Sparse Grid Meta-Models for Model Updating

Andreas Klimke  Christopher J. Pye

Maya HTT Ltd.
4999, St. Catherine St. West, Suite 400
Montreal, Quebec, Canada, H3Z 1T3
{andreas.klimke;c.pye}@mayahtt.com

ABSTRACT

Sparse grid interpolation is a technique to optimally recover expensive, multivariate, smooth objective functions from few point-wise function evaluations. In this paper, we employ a multiple input, multiple output sparse grid interpolant to recover the eigenfrequencies and modal assurance criteria (MAC) of a structural finite element model for a given set of design parameter ranges and given reference (test) mode shapes. In order to minimize the number of model evaluations required to accurately predict the original model's behavior, we apply hierarchical, dimension-adaptive grid refinement.

Dimension-adaptive sparse grids are particularly well-suited for large-scale problems, since dimensional adaptivity provides a form of automatic effect screening (unimportant design variables are detected and the meta-model is not further refined with respect to these variables).

We proceed by minimizing a cost function involving the mode shape's MAC and the natural frequencies obtained from a modal test in order to find the optimal set of parameters to update the original analytical model.

Nomenclature

\[ [M] \]  Matrix \hspace{2cm} A(f) \quad \text{Sparse grid interpolant of } f

\( (M)_{ij} \)  Element \( i \) \( j \) of matrix \hspace{2cm} A_d(f) \quad \text{SG interpolant of dimension } d

\{v\}  Vector \hspace{2cm} N \quad \text{Number of sparse grid points}

\( (v)_i \)  Element \( i \) of vector \hspace{2cm} \|\cdot\|_\infty \quad \text{Maximum norm}

\{S\}  Set (capital letter) \hspace{2cm} [M] \quad \text{Mass matrix}

\( (S)_i \)  Element \( i \) of set \hspace{2cm} [K] \quad \text{Stiffness matrix}

#S  Number of elements of a set \( S \) \hspace{2cm} \{\phi\} \quad \text{Analysis mode shape vector}

d  Number of design variables (= dimension) \hspace{2cm} \{\psi\} \quad \text{Reference (test) mode shape vector}

m  Number of output variables \hspace{2cm} f_i \quad \text{Natural analysis frequency}

f  Objective model \hspace{2cm} g_j \quad \text{Natural reference (test) frequency}

1 INTRODUCTION

Finite element model updating is an important part of the model validation and verification process [1]. To this end, a cost function depending on design parameters is defined, measuring the difference between eigenmode analysis results (subjected to parameter variation) and experimental results (obtained via modal testing). This usually involves some combination of eigenfrequencies, mode shapes, and frequency response functions. The set of design parameter values providing the best match between analytical model and test data is then obtained by globally minimizing the cost function, where the design parameter values are often limited to predefined ranges.
Multivariate global optimization problems are difficult and time consuming to solve — it is usually an \textit{np}-hard problem for all but simple cases with special structure (such as separable problems). In the case of finite element models, extraction of modes of even relatively small models takes considerable computing time, and as the number of design parameters grows, it quickly becomes impractical trying to solve the given global optimization problem directly. Model updating tools therefore resort to model reduction, meta-modeling techniques, or attempting to only locate a (possibly suboptimal) local optimizer. This paper focuses on the meta-modeling approach.

The outline is as follows: In Section 2, we introduce sparse grid meta-modeling, highlighting the method’s key properties. Section 3 discusses the particular problem of obtaining sparse grid meta-models suitable for model updating. Section 4 shows a practical example, the updating of a bicycle frame FE model. Section 5 concludes with a summary of the approach and its limitations.

2 SPARSE GRID META-MODELS

In the following, we summarize the basic concepts of sparse grid interpolation, which is the fundamental technique underlying the approach proposed in this paper. We focus on sparse grid meta-models based on interpolation rather than regression, assuming that the objective model evaluations used to construct the meta-model will stem from finite element runs that carry a neglectable (and reproducible) numerical error [2]. To arrive at a method suitable for real-world problems, additional techniques are needed that go beyond basic sparse grid interpolation. These include dimension-adaptive grid refinement (to better address models with a large number of input parameters) [3, 4], and handling multiple model outputs efficiently.

2.1 Sparse Grid Interpolation

Sparse grids are a well-studied technique first applied in numerical quadrature to address the curse of dimensionality caused by the exponential increase in volume of the design space when increasing the number of design variables. For a recent survey on sparse grids and applications, see [5]. More specifically, the interpolation problem considered with sparse grid interpolation is an optimal recovery problem (i.e., the selection of points such that a smooth multivariate function can be approximated with a suitable interpolation formula) [6]. This is achieved by employing Smolyak’s construction, which forms the basis of all sparse grid methods: Well-known univariate interpolation formulas are extended to the multivariate case by using tensor products in a special way. As a result, one obtains a powerful interpolation method that requires significantly fewer support nodes than conventional interpolation on a full grid. For a more detailed description of the sparse grid algorithm, including grid construction, error bounds, basis functions employed, etc., please refer to [7, 8].

In the following, we denote a sparse grid meta-model of an objective function (or model) \( f \) by \( A(f) \). We denote its dimension by \( d \) (equivalent to the number of design variables) the total number of grid points by \( N \). To obtain the interpolant \( A(f) \), \( f \) must be evaluated at the \( N \) grid points.

2.2 Accuracy and Computational Cost

The achievable accuracy of sparse grid interpolation depends on the basis functions employed as well as the smoothness of an objective function \( f \). For polynomial basis functions, error bounds are derived in [6]. For \( f \in F^k_d \),

\[
F^k_d = \{ f : [-1,1]^d \to \mathbb{R} | D^\beta f \text{ continuous if } \beta_i \leq k \forall i \},
\]

with \( \beta \in \mathbb{N}_0^d \),

\[
D^\beta f = \frac{\partial^{|eta|} f}{\partial x_1^{\beta_1} \cdots x_d^{\beta_d}},
\]

the order of the interpolation error in the maximum norm is given by

\[
\| f - A_d(f) \|_\infty = \mathcal{O}(N^{-k} \cdot |\log N|^{(k+2)(d+1)+1}).
\]

Note that the dimension \( d \) occurs only in the logarithmic term, indicating the method’s excellent suitability for high-dimensional problems.

The computational cost of evaluating the sparse grid interpolant depends linearly on the number of input variables if polynomial basis functions are used, regardless of the number of input variables. Performance is even better for piecewise
linear basis functions. Construction of the interpolant is of slightly higher complexity, but can be done for tens of thousands of points in a few seconds on a standard PC [8]. Note that once a sparse grid meta-model has been constructed, global sensitivity information can be extracted efficiently as well [9].

2.3 Dimensional Adaptivity and Multiple Outputs

One can further reduce the computational effort for objective functions where not all design variables have equally significant effect on the model output. This is especially important in the case of higher-dimensional problems, where it is often observed that some design variables have either insignificant effect, or have a nearly linear contribution. Since it is usually not known a priori which design variables are important, an adaptive strategy is needed when building the meta-model to automatically detect which dimensions and variable interactions are the more or less important ones, without wasting any function evaluations, as proposed in [3, 4]. The concepts of dimensional adaptivity can be interpreted as a form of effect screening (automatically detecting main, linear, and higher-order interaction). This becomes possible due to the sparse grid’s hierarchical decomposition scheme that allows to measure the contribution of the individual terms, and direct the refinement accordingly, as illustrated with many examples in [3]. As a consequence, dimension-adaptive sparse grid algorithms can automatically detect and take advantage of separability (or partial separability) encountered in problems with additive (or nearly additive) structure.

Sparse grids are very well-suited to deal with multiple model outputs, since the regular structure allows to construct good approximations for multiple output variables at once. The cost of evaluating the objective model remains constant with increasing \( m \), while the cost of constructing the interpolant and its evaluation grows linearly with \( m \).

3 SPARSE GRID META-MODELS FOR MODEL UPDATING

3.1 Meta-Model Output Definition

Let \( \{ \psi_j \}, j = 1, \ldots, n_K \) denote the reference mode shapes (e.g., obtained by a modal test), with corresponding natural frequencies \( g_j \). Furthermore, considering the equations of motion for free vibration of an undamped MDOF system, the natural frequencies \( f_i = \sqrt{\omega_i / (2\pi)} \) and modes shapes \( \phi_i \) of the FE model are found by solving the eigenvalue problem

\[
\left( [K(x_i)] - \omega_i^2 [M(x_i)] \right) \{ \phi_{i[x],j} \} = \{ 0 \},
\]

where matrices \([K]\) and \([M]\) depend on a set of design parameters \( x_k \in [a_k, b_k], k = 1, \ldots, d \). Let \( n_A \) denote the number of modes extracted.

In order to build a meta-model, we could consider the design parameter vector as the model input, and the extracted analysis mode shapes and natural frequencies as the model outputs. However, to obtain an efficient meta-model suitable for model updating, we suggest to reduce the number of model outputs to a reasonable amount. This includes taking only a subset of natural frequencies into account, as well as computing MAC values directly as model output instead of recovering entire analysis mode shape vectors. To preselect the MAC values and natural frequencies to approximate with the meta-model, we compute the MAC matrix between the midpoint \( \{ p \} \) of the design parameter hypercube and the test mode shapes. With the elements of the MAC matrix \([\text{MAC}(\phi,\psi)]\) between the two mode shape matrices \([\phi]\) and \([\psi]\) defined as

\[
(\text{MAC}(\phi,\psi))_{ij} = \frac{| \{ \psi_j \}^T \{ \phi_i \}^2 |}{| \{ \psi_j \}^T \{ \phi_i \} \{ \phi_i \}^T \{ \phi_i \} |},
\]

we only select the elements

\[
(\text{MAC}(\phi(p),\psi))_{ij} > t,
\]

where \( t \) denotes a threshold, e.g., \( t > 0.6 \). Let us denote the set of pairs with these MAC values as

\[
\{ P \} = \{ (i, j) \mid (\text{MAC}(\phi(p),\psi))_{ij} > t \}
\]

The considered natural frequencies follow directly from this condition by picking the analysis modes belonging to these pairs, so the total number of output becomes \( m = 2\#P \).
3.2 Mode Tracking

When solving the eigenvalue problem Eq. (4), the ordering of the modes is arbitrary (most often, solvers sort the modes by increasing magnitude of the eigenvalue). Since the ordering will likely change with design parameter variation due to shifting of natural frequencies, a mode tracking mechanism is required. To this aim, we compute the MAC for the analysis modes extracted for each instance of the design parameter vector \( \{x\} \) with respect to the ones extracted the midpoint \( \{p\} \). By doing so, we can restore the original (i.e., midpoint) ordering by changing the mode indexing according to the best pairs

\[
\{S\} = \{(i, j) \mid j = 1, \ldots, n_A, i = \arg \max_{i=1,\ldots,n_A} (\text{MAC}(\phi_{(x)}, \phi_{(p)}))_{ij}\},
\]

and as a consequence, achieve consistent ordering of the model outputs (consisting of the frequencies and MAC values selected according to the pairs from Eq. (7)) – provided that the mode shapes are sufficiently distinct.

To summarize, we can express the objective model as a multiple input – multiple output function \( f : \{x\} \rightarrow \{y\} \), with the design parameters \((x)_k \in [a_k, b_k], k = 1, \ldots, d\) as the inputs, and

\[
\begin{align*}
(y)_l = \begin{cases}
\text{MAC}(\phi_{(x)}, \psi)_{ij} & (i, j) = (P)_l, 1 \leq l \leq n \frac{m}{2} \\
\frac{f_{i, (x)}}{g_j} - 1 & (i, j) = (P)_{\frac{m}{2} + l}, \frac{m}{2} < l \leq m
\end{cases}
\end{align*}
\]

as the outputs. Note that this implies that the analysis modes have been reordered according to Eq. (8).

We can now easily compute the sparse grid meta-model \( A(f) \) by applying the dimension-adaptive sparse grid algorithm as a black box to the objective function. An implementation is available as free software [10]. Note that for the dimension-adaptive algorithm to perform well for multiple output arguments, the outputs should be scaled similarly – hence the usage of frequency ratios in Eq. (9).

3.3 Target Cost Function

Since we have built the meta-model for MAC values and relative frequency error, we can directly derive a cost function from these output parameters, for instance, optimize for high MAC values and low frequency error. Denoting the output of \( A(f) \) as \( \{y^*\} \), we can use

\[
c\{x\} = \sum_{l \in 1 \ldots \frac{m}{2}} |1 - (y^*)_l| + \sum_{l \in (\frac{m}{2} + 1) \ldots m} |(y^*)_l|.
\]

Note that the sums do not necessarily run over all elements in \( \{y^*\} \). Depending on the chosen threshold \( r \), multiple possible pairs for frequency ratios and MAC values may have been approximated by the meta-model. Only the best match for each reference frequency should be included in the sum terms.

Eq. (10) is a very straightforward cost function. Of course, different weighting of targets, penalizing large design variable changes, etc. could be added easily.

4 APPLICATION: BICYCLE FRAME

In this section, we report on an application of the sparse grid meta-modeling approach developed in Section 3 to perform model updating of an FE model of a steel bicycle frame, see Fig.1. The frame, stripped of all parts, is a brazed assembly of individual tubes into steel lugs (pre-manufactured corner fittings). The frame is manufactured from Reynolds 531, the premiere bicycle steel in the 1960’s and 1970’s.

4.1 FE Model and Test Description

The FE model was built of shell, quad and triangle elements. Some tube thicknesses were unknown as the frame could not be cut up, so best guesses were used; extra thickness due to the lugs was ignored. For the initial configuration, real normal modes were extracted using SOL103 of NX NASTRAN® v5 (free-free boundary conditions). The mode shapes and natural frequencies obtained are shown in Fig.2. Solving for the first 8 modes (omitting rigid body modes) took about one minute on a standard desktop PC (the FE model has just over 100,000 DOF).

The modal test of the bike frame was conducted under simulated free-free boundary conditions. 27 test nodes were selected by eye and assigned an equivalent displacement coordinate system. FRF measurements were taken in two
directions at each test node: normal to the plane of the frame, and in plane of the frame, perpendicular to the tube. The test geometry is shown in Fig.1. Drive force locations and directions are indicated by the purple arrows. Five complex mode shapes and natural frequencies were extracted using the Advanced Modal Wizard of Test for I-DEAS® 5. To correlate them with the real normal modes from analysis, we used Niedbal's complex transformation [1].

4.2 Meta-Model Construction and Updating

For FE model updating, \( d = 11 \) design parameters were selected. The parameter range was chosen as 50% to 200% compared to the initial configuration listed in Table 1.

<table>
<thead>
<tr>
<th>Design parameter</th>
<th>Value [SI]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Young's modulus</td>
<td>206.8e+9</td>
</tr>
<tr>
<td>Top tube thickness</td>
<td>1.0e-3</td>
</tr>
<tr>
<td>Head tube thickness</td>
<td>1.0e-3</td>
</tr>
<tr>
<td>Down tube thickness</td>
<td>1.0e-3</td>
</tr>
<tr>
<td>Seat tube thickness</td>
<td>1.0e-3</td>
</tr>
<tr>
<td>Seat stays thickness</td>
<td>1.0e-3</td>
</tr>
<tr>
<td>Chain stays thickness</td>
<td>1.0e-3</td>
</tr>
<tr>
<td>Bottom bracket thickness</td>
<td>3.36e-3</td>
</tr>
<tr>
<td>Left rear dropout thickness</td>
<td>4.76e-3</td>
</tr>
<tr>
<td>Right rear dropout thickness</td>
<td>4.76e-3</td>
</tr>
<tr>
<td>Rear brake bar thickness</td>
<td>2.0e-3</td>
</tr>
</tbody>
</table>

We selected the set of modes to consider for building the meta-model, according to Eqs. (6), (7), using \( t = 0.7 \). The pairs, MAC values, and corresponding natural frequencies found are shown in Table 2. Analytical mode 2 was missed by the test.

<table>
<thead>
<tr>
<th>Test mode</th>
<th>Analysis mode</th>
<th>MAC</th>
<th>Freq. error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 83.6 Hz</td>
<td>1 – 71.8 Hz</td>
<td>0.959</td>
<td>-0.141</td>
</tr>
<tr>
<td>2 – 166.0 Hz</td>
<td>3 – 149.0 Hz</td>
<td>0.919</td>
<td>-0.102</td>
</tr>
<tr>
<td>3 – 186.1 Hz</td>
<td>4 – 179.5 Hz</td>
<td>0.961</td>
<td>-0.035</td>
</tr>
<tr>
<td>4 – 236.3 Hz</td>
<td>5 – 227.9 Hz</td>
<td>0.878</td>
<td>-0.036</td>
</tr>
<tr>
<td>5 – 265.3 Hz</td>
<td>6 – 251.9 Hz</td>
<td>0.909</td>
<td>-0.050</td>
</tr>
<tr>
<td>Abs. average</td>
<td></td>
<td>0.925</td>
<td>0.073</td>
</tr>
</tbody>
</table>

We computed sparse grid meta-models with increasing refinement, starting with \( N = 2d + 1 \) support nodes. \( N \) was increased successively by the dimension-adaptive algorithm, up to \( N = 343 \). Note that the support nodes are selected automatically by the algorithm, depending on error indicators derived from successive hierarchical refinement. A custom interface handles triggering of new model evaluations (NASTRAN SOL103 runs), importing of mode shape and natural frequency results from the generated .op2 file, and extraction of the output vector \( \{y\} \).

At each iteration, we minimized the cost function Eq. (10) to locate an optimizer \( \hat{x} \). We simply used MATLAB®’s `fminsearch` command to accomplish this task. To enforce the parameter bounds, we added a penalizing term to Eq. (10). We then evaluated the original model at \( \hat{x} \) and computed the actually achieved MAC values and frequency errors. The results are summarized in Table 3. The best set of parameters \( \hat{x} = (200.78e+9, 2.0000e-3, 2.0000e-3, 2.0000e-3, 1.8568e-3, 2.0000e-3, 2.0000e-3, 6.7200e-3, 2.3800e-3, 8.7631e-3, 2.2155e-3) \) was found at iteration 10.

To illustrate the quality of the meta-model, we computed the objective model outputs \( \{y\} \) and the meta-model outputs \( \{y^*\} \) at 100 randomly sampled points (uniformly distributed in the design parameter hypercube), and computed the mean squared error (MSE) and the maximum absolute error \( e_{est, max} \) over all outputs. The sparse grid algorithm estimates the maximum absolute error, denoted by \( e_{est, max} \), see Table 3.
Figure 1: Bike frame FE model and test geometry.

Table 3: Meta-model accuracy and optimization results.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>23</td>
<td>31</td>
<td>41</td>
<td>55</td>
<td>73</td>
<td>95</td>
<td>121</td>
<td>151</td>
<td>185</td>
<td>223</td>
<td>271</td>
<td>343</td>
</tr>
<tr>
<td>$e_{est,max}$ of $A(F)$</td>
<td>0.404</td>
<td>0.142</td>
<td>0.090</td>
<td>0.113</td>
<td>0.231</td>
<td>0.068</td>
<td>0.074</td>
<td>0.074</td>
<td>0.059</td>
<td>0.057</td>
<td>0.037</td>
<td></td>
</tr>
<tr>
<td>$e_{act,max}$ of $A(F)$</td>
<td>0.171</td>
<td>0.180</td>
<td>0.164</td>
<td>0.175</td>
<td>0.122</td>
<td>0.106</td>
<td>0.103</td>
<td>0.108</td>
<td>0.117</td>
<td>0.119</td>
<td>0.096</td>
<td></td>
</tr>
<tr>
<td>MSE of $A(F)$</td>
<td>5.4e-4</td>
<td>4.9e-4</td>
<td>4.4e-4</td>
<td>3.3e-4</td>
<td>2.7e-4</td>
<td>1.8e-4</td>
<td>1.4e-4</td>
<td>1.2e-4</td>
<td>8.5e-5</td>
<td>7.9e-5</td>
<td>5.4e-5</td>
<td></td>
</tr>
<tr>
<td>$(MAC(\phi_{13}, \psi))_{11}$</td>
<td>0.959</td>
<td>0.951</td>
<td>0.941</td>
<td>0.948</td>
<td>0.955</td>
<td>0.953</td>
<td>0.944</td>
<td>0.961</td>
<td>0.951</td>
<td>0.971</td>
<td>0.968</td>
<td>0.958</td>
</tr>
<tr>
<td>$(MAC(\phi_{13}, \psi))_{23}$</td>
<td>0.933</td>
<td>0.936</td>
<td>0.940</td>
<td>0.936</td>
<td>0.928</td>
<td>0.931</td>
<td>0.944</td>
<td>0.943</td>
<td>0.935</td>
<td>0.943</td>
<td>0.940</td>
<td>0.943</td>
</tr>
<tr>
<td>$(MAC(\phi_{13}, \psi))_{34}$</td>
<td>0.962</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.964</td>
<td>0.966</td>
<td>0.971</td>
<td>0.961</td>
<td>0.968</td>
<td>0.956</td>
<td>0.957</td>
<td>0.962</td>
</tr>
<tr>
<td>$(MAC(\phi_{13}, \psi))_{45}$</td>
<td>0.895</td>
<td>0.892</td>
<td>0.884</td>
<td>0.888</td>
<td>0.893</td>
<td>0.890</td>
<td>0.888</td>
<td>0.927</td>
<td>0.900</td>
<td>0.933</td>
<td>0.939</td>
<td>0.924</td>
</tr>
<tr>
<td>$(MAC(\phi_{13}, \psi))_{56}$</td>
<td>0.925</td>
<td>0.934</td>
<td>0.933</td>
<td>0.936</td>
<td>0.931</td>
<td>0.932</td>
<td>0.933</td>
<td>0.923</td>
<td>0.941</td>
<td>0.910</td>
<td>0.908</td>
<td>0.937</td>
</tr>
<tr>
<td>Average MAC</td>
<td>0.935</td>
<td>0.936</td>
<td>0.935</td>
<td>0.934</td>
<td>0.934</td>
<td>0.936</td>
<td>0.943</td>
<td>0.939</td>
<td>0.943</td>
<td>0.942</td>
<td>0.945</td>
<td></td>
</tr>
<tr>
<td>$f_k/g_l - 1$</td>
<td>-0.011</td>
<td>-0.014</td>
<td>-0.015</td>
<td>-0.011</td>
<td>-0.008</td>
<td>-0.010</td>
<td>-0.006</td>
<td>-0.010</td>
<td>0.001</td>
<td>-0.016</td>
<td>-0.020</td>
<td>-0.044</td>
</tr>
<tr>
<td>$f_3/g_2 - 1$</td>
<td>-0.057</td>
<td>-0.038</td>
<td>-0.026</td>
<td>-0.024</td>
<td>-0.026</td>
<td>-0.029</td>
<td>-0.019</td>
<td>-0.012</td>
<td>-0.022</td>
<td>-0.028</td>
<td>-0.029</td>
<td>-0.036</td>
</tr>
<tr>
<td>$f_4/g_3 - 1$</td>
<td>-0.002</td>
<td>-0.002</td>
<td>-0.000</td>
<td>0.005</td>
<td>0.003</td>
<td>0.001</td>
<td>0.001</td>
<td>0.023</td>
<td>0.005</td>
<td>0.004</td>
<td>0.015</td>
<td>0.000</td>
</tr>
<tr>
<td>$f_5/g_4 - 1$</td>
<td>0.007</td>
<td>0.025</td>
<td>0.031</td>
<td>0.043</td>
<td>0.039</td>
<td>0.034</td>
<td>0.045</td>
<td>0.012</td>
<td>0.049</td>
<td>-0.002</td>
<td>-0.002</td>
<td>0.003</td>
</tr>
<tr>
<td>$f_6/g_5 - 1$</td>
<td>-0.002</td>
<td>0.001</td>
<td>0.004</td>
<td>0.003</td>
<td>0.002</td>
<td>0.006</td>
<td>0.001</td>
<td>-0.001</td>
<td>0.001</td>
<td>-0.001</td>
<td>-0.003</td>
<td></td>
</tr>
<tr>
<td>Average freq. error</td>
<td>0.016</td>
<td>0.016</td>
<td>0.014</td>
<td>0.017</td>
<td>0.016</td>
<td>0.015</td>
<td>0.016</td>
<td>0.012</td>
<td>0.016</td>
<td>0.010</td>
<td>0.013</td>
<td>0.017</td>
</tr>
</tbody>
</table>
Mode 1: 71.8 Hz
Mode 2: 95.4 Hz
Mode 3: 149.0 Hz
Mode 4: 179.5 Hz
Mode 5: 227.9 Hz
Mode 6: 251.9 Hz

Figure 2: FE model, modes 1–6.
In this paper, a meta-modeling technique for model updating based on dimension-adaptive sparse grids was proposed. We showed how suitable target outputs can be selected based on the MAC matrix. To be able to construct the meta-model, it is important that mode ordering is consistent. To this end, we proposed to use the mode order for the midpoint of the design parameter space as a reference. We were able to demonstrate the efficiency of the approach in terms of low number of required model evaluations for high meta-model accuracy.

Despite the encouraging results reported in Section 4, we found that the approach has some limitations:

- Mode tracking by MAC using the midpoint modes as references can fail if the design parameter ranges get too large. This problem increases with dimension since the corner points of the design parameter hypercube grow more distant with respect to the midpoint. Failure to always correctly order the modes can cause wrong function values to enter the adaptive meta-model construction process, resulting in a much reduced meta-model accuracy.

- Strong local features (peaks and valleys with steep slope) are hard to resolve with dimension-adaptive refinement. While we found this unlikely to occur for the relatively small permissible parameter variability used for model updating, we did encounter this problem for a MAC output parameter of another FE model we studied, where the MAC value would suddenly drop sharply. This resulted in a significantly increased number of required iterations until sufficient meta-model accuracy was reached, or even failure of mode tracking, as described above.

References


