

POD Analysis of Turbulent Pipe Flow

M. Raba

Created: 2025-09-15 Mon 14:27

1. Code Execution and Layout

1.1. Layout

1. b7.m

2. initSpectral.m

- reads in binary files, takes eg m-fft

3. $\hookrightarrow$ initEigs.m

- forms corrMat, finds eigenvalues

1.2. Layout 2

1. $\hookrightarrow$ initPod.m

- carries out POD calculations (quadrature, multiplication ggf between $\alpha\Phi$) according to Papers (Citriniti George 2000 for Classic POD, Hellstrom Smits 2017 for Snapshot POD)

2. $\hookrightarrow$ timeReconstructFlow.m

- performs 2d reconstruction + plotSkmr (generates 1d radial graph)

1.3. Important Switches

`pipe = Pipe();` creates a Pipe Class. As the functions (above) are called, data is stored in sub-structs:

1. `obj.CaseId` - stores properties like Re , rotation number S , experimental flags such as quadrature (simpson/trapezoidal), number of gridpoints, frequently called vectors (`rMat` $r = 1, \dots, 0.5$)
2. `obj.pod` - eigen data, used for calculating POD
3. `obj.solution` - computed POD modes
4. `obj.plt` - plot configuration

2. Equations Used in Code Procedure

2.1. Classic POD Equations

The following equations are used in the above code.

$$\begin{aligned}\int_{r'} \mathbf{S}(k; m; r, r') \Phi^{(n)}(k; m; r') r' dr' &= \lambda^{(n)}(k; m) \Phi^{(n)}(k; m; r) \\ \mathbf{S}(k; m; r, r') &= \lim_{\tau \rightarrow \infty} \frac{1}{\tau} \int_0^\tau \mathbf{u}(k; m; r, t) \mathbf{u}^*(k; m; r', t) dt \\ \alpha^{(n)}(k; m; t) &= \int_r \mathbf{u}(k; m; r, t) \Phi^{(n)*}(k; m; r) r dr\end{aligned}$$

2.2. Classic POD Equations (Fixed)

$$\begin{aligned} &\int_{r'} \underbrace{r^{1/2} S_{i,j} (r, r'; m; f)}_{W_{i,j}(r,r';m;f)} \underbrace{r^{1/2} \phi_j^{*(n)} (r'; m; f)}_{\hat{\phi}_j^{\psi(i)}(r';m;f)} r^{1/2} \mathrm{d}r' \\ &= \underbrace{\lambda^{(n)}(m, f)}_{\hat{\lambda}^{(n)}(m;f)} \underbrace{r^{1/2} \phi_i^{(n)} (r; m; f)}_{\hat{\phi}_i^{(n)}(r,m;f)} \\ \alpha_n(m; t) &= \int_r \mathbf{u}(m; r, t) r^{1/2} \Phi_n^*(m; r) dr \end{aligned}$$

2.3. Snapshot POD Equations

$$\begin{aligned}
& \lim_{\tau \rightarrow \infty} \frac{1}{\tau} \int_0^\tau \mathbf{u}_T(k; m; r, t) \alpha^{(n)*}(k; m; t) dt \\
&= \Phi_T^{(n)}(k; m; r) \lambda^{(n)}(k; m) \\
\mathbf{R}(k; m; t, t') &= \int_r \mathbf{u}(k; m; r, t) \mathbf{u}^*(k; m; r, t') r \, dr \\
& \lim_{\tau \rightarrow \infty} \frac{1}{\tau} \int_0^\tau \mathbf{u}_T(k; m; r, t) \alpha^{(n)*}(k; m; t) \, dt \\
&= \Phi_T^{(n)}(k; m; r) \lambda^{(n)}(k; m).
\end{aligned}$$

2.4. Reconstruction

The reconstruction is given by

$$q(\xi, t) - \bar{q}(\xi) \approx \sum_{j=1}^r a_j(t) \varphi_j(\xi) \Rightarrow$$
$$q(r, \theta, t; x) = \bar{q}(r, \theta, t; x) + \sum_{n=1} \sum_{m=0} \alpha^{(n)}(m; t) \Phi^{(n)}(r; m; x)$$

Since the snapshot pod implementation is not error-free, the reconstruction can only be recovered by writing for factor $\gg 0$.

$$q(r, \theta, t; x) = \bar{q}(r, \theta, t; x) + (\text{factor } \gamma) \sum_{n=1} \sum_{m=0} \alpha^{(n)}(m; t) \Phi^{(n)}(r; m; x)$$

2.5. Reconstruction

In order to reconstruct in code, `caseId.fluctuation = 'off'`. This is incorrect. The necessary use of (factor γ) is incorrect

3. Derivation

To derive the questioned equation, consider the integral:

$$\frac{1}{\tau} \int_0^\tau \mathbf{u}_T(k; m; r, t) \alpha^{(n)*}(k; m; t) dt.$$

Substitute $\mathbf{u}_T$ with its expansion:

$$\frac{1}{\tau} \int_0^\tau \left(\sum_l \Phi_T^{(l)}(k; m; r) \alpha^{(l)}(k; m; t) \right) \alpha^{(n)*}(k; m; t) dt.$$

3.1. 4 Derivation

Exchange the order of summation and integration, and apply orthogonality,

$$\sum_l \Phi_{\text{T}}^{(l)}(k; m; r) \left(\frac{1}{\tau} \int_0^\tau \alpha^{(l)}(k; m; t) \alpha^{(n)*}(k; m; t) dt \right).$$

Due to the orthogonality, namely that $\alpha^{(n)}$ and $\alpha^{(p)}$ are uncorrelated

$$\langle \alpha^{(n)} \alpha^{(p)} \rangle = \lambda^{(n)} \delta_{np}$$

all terms where $l \neq n$ will vanish, and there remains only the $l = n$ term,

$$\Phi_{\text{T}}^{(n)}(k; m; r) \left(\frac{1}{\tau} \int_0^\tau \alpha^{(n)}(k; m; t) \alpha^{(n)*}(k; m; t) dt \right).$$

This derivation assumes the normalization of modes and their orthogonality, along with the eigenvalue relationship to simplify the original integral into a form that reveals the spatial structure ($\Phi_{\text{T}}^{(n)}$) of each mode scaled by its significance ($\lambda^{(n)}$).

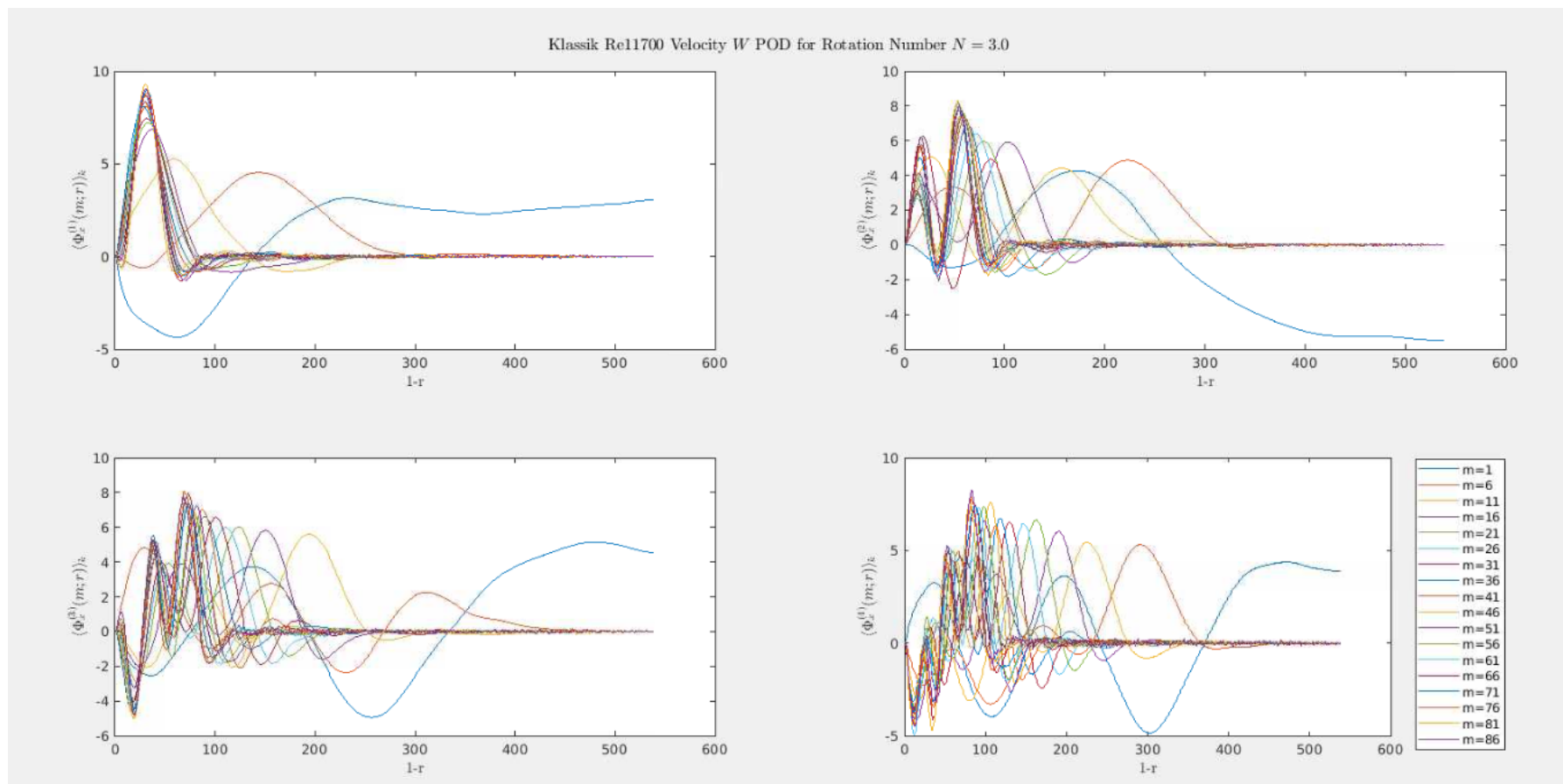
3.2. 6 Derivation

The cross-correlation tensor $\mathbf{R}$ is defined as $\mathbf{R}(k; m; t, t') = \int_r \mathbf{u}(k; m; r, t) \mathbf{u}^*(k; m; r, t') r \, dr$. This tensor is now transformed from $[3r \times 3r']$ to a $[t \times t']$ tensor. The n POD modes are then constructed as,

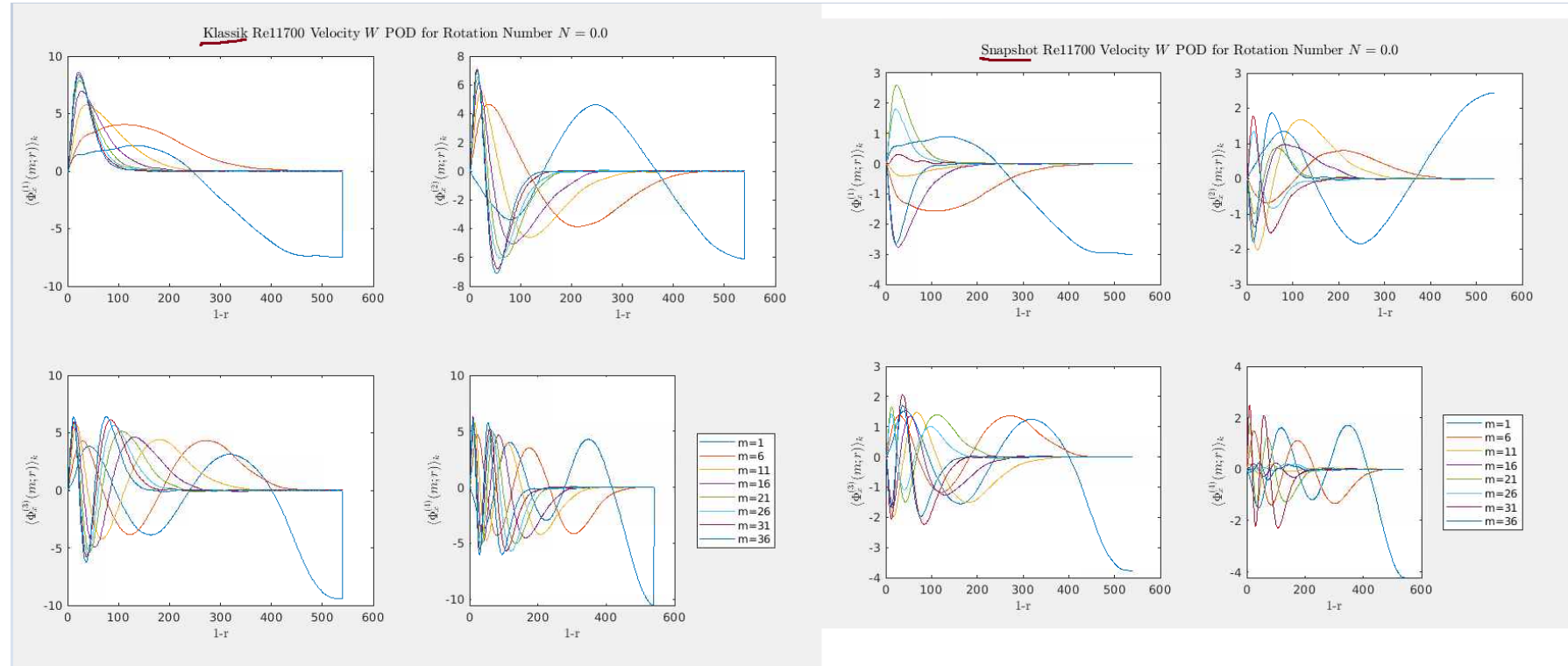
$$\lim_{\tau \rightarrow \infty} \frac{1}{\tau} \int_0^\tau \mathbf{u}_T(k; m; r, t) \alpha^{(n)*}(k; m; t) dt = \Phi_T^{(n)}(k; m; r) \lambda^{(n)}(k; m).$$

4. Result Comparison Classic/Snapshot

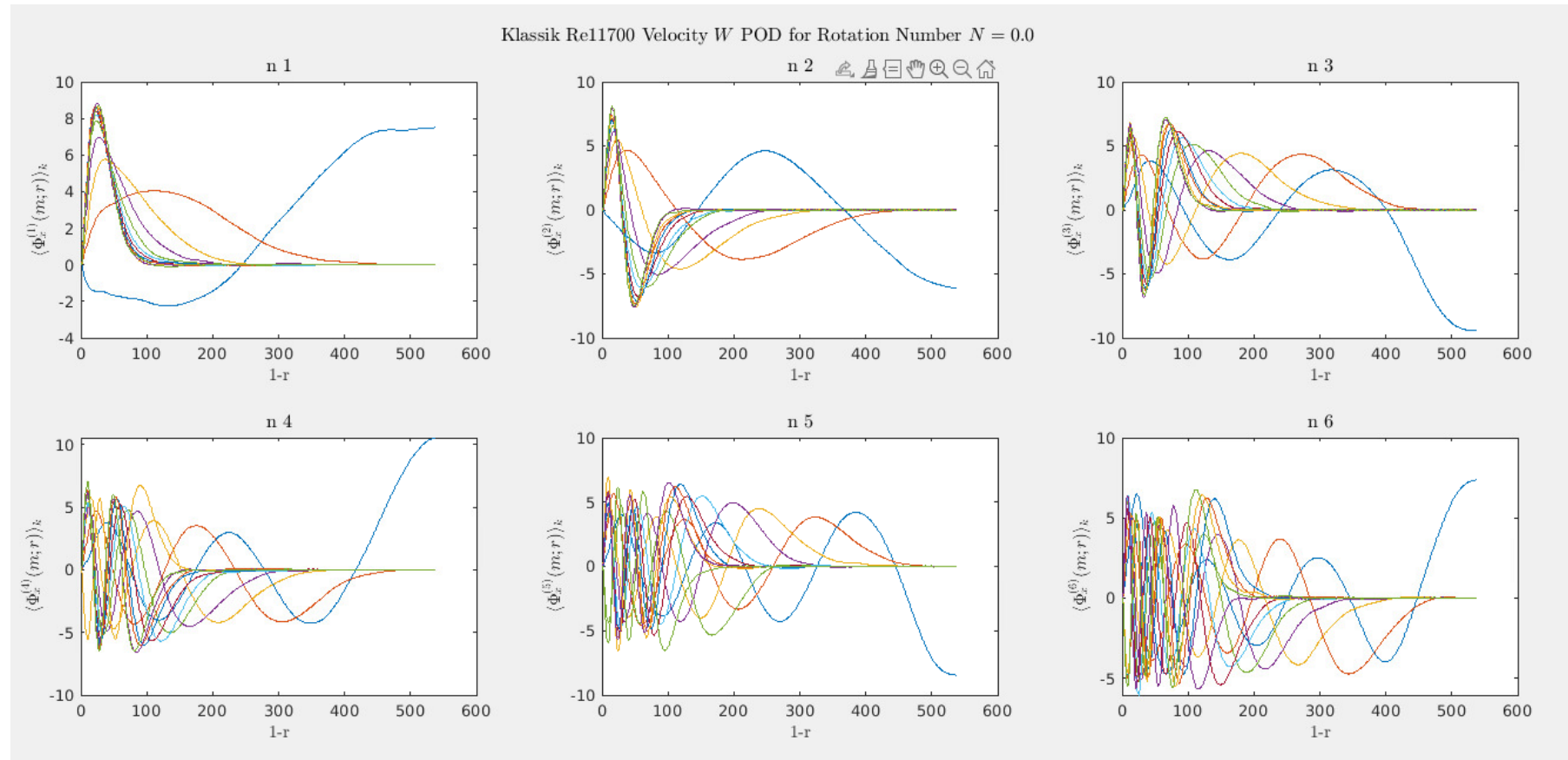
4.1. Radial Classic



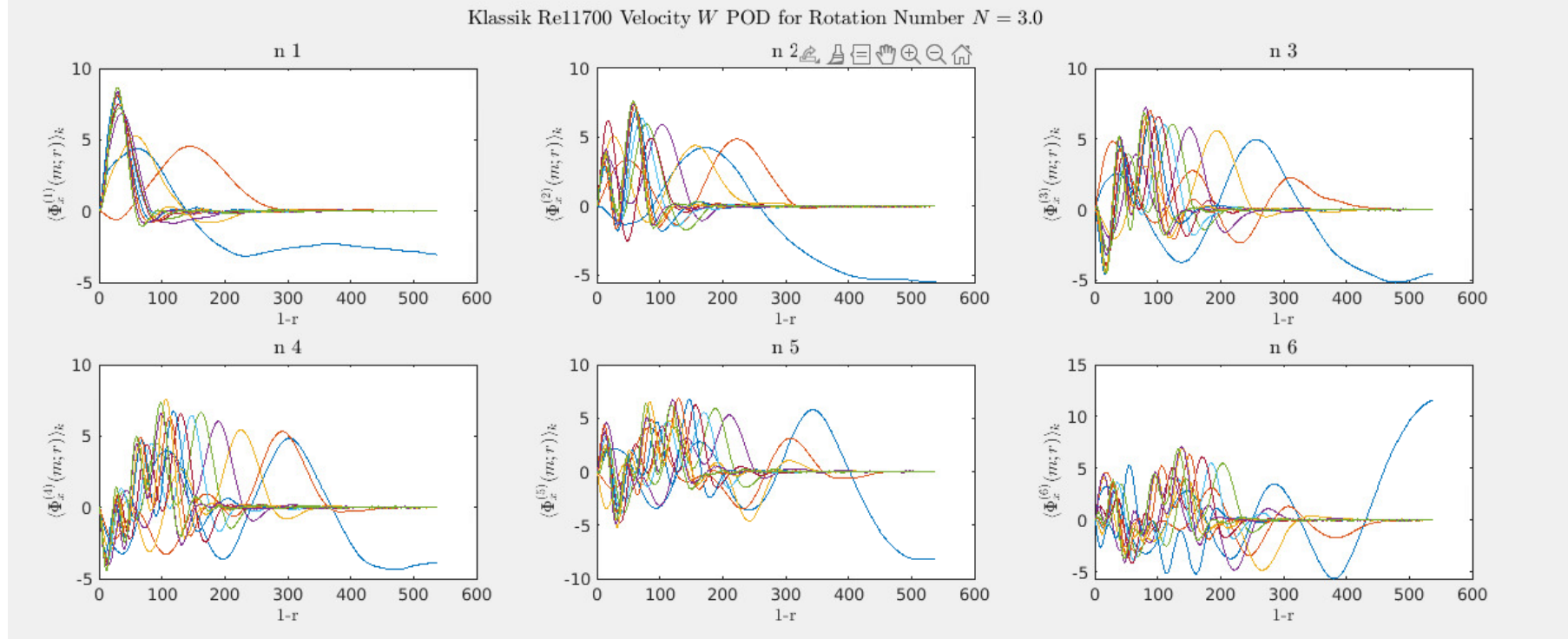
4.2. Snapshot-Classic Comparison



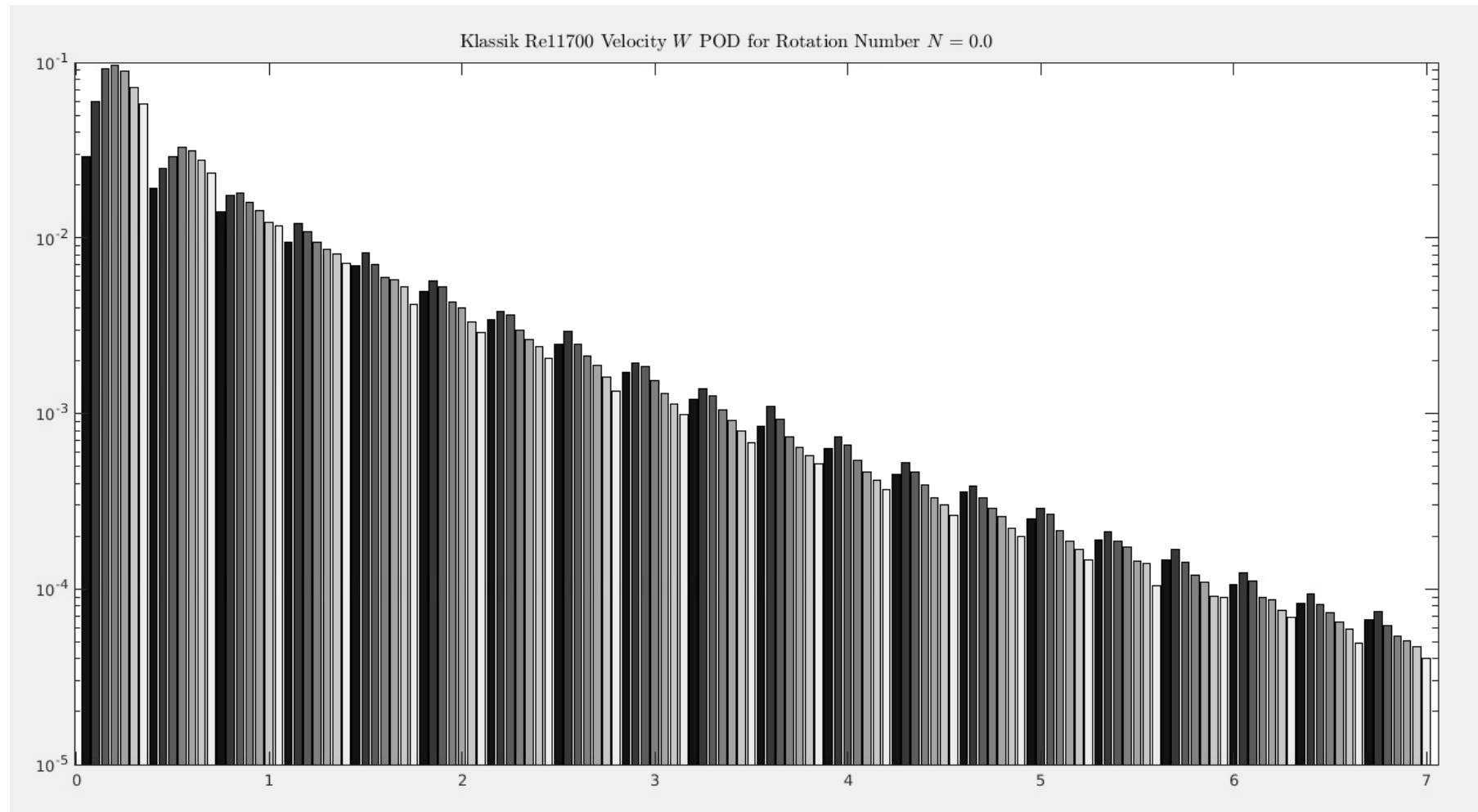
4.3. Klassik POD S=0.0



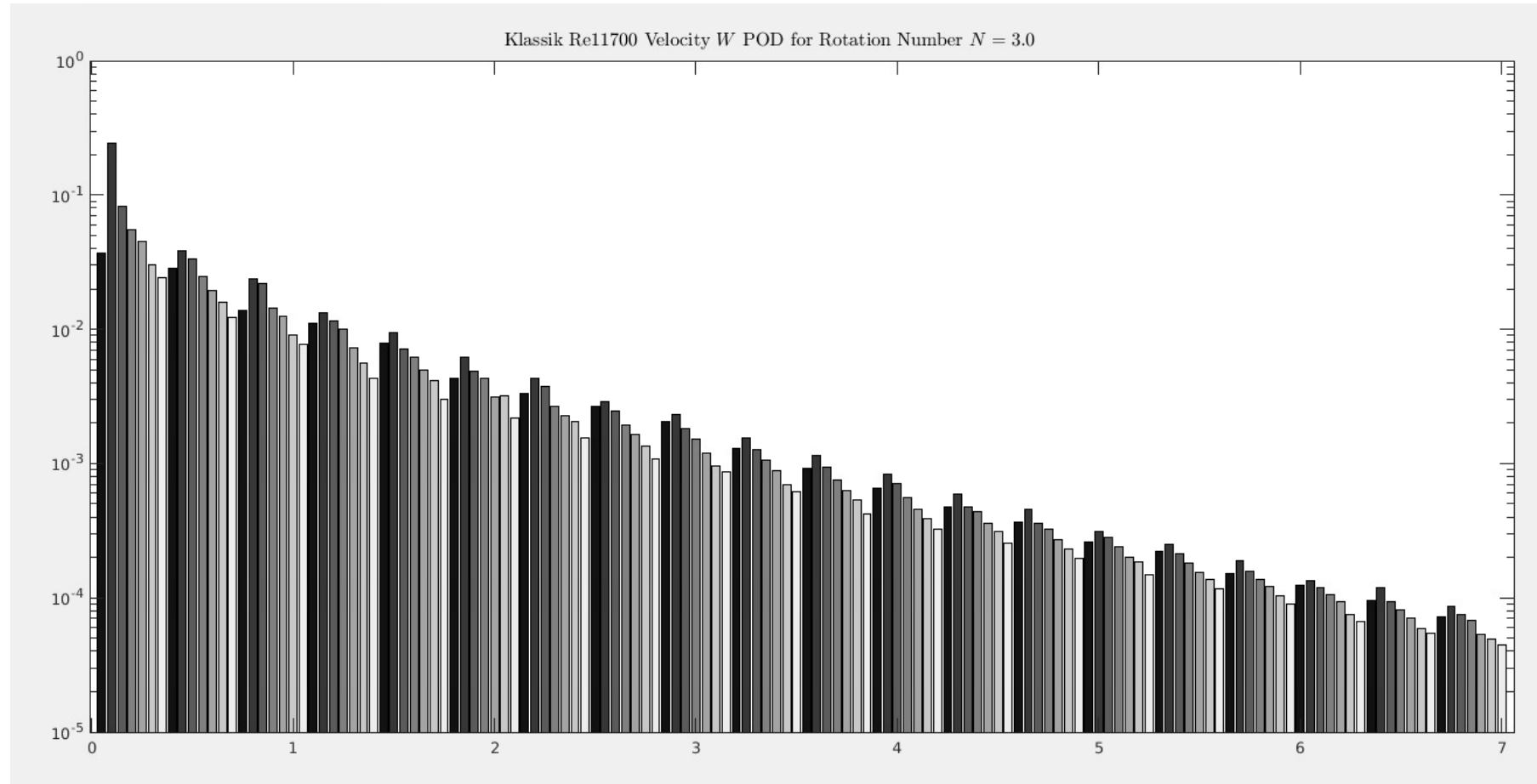
4.4. Klassik POD S=3.0



5. Energy $n=0$ Classic



5.1. $n=3$ Classic



5.2. Analysis

6. Reconstruction

6.1. Reconstruction

