $\gamma_k = \mathbb{E}\left[\Phi_k^2\right]$.

$$\mathbb{E}\left[\Phi_i \Phi_j \Phi_k\right] = \int_{-\infty}^{\infty} \Phi_i(\xi) \Phi_j(\xi) \Phi_k(\xi) w(\xi)\, d\xi \qquad (14)$$

Introduce third order tensor

- in `createFields.H` introduce a new table with values from a symbolic calculation,

```
scalar e[3][3][3];
e[0][0][0] = 1.0;
e[1][1][0] = 1.0;
e[2][2][0] = 1.0;

e[0][1][1] = 1.0;
e[1][0][1] = 1.0;
e[1][2][1] = Foam::sqrt(2.0);
e[2][1][1] = Foam::sqrt(2.0);

e[0][2][2] = 1.0;
e[2][0][2] = 1.0;
e[1][1][2] = Foam::sqrt(2.0);
e[2][2][2] = 2.0 * Foam::sqrt(2.0);
```

Correct equation terms

- in `gPCBurgersFoam.C` incoroprate $e_{i,j,k}$ tensor in "convection" term

```
fvVectorMatrix UhatEqn(
    fvm::ddt(Uhat[k])
    -
    fvm::laplacian(nu, Uhat[k])
);

forAll(Uhat, i){
    forAll(Uhat, j){
        UhatEqn += e[i][j][k] * fvm::div(phihat[i], Uhat[j]);
    }
}
solve(UhatEqn);
phihat[k] = linearInterpolate(Uhat[k]) & mesh.Sf();
```

- try out the case (will not work!),

Make it work

- in `gPCBurgersFoam.C` use `fvc` and `fvm` where necessary,

```
fvVectorMatrix UhatEqn(
    fvm::ddt(Uhat[k])
    -
    fvm::laplacian(nu, Uhat[k])
);

forAll(Uhat, i){
    forAll(Uhat, j){
        if(j == k)
            UhatEqn += e[i][j][k] * fvm::div(phihat[i], Uhat[j]);
        else
            UhatEqn += e[i][j][k] * fvc::div(phihat[i], Uhat[j]);
    }
}
```

- try out the case (may not work, still),

Concepts covered

- stochastic Galerkin method for Burgers' equation,
- `wmake` build process,
- basic input control,
- difference between `fvc` and `fvm`.
- segregated nature of OpenFOAM solvers.
- selectors and constructors.

Other

- use aliases and environmental variables defined by OpenFOAM,
- OpenFOAM code contains multiple snippets, many of them can be found in

```
${WM_PROJECT_DIR}/src/finiteVolume/lnInclude
```

- use regular expressions in `fvSchemes` and `fvSolution` but remember to escape parenthesis.