Master Thesis

Multipurpose synthetic particle image velocimetry

Author(s):
Küchlin, Stephan

Publication Date:
2013

Permanent Link:
https://doi.org/10.3929/ethz-a-010075288

Rights / License:
In Copyright - Non-Commercial Use Permitted
Multipurpose Synthetic Particle Image Velocimetry

Stephan Küchlin

Studiengang Maschineningenieurwissenschaften Master

Masterarbeit FS 2013

Institut für Fluyddynamik
ETH Zürich

Erarbeitet am
Fluid Mechanics Laboratory
NASA Ames Research Center
Moffett Field, CA (USA)

Betreuer: James Ross (NASA Ames)
James Heineck (NASA Ames)
Scott Murman (NASA Ames)
Manikandan Ramasamy (NASA Ames)

Professor: Prof. Dr. Patrick Jenny
Abstract

Particle Image Velocimetry (PIV) has rapidly evolved into the method of choice for obtaining instantaneous, non-intrusive, three-vector-of-velocity field data from flow experiments, from lab-to full-scale dimensions. The technique uses high-speed cameras to record images of tracer particles in a flow at known time instances in quick succession. Using cross-correlation techniques, the fluid velocity may be subsequently inferred. Because it is nearly impossible to generate a real-world flow with perfectly known velocity profile, the identification and quantification of uncertainties in the technique is challenging.

To provide known baseline cases, to isolate the influence of different real-world effects and to improve the experimental setup, a tool to generate synthetic PIV from either analytical flow fields or Computational Fluid Dynamics (CFD) results was developed and implemented. The tool will also be useful in providing a processing-independent comparison between experimental results and CFD by allowing for the computational results to be processed in exactly the same way as the experimental data. This will allow for a detailed PIV uncertainty quantification and support CFD validation via PIV. For rotorcraft research at the National Aeronautics and Space Administration (NASA) Ames Research Center (ARC), the new tool will be used to generate reference images of analytically specified rotor blade wakes to improve vortex identification models.
I would like to thank Professor Patrick Jenny (ETH) and James Ross (ARC-AOX) for providing me with the “once-in-a-lifetime” opportunity to write my Master’s thesis at the Fluid Mechanics Laboratory at the NASA Ames Research Center.

This work would not have been possible without the help, advice and data I received from James T. Heineck (ARC-AOX), Scott Murman (ARC-TNF) and Manikandan Ramasamy (ARC-YA). The amount of time they were willing to invest in this project is gratefully acknowledged. Their support made working and learning at NASA a great pleasure.

I would like to thank everyone who made these six months a universally positive experience.

Stephan Kuechlin, September 2013
Contents

Abstract iii

Acknowledgements v

Acronyms ix

1. Introduction 1

2. Particle Image Velocimetry (PIV) 3
   2.1. Setup: Angular Stereoscopic PIV 3
   2.2. PIV Processing 4
      2.2.1. Determining the Particle Image Displacement - Image Interrogation 5
   2.3. Accuracy and Uncertainty in PIV 6
   2.4. Synthetic PIV 8
      2.4.1. The Synthetic Image Generator (SIG) 9

3. The CFD2PIV Tool 12
   3.1. Idea and Design Goals 12
   3.2. Basic Principle 12
   3.3. Synthetic PIV from CFD 13
      3.3.1. Murman’s Algorithm with Optimized Tree Look-Up 14
   3.4. Synthetic PIV from an Analytical Flow Field 16
   3.5. Time Integration 18
   3.6. Seeding Non-Uniformity Simulation 18
   3.7. Calibration Target 19
   3.8. Field of View Matching 20
   3.9. Outline of the CFD2PIV Program 20
      3.9.1. Input, Configuration and Parameterization 20
      3.9.2. Data Structures and Program Control Flow 20
   3.10. From CFD to PIV Using the CFD2PIV Tool 23

4. Test Case A: Synthetic PIV from Coupled Unstructured, Unsteady RANS/Adaptive Cartesian Euler CFD 25
## Contents

5. Test Case B: Synthetic PIV from Analytical Flow Fields in Support of Rotor Tip-Vortex Identification 29  
  5.1. Synthetic PIV Configuration ........................................... 29  
  5.2. Vortex Identification Model Testing ............................... 29  
      5.2.1. Results .......................................................... 37  

6. Test Case C: Analysis of Synthetic PIV from SST-DES of a Cylinder Wake with a Comparison to Experimental PIV Data 41  
  6.1. Experimental Data ....................................................... 41  
  6.2. CFD Data ................................................................. 41  
  6.3. Synthetic PIV ............................................................ 43  
      6.3.1. Image Generation .................................................. 44  
      6.3.2. PIV Processing .................................................... 44  
  6.4. Temporal Averages ..................................................... 44  
      6.4.1. Oversampling via Re-Initialization .............................. 46  
      6.4.2. Comparison of Experimental, CFD and Synthetic Temporal Averages .......................... 46  
      6.4.3. Influence of the Interrogation Window Size ................... 49  
  6.5. Spatial Spectra .......................................................... 52  
  6.6. Velocity PDF Estimates ............................................... 57  
  6.7. Influence of Seeding Non-Uniformity on PIV Performance ........ 58  

7. Conclusion ................................................................. 62  

A. Source Code Listings ...................................................... 63  
  A.1. Bounding Box Construction for the Convex Hull Algorithm .......... 63  
  A.2. N-linear Interpolation via Hypercube Reduction .................... 64  

References ................................................................. 67
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>ARC</td>
<td>Ames Research Center</td>
</tr>
<tr>
<td>AFDD</td>
<td>Aeroflightdynamics Directorate</td>
</tr>
<tr>
<td>AMR</td>
<td>Adaptive Mesh Refinement</td>
</tr>
<tr>
<td>CCD</td>
<td>Charge-Coupled Device</td>
</tr>
<tr>
<td>CFD</td>
<td>Computational Fluid Dynamics</td>
</tr>
<tr>
<td>DES</td>
<td>Detached Eddy Simulation</td>
</tr>
<tr>
<td>DNS</td>
<td>Direct Numerical Simulation</td>
</tr>
<tr>
<td>DSR</td>
<td>Dynamic Spatial Range</td>
</tr>
<tr>
<td>DVR</td>
<td>Dynamic Velocity Range</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FMM</td>
<td>Fast Multipole Method</td>
</tr>
<tr>
<td>FML</td>
<td>Fluid Mechanics Laboratory</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>LDV</td>
<td>Laser Doppler Velocimetry</td>
</tr>
<tr>
<td>LES</td>
<td>Large Eddy Simulation</td>
</tr>
<tr>
<td>NFAC</td>
<td>National Full-Scale Aerodynamics Complex</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>PIV</td>
<td>Particle Image Velocimetry</td>
</tr>
<tr>
<td>PSD</td>
<td>Power-Spectral Density</td>
</tr>
<tr>
<td>RANS</td>
<td>Reynolds-Averaged Navier-Stokes</td>
</tr>
<tr>
<td>RDECOM</td>
<td>U.S. Army Research, Development, and Engineering Command</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
</tr>
<tr>
<td>SIG</td>
<td>Synthetic Image Generator</td>
</tr>
<tr>
<td>SST</td>
<td>Shear-Stress Transport</td>
</tr>
</tbody>
</table>
1. Introduction

Following the first use of the term Particle Image Velocimetry (PIV) by Adrian [1], PIV has evolved to be the “dominant method” [3, p. 3] to experimentally obtain vector fields of fluid flow.

Many books and reviews provide in-depth treatment of every detail related to PIV, from its history over measurements and instrumentation to data processing. Adrian [2] highlights the milestones of progress of the technique throughout twenty years. Adrian and Westerweel [3] provide a comprehensive description of all facets of PIV, and serve as the main reference for this work regarding theoretical aspects. Raffel et al. [24], provide not only a more practically oriented treatment of the subject, but also many examples of application. Willert and Gharib [35] and Westerweel [31] laid the foundation of digital PIV. The latter also contributed excellent reviews of the subject ([32], [34]).

At the National Aeronautics and Space Administration (NASA) Ames Research Center (ARC), PIV is employed whenever high-resolution vector-fields of a fluid flow are of interest to the researchers. Examples range from the study of rotor-wakes [12] to the qualification testing of the supersonic parachute used for the Mars Science Laboratory (MSL) [28].

Although the analysis of uncertainty in PIV and the optimization of parameters is about as old as the technique itself (see for example [14]) the flow- and processing-dependent nature of the uncertainty ([29]) makes it difficult to obtain results with universal applicability. Furthermore, for complex multi-pass algorithms that are standard options in modern PIV processing tools, it is difficult to a priori quantify the impact that the processing step has on the results. This led to the desire at ARC’s Experimental Aero-Physics branch to develop a means of a posteriori analysis: if the data acquisition step can be simulated, the effect of the processing can be isolated by comparing the result to the known input. As Computational Fluid Dynamics (CFD) predictions play an increasingly important role in aerodynamics research, the validation and model-tuning of the CFD codes by means of experimental results become more important as well. It is therefore particularly desirable to investigate how PIV processing influences the data used to compare to the CFD results.

The process of simulating the data acquisition step is referred to as synthetic PIV. To the author’s knowledge, its use has so far been limited to randomized (Monte-Carlo) studies (e.g. [14]) or to very generic flow fields, either analytically specified or obtained from Direct Numerical Simulation (DNS) (e.g. [29]).
The main contribution of this work is the design and implementation of a tool to generate—in conjunction with existing software—synthetic PIV images from arbitrary CFD results or from analytical flow fields, to be used from the planning phase of an experiment over the result processing to CFD validation.

The thesis is structured as follows: Chapter 2 gives a short overview of the PIV technique and factors contributing to uncertainty of its results. Synthetic PIV and its previous uses are covered as well. As mentioned above, the literature on PIV is extensive, and the overview given here thus limited to the relevant aspects. Chapter 3 provides an in-depth description of the newly designed tool. Following this exposition are three chapters dedicated to different applications of the tool and ways in which newly gained a posteriori analysis capabilities may be applied to the results: Chapter 4 demonstrates the tool’s ability to treat CFD results on complex meshes; Chapter 5 illustrates how the tool can now be used by researchers at the U.S. Army Research, Development, and Engineering Command (RDECOM) Aeroflightdynamics Directorate (AFDD) at NASA Ames to validate and optimize rotor tip-vortex identification codes; and Chapter 6 illustrates an in-depth analysis of synthetic PIV based on a Detached Eddy Simulation (DES) of a cylinder wake. Finally, Chapter 7 concludes the thesis.
2. PIV

PIV is a non-intrusive measurement technique in which the displacements of particles following fluid motion is measured over a known time increment to obtain an instantaneous vector field of the fluid velocity. This is accomplished by seeding a flow with adequate particles and then illuminating the region of interest in the flow with two (or more) consecutive laser pulses. One or more cameras record the images of the particles in the flow at these precise moments in time. Together with the known properties of the optical setup, the particle displacements may be inferred from the displacement of their images, and their instantaneous velocity estimated via the finite-difference approximation

\[
\mathbf{u}_p(\bar{x}_p, t^*) = \frac{\Delta x_p}{\Delta t} + O(\Delta t^2),
\]

(2.0.1)

where \(\bar{x}_p = \frac{1}{2}[x_p(t) + x_p(t + \Delta t)]\) and \(t^* = t + 1/2\Delta t\). The particle velocity \(\mathbf{u}_p(\bar{x}_p, t^*)\) is then equated to the Eulerian velocity-field \(\mathbf{u}(\bar{x}_p, t^*)\).

2.1. Setup: Angular Stereoscopic PIV

The PIV setup used in the applications studied in this work is Stereoscopic PIV, a review of which was given by Prasad [23]. This technique utilizes two synchronized cameras viewing the flow field at different angles to allow the reconstruction of the entire three-vector of velocity. In particular, the configuration used for this work has one camera on either side of the laser light-sheet. Figure 2.1 shows a schematic of this optical arrangement. This arrangement has the advantage that both views are operated in forward scatter and have equal signal-to-noise ratios [23]. Of course, viewing the flow at an angle causes a non-uniformity in magnification. Also, choosing the included half-angle \(\theta\) between the cameras is effectively a trade-off between in-plane and out-of-plane resolution, with the ratio of out-of-plane to in-plane error scaling as \(\sim 1/\tan\theta\) [23]. The setup used respects the Scheimpflug condition, i.e. object plane, lens plane and image plane intersect in a common line. This ensures that, despite non-uniform magnification, the entire field of view is in focus.
2.2. PIV Processing

Equation (2.0.1) assumes knowledge of the particle displacement $\Delta x$ in physical (world) space. This information, however, must first be determined from the recorded images. Two steps are necessary for this:

1. determine the particle displacement in image coordinates, $\Delta X$.
2. map the particle image displacements to particle displacements in world coordinates, $\Delta x = F^{-1}(\Delta X)$.

The mapping function $F$ is used as a general description of the entire optical setup, which of course can only be known and modeled up to a certain accuracy. The inverse of this mapping is usually determined via a calibration process: a precisely known target is introduced into the field

Figure 2.1.: Angular stereoscopic PIV setup ([16, Figure 2]).
2.2. PIV Processing

of view of the system at the light-sheet position, and the processing software fits for example a polynomial function to features on the target. It should be noted that the 2D calibration procedure based on geometric pinhole imaging employed by the available processing software does not account for this Scheimpflug angle. Another point to be considered is that while the displacements are determined in 3D, the locations of the displacements are 2D. This means that stereoscopic PIV effectively performs a volume average over the light sheet thickness.

2.2.1. Determining the Particle Image Displacement - Image Interrogation

If the mean particle image spacing is smaller than the mean particle image displacement, it is impossible to identify corresponding particle images in subsequent images, necessitating a statistical approach based on particle patterns [3, p. 317]. All test cases studied in conjunction with this work correspond to this “high image density” case. The higher seeding density means a higher concentration of information in the images.

The method of choice to determine pattern displacements is to evaluate the image cross-correlation function in the frequency domain via the Fast Fourier Transform (FFT) [6]. This means that the image is subdivided into (potentially overlapping) interrogation windows. For each window, the maximum of the spatial cross-correlation between the same window in consecutive images is determined. The location of this maximum with respect to the center of the interrogation window is then accepted as the estimator for the particle pattern displacement vector. In order to obtain sub-pixel accuracy, one typically fits a continuous model function to the empirical, discrete cross-correlation function and uses the maximum of the model function. This process is referred to as sub-pixel interpolation [3, Chapter 8.5.2]. Using sub-pixel interpolation, the correct location of the maximum may be determined up to a precision of 0.1 to 0.2 pixel units under optimal conditions [3, Chapter 8.5.4].

The information contained in an image is two-dimensional, the measurement domain, however, is three-dimensional because of the nonzero light sheet thickness $\Delta z$. This means that any evaluation of image information content will be implicitly based on data integrated along each light-ray passing through the measurement volume. Furthermore, because the correlation is performed over a finite window size, any single displacement vector obtained by the technique will be based on the true displacements found in a volume around the point where it is assigned in the camera focal plane. The displacement obtained from cross-correlating patterns is effectively a (non-linear) low-pass-filtered representation of the underlying flow field [3, Chapter 8.3.8]. Under certain assumptions one may derive analytical expressions for the filtering. If the flow is uniform over the interrogation window, the cross-correlation is approximately a volume average. For a general flow analyzed by advanced algorithms that include multiple passes with differently shaped and positioned interrogation windows, finding analytical expressions becomes increasingly difficult, if not impossible. A characterization of a given PIV system in terms of
its frequency domain transfer-function may then only be performed empirically and requires
the exact knowledge of the true frequency content of the flow. Such knowledge is however only
available for analytically defined flow fields or CFD results, hence necessitating synthetic PIV
and the means to generate it from arbitrary known flow fields.

2.3. Accuracy and Uncertainty in PIV

Choosing experimental parameters has a profound impact on the accuracy of the measurements.
Optimizing these parameters has therefore received high attention in the literature (cf. e.g.
[14], [13]). The uncertainty in PIV has been shown to be a “very complex function of several
parameters” [30] and is highly dependent on the algorithm [29]. The quantification of the sources
of uncertainty in PIV is ongoing research. Approaches include analytical derivations, e.g. by
Westerweel [33], Monte Carlo simulations used to generate synthetic images, e.g. Timmins et al.
[30] and many others, as cited in Raffel et al. [24, p. 164], as well as a posteriori disparity
calculations based on super-resolution PIV (Sciacchitano et al. [27]). Ramasamy and Leishman
[26] provide a review of PIV uncertainty contributors especially relevant to rotor wake vortex
flow and use Laser Doppler Velocimetry (LDV) to benchmark the PIV technique.

The main sources of measurement uncertainty, as described in Adrian and Westerweel [3, Chap-
ter 10.7], are:

Tracer Dynamics

The implicit assumption in any PIV experiment is that the Lagrangian particle velocity may be
equated to the Eulerian velocity-field at the midpoint of the inferred particle trajectory. This
is only true for seeding particles that perfectly follow the flow, and the associated slip error is
typically of the order of 1%.

Image Mapping

In order to estimate particle displacements from particle image displacements, these have to be
mapped from the image to the experimental domain. The functions used to model the true
mapping can only approximate the actual optical setup.

Interrogation

As mentioned in Section 2.2.1, there is a minimum uncertainty of about 0.1 to 0.2 pixels, corre-
sponding to about 1–2% maximum velocity, associated with the exact location of the maximum
of the spatial cross-correlation. This value is particularly sensitive to local variations in image density, as well as to loss of correlation pairs due to particles entering or leaving a specific interrogation window between consecutive images.

Resolution

The performance of any PIV system is limited in terms of Dynamic Velocity Range (DVR) and Dynamic Spatial Range (DSR) [3, Section 1.3]. This means that for a given minimum observable displacement, determined by the uncertainty in the displacement of the particle image centroid location, \( \sigma_{\Delta X} \), there is a limit on the maximum observable displacement. This stems from the fact that the centroid displacement uncertainty is inversely proportional to the particle image diameter \( d_\tau \), which in turn depends linearly on the magnification. The “PIV uncertainty principle” ([3, Page 35]) is given by

\[
\text{DVR} \times \text{DSR} = \frac{L_X}{c_\tau d_\tau},
\]

where \( c_\tau \) is a measure of the interrogation algorithm performance and relates the displacement uncertainty to the particle image diameter as \( \sigma_{\Delta X} = c_\tau d_\tau \) and \( L_X \) is the size of the recording medium. This effectively limits the Reynolds number up to which, for a given system, all relevant length scales of a flow field may be resolved. When estimating spatial spectra from PIV data, this limitation has to be taken into account. In the case of synthetic PIV, this constraint is not as stringent because the centroid displacement uncertainty can be strongly reduced by eliminating all lens aberration, noise or interference effects.

Vector Placement

Equation (2.0.1) is only accurate to second order if the velocity vector is placed at the midpoint of the particle displacement trajectory. A non-uniformity in image density inside an interrogation window will cause a misplacement because the center of the window no longer coincides with the pattern centroid. The error associated with this effect is proportional to the velocity gradient in the interrogation window and vanishes for uniform flow.

Sampling Errors

To evaluate statistical properties of a flow, a large number of PIV measurements are required in order to adequately reduce the variance of a given estimator. For any mean, the standard deviation scales proportional to the inverse square-root of the number of independent samples. The independence criterion is however only fulfilled for intervals between subsequent recordings.
that are larger than the flow’s integral time scale. High-speed PIV systems typically record at a much higher frequency and consequently need more samples to reduce the variance.

**Errors due to velocity gradients**

The presence of strong flow gradients inside of an interrogation window can lead to an error explained in [24, Section 5.5.7]: particles with large velocities may not be present in the second interrogation window, even if the window is offset by the mean particle velocity, and thus bias the results towards lower velocities. Foucaut et al. [6, Section 2.1.7] investigated this effect and found a Root Mean Square (RMS) error of the order of 0.1 pixels for a gradient of 0.05 pixels per pixel, using $32 \times 32$ pixel interrogation windows containing 20 particle images each.

**Errors due to streamline curvature**

The only input to any PIV algorithm operating on data from double-pulsed systems are the particle positions at times $t$ and $t + \Delta t$. Any information on the actual trajectory is lost, one is left to approximate the distance traveled by the particle with the straight-line distance between the particle images. In the presence of strong flow curvature (e.g. rotor wake vortices, cf. [26]), this may lead to significant error.

**2.4. Synthetic PIV**

The process of generating synthetic PIV images from computed flow fields is fairly straightforward: given the velocity field and flow domain, one initializes the domain with a set of particle positions (position list A) and calculates from the velocity field the positions at which the particles would end up after being advected for the duration of one pulse separation time (position list B). An image generation software then computes for each position list one synthetic image per camera view. This is the procedure used in this work.

A different strategy has found use in the context of 2D flows: given an actual recorded image, flow features corresponding to presumably known velocity fields are extracted. One then computes a *synthetic frame B* by first interpolating the image to a larger size, applying integer displacements according to the assumed velocity field to the pixel values and then interpolating back to the original size. This technique was first used in [26], where the authors also mention that it is strictly limited to 2D flows. It has found application in the validation of the vortex identification code [38], to which the method developed as a part of this work will also be applied.

Another strategy to generate synthetic PIV images which is also limited to 2D flows is to use an actual experimental camera setup, but instead of a fluid carrying seed particles, a plate with a
2.4. Synthetic PIV

printed particle pattern is moved at a precisely known velocity through the region of interest. This procedure is capable of capturing all real-world effects related to the optical setup. None of the phenomena influenced by light scattering at the particles can, however, be reproduced. It is also difficult to illuminate the plate in exactly the same way as a translucent flow field. The generation of synthetic PIV images from computed flow fields has found many applications. As mentioned in Section 2.3, a number of authors have used the technique to study uncertainty in PIV. Timmins et al. [30], for example, used synthetic images of analytically specified steady flow to empirically derive uncertainty bounds of a given interrogation algorithm. In the framework of the “International PIV Challenge”, synthetic images from DNS as well as from analytically specified flow were used to study the performance of several competing algorithms. The results of the third such challenge are summarized in [29]. It was concluded that “[...] not one algorithm does yet the best on all test cases. The skill of the user, the pre-processing selected, the filters applied in the course of the iterations play a significant role, which vary, depending on the test case.” This result is another pointer towards the necessity of test-case specific synthetic PIV capability.

2.4.1. The Synthetic Image Generator (SIG).

The program used to generate the synthetic images in this work is the EUROPIV-2 SIG, which was developed by Lecordier and Westerweel [16]. Their publication cites various uses of the program. Foucaut et al. [6], for example, used the software to optimize experimental parameters. In Petracci et al. [22], the SIG was used to study the effects of calibration target misalignment using synthetic calibration images generated by a process similar to that described in Section 3.7 below.

The SIG takes a list of point coordinates in 3D, and, together with an appropriate configuration file, generates a corresponding image of particles as a camera would have imaged them in an experiment.

The imaging is based on geometric pinhole camera equations. The program is able to treat a broad range of optical setups including cameras with tilted sensors and can incorporate laser light-sheet thickness, particle size distribution, sensor pitch and fill-rate as well as other settings described in detail in [16]. Among the real-world effects not modeled are more complex optical distortions like aberration effects, as well as the generation of interference patterns in high seeding density cases.

Geometric imaging

For the cases presented here, the angular stereoscopic projection model was used. The geometric imaging equations which map the world coordinates \((X, Y, Z)\) to image coordinates \((x, y)\), as
2.4. Synthetic PIV

implemented in the SIG, are:

\[
x = d_i \sin(\alpha) + \frac{p_i \cos(\alpha) - r_i \sin(\alpha) - d_i \sin(\alpha)}{p_i \sin(\alpha) + r_i \cos(\alpha) + d_i \cos(\alpha)}
\]

\[
y = \frac{q_i d_i \cos(\alpha)}{p_i \sin(\alpha) + r_i \cos(\alpha) + d_i \cos(\alpha)},
\]

with

\[
p_i = X \cos(\theta) - Z \sin(\theta)
\]

\[
q_i = Y
\]

\[
r_i = X \sin(\theta) + Z \cos(\theta) - (d_0 + d_i),
\]

where \( \theta \) is the optical angle of the camera axis, \( \alpha \) is the sensor tilt-angle, \( d_0 \) is the object distance and \( d_i \) is the image distance. As mentioned in Section 2.1, the sensor tilt-angle was chosen to meet the Scheimpflug condition. The condition is met when the sensor tilt-angle is chosen as

\[
\alpha = \tan^{-1}\left[ \frac{d_i}{d_0 \tan(\theta)} \right].
\]  

(2.4.2)

A schematic of the setup along with an indication of the angles is provided in Figure 2.1 ([16, Figure 2]).

Particle Image Intensity Pattern

The SIG models the image generated by each particle on the camera’s Charge-Coupled Device (CCD) sensor as a 2D Gaussian curve, as shown in Figure 2.2 ([16, Figure 4]). The location of the maximum of the intensity pattern is determined by the aforementioned Equations 2.4.1. The standard deviation is a pre-configured parameter. The intensity pattern generated by each particle is integrated over the sensitive area of each pixel it covers, and the contributions of each particle are superposed to yield the final intensity value of each pixel. Equation 2.4.3 ([16, Equation 6]) is used to calculate the contribution of the individual particle image intensity pattern with maximum located at the particle position \((x_p, y_p)\) in image coordinates to the grey-level of the \( i \)th pixel, located at position \((x, y)\):

\[
I(x_i, y_i) \propto \frac{\pi}{8} d_p^2 \sigma_{px} \sigma_{py} \times \left[ \text{erf}\left\{ \frac{x - x_p + \frac{1}{2} f_{rx}}{\frac{\sigma_{px} \sqrt{2}}{2}} \right\} - \text{erf}\left\{ \frac{x - x_p - \frac{1}{2} f_{rx}}{\frac{\sigma_{px} \sqrt{2}}{2}} \right\} \right] \times \left[ \text{erf}\left\{ \frac{y - y_p + \frac{1}{2} f_{ry}}{\frac{\sigma_{py} \sqrt{2}}{2}} \right\} - \text{erf}\left\{ \frac{y - y_p - \frac{1}{2} f_{ry}}{\frac{\sigma_{py} \sqrt{2}}{2}} \right\} \right].
\]  

(2.4.3)
where $d_p$ is the particle diameter, $\sigma_{px}$ and $\sigma_{py}$ are the image widths (standard deviation) in the $x$- and $y$-direction, respectively, $f_{rx}$ and $f_{ry}$ are the sensitive fractions of each pixel in the $x$- and $y$-direction, respectively. The diameter of the individual particles need not be known in physical coordinates. In fact, in the cases presented below, this parameter was manually adjusted to achieve an acceptable overall image intensity for various degrees of seeding.

**Figure 2.2.:** Particle image intensity pattern ([16, Figure 4]).
3. The CFD2PIV Tool

As mentioned in Chapter 2.4, a tool to generate synthetic PIV images—the SIG—already exists. The main contribution of this work is the development of a multipurpose tool—“CFD2PIV” hereafter—to generate the particle position files from which the SIG may then generate the images. The term multipurpose is meant to illustrate the program’s capability to generate particle position files from different sources and for different applications.

3.1. Idea and Design Goals

The development goal was a program that would generate synthetic PIV images from both analytic flow fields and flow fields resulting from CFD calculations. The intended users are any applicants of PIV who wish to optimize experimental setup, investigate the magnitude of uncertainty in the results, obtain a way to compare experiment and simulation, or all of the above.

The development was guided by the interest of two groups at the NASA ARC: the experimental Aero-physics branch and the RDECOM AFDD. For the former, the focus lies on the investigation of uncertainties in the measurements and comparability to CFD results. The test application for this case are high-resolution CFD data as well as high-quality PIV data obtained from a cylinder wake. The latter is interested in improving the identification and quantification of helicopter rotor tip-vortex evolution. As a test case, analytical vortex flow fields were generated and analyzed.

The vast amount of data processed for this work and the many different flow cases studied made an efficient and fully parameterized implementation indispensable. CFD2PIV was implemented in FORTRAN using a modular programming philosophy.

3.2. Basic Principle

The aforementioned applications all consider a known flow field $u$, from which the positions of hypothetical tracer particles in the flow are to be generated. It is assumed that the PIV tracer particles are advected as passive scalars in the flow, i.e., that the particle density equals that of the fluid and that the particles do not exhibit finite size effects such as rotation. Under
these assumptions, the tracer particles may be equated to fluid particles. This is an implicit assumption in all PIV experiments. The particle positions then evolves as

\[ \frac{\partial x}{\partial t} = u. \]  \hspace{1cm} (3.2.1)

Note that this is equivalent to Equation (2.0.1).

The basic principle of generating a set of particle positions is straightforward:

1. initialize the flow domain with a set of particle positions
2. integrate (3.2.1) numerically in time starting at each position

Because we are not interested in the long-term evolution of the particle trajectories, but only in the movement during the time between two laser pulses, the errors imparted by assumption (3.2.1) are small, and may be further reduced by initializing the particle positions at the center of the time step and then integrating both forwards and backwards in time. This strategy also reduces the error of the numerical integration by effectively doubling the number of minor time steps.

### 3.3. Synthetic PIV from CFD

Generating synthetic PIV from arbitrary CFD output faces a challenge when the data is provided on an unstructured mesh, a condition that is assumed henceforth: the integration of Equation (3.2.1) requires the velocity field \( u \) to be evaluated at locations that do not correspond to the grid points. In other words, the problem at hand is how to perform a vast amount of space- and time-interpolations in a point cloud, while maintaining performance and accuracy at the same time. Interpolation methods may be classified as either local or global [8]. In the case of the former, the interpolant is constructed from the known values at the support points lying in some neighborhood of the query point. In the latter, an analytic expression for the interpolant is derived from the entire data and then evaluated at the query points. Global methods often require the solution of a linear system of the size of the support point field. In the case studied here, we seek to interpolate not only a scalar but a vector quantity and would have to construct a separate approximating function for every vector component.

Radial basis functions, inverse distance weighting and Shephard’s interpolation are all susceptible to Runge’s phenomenon when discontinuities are present in the data, as is the case in a general compressible flow field.

The only remaining alternatives for the problem at hand are therefore local linear schemes: linear interpolation in a tetrahedron using barycentric coordinates and tri-linear interpolation
3.3. Synthetic PIV from CFD

in a cube. Both methods, however, require the convex hull in terms of the support points to be constructed for the query point. One solution is to first compute a Delaunay triangulation of the point field and then look up the containing tetrahedron for every query. This was the first method implemented, but proved to yield unacceptable performance.

In a recent article, Murman [19] describes an algorithm to find the containing cube for a point in a point cloud. The algorithm, along with an adapted data structure, are described in the following. While the algorithm used to find a containing hypercube itself is provided in [19], the specific implementation discussed here is not.

3.3.1. Murman’s Algorithm with Optimized Tree Look-Up

The key idea of Murman’s algorithm is to construct a containing hypercube around a point by partitioning the $d$-dimensional search space by the $2^d$ sign permutations of the coordinates. For example, in 2D, given the query point $x_q = (x_q, y_q)$, the four partitions would be

$$
\begin{align*}
    x &\in (x_q, +\infty), y \in (y_q, +\infty); \\
    x &\in (-\infty, x_q], y \in (y_q, +\infty); \\
    x &\in (-\infty, x_q], y \in (-\infty, y_q]; \\
    x &\in (x_q, +\infty), y \in (-\infty, y_q].
\end{align*}
$$

In order to obtain the $i^{th}$ partition of a $d$-dimensional space, one may proceed in analogy to the conversion of $i$ to a $d$-digit binary number:

1. Initialize all lower bounds to $-\infty$ and all upper bounds to $+\infty$.
2. for $j = 1$ to $2^d$
   1.1. if modulo($i, 2$) > 0: set the $j^{th}$ lower bound to the $j^{th}$ element of the query vector.
   1.2. else set the $j^{th}$ upper bound to the $j^{th}$ element of the query vector.
   2. $i \leftarrow \lfloor i/10 \rfloor$

The source code used to construct the bounding box for the search space for the individual points in $d$ dimensions is provided in Listing A.1.

The algorithm will next search for the nearest neighbor (point with closest radial distance) in each partition. In case one or more partitions do not contain support points, the algorithm will discard the points with the greatest distance until a complete hypercube of a lower dimension is formed. This procedure corresponds to $0^{th}$-order extrapolation from the value at the projection of the query point onto the convex hull of the support-point cloud.
Interpolation is performed by successively reducing the hypercube along its dimensions: while the hypercube dimension is greater than 0, the query point is projected along the first remaining dimension onto the edges of the hypercube, the new vertices forming a new, nested hypercube. The relative distances to the edge endpoints are stored. As soon as the hypercube has been reduced to the query point, the interpolation weights of the original vertices are calculated by multiplying the complements of their relative distances to corresponding vertices of lower dimension. This corresponds to calculating the $2^d$ interpolation weights.

The implementation used to calculate the weights essentially corresponds to building a binary tree from leaves to root: the original vertices form the lowest level, or leaves, of the tree. Neighboring leaves form an edge. The parent node of two child nodes is constructed by projecting the query point onto that edge. Each level in the tree thus corresponds to a hypercube, the root being the query point itself.

In pseudo-code, the implementation may be described as follows:

1. a) Initialize a list $v$ of all vertices (nodes)—the first $2^d$ being the support points—and a corresponding list of complementary distances $d$. The total amount of vertices is $2^{d+1}$.

   b) Initialize a list $w$ of the $2^d$ weights of the support points to 1.

   c) Initialize a vertex counter $c$.

2. Build the tree:
   
   for $i = 2^d + 1$ to $2^{d+1}$
   
   2.1 Create a new vertex by projecting the query point $x_q$ onto an edge formed by vertices $a = v(c), b = v(c + 1)$:

      a) Compute the normalized distance of $a$ and $b$ to their new parent vertex:

         $d(c + 1) = \frac{(x_q - a - a)}{(b - a - a)}$

         $d(c) = 1 - d(c + 1)$

      b) The new projected vertex is itself just a linear combination of $a$ and $b$:

         $v(i) = d(c) \times a + d(c + 1) \times b$

   2.2 Increment the vertex counter: $c \leftarrow c + 2$

3. Traverse the tree from each leaf to the root and compute the weights:
   
   for $i = 1$ to $2^d$
   
   3.1 Initialize an index of the current node: $j = i$

   3.2 while $j < 2^{d+1}$

      a) Update the weight: $w(i) \leftarrow w(i) \times d(i)$
3.4. Synthetic PIV from an Analytical Flow Field

b) Find the index of the parent node: \( j \leftarrow (j + 2^d - \lfloor j/2 \rfloor) \)

The source code is provided in Listing A.2.

The most expensive step is the nearest neighbor search. For moderately clustered data in lower dimensions, this step may be vastly accelerated by pre-computing a tree-structure of the data. For this work, a k-d tree was chosen. Because the binning of the tree structure is aligned with the binning in the convex hull construction, the tree look-up is further accelerated by discarding extraneous regions. The implementation by Kennel [15], with a slight modification to make the implementation thread-safe, was used for this work. It was extended to allow the search space to be restricted in terms of the partitioning: the algorithm now discards any branches of the tree that have only nodes outside the constrained search space during the nearest neighbor search. For well-balanced trees, the look-up incurs an average cost of \( O(\log N) \) steps. For high-dimensional and strongly clustered data, the tree structure used for this work becomes less optimal.

3.4. Synthetic PIV from an Analytical Flow Field

When the flow field under consideration is known analytically, no interpolation is necessary. For this use-case, the focus in programming lies on how to provide the necessary flexibility to specify a wide range of flows. The approach chosen was to have the user specify individual flow elements via an input file, which will then be superposed to find the velocity at a given point. Aside from uniform flow, different vortex models are implemented to simulate rotor wake flow. Currently implemented are (in cylindrical coordinates \( u = [u_r, u_\Theta, u_z] \)):

- the Lamb-Oseen vortex model [5, Eq. 8]

\[
\begin{align*}
  u_\Theta(r) &= \frac{\Gamma_V}{2\pi r} \left( 1 - e^{-\alpha(r_c)^2} \right) \\
  u_r &= 0 \\
  u_z &= 0
\end{align*}
\]  

(3.4.1a) (3.4.1b) (3.4.1c)

with Oseen parameter \( \alpha = 1.25643 \), core radius \( r_c \) and the total vortex circulation \( \Gamma_V \). The core radius may be either fixed or specified at a time \( t_0 \). The latter will cause the model to assume the core to diffuse as \( r_c(t) \sim \sqrt{t} \) and to compute the core radius as \( r_c(t) = r_c(t_0) \sqrt{t/t_0} \).

- the Newman vortex model [20, Equations 9–11]
3.4. Synthetic PIV from an Analytical Flow Field

\[ u_\Theta(r, z) = \frac{\Gamma_V}{2\pi r} \left(1 - e^{-\left(\frac{r}{r_c}\right)^2}\right) \]  
\[ u_r(r, z) = -\frac{Ar}{2z^2} e^{-\left(\frac{r}{r_c}\right)^2} \]  
\[ u_z(r, z) = W - \frac{A}{z} e^{-\left(\frac{r}{r_c}\right)^2}, \]

with free-stream axial velocity \( W \) and the constant \( A \). \( A \) is derived from considerations of momentum to be \( A = D_0/4\pi \rho \nu \), with the profile drag of the vortex generating body \( D_0 \) and the fluid density and kinematic viscosity \( \rho \) and \( \nu \), respectively. The core radius \( r_c \) in the Newman model is the radius of an equivalent Rankine vortex, \( r_c(z) = 2z\sqrt{\nu/Wz} \).

The user may specify an arbitrary number of these flow elements along with respective position and spatial orientation vectors. If the vortex is specified to reside at \( x_V \) with the direction of its axis given by the vector \( d_V \), the following operations are required to find the induced velocity at a position \( x \):

- projection of \( x \) onto the vortex axis:
  \[ x_a = x_V + \frac{(x - x_V) \cdot d_V}{d_V \cdot d_V} d_V, \]  
- axial and radial distance of \( x \) to the vortex origin:
  \[ z = \|x_a - x_V\| \]  
  \[ r = \|x - x_a\|, \]
- transformation of the induced velocity from cylindrical to Cartesian coordinates:
  the tangential component: \[ u_t = u_\Theta d_V \times \frac{(x - x_a)}{r} \]
  the axial component: \[ u_a = u_z d_V \]
  the radial component: \[ u_r = u_r \frac{(x - x_a)}{r}. \]

The total induced velocity vector \( u \) is then calculated as \( u = u_r + u_a + u_r \).

The strategy to superpose multiple vortices to approximate a flow field is essentially a vortex method (see e.g. the review by Leonard [17]). The cost of \( N \) evaluations of the velocity induced by \( M \) analytical flow-field elements scales as \( N \times M \) in the current implementation.
3.5. Time Integration

The cases studied in this work have \( N \gg M \) and can be treated efficiently with the available implementation. To study more complex flows, however, it would be worthwhile to consider the implementation of the Fast Multipole Method (FMM) (cf. e.g. [9]) to evaluate the field in \( O(N + M) \), as the “naive” implementation quickly becomes computationally prohibitive for large \( M \). This would be especially beneficial in an extended study of wake flow fields, as one could directly implement a lifting-line based wake approximation. For example, the algorithm given in [4, Chapter 5.4] computes from a given lift distribution the individual locations and strengths of a number of vortices.

3.6. Seeding Non-Uniformity Simulation

An important error source in real-world experiments is the seeding non-uniformity that occurs when, for example, strong eddies centrifuge the seeding particles out of the vortex core (see Section 2.3). In order to investigate this effect, it is desirable to mimic it in the code. To accomplish this, acceptance-rejection sampling was implemented to allow the initial seeding particle locations to be sampled from arbitrary spatial seeding distributions. These may include specific locations of seeding deficiency such as vortex cores in the case of an analytic flow, measured average seeding densities or derived distributions such as a seeding density inversely proportional to local vorticity. The advantage of acceptance-rejection sampling is that the sampled distribution need not be normalized, which would be difficult to achieve especially for empirically specified distributions.

A simple yet efficient way to achieve seeding non-uniformity based on the flow field is also implemented: the reason for seeding non-uniformity in experiments is the finite particle response time that causes particles with a density greater than that of the fluid to lag behind the fluid motion. In other words, the particles are unable to perfectly follow streamline curvature. To mimic this effect, the user may specify an initial time span over which the particles are advected from uniform random positions tangentially to local streamlines. After this initialization period, e.g. vortex cores will be de-seeded. The process is implemented by reusing the normal advection
code but simply passing the forward Euler scheme as an integration routine. Particles lighter than the fluid (which will hence over-respond) cannot be mimicked by this procedure.

### 3.7. Calibration Target

In order for the PIV software to reconstruct the velocity field, it needs to learn the camera setup via a calibration image. This is an image of a known calibration target, usually a plate, that contains individual markings of known size and positions. The software can output a synthetic calibration image by placing particles such as to mimic the appearance of actual calibration markings. Note that this method to generate synthetic targets for use with the SIG was previously demonstrated in Reference [22]. The user may specify the size and spacing of the synthetic targets. The code then assigns a fraction of the given particle number to “draw” a thin border line along the \((x, y)\) borders of the domain. The rest is used to populate a regular grid spaced according to the user settings, of circles in the domain, by picking random radii within the user specified target size, and angles. Figure 3.1 shows an example of such a synthetic calibration image.

![Figure 3.1.: Synthetic calibration image.](image-url)
3.8. Field of View Matching

In the test cases for this work, the actual camera setup was only known approximately. Because of the angular camera setup, it was very difficult to match the image extent exactly to the extent of the experimental field of view. Analysis of the SIG source code revealed that the parameters used to specify the dimensions of the particle space do not actually exclude particles with positions outside of the specified domain. The values are only used to center all coordinates before calculating the projection. This provides a method to shift the particles in the virtual field of view.

In order to fit the simulated particle domain exactly to the camera field of view, the camera projection model of the SIG (Equation 2.4.1) was implemented in MATLAB and the fsolve numerical optimization function is used to find the required shift and viewing angle so that the maximum/minimum horizontal particle domain coordinates are exactly mapped to the maximum/minimum horizontal pixel coordinates.

3.9. Outline of the CFD2PIV Program

3.9.1. Input, Configuration and Parameterization

The CFD2PIV program is fully configurable and parameterizable via a settings file. The settings file is a standard text file containing “key = value” specifications of the parameters. Individual settings may be overridden by command-line arguments. The program can parse numerical specifications along with a physical unit specified as a string, so that e.g. the reference length of the problem may be specified in inches, while the reference velocity is provided in meters per second. For further processing, all numerical input values are appropriately non-dimensionalized. CFD results may be passed to the program in terms of a grid file and multiple result files. In addition to structured-grid results provided as OVERFLOW or Plot3D files, the FieldView unstructured grid format can be read. The code can read both single and multigrid files written in either big-endian or small-endian byte-ordering. Analytical flow fields may be specified via a separate file.

3.9.2. Data Structures and Program Control Flow

The first step in the program is to initialize a data structure holding all relevant settings with default values. The program then reads any command-line settings provided, and keeps track of which values have been set via command-line, as these will take precedence over any values that are specified in a settings file. This is especially convenient for scripted calls to the program, as for example the number of threads to be used can be passed by the calling script to handle
3.9. Outline of the CFD2PIV Program

load-balancing. The next step is to parse any settings file that was provided via command-line, and to complete the settings structure. This includes calculating values which are derived from other settings, such as the reference time-scale \( \tau = \frac{l_{ref}}{V_{ref}} \). The program also checks for the availability of the specified input files, and in case the user specifies input files using a wildcard, the program gathers the names of all available results files for later processing.

Once the settings structure is populated with values, the value of the action parameter determines which high-level functionality of the code should be executed. Currently implemented are:

1. Convert OVERFLOW q-files to Plot3D files.
2. Convert OVERFLOW q-files to TecPlot ASCII files.
3. Write a calibration target.
4. Synthetic PIV from analytical flow.
5. Synthetic PIV from an OVERFLOW q-file result set.
6. Synthetic PIV from a single result available as a combination of OVERFLOW or Plot3D q-file and an unstructured result in the FieldView unstructured grid format.

Other high-level functionality can be realized simply by reusing appropriate lower-level functionality, such as an output of an analytically specified flow onto a regular grid, but implementing all possible I/O configurations was deemed a secondary priority in the overall project. Each high-level procedure takes the settings structure as a single argument. It then starts by asserting that the necessary input is specified. Following is a detailed description of Items 4, 5 and 6.

**Procedure 4—Analytical Flow Fields**

To process analytical flows, the procedure first parses the text file containing the parameters for the different flow elements, which was specified in the settings file. In order to allow for an arbitrary amount of flow elements without having to explicitly specify the number, the information is stored in a linked list. Once the file is parsed the list is traversed and all parameters are appropriately non-dimensionalized. Next, the particle list is initialized. If any of the specified flow elements contain information about seeding non-uniformity (currently implemented: a void fraction in vortex cores), the particle positions are sampled using acceptance-rejection. Otherwise, they are just uniform-randomly distributed over the domain. Calculating the advected particle positions is next. For each of the two half time-steps, the list of particles is traversed. For each particle, the integration routine receives the particle position, the initial time and time-step length, as well as a function to calculate the velocity from the flow-element list as an input. In order to calculate the velocity at a given time and location, the linked list of flow
3.9. Outline of the CFD2PIV Program

elements is traversed, and the contribution of that element to the velocity at the query point is added. The new particle positions are saved in two separate variables and passed to the output routine. This procedure simply writes the three-component position list to the specified output file. It automatically chooses the executable’s location as a backup if the specified filename is unavailable for writing, which may happen for example if the program is called twice with the same settings file. In this case, the filename is also made unique with a time stamp.

Procedure 5—CFD Results as Q-Files

At first, the grid file is read and the information stored in a data structure equivalent to that in the file. Any user-specified orientation of the coordinate axes is taken into account during read-in. After this, the grid points are copied to a separate data structure as a simple point list, removing any structure from the data. This list is then sorted to remove any duplicate points. If the user has specified a domain for the computation, any points outside will be removed in this step. Next, the point list is organized in a $k - d$ tree, which will be used to efficiently find nearest neighbors in the interpolation routine. The routine responsible for aggregating the settings has already established the list of results files to be processed. Before this step, the particle positions are initialized to be uniform-randomly distributed throughout the domain. If a non-zero initialization time was specified, the code advects the particles using the forward Euler scheme to produce a seeding non-uniformity effect. Once the initial particle positions are determined, the list of available q-files is processed sequentially. Every time step will reuse the same initial positions. To perform time-interpolation, more than one q-file is held in memory simultaneously. For each q-file, the code first checks which other files correspond to time steps needed for the time-interpolation. It then discards q-file information currently in memory that will not be used in the current time-step and reads any q-files that are required but not yet loaded. The read data is transformed to a simple list and permuted such as to maintain index-value correspondence with the grid data. The data structure responsible for holding the point and value lists as well as the tree structure in memory is referred to internally as a field. Implementing a separate structure for this purpose serves as a layer of abstraction between the information and the format in which it was read, making it easier to extend the code to be able to process additional result formats. Particle advection occurs similar to the previous case (3.9.2). Only the function responsible to retrieve the velocity at a given point and time passed to the integration routine is exchanged for the spatial-temporal interpolation routine. After the two new particle lists containing the new positions have been generated, they are processed by the same output routine described earlier, whereby the time-step number is appended to the output filename.
3.10. From CFD to PIV Using the CFD2PIV Tool

**Procedure 6—CFD Results on both Structured and Unstructured Grids**

In contrast to the previously discussed procedure, the code is not yet able to process more than one result time step, i.e. no time-interpolation is performed. This functionality can however be easily implemented by adopting a file naming convention to establish a time-step wise correspondence between the structured and un-structured data. The only difference in processing data from multiple sources compared to the previously described procedure is how the field structure is populated. Once the abstraction from the information-carrying format is complete, advecting the particles and writing the output files proceeds in complete analogy to procedure 5. In order to assemble the field structure, first both grid formats are read into in-memory representations of the file structure, whereby no connectivity information from the unstructured grid-file is read. Next, the result files are read. The grid points are then exported to point lists, ignoring any out-of-domain points if applicable. Because the grids overlap in space, each point carries an index that signifies whether it is actually part of the solution. This information is stored in the grid itself in the structured data and in the results file of the unstructured data. It is retrieved by a separate function for both data sources and used to further filter the point lists. The point lists are then sorted first separately and then combined. This combined list is finally used to construct a $k-d$ tree. The mapping of the original grid-point indices into the sorted combined list is used to assemble an index-equivalent list of the data, thus completing the field structure.

**3.10. From CFD to PIV Using the CFD2PIV Tool**

This section is intended to describe use-cases for the tool and the steps involved to go through the entire result-generation process.

To what end the CFD2PIV tool may be used depends on the available data. Assuming that no CFD is available, specifying analytical flows can still be used to gauge the performance of the PIV processing for a given setup and to optimize parameters such as pulse separation or viewing angles. Reynolds-Averaged Navier-Stokes (RANS) simulations may provide a more realistic “proving-ground” for the PIV processing, but of course, the comparison of turbulent spectra is impossible. The ability of the PIV processing to reproduce the mean flow may, however, still be analyzed. One could also validate the Reynolds-stresses. Analyzing processing performance in terms of resolution, for example via spatial spectra, is restricted to the availability of Large Eddy Simulation (LES), DES, or Direct Numerical Simulation (DNS) solutions that resolve at least as many turbulent scales as the PIV processing. This may lead to a limitation in terms of Reynolds number of the flows to which this method my be applied.

To generate synthetic PIV images from CFD using the CFD2PIV tool, the first step is to write the necessary settings files. One settings file is needed to process the results and generate the particle
3.10. From CFD to PIV Using the CFD2PIV Tool

position lists, the other to write a particle list corresponding to a synthetic calibration target. An important setting is the correct specification of the result-file coordinate system: all further steps assume that the $z$ coordinate direction is perpendicular to the laser light-sheet. The number of seeding particles and hence the resultant image density, as well as the laser pulse-separation, need also be specified. The CFD2PIV tool can handle any coordinate axis permutation and flipping. It has proven to be convenient to use scripting to automatically generate the settings files from a template if many similar cases, for example to conduct a parameter study, are to be generated. Scripting the call to the CFD2PIV tool can also be convenient. Although the program handles the advection computation in parallel, it may be advantageous to have many instances of the program working in parallel, each with a reduced number of threads available. This is because for moderate numbers of particles, reading the grid and the results is more time-consuming than computing the particle positions. One should therefore balance between disk and CPU load.

After the particle position lists are generated, they need to be passed to the SIG for image generation. The SIG again depends on appropriate settings files which give detailed information about the optical setup to be simulated. The positions written by the CFD2PIV tool are non-dimensional. The SIG settings must therefore also be specified non-dimensionally. Often, the exact optical setup is unknown, and/or the grid domain uses a different coordinate system. For this case, a MATLAB script is provided that will compute the domain parameters needed in the SIG settings such that the computed particle field will fill the entire synthetic image. For multi-camera setups such as stereoscopic double-exposure PIV, one settings file per camera is required. The SIG needs to be called once with each settings file for each produced particle list. The particle list corresponding to the synthetic calibration target is processed using identical settings to those used for the seed particles, ensuring that the processing software is provided with the correct information to perform the calibration.

The calibration and particle-field images are now ready to be processed in the same way as those resulting from an actual experiment.
4. Test Case A: Synthetic PIV from Coupled Unstructured, Unsteady RANS/Adaptive Cartesian Euler CFD

One of the unique selling propositions of the CFD2PIV tool is its ability to perform a vast number of interpolations efficiently and in parallel, without assuming any structure of the data. In order to demonstrate this capability, a challenging test case ([36]) was kindly provided by the RDECOM AFDD at NASA Ames:

As documented in [37], the innovative simulation framework HELIOS allows to couple an unsteady RANS simulation on an unstructured near-body mesh with the solution of the Euler equations on a multi-level Cartesian grid where Adaptive Mesh Refinement (AMR) is employed. This CFD approach was applied to the simulation of an isolated rotor in hover conditions. The geometry is modeled after the rotor used in the V-22 “Osprey” tilt-rotor aircraft. Figure 4.1 shows the vorticity obtained from the solution overlaid on the hybrid mesh. A color plot of momentum magnitude on two perpendicular planes through the domain is depicted in Figure 4.2.

![Figure 4.1.: Wake vorticity overlaid on adaptive eight level off-body mesh ([37, Figure 7c]).](image)

The data was consequently provided on two meshes: the structured part in the Plot3D format, containing over 88 million grid points, and the near body unstructured mesh in the FieldView unstructured grid format, containing close to 8 million grid points. Analogously to the rotor tip-vortex application presented in Section 5, it was chosen to perform synthetic PIV on a plane perpendicular to the rotor-plane, immediately behind a rotor-blade. All other processing and setup parameters were also chosen equal to that case (c.f. Tables 5.1.1, 5.2.1 and 5.2.2), unless
Figure 4.2.: Momentum magnitude of the rotor wake on two perpendicular slices through the simulation domain, obtained from the data provided by Wissink [36].

indicated otherwise. The CFD2PIV program was configured to discard any point outside this region of interest, which left about 27,500 support-points in the domain, which in physical space corresponds to $150 \times 200 \times 1$ in. This point subset is shown in Figure 4.3. To make the reference CFD solution accessible to interpolation in MATLAB, the CFD2PIV program itself was used to interpolate the result onto a regular grid. The result is shown in Figure 4.4. For the synthetic optical setup, camera angles of $\pm 45^\circ$ were chosen, and the numerical procedure outlined in Section 3.8 was used to find the object distance $d_0$ that would exactly fit the $x$-dimension into the horizontal image size. This left a visible $y$ domain extent of about 60 in. After particle position generation, calibration image generation and PIV processing, the result shown in Figure 4.5 was obtained. The velocity vectors (constant length) and v-velocity component coloring clearly show the wake velocity deficit, the young tip-vortex, and the tip-vortex shed by the previous blade. Comparing the two figures, synthetic data and interpolated CFD appear to be in good agreement. The chosen pulse separation of 37 µs appears to be sufficient to allow for the resolution of the vortices.

The successful completion of the data-generation cycle for this case demonstrates the usefulness of the CFD2PIV tool for applications with data provided on arbitrary combinations of unstructured and structured meshes. It may be used for flow-specific optimization of future PIV measurements (pulse-separation, field of view, angles, etc.) and to validate certain CFD predictions (e.g. average velocity, vortex core-size etc.) with measurements. It should be noted, however, that although the provided data were of considerable size, the number of grid-points in the investigated domain corresponds to about 0.0026 support points per pixel, or about 0.16
Figure 4.3.: The grid points in the domain sub-region chosen for synthetic PIV, colored by momentum magnitude. Data provided by Wissink [36].

points per vector, if $32 \times 32$ px windows with 75% overlap are used for processing. This particular set of input should therefore not be used to assess PIV resolution. Because the data are obtained from an Euler/RANS solution, it is also not possible to compare turbulent spectra.
Figure 4.4.: The CFD2PIV program was used to interpolate the data ([36]) onto a regular grid. The figure shows the Mach number in $y$-direction on a single plane of the interpolated grid.

Figure 4.5.: Processed synthetic PIV (velocity vectors as well as color by velocity $v$-component magnitude) on a plane perpendicular to the rotor plane, immediately behind a rotor blade. The underlying CFD data were provided by Wissink [36].
5. Test Case B: Synthetic PIV from Analytical Flow Fields in Support of Rotor Tip-Vortex Identification

In “Wind Tunnel Measurements of Full-Scale UH-60A Rotor Tip Vortices” ([38]), the authors describe full-scale measurements of a UH-60 rotor in the National Full-Scale Aerodynamics Complex (NFAC) 40- by 80-Foot Wind Tunnel at NASA Ames. Two objectives of this study were the identification of vortex properties via the least-squares fit of an assumed vortex model to PIV data and the estimation of measurement uncertainties. This section describes how Synthetic PIV may be used to contribute to both goals. All PIV processing was conducted using the commercial software DaVis 8.5.1 by LaVision.

5.1. Synthetic PIV Configuration

The first objective was to recreate the experimental setup used in the rotor-wake study of Yamauchi et al. [38]. The setup-relevant data available from the publication are summarized in Table 5.1.1. The missing camera view and sensor-tilt angles were determined via the numerical procedure described in Section 3.8.

| Table 5.1.1.: Data available about the optical setup used in [38]. |
|----------------------|------------------|
| pixel pitch          | 9 µm             |
| object distance      | 21 ft            |
| image distance       | 128 mm           |
| horizontal field of view | 14.6 ft   |
| horizontal image dimension | 4008 px   |
| vertical image dimension | 2672 px   |

5.2. Vortex Identification Model Testing

The wake model used in [38] consists of one or more Lamb-Orseen vortices that generate a planar velocity field which is in turn fit to the measured data. Synthetic PIV may be used to generate test data for the model: the model itself provides the input and is then used to recover the simulation parameters. Because the simulation environment provides full flexibility in defining
the measurement conditions, the model performance and robustness may be assessed by using test cases of varying complexity and data quality.

In close cooperation with the authors of [38] at the RDECOM AFDD at NASA ARC, a systematic study of different vortex configurations was developed for this purpose [25]. Table 5.2.3 lists the test cases that were generated. All images were produced using the settings for the Synthetic Image Generator (SIG) listed in Table 5.2.2. The CFD2PIV settings listed in Table 5.2.1 were used for all cases. As in the real measurements, the pulse separation was chosen such that for an out-of-plane velocity of $33 \text{ m s}^{-1}$, the distance traveled perpendicular to the laser light-sheet is $\frac{1}{4}$th of the laser light-sheet thickness. This means that on average, 75% of the particles will remain inside the light sheet. The particle density was chosen to be extremely high (10000000 particles total) to achieve maximum correlation quality.

| Table 5.2.1.: Settings for the CFD2PIV program. |
| Setting | Description | Value |
| l_ref | Reference length | $7/32\text{ in}$ |
| v_ref | Reference velocity | $33\text{ m/s}$ |
| N | Number of particles | $10,000,000$ |
| DT | Time between consecutive images | $84.185606\mu\text{s}$ $\equiv \frac{1}{2} \cdot \frac{l_{\text{ref}}}{v_{\text{ref}}}$ |
| | | $42.092803\mu\text{s}$ $\equiv \frac{1}{4} \cdot \frac{l_{\text{ref}}}{v_{\text{ref}}}$ |
| | | $28.061869\mu\text{s}$ $\equiv \frac{1}{6} \cdot \frac{l_{\text{ref}}}{v_{\text{ref}}}$ |
| Domain | Spatial region in which to simulate the particles, corresponding to the PIV ROI | $X$ [min, max] $-400.5$, $400.5$ |
| | | $Y$ $-109.7$, $109.7$ |
| | | $Z$ $-0.5$, $-0.5$ |
| Overlap | In order to simulate particles entering and leaving the domain during the timestep, their initial positions overlap the domain | $[0,0.50\%]$ |
5.2. Vortex Identification Model Testing

Table 5.2.2.: Settings for the SIG.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_dimX</td>
<td>Image width in pixel</td>
<td>4008</td>
</tr>
<tr>
<td>p_dimY</td>
<td>Image height in pixel</td>
<td>2672</td>
</tr>
<tr>
<td><strong>Particle space dimension</strong></td>
<td>Used to calculate effective image center</td>
<td></td>
</tr>
<tr>
<td>r_xmin, r_xmax</td>
<td>-283.840</td>
<td>517.160</td>
</tr>
<tr>
<td>r_ymin, r_ymax</td>
<td>-109.7</td>
<td>109.7</td>
</tr>
<tr>
<td>r_zmin, r_zmax</td>
<td>-0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>l_sheet_type</td>
<td>Light sheet variation in z</td>
<td>Uniform (top hat distribution)</td>
</tr>
<tr>
<td>lsheet_rpos_z</td>
<td>Position of the light sheet center in z</td>
<td>0</td>
</tr>
<tr>
<td>lsheet_rthickness</td>
<td>Thickness of the light sheet in z</td>
<td>1 (reference length)</td>
</tr>
<tr>
<td>part_distribution</td>
<td>Particle distribution</td>
<td>uniform</td>
</tr>
<tr>
<td>part_mean_diam</td>
<td>The diameter of the particles used in the intensity calculation, (I \sim d^2)</td>
<td>0.25</td>
</tr>
<tr>
<td>pattern_type</td>
<td>Shape of the intensity pattern of a particle image on the CCD</td>
<td>gaussian</td>
</tr>
<tr>
<td>pattern_mean(&lt;x</td>
<td>y&gt;</td>
<td>)</td>
</tr>
<tr>
<td>projection_type</td>
<td>The projection model to use</td>
<td>angular</td>
</tr>
<tr>
<td>projection angle</td>
<td>The angle at which the camera views the ROI, 0 corresponding to a view directly upstream</td>
<td>Cam 1 63.74247373° Cam 2 116.25752627°</td>
</tr>
<tr>
<td>projection_tilt_angle</td>
<td>Tilt angle of the CCD, calculated from the Scheimpflug condition</td>
<td>Cam 1 2.32135635° Cam 2 -2.32135635°</td>
</tr>
</tbody>
</table>
### 5.2. Vortex Identification Model Testing

#### Table 5.2.2.: (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccd_fill_ratio_&lt;x</td>
<td>y&gt;</td>
<td>Sensitive fraction of each pixel</td>
</tr>
<tr>
<td>ccd_pixel_&lt;horizontal</td>
<td>vertical&gt;_pitch</td>
<td>The number of pixels per reference length</td>
</tr>
<tr>
<td>optic_object_distance</td>
<td>The effective distance of the object to the lens plane (do)</td>
<td>11521_reference, 21 ft</td>
</tr>
<tr>
<td>optic_image_distance</td>
<td>The effective distance of the image to the lens plane (di)</td>
<td>23.037,120,361 reference, 128 mm</td>
</tr>
</tbody>
</table>

In the following, a brief description of each test case studied is given. Details can be found in Table 5.2.3.

**Case 1: Uniform Flow**

A uniform flow field is ideal to gauge the performance of the PIV processing software used and to evaluate the quality of the calibration.

**Case 2: A Pure, Isolated Vortex**

Case 2 is a baseline case for the vortex identification code.

**Case 3: A Vortex with Out-of-Plane Flow**

In forward flight, the blade vortices are embedded in a constant flow along their axis. This superposition is simulated in Case 3.

**Case 4: Vortex Tilt**

It is rarely the case that the observed vortices are exactly perpendicular to the laser light-sheet. Case 4 therefore investigates a vortex with 4 combinations of tilt-angles.
5.2. Vortex Identification Model Testing

Case 5: 2 Vortices

Multiple vortices in close proximity pose a challenge for the code to pick up the correct vortex strengths. This is simulated with two vortices of 2 in core radius separated by 8 in.

Case 6: Vortex Convection

Case 6 was intended to simulate vortex convection. At present, however, the CFD2PIV tool cannot compute any time evolution of the specified analytical flow field beyond the vortex core diffusion. Therefore, the case was dropped from the simulation matrix.

Case 7: Vortex-Core Seed-Void Study

This case tries to identify the limit of core de-seeding up to which vortex parameters may still be identified. The 25 flow cases generated for this study consist of a single Lamb-vortex with out of plane uniform flow. A circular seed-void centered in the vortex core is varied from 5% to 125% of the core radius over the cases. Figure 5.1 shows a synthetic image generated as part of this study.

Figure 5.1.: Synthetic particle image for Case 7, the analytically specified seed-void in the vortex core is clearly visible.
5.2. Vortex Identification Model Testing

Case 8: The Newman Vortex Model

Case 8 consists of a single Newman vortex, which includes in its formulation an out-of-plane velocity component and is thus similar to Case 3. The drag-per-density parameter used is based on the total torque of a UH-60 helicopter and standard conditions.

Case 9: Vortex Sheet

In order to simulate a rotor blade wake, an analytical flow field based on a discrete approximation of the vorticity distribution in a blade wake was implemented, based on Prandtl’s lifting line theory [4, Chapter 5]:

Following the Kutta-Joukowski theorem, we equate the circulation around the blade as a function of the radial position \( \Gamma(r) \) to the local lift per unit span, \( \tilde{L}(r) \), divided by the local inflow momentum:

\[
\Gamma(r) = \frac{\tilde{L}(r)}{\rho V(r)}.
\]  

(5.2.1)

The rotor blade is approximated as a constant cross-section wing and is assumed to be uncambered. We may then write the local lift per unit span as

\[
\tilde{L}(r) = \frac{1}{2} \rho V(r)^2 c \cdot 2\pi \alpha,
\]  

(5.2.2)

where \( c \) is the (constant) blade chord and \( \alpha \) is the inflow angle in radians, also assumed constant. For a rotorcraft in hover, the inflow velocity is given in terms of the tip velocity \( V_{tip} \) and the blade tip radius \( R \) as \( V(r) = V_{tip} r / R \), so that we arrive at the (constant) circulation derivative in radial direction

\[
\Gamma' = \frac{1}{2} c \cdot 2\pi \alpha.
\]  

(5.2.3)

According to Prandtl’s lifting-line theory, the radial change in circulation is compensated by a vortex sheet shed from the trailing edge. We approximate this by a number of discrete vortices that will be assigned to blade elements and carry a circulation equal to change of total circulation over their respective element. In this case, the circulation is increasing radially, which means that the individual vortices have counter-clockwise negative circulation. Because the circulation has to vanish at the blade tip, a tip vortex with opposite sign is added. Its circulation strength is based on a common empirical factor of 70% of the combined circulation of the other vortex elements.
5.2. Vortex Identification Model Testing

The aforementioned equations were all implemented in a MATLAB script that will automatically generate the input flow field file for the CFD2PIV program for an arbitrary number of vortices chosen to discretize the circulation distribution.

For the study presented here, a total of 21 vortices was used. Figure 5.2 shows the processed velocity field.

![Figure 5.2: Case 9, processed. The constant length arrows are showing the direction of the velocity vector. The image is colored by the y-component of velocity.](image)

**Table 5.2.3.:** Description of the synthetic images generated for vortex identification.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cal</td>
<td><strong>The calibration images</strong></td>
</tr>
<tr>
<td></td>
<td>Target spacing</td>
</tr>
<tr>
<td></td>
<td>X: 10 cm; Y: 10 cm</td>
</tr>
<tr>
<td></td>
<td>Target diameter</td>
</tr>
<tr>
<td></td>
<td>0.375 in=9.5250 mm</td>
</tr>
<tr>
<td></td>
<td>Target z position</td>
</tr>
<tr>
<td></td>
<td>−20 mm, −15 mm, . . . , 20 mm</td>
</tr>
<tr>
<td>Case 1</td>
<td><strong>3 uniform flow fields</strong></td>
</tr>
<tr>
<td>Flow 1</td>
<td>[u, 0, 0]; u = 33 m/s</td>
</tr>
<tr>
<td>Flow 2</td>
<td>[0, 0, w]; w = 33 m/s</td>
</tr>
<tr>
<td>Flow 3</td>
<td>[u, 0, w]; u = w = 33 m/s</td>
</tr>
</tbody>
</table>
### Table 5.2.3.: (continued)

<table>
<thead>
<tr>
<th>Case 2</th>
<th>A pure, isolated vortex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 1</td>
<td>Model: Lamb-Oseen</td>
</tr>
<tr>
<td></td>
<td>Total circulation: $\Gamma = 10 \text{ m}^2/\text{s}$</td>
</tr>
<tr>
<td></td>
<td>Core radius: 1 in</td>
</tr>
<tr>
<td></td>
<td>Position: $[0, 0, 0]$</td>
</tr>
<tr>
<td></td>
<td>Orientation: $[0, 0, 1]$</td>
</tr>
<tr>
<td>Flow 2</td>
<td>Core radius: 2 in</td>
</tr>
<tr>
<td>Flow 3</td>
<td>Core radius: 3 in</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 3</th>
<th>A vortex with a uniform flow superposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 1</td>
<td>Vortex as in Case 2, Flow 2 with Case 1, Flow 2 superposed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 4</th>
<th>An inclined vortex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 1</td>
<td>$[0, 5^\circ, 0]$</td>
</tr>
<tr>
<td>Flow 2</td>
<td>properties as in Case 2, $[0, 15^\circ, 0]$</td>
</tr>
<tr>
<td>Flow 3</td>
<td>Flow 2 with Tilt: $[0, 25^\circ, 0]$</td>
</tr>
<tr>
<td>Flow 4</td>
<td>$[15^\circ, 15^\circ, 0]$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 5</th>
<th>Two vortices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 1</td>
<td>Properties as in Case 2, Flow 2 with:</td>
</tr>
<tr>
<td></td>
<td>Position: $[-4 \text{ in}, 0, 0] \quad [4 \text{ in}, 0, 0]$</td>
</tr>
<tr>
<td></td>
<td>Orientation: $[0, 0, 1] \quad [0, 0, -1]$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 7</th>
<th>1 vortex with z-flow and varying seeding void in the core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 1–Flow 25</td>
<td>Properties as in Case 3, Flow 1 with void radius $[5, 10, \ldots, 125]%$ of the core radius</td>
</tr>
</tbody>
</table>
5.2. Vortex Identification Model Testing

Table 5.2.3.: (continued)

<table>
<thead>
<tr>
<th>Case 8</th>
<th>1 Newman vortex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 1</td>
<td>Total circulation $\Gamma = 10 \text{ m}^2/\text{s}$</td>
</tr>
<tr>
<td></td>
<td>Axial free-stream $W = 33 \text{ m/s}$</td>
</tr>
<tr>
<td></td>
<td>Kinematic viscosity $\nu = 14.778,843 \text{ mm}^2/\text{s}$</td>
</tr>
<tr>
<td></td>
<td>Drag per density $52.51 \text{ m}^4/\text{s}^2$</td>
</tr>
<tr>
<td></td>
<td>Position $[0, 0, -0.218, 75 \text{ in}]$</td>
</tr>
<tr>
<td></td>
<td>Orientation $[0, 0, 1]$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 9</th>
<th>Vortex sheet containing 21 Lamb-Orseen vortices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 1</td>
<td>Total circulation $\Gamma = 1.370 \text{ m}^2/\text{s}^{-1}$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma = 19.177 \text{ m}^2/\text{s}^{-1}$ (tip vortex)</td>
</tr>
<tr>
<td></td>
<td>Core radius 2 in</td>
</tr>
<tr>
<td></td>
<td>Position Uniformly distributed between $[-5.343 \text{ m}, 0, 0]$ and $[1.830 \text{ m}, 0, 0]$</td>
</tr>
<tr>
<td></td>
<td>Orientation $[0, 0, -1]$, tip vortex: $[0, 0, 1]$</td>
</tr>
</tbody>
</table>

5.2.1. Results

In order to perform higher-order calibration, multiple synthetic calibration images, each corresponding to a different $z$ (plane normal) location, were generated. The calibration quality, however, proved to be unacceptably bad with this approach, so the classical pinhole model was used instead.

Case 1

An interesting result obtained early in the process was a pronounced sensitivity of the results to the quality of the calibration. Even with a calibration RMS error of the fit of below 1 pixel, an in-plane uniform flow (Case 1, Table 5.2.3) could only be reconstructed to within $\pm 5\%$. Furthermore, the error was spatially dependent and more pronounced in the camera far-field. These calibration-related errors could be significantly reduced via repeated self-calibration. The final RMS deviation of the synthetic to the analytical uniform flow was between 0.1 and 0.2 pixels, which corresponds to the lower theoretical bound achievable by cross-correlation interrogation (see Section 2.3).
5.2. Vortex Identification Model Testing

Cases 2 through 9

Cases 2 through 9 were processed using the same processing parameters used in Reference [38]. The vortex identification was kindly undertaken at the AFDD. Preliminary results ([25]) for cases 2, 3, 4 and 7 are presented in Table 5.2.4. The cases containing multiple vortices (5 and 9) and case 8 (Newman model), were not yet processed by the vortex identification at the time of writing. The parameters identified via numerical optimization from synthetic PIV are the core radius $R_c$, the circulation $\Gamma$, the $(x, y)$ position of the vortex core and the vortex tilt angles about the $x$ and $y$ axis, $\theta_x$ and $\theta_y$, respectively.

The vortex identification code is able to identify the core size to within about $\pm 4.5\%$ of the true value for all cases. The error in the radius identification begins to rise with the size of the core seed-void, beginning with Case 7 Flows 18. Case 7 Flow 18 corresponds to a seed-void in the core center of 90\% of the core radius. This percentage is increased by 5\% for each following flow number. For the cases without seed-voids and for seed-voids smaller than 90\%, the relative error in the core size estimation was below $\pm 2.5\%$, with Case 7 Flow 10 forming an exception. In this case, the flow angle was considerably over-predicted (15° instead of 0°). This case was re-run with the vortex tilt forced to zero, and the resultant relative error is 0.2\%.

The circulation of the vortices is estimated to within $\pm 1.7\%$ of the true value for all cases, with no consistent trend in the error immediately apparent. The average relative error in the circulation identification is 0.5\%.

The vortex core position is identified to within 0.25\% of the true position in the $x$- and to within 3\% of the true position in the $y$-direction.

Tilt-angles appear to be more difficult to identify correctly. Because the true value is 0° except in Case 4, no relative errors are reported. The absolute errors reach up to 12° in the Case 7 seed-void study. The deliberate vortex tilt in Case 4 is matched very well, to within $\pm 2.4°$. It is striking that the angle estimation performs worse for cases with an out-of-plane flow (Cases 3 and 7). Using the CFD2PIV tool, for future experiments comparable to the one in Reference [38], experimental parameters such as light sheet thickness, camera positions and the laser pulse separation duration can be optimized beforehand in order to mitigate this effect.
### Table 5.2.4: Results of the vortex identification, cases 2, 3, 4 and 7 ([25]). The parameters identified from synthetic PIV are the core-radius $R_c$, the circulation $\Gamma$, the $(x, y)$ position of the vortex core and the vortex tilt angles about the $x$ and $y$ axis, $\theta_x$ and $\theta_y$, respectively. The true values of the parameters are listed on top of the table, exceptions are shown in parentheses beneath the respective result.

<table>
<thead>
<tr>
<th></th>
<th>$R_c$ [mm]</th>
<th>$\Gamma$ [mm s$^{-1}$]</th>
<th>$x$ [mm]</th>
<th>$y$ [mm]</th>
<th>$\theta_x$ [$^\circ$]</th>
<th>$\theta_y$ [$^\circ$]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True value</strong></td>
<td>50.80</td>
<td>10000.00</td>
<td>-650</td>
<td>-50</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td><strong>Case 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow 1</td>
<td>25.95</td>
<td>10048.76</td>
<td>-649.90</td>
<td>-50.12</td>
<td>-1.29</td>
<td>4.84</td>
</tr>
<tr>
<td>(25.40)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow 2</td>
<td>50.68</td>
<td>9970.24</td>
<td>-649.98</td>
<td>-50.06</td>
<td>2.40</td>
<td>1.54</td>
</tr>
<tr>
<td>(76.20)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow 3</td>
<td>76.75</td>
<td>10004.00</td>
<td>-650.07</td>
<td>-49.98</td>
<td>3.71</td>
<td>1.08</td>
</tr>
<tr>
<td><strong>Case 3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow 1</td>
<td>49.71</td>
<td>9972.22</td>
<td>-649.40</td>
<td>-49.47</td>
<td>-3.44</td>
<td>7.91</td>
</tr>
<tr>
<td><strong>Case 4</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow 1</td>
<td>51.13</td>
<td>10000.64</td>
<td>-649.80</td>
<td>-50.06</td>
<td>2.20</td>
<td>7.40</td>
</tr>
<tr>
<td>(5.00)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow 2</td>
<td>51.16</td>
<td>9998.71</td>
<td>-650.07</td>
<td>-50.06</td>
<td>-0.59</td>
<td>16.11</td>
</tr>
<tr>
<td>(15.00)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow 3</td>
<td>51.27</td>
<td>10025.67</td>
<td>-650.37</td>
<td>-50.26</td>
<td>0.30</td>
<td>25.17</td>
</tr>
<tr>
<td>(25.00)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow 4</td>
<td>51.58</td>
<td>10072.93</td>
<td>-650.10</td>
<td>-49.97</td>
<td>14.91</td>
<td>13.84</td>
</tr>
<tr>
<td>(15.00)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Case 7</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow 1</td>
<td>50.33</td>
<td>9910.83</td>
<td>-650.86</td>
<td>-49.36</td>
<td>6.78</td>
<td>10.30</td>
</tr>
<tr>
<td>Flow 2</td>
<td>49.45</td>
<td>9987.06</td>
<td>-649.78</td>
<td>-49.28</td>
<td>6.59</td>
<td>9.38</td>
</tr>
<tr>
<td>Flow 3</td>
<td>50.67</td>
<td>9967.49</td>
<td>-649.21</td>
<td>-50.46</td>
<td>-5.78</td>
<td>-0.19</td>
</tr>
<tr>
<td>Flow 4</td>
<td>51.76</td>
<td>10036.77</td>
<td>-650.06</td>
<td>-50.77</td>
<td>-1.56</td>
<td>3.08</td>
</tr>
</tbody>
</table>
5.2. Vortex Identification Model Testing

<table>
<thead>
<tr>
<th>Flow</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
<th>Value 5</th>
<th>Value 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 5</td>
<td>51.58</td>
<td>10062.73</td>
<td>-650.55</td>
<td>-50.04</td>
<td>7.19</td>
<td>1.07</td>
</tr>
<tr>
<td>Flow 6</td>
<td>50.14</td>
<td>9890.02</td>
<td>-649.70</td>
<td>-50.67</td>
<td>6.99</td>
<td>9.45</td>
</tr>
<tr>
<td>Flow 7</td>
<td>49.91</td>
<td>10005.35</td>
<td>-649.74</td>
<td>-49.40</td>
<td>6.27</td>
<td>-0.85</td>
</tr>
<tr>
<td>Flow 8</td>
<td>50.57</td>
<td>9973.31</td>
<td>-649.85</td>
<td>-51.02</td>
<td>-12.01</td>
<td>9.39</td>
</tr>
<tr>
<td>Flow 9</td>
<td>50.01</td>
<td>10039.96</td>
<td>-649.52</td>
<td>-51.28</td>
<td>10.66</td>
<td>2.43</td>
</tr>
<tr>
<td>Flow 10</td>
<td>48.74</td>
<td>9862.18</td>
<td>-649.28</td>
<td>-49.77</td>
<td>-0.47</td>
<td>15.31</td>
</tr>
<tr>
<td>Flow 11</td>
<td>50.13</td>
<td>9936.10</td>
<td>-650.06</td>
<td>-49.86</td>
<td>-0.18</td>
<td>9.46</td>
</tr>
<tr>
<td>Flow 12</td>
<td>50.51</td>
<td>10028.18</td>
<td>-649.71</td>
<td>-50.48</td>
<td>5.82</td>
<td>4.05</td>
</tr>
<tr>
<td>Flow 13</td>
<td>51.23</td>
<td>10021.50</td>
<td>-650.06</td>
<td>-49.72</td>
<td>-1.23</td>
<td>8.96</td>
</tr>
<tr>
<td>Flow 15</td>
<td>49.58</td>
<td>9836.51</td>
<td>-650.27</td>
<td>-51.19</td>
<td>-1.64</td>
<td>13.35</td>
</tr>
<tr>
<td>Flow 16</td>
<td>50.83</td>
<td>9933.74</td>
<td>-650.45</td>
<td>-49.48</td>
<td>-7.75</td>
<td>4.69</td>
</tr>
<tr>
<td>Flow 17</td>
<td>51.10</td>
<td>10082.57</td>
<td>-650.14</td>
<td>-50.03</td>
<td>7.08</td>
<td>-10.23</td>
</tr>
<tr>
<td>Flow 18</td>
<td>51.43</td>
<td>9974.54</td>
<td>-651.54</td>
<td>-51.55</td>
<td>-2.66</td>
<td>6.86</td>
</tr>
<tr>
<td>Flow 19</td>
<td>52.97</td>
<td>10153.25</td>
<td>-651.12</td>
<td>-49.82</td>
<td>12.11</td>
<td>1.23</td>
</tr>
<tr>
<td>Flow 20</td>
<td>51.48</td>
<td>10104.74</td>
<td>-650.24</td>
<td>-49.71</td>
<td>-1.09</td>
<td>13.98</td>
</tr>
<tr>
<td>Flow 21</td>
<td>50.23</td>
<td>9891.30</td>
<td>-649.64</td>
<td>-51.27</td>
<td>3.39</td>
<td>9.65</td>
</tr>
<tr>
<td>Flow 22</td>
<td>53.02</td>
<td>10002.38</td>
<td>-649.69</td>
<td>-49.11</td>
<td>10.32</td>
<td>-3.83</td>
</tr>
<tr>
<td>Flow 23</td>
<td>53.49</td>
<td>10034.68</td>
<td>-648.98</td>
<td>-51.14</td>
<td>-0.80</td>
<td>-4.98</td>
</tr>
<tr>
<td>Flow 24</td>
<td>52.24</td>
<td>10061.04</td>
<td>-649.10</td>
<td>-49.78</td>
<td>-2.51</td>
<td>10.85</td>
</tr>
<tr>
<td>Flow 25</td>
<td>52.94</td>
<td>9998.02</td>
<td>-649.65</td>
<td>-50.81</td>
<td>3.77</td>
<td>6.00</td>
</tr>
</tbody>
</table>
Chapter 6. Test Case C: Analysis of Synthetic PIV from SST-DES of a Cylinder Wake with a Comparison to Experimental PIV Data

6. Test Case C: Analysis of Synthetic PIV from SST-DES of a Cylinder Wake with a Comparison to Experimental PIV Data

6.1. Experimental Data

As part of the “Fundamental Aero” series of experiments at the Fluid Mechanics Laboratory (FML) at NASA Ames, the wake flow of a cylinder in cross-flow was measured in a wind tunnel using PIV, and the data ([11]) were kindly made available for the present study. The tunnel conditions were 165 fps (56 m s\(^{-1}\)) free stream velocity in the test section, at a dynamic pressure of \(Q = 33\) psf (1580 Pa). The Reynolds number based on the cylinder diameter \(D = 1.5\) in (38.1 mm) was \(Re_D = 140,000\). An overview of the setup is given in Figure 6.1. The camera orientation to the cylinder is shown in Figure 6.2. Data were acquired at 1 kHz using Phantom 641 cameras. The field of view in object space was 405 mm \(\times\) 165 mm (w \(\times\) h). The sensor resolution was 2554 \(\times\) 998 px, which makes the object space resolution approximately 6 px/mm. Data were processed using LaVision 8.1.3, with an iterative processing which started at 64 \(\times\) 64 px down to 32 \(\times\) 32 px interrogation windows. The windows were chosen to overlap by 75%. On the last interrogation pass, a Gaussian weighting function was applied. The data were acquired in five batches of 1000 instances. Figure 6.3 shows the streamwise Mach number field of an exemplary instance. A number of frames were eliminated prior to processing due to lack of seeding, though not so many to cause the statistics to become skewed. The total came to 4810 instances that contributed to the statistical calculations.

6.2. CFD Data

The CFD simulations ([18]) used in this work to analyze the PIV performance were generated using NASA’s OVERFLOW2 solver [21]. In order to facilitate qualitative comparisons, the flow conditions were chosen to match the existing experimental dataset mentioned above from the FML at NASA Ames for a cylinder with diameter \(D = 1.5\) in at Mach number \(M = 0.2\) and Reynolds number \(Re_D = 140,000\). At this Reynolds number the experimental boundary layer on the cylinder is expected to be transitional, however no data on the boundary layer state were taken. The CFD simulations were treated as “fully turbulent” using the Shear-Stress Transport (SST)-Detached Eddy Simulation (DES) hybrid-RANS model. As such, there is no expectation that the experiment and CFD simulations agree, but we expect the qualitative features to be similar.
The CFD simulations used a wake mesh with isotropic spacing in the streamwise and transverse directions of $\Delta = D/225$. This corresponds to the identical pixel resolution of the experimental PIV system. The spanwise direction is treated as periodic with dimension $\pi$ and spacing $\pi \Delta$. The computational mesh contains a total of 400M grid points. Six spanwise stations of the isotropic wake box corresponding to the laser light-sheet plane were saved every 5.55 $\mu$s of the simulation, about half the pulse separation time of the experiment.
6.3. Synthetic PIV

**Figure 6.2.** Camera orientation to cylinder and lens plane ([11]). Note that the cylinder was 150 mm above the floor in proximity of the window. The laser propagated from the downstream mirror location.

**Figure 6.3.** Instantaneous experimental streamwise Mach number field ([11]).

6.3. Synthetic PIV

The parameters used to generate the synthetic data were chosen as close as possible to the experiment. While parameters like the pulse separation can be matched exactly, other parameters like the camera viewing angles were unknown.
6.4. Temporal Averages

6.3.1. Image Generation

The synthetic images were generated from the unmodified CFD data described in Section 6.2. Seeding the domain with 65,000 particles yielded a realistic seeding density, corresponding to approximately 0.06 particles per pixel. The complete set of parameters used to configure the CFD2PIV are listed in Table 6.3.1.

Table 6.3.1.: Configuration of the CFD2PIV program for the cylinder case.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of particles</td>
<td>65,000</td>
</tr>
<tr>
<td>Reference length</td>
<td>1.5 in</td>
</tr>
<tr>
<td>Reference Velocity</td>
<td>1126 fps</td>
</tr>
<tr>
<td>Pulse separation time</td>
<td>10.7 µs</td>
</tr>
<tr>
<td>Domain overlap</td>
<td>15% (in z direction)</td>
</tr>
</tbody>
</table>

The domain covered in the CFD computation is smaller than that imaged in the experimental setup. In order to operate with comparable magnifications, the image and object distances were adjusted manually to accommodate the simulated particle domain, which leaves about half of the created synthetic image blank. The parameters used to configure the synthetic image generation are listed in Table 6.3.2.

In order to generate dimensional results and–more importantly–to be able to correctly apply stereoscopic cross-correlation, the PIV software is supplied with synthetic calibration images.

6.3.2. PIV Processing

The synthetic data were processed using DAVIS’ LAVISION commercial PIV software in version 8.5.1. The processing was done using three different final pass window sizes: 32 x 32, 24 x 24 and 16 x 16 pixels. The full list of processing parameters is provided in Table 6.3.3. The RMS of fit of the calibration is reported by the software as 0.027 pixel.

6.4. Temporal Averages

An instantaneous comparison of experiment and simulation is prohibited by the impossibility to recreate the real-world initial conditions. Comparisons based on time-averaged quantities are, however, possible. The highly turbulent nature of the flow studied necessitates a large number of time-steps in order to obtain converged flow statistics. The periodic nature of the vortex
Table 6.3.2.: Configuration of the \textbf{SIG} for the cylinder case, $L_{ref} = 1.5$ in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>horizontal image dimension</td>
<td>2554px</td>
</tr>
<tr>
<td>vertical image dimension</td>
<td>996px</td>
</tr>
<tr>
<td>horizontal particle domain</td>
<td>$(0, 10.4987)L_{ref}$</td>
</tr>
<tr>
<td>vertical particle domain</td>
<td>$(-2.1391, 2.1391)L_{ref}$</td>
</tr>
<tr>
<td>light sheet type</td>
<td>gaussian</td>
</tr>
<tr>
<td>light sheet $z$ position</td>
<td>$0.035L_{ref}$</td>
</tr>
<tr>
<td>light sheet thickness</td>
<td>$0.0787L_{ref}$</td>
</tr>
<tr>
<td>particle size distribution</td>
<td>uniform</td>
</tr>
<tr>
<td>particle diameter</td>
<td>0.25 [-]</td>
</tr>
<tr>
<td>particle image intensity pattern</td>
<td>gaussian</td>
</tr>
<tr>
<td>projection angle</td>
<td>$\pm 5^\circ$</td>
</tr>
<tr>
<td>sensor tilt angle</td>
<td>$\pm 0.3121^\circ$</td>
</tr>
<tr>
<td>CCD fill ratio $(x,y)$</td>
<td>0.75</td>
</tr>
<tr>
<td>pixel pitch</td>
<td>10$\mu$m</td>
</tr>
<tr>
<td>object distance</td>
<td>$18.95L_{ref}$</td>
</tr>
<tr>
<td>image distance</td>
<td>$1.18L_{ref}$</td>
</tr>
</tbody>
</table>

Table 6.3.3.: Configuration of the PIV processing software for the cylinder case.

<table>
<thead>
<tr>
<th>Method</th>
<th>“stereo cross-correlation on GPU”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passes</td>
<td>multi-pass (decreasing size)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pass</th>
<th>Interrogation Window</th>
<th>Window shape</th>
<th>Overlap</th>
<th>Number of passes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64 $\times$ 64 px</td>
<td>square</td>
<td>50 %</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>32 $\times$ 32 px</td>
<td>circular</td>
<td>75 %</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>48 $\times$ 48 px</td>
<td>square</td>
<td>50 %</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>24 $\times$ 24 px</td>
<td>circular</td>
<td>75 %</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>32 $\times$ 32 px</td>
<td>square</td>
<td>50 %</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>16 $\times$ 16 px</td>
<td>circular</td>
<td>75 %</td>
<td>2</td>
</tr>
</tbody>
</table>

shedding also demands that as many shedding cycles as possible be present in the average. This means that there is a requirement not only for a high number of instances, but also for a high
6.4. Temporal Averages

total time in the time average. The theoretical convergence rate of descriptive statistics obtained from PIV images is described in [3, Section 9.4.2]. Unfortunately, the available CFD data did not fulfill this requirement. Because a computation of sufficient length with the necessary temporal and spatial resolution would have exceeded the time of the entire project, an alternative way of generating new realizations from the existing data was devised.

6.4.1. Oversampling via Re-Initialization

Given the time constraints of this project, it was not possible to get sufficient computational data at the high experiment Reynolds number (140,000) to demonstrate statistical convergence. Instead, each saved wake box was sub-integrated forward in time using characteristic boundary conditions. This reduced the computational effort substantially and provided statistical convergence in the wake region. While these statistical predictions are not physical, they are generated using the Navier-Stokes equations and turbulence model, and hence are representative of the physical processes. The initial data set was comprised of 313 samples. For each sample, 10 sub-integration instances were saved and treated as individual realizations at corresponding major time steps. The total number of samples thus came to 3443.

Convergence Comparison of Experimental and Synthetic Oversampled Temporal Averages

To validate the procedure, the convergence of different flow statistics of experiment and synthetic PIV from the computation are compared. The comparison is given in terms of the RMS of first-order differences of a series of averaged fields with successively increased sample size. The results are depicted in Figures 6.4 and 6.5. The absolute RMS of the synthetic, oversampled data is even lower than that of the experimental data. However, a direct comparison of the velocity fields shows that the averaged field provided by the experiment is more symmetric and appears to be smoother than those of the CFD and synthetic data (see Figures 6.8 and 6.7). This is to be expected in so far as due to the longer interval between individual measurements, many different states of the wake are included in the average. The samples from the CFD data, on the other hand, are separated only by a time step of the order of the pulse separation duration, and are thus strongly correlated.

6.4.2. Comparison of Experimental, CFD and Synthetic Temporal Averages

To perform quantitative comparisons between the data, extensive processing is necessary.

While the PIV processing software allows exporting the averaged results in the Tecplot format, the averages were not available for the CFD data. A parser for the binary CFD result format was
6.4. Temporal Averages

**Figure 6.4.** Convergence of synthetic PIV fields with sample size.

**Figure 6.5.** Convergence of experimental PIV fields with sample size.

therefore implemented in MATLAB and statistical moments of the flow fields of all realizations were computed.

The second challenge is to determine the offset of the coordinates of the different data: the coordinates of the synthetic data output by PIV processing are slightly shifted from the original CFD, the experiment has a different field of view (starting immediately downstream of the cylinder) all
6.4. Temporal Averages

together. To overcome this discrepancy, MATLAB’s numerical optimization function *fminsearch* was used to maximize the normalized cross-correlation of overlapping data segments:

(1) Because it is the highest resolution data, the CFD data are chosen to be interpolated to the locations of the other data. Linear interpolation is performed via MATLAB’s *interp3*. This is only possible for structured data.

(2) Let the correlation coefficient of two 2D fields $A$ and $B$ be defined as

$$
r(A, B) = \frac{1}{N_{x,y}} \sum_{x,y} \left( A(x,y) - \bar{A} \right) \left( B(x,y) - \bar{B} \right),$$

where $\bar{C}$ and $\sigma_C$ are the mean and standard deviation of field $C$, respectively and the sum is taken only over those coordinates at which both fields are defined, the total number of which being here denoted as $N_{x,y}$.

(3) Define the objective function

$$o(dx, dy) = -r(A, B(x + dx, y + dy)),$$

where $A$ is either the synthetic or experimental data, and $B$ is the CFD data, interpolated to coordinates $(x, y)$ at which $A$ is available, plus the offsets $(dx, dy)$ which are to be determined numerically.

(4) Numerical minimization of $o(dx, dy)$ yields the offsets $dx^*, dy^*$ that maximize the normalized cross-correlation of overlapping data segments, which were accepted as the optimal coordinate shifts.

The resultant averaged fields are shown in Figures 6.8, 6.6, and 6.7. The experimental results shown in Figures 6.8 exhibit a very slight upward skewing of the wake. This is due to the fact that the cylinder was mounted closer to the bottom wall than the top wall in the wind tunnel. As mentioned before, the CFD data is not intended to match the experiment quantitatively. Comparing the CFD data with the synthetic results, however, can provide a lower bound on PIV processing uncertainty for this flow situation, since the synthetic processing emulates a very idealized measurement. The percentage difference in Mach number in streamwise direction is shown in Figure 6.9. The highest error is present in the shear-layer, and reaches up to 10% of the average Mach number. This is certainly a consequence of the very steep velocity gradient and overall highest particle image displacement in this region. The velocity gradient can not be accurately resolved with the finite interrogation window size. It also introduces a bias error into the results (cf. Section 2.3). A Monte-Carlo study conducted by Westerweel [33] showed that the RMS error scales linearly with the image displacement. This error pattern is therefore to be expected. In the wake, the error is smaller than about ±5%, with higher values occurring
closer to the cylinder, again in the regions of higher overall velocity. Because stereoscopic PIV processing yields velocity data on a plane, each velocity vector corresponds to a volume average around the point where the vector is assigned. The deviation between the synthetic and the CFD data shown in Figure 6.9 is computed with respect to a single \((x, y)\)-plane of the CFD results. The shown deviation will thus be higher in regions of strong variation of the velocity with respect to the spanwise direction. This also explains a higher value of the error in the near wake.

![Average streamwise Mach number - synthetic data-](image)

**Figure 6.6:** Averaged streamwise velocity field from synthetic PIV.

### 6.4.3. Influence of the Interrogation Window Size

One of the most important processing parameters for the image interrogation is the interrogation window size. In order to analyze the impact of this parameter, the synthetic data were processed with three different window sizes. The scaling of the RMS percentage difference in streamwise Mach number is plotted in Figure 6.10. A strong reduction of this error with decreasing window size is observed. Because the highest errors are observed in regions of strong gradients, the reduction of these errors by using a higher resolution interrogation method is not surprising.

The ability to further reduce this error via further window size decrease is, however, limited by the correlation peak detectability (cf. [3, Section 8.3.5]: if the window is too small, the fraction of particles that will enter and leave corresponding interrogation windows in the images at the two time steps relative to the number of particles that remain inside will become too large for the
6.4. Temporal Averages

**Figure 6.7.** Averaged streamwise velocity field from CFD ([18]).

**Figure 6.8.** Averaged streamwise velocity field from the experiment ([11]).

algorithm to identify the maximum of the correlation function that corresponds to the actual mean particle displacement. Apparently, this limit was not yet reached with the $16 \times 16$ px window size for the synthetic seeding density.
6.4. Temporal Averages

Figure 6.9.: Difference between synthetic PIV and CFD in the averaged streamwise velocity component for 16 x 16 px interrogation window size.

Figure 6.10.: Scaling of the RMS difference in the averaged streamwise velocity component with interrogation window size.

The overall observed (averaged) RMS error of 1 – 2 % free stream velocity corresponds well with the value of 1–2 % maximum velocity (instantaneous) that is typically to be expected when using image interrogation via cross-correlation and sub-pixel interpolation (cf. Section 2.3).
6.5. Spatial Spectra

The analysis of spatial spectra is an important tool used to compare and assess the performance of PIV processing methods ([29], [3, Section 9.4.7], [7]).

As mentioned in Section 2.2.1, the effect of determining the particle displacement field via cross-correlation corresponds approximately to a volume average. This process is analogous to a low-pass spatial filtering. In the frequency domain, filtering is described by a multiplication of the Fourier transform of the true field with the Fourier transform of the weighting function of the volume average. For certain weighting functions, the Fourier transform is available analytically. For example, [3, Eq. 1.42] gives the frequency characteristic of a top-hat weighting function \( W \), corresponding to a pure volume average:

\[
\mathcal{F}\{\mathcal{W}\} = \frac{\sin(k_x D_Ix) \sin(k_y D_Iy) \sin(k_z \Delta z_0)}{(k_x D_Ix)(k_y D_Iy)(k_z \Delta z_0)},
\]  

(6.5.1)

where \( \mathcal{F}\{\cdot\} \) denotes the Fourier transform operator, \( \mathbf{k} = (k_x, k_y, k_z) \) is the wave number vector, \( D_Ix, D_Iy \) are the interrogation window sizes in the \( x \) and \( y \) direction, respectively, and \( \Delta z_0 \) is the laser light sheet thickness.

Because of this inherent relationship between filtering and PIV processing, the spatial spectra of the experiment, the CFD data and the synthetic PIV were compared. Note that for this comparison, it is assumed that all data is available on a regular grid. The quantity chosen for comparison was, again following the approach in [29], the streamwise fluctuating velocity component, its Power-Spectral Density (PSD) computed from the streamwise and cross-stream directions and averaged over the respective other direction. For example, the equation for the computation in streamwise direction is:

\[
\text{PSD}_{11} = \frac{1}{N_y} \sum_{i=1}^{N_y} \left\{ \frac{\Delta x}{N_x} \left( \sum_{j=1}^{N_x} u'_1(x_j, y_i) e^{-i\omega j} \right)^2 \right\}. 
\]

(6.5.2)

Equation 6.5.2 may be evaluated efficiently using the FFT algorithm:

\[
\text{PSD}_{11} = \frac{1}{N_y} \sum_{i=1}^{N_y} \left\{ \frac{\Delta x}{N_x} [\text{FFT}_x[u'_1(y_i)]]^2 \right\},
\]

(6.5.3)

where \( \text{FFT}_x[\cdot] \) denotes the FFT operator applied in the \( x \) (streamwise) direction. The power density spectrum is plotted over the wave number vector, which is given as \( \mathbf{k}_1 = \frac{2\pi}{\Delta x}[0, 0 + 1/(\Delta x), \ldots, 1] \).
6.5. Spatial Spectra

Because the flow itself is not periodic in the streamwise direction, its Fourier transform does not exist. Applying the FFT operator will cause spectral leakage generated because of the discontinuous step between beginning and end of the signal. A common technique to mitigate this problem in the context of finite signals is windowing [10]: the original data is multiplied by a function with compact support that goes to zero at the edge of the interval. For this comparison, a cosine-tapered window [10, Equation (38)] was chosen. It is defined as

\[
w(i) = \begin{cases} 
\frac{1}{2} \left[ 1 + \cos \left( \pi \left( \frac{2i}{\alpha(N_x - 1)} - 1 \right) \right) \right], & 0 \leq i \leq \frac{\alpha(N_x - 1)}{2} \\
\frac{1}{2} \left[ 1 + \cos \left( \pi \left( \frac{2i}{\alpha(N_x - 1)} - \frac{2}{\alpha} + 1 \right) \right) \right], & (N_x - 1)(1 - \frac{\alpha}{2}) \leq i \leq (N_x - 1) \\
1, & \text{otherwise.}
\end{cases}
\]  

(6.5.4)

The window function is shown in Figure 6.11. An example of the windowed experimental fluctuating velocity field is shown in Figure 6.12. The windowed field is smoothly reduced to zero at the streamwise edges in order to artificially make the field periodic in that direction.

For the computations in the cross-stream direction, this procedure is not necessary, as the flow approaches free-stream conditions at the edges anyway. However, because the frequency content of the cylinder wake changes as eddies are diffused while traveling downstream, the computation of the cross-stream PSD was split into two parts, the first, referred to as “near-field” hereafter, extends to about 2.5 cylinder diameters down-stream. The “far-field” part is consequently the remaining part of the domain.

The spatial spectra were averaged over 100 individual results, which were processed with 32 × 32 px interrogation window size, and an overlap of 75%, which means that there is a vector every 8 pixels.
6.5. Spatial Spectra

Figure 6.12.: Windowed experimental fluctuating streamwise Mach number.

The plots of the resulting spatial spectra are shown in Figures 6.13, 6.14 and 6.15 in double-logarithmic scale. Wave numbers corresponding to length scales of 32, 16 and 8 px, as well as the cylinder diameter $L$, are marked separately. Because the numerical results stem from a DES, no distinct filter width can be attributed to the CFD results. In the spectra of all three sources, the noise dominated scales are identifiable by the flattening (far-field) or even sharp increase (near-field) of the spectrum towards the higher end of the resolved wave-number region.

Figure 6.13.: Comparison of the streamwise power spectral density of the windowed fluctuating streamwise Mach number ($32 \times 32$ px interrogation window).

The most noticeable differences between experimental and synthetic spectra are in the near-field cross-stream spectrum, which is also less smooth than the others. This might be the effect of spectral leakage, i.e. a wave-number that is unresolved on the grid is projected onto the entire
6.5. Spatial Spectra

Figure 6.14.: Comparison of the cross-stream power spectral density of the fluctuating streamwise Mach number in the near-field (32 × 32 px interrogation window).

Figure 6.15.: Comparison of the cross-stream power spectral density of the fluctuating streamwise Mach number in the far-field (32 × 32 px interrogation window).

resolved spectrum, or there are indeed a large number of very similar-sized structures in the flow.

Although the computation is not intended as a reproduction of the experiment, the overall trend in the experimental spectrum is fairly well predicted by the synthetic data, indicating that indeed the PIV processing is the driving factor for the nature of the observed spatial spectrum.

Comparing synthetic data to the underlying CFD, it is evident that the synthetic spectrum does not exactly match that of the CFD up to the smallest filter width of 32 × 32 px, indicating that the “true” resolution, i.e. the resolution up to which flow features are accurately resolved, is
lower. Indeed, even the top-hat or pure volume average filter (Equation 6.5.1) would cause such an earlier departure of the spectrum from that of the unfiltered data.

Figures 6.16 and 6.17 show the cross-stream PSD of the synthetic data in the near- and far-field, respectively, for an interrogation window size of 16 × 16 px, together with the spectrum of the CFD data and the synthetic data processed with 32 × 32 px windows, for comparison. The data processed with the smaller window size shows a considerable improvement over the data processed with the larger. The higher resolution data matches the CFD spectrum better by about half an order of magnitude outside the noise dominated scales.

Stanislas et al. [29, Figure 16] (provided here for convenience as Figure 6.18) show similar trends for data obtained for a flow field computed from a predefined spectrum via DNS. Figure 6.18 shows power spectra obtained from synthetic PIV compared to the reference spectrum of the underlying DNS (black, “Ref.”). The cyan curve titled “LAVIS” was obtained by using an algorithm implemented in the software DaVis 7. Contrary to the algorithm used in this work
6.6. Velocity PDF Estimates

Figure 6.18.: [29, Figure 16]. The cyan curve (LAVIS) corresponds to the power spectrum obtained by processing synthetic PIV data using an algorithm by the same company that provides the software used in the analysis in this work. The black line is the reference spectrum of the underlying DNS.

(part of DaVis 8.1.5), adaptive window-deformation and filtering were employed. It is, however, the closest correspondence to this study that could be found in the literature. The widow size used was also $16 \times 16$ px, with an overlap of 50%. The departure of the synthetic spectrum from that of the underlying DNS at wave numbers above those corresponding to a 32 px length scale matches the findings presented here.

6.6. Velocity Probability Density Function (PDF) Estimates

Probability Density Function (PDF) estimates of the fluctuating streamwise Mach number were computed; the curves are averaged over 100 instantaneous results. To illustrate the changing wake characteristics depending on the distance from the cylinder, the domain was again split at a distance of about 2.5 cylinder diameters for the analysis.

Figures 6.19, 6.20, and 6.21 show the resultant PDF estimates for processing results using a final pass interrogation window size of $32 \times 32$ px. Although it should again be noted that the computation is not intended to exactly reproduce the experiment, the overall shapes of the curves bear similarities, especially for the part of the wake closer to the cylinder; and the change in shape of the PDF estimates from near- to far-field is also remarkably similar.

Comparing the result of the synthetic PIV to the CFD data, the most noticeable feature is a bias in the results, which is negative in the near-, and positive in the far-field. The bias in the entire field is again positive. This result is unexpected and warrants further investigation. A possible cause for this result might be an insufficiently accurate calibration.
6.7. Influence of Seeding Non-Uniformity on PIV Performance

Figure 6.19.: Comparison of the PDF estimates of the fluctuating streamwise Mach number (32 × 32 px interrogation window).

Figure 6.20.: Comparison of the PDF estimates of the fluctuating streamwise Mach number in the near-field (32 × 32 px interrogation window).

Figures 6.22, 6.23, and 6.24 show the same analysis, repeated with 16 × 16 px interrogation windows. The results appear to match considerably better than those obtained with the larger window size, especially in the far-field, where most of the bias error is no longer present. Nonetheless, a slight positive bias of the processed results compared to the underlying CFD data remains in all three PDF estimate comparisons.

6.7. Influence of Seeding Non-Uniformity on PIV Performance

To investigate the influence of seeding non-uniformity caused by inertial effects of the particles, images with different degrees of non-uniformity were produced according to the procedure de-
6.7. Influence of Seeding Non-Uniformity on PIV Performance

Figure 6.21.: Comparison of the PDF estimates of the fluctuating streamwise Mach number in the far-field (32 × 32 px interrogation window).

Figure 6.22.: Comparison of the PDF estimates of the fluctuating streamwise Mach number (16 × 16 px interrogation window).

scribed in Section 3.6. Figure 6.25 shows an example of synthetic non-uniform seeding in the investigated flow.

The scaling of the instantaneous RMS difference between the processed synthetic data and the underlying CFD is depicted in Figure 6.26. It is interesting to note that the error actually decreases for a small amount of non-uniformity, before rising sharply for higher degrees of vortex core de-seeding. This effect is probably due to the fact that the instantaneous differences between the synthetic data and the underlying CFD are most pronounced in regions of high stream-line curvature. These are incidentally the regions where the seeding non-uniformity is most pronounced. As long as the de-seeded regions do not reduce the amount of particles per
6.7. Influence of Seeding Non-Uniformity on PIV Performance

Figure 6.23.: Comparison of the PDF estimates of the fluctuating streamwise Mach number in the near-field (16 × 16 px interrogation window).

Figure 6.24.: Comparison of the PDF estimates of the fluctuating streamwise Mach number in the far-field (16 × 16 px interrogation window).

interrogation window below a critical level where the correlation starts to degrade, the resulting averaging may help in smoothing out the error. The results presented in Sections 6.4.3, 6.6, and 6.5 all show a decrease in error with decreasing interrogation window size, which is consistent with the observation that a sufficient number of particles is present for the interrogation with the 32 × 32 px window, even with a limited amount of de-seeding.
6.7. Influence of Seeding Non-Uniformity on PIV Performance

Figure 6.25.: Synthetic PIV image with 75 µs straight-line advection prior to stream-tracking of the particles.

Figure 6.26.: Scaling of the instantaneous RMS difference between synthetic PIV and CFD in the streamwise velocity component with seeding non-uniformity for the $32 \times 32$ px interrogation window size.
7. Conclusion

A new software to generate synthetic Particle Image Velocimetry images in conjunction with
the Synthetic Image Generator (SIG) program from both analytical flow fields and CFD results
on structured or unstructured meshes was designed and implemented. The program is fully
configurable via a settings file to make it suitable for the “multipurpose” applications it was
designed for. Consequently designed towards performance and with parallelization of the key
processing step, tracing $10^6$ particles with the program running on a model 2012 laptop is easily
feasible.

The ability of the tool to treat complex meshes was demonstrated by generating synthetic images
from CFD results of a tilt-rotor flow field given on a hybrid multi-level Cartesian/unstructured
mesh. Validation and improvement of a numerical procedure to identify vortices in rotor wakes
using the analytical-flow processing capability of the program is currently underway at the
U.S. Army Research, Development, and Engineering Command (RDECOM) Aeroflightdynamics
Directorate (AFDD) at NASA Ames. Preliminary results indicate that the vortex identification
capability is adversely affected by the presence of out-of-plane flow. With the new tool at
hand, experimental parameters such as light sheet thickness, camera positions and the laser
pulse separation duration can be optimized in order to mitigate this effect. The capability
to produce synthetic PIV from CFD generates new possibilities for flow-specific uncertainty
analysis. To demonstrate what such an analysis might look like, a large data set of Shear-Stress
Transport (SST)-Detached Eddy Simulation (DES) results from a cylinder wake were processed
and compared to experimental data. Studied parameters include interrogation window size and
seeding non-uniformity. It was found that using the smallest interrogation window size leads to
the least errors, and that velocity PDFs and turbulent spectra obtained from the results depend
strongly on the window size. Spectral analysis revealed that large differences in the experimental
and CFD spectra can largely be attributed to the effect of processing.

The presented analysis is not complete: only one velocity component was analyzed, no higher
order statistics were compared. The tool to perform these analyses, however, is now in place. For
future projects involving CFD validation via PIV data, the impact of processing can henceforth
be isolated. Future PIV experiments may be optimized specifically towards the flow they are
meant to study, before they are performed.

Conducting the full parameter study to obtain the PIV uncertainty has to be left to future work.
This might also include the addition of further vortex models to the analytical flow module of the
code, and the extension of the batch processing capability to results on hybrid meshes.
A. Source Code Listings

A.1. Bounding Box Construction for the Convex Hull Algorithm

```plaintext
! construct the i th xq centered
! permutation of the intervals
! [0,+inf],[ inf,0] for each dimension
!
! function build_bb(xq,i) result(bb)
!! the box coordinate origin
! real(kind = 8) :: xq(:)
!! the permutation to use
! integer :: i
!! the bounding box for the search
! type(interval) :: bb(size(xq))
!
! local
integer :: j, dec
!
bb%lower = huge(1._8)
bb%upper = +huge(1._8)
!
! obtain permutations of coordinate signs
! by conversion of i to binary
!
dec = i 1
!
! do j = 1, size(xq)
!! if (modulo(dec, 2) .gt. 0) then
!! bb(j)%lower = xq(j)
!! else
!! bb(j)%upper = xq(j)
!! end if
!
dec = floor(dec/2.)
```

A.2. N-linear Interpolation via Hypercube Reduction

Listing A.1: Construction of the i-th bounding box for the convex hull algorithm.

```
A.2. N-linear Interpolation via Hypercube Reduction

1  ! n linear interpolation in n d.
2  ! iteratively reduce an n dimensional
3  ! hypercube X to a single point xq.
4  ! returns the relative weights
5  ! of the original vertices.
6  !
7  ! function reduce_hc(xq,X) result(alpha)
8  ! real(kind = 8) :: xq(:)
9  ! real(kind = 8) :: X(:,2**ndim)
10  ! real(kind = 8) :: alpha(2**ndim)
11  !
12  function reduce_hc(xq,X) result(alpha)
13    real(kind = 8) :: xq(:) ! the query point
14    real(kind = 8) :: X(:,;:) ! the hypercube vertices
15    real(kind = 8) :: alpha(size(X,2)) ! the weights of the vertices
16
17    ! local
18    integer :: i, idx, cnt
19    integer :: ndim
20    integer :: nvrt
21
22    ! the hypercube reduction may be thought of as building a
23    ! binary tree from the leaves to the root.
24    ! each level in the tree corresponds to a hypercube of dimension equal to the
25    ! level depth (starting at 0).
26    ! 2 consecutive vertices (child nodes) form an edge. Projecting the query
27    ! point on that edge yields the parent node.
28    ! We store the distance complements of the children to their parents, which
29    ! are then combined to form the interpolation weights
30
31    ! let d be the number of dimensions
32    ! the tree height h is d + 1
33    ! the number of nodes in a binary tree is 2**h  
34    ! the number of vertices in a hypercube of dimension d is 2**d
35
36    ! the vertex coordinates
37    real(kind = 8) :: vrt (size(xq),2**(int(log(real(size(X,2)))/log(2.))+1) 1)

end function build_bb
```
A.2. N-linear Interpolation via Hypercube Reduction

! a vector of distances complements
real(kind = 8) :: d(size(vrts,2))

! two temporary variables holding coordinates of adjacent vertices
real(kind = 8) :: a(size(xq)), b(size(xq))

! the number of dimensions
ndim = int(log(real(size(X,2)))/log(2.))

! the number of vertices, including the new ones
! created during reduction
nvrt = 2**(ndim + 1) - 1

! initialize the weights
alpha = 1._8

if (size(X,2).eq. 1) return
if (size(X,2).ne. 2**ndim) call error('number of vertices input to hypercube
reduction must be power of 2')

! the first 2**d vertices are the support points
vrts(:,1:2**ndim) = X

! a counter for the edges of the hypercube
cnt = 1

! consecutive vertices constitute an edge
! iterate over edges and project query point
! consecutive new vertices constitute new edges
do i = 2**ndim+1,nvrt

! create a new vertex by projecting the query point
! onto an edge with endpoints a and b:

! the edge
a = vrts(:,cnt)
b = vrts(:,cnt+1)

! compute the (normalized) distance complement of the edge endpoints
! to their new parent vertex
d(cnt+1) = dot_product(xq,a,b) / dot_product(b,a,b)
d(cnt) = 1._8*d(cnt+1)

! the new vertex (itself just a linear interpolation of the coordinates)
vrts(:,i) = d(cnt)*a + d(cnt+1)*b

! increment the edge count
cnt = cnt + 2
A.2. N-linear Interpolation via Hypercube Reduction

Listing A.2: The n-dimensional linear interpolation via hypercube reduction.

```fortran
end do

! compute the vertex weights by traversing the tree
! from each leaf to the root
do i = 1, 2**ndim
  idx = i
  do while (idx .lt. nvrt) ! there should be exactly ndim iterations
    ! update the weight
    alpha(i) = alpha(i) * d(idx)
    ! find the index of the parent node
    idx = idx + 2**ndim floor(idx / 2.)
  end do
end do

! sanity check: the weights have to sum to 1
if (abs(sum(alpha) _8) .gt. 5.*eps) then
  write(*,'(E18.10)') alpha, sum(alpha)
  call error(&
    'hypercube reduction returned inconsistent weights')
end if
end function reduce_hc
```
References

“Scattering particle characteristics and their effect on pulsed laser measurements of fluid flow: speckle velocimetry vs particle image velocimetry”.  
(Cit. on p. 1).

“Twenty years of particle image velocimetry”.  
In: *Experiments in Fluids* 39.2 (2005), pp. 159–169  
(Cit. on p. 1).

*Particle image velocimetry*.  
Cambridge University Press, 2010  
(Cit. on pp. 1, 5 sqq., 46, 49, 52).

*Fundamentals of aerodynamics*.  
Vol. 2.  
(Cit. on pp. 18, 34).

“Generalized viscous vortex model for application to free-vortex wake and aeroacoustic calculations”.  
In: *58th American Helicopter Society Annual Forum 2002*.  
Vol. 58.  
2.  
American Helicopter Society International. June 2002,  
Pp. 2042–2057  
(Cit. on p. 16).
“Characterization of different PIV algorithms using the EUROPIV synthetic image generator and real images from a turbulent boundary layer”.
In: Particle Image Velocimetry: Recent Improvements. Springer, 2004,
Pp. 163–185
(Cit. on pp. 5, 8 sq.).

“Some considerations on the accuracy and frequency response of some derivative filters applied to particle image velocimetry vector fields”.
(Cit. on p. 52).

“Scattered data interpolation: tests of some methods”.
(Cit. on p. 13).

“Efficient FMM accelerated vortex methods in three dimensions via the Lamb-Helmholtz decomposition”. English.
(Cit. on p. 18).

“On the use of windows for harmonic analysis with the discrete Fourier transform”.
(Cit. on p. 53).

personal communication.
Wind Tunnel Division, Experimental Aero-Physics Branch at NASA Ames Research Center.
2013
(Cit. on pp. 41 sqq., 50).

Application of three-component PIV to a hovering rotor wake.
Tech. rep.
DTIC Document, 2000
(Cit. on p. 1).
In: *Experiments in Fluids* 52.6 (2012), pp. 1629–1639
(Cit. on p. 6).

“Optimization of Particle Image Velocimeters. 1. Double Pulsed Systems.”
(Cit. on pp. 1, 6).

*KDTREE 2: Fortran 95 and C++ software to efficiently search for near neighbors in a multi-dimensional Euclidean space.*
Aug. 2004
(Cit. on p. 16).

[16] Bertrand Lecordier and J Westerweel.
“The EUROPIV synthetic image generator (SIG)”.
In: *Particle Image Velocimetry: Recent Improvements.*
Springer, 2004,
Pp. 145–161
(Cit. on pp. 4, 9 sqq.).

“Vortex methods for flow simulation”.
In: *Journal of Computational Physics* 37.3 (1980), pp. 289–335
(Cit. on p. 17).

[18] Scott M. Murman.
personal communication.
NASA Advanced Supercomputing (NAS) Division, Fundamental Modeling and Simulation Branch at NASA Ames Research Center.
2013
(Cit. on pp. 41, 50).

“Monotone Multivariate Interpolation of Scattered Data Using Nested Hypercubes”.
(Cit. on p. 14).

“Flow in a viscous trailing vortex”.
(Cit. on p. 16).
References

“Solver and Turbulence Model Upgrades to OVERFLOW 2 for Unsteady and High-Speed Applications”.
In: Fluid Dynamics and Co-located Conferences.
American Institute of Aeronautics and Astronautics, June 2006
(Cit. on p. 41).

“Analysis of stereoscopic PIV measurements using synthetic PIV images”.
In: Particle Image Velocimetry: Recent Improvements.
Springer, 2004,
Pp. 199–210
(Cit. on pp. 9, 19).

“Stereoscopic particle image velocimetry”.
In: Experiments in Fluids 29.2 (2000), pp. 103–116
(Cit. on p. 3).

Particle Image Velocimetry: A Practical Guide.
2nd ed.
Springer, 2007
(Cit. on pp. 1, 6, 8).

personal communication.
2013
(Cit. on pp. 30, 38 sq.).

“Benchmarking PIV with LDV for Rotor Wake Vortex Flows”.
In: Fluid Dynamics and Co-located Conferences.
American Institute of Aeronautics and Astronautics, June 2006
(Cit. on pp. 6, 8).

[27] Andrea Sciacchitano, Bernhard Wienke, and Fulvio Scarano.
“PIV uncertainty quantification by image matching”.
(Cit. on p. 6).
References


[29] Michel Stanislas, K Okamoto, CJ Kähler, J Westerweel, and F Scarano. “Main results of the third international PIV challenge”. In: Experiments in fluids 45.1 (2008), pp. 27–71 (Cit. on pp. 1, 6, 9, 52, 56 sq.).


   personal communication.
   U.S. Army Research, Development, and Engineering Command (RDECOM), Aeroflight-
   dynamics Directorate at NASA Ames Research Center.
   2013
   (Cit. on pp. 25 sqq.).

[37] Andrew Wissink, Mark Potsdam, Venkateswaran Sankaran, Jayanarayanan Sitaraman,
   Zhi Yang, and Dimitri Mavriplis.
   “A coupled unstructured-adaptive Cartesian CFD approach for hover prediction”.
   Vol. 3.
   66.
   American Helicopter Society International. May 2010,
   Pp. 2612–2629
   (Cit. on p. 25).

[38] Gloria Yamauchi, Alan Wadcock, Wayne Johnson, and Manikandan Ramasamy.
   “Wind Tunnel Measurements of Full-Scale UH-60A Rotor Tip Vortices”.
   Vol. 1.
   68.
   American Helicopter Society International. May 2012,
   Pp. 569–593
   (Cit. on pp. 8, 29 sq., 38).