Journal of Fluids and Structures 23 (2007) 874–884

JOURNAL OF FLUIDS AND STRUCTURES

www.elsevier.com/locate/jfs

Recovering rotations in aeroelasticity

R. Ahrem[a,1], A. Beckert[b], H. Wendland[c,*,2]

[a]Institut für Numerische und Angewandte Mathematik, Universität Göttingen, Lotzestr. 16-18, D-37083 Göttingen, Germany
[b]European Aeronautic and Defence Company (EADS), Germany
[c]Department of Mathematics, University of Sussex, Brighton, BN1 9RF, UK

Received 12 December 2005; accepted 17 February 2007
Available online 20 April 2007

Abstract

We present a new scheme for loose coupling in fluid–structure-interaction problems as they typically appear in the context of aircraft design. This coupling scheme is based upon generalized multivariate scattered data interpolation and is tailored for small structural models, which bear in addition to deformations also rotational information. In contrast to classical coupling schemes, we also employ this rotational information to build a more accurate reconstruction.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Fluid–structure-interaction; Scattered data interpolation

1. Introduction

In computational aeroelasticity, one major goal is to describe the influence of structural deformations on the aerodynamic load distribution and *vice versa*. Deformations of an aircraft during flight may have severe consequences on the aerodynamic performance, maneuverability, and handling qualities (Försching, 1974). For this reason, the elasticity of an aircraft has to be taken into account during the early stages of design.

A main task concerning the treatment of coupled aeroelastic systems is the simulation of fluid–structure interaction (FSI). Research on FSI in the field of numerical aeroelastic simulation has recently strongly increased [see for example Beckert and Wendland (2001), Edward (1993), Ahrem et al. (2006), Farhat et al. (1998), Farhat and Lesoinne (1998), Försching (1994)]. An FSI method derives an adequate numerical distribution of aerodynamic loads at the structural nodes of the FE-model—using the aerodynamic pressures given in finite volumes, volume elements, or panels of the discretized flow-field or surface—as well as an adequate deformation of the aerodynamic shape, using the displacements and rotations given at the nodes of the FE-model.

In the literature, there exist two main formulations to describe the aeroelastic problem: the monolithic and the coupled field formulation (Farhat and Lesoinne, 1998; Kutler, 1993). In the first case, and especially for simple and small-scale structural problems, the fluid and the structural state equations are combined and treated as a single

*Corresponding author.
E-mail address: h.wendland@sussex.ac.uk (H. Wendland).

[1]Part of this work has been supported within the joint DLR Institute of Aeroelasticity and Fraunhofer Institute for Scientific Computing (SCAI) project AKITA.

[2]Large parts of this paper have been written during this author's stay at: Institut für Wissenschaftliches Rechnen, Mathematische Fakultät, TU Dresden, D-01062 Dresden.

monolithic system of equations. However, for complex aeroelastic problems, the fluid and the structural domains show different mathematical and numerical properties and require distinct numerical solvers. For the monolithic approach, a rewriting of the structural and fluid computer programs is necessary and the extension to other disciplines is difficult to realize. Consequently, the simultaneous solution by a monolithic scheme is in general computationally challenging, mathematically and economically suboptimal, and software-wise unmanageable (Farhat and Lesoinne, 1998).

In the non-monolithic approach, existing well-established numerical solvers can be used. The interaction between the fluid and structural codes is limited to the exchange of surface loads and surface deformation information using partitioned or staggered procedures (Farhat and Lesoinne, 1998). This approach allows an easier extension to multidisciplinary problems as they appear, for example, in aerothermoelasticity or aeroservoelasticity.

Usually the solution of dynamic aeroelastic problems needs a coupling in space and time, whereas in static problems only coupling in space is necessary. For coupling in time, partitioned or staggered solution procedures (Farhat and Lesoinne, 1998) are widely used. For coupling in space recently, both purely mathematically and physically motivated approaches (Cebral and Löhner, 1997; Harder and Desmarais, 1972; Hounjet and Meijer, 1994; Maman and Farhat, 1995) were presented.

Because both the aerodynamic and structural models are discretized in a physically different manner, they do not match at the boundary. This means that the models do not share the same grid points at their common boundary. Consequently, the structural discretization is not useful to model the aerodynamic shape since there are, in general, not enough elements or nodes to represent a sufficiently smooth surface. Of course, for the structural problem this is indeed not necessary. However, since the structural model is often smaller than the aerodynamic model by orders of magnitude, this can cause problems in defining a genuine deformation field. The remedy to this problem is either to employ structural models of a higher resolution, resulting in a longer computing time, or to employ more sophisticated reconstruction processes for the deformation field, incorporating also additional information such as the given rotations at the grid nodes.

Since loose coupling processes have been widely studied, we will concentrate on describing a new form of transferring deformations from the structural to the aerodynamic side.

Our new method aims in particular at the situation, where the structural model is much smaller than the aerodynamical model, which is often the case when beam- or bar-like structural models are employed. In such a situation, the structural model often consists of far less than a hundred structural nodes. However, for such models often not only deformations are given at the structural nodes but also rotations. Nonetheless, carrying the deformation over to the aerodynamic mesh means in such a situation that one has to deal not only with an interpolation but actually with an extrapolation scheme.

It is our goal, in contrast to previously derived interpolation methods, to incorporate also the recovery of rotations into the reconstruction process to produce a more accurate solution.

The reconstruction of rotations or torsions is well known in beam-spline structural models, which we will shortly review in the next section. Then, we will introduce the concept of generalized interpolation in Section 3 and describe our new coupling method in Section 4. In the final section, as an example, we apply our new method as well as a standard interpolation method by radial basis function interpolation to the AMP test wing (Hönlinger et al., 1991; Zingel et al., 1991) and an analytic deformation.

2. Classical beam-splines

In the univariate setting, the recovery of rotations is solved by employing beam splines. The idea behind beam splines can be described as follows.

Suppose we are given N univariate data sites $x_1, \ldots, x_N$. At these data sites we have function values $f_j = f(x_j)$ as well as derivative information $f'_j = f'(x_j)$ coming from an unknown function $f \in C^1(\mathbb{R})$. The goal now is to recover not only the function values but also the derivative information. This can be done by the general approach

$$s(x) = \sum_{j=1}^{N} \alpha_j \phi(x - x_j) + \sum_{j=1}^{N} \beta_j \phi'(x - x_j) + a_0 + a_1 x, \tag{1}$$

and the coefficients are determined by the $2N$ interpolation conditions

$$s(x_j) = f(x_j), \quad s'(x_j) = f'(x_j), \quad 1 \leqslant j \leqslant N,$$

and the additional constraints

$$\sum_{j=1}^{N} \alpha_j = 0, \quad \sum_{j=1}^{N} (\alpha_j x_j + \beta_j) = 0.$$

In classical beam-spline theory, ϕ is given by the cubic function $\phi(r) = |r|^3$, which gives the fundamental solution to the bi-Laplacian operator.

3. Generalized interpolation

To recover function values f_j at data sites $\mathbf{x}_j \in \mathbb{R}^d$, the classical interpolation scheme using radial basis functions starts with a conditionally positive definite kernel $\Phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.

Definition 3.1. A continuous function $\Phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called a conditionally positive definite kernel of order m if for any $N \in \mathbb{N}_0$, any distinct points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$, and any coefficient vector $\boldsymbol{\alpha} \in \mathbb{R}^N \backslash \{0\}$, satisfying

$$\sum_{j=1}^{N} \alpha_j p(\mathbf{x}_j) = 0$$

for all d-variate polynomials of degree $m-1$, the quadratic form

$$\sum_{j,k=1}^{N} \alpha_j \alpha_k \Phi(\mathbf{x}_j, \mathbf{x}_k)$$

is positive.

Then, one sets up an interpolant of the form

$$s(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \Phi(\mathbf{x}, \mathbf{x}_j) + p(\mathbf{x}),$$

where $p \in \pi_{m-1}(\mathbb{R}^d)$ is a d-variate polynomial of degree $m-1$, where m corresponds to the order of the kernel. To cope with the additional degrees of freedom, the interpolation conditions $s(\mathbf{x}_i) = f_j$, are enriched by the additional conditions

$$\sum_{j=1}^{N} \alpha_j p_k(\mathbf{x}_j) = 0, \quad 1 \leqslant k \leqslant Q,$$

where $p_1, \ldots, p_Q$ is a basis of the space of all d-variate polynomials of degree at most $m-1$. It is well known, that this interpolation problem has, under very mild conditions on the location of the data sites, a unique solution (Wendland, 2005).

However, from the last section and our goal to recover rotations, we can conclude that we have to extend the concept of classical interpolation in two ways. First of all, we want to allow more generalized functionals than point evaluation functionals. Second, for the reconstruction of rotations, the three components of the displacement field cannot be modeled independently any longer. Hence, we have to introduce vector-valued interpolants.

We start by defining the generalized interpolation problem. Suppose we are given N linearly independent functionals $\lambda_1, \ldots, \lambda_N : C^k(\mathbb{R}^d) \to \mathbb{R}$ and we want to recover values $f_j = \lambda_j(f)$ stemming from an unknown function $f \in C^k(\Omega)$.

Then, in accordance to Eq. (1) and by denoting the set of functionals by Λ, i.e. by setting $\Lambda = \{\lambda_1, \ldots, \lambda_N\}$, we form the interpolant as

$$s_{f,\Lambda}(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j^{\mathbf{y}} \Phi(\mathbf{x}, \mathbf{y}) + \sum_{k=1}^{Q} \beta_k p_k(\mathbf{x}), \tag{2}$$

where the superscript $\mathbf{y}$ indicates that λ_j is acting on Φ with respect to its second variable. The interpolation conditions now become

$$f_i \overset{!}{=} \lambda_i(s_{f,\Lambda}) = \sum_{j=1}^{N} \lambda_i^{\mathbf{x}} \lambda_j^{\mathbf{y}} \Phi(\mathbf{x}, \mathbf{y}) + \sum_{k=1}^{Q} \beta_k \lambda_i(p_k), \quad 1 \leqslant i \leqslant N. \tag{3}$$

Again, they are completed by orthogonality conditions, which now become

$$\sum_{j=1}^{N} \alpha_j \lambda_j(p_k) \overset{!}{=} 0, \quad 1 \leqslant k \leqslant Q. \tag{4}$$

It is well known, that this set-up has a unique solution provided that the functionals are unisolvent (see Wendland, 2005).

Theorem 3.2. *Suppose Φ is a conditionally positive definite kernel of order m. Suppose further, the functionals $\lambda_1, \dots, \lambda_N$ are linearly independent and $\pi_{m-1}(\mathbb{R}^d)$-unisolvent, which means that the only polynomial $p \in \pi_{m-1}(\mathbb{R}^d)$ with $\lambda_j(p) = 0$ for all j is given by $p = 0$. Then, there exists exactly one function* (2), *which satisfies both sets of conditions* (3) *and* (4).

4. Recovery of rotations

For small angles, rotations can be recovered from derivatives of translations via

$$\begin{pmatrix} \theta_x \\ \theta_y \\ \theta_z \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 & -\frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & -\frac{\partial}{\partial x} \\ -\frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \end{pmatrix} \begin{pmatrix} g_1 \\ g_2 \\ g_3 \end{pmatrix} =: \frac{1}{2} \nabla \times \mathbf{g}.$$

Hence, suppose for the N data sites $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^3$, we are given translations

$$\mathbf{g}(\mathbf{x}_i) = (g_1(\mathbf{x}_i), g_2(\mathbf{x}_i), g_3(\mathbf{x}_i))^{\mathrm{T}} \in \mathbb{R}^3, \quad 1 \leqslant i \leqslant N,$$

and rotations

$$\boldsymbol{\theta}(\mathbf{x}_i) = (\theta_1(\mathbf{x}_i), \theta_2(\mathbf{x}_i), \theta_3(\mathbf{x}_i))^{\mathrm{T}} \in \mathbb{R}^3, \quad 1 \leqslant i \leqslant N.$$

Then, we are looking for an interpolant $\mathbf{s} : \mathbb{R}^3 \to \mathbb{R}^3$ satisfying

$$\mathbf{s}(\mathbf{x}_i) = \mathbf{g}(\mathbf{x}_i), \quad 1 \leqslant i \leqslant N,$$
$$(\nabla \times \mathbf{s})(\mathbf{x}_i) = 2\boldsymbol{\theta}(\mathbf{x}_i), \quad 1 \leqslant i \leqslant N.$$

To put this into the framework of generalized interpolation, we first have to define the action of a vector-functional $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$ onto a vector-valued function. We simply set

$$\boldsymbol{\lambda}(\mathbf{s}) := \lambda_1(s_1) + \lambda_2(s_2) + \lambda_3(s_3), \tag{5}$$

[see Narcowich and Ward (1994)].

In our situation, this means that we have to define two sets of functionals, point evaluations and rotations. As usual, let us denote point evaluation of a function at a point $\mathbf{x}$ by $\delta_{\mathbf{x}}$. Then, the functionals for point evaluations are simply given by

$$\boldsymbol{\lambda}_{3j-2} := \begin{pmatrix} \delta_{\mathbf{x}_j} \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{\lambda}_{3j-1} := \begin{pmatrix} 0 \\ \delta_{\mathbf{x}_j} \\ 0 \end{pmatrix}, \quad \boldsymbol{\lambda}_{3j} := \begin{pmatrix} 0 \\ 0 \\ \delta_{\mathbf{x}_j} \end{pmatrix}, \tag{6}$$

for $1 \leqslant j \leqslant N$.

Because of definition (5), these functionals indeed satisfy the required conditions. For example, we have $\lambda_{3j-2}(\mathbf{s}) = s_1(\mathbf{x}_j) + 0 + 0 = s_1(\mathbf{x}_j)$.

Next, we define the functionals for the rotations for $1 \leqslant j \leqslant N$ by

$$\boldsymbol{\lambda}_{3(N+j)-2} = \begin{pmatrix} 0 \\ -\delta_{\mathbf{x}_j} \circ \frac{\partial}{\partial z} \\ \delta_{\mathbf{x}_j} \circ \frac{\partial}{\partial y} \end{pmatrix}, \quad \boldsymbol{\lambda}_{3(N+j)-1} = \begin{pmatrix} \delta_{\mathbf{x}_j} \circ \frac{\partial}{\partial z} \\ 0 \\ -\delta_{\mathbf{x}_j} \circ \frac{\partial}{\partial x} \end{pmatrix}, \quad \boldsymbol{\lambda}_{3(N+j)} = \begin{pmatrix} -\delta_{\mathbf{x}_j} \circ \frac{\partial}{\partial y} \\ \delta_{\mathbf{x}_j} \circ \frac{\partial}{\partial x} \\ 0 \end{pmatrix}. \tag{7}$$

Once again, these functionals give the correct representation. For example, we have

$$\boldsymbol{\lambda}_{3(N+j)-2}(\mathbf{s}) = 0 - \frac{\partial s_2}{\partial z}(\mathbf{x}_j) + \frac{\partial s_3}{\partial y}(\mathbf{x}_j),$$

which corresponds to the first component of $(\nabla \times \mathbf{s})(\mathbf{x}_j)$.

Using these functionals, we can set-up a vector-valued interpolant of the form

$$\begin{aligned}\mathbf{s}(\mathbf{x}) &= \sum_{j=1}^{6N} \alpha_j \boldsymbol{\lambda}_j^{\mathbf{y}} \begin{pmatrix} \Phi_1(\mathbf{x},\mathbf{y}) & 0 & 0 \\ 0 & \Phi_2(\mathbf{x},\mathbf{y}) & 0 \\ 0 & 0 & \Phi_3(\mathbf{x},\mathbf{y}) \end{pmatrix} + \mathbf{q}(\mathbf{x}) \\ &:= \sum_{j=1}^{6N} \alpha_j \boldsymbol{\lambda}_j^{\mathbf{y}} \boldsymbol{\Phi}(\mathbf{x},\mathbf{y}) + \mathbf{q}(\mathbf{x}), \end{aligned} \tag{8}$$

with a vector-valued polynomial $\mathbf{q}$ and a matrix-valued kernel $\Phi : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$. In this simplest form the matrix-valued kernel consists of a diagonal matrix with three (possibly different) basis functions $\Phi_i(\mathbf{x},\mathbf{y})$ on its diagonal. Note that, because of our definition of how a vector-valued functional acts on a vector-valued function, we need to set-up a matrix-valued kernel to allow the functionals to act on both variables of the kernel.

As a matter of fact, the matrix-valued kernel used in Eq. (8) can be replaced by a more arbitrary positive definite matrix-valued kernel $\Phi : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$, furthermore, additional requirements can be built into the kernel; see Narcowich and Ward (1994). However, in our situation, the simple kernel employed in Eq. (8) will suffice.

The action of a vector-valued functional to a matrix is defined by acting on its rows.

In component-wise form Eq. (8) becomes

$$s_1 = \sum_{j=1}^{N} \left[\alpha_{3j-2} \Phi_1(\cdot, \mathbf{x}_j) + \alpha_{3(N+j)-1} \frac{\partial_2 \Phi_1(\cdot, \mathbf{x}_j)}{\partial z} - \alpha_{3(N+j)} \frac{\partial_2 \Phi_1(\cdot, \mathbf{x}_j)}{\partial y} \right] + q_1,$$

$$s_2 = \sum_{j=1}^{N} \left[\alpha_{3j-1} \Phi_2(\cdot, \mathbf{x}_j) - \alpha_{3(N+j)-2} \frac{\partial_2 \Phi_2(\cdot, \mathbf{x}_j)}{\partial z} + \alpha_{3(N+j)} \frac{\partial_2 \Phi_2(\cdot, \mathbf{x}_j)}{\partial x} \right] + q_2,$$

$$s_3 = \sum_{j=1}^{N} \left[\alpha_{3j} \Phi_3(\cdot, \mathbf{x}_j) + \alpha_{3(N+j)-2} \frac{\partial_2 \Phi_3(\cdot, \mathbf{x}_j)}{\partial y} - \alpha_{3(N+j)-1} \frac{\partial_2 \Phi_3(\cdot, \mathbf{x}_j)}{\partial x} \right] + q_3,$$

where the additional index 2 in ∂_2 indicates that the derivative of Φ_k has to be taken with respect to the second argument.

In general, if no additional information is given, one can simply choose one scalar-valued kernel $\Phi_1 = \Phi_2 = \Phi_3 =: \varphi$.

If Φ is conditionally positive definite of order m, the polynomials q_1, q_2, q_3 are chosen from $\pi_{m-1}(\mathbb{R}^3)$, resulting in another set of $3Q$ coefficients.

To determine all $6N + 3Q$ coefficients, we have for $1 \leqslant i \leqslant N$ the $6N$ interpolation conditions

$$\begin{aligned} &\boldsymbol{\lambda}_{3i-2}(\mathbf{s}) = g_1(\mathbf{x}_i), \quad \boldsymbol{\lambda}_{3i-1}(\mathbf{s}) = g_2(\mathbf{x}_i), \quad \boldsymbol{\lambda}_{3i}(\mathbf{s}) = g_3(\mathbf{x}_i), \\ &\boldsymbol{\lambda}_{3(N+i)-2}(\mathbf{s}) = 2\theta_1(\mathbf{x}_i), \quad \boldsymbol{\lambda}_{3(N+i)-1}(\mathbf{s}) = 2\theta_2(\mathbf{x}_i), \quad \boldsymbol{\lambda}_{3(N+i)}(\mathbf{s}) = 2\theta_3(\mathbf{x}_i), \end{aligned} \tag{9}$$

and the $3Q$ orthogonality conditions

$$\sum_{j=1}^{6N} \alpha_j \boldsymbol{\lambda}_j(\mathbf{p}) = 0, \quad \mathbf{p} = p_i \mathbf{e}_k, \quad 1 \leqslant i \leqslant Q, \quad 1 \leqslant k \leqslant 3, \tag{10}$$

where $\mathbf{e}_k$ is the kth unit vector in $\mathbb{R}^3$.

5. More details on the interpolation matrix

As is the case with all interpolation processes, we can write our particular interpolation problem as a linear system of the form

$$\begin{pmatrix} A_{\Phi,X} & P_X \\ P_X^{\mathrm{T}} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \mathbf{r} \\ \mathbf{0} \end{pmatrix}. \tag{11}$$

Here, the block matrices and vectors have the following dimensions: $A_{\Phi,X} \in \mathbb{R}^{6N\times 6N}$, $P_X \in \mathbb{R}^{6N\times 3Q}$, $\boldsymbol{\alpha} \in \mathbb{R}^{6N}$, $\boldsymbol{\beta} \in \mathbb{R}^{3Q}$, $\mathbf{r} \in \mathbb{R}^{6N}$, $0 \in \mathbb{R}^{3Q\times 3Q}$, and $\mathbf{0} \in \mathbb{R}^{3Q}$.

It is now our goal to describe the entries of the matrices in more detail. To this end, let us set

$$\mathbf{q} = \sum_{j=1}^{Q} [\beta_{3j-2}\mathbf{p}_{3j-2} + \beta_{3j-1}\mathbf{p}_{3j-1} + \beta_{3j}\mathbf{p}_{3j}]$$

with

$$\mathbf{p}_{3j-2} = \begin{pmatrix} p_j \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{p}_{3j-1} = \begin{pmatrix} 0 \\ p_j \\ 0 \end{pmatrix}, \quad \mathbf{p}_{3j} = \begin{pmatrix} 0 \\ 0 \\ p_j \end{pmatrix}.$$

We compute the matrix entries $A_{ij} = \lambda_i^{\mathbf{x}}\lambda_j^{\mathbf{y}}\Phi(\mathbf{x},\mathbf{y})$ of $A = A_{\Phi,X}$ according to the previously introduced indices, using also the notation

$$\Phi(\mathbf{x},\mathbf{y}) = \begin{pmatrix} \Phi_1(\mathbf{x},\mathbf{y}) & 0 & 0 \\ 0 & \Phi_2(\mathbf{x},\mathbf{y}) & 0 \\ 0 & 0 & \Phi_3(\mathbf{x},\mathbf{y}) \end{pmatrix}.$$

Using the previous way of indices, straightforward computations show that the matrix entries of A fall into different blocks, which we summarize now as

$$\begin{aligned}
&A_{3i-2,3j-2} = \Phi_1(\mathbf{x}_i,\mathbf{x}_j), \quad A_{3i-2,3(N+j)-2} = 0,\\
&A_{3i-2,3j-1} = 0, \quad A_{3i-2,3(N+j)-1} = \frac{\partial_2}{\partial z}\Phi_1(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3i-2,3j} = 0, \quad A_{3i-2,3(N+j)} = -\frac{\partial_2}{\partial y}\Phi_1(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3i-1,3j-2} = 0, \quad A_{3i-1,3(N+j)-2} = -\frac{\partial_2}{\partial z}\Phi_2(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3i-1,3j-1} = \Phi_2(\mathbf{x}_i,\mathbf{x}_j), \quad A_{3i-1,3(N+j)-1} = 0,\\
&A_{3i-1,3j} = 0, \quad A_{3i-1,3(N+j)} = \frac{\partial_2}{\partial x}\Phi_2(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3i,3j-2} = 0, \quad A_{3i,3(N+j)-2} = \frac{\partial_2}{\partial y}\Phi_3(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3i,3j-1} = 0, \quad A_{3i,3(N+j)-1} = -\frac{\partial_2}{\partial x}\Phi_3(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3i,3j} = \Phi_3(\mathbf{x}_i,\mathbf{x}_j), \quad A_{3i,3(N+j)} = 0,\\
&A_{3(N+i)-2,3j-2} = 0, \quad A_{3(N+i)-2,3(N+j)-2} = \frac{\partial_1}{\partial z}\frac{\partial_2}{\partial z}\Phi_2(\mathbf{x}_i,\mathbf{x}_j) + \frac{\partial_1}{\partial y}\frac{\partial_2}{\partial y}\Phi_3(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3(N+i)-2,3j-1} = -\frac{\partial_1}{\partial z}\Phi_2(\mathbf{x}_i,\mathbf{x}_j), \quad A_{3(N+i)-2,3(N+j)-1} = -\frac{\partial_1}{\partial y}\frac{\partial_2}{\partial x}\Phi_3(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3(N+i)-2,3j} = \frac{\partial_1}{\partial y}\Phi_3(\mathbf{x}_i,\mathbf{x}_j), \quad A_{3(N+i)-2,3(N+j)} = -\frac{\partial_1}{\partial z}\frac{\partial_2}{\partial x}\Phi_2(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3(N+i)-1,3j-2} = \frac{\partial_1}{\partial z}\Phi_1(\mathbf{x}_i,\mathbf{x}_j), \quad A_{3(N+i)-1,3(N+j)-2} = -\frac{\partial_1}{\partial x}\frac{\partial_2}{\partial y}\Phi_3(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3(N+i)-1,3j-1} = 0, \quad A_{3(N+i)-1,3(N+j)-1} = \frac{\partial_1}{\partial z}\frac{\partial_2}{\partial z}\Phi_1(\mathbf{x}_i,\mathbf{x}_j) + \frac{\partial_1}{\partial x}\frac{\partial_2}{\partial x}\Phi_3(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3(N+i)-1,3j} = -\frac{\partial_1}{\partial x}\Phi_3(\mathbf{x}_i,\mathbf{x}_j), \quad A_{3(N+i)-1,3(N+j)} = -\frac{\partial_1}{\partial z}\frac{\partial_2}{\partial y}\Phi_1(\mathbf{x}_i,\mathbf{x}_j),\\
&A_{3(N+i),3j-2} = -\frac{\partial_1}{\partial y}\Phi_1(\mathbf{x}_i,\mathbf{x}_j), \quad A_{3(N+i),3(N+j)-2} = -\frac{\partial_1}{\partial x}\frac{\partial_2}{\partial y}\Phi_2(\mathbf{x}_i,\mathbf{x}_j),
\end{aligned}$$

$$A_{3(N+i),3j-1} = \frac{\partial_1}{\partial x}\Phi_2(\mathbf{x}_i,\mathbf{x}_j), \quad A_{3(N+i),3(N+j)-1} = -\frac{\partial_1}{\partial y}\frac{\partial_2}{\partial z}\Phi_1(\mathbf{x}_i,\mathbf{x}_j),$$
$$A_{3(N+i),3j} = 0, \quad A_{3(N+i),3(N+j)} = \frac{\partial_1}{\partial y}\frac{\partial_2}{\partial y}\Phi_1(\mathbf{x}_i,\mathbf{x}_j) + \frac{\partial_1}{\partial x}\frac{\partial_2}{\partial x}\Phi_2(\mathbf{x}_i,\mathbf{x}_j).$$

Note that we have the following symmetries:

$$\Phi_k(\mathbf{x},\mathbf{y}) = \Phi_k(\mathbf{y},\mathbf{x}), \quad \frac{\partial_1}{\partial u}\Phi_k(\mathbf{x},\mathbf{y}) = \frac{\partial_2}{\partial u}\Phi_k(\mathbf{y},\mathbf{x}), \quad \frac{\partial_1}{\partial u}\frac{\partial_2}{\partial v}\Phi_k(\mathbf{x},\mathbf{y}) = \frac{\partial_1}{\partial v}\frac{\partial_2}{\partial u}\Phi_k(\mathbf{x},\mathbf{y}),$$

and so on. Here, to avoid writing down too many equations, we have denoted the derivatives with respect to the generic variables u and v, which can be replaced by any of the coordinates x, y, and z. Moreover, if $\Phi_1(\mathbf{x},\mathbf{y}) = \Phi_2(\mathbf{x},\mathbf{y}) = \Phi_3(\mathbf{x},\mathbf{y}) = \varphi(\mathbf{x}-\mathbf{y})$, which is the case for all radial basis functions, only the following information on φ and its derivatives are necessary:

$$\begin{aligned}
&\varphi(\mathbf{x}_i - \mathbf{x}_j),\\
&\frac{\partial}{\partial x}\varphi(\mathbf{x}_i - \mathbf{x}_j), \quad \frac{\partial}{\partial y}\varphi(\mathbf{x}_i - \mathbf{x}_j), \quad \frac{\partial}{\partial z}\varphi(\mathbf{x}_i - \mathbf{x}_j),\\
&\frac{\partial^2}{\partial x^2}\varphi(\mathbf{x}_i - \mathbf{x}_j), \quad \frac{\partial^2}{\partial y^2}\varphi(\mathbf{x}_i - \mathbf{x}_j), \quad \frac{\partial^2}{\partial z^2}\varphi(\mathbf{x}_i - \mathbf{x}_j),\\
&\frac{\partial^2}{\partial x\,\partial y}\varphi(\mathbf{x}_i - \mathbf{x}_j), \quad \frac{\partial^2}{\partial x\,\partial z}\varphi(\mathbf{x}_i - \mathbf{x}_j), \quad \frac{\partial^2}{\partial y\,\partial z}\varphi(\mathbf{x}_i - \mathbf{x}_j),
\end{aligned}$$

i.e. all derivatives up to order two. In particular, setting up this interpolation problem works only if basis functions are employed that are at least $C^2(\mathbb{R}^3)$.

For the matrix $P = P_X$ we derive similarly

$$\begin{aligned}
&P_{3i-2,3j-2} = p_j(\mathbf{x}_i), \quad P_{3i-2,3j-1} = 0, \quad P_{3i-2,3j} = 0,\\
&P_{3i-1,3j-2} = 0, \quad P_{3i-1,3j-1} = p_j(\mathbf{x}_i), \quad P_{3i-1,3j} = 0,\\
&P_{3i,3j-2} = 0, \quad P_{3i,3j-1} = 0, \quad P_{3i,3j} = p_j(\mathbf{x}_i),
\end{aligned}$$

and

$$\begin{aligned}
&P_{3(N+i)-2,3j-2} = 0, \quad P_{3(N+i)-2,3j-1} = -\frac{\partial}{\partial z}p_j(\mathbf{x}_i), \quad P_{3(N+i)-2,3j} = \frac{\partial}{\partial y}p_j(\mathbf{x}_i),\\
&P_{3(N+i)-1,3j-2} = \frac{\partial}{\partial z}p_j(\mathbf{x}_i), \quad P_{3(N+i)-1,3j-1} = 0, \quad P_{3(N+i)-1,3j} = -\frac{\partial}{\partial x}p_j(\mathbf{x}_i),\\
&P_{3(N+i),3j-2} = -\frac{\partial}{\partial y}p_j(\mathbf{x}_i), \quad P_{3(N+i),3j-1} = \frac{\partial}{\partial x}p_j(\mathbf{x}_i), \quad P_{3(N+i),3j} = 0.
\end{aligned}$$

This defines the interpolation matrix. The right-hand side $\mathbf{r} \in \mathbb{R}^{6N}$ of system (11) is given by

$$\begin{aligned}
&\mathbf{r}_{3i-2} = g_1(\mathbf{x}_i), \quad \mathbf{r}_{3i-1} = g_2(\mathbf{x}_i), \quad \mathbf{r}_{3i} = g_3(\mathbf{x}_i),\\
&\mathbf{r}_{3(N+i)-2} = 2\theta_1(\mathbf{x}_i), \quad \mathbf{r}_{3(N+i)-1} = 2\theta_2(\mathbf{x}_i), \quad \mathbf{r}_{3(N+i)} = 2\theta_3(\mathbf{x}_i).
\end{aligned}$$

Some remarks are necessary.

First of all, since the dimension of the problem grows now by $6N$ rather than by N, recovery of rotations is expensive and should only be used with small models.

For small models, however, one cannot expect an interpolated displacement field to behave perfectly in extrapolating points. This is inherent to the little information that is given and is almost independent of the chosen reconstruction. Hence, in particular for small models it is crucial to incorporate as much information into the reconstruction process as possible, which means that the additional recovery of rotations can yield a better reconstruction.

Though we have restricted ourselves to evaluating the interpolant at the fluid points, it is also possible to derive an approximation to the rotation at the fluid (or arbitrary) points by taking the curl of the interpolant, i.e. by forming

$$\nabla \times \mathbf{s} = \sum_{j=1}^{6N} \alpha_j \nabla_1 \times \boldsymbol{\lambda}_j^{\mathbf{y}} \Phi(\cdot,\mathbf{y}) + \nabla \times \mathbf{q}.$$

This, can for example be used to verify that the interpolant indeed interpolates also the rotations at the structural points.

6. An example

We apply the new interpolation scheme to transfer displacements from the structural to the aerodynamic model of the AMP (Aeroelastic Model Program) test wing (Hönlinger et al., 1991; Zingel et al., 1991).

The structural model of the AMP test wing consists of a single, slightly curved beam with 25 nodes and 24 line elements. In contrast, the aerodynamical model consists of a highly resolved surface of 54 653 nodes and 109 216 triangles. The coordinate system is as follows. The x-axis is along the root chord, the y axis is normal to the root chord, pointing roughly in the same direction as the wing, and the z axis is perpendicular to the wing.

To have a feeling for the dimensions, the bounding boxes of both grids are roughly given by:

(i) CFD mesh: $[0, 0.75] \times [0, 1.06] \times [-0.05, 0.02]$,
(ii) FEM mesh: $[0.16, 0.70] \times [0, 1.05] \times [-0.02, 0.01]$

Fig. 1 shows the aerodynamic mesh both as a surface and as the unstructured triangle grid. Fig. 2 shows the beam-like structural grid and its position within the aerodynamic grid.

From this given configuration it is obvious that we not only have to deal with a very small structural model, but also have to face the problem of *extrapolation*, i.e. evaluating the interpolant quite far away from the given information. The beam being almost two-dimensional adds further difficulties.

Nonetheless, it is the purpose of this section to show that, for such a reduced structural model, employing rotational information improves the reconstruction of the deformation field significantly.

For testing purposes, we applied the following simple, analytic displacements to the structural grid:

$$\mathbf{g}(\mathbf{x}) = \left(0, 0, 0.1y^2 + 0.1x\right)^{\mathrm{T}}, \tag{12}$$

which induces the rotations

$$2\boldsymbol{\theta}(\mathbf{x}) = \nabla \times \mathbf{g}(\mathbf{x}) = (0.2y, -0.1, 0)^{\mathrm{T}}. \tag{13}$$

In Fig. 3, the analytical result of this deformation when applied to the CFD mesh is presented. For reasons of comparison, the figure contains both the undeformed and the deformed CFD surface. Obviously, the maximum deflection is at the wing tip and it is of size 0.186021.

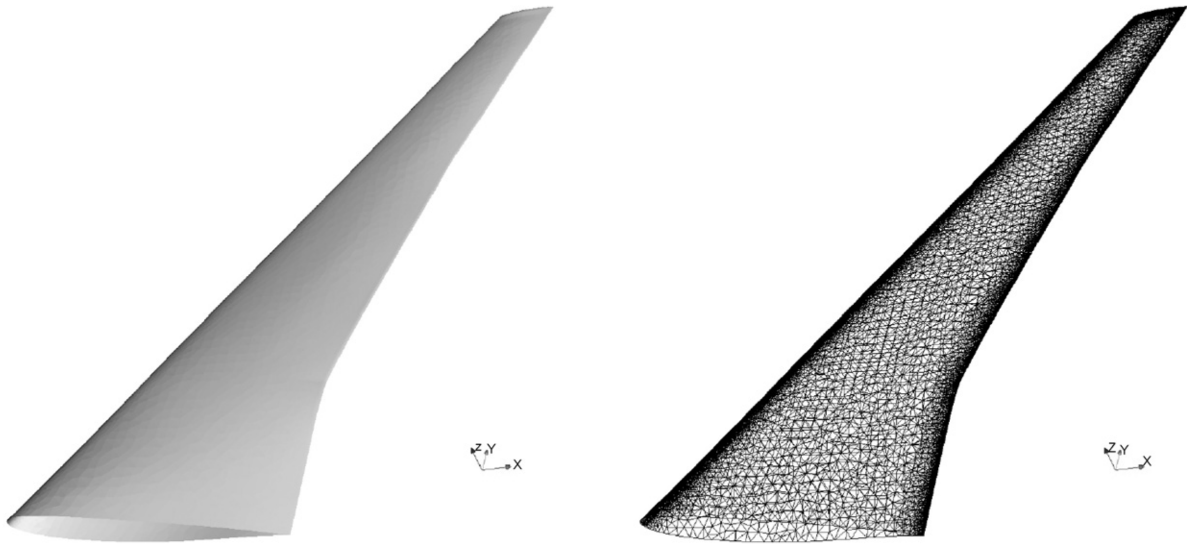

Fig. 1. The aerodynamic grid.

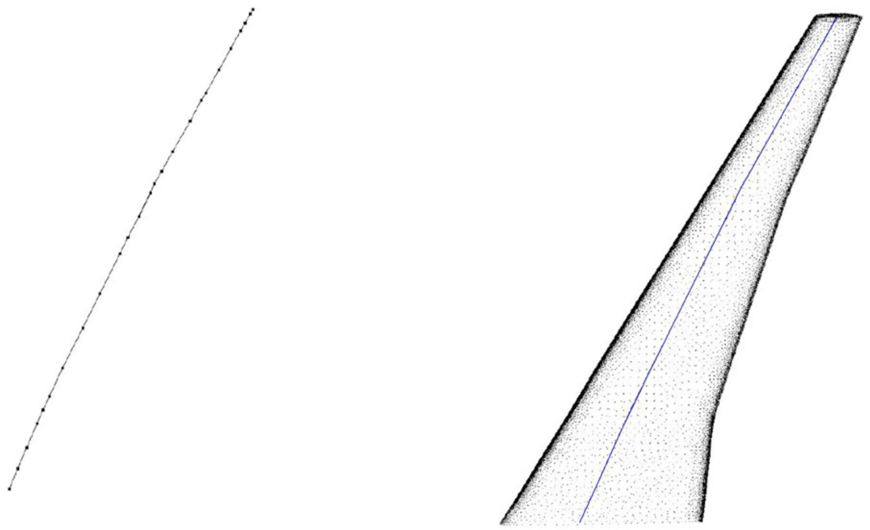
Fig. 2. The structural grid. The right part of the figure shows both meshes relative to each other.

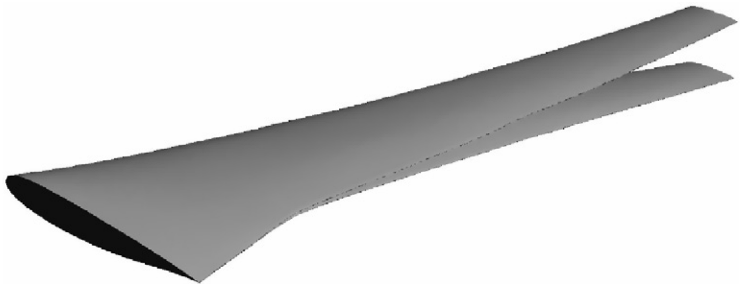
Fig. 3. The results of the analytical deformation applied to the CFD surface.

Naturally, such a simple deflection as the one given in Eq. (12) can also exactly be recovered by other, simpler reconstruction processes. Furthermore, classical interpolation by radial basis functions and our new approach recover this deformation exactly as soon as quadratic polynomials are added to the interpolant. However, our goal is *not* to use any information on the displacement field, since it will, in general, not be at hand in real-world applications. We have chosen this simple deformation field since it induces a simple non-constant rotation and hence allows us to study the error of our method analytically, without being too far away from a realistic deformation field.

We have applied the classical interpolation scheme based upon radial basis functions to the displacement field generated by Eq. (12) as well as our new generalized interpolation scheme incorporating also the rotations given by Eq. (13). To ensure the exact reconstruction of rigid body motions, we also included linear polynomials in both cases.

In both cases, we used the same radial basis function, the Gaussian $\varphi(\mathbf{x}) = \exp(-\|\mathbf{x}\|_2^2/\delta^2)$ with a "support" radius $\delta = 2.1$, which is roughly twice the length of the wing. Though the choice of the radius has an influence on the error, several tests executed with different radii showed that the new method is always significantly superior to the classical interpolation method. Moreover, to avoid ill-conditioning, we stabilized the interpolation matrices by adding $\varepsilon = 10^{-10}$ to the diagonal of the non-polynomial part of the matrix. This is a well-known method for stabilizing ill-conditioned radial basis function matrices by *smoothing*; see Wahba (1990) and Wendland and Rieger (2005). The theoretical background for this in the case of our new interpolation method is still under investigation. However, it works numerically very well and there exists already some theoretical backup in the case of other generalized interpolation matrices (Wendland, 2007).

Fig. 4 shows the errors for both methods on the CFD mesh, i.e. the differences between the analytic deformation and the one computed. Clearly, in both cases the error is largest close to the root of the wing and at the leading and trailing

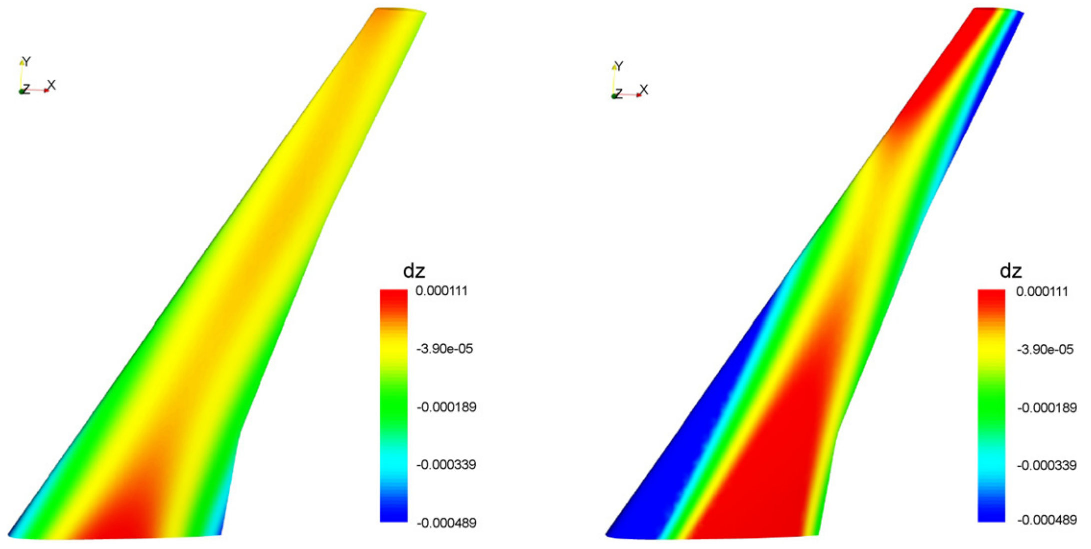

Fig. 4. Results for the AMP test wing with (left) and without (right) the reconstruction of rotations.

Table 1
Errors for deformation and rotation on the CFD surface

Method	e-Type	dx	dy	dz	dr
curl	fval	1.788694×10^{-4}	9.264122×10^{-5}	4.888944×10^{-4}	5.287669×10^{-4}
curl	curl	2.958215×10^{-3}	1.085164×10^{-3}	7.723361×10^{-4}	3.244244×10^{-3}
std	fval	0	0	1.802852×10^{-3}	1.802852×10^{-3}
std	curl	2.047974×10^{-1}	7.286678×10^{-2}	0	2.173742×10^{-1}

edge, where structural and aerodynamic meshes differ most. But it is also apparent that the new method is much more capable of recovering the caused rotational deformations at the leading and trailing edge of the aerodynamic model.

Next, Table 1 contains the maximum errors. Here, the first column indicates the employed methods and the second column refers to the type of errors which are listed in the following columns. To be more precise fval stands for measuring the error of the deformations, while curl means measuring the error of the rotations of the deformation field; both are measured on the CFD mesh. The columns denoted by dx, dy, dz give the maximum error on the CFD mesh in direction x, y, z. Finally, dr gives the Euclidean norm of this error.

Note that the important quantities are dz for fval and dx and dy for curl, since we only have a deformation or rotational part in these components. This is also the reason why the standard method has zero error in all other components, since the constant zero is exactly reproduced. However, since the new method models all directions dependently, this is no longer the case here. But note that the error is always dominated by the previously mentioned main quantities.

Clearly, the classical method has problems close to the leading and trailing edge of the wing. On the other hand, the new method produces much better and more reliable results. Nonetheless, when evaluating interpolants in extrapolation regions, special care is always necessary, since the coarse spatial resolution of the structural grid allows, at least theoretically, only to hope for a limited accuracy.

What remains is to demonstrate the superiority of the new method in a complete aeroelastic computation. This, however, will be the subject of further research.

Acknowledgement

The authors are grateful to the anonymous referees for their hints and suggestions.

References

Ahrem, R., Beckert, A., Wendland, H., 2006. A meshless spatial coupling scheme for large-scale fluid-structure-interaction problems. Computer Modeling in Engineering & Sciences 12, 121–136.

Beckert, A., Wendland, H., 2001. Multivariate interpolation for fluid-structure-interaction problems using radial basis functions. Aerospace Science and Technology 5, 125–134.

Cebral, J.R., Löhner, R., 1997. Conservative load projection and tracking for fluid-structure problems. AIAA Journal 35, 687–692.

Edward, J.W., 1993. Computational aeroelasticity. In: Flight-Vehicles, Materials, Structures and Dynamics—Assessment and Future Directions, vol. 5. ASME, New York.

Farhat, C., Lesoinne, M., 1998. Higher-order staggered and subiteration free algorithms for coupled dynamic aeroelasticity problems. In: 36th Aerospace Sciences Meeting and Exhibit, AIAA 98-0516, Reno/NV.

Farhat, C., Degand, C., Koobus, B., Lesoinne, M., 1998. An improved method of spring analogy for dynamic unstructured fluid meshes. In: AIAA 98-2070, 39th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics and Materials Conference. Long Beach, California.

Försching, H., 1974. Grundlagen der Aeroelastik. Springer, Heidelberg.

Försching, H., 1994. New ultra high capacity aircraft (uhca)-challenges and problems from an aeroelastic point of view. Zeitschrift für Flugwissenschaften und Weltraumforschung 18, 219–231.

Harder, R.L., Desmarais, R.N., 1972. Interpolation using surface splines. Journal of Aircraft 9, 189–197.

Hönlinger, H., Manser, R., Gravelle, A., Viguier, P., 1991. Measurement of wind tunnel model deformation under airload. DGLR-Bericht 91-069, Aachen, European Forum on Aeroelasticity and Structural Dynamics.

Hounjet, M.H.L., Meijer, J.J., 1994. Evaluation of elastomechanical and aerodynamic data transfer methods for non-planar configurations in computational aeroelastic analysis. Technical Report, ICAS-Publication.

Kutler, P., 1993. Multidisciplinary computational aerosciences. In: Proceedings of the 5th International Symposium on Computational Fluid Dynamics, Sendai, pp. 109–119.

Maman, N., Farhat, C., 1995. Matching fluid and structure meshes for aeroelastic computations: a parallel approach. Computers and Structures 54, 779–785.

Narcowich, F.J., Ward, J.D., 1994. Generalized Hermite interpolation via matrix-valued conditionally positive definite functions. Mathematics of Computation 63, 661–687.

Wahba, G., 1990. Spline models for observational data. CBMS-NSF, Regional Conference Series in Applied Mathematics. SIAM, Philadelphia.

Wendland, H., 2005. Scattered data approximation. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, UK.

Wendland, H., 2007. On the stability of meshless symmetric collocation for boundary value problems. BIT Numerical Mathematics, in press, doi:10.1007/s10543-007-0121-4.

Wendland, H., Rieger, C., 2005. Approximate interpolation with applications to selecting smoothing parameters. Numerische Mathematik 101, 643–662.

Zingel, H., Jajes, U., Vogel, S., 1991. Aeroelastisches Modellprogramm I, Stationäre und instationäre Luftkräfte. Deutsche Airbus DA/BRE/91-111.