FREE CODE
FOR PROPER ORTHOGONAL DECOMPOSITION OF VELOCITY DISTRIBUTIONS

The POD codes for the work described in this paper were developed using commercial software MATLAB®, Ver. 2009b. The code for velocity distributions is provided below. Comments to explain the code are shown in '*italics*'.

Details of the code and how to use POD (also for scalar fields) are provided in the original paper published in the International Journal of Engine Research 2012

A practical guide for using proper orthogonal decomposition in engine research

Hao Chen[1], David L. Reuss[2], David L.S. Hung[3], Volker Sick[2]

[1] Shanghai Jiao Tong University

[2] The University of Michigan

[3] University of Michigan-Shanghai Jiao Tong University Joint Institute

The paper is accepted and will soon be available online.

When using this code, please reference this paper.

Also consider reading the following:

"On the use and interpretation of proper orthogonal decomposition of in-cylinder engine flows" Hao Chen, David L. Reuss, Volker Sick
Measurement Science and Technology, 2012, Vol. 23, 085302
Doi: 10.1088/0957-0233/23/8/085302

Contact: Volker Sick, vsick@umich.edu

```
function VelocityDistributionPOD (SnapshotsAddress)
Method of Snapshots
Section 1 - Input snapshots
Each snapshot (txt file) contains four columns. The first two columns are the velocity
distribution grid point coordinates for x and y direction, respectively. The last two columns
are u and v velocities, respectively.
files = dir([SnapshotsAddress,'*.txt']);
n_snapshots = size(files,1);
for j=1:n_snapshots
   fid = fopen([SnapshotsAddress,files(j).name], 'r');
   data = fscanf(fid,'%f %f %f %f',[4,inf]);
   x = data(1,:);      % x coordinate
   y = data(2,:);      % y coordinate
   U(j,:) = data(3,:);      % u velocity
```

```
   V(j,:) = data(4,:);      % v velocity
   fclose(fid);
end
```

Section 2 - Compute spatial correlation matrix C

```
c1 = U*U';
c2 = V*V';
C = (c1+c2)/n_snapshots;
```

Section 3 - Solve the eigenvalue problem: C * EigenVector = EigenValue * EigenVector

```
[beta, lmd] = svd(C);
```

Section 4 - Calculate basis functions

```
phix = U'*beta;
phiy = V'*beta;
% Normalize basis functions
GridNum = size(x,2);
for j=1:n_snapshots
PhiNor = 0;
   for i=1:GridNum
   PhiNor = PhiNor + phix(i,j)^2 + phiy(i,j)^2;
   end
PhiNor = sqrt(PhiNor);
phix(:,j)= phix(:,j)/PhiNor;
phiy(:,j)= phiy(:,j)/PhiNor;
end
```

Section 5 - Calculate coefficient

```
TimCoeU = U*phix;
TimCoeV = V*phiy;
TimCoe = TimCoeU + TimCoeV;
```

Section 6 - Export basis functions

```
for a=1:n_snapshots
   FilNamPhi = 1000+a;
   PhiOut = fopen([SnapshotsAddress,num2str(FilNamPhi),'.txt']', 'wt');
   fprintf(PhiOut, '#DaVis 7.2.2 2D-vector 16 145 145 "position" "mm" "position" "mm"
"velocity" "m/s"\n');
   phia = [x;y;phix(:,a)';phiy(:,a)'];
   fprintf(PhiOut, '%20.9f %20.9f %20.9f %20.9f\n',phia);
   fclose(PhiOut);
end
% Write coefficients into excel file
xlswrite([SnapshotsAddress,'TimCoe.xlsx'],TimCoe);
```