University of Nottingham

Department of Mechanical, Materials and Manufacturing Engineering

MPhil Thesis

Development of the next generation of silent aerospace propulsion configurations

Author:
Mr. Tobias Long

Supervisors:
Prof. M. Icardi
Prof. R. Jefferson-Loveday
Prof. S. Ambrose

Abstract

Distributed electric propulsion (DEP) is a promising development in the aviation industry, with the possibility of advances in efficiency, design and noise-reduction of air transport. This is in part due to the flexibility in distributing aircraft thrust across multiple, electrically-powered propellers across the aircraft. Knowledge of the airflow generated by different distributed propeller layouts is important for analysis and hence the optimisation of propeller layout to minimise the noise generated by these aircraft. In this thesis, a novel data transform originating from the image processing space, the Radon-CDT, is investigated and evaluated as a potential method to be applied in a reduced-order modelling framework for accurate and efficient calculation of flows for DEP configurations.

September 18^{th}, 2023

Contents

1 Introduction

One of the most promising developments in the aviation industry in recent decades is the concept of distributed electric propulsion (DEP). DEP is a specific case of distributed propulsion - where aircraft propulsion is delivered through multiple propellers, distributed upon the aircraft - which utilises electrically-driven propellers, as opposed to propellers driven by internal combustion engines as currently used by the vast majority of the industry (see Figure 1). DEP offers higher potential capabilities in flight efficiency, aircraft control and robustness when compared to combustion engine aircraft [1, 2]. One large benefit of DEP is the flexibility of propeller placement on an aircraft, as propellers and their power sources can be placed in different locations and then connected easily. Although DEP is a promising concept, with small-scale DEP aircraft already in testing, many challenges remain before the concept can offer beneficial results in the real world and be broadly applied to large aircraft. A large part of this challenge arises from the electrical components, although electrically-powered propellers are hugely beneficial in design flexibility and efficiency, battery technology is not sufficiently developed for the energy density and battery life needed to power a realistic DEP aircraft [2, 3].

One area of interest that poses a unique challenge in DEP systems is the noise generated by the propeller layouts, or more broadly, the aeroacoustics associated with DEP aircraft design. The increased flexibility in propeller layout allows for more control of the aeroacoustic profiles generated by different propeller setups, hence minimising both near-field (for any passengers inside the aircraft) and far-

field sound (to the public on the ground) is a design challenge of high importance, considering a recent focus on noise pollution in the aviation industry [4]. Another possible benefit to DEP is that the noise produced by electric propellers is associated with higher frequency sound waves than those produced by internal combustion propellers. This is beneficial for noise reduction because higher frequency sound waves have increased atmospheric absorption, so noise is naturally dampened by the air [5]. The assessment and possible suppression of noise from DEP is the focus of the *SILENTPROP* project [6], part of the *Clean Sky 2* Joint Undertaking of the European Union's *Horizon 2020* innovation programme [7]. The aims of *SILENTPROP* are to develop computational and experimental methods for the assessment of the noise produced by DEP configurations, and the assessment of possible noise suppression technologies to reduce the noise generated by DEP aircraft as much as possible. The work described in this report is part of the numerical branch of the *SILENTPROP* workflow, with the main long-term goal for the PhD project being to develop a reduced-order model (ROM) for accurate and efficient noise assessment

Figure 1: Digital model of the X-57 Maxwell plane in development by NASA (Source: https://sacd.larc.nasa.gov/x57maxwell/).

of DEP configurations.

In this work we employ the Radon-CDT transform to investigate the interpolation of velocity field data generated by computational fluid dynamics (CFD) simulations of various propeller layouts. This interpolation allows the estimation of an unknown velocity field generated by a propeller layout from the velocity fields of previously simulated propeller setups. This will potentially negate the requirement to perform a full, time-consuming CFD simulation for a new propeller layout and replace this with a much shorter transform-based velocity field generation (the order of minutes, rather than hours or days). The Radon-CDT is a novel nonlinear, invertible transform that so far has only been applied to data and image classification problems as an alternative method to neural networks and similar classification techniques [8, 9]. Furthermore, we will expand on the use of this transform by utilising the Radon-CDT transform space for statistical manipulation, in order to obtain improved interpolation of data compared to interpolation in physical space. Initial investigations using the Radon-CDT are outlined in §2, which preceded the work submitted for publication in §3.

In order to reach this goal a large library of detailed and accurate propeller simulations will be required. These will form a basis upon which model parameters and methods can be built, tested and verified. This facilitates a number of detailed simulations for a wide range of parameter variations (for example number of blades, propeller RPM, propeller phase-locking and blade size), which would need a huge amount of computing resources for fully modelled propeller simulations. Instead, we explore an alternative methodology for propeller simulation to be utilised for this

project, allowing a faster computation of sufficiently high-fidelity propeller simulations. This approach is discussed in §4.1.2.

2 Initial Investigations Using The Radon-CDT

2.1 Methodology

2.1.1 Radon-CDT

The Radon-CDT [8] is a combination of the cumulative distribution transform (CDT) and the Radon transform. The Radon transform is used to reduce high-dimensional signals down to a series of 1-D signals, these are then passed into the CDT which interprets these 1-D signals as probability density functions. To avoid repeated definitions the full definition of the transform will not be discussed here, please refer to chapter 2 of the work submitted for publication (§3) for an expanded description.

The Radon-CDT $\hat{f}$ is obtained by applying the CDT along each projection angle in the prior Radon transform, which results in

$$\int_{s_1}^{\hat{f}(s,\theta)} \tilde{f}(s',\theta)\,\mathrm{d}s' = \int_{s_1}^{s} \tilde{r}(s',\theta)\,\mathrm{d}s' \tag{1}$$

for the 2-D case and

$$\int_{s_1}^{\hat{f}(s,\theta,\phi)} \tilde{f}(s',\theta,\phi)\,\mathrm{d}s' = \int_{s_1}^{s} \tilde{r}(s',\theta,\phi)\,\mathrm{d}s' \tag{2}$$

for the 3-D case, where $\tilde{r}$ denotes the Radon transform of $r(x,y;z)$.

2.1.2 Data Synthesis

All fluid dynamics simulations were performed using *OpenFOAM* software (version *v-1812*) [1], running on the on-site university high-performance computer, *Augusta*. Meshes for the simulations were constructed using the *ANSYS ICEM CFD* meshing software.

The propeller flows were estimated using an actuator disk model with radius R and streamwise disk thickness of Δz. The force per unit volume acting on the fluid flow generated by the disc is given by

$$\mathbf{f} = -\frac{1}{\pi R^2 \Delta z^2}\left[\int_\Omega (\mathbf{s}\cdot\mathbf{u})\cdot(\mathbf{u}\cdot\mathbf{s})\,\mathrm{d}\Omega\right]\mathbf{s}, \tag{3}$$

where $\mathbf{s}$ is the streamwise direction unit vector and $\mathbf{u}$ is the velocity of the fluid at the disk. Simulations were carried out using LES with a basic Smagorinsky model, in a cylindrical domain of radius 25 meters and length 150 meters (see Figure 2). For all the results in this work a uniform inlet velocity has been used to reduce computational resources and to save time. Simulations are carried out for 15 seconds, with a timestep size of 0.005 seconds (corresponding to a maximum Courant number of 0.5), with every 20 timesteps being saved to files (increments of $0.1s$). These velocity fields are then averaged over the final 5 seconds of the simulation (adequate time due to uniform inlet velocity, and allowing the wakes to fully develop in the initial 10 seconds of simulation time) and this averaged velocity field is then used to calculate the velocity magnitude at each cell in the domain. Raw velocity data was averaged

[1] Available at https://www.openfoam.com/

and exported to *csv* files using *Paraview* (software included with *OpenFOAM*).

2.1.3 Radon-CDT Workflow

The Radon-CDT methodology and post-processing of data are carried out using *MATLAB*. Data files are read in as tables before the average velocity magnitude is calculated for each cell. These tables are then converted to *MATLAB* arrays, on which the radon transform followed by the CDT can be performed. This is done using the *scatteredInterpolant* function [2] - this samples the velocity magnitude values from the coordinates in the tables and returns an interpolated sample of the data in a 3D array, with a size of $n \times n \times n$, where n is the number of sample points defined by the user. For the results in this report, $n = 250$ is used (unless stated otherwise). This value allows the data arrays to be of high enough resolution to sufficiently represent the flow fields, whilst optimising the runtime and memory requirements. All the data sets in this work are defined within a rectangular box along the length of the full simulated domain, centred on the propellers. This box is defined between ± 3 in the x and y directions, and between 47.5 and 72.5 in the z-direction (see Figure 2). Example velocity contours from the data can be seen in Figure 3.

Once the data sets have been generated and converted to arrays, they are normalised (dividing all array values through by the magnitude of the maximum array value) before being transformed by the Radon-CDT method. For all results in this report, the step size of the θ array used in the Radon transform was set as $\Delta\theta = 1$. This was done to optimise result accuracy and code runtime. The Radon-CDT trans-

[2] Documentation: https://uk.mathworks.com/help/matlab/ref/scatteredinterpolant.html

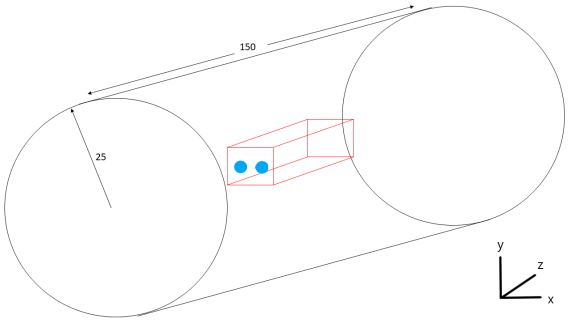


Figure 2: An illustration of the full CFD domain (outer cylinder) and the box (shown in red) containing the propellers (blue) defining the data sets in this work.

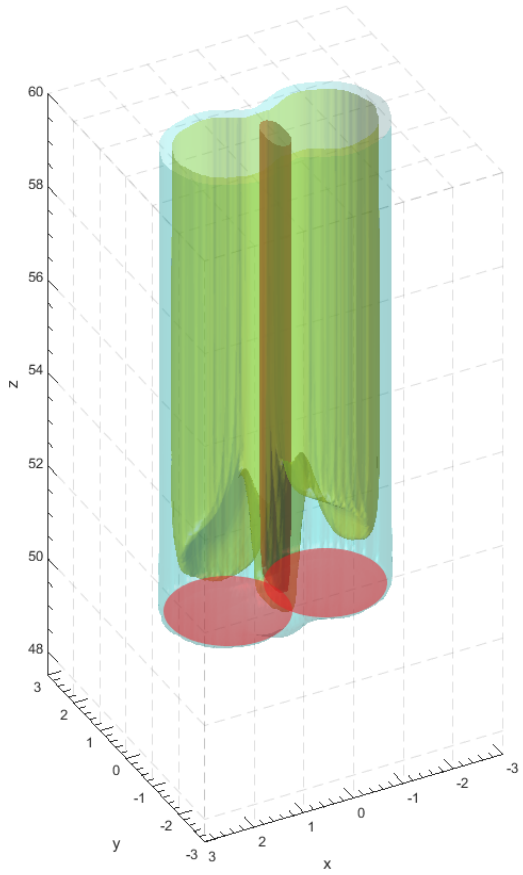


Figure 3: Contours of velocity magnitude in an example data set. Contours correspond to values of 10 (cyan), 12.5 (yellow) and 15 (orange). Actuator disk modelling propellers are shown by the red disks.

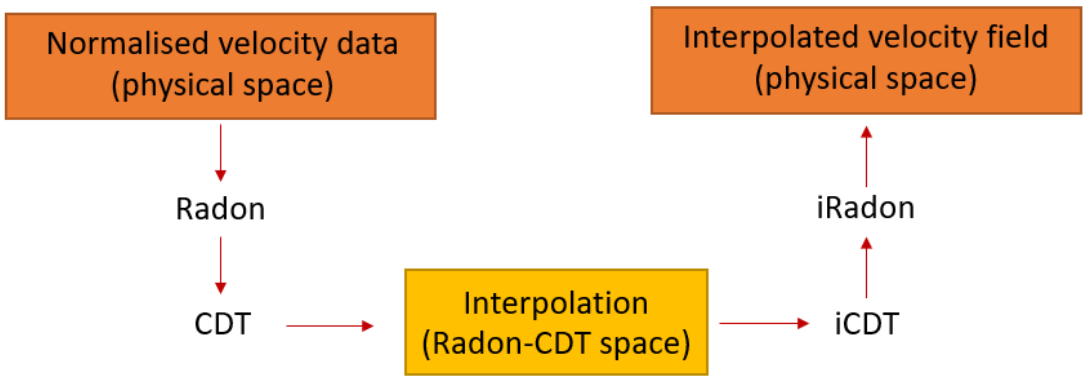


Figure 4: Workflow for results in this article.

form is not applied to the 3D data as a whole but instead is applied individually to 2D slices of the 3D data set. For example, when the data is sampled from the data set by the *scatteredInterpolant* function using 250 sample points in each dimension, our Radon-CDT algorithm is applied individually to the 250 xy, yz, or xz slices in this sampled data. A full 3D algorithm is in development, though is currently not ready for use. Interpolation is then carried out by averaging the values of the arrays in Radon-CDT space, before the application of inverse CDT and radon transforms to return the array values to physical space. The values in these arrays are then scaled back to the original range of data values using a weighting in accordance with the interpolation parameter, α (see §3.6). The procedure for interpolating the data can be seen in Fig. 4.

2.1.4 Interpolation Methods

Interpolation of data sets has been carried out using arithmetic averaging and geometric averaging. For arithmetic averaging, we interpolate N data sets of velocity magnitudes v_i to obtain the interpolated velocity magnitude v_{int} using

$$v_{int} = \sum_{i=0}^{N} \alpha_i v_i \ , \quad \text{where} \quad \sum_{i=0}^{N} \alpha_i = 1. \tag{4}$$

Similarly, when using geometric averaging, the interpolated velocity magnitudes are given by

$$v_{int} = \prod_{i=0}^{N} \alpha_i v_i \ , \quad \text{where} \quad \prod_{i=0}^{N} \alpha_i = 1. \tag{5}$$

2.1.5 Error Quantification

To quantify the accuracy of the interpolation in each case, a squared error parameter Δ will be calculated for each cell within the domain (of total volume V) using the formula

$$\Delta = \frac{\int_V (v_{int} - v_{targ})^2 \, dV}{\int_V v_{targ}^2 \, dV}, \tag{6}$$

where v_{targ} denotes the velocity magnitude values in the target data set. In the case of a uniformly distributed domain, this effectively reduces to the square of the error in a given cell divided by the target value for that cell. The average percentage error for the resulting array will also be used, as this is a good metric for intuitive analysis of the accuracy of the result. The percentage error from the target will be shown for each cell in the resulting array, as well as an overall average percentage error for the full domain for quick comparisons between results. To calculate this value, after the percentage difference from the target array value is calculated for every cell in the result array, the percentages are averaged for the full data set to

return a single percentage value for the error.

2.2 Investigation Results

Firstly the Radon-CDT framework was used to interpolate velocity field data sets with two propellers of identical radius, with differing separations between the propellers. Similarly, the framework is then used to carry out similar interpolation on varying propeller radii, with constant separation between propellers. Finally, the Radon-CDT is used to estimate flows where the separation of propellers and the size of the propellers both vary between data sets. An example of separation, radius and combined interpolation between two propeller data sets can be seen in Figures 5 and 6, Figures 7 and 8, and Figures 9 and 10 respectively. Note that when referring to the domain, this is the box defined previously in §3.5 (see Figure 2). In the results, we will use the notation $[a_1, a_2, a_3]$ for demonstrating the separations between propellers, or radii of propellers used in the interpolation (all values given in meters). a_1 and a_3 are the parameters of the two data sets used for the input of the interpolation method, with a_2 being the interpolated (and hence target) parameter. Whether these parameters refer to separation or radius will be made clear.

2.2.1 Separation Interpolation

Interpolation between data sets where identical propellers are separated by differing distances was carried out in three different cases, the results of which are seen in Table 1. For the cases with propeller radius 0.50, the error for the case with propellers closer together has a lower value, signifying this interpolation result is more accurate

Propeller Radius	Propeller Separations	Arithmetic Error	Geometric Error
0.50	[1.00, 1.25, 1.50]	4.81×10^{-3} / 1.83%	4.81×10^{-3} / 1.83%
	[1.00, 1.50, 2.00]	2.28×10^{-2} / 6.26%	2.26×10^{-2} / 6.27%
0.75	[1.00, 1.50, 2.00]	6.70×10^{-3} / 4.40%	6.70×10^{-3} / 3.39%

Table 1: Error values for separation interpolation between propeller data sets of constant radius. Both arithmetic and geometric averaging interpolation with equal weighting ($\alpha_i = 0.50$) have been used and the errors for each method are shown. The error values shown are: integrated error values / average percentage errors over the whole domain (both to 3 s.f.).

than the case with propellers further apart. Comparing this with the case with the same separations but a radius of 0.75, we note that the case with a large propeller radius has a lower error. Finally, not just for separation interpolation, but indeed for all three interpolation tests in this report, there is very little difference in the error of the result when using arithmetic and geometric averaging.

2.2.2 Radius Interpolation

The results for interpolation using a constant propeller separation and differing propeller radius can be seen in Table 2. For each constant separation, the cases with the largest propeller radii had the lowest error, however as seen in the separation 2.00 cases, there is no trend between error and propeller radius.

2.2.3 Radius-Separation Interpolation

The results for interpolation involving varying separation and radius are found in Table 3. For this combined interpolation, there is a small difference in the errors favouring the geometric method. This differs by a small amount when comparing this

Propeller Separation	Propeller Radii	Arithmetic Error	Geometric Error
1.50	[0.50, 0.75, 1.00]	9.68×10^{-3} / 4.95%	9.65×10^{-3} / 4.94%
	[0.75, 1.00, 1.50]	5.69×10^{-3} / 3.73%	5.65×10^{-3} / 3.73%
2.00	[0.50, 0.75, 1.00]	6.21×10^{-3} / 5.17%	6.20×10^{-3} / 5.17%
	[0.75, 1.00, 1.25]	2.59×10^{-3} / 2.42%	2.57×10^{-3} / 2.41%
	[1.00, 1.25, 1.50]	3.04×10^{-3} / 3.42%	3.03×10^{-3} / 3.42%
	[1.25, 1.50, 1.75]	6.96×10^{-4} / 1.53%	6.89×10^{-3} / 1.52%

Table 2: Error values for radius interpolation between propeller data sets of constant separation. Both arithmetic and geometric averaging interpolation with equal weighting ($\alpha_i = 0.50$) have been used and the errors for each method are shown. The error values shown are: integrated error values / average percentage error over the whole domain (both to 3 s.f.).

to the previous results for separation and radius interpolation alone. This difference in error is large enough to be considered as significant.

2.2.4 Interpolation Parameter

Parameter sweeps for the interpolation weighting parameter, α were performed to obtain an estimate for the optimal weighting in each of the cases. An example of this for the two-propeller interpolation cases can be seen in Figure 11. From these plots we can infer that there is little difference in the results for two-propeller data sets when using arithmetic or geometric averaging for the interpolation, at all values of α. The optimal value of α varies slightly, from $\alpha = 0.5$ (equal weighting) for separation interpolation, to $\alpha = 0.40 - 0.45$ and $\alpha = 0.35 - 0.40$, for radius and combined radius-separation interpolation respectively.

Propeller Separations	Propeller Radii	Arithmetic Error	Geometric Error
$[1.00, 2.00, 3.00]$	[0.50, 0.75, 1.00]	2.72×10^{-2} / 9.46%	2.72×10^{-2} / 9.46%
	[0.75, 1.25, 1.75]	5.41×10^{-3} / 5.10%	5.34×10^{-3} / 5.04%
	[0.50, 1.25, 2.00]	2.77×10^{-2} / 9.52%	2.75×10^{-2} / 9.38%
$[1.00, 1.50, 2.00]$	[0.50, 0.75, 1.00]	1.44×10^{-2} / 7.25%	1.43×10^{-2} / 7.23%
	[0.75, 1.00, 1.25]	4.47×10^{-3} / 4.02%	4.42×10^{-3} / 3.98%

Table 3: Error values for radius-separation interpolation between propeller data sets. Both arithmetic and geometric averaging interpolation with equal weighting ($\alpha_i = 0.50$) have been used and the errors for each method are shown. The error values shown are: integrated error value / average percentage error over the whole domain (both to 3 s.f.).

2.3 Brief Discussion of Results

Visually, the results of the Radon-CDT interpolation produce a very similar profile to the target image for separation interpolation (see Figure 5 - 6) and radius interpolation (Figure 7 - 8). For separation interpolation, the correct shape for the propeller layout and resulting wake is visible, as is demonstrated by the low error values in the wake and around the propellers (Figure 5/6c). A high amount of error (approximately 120%) can be seen in a small region between the propellers, although this is an area of relatively low velocity compared to the wake values and so the absolute difference in values here is not the largest in the domain (see Figure 5/6d). Radius interpolation delivers similar results around the propellers, where there is a larger peak area of error (see Figure 78c), however, the percentage error does not peak at such a high value as the previous case (peaking at around 40% for this case). The error in the wake for radius interpolation is similar to the error seen for separation interpolation - the percentage error is found to be around 10-40% throughout the

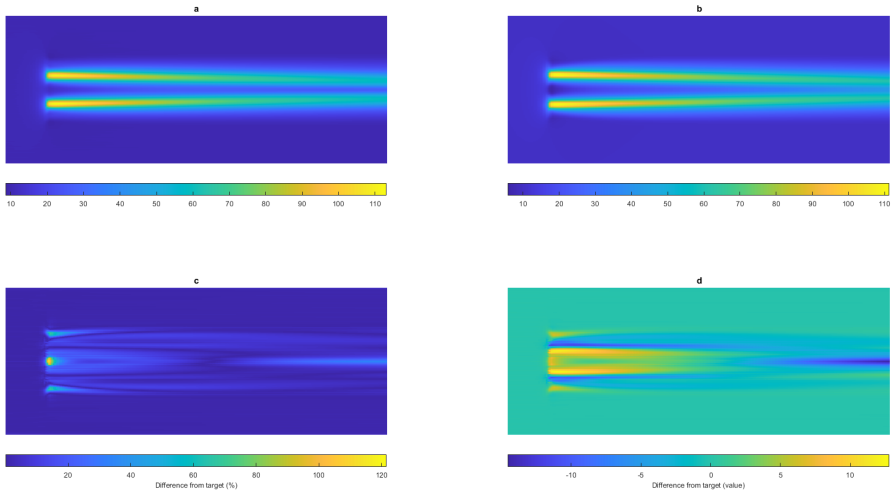


Figure 5: Example xz-plane slice of velocity magnitude through the centre of the domain for separation interpolation [1.00, 1.25, 1.50]. (a) Radon-CDT interpolated slice, (b) Target slice, (c) Percentage error between each cell of (a) and (b), (d) Absolute value difference between each cell of (a) and (b). Propeller radius = 0.50, $\alpha = [0.50, 0.50]$

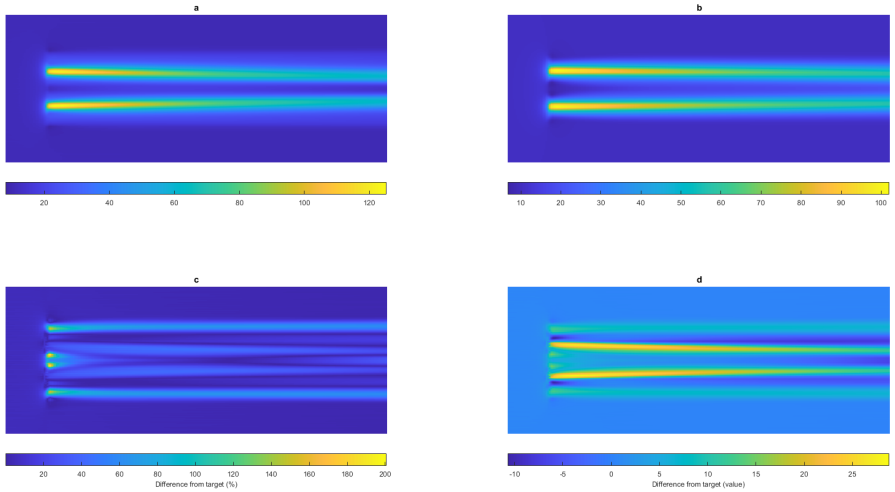


Figure 6: Example xz-plane slice of velocity magnitude through the centre of the domain for separation interpolation [1.00, 1.50, 2.00]. (a) Radon-CDT interpolated slice, (b) Target slice, (c) Percentage error between each cell of (a) and (b), (d) Absolute value difference between each cell of (a) and (b). Propeller radius = 0.50, $\alpha = [0.50, 0.50]$

wake - although it is distributed differently in each case. For both interpolation methods, the shape of the flows is visually representative of the target but differs with numerical value. This leads us to believe that the error in the numerical values for each cell arises from the scaling applied after the Radon-CDT method, to return the velocity values to their original ranges, rather than from the Radon-CDT method itself, which does not depend on the magnitude of array values but the relative difference between these values. Therefore, focusing on how to better average these magnitudes for scaling may prove beneficial to the accuracy of the results in the future.

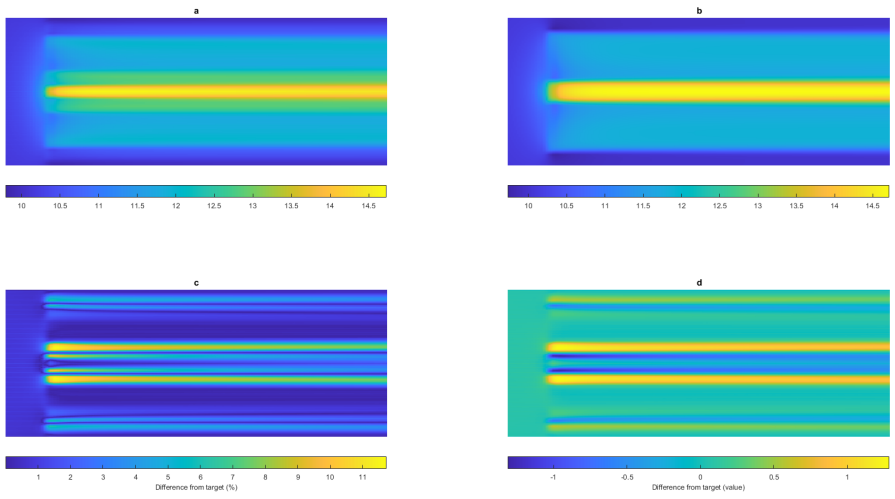


Figure 7: Example xz-plane slice of velocity magnitude through the centre of the domain for radius interpolation [1.25, 1.50, 1.75]. (a) Radon-CDT interpolated slice, (b) Target slice, (c) Percentage error between each cell of (a) and (b), (d) Absolute value difference between each cell of (a) and (b). Propeller separation = 2.00, α = [0.500.50]

For the combined interpolation case, the shape of the propellers and resulting wake are not accurately estimated by the Radon-CDT method for the cases where the propeller wakes do not overlap (Figure 10). The result is more accurate visually for

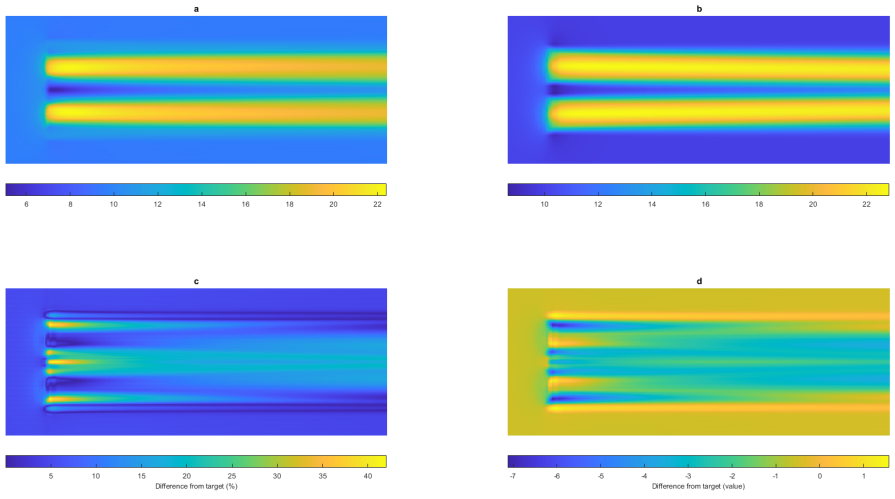


Figure 8: Example xz-plane slice of velocity magnitude through the centre of the domain for radius interpolation [0.50, 0.75, 1.00]. (a) Radon-CDT interpolated slice, (b) Target slice, (c) Percentage error between each cell of (a) and (b), (d) Absolute value difference between each cell of (a) and (b). Propeller separation = 2.00, $\alpha = [0.50, 0.50]$

the overlapping case (Figure 9), but is still not fully capturing the correct morphology of the target flow. However, in terms of error values, we should note that there appears to be no trend between whether the wakes overlap and the numerical error, this is also the case with radius interpolation. This could also be investigated for separation interpolation, but as of yet this has not been tested. It is particularly clear for combined interpolation that the estimated array struggles to capture the combined shape of the input arrays. In Figures 9a and 10a there are clearly areas surrounding the 'strong' wake where the original input data are still visible - the method has failed to completely combine the inputs into the correct size 'strong' wake seen in the target images. This is most obvious in the case where the wakes of the target are not overlapping (Figure 10). This is unlikely down to the interpolation weighting, as shifting the weighting towards the smaller propeller input will remove

this larger propeller 'shadow', but further reinforce the wakes that are already too small when compared to the target wakes. We can also see from Figure 11 that the interpolation parameter does not significantly reduce the error from the case with $\alpha = 0.5$ shown, and in fact, increases when the interpolation is weighted more heavily in favour of the smaller propeller data set (larger α in the plots). A more thorough investigation of the Radon-CDT interpolation is needed to overcome this, which will be the focus of future work. It is also possible that using reduced-order modelling will eliminate this problem of diffuse values, through having less complicated morphology in the arrays.

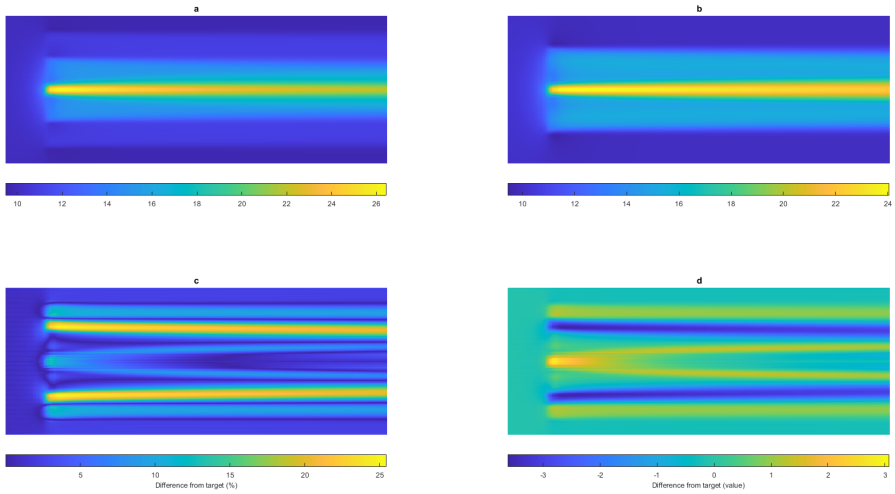


Figure 9: Example xz-plane slice of velocity magnitude through the centre of the domain for combined separation-radius interpolation with data sets of propeller separation/radius [1.00/0.75, 1.50/1.00, 2.00/1.25]. (a) Radon-CDT interpolated slice, (b) Target slice, (c) Percentage error between each cell of (a) and (b), (d) Absolute value difference between each cell of (a) and (b). $\alpha = [0.50, 0.50]$

The initial results presented here are promising and confirm the further investigation of applying the novel Radon-CDT transform to data interpolation for distributed propeller layouts is worthwhile. Preliminary tests using 2D Radon-CDT on data sets

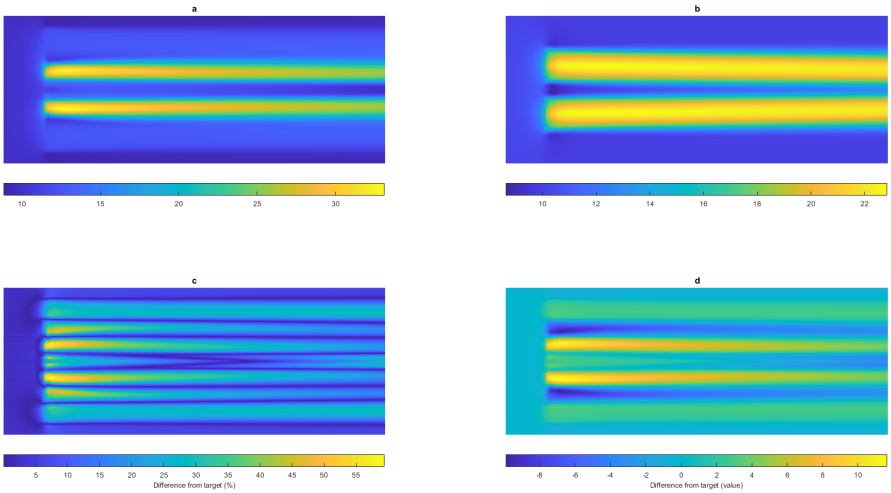
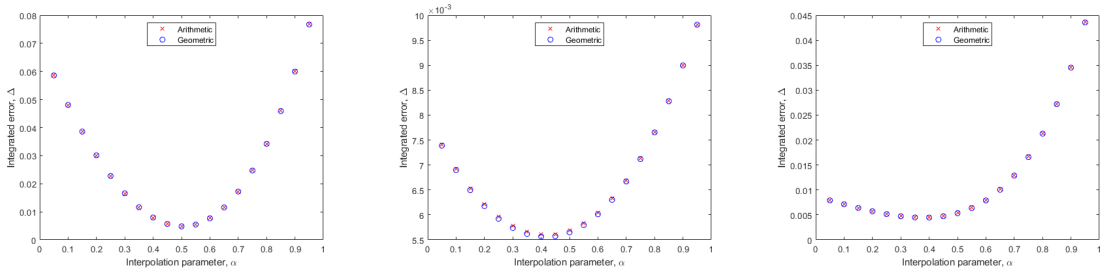


Figure 10: Example xz-plane slice of velocity magnitude through the centre of the domain for combined separation-radius interpolation with data sets of propeller separation/radius [1.00/0.50, 2.00/0.75, 3.00/1.00]. (a) Radon-CDT interpolated slice, (b) Target slice, (c) Percentage error between each cell of (a) and (b), (d) Absolute value difference between each cell of (a) and (b). $\alpha = [0.50, 0.50]$



(a) (b) (c)

Figure 11: Integrated error in the velocity magnitude Δ for interpolation of two-propeller data sets, both arithmetic and geometric averaging, over a range of interpolation parameter α values for the first input data set (the second input is weighted by $1 - \alpha$). (a) Propeller separation interpolation, separation distances [1.00, 1.25, 1.50], for a propeller radius of 0.50. (b) Propeller radius interpolation, propeller radii [0.75, 1.00, 1.25], for a propeller separation of 1.50. (c) Combined radius-separation interpolation, separations [1.00, 2.00, 3.00], radii [0.75, 1.25, 1.75] respectively.

have demonstrated that the Radon-CDT space manipulation is a useful tool for interpolation of separated-propeller data sets, allowing the merging of flows without duplication of features, in contrast to physical space and other transform spaces. Results from the interpolation of raw data are promising for future work where these methods will be applied to reduced-order models, in which the features of each data set will be of lower complexity, with a strong possibility of producing more accurate results.

3 Published Work

The following section contains the work submitted for publication. The paper appears here in the submitted format with no changes, as found in the public domain [10].

A novel reduced-order model for advection-dominated problems based on Radon-Cumulative-Distribution Transform

Tobias Long*[1], Robert Barnett†[1], Richard Jefferson-Loveday‡[2], Giovanni Stabile§[3], and Matteo Icardi¶[1]

[1]School of Mathematical Sciences, University of Nottingham, NG7 2RD, Nottingham, UK
[2]School of Engineering, University of Nottingham, NG7 2RD, Nottingham, UK
[3]Department of Pure and Applied Sciences, Informatics and Mathematics Section, University of Urbino Carlo Bo, Piazza della Repubblica, 13, I-61029 Urbino, Italy

Abstract

Problems with dominant advection, discontinuities, travelling features, or shape variations are widespread in computational mechanics. However, classical linear model reduction and interpolation methods typically fail to reproduce even relatively small parameter variations, making the reduced models inefficient and inaccurate. In this work a novel reduced order modelling approach is proposed based on the Radon-Cumulative-Distribution transform (RCDT). We show that this non-linear transformation can significantly improve the dimensionality of proper orthogonal decomposition (POD) reconstructions and is capable of interpolating accurately some advection-dominated phenomena. The method is tested on various testcases in multiphase fluid dynamics.

Keywords

Reduced Order Modelling; Non-linear transformations, advection-dominated problems, Radon transform, Cumulative Distribution, Proper Orthogonal Decomposition

*Tobias.Long@nottingham.ac.uk
†Robert.Barnett@nottingham.ac.uk
‡Richard.Jefferson-Loveday@nottingham.ac.uk
§giovanni.stabile@uniurb.it
¶Matteo.Icardi@nottingham.ac.uk

1 Introduction

The importance of modelling advective-dominant flows accurately, and efficiently, is incorporated within a multitude of scientific fields of study. Examples include the aviation industry, both traditional combustion and newly emerging distributed electric propulsion (DEP) for electrical aircraft [1, 2, 3]. Using a configuration of electric propellers to provide propulsion to the aircraft, as opposed to the typical combustion propellers in use today, allowing more flexibility in aircraft design in comparison. In both cases highly turbulent, or advective-dominant, flows are generated by said propellers; therefore, requiring appropriate modelling. Of particular importance to aviation, and especially DEP, has recently been noise pollution [4, 5]. With aims to reduce the aeroacoustics generated by said propellers; both near-field where passengers are located, and far-field such as public on the ground. Again, this requires appropriate models of the aircraft's flow to decipher the sound frequencies produced. Other examples of advective-dominant flow presence are the accretion disks of energy formed by black holes and certain stars [6, 7]; aerodynamically generated noise, i.e. aeraucostics, of helicopter rotors [8]; similarly, the near-field aerodynamics of wind turbines, close to their blades, within wind energy [9]; and the field of turbulent flows, advective-dominance being a key contributing factor [10, 11, 12].

The common interest in all these problems is the modelling of highly complex non-linear advective-dominant flows, with aims at being able to predict a variety of flows; considering variations in the initial, or dynamic, geometrical parameters of the given flow. However, due to the severe non-linearity of the problem, this can be a demanding task to accomplish. Since flow variations cannot be linearly separated, predicting 'interim' variations – those between two solved variational flows – via physical interpolation of the solved flows often yields unsatisfactory results. What's more the high-dimensionality poses a demanding computational task, even when attempting other forms of predictive interpolation, forming significantly large data arrays to handle. As such, results of solving the flows in their full-dimensional form are often slow and computationally inefficient. An established methodology to combat this computational issue, producing order-reduced flows of lower dimension, is reduced order modelling (ROM) [13, 14], or, model order reduction.

ROM is widely-used in computational science and engineering applications. It plays a key role in applications that require simulating systems for many scenarios with different parameters efficiently. Example applications are system control [15], uncertainty quantification [16] and optimal system design [17]. With reduced order modelling of a system, the set of differential equations – which describe the physical system we are interested in – are solved numerically in low-dimensional reduced spaces. This is in contrast to full order models that are formulated in the full high-dimensional space, often solved with finite element or volume methods. The reduced space is constructed from data in an 'offline' phase and is then applied in an 'online' phase, returning an approximate solution typically in a far shorter time than solving for the full, non-reduced, model. This greatly increases the speed at which many different parameters can be tested and results returned. However for advection-dominated phenomena, a keen concern in the reduced order modelling community of fluid dynamics, an issue arises. This is because breaking the Kolmogorov n-width barrier in problems that are difficult to compress by linear reduced order modelling strategies remains a particularly difficult problem to solve [18, 19]. Advection-dominated problems exhibit a slow decay in Kolmogorov n-width, thus rendering the reduced order model (ROM) construction inefficient [20]. Current approaches to overcome this problem include the use of nonlinear compression strategies (eg. autoencoders, convolutional autoencoders) [21, 22], piecewise linear subspaces (involving clustering into the parameter space or solution space) [23] and nonlinear transformation of the solution manifold in order to then apply standard compression strategies. This last approach can further be divided into a few distinct approaches where the transformation from the physical problem is known [24], where the transformation is computed by solving an optimal transport problem [25] and finally where the transformation is learnt with a neural network [26].

Instead, our approach turns towards the conjunction of two transformations: the Radon transform, and the recent cumulative distribution transform (CDT) [27]. Transforming high-dimensional 'signals' e.g. a travelling wave, to a series of one-dimensional ones; finally represented as one-dimensional probability density functions in the new RCDT space. Through the conjunction of these two transforms, particularly CDT due to its non-linear form, flows become linearly separable in RCDT space. Enabling typical ROM methods, such as proper orthogonal decomposition (POD), to be used in the transform space; therefore, forming a reduced-order transform space which can be inverted. Finally arriving at a reduced-order, i.e. lower dimensional, space of the original. With better computation efficiency over the full-order model when solving or predicting flows. Furthermore, since flows become separable in the RCDT space, interpolation can be done through standard approaches, such as linear interpolation, to arrive at predicted

flows when done within the RCDT space.

In this work we utilise the unique properties of the RCDT to capture geometric and spatial variations within a parameterised input and use this to produce an approximate solution for system parameters in a reduced order modelling methodology. Initially, we investigate the properties of the RCDT with simplified test cases to gauge the strengths and weaknesses of the methodology for potential use in the ROM and CFD communities. Later the RCDT is applied to a ROM workflow, following proper orthogonal decomposition (POD), and later tested on a number of computational fluid dynamics (CFD) data sets. Verifying flow retention when transformed between spaces, alongside the accuracy of ROM's flow reconstruction at reduced order. We also give a preliminary study into the interpolation error introduced in predicting flows; giving an initial qualitative gauge on RCDTs' applicability to flow prediction via interpolating known, i.e. solved, flows.

Both the implementation of the RCDT and ROM workflows have been written in *Python 3.9.7*. Making use of two packages, *PyTransKit* [28] and *EZyRB* [29]; implementing the discretised form of RCDT – with subsequent forward/inverse transforms – and reduce order modelling (ROM) functionality, respectively. For the ROM side, i.e. *EZyRB*, model reduction is approached using proper orthogonal decomposition (POD), see [30, 31], for example, in applying *EZyRB* towards shape optimization problems. Specifically the singular value decomposition (SVD) – discussed more in §2.4 – is used, as a way to determine the modes of POD for the order-reduced model. SVD is not the only way to compute the POD, though, an alternative approach is given by the method of snapshots [10, 32]. Three distinct workflows have been implemented: performing RCDT upon a single snapshot image/flow, transforming then subsequently inverting the image to observe intrinsic error induced by the non-linear transform; the RCDT-POD reconstruction/projection error to evaluate the effect of the non-linear transformation in the POD models; and, the complete RCDT-POD ROM workflow on a series of snapshots of a flow at separate time points.

To analyse RCDT, POD, and RCDT-POD ROM's applicability towards modelling advective-dominant flows we consider the relative errors introduced by these transformations. Performing a number of test cases to observe each of their magnitudes. These errors are: the intrinsic error given off by RCDT, due to being a non-linear transformation; the reconstruction/projection error of RCDT-POD, influenced by the number of POD modes, when reconstructing the flow after order-reduction; and RCDT-POD ROM interpolation error, introduced from predicting between "known" flow configurations to gauge the capability of predicting, via interpolation in their respective space, along a given parameter e.g., time, distance, 'thickness', etc.

To observe and test the respective errors summarised above we perform a multitude of image and flow case studies. These can be found in §3.2 through 4.2.1. In chronological order of appearance these are: fig. 3 of a unit circle image, with circle and background defined as one and zero or vice versa, to observe intrinsic error in the RCDT workflow; fig. 4, a circular ring with sharp and thin width, to observe the same error as above; fig. 5, likewise looking at intrinsic error, but instead for a normal Gaussian distributions and its 'inverted' twin, construable as an extreme case of a smoothed boundary and subsequent observance in potential error-reduction; table 1 is a more quantitative measurement of the error-reduction from smoothed boundaries, given as the average relative L_2-norm calculated in physical space for both smoothed and sharp cases; figs. 6 and 7 contains, respectively, a pair of input twin jet flow images in differing separation widths and their interpolation in RCDT space compared to the physical, with the target jet 'configuration' alongside both image pairs, and intention as an initial test into the interpolation error introduced when 'predicting' flows whilst in RCDT space; fig. 8 is an inputted random-travelling Gaussian distribution, computed into RCDT space and order-reduced using POD's five highest value 'modes', compared alongside 'standard' POD in physical space, where the difference is taken against the input to compare reconstructive error of both, while fig. 9 provides insight into the efficiency of decomposition of POD in RCDT space, compared against Fourier and in the physical; in figs. 11 and 12 we input a multi-phase wave – using now high-resolution computational fluid dynamic (CFD) data – into the same RCDT-POD workflow as before, done twice for the five and twenty highest 'modes' respectively, to determine if mitigation of reconstructive error by considering more 'modes' is still retained in RCDT space, with fig. 13 comparing RCDT-POD against physical and Fourier decomposition via singular value ratios; same goes for figs. 16 to 18 for CFD data of flow around an airfoil, with the same intention as the previous case and again for the top five and twenty modes, next to a singular value ratio comparison against Fourier and physical; finally, we have figs. 14 and 19, our final cases, extending the initial test study of interpolation error in figs. 6 and 7. Here instead performing interpolation in RCDT-POD space for the same multi-phase and airfoil CFD before, respectively. The desire being to gauge the effectiveness of the RCDT-POD beneficial predictive capabilities via interpolation.

The results of our analysis show that RCDT, alongside POD, is a promising novel approach for the model order reduction of advective-dominant flows and subsequent predictive interpolation. Giving good computational efficiency to even the more challenging cases containing highly complex flow patterns, while retaining a good degree of accuracy. Additionally our preliminary interpolation error results signify that, while the intrinsic and reconstructive errors hold importance, the interpolation error when predicting flows is order magnitudes larger. Making them negligible in comparison and a great deal easier to remedy via post-processing options.

This report is organised as follows: RCDT methodology is introduced in §2; before numerical implementation of the transform is investigated with simple test cases in §3; RCDT is afterwards applied in the context of ROM for CFD data in §4; concluding the report with future directions of the work in §5.

2 Methodology

Within §2 we provide definitions of the Radon and cumulative distribution (CDT) transformations, given in §2.1 and §2.2. Reducing high-dimensional signals to a series of one-dimensional ones and interpreting 1-D signals as probability density functions respectively. The combined RCDT is later defined in §2.3. Finally, the proper orthogonal decomposition (POD) method of model order reduction is defined in §2.4, using the singular value decomposition.

2.1 Radon Transform

The Radon transform is a widely used image transform in fields such as medical imaging and optics. Its inverse transform (iRadon) can be used to reconstruct images from probe-measured distributions, see [33, 34]. The Radon transform $\mathcal{R}f : \mathcal{S}^{n-1} \times \mathbb{R} \to \mathbb{R}$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ can be defined as

$$\mathcal{R}f(\boldsymbol{\xi}, p) = \int_{\mathbf{x}\cdot\boldsymbol{\xi}=p} f(\mathbf{x}) \mathrm{d}m(\mathbf{x}), \tag{1}$$

where $\mathrm{d}m$ is the Euclidean measure over the hyperplane. For any fixed pair $(\boldsymbol{\xi}, p) \in \mathcal{S}^{n-1} \times \mathbb{R}$ the set $\{\boldsymbol{x} \in \mathbb{R}^n | p = \boldsymbol{x} \cdot \boldsymbol{\xi}\}$ defines a hyperplane. Therefore, the transform is an integration of the function over this hyperplane. In $\mathbb{R}^2$, the Radon transform integrates a function $f(x, y)$ over a series of lines and can be written explicitly as

$$\mathcal{R}f(\theta, s) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, \delta(s - x\cos\theta - y\sin\theta)\, \mathrm{d}x\, \mathrm{d}y. \tag{2}$$

Here s denotes the oriented distance of the line $l_{\theta,s}$ to the origin, and θ is the angle as defined in polar coordinates. Here θ lies in the range $0 \leq \theta < \pi$ due to the symmetry of line integrals for a 2D function in polar coordinates.

2.2 Cumulative Distribution Transform

The Cumulative Distribution Transform (CDT), such as in [27], is a novel 1D signal transform defined for signals that can be interpreted as 1-D probability density functions. The CDT has been developed to aid in object recognition problems, rendering certain types of classification problems linearly separable in the transform space (data classification examples include hand gestures, accelerometer data [27], wave signals [35] and optimal mass transport signal processing, see [36]). The CDT has been developed from work on linear optimal transport methods; in contrast to linear transform methods such as Fourier and Wavelet, which consider intensity variation at fixed coordinates, the CDT considers the location of intensity variations within a signal.

Say we have an input signal $f(x)$ and a reference signal $r(x)$ that are both positive and defined on $[x_1, x_2]$, which are then normalised such that

$$\int_{x_1}^{x_2} f(x)\, \mathrm{d}x = \int_{x_1}^{x_2} r(x)\, \mathrm{d}x = 1. \tag{3}$$

The CDT of $f(x)$ with respect to the reference function $r(x)$ is defined as the strictly increasing function $\hat{f}(x)$, which satisfies the relation:

$$\int_{x_1}^{\hat{f}(x)} f(x')\,\mathrm{d}x' = \int_{x_1}^{x} r(x')\,\mathrm{d}x'. \tag{4}$$

The inverse of the CDT (iCDT) is given by

$$f(x) = r(\hat{f}^{-1}(x))\frac{d}{dx}\hat{f}^{-1}(x), \tag{5}$$

where $\hat{f}^{-1}(\hat{f}(x)) = x$. The function can then be de-normalised to return to the original value range. The CDT is an invertible nonlinear signal transform, transforming from the space of smooth probability densities to the space of differentiable functions. This allows the CDT framework to consider both the pixel intensity variations and the locations of the intensity variation within the signal [27]. In contrast, linear signal transforms (such as the Fourier transform) lack the ability to tie intensity variations to locations within the signal and deal with intensity variations at fixed points. A consequence, however, of being a nonlinear signal transform is the subsequent intrinsic error left after inverting back to physical space. This will be looked into more later on.

2.3 Radon Cumulative Distribution Transform

The Radon Cumulative Distribution Transform (RCDT) [37, 38, 39] is a combination of the cumulative distribution transform (CDT) and the Radon transform. The Radon transform is used to reduce high-dimensional signals down to a series of 1-D signals, these are then passed into the CDT and interpreted as probability density functions. This allows us to expand the use of one-dimensional CDT to higher dimensions, such as two or three-dimensional flow velocity data. In this particular work, we utilise the RCDT to transform 2-D data in the form of uniformly-spaced grids, which can be interpreted as images. For 3-D data sets we have chosen to take 2-D slices along one dimension of the data and feed these into the 2-D RCDT procedure, rather than using a 3-D Radon transform algorithm. This is due to the large amount of artefacts introduced to the data by the 3-D Radon and subsequent inverse Radon transform algorithms.

The RCDT $\tilde{f}$ is obtained by applying the CDT along each projection angle in the prior Radon transform, which results in

$$\int_{s_1}^{\tilde{f}(s,\theta)} \mathcal{R}f(s',\theta)\,\mathrm{d}s' = \int_{s_1}^{s} \mathcal{R}r(s',\theta)\,\mathrm{d}s' \tag{6}$$

for the 2D case where $\mathcal{R}$ denotes the Radon transform.

For the interested reader, details of the computational algorithms for the discrete versions of the CDT and combined RCDT that are used here are laid out in [27] and [37] respectively. In previous applications within image processing, the RCDT has been shown to capture scaling and transport behaviours well, allowing interpolation between geometrical features within data to great success (note that this feature arises within the CDT rather than Radon transform). For example, the RCDT was used for interpolation between pictures of human faces [37] and used to increase efficiency of image classification with neural networks [38]. In this work we make use of the *PyTransKit* package to carry out the RCDT, which has been developed by Rubaiyat *et al.* [28].

A caveat, however, that comes from the non-linearity of RCDT is the introduction of intrinsic error following the forward and inverse transforms between original and RCDT space. Later in §3 this error is visible on image and flow boundaries when performing RCDT, and inverse; comparing to the original image or flow.

2.4 Reduced Order Modelling with Proper Orthogonal Decomposition

Proper Orthogonal Decomposition (POD) is the most widely used technique to compress the solution manifold of a variety of problems like the unsteady Navier-Stokes equations, and has been applied ubiquitously in ROM research during the past decades. The method dates back to the work of *Pearson* [40], with POD first being applied to turbulent flows by *Lumley* in 1967 [41]. POD is closely related to methods in other areas of mathematics e.g., principal component analysis (PCA) in statistical analysis [42] and the Karhunen–Loève expansion in stochastic modelling [43]. POD is a technique for computing an orthonormal reduced basis for a given set of experimental, theoretical or computational data. Specifically,

the POD basis is obtained performing the singular value decomposition of the snapshots' matrix, which is assembled with sampled data at distinct time instances ('snapshots'). The computation is described for the example solution $\mathbf{u}(\mathbf{x}, t)$, for $\mathbf{x} \in \mathbb{R}^3$ and $0 \leq t \leq T$. Decomposing into a time-averaged base solution and sum of POD modes taken, along with their time-dependent coefficients. The numerical implementation for the computation of the POD in this work is handled by the python package *EZyRB* [29].

SVD is a way to obtain POD of a dynamical system when given a sequence of data snapshots at various time instances. First, we seek an approximation to the solution $\mathbf{u}(\mathbf{x}, t)$ by the sum of a base stationary flow $\bar{\mathbf{u}}$ and a linear combination of spatial modes $\Psi_i(\mathbf{x})$ and temporal coefficients $a_i(t)$ for a chosen $i = 1, \ldots, N_r$. That is,

$$\mathbf{u}(\mathbf{x}, t) \approx \mathbf{u}_r(\mathbf{x}, t) = \bar{\mathbf{u}}(\mathbf{x}) + \sum_{i=1}^{N_r} a_i(t)\Psi_i(\mathbf{x}). \tag{7}$$

Here, $\bar{\mathbf{u}}(\mathbf{x}) = \int_0^T \mathbf{u}(\mathbf{x}, t)dt$ is the time-averaged base stationary flow for $N_r \geq 1$ modes. This is reasonable for a fluid flow that can be approximated as a stochastic and stationary process in time and ergodic. The spatial modes are orthogonal, that is to say $\langle \Psi_i, \Psi_j \rangle = 0$ for $i \neq j$ where $\langle \cdot, \cdot \rangle$ denotes the L_2 inner product. The time coefficients can be calculated using various methods, the most common being via Galerkin projection of the original system onto the spatial modes, where the resulting ROM is then termed a *Galerkin ROM* or *POD-Galerkin ROM* [44, 45]. Other methods for developing a ROM from POD are *POD with interpolation (PODI)*, *POD with Gaussian process regression (POD-GPR)* and *POD with neural networks (POD-NN)*. The first approach falls into the category of intrusive ROMs while the second one falls into the category of non-intrusive ROMs [46]. Intrusive methods exploit the discretised equations and project them onto the space spanned by the POD modes to obtain a lower dimensional system of ODEs. This technique, exploiting the governing equations, is naturally *phyiscs-based* and, usually, has better extrapolation properties. However, it requires a larger implementation effort and needs access to the discretised differential operators assembled by the full order model (hence the name intrusive). Moreover, in the case of non-linear/non-affine problems, the speed-up that can be achieved is usually smaller with respect to non-intrusive approaches [47]. On the other hand, non-intrusive methods are purely data-driven and can guarantee a speed-up also in non-linear/non-affine cases. In this article we will focus our attention only on non-intrusive methods. The modes $\Psi_i(\mathbf{x})$ of the decomposition are calculated from snapshots of the data. The data snapshots are taken as state solutions of the system; computed at different points in time, or at different parameter values, denoted by $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ where $\mathbf{x}_j = \mathbf{x}(t_j, \mathbf{p}_j) \in \mathbb{R}^N$ denotes the j^{th} snapshot at time t_j and parameter values $\mathbf{p}_j$, with n being the total number of snapshots. Define a snapshot matrix $\mathbf{X} \in \mathbb{R}^{N \times n}$ constructed with columns as each snapshot $\mathbf{x}_j$. The singular value decomposition – the generalised form of eigen-decomposition for non-square matrices – of the snapshot matrix $\mathbf{X}$ is then,

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^T, \tag{8}$$

where the columns of $\mathbf{U} \in \mathbb{R}^{N \times n}$ and $\mathbf{V} \in \mathbb{R}^{n \times N}$ are the left ($\mathbf{u}_j$) and right ($\mathbf{v}_j$) singular vectors of matrix $\mathbf{X}$ respectively. The columns of $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, so they satisfy $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbb{I}$, $\mathbb{I}$ being the identity matrix. The singular values of $\mathbf{X}$, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$, are found along the diagonal of Σ, such that $\Sigma \in \mathbb{R}^{n \times n} = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_n)$. The POD basis vectors can then be chosen as the r largest singular values and their corresponding left singular vectors in $\mathbf{U}$, $\mathbf{u}_j$. The r singular values are generally chosen based on the ratio of the 'energy' contained in the r modes and the total 'energy' in all n modes, that is the ratio (or tolerance, κ) given by,

$$\frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2} = \kappa. \tag{9}$$

The POD can be applied to both the time domain in time-dependent problems, or to the parameter domain in parametric flow problems, in both cases, the method remains the same and the snapshots are sampled in either time or parameter space.

An obvious consequence of taking the reduced r modes is the subsequent lost 'energy' of the $n-r$ modes neglected for modelling efficiency. This comes in the form of reconstruction error in the geometrical shape of the POD results; in the flow or moving object geometry. Taking a sufficiently large number of modes reconciles these errors and will be observable in the test cases of §4.

3 RCDT implementation and applications

This section focuses on the algorithmic procedure of RCDT and on testing its consistency in image capture and interpolation. The implemented steps of the RCDT process have been written using *Python 3.9.7*. Construction of the discrete RCDT image is done via use of the package *PyTransKit* [28], used for example in [38], containing the RCDT class for forward and inverse transforms of two-dimensional images, defined as a data array, assuming the image is normalised.

To test RCDT consistency, we present some initial test cases as image-like inputs to the RCDT algorithm. These are transformed into RCDT space through the numerical algorithm, a discretisation of the continuous RCDT presented previously, before being inverse transformed back into the original 'physical space'. Comparing the output image to the original input image then allows analysis of the numerical RCDT algorithms consistency, identifying any error introduced in the procedure; allowing us to observe the intrinsic error caused by RCDT. Knowledge of this error is necessary for later studies involving reduced order modelling and predictive interpolation, so the intrinsic error from the RCDT itself is already accounted for.

3.1 Numerical implementation

Here we provide the algorithmic steps implemented – see algorithm 1 – to analyse RCDTs' capability in retaining a single snapshot images' geometrical shape, i.e. its consistency; observable by successive forward and inverse RCDT on the given image. Comparing the outputted forward and inverse image with the original displays the intrinsic error – a consequence of non-linear transformation – that we desire. Finally, the average relative L_2-norm error is computed as a scalar quantifier to help compare among image cases; defined as,

$$e_2 = \frac{||x - \tilde{x}||}{||x||} = \frac{\left(\sum (x_i - \tilde{x_i})^2\right)^{1/2}}{\left(\sum (x_i^2)\right)^{1/2}}, \tag{10}$$

where x_i is the true cell value and $\tilde{x_i}$ is the approximated cell value at cell i.

Qualitative results, i.e. plots, allow observance of how RCDTs' residual artefacts (intrinsic error) distribute throughout the outputted image, while the L_2-norm error lets us quickly identify image 'traits', for example smoothed image boundaries, which cause comparably better RCDT consistency than their counterparts i.e., sharp boundaries.

Algorithm 1: Implemented pseudo-code algorithmic steps for RCDT of a given input image.

Data: Input image array K; CDT reference array B;
Result: Apply RCDT, followed by its inverse, on input K. Compare result with K to observe RCDT consistency and error.

$\hat{K} \leftarrow K \div ||K||$; /*Normalise input image.*/
$\hat{\ell} \leftarrow [\min(\hat{K}),\ \max(\hat{K})]$; /*Store limits of normalised image.*/
$R_c \leftarrow$ RadonCDT() ; /*Create RadonCDT object from *pytranskit* package.*/
$R_c^f \leftarrow R_c$.forward$([0,1], B, \hat{\ell}, \hat{K})$; /*Perform forwards RCDT on $\hat{K}$; inputs: reference signal range; reference signal array; signal to transform range; signal to transform.*/
$R_c^r \leftarrow R_c$.inverse$(R_c^f, B, \hat{\ell})$; /*Perform inverse RCDT (iRCDT) on R_c^f; inputs: signal to inverse; reference signal; inverse signal range.*/
$R_c^r \leftarrow R_c^r \times ||K||$; /*De-normalise the outputted RCDT to iRCDT image, R_c^r.*/
$\gamma \leftarrow ||R_c^r - K|| \div ||K||$; /*Calculate relative L_2-norm error*/
Plot $\leftarrow K$, R_c^r & $K - R_c^r$; /*Plot input image; outputted RCDT to iRCDT image; give comparison*/

3.2 Algorithm verification

Here we present a selection of preliminary results focused on testing the consistency of the discrete RCDT algorithm and the intrinsic error introduced. These test cases consist of simple images such as a circle, fig. 3, a ring obtained from applying an edge filter to a circle, fig. 4, and a Gaussian function, fig. 5. The domain in these images was chosen to be 0 values, with the features as 1 values, or in the case of the the continuous functions, values between 0 and 1 (from here these images will be referred to as the 'regular' images). Tests were also performed when these test images were 'inverted'. That is to say the 'inverted' image pixels had values of $1 - x$ where x is the original pixel value. In practice this leads to the image features taking on a value of 0, whereas the background domain values are of value 1 (in the following referred to as the 'inverted' images; for example, see fig. 3). For each test case the RCDT was performed on the image, followed by the inverse RCDT (iRCDT) to return back to the physical domain. The outputted image is then compared to the original input image; the difference between them is calculated cell-wise by eq. (10); therefore, letting us observe any intrinsic error, or in other words artefacts, of RCDT.

All test cases within this section of observing RCDT's intrinsic error, i.e. image consistency, are performed on a uniform two-dimensional grid of 250 by 250 pixels (px); ergo, 62500px total. The various images used to test the intrinsic error are constructed as 250 by 250 data arrays, each element representing a pixel and its 'intensity' – e.g. flow velocity, or, an object – within the array.

Each shows some inconsistency in the RCDT, iRCDT procedure, as expected by a non-linear transform. In the outputted images we observe an introduction of artefacts. In the circle test case seen in fig. 3, the resulting image differs from the input around the edge of the circle. In particular, we observe an undershoot and overshoot pattern, with the inner edge of the circle showing a slightly lower predicted value and the outer edge showing higher. A similar effect is observed in the circle edge test image in fig. 4, however the extent of the error appears more localised, possibly due to the small width of the feature in the image.

In the Gaussian test case, fig. 5, the error around the edges of the Gaussian feature are similar to the circle cases, but with additional over and undershooting inside the feature. Despite this extra qualitative error, the quantitative error is actually an order of magnitude smaller than the circle test cases. This under/overshoot is possibly being caused by the change of values in the image domain, with a discontinuous and large value change being associated with a larger error, for example the error values are much larger for the discontinuous circle test case compared to the continuous Gaussian test case. This is also supported by the smoothed circle and smoothed edge test cases, seen in fig. 1 respectively. In these cases the discontinuous edges of the features are smoothed out with a Gaussian filter, in order to render the changes in the image continuous. We see that both the under/overshoot error magnitudes and the overall image errors in these cases are of smaller magnitude than their discontinuous counterparts, supporting the hypothesis that a smooth change in values is handled better by the transform and inverse transform algorithms and results in a more consistent result.

In the Gaussian test case, fig. 5, the error around the edges of the Gaussian feature are similar to the circle cases, but with additional over and undershooting inside the feature. Despite this extra qualitative error, the quantitative error is actually an order of magnitude smaller than the circle test cases. This under/overshoot is possibly being caused by the change of values in the image domain, with a discontinuous and large value change being associated with a larger error, for example the error values are much larger for the discontinuous circle test case compared to the continuous Gaussian test case. This is also supported by the smoothed circle and smoothed edge test cases, seen in figs. 1 and 2 respectively. In these cases the discontinuous edges of the features are smoothed out with a Gaussian filter, in order to render the changes in the image continuous. We see that both the under/overshoot error magnitudes and the overall image errors in these cases are of smaller magnitude than their discontinuous counterparts, supporting the hypothesis that a smooth change in values is handled better by the transform and inverse transform algorithms and results in a more consistent result.

In the inverted cases another unique artefact is seen at the corners of the resulting images. For example, in fig. 3 there is a relatively large overshooting of the background value in the corners of the image, extending partially along the edges of the image. This effect is exclusive to the inverted image cases and is not seen at all in the regular images.

Upon further investigation it is hypothesised that these errors arise from the numerical implementation of the CDT being used for the transformation, rather than from the Radon transform algorithm. This was concluded after the images were transformed first into Radon space, this Radon space image was then iRadon transformed and this result compared to the image after applying CDT-iCDT before iRadon once again. The Radon-iRadon image was free of artefacts, whereas the RCDT-iRadonCDT image contained

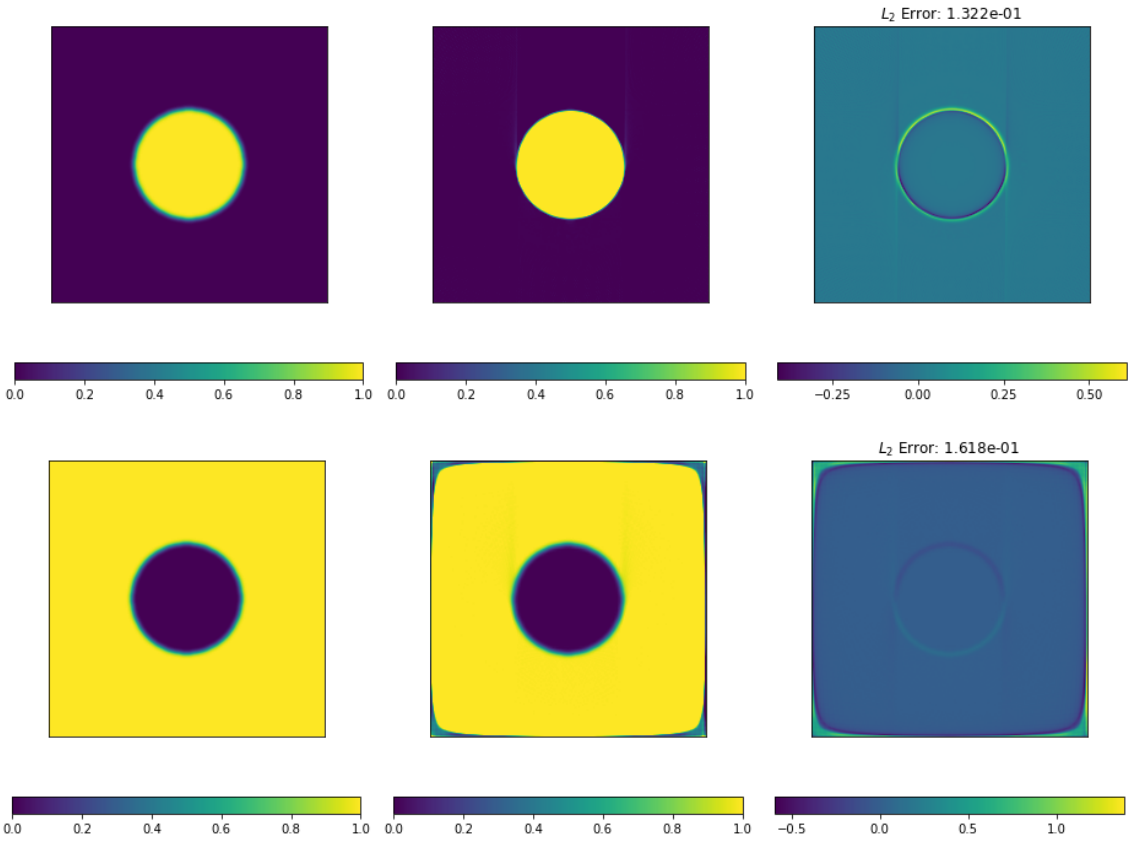


Figure 1: **Left**: input image. **Centre**: result of RCDT followed by iRCDT on input image. **Right**: difference between stated input and result, and L_2-norm error value.

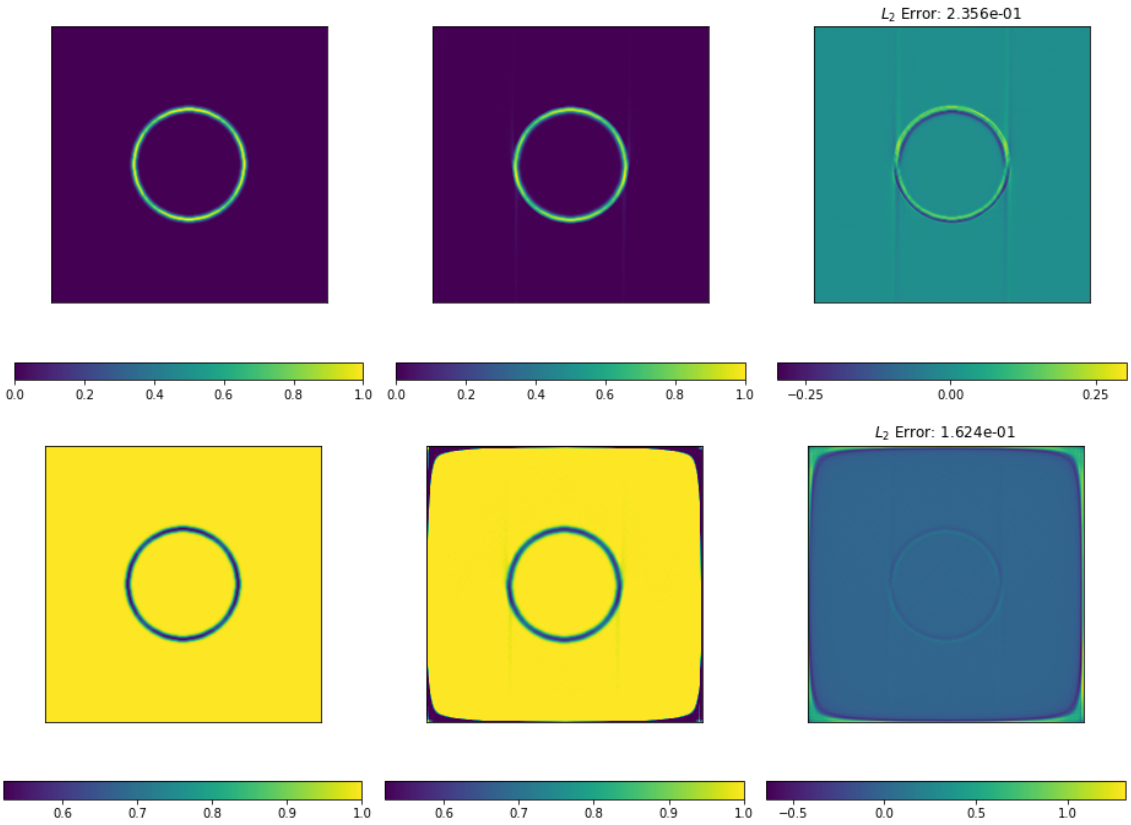


Figure 2: **Left**: input image. **Centre**: result of RCDT followed by iRCDT on input image. **Right**: difference between stated input and result, and L_2-norm error value.

similar edge artefacting to shown in the examples discussed here. The result of the Radon space image undergoing CDT-iCDT back to Radon space was also compared with the original Radon-transformed image, where error in Radon space was introduced by the CDT procedure.

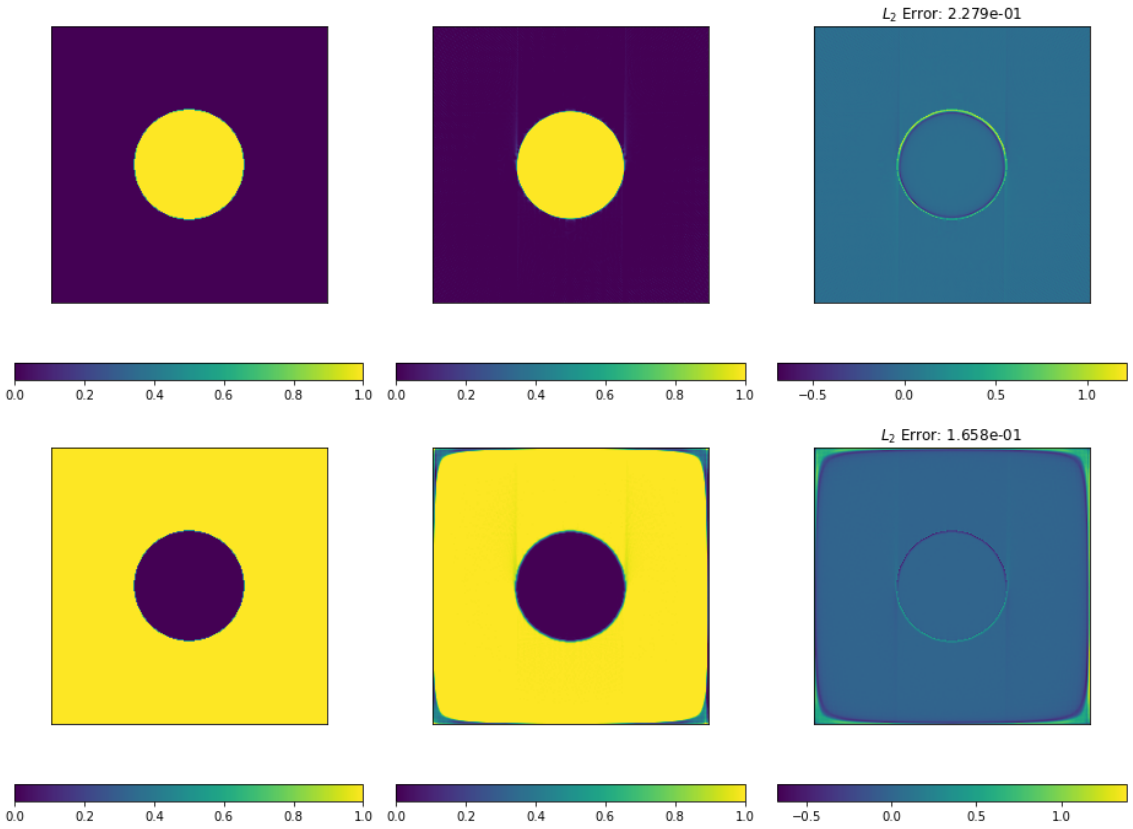


Figure 3: **Left:** input images of a circle defined by ones (top) or zeros (bottom) and vice versa for background. **Centre:** result of RCDT followed by iRCDT on the input images. **Right:** difference between stated input images and RCDT iRCDT results, and L_2-norm errors.

The error values for all the RCDT test cases can be found in table 1. Notably the error for the standard Gaussian case is an order of magnitude lower than all other examples tested, whereas the least accurate reconstruction was in the circle edge case. Of our examples, the Gaussian case is the example where the inputs to the CDT function from the Radon transform algorithm most resemble a well-defined, smooth probability density function. Seeing as the CDT is defined from the space of smooth probability density functions, it is natural that this example returns the most consistent result, compared to the others where the input to the CDT are less well-defined or non-smooth.

From the features seen and errors encountered with these test examples, we can gain some insight as to how useful the RCDT may be when applied to the area of CFD. Firstly, the fact that many examples of CFD data will not exhibit sharp discontinuities is helpful for the RCDT, because although it does deal with sharp boundaries, it is much less error-prone when used on smooth transitions and boundaries. Another feature seen is the rounding of edges and addition of zero values along the non-zero borders of the inverted images. This could lead to problems when applied to CFD data and must be accounted for, if not resolved with a modified RCDT algorithm. This is only a problem when the boundary values are non-zero, so in some cases e.g., interacting twin jet flows in fig. 7, this is not an issue.

3.3 Interpolation study

For studies of real-world applications 'controlled' errors, like the intrinsic error of RCDT above, are an acceptable compromise for the capability of interpolating highly nonlinear geometric flow features. A crucial aspect for modelling a variety of singular and multiple flow configurations for various parameter (spatial and transient) variations. This is in part due to how these 'controlled' errors can be compensated

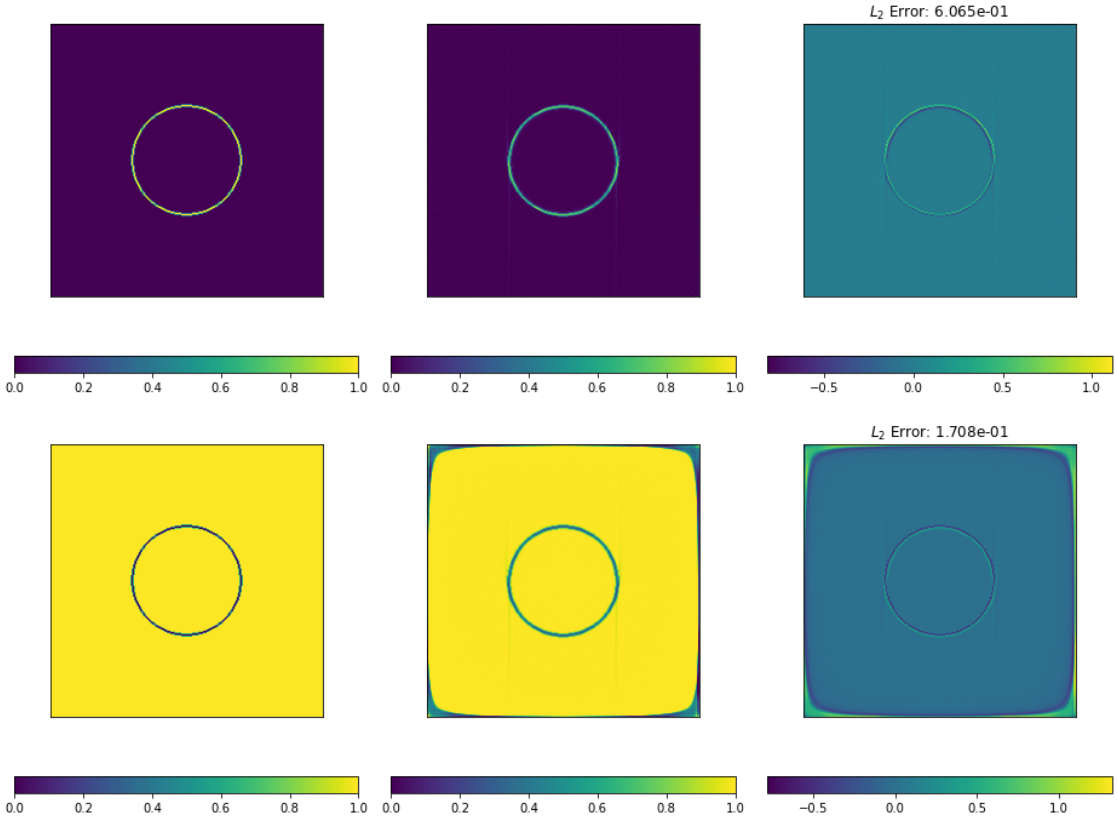


Figure 4: **Left:** input images of a circular edge ring defined by ones (top) or zeros (bottom) and vice versa for background. **Centre:** result of RCDT followed by iRCDT on the input images. **Right:** difference between stated input images and RCDT iRCDT results, and L_2-norm errors.

RCDT test cases	**L_2-norm error**
Circle	2.279×10^{-1}
Circle inverse	1.658×10^{-1}
Circle smoothed	1.322×10^{-1}
Circle smoothed inverse	1.618×10^{-1}
Circle edge	6.065×10^{-1}
Circle edge inverse	1.708×10^{-1}
Circle edge smoothed	2.356×10^{-1}
Circle edge smoothed inverse	1.624×10^{-1}
Gaussian	3.204×10^{-2}
Gaussian inverse	1.628×10^{-1}

Table 1: L_2-norm error values for RCDT test cases. Quantitatively, errors for the inverted version of each case fared better than their original counterparts, though marginally. Contrasting to the qualitative differences seen in test case figures. Smoothed cases – of the originals – for the circle and circle edge also fared better by roughly factors of a half and third, respectively.

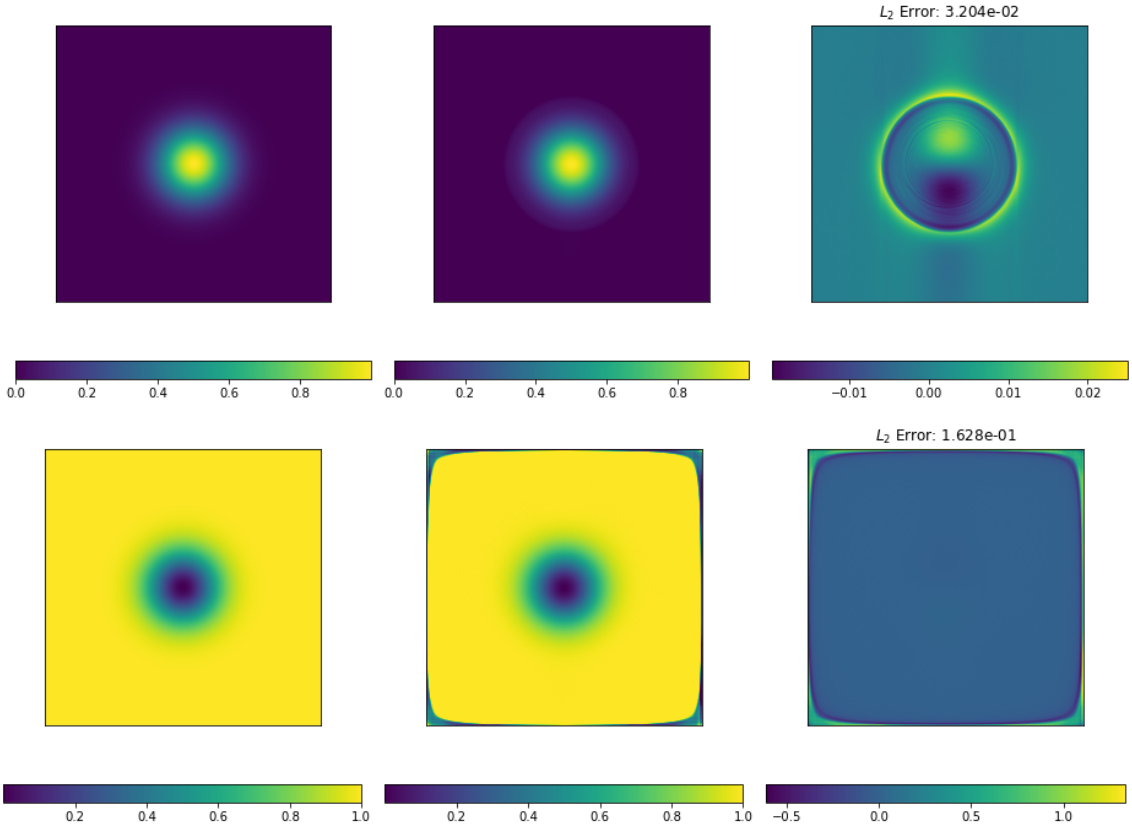


Figure 5: **Left:** input images of a normal Gaussian distribution function (top) ranging from 0 to 1, and vice versa (bottom). **Centre:** result of RCDT followed by iRCDT on the input images. **Right:** difference between stated input images and RCDT iRCDT results, and L_2-norm errors.

for – or ultimately removed – in post-process. RCDTs intrinsic error is mostly observed around image and flow boundaries (see figs. 3 and 4 and later on fig. 12) but still permits good image/flow capture. Same can be said of the 'controlled' POD reconstructive error mentioned in §2.4; analysed later in §4. Considering a large number of spatial modes negates the error, but post-processing could again be utilised. Despite these caveats of RCDT – and likewise ROM/POD – in introducing systematical errors, the novel capability of interpolating nonlinear flow features makes it a small price to pay for such a valuable function.

What's more our preliminary results of interpolating in-between two flow configurations, figs. 6 and 7, suggest these 'controlled' errors are likely negligible compared to error with respect to physical flow interpolation. The figs. 6 and 7 are the result of two duo jet flow configurations, differing by the separation width of the duo jets and subsequent intertwining flows; interpolated in-between to arrive at a desired target flow image given in the right-side sub-figures of figs. 6 and 7. Interpolation is done in physical – using POD mentioned in §2.4 – and RCDT space for comparison. The left and centre images of fig. 7 are the resultant physical POD and RCDT interpolations respectively, showing clear failure of physical space i.e., 'standard', POD in interpolating; a stark to the results of RCDT space interpolation. The predicted flow of RCDT, while having some over-shoot in inter-flow contact and dissipation, is a strikingly good result, at least qualitatively shown here. Matching the desired target image with only some degree of error, both intrinsic and interpolated, that appears to smooth the outer jet flow 'boundaries', while the inner 'boundary' is sharpened.

Whilst only qualitatively shown for now, it seems interpolation error far over-shadows the intrinsic. Making it negligible in comparison; much like the reconstruction error of POD/ROM, observed and tested later in §4; determined by the number of spatial modes taken into account in model reduction. The more modes the better POD is able to reconstruct the image, and so the less reconstruction error seen in the output. It should be stated again these 'controlled' errors can be removed in image post-processing; later studies will utilise this to focus more on analysing the interpolation error briefly shown in fig. 7. With the end goal of accurately predicting flows using our RCDT-POD ROM workflow for a large number of parameter variations and flow configurations.

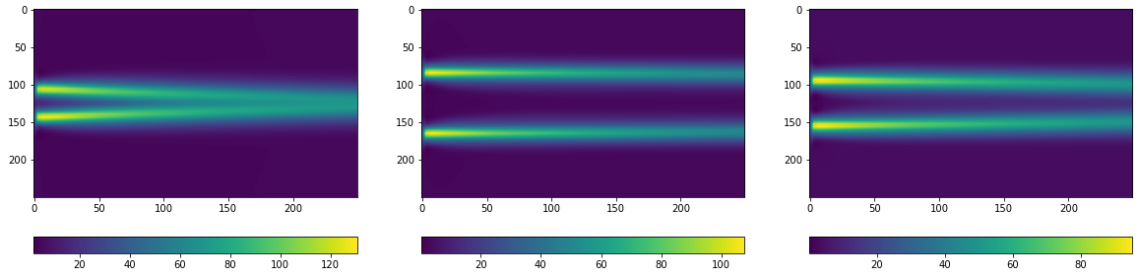


Figure 6: Input and target twin jet flow configuration images for differing separation widths. **Left**: Input image 1; Jets are close together; flows interact and merge close to sources. **Centre**: Input image 2; jet sources are further separated; therefore, flows do not merge together. **Right**: Target image; a jet configuration in between input images 1 & 2.

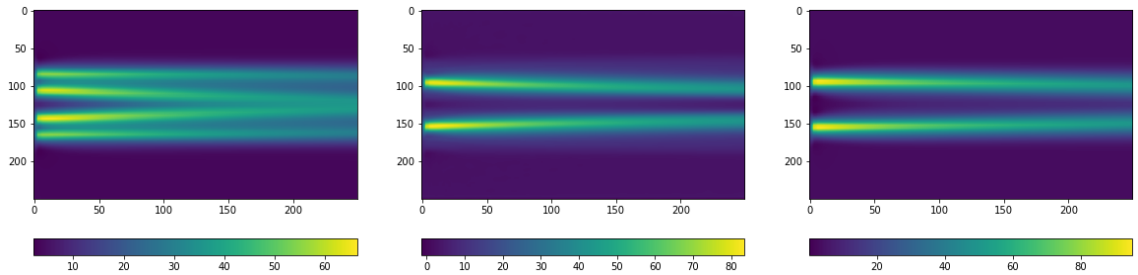


Figure 7: The resulting jet flow configuration images from interpolating the two input images in fig. 6. **Left**: Interpolation in physical space; failure of *standard* POD. **Centre**: Interpolation in RCDT space; qualitatively similar to target image, some over-shoot in inter-flow contact angle and smoothed jet boundaries. **Right**: Target image.

4 RCDT-POD for model order reduction

To gather an understanding of the reconstruction error introduced by the POD modes component of our RCDT-POD workflow, we looks towards comparison against high-fidelity CFD simulated data in the following subsections. The RCDT-POD procedure is compared against standard POD in physical space and as such the aforementioned intrinsic error is present. To observe changes in the reconstructive error of POD we make comparison with the original data for choices of $r = 5$ modes and $r = 20$ modes per subsequent case. Apart from §4.1, an initial example where $r = 5$ modes almost perfectly constructs the original image. In the following cases we construct the ROM using POD for the spatial mode decomposition, and a Gaussian process regression (GPR) for the model prediction [48, 49].

It should be noted that from the following examples all but one, §4.1, use data from CFD simulations. This coordinated data can directly be used for physical space ROM, however, the RCDT requires a uniformly-spaced grid (image-like data) for the input. This therefore requires the original CFD data to be uniformly interpolated across the domain. This was done using the *scipy* interpolation function *griddata*. Consequently, the RCDT-POD result is compared to the uniform grid input data, and not the original coordinated CFD data.

For implementation of proper orthogonal decomposition (POD) and reduced order modelling (ROM) we use the *EZyRB* package [29]. See algorithm 2 for the algorithmic procedure for the RCDT-POD ROM workflow of a given K_i input images at snapshot times t_i for $i = 0, \ldots, N_t$ and $N_t \geq 0$. For construction of the ROMs we employ POD for snapshot decomposition using a reduced P modes, given some input training data. Then, either the Radial Basis Function (RBF) method, using default arguments where smooth and shape parameters are ignored, or Gaussian process regression (GPR) is used for the snapshot approximation, i.e. interpolation, via the reduced order model constructed by POD.

For simplicity the RCDT component of algorithm 2 is neglected. Instead delegated to being algorithm 1's outputted RCDT, followed by iRCDT, image of the given input snapshot image and reference. Outputs are similar to before, calculating the averaged relative L_2-norm error per snapshot image and plotting the result of RCDT-POD ROM and difference to the given input snapshot images. Additional outputs unlike before include the mean L_2-norm error over all snapshots, quantifying RCDT-POD ROMs' overall accuracy; the plotted arranged singular values of POD, discussed in §2.4, divided by the largest singular value and compared against fast fourier transform (FFT) and standard POD on the physical space.

The next few sections are dedicated to employing our RCDT-POD workflow to ROMs for a number of real-world applications and results within literature. This is to gauge RCDT-POD ROMs' capabilities in handling complex high-fidelity computational flow dynamics (CFD) data, using standard reduced order modelling on the physical space as a comparative measure. Furthermore, by comparing outputs with differing number of POD modes taken we can observe the reconstructive error of ROM; reduced by taking a large number of modes as discussed beforehand.

4.1 Gaussian pulse

In fig. 8 we reproduce the results for a randomly-travelling Gaussian pulse in a 2D domain, originally carried out by Ren *et al* [39]. In this case we create a domain of size 100×100 cells and then add a randomly-travelling Gaussian pulse through the variation of the Gaussian mean (standard notation μ) in the x and y directions for each time step. The Gaussian standard deviation σ is kept constant (in this example we have used $\sigma_{x,y} = 5.3$) and the means $\mu_{x,y}$ are randomly chosen from the value range $[10, 90]$. The number of time steps used is 100.

In this example, we can clearly see that the RCDT-based POD far exceeds the accuracy of the physical space-based POD, in fact almost perfectly replicating the original snapshot. In fig. 9 it is clear that the number of POD modes required in RCDT space to accurately capture most of the snapshot energy is very low, whereas for the physical space case the drop off in singular values in the POD decomposition is very slow, and hence requires a very large amount of modes to capture the energy needed to reproduce the snapshots accurately. This demonstrates the ability of the RCDT space to capture travelling wave behaviour and linearise these features such that the POD procedure can remain efficient in decomposing the problem, especially when compared to the physical space POD. Also note that for each individual snapshot we see a similar error pattern to the Gaussian cases presented previously, so these POD results are consistent with the error produced by the RCDT algorithm alone.

In fig. 9 the singular values of the POD decomposition (normalised by the first singular value) for the Gaussian POD are compared. The same example constructed in Fourier space has also been included for comparison here, as well as in future examples in this section. Singular values represent the amount of 'energy' stored in each corresponding POD mode (ordered from largest singular value to smallest).

Algorithm 2: Implemented algorithmic steps for a RCDT-POD workflow

Data: Input image arrays K_i; image snapshot time points t_i for $i = 0, \ldots, N_t$; CDT reference image array B; snapshot target image index k; number of POD modes used P

Result: RCDT-POD prediction of input images K_i at snapshot time points t_i; plots taken at snapshot time index k i.e., at t_k.

for $i = 0$ **to** N_t **do**
  $\alpha_i \leftarrow$ RCDT to iRCDT image of algorithm 1 (R_c^r) with inputs K_i & B;
end
POD $\leftarrow$ POD(P) `/*Construct POD object for` P `modes */`
for $i = 0$ **to** N_t **do**
  ROM$_i \leftarrow$ ROM(α_i, POD, $\bullet$) `/*Perform model reduction using POD and` $\bullet$ `= GPR, RBF or Linear, for each` α_i `*/`
end
$s \leftarrow$POD().singular_values() `/*Get all` P `singular values of POD */`
for $i = 0$ **to** N_t **do**
  $R_i \leftarrow$ ROM$_i$.predict(i) `/*Construct predictions of RCDT-POD at snapshot times */`
  $R_i^I \leftarrow$RadonCDT().inverse($R_i, B, [0, 1]$) `/*Invert results` R_i `back to original space */`
  $R_i^I \leftarrow R_i^I \times ||K_i||$ `/*De-normalise */`
  $E_i \leftarrow ||R_i^I - K_i|| \div ||K_i||$ `/*Get relative` L_2`-norm errors per snapshot */`
  if $i = k$ **then**
    Plot R_i^I & $K_i - R_i^I$ `/*Plot results at snapshot index` k `*/`
  end
end
$\bar{E} \leftarrow$ mean(E_i) `/*Calculate mean` L_2`-norm error over all snapshots */`
Plot singular values $s \div$ by the first singular value

Therefore, a sharp decay in singular values corresponds to a larger proportion of the total energy of the system captured by the first few POD modes, in turn allowing for a more accurate POD with fewer modes needed to capture the behaviour in the system. In this example, we see that both the Fourier space and physical space POD results in the first 20 POD mode singular values being of a similar magnitude. This means that all of these POD modes contain a relatively large amount of the total energy of the system, and hence we would need to include many of the modes in the POD construction to get an accurate result. Conversely, in the case of POD in the RCDT space, we see a very sharp drop off in singular values for the first 4 modes, as they contain the majority of the total energy of the system. As seen in fig. 8, this results in an accurate POD using only the first 5 modes, whereas in physical space the POD is completely inaccurate for the first 5 modes.

4.2 Multi-phase wave

The dataset for this example has been generated solving the unsteady Navier Stokes equations for two incompressible, isothermal immiscible fluids. The domain is given by the rectangle $\Omega = [-2.5, 3.5] \times [-0.5, 1.2]$ and the computational domain (see fig. 10) is obtained with a structured grid composed by 250×75 hexahedral cells. The results are obtained using the `interFoam` solver developed in the OpenFOAM finite volume library. For what concerns the velocity field we apply a uniform velocity $U = (0.25, 0)$m/s on the left side of the domain, a `noslip` condition on the bottom of the domain, and a `zeroGradient` (i.e. $\nabla U \cdot n = 0$) condition on the bottom and right side of the domain. For the pressure field we used, a uniform `totalPressure` with $p = 0$kg/(ms^2) to the bottom boundary, and a uniform `fixedFluxPressure` with $p = 0$kg/(ms^2) on other boundaries. The α_w field, that varies between 0 and 1 and represents the fraction of volume of water in each cell, has a uniform condition $\alpha_w = 0$ on the bottom boundary and a `zeroGradient` condition on the other boundaries. The properties of the two phases are set as $\rho_1 = 10^3$kg/m^3, $\nu_1 = 10^{-6}$m^2/s, and $\rho_2 = 1$kg/m^3, $\nu_2 = 1.48 \cdot 10^{-}5$m^2/s, where ρ_1, ρ_2 and ν_1, ν_2 are the density and the kinematic viscosity of the first and second phase, respectively. Snapshots are collected in the time window $[0, 5]$. The figs. 11 and 12 are the result of applying the same RCDT-POD projection procedure proceeding, now to the multi-phase wave data set generated by the above setting (the phase parameter α_w, denoting the fraction of water/air, is shown in these figures), using 5 and 20 modes respectively. For RCDT-POD, the CFD data is interpolated onto a uniform grid of size 200×150.

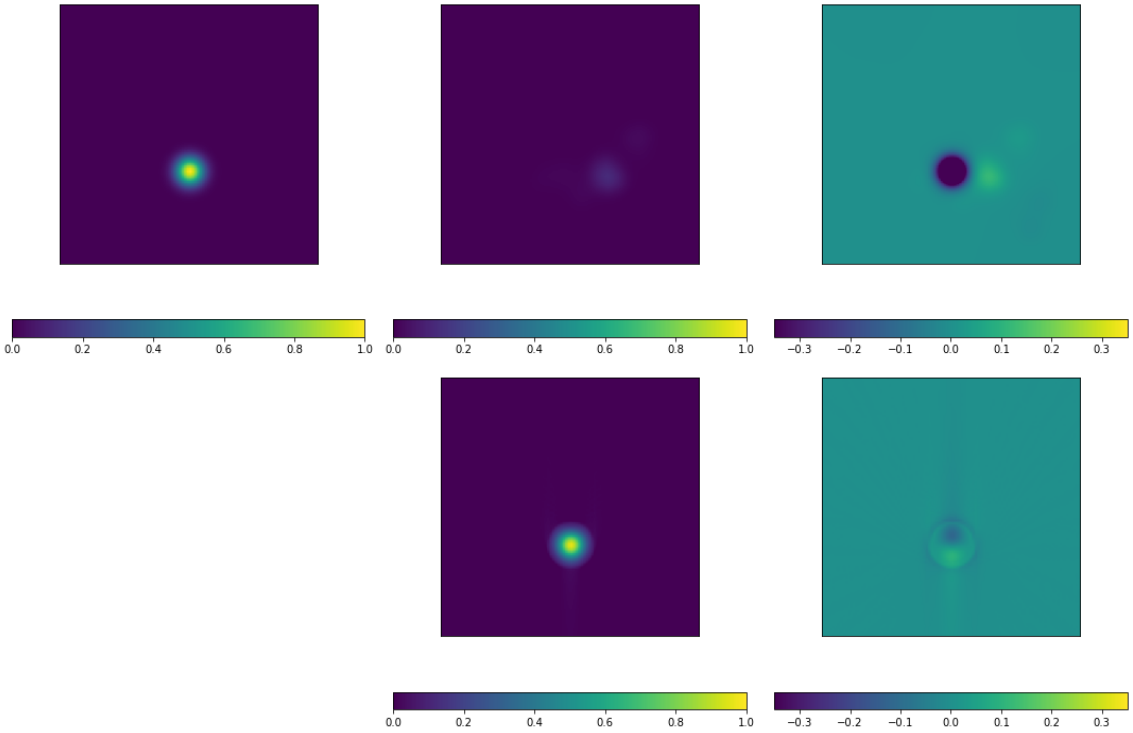


Figure 8: Single snapshot result from the randomly-travelling Gaussian POD (using top 5 modes). **Top-left:** original snapshot image. **Top-middle:** physical space POD projection. **Bottom-middle:** RCDT-POD projection. **Top/bottom-right:** respective differences for each projection from original snapshot image.

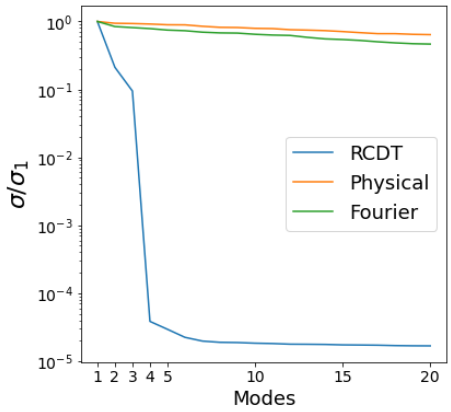
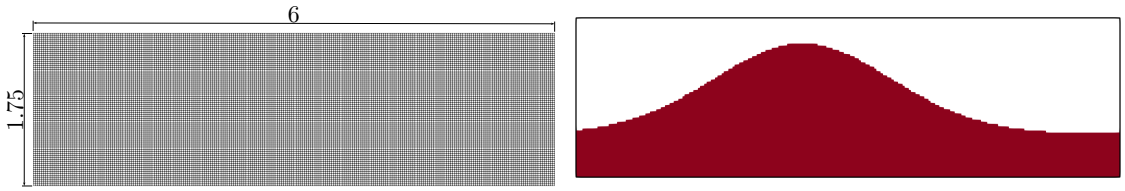


Figure 9: Plot of ratio of each singular value to first singular value in POD decomposition for physical space, Fourier space and RCDT space POD's.

The number of time steps in this data set is $N_t = 200$. The initial condition for the velocity field is uniform $U = (0.25, 0)$m/s, for the pressure field is uniform $p = 0$kg/(ms^2), while for the α_w field is set according to following profile:

$$\begin{cases} \alpha_w = 1 & \text{if} \quad y < e^{-0.5x^2}, \\ \alpha_w = 0 & \text{otherwise.} \end{cases} \tag{11}$$

The time integration is performed using an implicit first order Euler scheme.



(a) (b)

Figure 10: Computatonal domain of the multiphase CFD data set: (a) domain size and mesh grid structure, (b) initial condition for the α_w field

The data set exhibits the wave shown travelling across the domain from left to right during the time series. There is a considerable difference in the physical POD between the two different results. At 20 modes the physical POD is relatively accurate when compared with the original data, however when reducing the number of modes the physical POD struggles to capture the shape of the original wave. This is in contrast to the RCDT-POD, which retains more of the shape of the original when only using 5 modes for reconstruction. The result improves slightly for 20 modes, but the majority of the shape is captured in the first few modes unlike in the physical case. This is made clear in fig. 13 where the decay of the POD singular values for the RCDT-POD case is much quicker than in the physical and FFT space cases.

Another clear difference in the RCDT-POD is the appearance of edge-bound errors around the wave along the bounds of the domain, especially in the bottom corners. This is a consequence of the RCDT transform algorithm requiring a zero value in the first array position (this is set by the algorithm in the CDT when requiring the input to be a probability density function). Future work on this example will test the effect of adding a padding region around the data before it is used in the RCDT-POD methodology to evaluate whether adding padding, and then removing it in post-processing, will help reduce the errors around the edge of the domain (due to the zero value being set outside of the area of interest, and then any error caused by this at the domain edges being removed). There is also the possibility of using other forms of pre- and post-processing such as thresholds and edge detection, which may improve results in certain cases.

4.2.1 Predictive time interpolation

Here we give a case of the full RCDT-POD ROM workflow for predictive interpolation in time, like the preliminary study in §3.3. Displaying the primary draws of the RCDT-POD ROM: accurate image capture, and linear predictive capabilities introduced by the CDT component. The case settings we consider are similar as in §4.2 and §4.3, the multiphase and airfoil cases respectively. The difference here being, before inverting results from RCDT-POD space into the physical, snapshots determined by the RCDT-POD workflow are first linearly interpolated in time, whilst still in RCDT-POD space. Utilising a significant benefit of RCDT; linear separability of fluid flow with respect to time, or whatever parameter was fed to the CDT portion. Within fig. 14 are the results of performing the previous RCDT-POD ROM interpolation on the multiphase CFD data of §4.2, followed by prediction at a chosen target snapshot using linear interpolation in the reduced RCDT-POD space. Comparison is given against the same proper orthogonal decomposition (POD) and interpolation, done instead in the physical, and similarly Fourier, space. All transformation's, decomposition's, and interpolation's are done using a reduced training dataset of the original multiphase CFD data. This reduced dataset is composed of every fiftieth snapshot of the original $N_t = 200$ snapshots, i.e. the 1$^{\text{st}}$, 50$^{\text{th}}$, 100$^{\text{th}}$, 150$^{\text{th}}$ and 200$^{\text{th}}$ data snapshots, given on a 250 by 75 grid resolution. Linear interpolation is performed, in each space, at the chosen target snapshot index $k = 75$, so to not be a part of our training dataset, and compared against the original $k = 75$ snapshot multiphase data. Note, to better convey the error of interpolating, results

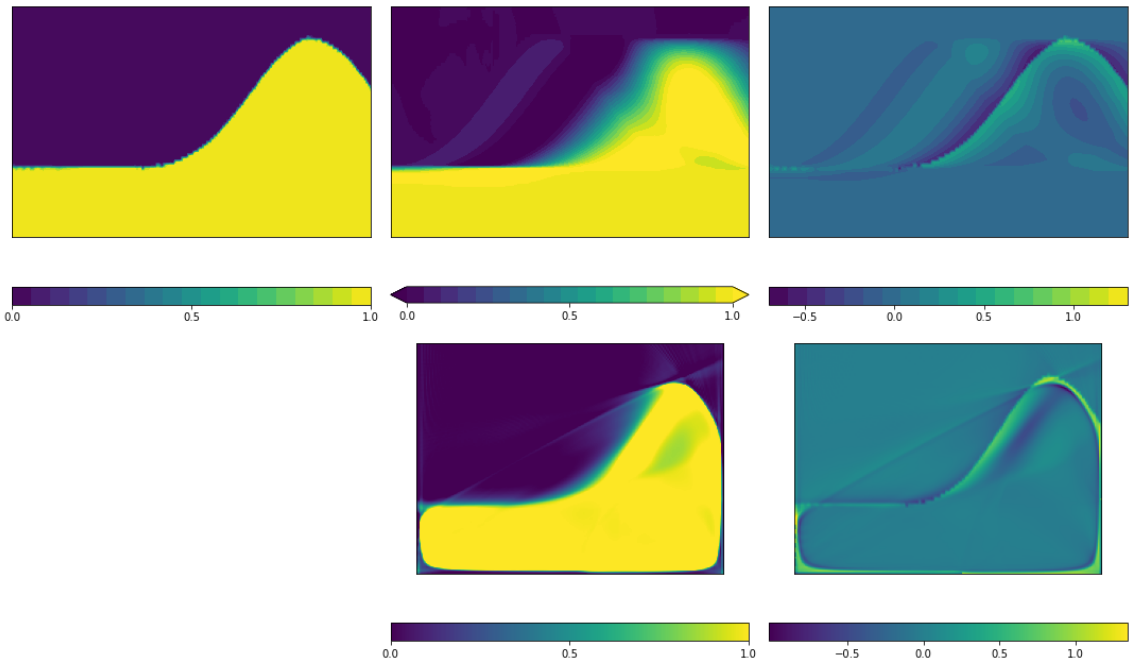


Figure 11: Single snapshot result from the multi-phase wave POD (using top 5 modes). **Top-left:** original snapshot image. **Top-middle:** physical space POD projection. **Bottom-middle:** RCDT-POD projection. **Top/bottom-right:** respective differences for each projection from the original snapshot image.

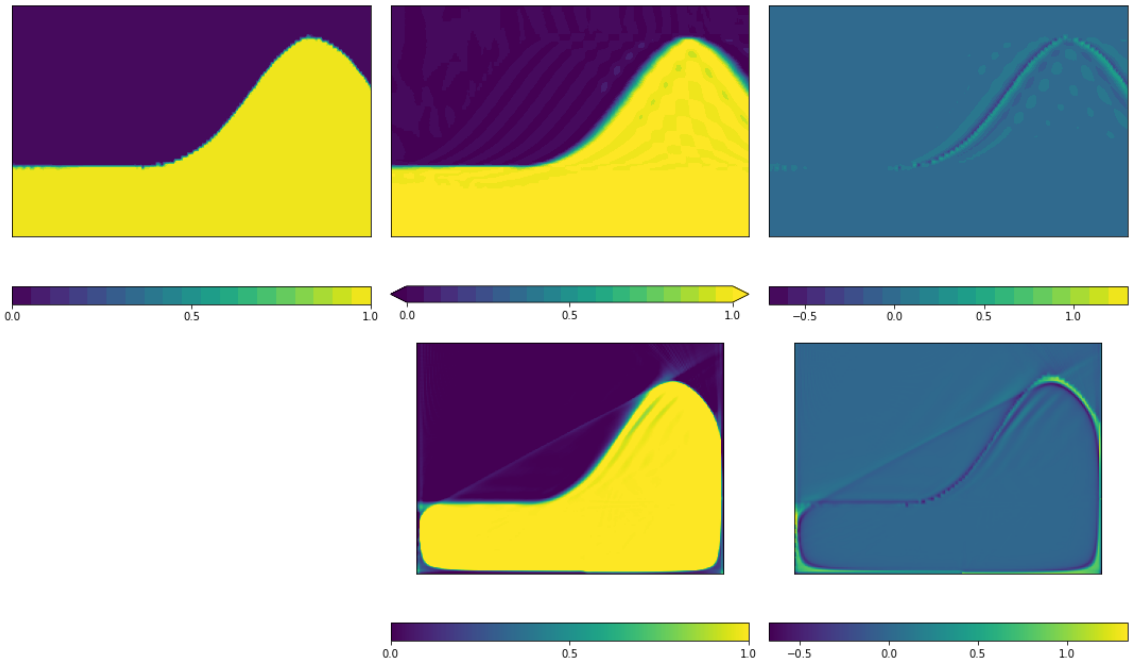


Figure 12: Single snapshot result from the multi-phase wave POD (using top 20 modes). **Top-left:** original snapshot, top-middle: physical space POD projection. **Bottom-middle:** RCDT-POD projection. **Top/bottom-right:** respective differences for each projection from the original snapshot image.

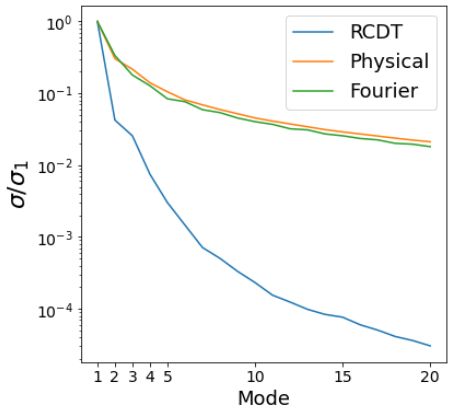


Figure 13: Plot of ratio of each singular value over the first singular value in POD decomposition for physical, Fourier, and RCDT-POD space multi-phase wave.

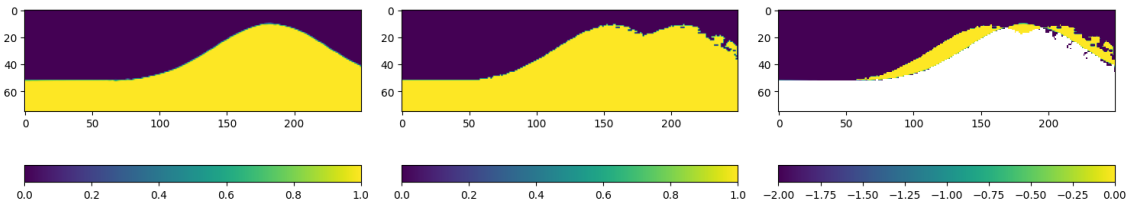
are ran through a threshold of 0.5, i.e., values above 0.5 are set to 1, otherwise they are set to 0. In fig. 14a are, from left to right: the target wave snapshot at time $k = 75$; predicted solution of linear interpolation in physical space, using POD decomposition; and, the respective base 10 logarithmic ($\log_{10}$) absolute difference between the target and predicted snapshots, with title containing the average L_1-norm error. Results of fig. 14a show clear failure of interpolation in the physical space, forming a bimodal wave with inaccurate peak location to the target snapshot, despite a quantitatively low average L_1 error. Within fig. 14b are the results of following the same procedure as in fig. 14a, but performed in Fourier space instead. Much like before, the figures show a distinct failure in interpolating the multi-phase wave within Fourier space, with the same story told by the L_1 error. Unlike the physical, we observe clustering of artefacts on wave boundary, in part due to the thresholding of results and Fouriers' wave-like nature. Inside fig. 14c are the results of interpolation within RCDT-POD space. As can be seen by the respective predicted snapshot and scaled absolute error image, compared to figs. 14a and 14b, there is a notable qualitative difference. The RCDT prediction gives a much more accurate location of the wave peak, and error at the wave boundary is primarily situated at the waves tail end. A consequence however, as seen in all other tests of RCDT, is some rounding of the outer boundaries is present. In fig. 14d we provide a verification of RCDT's accuracy when transforming, and subsequently inversely transforming, the original input snapshot data back and forth from RCDT space. Error is observed along the wave boundaries, including the external 'boundary' where we observe the rounding much like in fig. 14c.

4.3 Airfoil CFD dataset

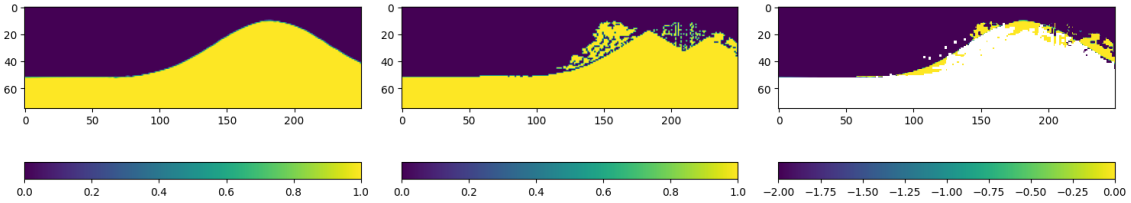
In this case the dataset is created solving the compressible Navier-Stokes equations around a NACA0012 profile. The mesh is structured and counts 4500 hexahedral elements. For more details on the mesh structure and domain dimension see Figure 15. The equations are solved employing the finite volume method and using the sonicFoam solver developed in the OpenFOAM library [50]. The simulation is 2D and boundary conditions are set according to fig. 15. In particular, the waveTrasmissive boundary condition is a special type of non-reflecting boundary condition implemented in OpenFOAM. In the picture U, T and p denote the velocity, temperature and pressure fields, respectively. The problem is transient, the time integration is carried out with an implicit Euler Scheme, and the simulation time window is $t \in [0, 3]$. The time step used to numerically solve the Navier-Stokes equations is $\Delta t = 0.001$s and we store every ten time steps. This means we acquire 300 snapshots to test our numerical pipeline. The methodology is tested on the velocity field. The inlet velocity U_x is indirectly parametrized through the Mach number Ma. In §4.3.1 we will provide more details on the different values of the Mach number used in the numerical simulations.

The figs. 16 and 17 are the result of applying the ROM procedure to an airfoil flow data set (the velocity magnitude is shown in these figures), using 5 and 20 modes respectively. For the RCDT-POD ROM the CFD data is interpolated onto a uniform grid of size 250×200. The number of time steps in this data set is $N_t = 300$.

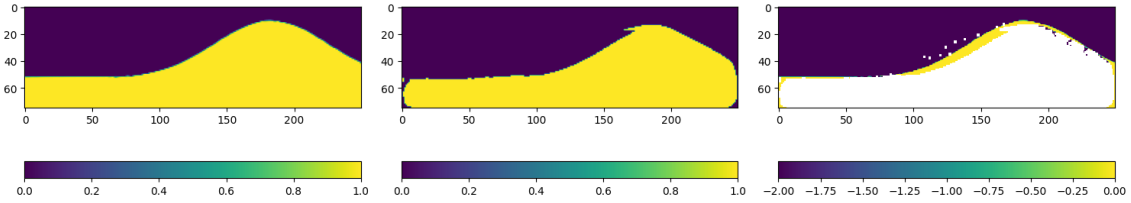
We observe that in both the physical and RCDT cases the qualitative results for each ROM do not differ considerably when moving from 5 modes to 20 modes. In the physical case, the error is reduced



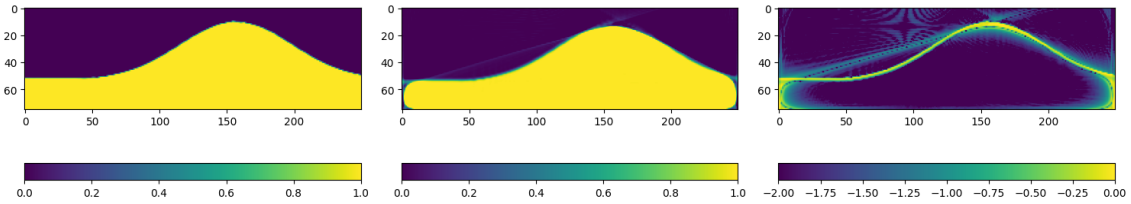
(a) Physical POD space interpolation results. **Left:** target snapshot. **Centre:** predicted solution of linear interpolation, in physical POD space. **Right:** absolute difference between target and result, given in base $\log_{10}$, alongside average L_1-norm error.



(b) Fast Fourier transform (FFT) POD space interpolation results. **Left:** target snapshot. **Centre:** predicted solution of linear interpolation, in FFT POD space, inverted back to physical. **Right:** absolute difference between target and result, given in base $\log_{10}$, alongside average L_1-norm error.



(c) RCDT-POD ROM interpolation results. **Left:** target snapshot. **Centre:** predicted solution of linear interpolation, in RCDT-POD space, inverted back to physical for comparison. **Right:** absolute difference between target and interpolation result, given in base $\log_{10}$, alongside average L_1-norm error.



(d) Accuracy test of RCDT's transformed snapshot from the original input snapshot data. **Left:** target snapshot. **Centre:** resultant image of RCDT, followed by its inverse back to physical space, at target snapshot. **Right:** difference between target snapshot and RCDT to iRCDT snapshot, given in base $\log_{10}$, alongside the average L_2-norm error.

Figure 14: Prediction of multiphase wave snapshot data, in time, using reduced order modelling (ROM) and linear interpolation i.e., RCDT-POD ROM. Interpolation in physical and Fourier space given as comparison against RCDT-POD space. space.

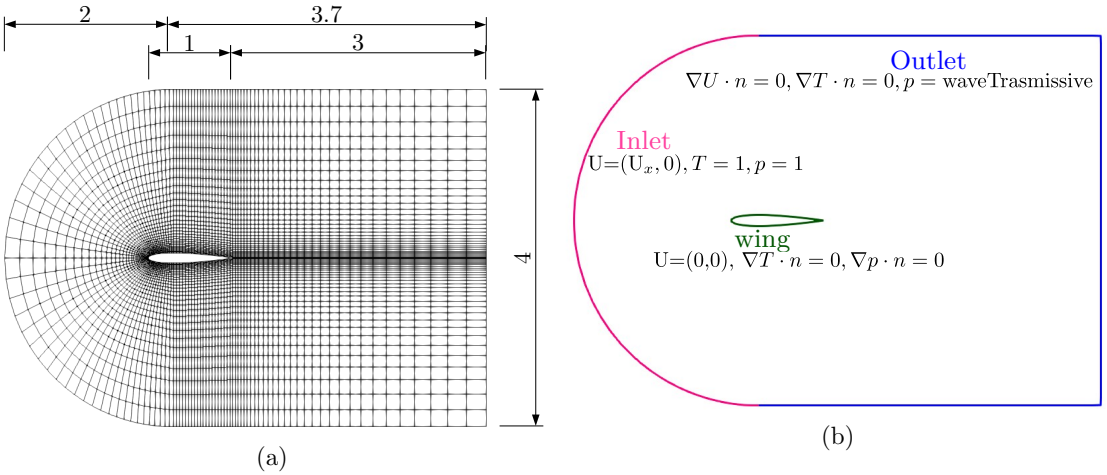


Figure 15: Computational domain of the airfoil CFD data set: (a) domain size and mesh grid structure, (b) domain patches and boundary conditions

by a factor of two, however, the error is of the order of 10^{-3} and so this is a relatively minor decrease when comparing the ROM results. We can see from fig. 18 that a large amount of the energy is captured in the first 5 modes of the POD decomposition in physical, Fourier and RCDT space. This supports the observations from the ROM results and errors, as adding modes to the ROM reconstructions beyond the first 5 modes will contribute relatively small amounts of information to the ROM and hence only improve results marginally when compared to the results from the first 5 modes.

4.3.1 Predictive Mach number interpolation

As mentioned in §4.2.1, we provide here a full RCDT-POD ROM interpolation case, using a linear interpolation study. However, in this case interpolation is performed with respect to the dimensionless Mach number, Ma, denoting the ratio of local flow velocity around the airfoil boundary to the speed of sound in said medium. The training dataset, of the original $N_t = 300$ (snapshot times) by $N_{\text{Ma}} = 13$ (Mach number values) CFD data snapshots provided, contains the two end-time snapshots for Ma $= 1.9, 3.1$. Like before, all transformation's, decomposition's, and interpolation's are performed using said training set. Decomposition via POD is performed using the top 10 modes, see fig. 18. It's worth noting that in this particular situation we use a signed variant of RCDT, applying separately the POD to the positive and negative parts of the inputted CFD data, and later re-combined. See [35] for further details on the transforming of signed measures in RCDT space. What's more, we use a cropped 160x100 uniform grid resolution of the original airfoil CFD data, removing the computational padding viewable in figs. 16 and 17, focusing on flow around the airfoil at hand. The chosen target snapshot is at Ma $= 2.5$, being somewhat midway to the two training snapshots, and part of the original CFD data for accurate comparison. Within fig. 19a are the three snapshots, training set and target, mentioned previously. The left and right are the training snapshots for Ma $= 1.9$ and 3.1, respectively, while the centre is our target snapshot for Ma $= 2.5$, which we stress is not part of the training. In fig. 19b is the target snapshot again (left), predicted solution of linear interpolation in physical POD space (middle) from the training shots, and absolute difference between target and prediction (right) in base $\log_{10}$, with the average L_2-norm error titled. While the break waves are somewhat predicted, the error shows clear failure in physical ROM to capture the wave magnitudes, despite a relatively low L_2 error. The snapshots of fig. 19c are the same procedure as in fig. 19b, this time instead performed within Fourier space. Likewise, both the average L_2-norm error and right-hand image paints a similar story to the physical case. Following the same workflow, now within RCDT-POD space, we get the results pictured in fig. 19e. Unlike the other spaces, we observe a qualitatively good comparison between the target snapshot and predicted solution, despite a quantitatively higher L_2-norm error than the previous counterparts. If we consider fig. 19e, which tests the accuracy (or intrinsic error) of RCDT at the training snapshot Ma $= 1.9$, as well as figs. 14a and 14b, the errors of fig. 19d can be viewed as a layering of both the intrinsic of RCDT, and the reconstructive error of POD.

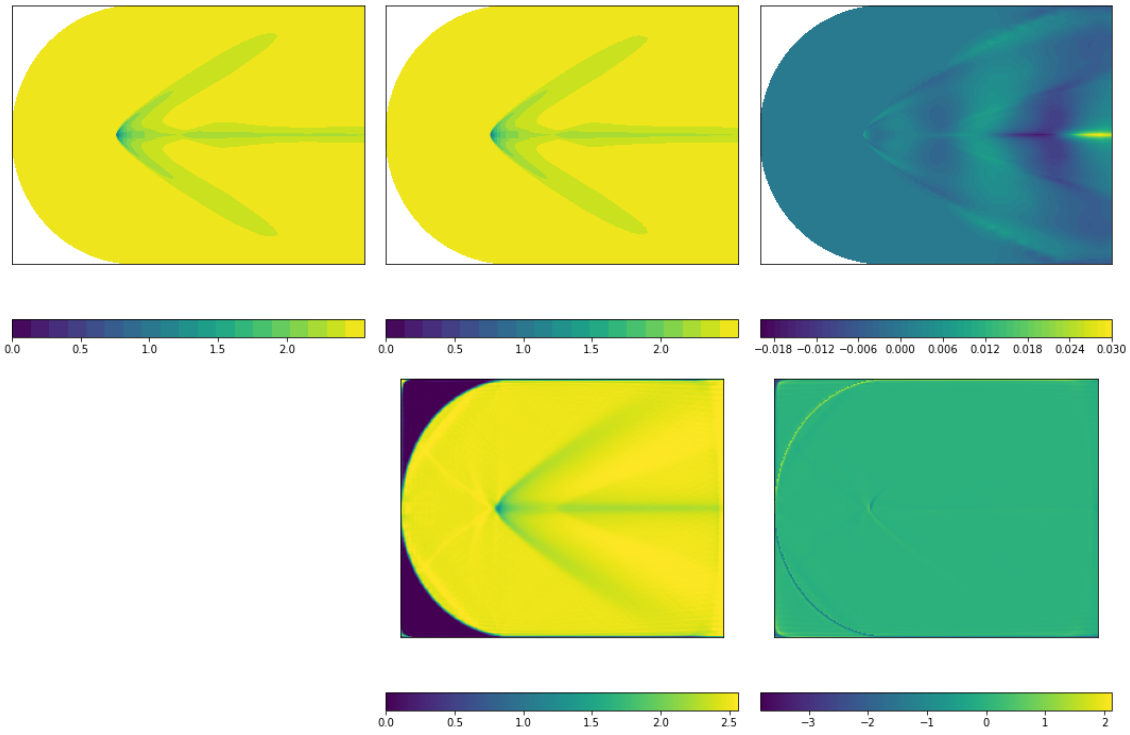


Figure 16: Single snapshot result from the airfoil POD (using top 5 modes). **Top-left:** original snapshot image. **Top-middle:** physical space POD projection. **Bottom-middle:** RCDT-POD space projection. **Top/bottom-right:** respective differences for each projection from the original snapshot image.

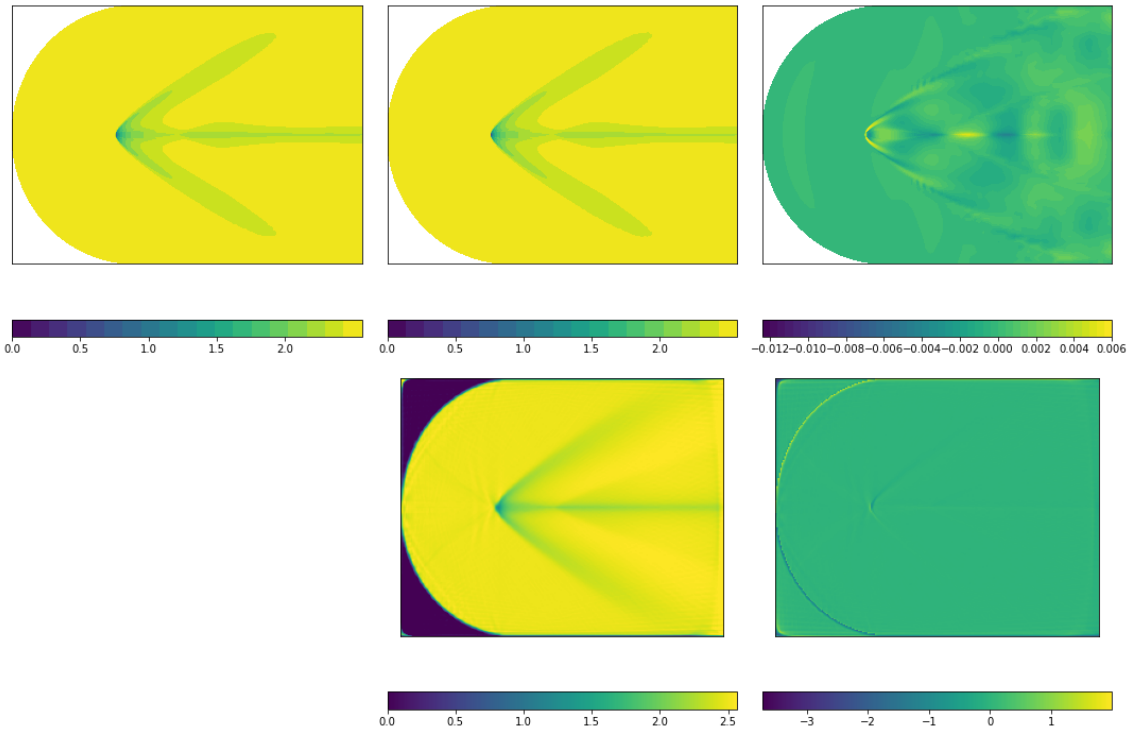


Figure 17: Single snapshot result from the airfoil POD (using top 20 modes). **Top-left:** original snapshot, top-middle: physical space POD projection. **Bottom-middle:** RCDT-POD space projection. **Top/bottom-right:** respective differences for each projection from the original snapshot image.

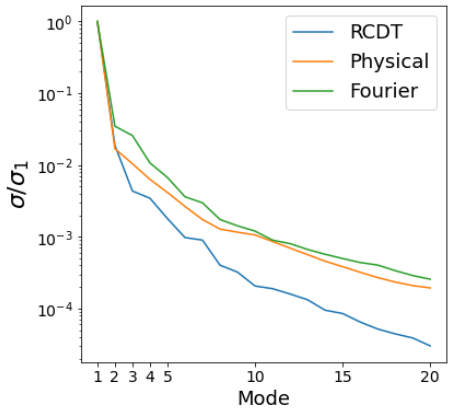


Figure 18: Plot of ratio of each singular value over the first singular value in POD decomposition for physical, Fourier and RCDT-POD space for the airfoil.

5 Conclusions

This work has focused on the implementation and verification of the Radon-Cumulative Distribution Transform (RCDT) for image and flow capture, as well assessing the transform's applicability to a Reduced Order Modelling (ROM) setting – under proper orthogonal decomposition (POD) – of high-fidelity CFD input data. Both RCDT and subsequent RCDT-POD ROM workflows were tested for accuracy; compared against either the original input image, or standard POD in physical space for ROMs' case. Based on the results in this work, we show that the RCDT transform may significantly reduce dimensionality and improve the interpolation in ROM with respect to real space POD/interpolation. Although the approach introduces additional artefacts and errors due to the numerical implementation of the forward and inverse transform, resulting in limited or no improvement in the error norms for some examples shown, we should note that this approach remains a promising prospect due to its unique properties and capabilities of capturing advection-dominated phenomena. The ability to capture and preserve geometrical features within complex CFD data sets is the most important novelty of this approach. For data sets that span differing configurations of the flow or image, see §3.3,4.2.1 and 4.3.1, this geometrical preservation is crucial; so far no other method can produce such flow preservation with ease.

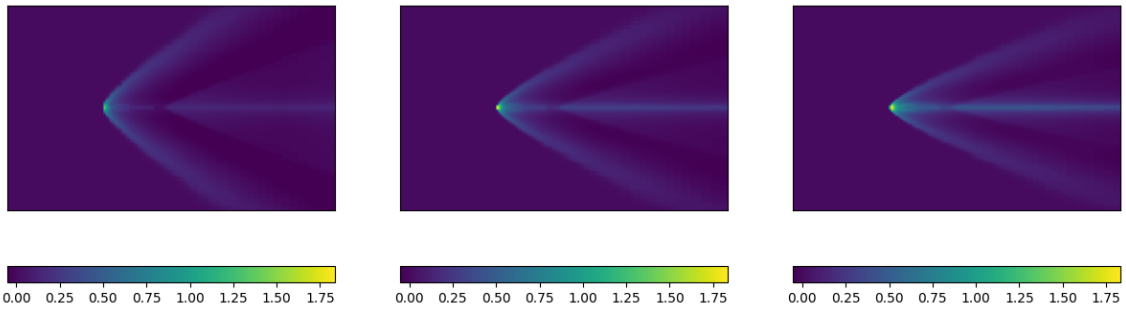
Although the results presented here are two-dimensional images in a one-dimensional parameter space, RCDT and its ROM application can be easily applied to higher-dimensional data sets. In future studies, the errors coming from the transform (i.e., intrinsic RCDT error), the POD truncation (reconstruction) and the interpolation will be further analysed separately to better explore the RCDT-POD ROM applicability when predicting advective-dominant flows of varying geometrical shapes, configurations, and speeds, and with more interpolating parameters. Other future works could involve improving RCDT's introduced for field data that does no go to zero at the boundary and pre-/post-processing of data to improve RCDT usage.

Acknowledgements

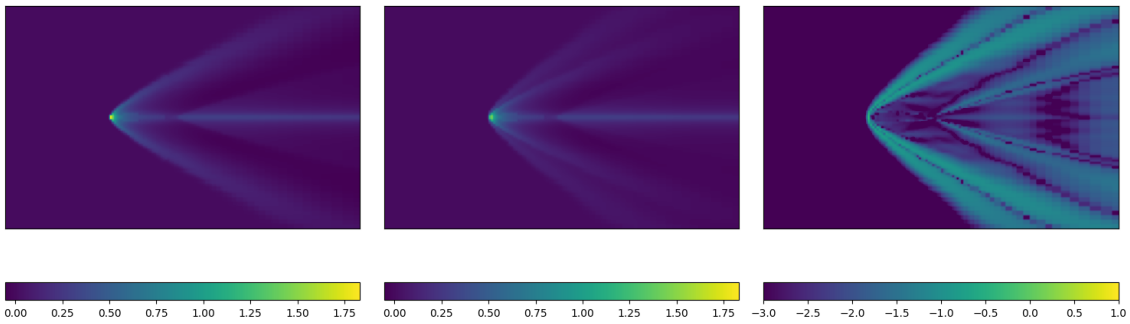
The authors thank Martina Cracco for her help in setting up the numerical algorithms. TL and RB have been funded by the European Union through the project SILENTPROP, "Assessing noise generation in aircraft with distributed electric propulsion" (Grant agreement ID 882842). The computations in this work have been performed with *PyTransKit* [28] and *EZyRB* [29]; we acknowledge developers and contributors to both libraries.

Statements and Declarations

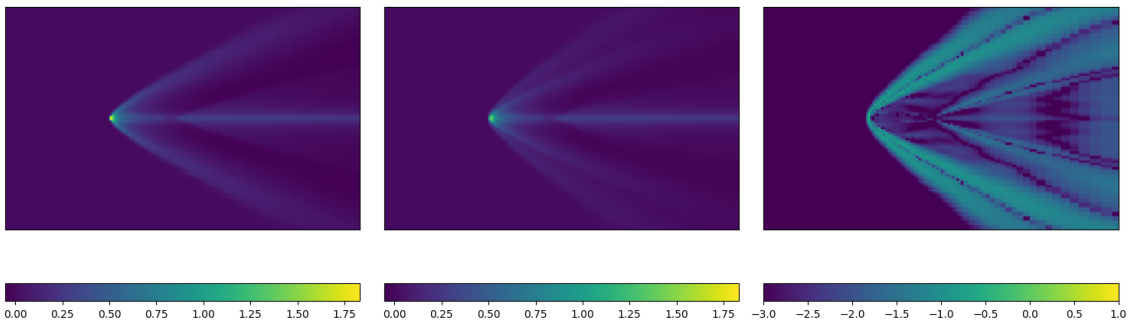
The authors declared that they have no conflict of interest.



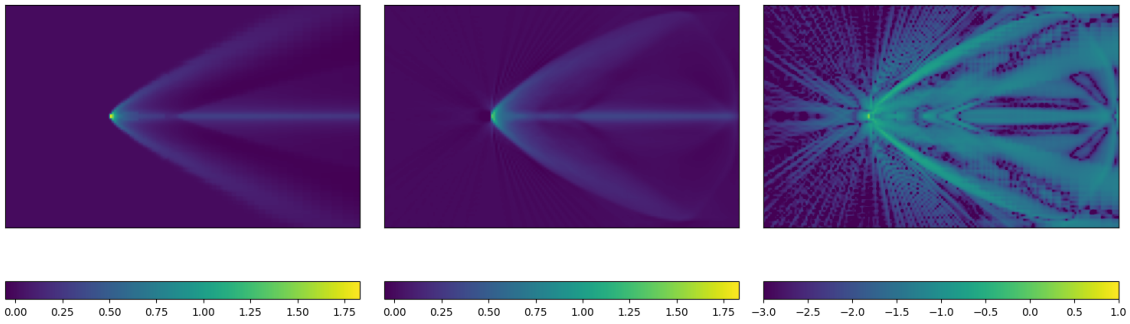
(a) Training and target snapshots. **Left:** training snapshot Ma = 1.9. **Centre:** target snapshot Ma = 2.5. **Right:** training snapshot Ma = 3.1.



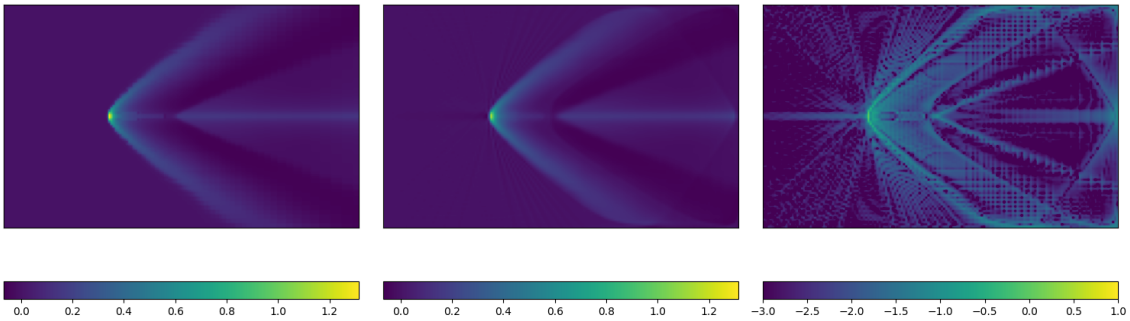
(b) Physical POD space interpolation results. **Left:** target snapshot. **Centre:** predicted solution of linear interpolation, in physical POD space. **Right:** absolute difference between target and result, given in base $\log_{10}$, alongside average L_2-norm error.



(c) Fast Fourier transform (FFT) POD space interpolation results. **Left:** target snapshot. **Centre:** predicted solution of linear interpolation, in FFT POD space, inverted back to physical. **Right:** absolute difference between target and result, given in base $\log_{10}$, alongside average L_2-norm error.



(d) RCDT-POD ROM interpolation results. **Left:** target snapshot. **Centre:** predicted solution of linear interpolation, in RCDT-POD space, inverted back to physical for comparison. **Right:** absolute difference between target and interpolation result, given in base $\log_{10}$, alongside average L_2-norm error.



(e) Accuracy test of RCDT's transformed snapshot from the original input snapshot data. **Left:** training snapshot Ma = 1.9. **Centre:** resultant image of RCDT, followed by its inverse back to physical space, at the given snapshot. **Right:** difference between training snapshot and RCDT to iRCDT snapshot, given in base $\log_{10}$, alongside the average L_2-norm error.

Figure 19: Prediction of airfoil CFD snapshot data, with respect to the Mach number, using reduced order modelling (ROM) and linear interpolation. Interpolation in physical and Fourier space given as comparison against RCDT-POD space.

References

[1] American Institute of Aeronautics and Astronautics, editor. *14th AIAA Aviation Technology, Integration and Operations Conference 2014: Atlanta, Georgia, USA, 16 - 20 June 2014 ; held at the AIAA Aviation Forum 2014*. Curran, Red Hook, NY, 2014.

[2] Hyun D. Kim, Aaron T. Perry, and Phillip J. Ansell. A Review of Distributed Electric Propulsion Concepts for Air Vehicle Technology. In *2018 AIAA/IEEE Electric Aircraft Technologies Symposium*, Cincinnati, Ohio, July 2018. American Institute of Aeronautics and Astronautics.

[3] Amir S. Gohardani. A synergistic glance at the prospects of distributed propulsion technology and the electric aircraft concept for future unmanned air vehicles and commercial/military aviation. *Progress in Aerospace Sciences*, 57:25–70, February 2013.

[4] Ted Ellif, Michele Cremasci, and Violaine Huck-Envisa. Impact of aircraft noise pollution on residents of large cities. *European Parliament*, page 62, 2020.

[5] Kenneth S. Brentner, Bolor Erdene Zolbayar, and Thomas F. Jaworski. An investigation of noise from electric, low-tip-speed aircraft propellers. 2018. AHS International Technical Meeting on Aeromechanics Design for Transformative Vertical Flight 2018 ; Conference date: 16-01-2018 Through 18-01-2018.

[6] Ramesh Narayan and Insu Yi. Advection-dominated Accretion: A Self-similar Solution. *The Astrophysical Journal*, 428:L13, June 1994.

[7] Rohan Mahadevan. Scaling Laws for Advection-dominated Flows: Applications to Low-Luminosity Galactic Nuclei. *The Astrophysical Journal*, 477(2):585, mar 1997.

[8] Kenneth S. Brentner and F. Farassat. Modeling aerodynamically generated sound of helicopter rotors. *Progress in Aerospace Sciences*, 39(2):83–120, February 2003.

[9] Matthew Churchfield, Scott Schreck, Luis A. Martínez-Tossas, Charles Meneveau, and P. R. Spalart. An Advanced Actuator Line Method for Wind Energy Applications and Beyond: Preprint. In *35th Wind Energy Symposium*, 2017.

[10] Lawrence Sirovich. Turbulence and the dynamics of coherent structures part I: Coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, 1987. Publisher: Brown University.

[11] J. E. Ffowcs Williams, D. L. Hawkings, and Michael James Lighthill. Sound generation by turbulence and surfaces in arbitrary motion. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 264(1151):321–342, May 1969. Publisher: Royal Society.

[12] Steven L. Brunton and Bernd R. Noack. Closed-Loop Turbulence Control: Progress and Challenges. *Applied Mechanics Reviews*, 67(5):050801, September 2015.

[13] Gianluigi Rozza, Giovanni Stabile, and Francesco Ballarin, editors. *Advanced Reduced Order Methods and Applications in Computational Fluid Dynamics*. Society for Industrial and Applied Mathematics, 2022.

[14] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced Basis Methods for Partial Differential Equations*. Springer International Publishing, 2016.

[15] Shivam Setia, Virendra S Kuwar, Prashant R Pawar, and Medha Santosh Jambhale. Model Order Reduction Technique to Aid Control System Design. In *Symposium on International Automotive Technology*. SAE International, September 2021. ISSN: 0148-7191.

[16] Xiang Sun and Jung-Il Choi. Non-intrusive reduced-order modeling for uncertainty quantification of space–time-dependent parameterized problems. *Computers & Mathematics with Applications*, 87:50–64, April 2021.

[17] Carsten Hartmann, Juan C. Latorre, Wei Zhang, and Grigorios A. Pavliotis. Optimal control of multiscale systems using reduced-order models. *Journal of Computational Dynamics*, 1(2):279–306, November 2014.

[18] Mario Ohlberger and Stephan Rave. Reduced Basis Methods: Success, Limitations and Future Challenges. *Proceedings of the Conference Algoritmy*, pages 1–12, 2016.

[19] Constantin Greif and Karsten Urban. Decay of the Kolmogorov n-width for wave problems. *Applied Mathematics Letters*, 96:216–222, 2019.

[20] Benjamin Peherstorfer. Model Reduction for Transport-Dominated Problems via Online Adaptive Bases and Adaptive Sampling. *SIAM Journal on Scientific Computing*, 42(5):A2803–A2836, 2020.

[21] Kookjin Lee and Kevin T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, March 2020.

[22] Francesco Romor, Giovanni Stabile, and Gianluigi Rozza. Non-linear manifold ROM with Convolutional Autoencoders and Reduced Over-Collocation method. *Journal of Scientific Computing*, 94(3), 2023.

[23] D. Amsallem, M. J. Zahr, and C. Farhat. Nonlinear model order reduction based on local reduced-order bases. *Int. J. Num. Meth. Eng.*, 92(10):891–916, 2012.

[24] J. Reiss, P. Schulze, J. Sesterhenn, and V. Mehrmann. The Shifted Proper Orthogonal Decomposition: A Mode Decomposition for Multiple Transport Phenomena. *SIAM Journal on Scientific Computing*, 40(3):A1322–A1344, January 2018.

[25] F. Bernard, A. Iollo, and S. Riffaud. Reduced-order model for the BGK equation based on POD and optimal transport. *J. Comput. Phys.*, 373:545–570, 2018.

[26] Davide Papapicco, Nicola Demo, Michele Girfoglio, Giovanni Stabile, and Gianluigi Rozza. The Neural Network shifted-proper orthogonal decomposition: A machine learning approach for non-linear reduction of hyperbolic equations. *Computer Methods in Applied Mechanics and Engineering*, 392:114687, March 2022.

[27] Se Rim Park, Soheil Kolouri, Shinjini Kundu, and Gustavo K. Rohde. The cumulative distribution transform and linear pattern classification. *Applied and Computational Harmonic Analysis*, 45(3):616–641, November 2018.

[28] Abu Hasnat Mohammad Rubaiyat, Mohammad Shifat E Rabbi, Liam Cattell, Xuwang Yin, Shiying Li, Yan Zhuang, Gustavo K. Rohde, Soheil Kolouri, and Serim Park. PyTransKit, September 2022. original-date: 2019-06-12T20:43:51Z.

[29] Nicola Demo, Marco Tezzele, and Gianluigi Rozza. EZyRB: Easy Reduced Basis method. *The Journal of Open Source Software*, 3(24):661, 2018.

[30] Nicola Demo, Marco Tezzele, Andrea Mola, and Gianluigi Rozza. A complete data-driven framework for the efficient solution of parametric shape design and optimisation in naval engineering problems. In *Proceedings of MARINE 2019: VIII International Conference on Computational Methods in Marine Engineering*, pages 111–121, 2019.

[31] Marco Tezzele, Nicola Demo, and Gianluigi Rozza. Shape optimization through proper orthogonal decomposition with interpolation and dynamic mode decomposition enhanced by active subspaces. In *Proceedings of MARINE 2019: VIII International Conference on Computational Methods in Marine Engineering*, pages 122–133, 2019.

[32] Zhu Wang, Brian McBee, and Traian Iliescu. Approximate partitioned method of snapshots for POD. *Journal of Computational and Applied Mathematics*, 307:374–384, 2016.

[33] Stanley R. Deans. *The Radon Transform And Some Of Its Applications*. John Wiley & Sons, Inc., 1983.

[34] Frank Natterer. *The mathematics of computerized tomography*. Society for Industrial and Applied Mathematics, 2001.

[35] Akram Aldroubi, Rocio Diaz Martin, Ivan Medri, Gustavo K. Rohde, and Sumati Thareja. The Signed Cumulative Distribution Transform for 1-D signal analysis and classification. *Foundations of Data Science*, 4(1):137, 2022. Publisher: American Institute of Mathematical Sciences.

[36] Soheil Kolouri, Se Rim Park, Matthew Thorpe, Dejan Slepcev, and Gustavo K. Rohde. Optimal Mass Transport: Signal processing and machine-learning applications. *IEEE Signal Processing Magazine*, 34(4):43–59, 2017.

[37] Soheil Kolouri, Se Rim Park, and Gustavo K. Rohde. The Radon cumulative distribution transform and its application to image classification. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 25(2):920–934, February 2016.

[38] Mohammad Shifat-E-Rabbi, Xuwang Yin, Abu Hasnat Mohammad Rubaiyat, Shiying Li, Soheil Kolouri, Akram Aldroubi, Jonathan M. Nichols, and Gustavo K. Rohde. Radon Cumulative Distribution Transform Subspace Modeling for Image Classification. *Journal of Mathematical Imaging and Vision*, 63(9):1185–1203, November 2021.

[39] Jie Ren, William R. Wolf, and Xuerui Mao. Model reduction of traveling-wave problems via Radon cumulative distribution transform. *Physical Review Fluids*, 6(8):L082501, August 2021. Publisher: American Physical Society.

[40] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, November 1901. Publisher: Taylor & Francis.

[41] J. L. Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric Turbulence and Radio Wave Propagation*, pages 166–178, 1967.

[42] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, April 2016. Publisher: Royal Society.

[43] H. Cho, D. Venturi, and G. E. Karniadakis. Karhunen–Loève expansion for multi-correlated stochastic processes. *Probabilistic Engineering Mechanics*, 34:157–167, October 2013.

[44] Giovanni Stabile, Francesco Ballarin, Giacomo Zuccarino, and Gianluigi Rozza. A reduced order variational multiscale approach for turbulent flows. *Advances in Computational Mathematics*, 45(5-6):2349–2368, 2019.

[45] Saray Busto, Giovanni Stabile, Gianluigi Rozza, and Maria Vázquez-Cendón. POD Galerkin reduced order methods for combined navier-stokes transport equations based on a hybrid FV-FE solver. *Computers & Mathematics with Applications*, 2019.

[46] Marco Tezzele, Nicola Demo, Giovanni Stabile, and Gianluigi Rozza. Chapter 9: Nonintrusive Data-Driven Reduced Order Models in Computational Fluid Dynamics. In *Advanced Reduced Order Methods and Applications in Computational Fluid Dynamics*, pages 203–222. Society for Industrial and Applied Mathematics, 2022.

[47] Giovanni Stabile, Matteo Zancanaro, and Gianluigi Rozza. Efficient Geometrical parametrization for finite-volume based reduced order methods. *International Journal for Numerical Methods in Engineering*, 121(12):2655–2682, 2020.

[48] N.C. Nguyen and J. Peraire. Gaussian functional regression for linear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 287:69–89, 2015.

[49] N.C. Nguyen and J. Peraire. Gaussian functional regression for output prediction: Model assimilation and experimental design. *Journal of Computational Physics*, 309:52–68, 2016.

[50] Hrvoje Jasak. *Error analysis and estimation for the finite volume method with applications to fluid flows*. PhD thesis, Imperial College, University of London, 1996.

3.1 Contributions to published work

The majority of the work displayed in this paper was carried out and written up by the candidate, Tobias Long. The exceptions to this are the work carried out by Robert Barnett in §4.2 and §4.3.1, including Figures 10, 14 and 15. The work was carried out with the aid of all authors listed, including editing and corrections to the written document.

This page marks the end of the work submitted for publication.

4 CFD Work

4.1 Propeller Modelling

Modelling propellers or turbines in CFD simulations can be done using a few different methods. These methods vary in complexity and hence the accuracy of results, and the user must decide on a method based on minimising resources needed for the model whilst maximising the accuracy of the results. For example, running a high-fidelity simulation where the propeller blades are fully modelled is very accurate but extremely complex, resulting in a huge amount of processing power needed for the simulation. In contrast, an actuator disk model (ADM) involves only a source of momentum being applied to a small volume of the domain, which requires very few resources but captures none of the behaviour of the propeller blades. Here each method will be discussed briefly.

4.1.1 Actuator Disk Method (ADM)

The ADM simulates a rotating propeller by projecting a force onto the fluid within the disc-shaped area spanned by the blades. Within this disc, actuator points are defined where the force is calculated from the fluid speed passing through the point, and then this force is projected onto the fluid as it passes through the point.

The force exerted by the propeller being modelled, with a radius of R and a streamwise disk thickness of Δz is given by

$$\mathbf{f} = -\frac{1}{\pi R^2 \Delta z^2} \left[\int_\Omega (\mathbf{s} \cdot \mathbf{u}) \cdot (\mathbf{u} \cdot \mathbf{s}) \, \mathrm{d}\Omega \right] \mathbf{s}, \tag{7}$$

where **s** is the streamwise direction unit vector and **u** is the velocity of the fluid at the disk. This is a very simple method for estimating the time-averaged force created by a propeller, requiring very little computational resources, at the expense of being a crude estimation of the force without any time-dependent flow features. For an in-depth description refer to [11].

4.1.2 Actuator Line Method (ALM)

The ALM simulates a rotating propeller using a series of actuator points (20-50 points typically) along the length of each blade. The force at each actuator point is then calculated in a lift and drag component, $\mathbf{f}_{N,m} = (L_{N,m}, D_{N,m})^T$, where the indices N and m represent the blade index and the actuator point index along each blade respectively. This is done by noting the local flow velocity and the angle of attack of the blade at a given actuator point, which are then applied to an airfoil lookup table, allowing the drag and lift forces to be calculated. These lift and drag forces are then projected back into the domain mesh using the body force term of the momentum equation. The force itself must be distributed over multiple mesh cells in order to avoid numerical instabilities. This is done with the use of a Gaussian kernel function at each actuator point location, this function is defined as follows

$$\eta(x, y, z) = \frac{1}{\varepsilon^3 \pi^{3/2}} \exp\left(-\frac{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}{\varepsilon^2}\right) \tag{8}$$

,

where (x_0, y_0, z_0) is the location for the function to be applied to (ie. each actuator

point N, m), and ε is the Gaussian width parameter (recommended to be twice the size of the mesh resolution). The force is applied to the domain mesh as a force vector $\mathbf{F}_p$, which is given by

$$\mathbf{F}_p(x_p, y_p, z_p, t) = -\sum_N \sum_m \mathbf{f}_{N,m}(x_{N,m}, y_{N,m}, z_{N,m}, t)\eta_{N,m} \tag{9}$$

,

where p denotes all grid mesh points that are affected by all actuator points (N, m) and $\mathbf{f}_{N,m}$ are the force vectors at each of the actuator points. As the ALM is not used in this work in favour of the advanced method to follow, this description is adequate for our purposes. For a more comprehensive coverage of the methodology of the ALM please refer to [12, 13].

4.1.3 Advanced Actuator Line Method (AALM)

The Advanced ALM is an advancement upon the base ALM theory and uses both a more realistic, blade-shaped projection function for the body forces and a more robust velocity sampling algorithm around the blades. This results in a more accurate calculation and projection of the blade forces onto the domain mesh. A brief description of this method is offered here as the full extent of the methodology and calculation of parameters is extensive. A full discussion of the method and the calculation of the parameters can be found in the original work [14].

The modified projection function, η_{AAL}, is defined as

$$\eta_{AAL}(x_c, x_t, x_r) = \frac{1}{\varepsilon_c \varepsilon_t \varepsilon_r \pi^{3/2}} \exp\left(- \frac{(x_c - x_{c,0})^2}{\varepsilon_c^2} - \frac{(x_t - x_{t,0})^2}{\varepsilon_t^2} - \frac{(x_r - x_{r,0})^2}{\varepsilon_r^2} \right) \quad (10)$$

,

where (x_c, x_t, x_r) are the coordinates of a point in the chord-wise, thickness-wise and radial-wise directions respectively. The subscript 0 denotes the location to which the function is applied and each coordinate direction has its own Gaussian width parameter $(\varepsilon_c, \varepsilon_t, \varepsilon_r)$.

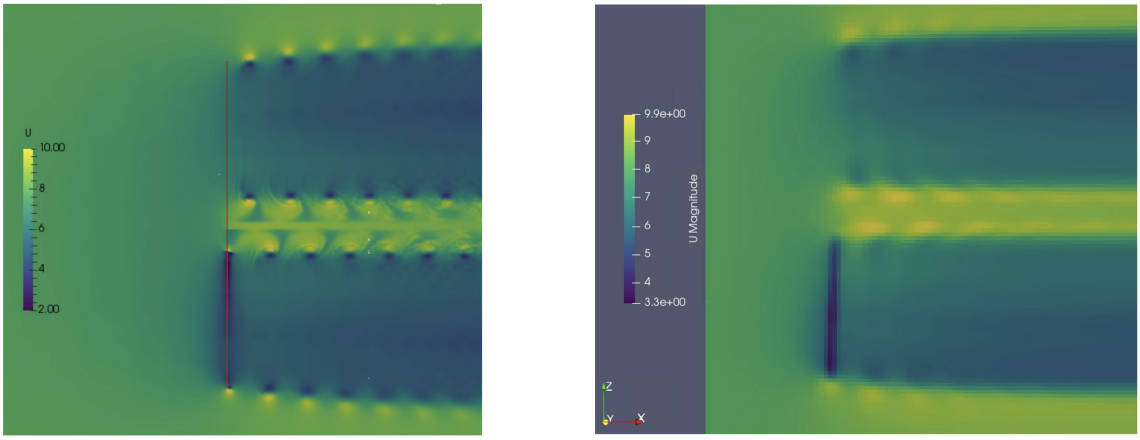
The velocity sampling used in the AALM is also different from the one used in the ALM. In the ALM, the velocity used for the airfoil look-up table is simply sampled from the centre of the actuator points in the mesh. The velocity is interpolated from the freestream velocity values in the surrounding mesh cells. This sampling method works well when used with a symmetric projection function such as the Gaussian function used in the ALM, but is not suitable for non-symmetric projection functions such as the one defined in (10). Instead, for the AALM a new velocity sampling method is required. The AALM uses integral velocity sampling where an integral weighted by the projection function, η, is used for calculating the velocity at the actuator points.

4.2 Code

4.2.1 Adapted SOWFA Code

The ALM used in this work has been adapted from the *SOWFA* (Simulator fOr Wind Farm Applications) package, developed by the National Renewable Energy Laboratory (NREL) [15]. Specifically, the code from the development branch of *SOWFA-6* was used for this adaptation, as this code is the most up-to-date version in development (in this case for *OpenFOAM v6*) [16]. The code produced by NREL is specifically for wind turbine simulations and requires minor modifications to the solver and the source files required for the ALM. The *horizontalAxisWindTurbinesALMAdvanced* model was chosen for adaptation as this was the source code used for simulations with the advanced ALM, which is the more accurate ALM as discussed in §4.1.3. A majority of the functions in the Advanced ALM code produced by NREL are wind turbine specific (such as controllers for RPM, torque and generators) and were removed to simplify the code, as all that is needed for this work is the ALM itself and the ability to set the RPM of the propeller(s). This new model is named *propALMAdvanced*. The solver was created by modifying the source code for the *pimpleFoam* solver adding the turbine objects to the source file, and the addition of a source term to the equations of motion. The new solver is named *pimpleALMFoam*.

Preliminary testing was carried out on a coarse mesh to compare the adapted AALM code to the original code from SOWFA. The results of this comparison can be seen in Figure 12. This test was carried out on a coarse mesh relative to the case found in the original literature in order to initially validate the algorithm and



(a) (b)

Figure 12: Comparison of AALM code outputs along y-axis slice for (a) original SOWFA AALM code [14] and (b) our adapted code described here.

check that the results generated were as expected to rule out large errors in the calculations. As seen in Figure 12 the results resemble the overall structure and amplitudes seen in the original test case, confirming the algorithm was working as expected, allowing work to proceed on validating the results on a finer mesh. Due to time shortage at this stage of the work, the results on the fine mesh have not yet been carried out and validated. This is reserved for future work, in order to confirm the algorithm is sufficiently accurate to produce the data needed for the reduced order modelling of DEP layouts. Despite this, confidence is high that this will be a trivial result due to the resemblance shown in the coarse mesh test.

4.2.2 Code Parameters

The polar file input for the *CLARK Y* airfoil was created using *JavaFoil*. The data points for plotting the airfoil shape were obtained from the website *Airfoil Tools* [3] and *JavaFoil* was used to simulate polar values for angles ±150 degrees. Blade geometry

[3] http://airfoiltools.com/airfoil/details?airfoil=clarky-il

data was obtained from the propeller geometry CAD file for a 9-inch diameter test case propeller. Seven separate planes along the radius of the propeller blade were used to extract blade geometry data for input to the ALM code. The propeller blade geometry is relatively uniform so a large number of planes were not necessary as interpolation between planes would generate sufficient accuracy for the blade geometry. The planes are shown in Fig. 13

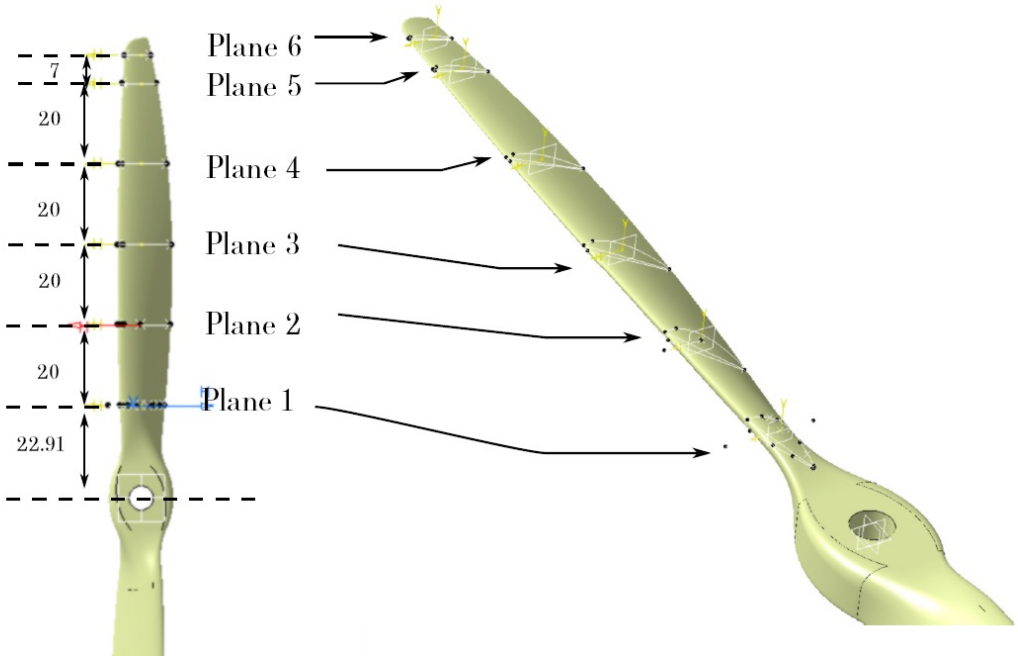


Figure 13: Planes used for extracting blade geometry data. Measurements shown are in mm.

5 Concluding Remarks

The aim of this work is to make the first steps in utilising a novel data transform, the Radon-CDT, to produce an efficient reduced order model methodology for quick and accurate estimation of single and multiple propeller configurations. This is a key step in the estimation and evaluation of noise generation in distributed electric propulsion (DEP) applications and beyond. The work presented in this thesis lays out the starting investigations and progress towards combining the Radon-CDT with a reduced order modelling workflow in order to accurately and swiftly estimate such propeller flows. Initial work has been carried out in order to better understand the behaviour and the computational implementation of the Radon-CDT on relevant test data, confirming the applicability of the transform for use in the context of propeller CFD data. The transform was then implemented and tested in a reduced order modelling workflow for various CFD test cases, where it was shown to be a worthwhile avenue of further research and use for the application of reduced order modelling for distributed electric propulsion layouts. Further to this, an algorithm for a suitable propeller simulation with sufficient accuracy and computational efficiency was developed, using previous work on an advanced actuator line model from the SOWFA laboratory. This algorithm can be used in future work in order to provide the outlined reduced-order model method with a large number of high-fidelity DEP layout data sets.

With the groundwork presented here, there is the promising possibility of developing a full ROM workflow incorporating the Radon-CDT for accurate DEP con-

figuration flow estimations, and hence noise prediction. Pre- and post-processing of the CFD data in an appropriate manner is a further step required to fully utilise the properties of the Radon-CDT in this context. This processing will allow errors from specific boundary effects to be minimised and hence will allow the Radon-CDT workflow to be developed beyond the current capabilities.

References

[1] A. I. of Aeronautics and Astronautics, ed., *14th AIAA Aviation Technology, Integration and Operations Conference 2014: Atlanta, Georgia, USA, 16 - 20 June 2014 ; held at the AIAA Aviation Forum 2014*. Red Hook, NY: Curran, 2014. Meeting Name: AIAA Aviation Technology, Integration, and Operations Conference Publication Title: 14th AIAA Aviation Technology, Integration and Operations Conference 2014 : Atlanta, Georgia, USA, 16 - 20 June 2014; held at the AIAA Aviation Forum 2014 ; Vol. 1.

[2] H. D. Kim, A. T. Perry, and P. J. Ansell, "A Review of Distributed Electric Propulsion Concepts for Air Vehicle Technology," in *2018 AIAA/IEEE Electric Aircraft Technologies Symposium*, (Cincinnati, Ohio), American Institute of Aeronautics and Astronautics, July 2018.

[3] A. S. Gohardani, "A synergistic glance at the prospects of distributed propulsion technology and the electric aircraft concept for future unmanned air vehicles and commercial/military aviation," *Progress in Aerospace Sciences*, vol. 57, pp. 25–70, Feb. 2013.

[4] T. Ellif, M. Cremasci, and V. Huck-Envisa, "Impact of aircraft noise pollution on residents of large cities," *European Parliment*, p. 62, 2020.

[5] K. S. Brentner, B. E. Zolbayar, and T. F. Jaworski, "An investigation of noise from electric, low-tip-speed aircraft propellers," in *AHS International Techni-*

cal Meeting on Aeromechanics Design for Transformative Vertical Flight 2018, 2018.

[6] U. of Nottingham, "Silentprop project webpage." `https://www.nottingham.ac.uk/aerospace/projects/cleansky/silentprop-project.aspx`. Accessed: 2023-06-01.

[7] C. Sky, "Clean sky website." `https://www.cleansky.eu/`. Accessed: 2023-06-01.

[8] S. Kolouri, S. R. Park, and G. K. Rohde, "The Radon cumulative distribution transform and its application to image classification," *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 25, pp. 920–934, Feb. 2016.

[9] S. R. Park, S. Kolouri, S. Kundu, and G. K. Rohde, "The cumulative distribution transform and linear pattern classification," *Applied and Computational Harmonic Analysis*, vol. 45, pp. 616–641, Nov. 2018.

[10] T. Long, R. Barnett, R. Jefferson-Loveday, G. Stabile, and M. Icardi, "A novel reduced-order model for advection-dominated problems based on radon-cumulative-distribution transform," 2023.

[11] M. Tzimas and J. Prospathopoulos, "Wind turbine rotor simulation using the actuator disk and actuator line methods," *Journal of Physics: Conference Series*, vol. 753, p. 032056, sep 2016.

[12] J. N. Sorensen and W. Z. Shen, "Numerical modeling of wind turbine wakes," *Journal of Fluids Engineering*, vol. 124, pp. 393–399, 05 2002.

[13] M. Ravensbergen, A. Bayram Mohamed, and A. Korobenko, "The actuator line method for wind turbine modelling applied in a variational multiscale framework," *Computers & Fluids*, vol. 201, p. 104465, 2020.

[14] M. Churchfield, S. Schreck, L. A. Martínez-Tossas, C. Meneveau, and P. R. Spalart, "An Advanced Actuator Line Method for Wind Energy Applications and Beyond: Preprint," 2017. Book Title: An Advanced Actuator Line Method for Wind Energy Applications and Beyond: Preprint.

[15] NREL, "Sowfa: Simulator for wind farm applications — wind research — nrel." `https://www.nrel.gov/wind/nwtc/sowfa.html`. Accessed: 2023-06-01.

[16] NREL, "Github - nrel/sowfa-6 at dev." `https://github.com/NREL/SOWFA-6/tree/dev`. Accessed: 2023-06-01.