Compression of Higher Derivative Tensors in Stochastic Significance Analysis

Jens Deussen and Uwe Naumann
LuFG Informatik 12, RWTH Aachen University
Software and Tools for Computational Engineering
52074 Aachen, Germany
Email: [deussen,naumann]@stce.rwth-aachen.de

***Abstract*—In the SCoRPiO project significance based approximate computing is used to reduce the energy consumption of a program execution by tolerating less accurate results. Part of the project is to define significance as an algorithmic property to quantify the impact of a computation to the output. In the last years we presented the interval-adjoint significance analysis which combines interval information with derivative information. Thus, the analysis can identify computations that can be evaluated less accurate, e.g. on low power but less reliable hardware. Unfortunately, by using interval arithmetic the analysis sometimes results in a large overestimation of intervals and hence large significance values.**

For that reason, we propose an alternative approach to obtain significance information by propagation of moment approximations of probability distributions. This method uses Taylor series which require higher derivatives to approximate statistical moments, e.g. the expectation and the variance. The computation of these derivatives is expensive even for adjoint algorithmic differentiation methods. Therefore, we exploit the symmetry and the sparsity structure of the higher derivatives to considerably improve the efficiency of the computation.

I. Introduction

In approximate computing computer programs are executed at lower cost (e.g. energy) by relaxing reliability constraints of the computation. A lot of applications accept approximate results, e.g. in the field of multimedia, audio, image and video processing respectively, or in data mining. The approach of significance-driven approximate computing exploit the fact that the computations of an applications have different importance for the quality of the result. Significant computations are more important to the quality of the output while insignificant computations just have a minor impact. In [1] statistical analysis is used to classify computations as significant or non-significant. Another approach was introduced in [2] in which significance information is obtained by interval arithmetic and error propagation based on local partial derivatives.

Significance-driven approximate computing is also researched in the FET-open project SCoRPiO[1]. Automatic code characterization techniques which use compile- and run-time analyses are developed to identify the significant and insignificant computations. By tolerating a controlled degree of imprecision, insignificant computations can be steered to low power yet less reliable hardware. Another option to exploit significance information is to save computing time by replacing insignificant computations by representative expressions.

As a part of SCoRPiO we developed the deterministic interval-adjoint significance analysis (IASA) to obtain significance information of each computation of a computer program for a user specified input domain automatically. IASA combines the forward propagation of interval values and the backward propagation of adjoints to define significance (see e.g. [3]). Therefore, interval arithmetic (IA) [4] and algorithmic differentiation (AD) [5], [6], [7] are applied. Binary interval splitting [8], [9] is defined to address difficulties introduced by the naive usage of interval arithmetic, e.g. unfeasible relational operators or the wrapping effect [10].

IASA is implemented in `dco/scorpio`[2]. The software transforms a given `C` or `C++` source code to a directed acyclic graph representing single assignment code. The user has to provide input intervals that are propagated through the graph, such that value intervals for each intermediate variable are computed. After that, adjoint intervals are propagated backwards to obtain derivatives of the output with respect to all input and intermediate variables. By combining the forward and the backward information significance values for all variables can be computed.

These significance values can be used in the SCoRPiO source-to-source compiler [11], [12] to set the significance of tasks. The software is extended by a significance exploration tool set [9] to support the analysis. This tool set enables the automatic visualization of the computational graph annotated with significance information to give the user a better understanding of the program. Furthermore, it is possible to generate optimized source code based on the computed significance values and apply graph algorithms to detect further insignificant computations.

In this paper we introduce an alternative approach for the computation of significance information. Instead of propagating intervals we propose to propagate probability density functions (PDFs). Therefore, we apply the moments method (see e.g. [13], [14], [15], [16], [17]) that uses Taylor series expansion to approximate the moments of the outputs, mean and variance in particular. Thus, for given input PDFs and derivative information the moments method computes

[1] http://www.scorpio-project.eu/outline/

[2] https://gitlab.stce.rwth-aachen.de/SCORPIO/dco_scorpio/

approximations of the moments of the output PDFs. Among others, this method is used for robust optimization in [13] and [17].

To improve the accuracy of the approximations of the moments higher-order terms in the Taylor series need to be involved that require higher derivatives. Again, AD can be used to compute these higher derivatives (see e.g. [18]). Due to the expenses that come along with these computations it is indispensable to exploit the sparsity structure and symmetry of the higher-order tensors. While [19] describes the sparsity of the a third derivative tensor, in [20] graph coloring algorithms are introduced that enhance the computation of first and second derivatives, Jacobian and Hessian matrices respectively. To our knowledge there is no publication considering coloring techniques for third and higher derivatives.

The document is organized as follows: Section II gives a new definition of significance. Furthermore, the moments method and algorithmic differentiation are outlined. After that, we focus on the computation of higher derivatives by exploiting symmetry and sparsity. Subsequently, in section III we illustrate the methods for the exploitation of symmetry and sparsity for third order derivatives by applying them to a minimal example.

II. Methodology

In this section we recall the basics of the significance analysis from [9] and introduce a new analysis based on the moments method and algorithmic differentiation. Moreover, we give a new definition of significance based on uncertainty information obtained by the moments method.

As in [9] we assume a computer program P implementing a multivariate scalar function $f : D \to I$ with domain $D \subseteq \mathbb{R}^n$, image $I \subseteq \mathbb{R}$ and $y = f(\mathbf{x})$ in which $\mathbf{x} = (x_1, \ldots, x_n)^T$. The implementation of function f can be decomposed into a sequence of p elemental functions (binary operations and intrinsic functions) by the three-part evaluation procedure from [6]. This transformation yields single assignment code in which each intermediate variable v_i stores the result of an elemental function.

A significance analysis should assign significance values $S_y(v_i)$ to all intermediate variables v_i for a given input domain D. In [9] we assumed that the input domain is given by intervals $D = [\mathbf{x}] = [\underline{\mathbf{x}}, \overline{\mathbf{x}}] = \{\mathbf{x} \in \mathbb{R}^n | \underline{\mathbf{x}} \leq \mathbf{x} \leq \overline{\mathbf{x}}\}$ with lower bound $\underline{\mathbf{x}} \in \mathbb{R}^n$ and upper bound $\overline{\mathbf{x}} \in \mathbb{R}^n$. From now on we will assume that the input domain is composed of random variables from PDFs that are described by their first four moments, the mean $\boldsymbol{\mu}_\mathbf{x}$, the variance $\boldsymbol{\sigma}^2_\mathbf{x}$, the skewness $\boldsymbol{\gamma}_\mathbf{x}$ and the kurtosis $\boldsymbol{\kappa}_\mathbf{x}$ in particular. Therefore, the analysis needs to quantify the influence of the input domain D on each intermediate variable v_i and the influence of each intermediate variable on the output y. The significance value should be a combination of both information.

A. Definition of Significance

In [9] we proposed the significance definition of an intermediate variable v_i that combines forward information of IA and backward information of AD for a given input range $[\mathbf{x}]$

$$S_y(v_i) = w[v_i] \cdot \max |\nabla_{[v_i]}[y]| \ . \tag{1}$$

In (1) $w([v_i]) = \overline{v_i} - \underline{v_i}$ denotes the width of the interval value $[v_i]$ that measures the impact of the input $\mathbf{x}$ on an intermediates variable v_i. A large width $w([v_i])$ indicates that the intermediate v_i is highly sensitive to the variation of all inputs $\mathbf{x}$ in the range $[\mathbf{x}]$. Furthermore, the absolute maximum of the first order derivatives $|\nabla_{[v_i]}[y]|$ of output interval $[y]$ with respect to intermediate v_i is a measurement for the individual influence of intermediate variables to the output. If the absolute value of this derivative $|\nabla_{[v_i]}[y]|$ is small, a change in the value of v_i has just a small impact on y.

We now introduce a new definition that uses the first two moments to quantify the significance of an intermediate variable v_i:

$$S_y(v_i) = \sigma_{v_i} \cdot \left| \frac{\partial \mu_y}{\partial \mu_{v_i}} \right| \ . \tag{2}$$

While the mean measures the central location of a PDF the variance measure its variability or width (see [21]). The square root of the variance is known as the standard deviation of the PDF. The standard deviation σ_{v_i} can be used as an interval estimator of the mean which is proportional to its confidence interval. We use σ_{v_i} analog to the width of the value interval in (1). Thus, a large value of σ_{v_i} means that the confidence interval of v_i is wide for the given input PDFs. Again, as a second factor we use a first derivative of the output y with respect to the intermediate variable v_i. Instead of using interval derivatives, we are now interested in the change of the mean of the output PDF by slightly changing the mean of the intermediate variable. This second factor describes the individual influence of μ_{v_i} on μ_y.

In the next subsection we outline how to obtain moments of the PDF of intermediate variables by using the moments method.

B. Moments Method

The moments method can be used to estimates the PDFs of all intermediate and output variables for given input PDFs. In [14] the moments method is derived with no assumption on the input PDF by an multivariate Taylor series expansion of $f(\mathbf{x})$ about the mean $\boldsymbol{\mu}_\mathbf{x}$.

The first two moments are defined as

$$\mu_y = E\left[f(\mathbf{x})\right] \tag{3}$$

$$\sigma_y^2 = E\left[\left(f(\mathbf{x}) - E\left[f(\mathbf{x})\right]\right)^2\right] \ . \tag{4}$$

By neglecting higher terms in the Taylor series of $f(\mathbf{x})$ the moments of the output in (3) and (4) can be approximated by

$$\mu_y \approx y^{\{0\}} + \sum_i \left(y^{\{ii\}} + \gamma_i y^{\{iii\}}\right) + \mathcal{O}\left(\sigma_\mathbf{x}^4\right), \tag{5}$$

and

$$\sigma_y^2 \approx \sum_i \Bigg(\left(y^{\{i\}}\right)^2 + 2\gamma_i y^{\{i\}} y^{\{ii\}} + (\kappa_i - 1)\left(y^{\{ii\}}\right)^2 + 2\kappa_i y^{\{i\}} y^{\{iii\}} + \sum_{i \neq j} \left(2\left(y^{\{ij\}}\right)^2 + 6 y^{\{i\}} y^{\{ijj\}}\right)\Bigg) + \mathcal{O}\left(\sigma_{\mathbf{x}}^4\right) \ , \tag{6}$$

in which $y^{\{0\}} = f(\boldsymbol{\mu}_{\mathbf{x}})$ and the Taylor coefficients $y^{\{i\}}$, $y^{\{ij\}}$ and $y^{\{ijk\}}$ are defined as

$$\begin{aligned} y^{\{i\}} &= \frac{\partial f}{\partial x_i}\sigma_{x_i} \ , \\ y^{\{ij\}} &= \frac{1}{2!}\frac{\partial^2 f}{\partial x_i \partial x_j}\sigma_{x_i}\sigma_{x_j} \ , \\ y^{\{ijk\}} &= \frac{1}{3!}\frac{\partial^3 f}{\partial x_i \partial x_j \partial x_k}\sigma_{x_i}\sigma_{x_j}\sigma_{x_k} \ . \end{aligned}$$

For a more detailed derivation see [14] or [16].

In (5) and (6) it can be seen that a fourth-order approximation already requires derivatives up to third order. Increasing the accuracy of the approximation can be done considering higher-order terms but their computation require even higher derivatives. In the following subsection, we discuss how to efficiently compute higher derivatives by AD.

C. Algorithmic Differentiation

The first derivative of a multivariate scalar function $f : \mathbb{R}^n \to \mathbb{R}$ is called gradient and looks like

$$\nabla f = \left(\frac{\partial y}{\partial x_i}\right)_{i=0,\ldots,n-1} \in \mathbb{R}^n \ .$$

The second derivative of f with respect to $\mathbf{x}$ is called Hessian and has the form of a matrix

$$H = \nabla^2 f = \left(\frac{\partial^2 y}{\partial x_i \partial x_j}\right)_{i,j=0,\ldots,n-1} \in \mathbb{R}^{n\times n} \ ,$$

and the third derivative is a 3-tensor

$$\mathcal{T} = \nabla^3 f = \left(\frac{\partial^3 y}{\partial x_i \partial x_j \partial x_k}\right)_{i,j,k=0,\ldots,n-1} \in \mathbb{R}^{n\times n\times n} \ .$$

If the function f is local differentiable AD can compute additionally to the function value its derivatives by using the chain rule. There are two basic modes: the tangent (also: *forward*) and the adjoint (also: *reverse*) mode. In the tangent mode the partial derivatives of elemental functions are accumulated from the inputs to the outputs. In contrast, the partial derivatives are accumulated the other way around, from the output to the inputs in the adjoint mode. In the following we will use the notation of [7] where tangents are denoted with superscripts and adjoints with subscripts.

An evaluation of the tangent model can be interpreted as a product of the first derivative with a user-defined tangent $\mathbf{x}^{(1)}$. By setting the tangent equals to the standard basis of $\mathbb{R}^n$ (also called seeding) the tangent model yields one element of the gradient. The value of this element can be retrieved (also called harvested) from $y^{(1)}$.

$$y^{(1)} = f^{(1)}(\mathbf{x}, \mathbf{x}^{(1)}) = \left\langle \nabla f(\mathbf{x}), \mathbf{x}^{(1)} \right\rangle = \nabla f(\mathbf{x}) \cdot \mathbf{x}^{(1)} \tag{7}$$

Thus, to obtain the whole gradient of f the tangent model has to be evaluated n times. On the other side, the adjoint model can be interpreted as a product of the transposed of the first derivative with a user-defined adjoint $y_{(1)}$.

$$\mathbf{x}_{(1)} = f_{(1)}(\mathbf{x}, y_{(1)}) = \left\langle y_{(1)}, \nabla f(\mathbf{x}) \right\rangle = \nabla f(\mathbf{x})^\top \cdot y_{(1)} \tag{8}$$

By seeding the standard basis of $\mathbb{R}$ (note that there is only one element), the adjoint model results in the whole gradient, such that a single evaluation of the adjoint model is sufficient.

D. Computation of Higher Derivatives

To obtain of the Hessian we can compute the first derivative of the first derivative. Due to the fact that we have two basic modes there are four combinations to compute the Hessian. The tangent-over-tangent model result from the application of the tangent mode to the first-order tangent model in (7).

$$\begin{aligned} y^{(1,2)} &= f^{(1,2)}(\mathbf{x}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}) \\ &= \left\langle \nabla^2 f(\mathbf{x}), \mathbf{x}^{(1)}, \mathbf{x}^{(2)} \right\rangle \\ &= \mathbf{x}^{(1)\top} \cdot \nabla^2 f(\mathbf{x}) \cdot \mathbf{x}^{(2)} \end{aligned}$$

By seeding the standard basis of $\mathbb{R}^n$ for the tangents $x^{(1)}$ and $x^{(2)}$ of the tangent-over-tangent model the Hessian can be computed with n^2 evaluations of $f^{(1,2)}$. The other three models have the same computational complexity because they apply the adjoint mode at least once. To obtain the whole Hessian the second-order adjoint model need to be evaluated n times. In this paper we will only consider the tangent-over-adjoint model where the tangent mode is applied to (8).

$$\begin{aligned} \mathbf{x}_{(1)}^{(2)} &= f_{(1)}^{(2)}(\mathbf{x}, \mathbf{x}^{(2)}, y_{(1)}) \\ &= \left\langle y_{(1)}, \nabla^2 f(\mathbf{x}), \mathbf{x}^{(2)} \right\rangle \\ &= y_{(1)}^\top \cdot \nabla^2 f(\mathbf{x}) \cdot \mathbf{x}^{(2)} \end{aligned} \tag{9}$$

Note that these models are simplified versions of the complete second-order models by setting mixed terms to zero. A detailed derivation can be found in [6] and [7].

To illustrate the procedure we suppose the matrix in (10) as an example Hessian.

$$\begin{array}{c} \begin{matrix} h_0 & h_1 & h_2 \end{matrix} \\ \begin{pmatrix} H_{00} & H_{01} & \\ H_{10} & H_{11} & \\ & & H_{22} \end{pmatrix} \end{array} \tag{10}$$

If we use (9) and seed $y_{(1)} = 1$ and $\mathbf{x}^{(2)} = (1,0,0)^\top$ we obtain $\mathbf{x}_{(1)}^{(2)} = h_0 = (H_{00}, H_{10}, 0)^\top$. To get the other two columns of the Hessian, we additionally need to seed $\mathbf{x}^{(2)} = (0,1,0)^\top$ and $\mathbf{x}^{(2)} = (0,0,1)^\top$.

To receive even higher derivative models the two basic modes can be applied recursively to f. As an example

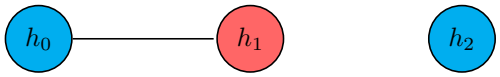

Fig. 1: Column intersection graph of the symmetric matrix from (10) and possible distance-1 coloring

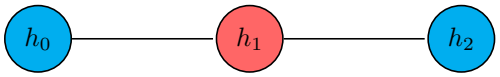

Fig. 2: Adjacency graph of the symmetric matrix from (12) and possible star coloring

the third-order adjoint model via tangent-over-tangent-over-adjoint can be written as

$$\begin{aligned}\mathbf{x}_{(1)}^{(2,3)} &= f_{(1)}^{(2,3)}(\mathbf{x}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, y_{(1)}) \\ &= \left\langle y_{(1)}, \nabla^3 f(\mathbf{x}), \mathbf{x}^{(2)}, \mathbf{x}^{(3)} \right\rangle \ .\end{aligned} \tag{11}$$

To derive the 3-tensor of the third derivative n^2 evaluations of $f_{(1)}^{(2,3)}$ are required. For large n this computation becomes very expensive in terms of runtime. These derivatives are symmetric and they are often sparse which means that they contain a lot of zero elements. In the next two subsections we give an overview how to exploit these facts.

E. Exploitation of Symmetry and Sparsity of Hessians

In [20] a comprehensive summary of coloring techniques is given that can be used for the exploitation of sparsity of Jacobian and Hessian matrices. The general idea is to identify columns or rows of the corresponding matrix that can be computed at the same time. The sparsity patterns of the matrices are required for these algorithms. One possibility to obtain these patterns is to propagate sets of indices. Nevertheless, the detection of sparsity patterns is out of the scope of this paper. Thus, we consider the sparsity pattern $\mathcal{P}$ of the higher derivatives to be known.

A first approach is to find orthogonal columns in the matrix. This problem can also be described as a graph coloring problem. The sparsity structure of the matrix induces a column intersection graph, where each column is represented as a node. Two nodes are connected by an edge if they have an element in the same row. After that, an distance-1 coloring is applied to the graph in that the same color cannot be assigned to adjacent nodes. The nodes and thus the columns of the Hessian which have the same color can be computed simultaneous.

As an example we can assume the sparsity structure from (10) again. The corresponding column intersection graph for the Hessian is given in Fig. 1. Note that the two adjacent nodes have different colors due to the distance-1 coloring. It would also be possible that h_1 and h_2 get the same color. By seeding $y_{(1)} = 1$ and $\mathbf{x}^{(2)} = (1, 0, 1)^\top$ the tangent-over-adjoint model from (9) results in $\mathbf{x}_{(1)}^{(2)} = h_0 + h_2 = (H_{00}, H_{10}, H_{22})^\top$. Due to the fact, that we know the sparsity structure of the Hessian we can directly recover the corresponding columns of the matrix.

By considering the symmetry of the Hessian ($H_{ij} = H_{ji}\ \forall i, j$) another coloring algorithm can be applied. Instead of computing each element of the Hessian we compute the symmetric elements at least once. The so called star coloring uses an adjacency graph where each node belongs to a column (or row). Each element H_{ij} in the matrix is an edge connecting row i and j in the graph. Then, a distance-1 coloring with the additional condition that every path of length four uses at least three colors is applied.

Assume that we have the following Hessian structure:

$$\begin{pmatrix} H_{00} & H_{01} & \\ H_{10} & H_{11} & H_{12} \\ & H_{21} & H_{22} \end{pmatrix}. \tag{12}$$

The resulting adjacency graph is visualized in Fig. 2. The additional coloring condition that every path of length four needs at least three colors is for this basic example trivial. Seeding $y_{(1)} = 1$ and $\mathbf{x}^{(2)} = (1, 0, 1)^\top$ the tangent-over-adjoint model from (9) results in $\mathbf{x}_{(1)}^{(2)} = h_0 + h_2 = (H_{00}, H_{10} + H_{12}, H_{22})^\top$. Furthermore, a seed of $y_{(1)} = 1$ and $\mathbf{x}^{(2)} = (0, 1, 0)^\top$ yields $\mathbf{x}_{(1)}^{(2)} = h_1 = (H_{01}, H_{11}, H_{12})^\top$. Due to the symmetry and the sparsity structure of the Hessian we can directly recover the corresponding columns of the matrix.

Another technique described in [20] is the acyclic coloring that also applies a distance-1 coloring to the adjacency graph. This coloring technique has the additional condition that each cycle in the graph needs at least three distinct colors. It enables the recovery of elements of the Hessian via substitution. In this method elements can be computed by solving a system of linear equations. This means that the value of H_{10} can be retrieved from $\mathbf{x}_{(1)}^{(2)} = (H_{00}, H_{10} + H_{12}, H_{22})^\top$ by subtracting the value of H_{12} from the second element.

With all these methods the complexity of the AD part is of order $n_{\mathrm{c},H}$ which is the number of assigned colors.

F. Exploitation of Symmetry and Sparsity of Third Derivatives

As already proposed in [19] the 3-tensor of the third derivative $\nabla^3 f$ is super-symmetric, which means that

$$\mathcal{T}_{ijk} = \mathcal{T}_{ikj} = \mathcal{T}_{jik} = \mathcal{T}_{jki} = \mathcal{T}_{kij} = \mathcal{T}_{kji} \ .$$

Those elements of $\mathcal{T}$ that have distinct indices for i, j and k appear six times. Considering this symmetry we only need to compute those elements where $i \geq j \geq k$. By evaluating the third-order adjoint model from (11) we get a column or a row vector of the 3-tensor. Thus, seeding e_i (a vector from the standard basis with an one in element i) for the first tangent and e_j for the second tangent, we only need to consider those tangents with $i \geq j$. This yields a computational complexity of $\frac{n \cdot (n+1)}{2}$ evaluations to obtain $\mathcal{T}$.

A second approach to exploit this symmetry is to compute the lower triangle of the for the first half of the tensor slices ($i \geq j$ for $i < \frac{n}{2}$) and the upper triangle for the second half ($i \leq j$ for $i \geq \frac{n}{2}$). We define a tensor slice $\mathcal{T}_i$ as the subdomain of the full 3-tensor for a fixed i. This approach

covers the required elements with $\lceil \frac{n}{2} \rceil \cdot \left(\lfloor \frac{n}{2} \rfloor + 1 \right)$ evaluations. Both, the first and the second approach are also applicable for dense higher derivatives due to the fact that further symmetric dimensions are added.

In [19] a so called *induced* sparsity $\tilde{\mathcal{P}}_{\mathcal{T}}$ of the third derivative was introduced where

$$\mathcal{T}_{ijk} = 0, \text{ if } H_{ij} = 0 \ \vee \ H_{ik} = 0 \ \vee \ H_{jk} = 0 \ .$$

Thus, the *induced* sparsity pattern of each slice of the third derivative 3-tensor $\tilde{\mathcal{P}}_{\mathcal{T}_i}$ is a subset of the sparsity pattern of the Hessian $\mathcal{P}_H$. Furthermore, the resulting pattern is an overestimation of the actual sparsity pattern of the third derivative $\mathcal{P}_{\mathcal{T}_i}$ which yields

$$\mathcal{P}_{\mathcal{T}_i} \subseteq \tilde{\mathcal{P}}_{\mathcal{T}_i} \subseteq \mathcal{P}_H \ .$$

A native approach to exploit this sparsity is to consider each tensor slice $\mathcal{T}_i$ for its own and apply the colors that are obtained by the Hessian coloring because $\mathcal{P}_{\mathcal{T}_i} \subseteq \mathcal{P}_H$. For the seeding this means that the first tangent is set equals to the standard basis vectors while the second tangent is determined through the colors. This approach results in $n \cdot n_{\mathrm{c},H}$ evaluations of the third-order adjoint model for the computation of $\mathcal{T}$. Due to the super-symmetry of $\mathcal{T}$ the Hessian colors can be applied for both tangents.

The first tangent will compress or merge the tensor slices that belong to one of the colors. Because $\mathcal{P}_{\mathcal{T}_i} \subseteq \mathcal{P}_H$ is valid for all i, the union of the corresponding sparsity patterns is still a subset of the Hessian pattern

$$\bigcup_{i \in c} \mathcal{P}_{\mathcal{T}_i} \subseteq \mathcal{P}_H \ , \tag{13}$$

where c denotes a set of indices that belong to the same color. Therefore, the colors of the Hessian coloring can be applied for the second tangent to compress the slices again. This approach yields a computational complexity of $n_{\mathrm{c},H}^2$ to obtain the whole tensor.

Since we assume to know the actual sparsity pattern of the third derivative we can try to improve the algorithms by considering this information. Thus, a last approach is to apply the colors of the Hessian for the first tangent to compress the tensor as explained in the last approach. Then, each slice of the compressed tensor $\bigcup_{i \in c} \mathcal{T}_i$ can be colored again by using the known compressed sparsity patterns. The coloring for the second tangent cannot be worse than the coloring of the Hessian computation due to (13). Consequently, using the coloring of the slices of the compressed tensor results in $n_{\mathrm{c},H} \cdot n_{\mathrm{c},\mathcal{T}}$ evaluations, where $n_{\mathrm{c},\mathcal{T}} \leq n_{\mathrm{c},H}$.

All the invented methods can be applied recursively to exploit the sparsity of even higher derivatives. In the next section we will apply these approaches to a simple example.

III. Example of Third Derivative Computation

To illustrate the proposed methods for the exploitation of symmetry and sparsity we propose the following example function.

$$f(\mathbf{x}) = \sum_{i=0}^{n-1} x_i^3 + \sum_{i=1}^{n-2} x_{i-1} \cdot x_i \cdot x_{i+1} \ .$$

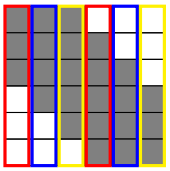

Fig. 3: Sparsity pattern of the Hessian with three colors that are obtained by acyclic coloring

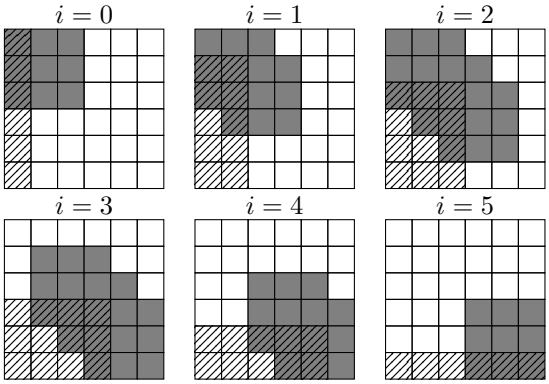

Fig. 4: Induced sparsity pattern of the third derivative (gray squares) and representation of those elements that need to be computed in the symmetric dense case (striped squares)

The corresponding sparsity pattern of the Hessian is visualized for $n = 6$ in Fig. 3. In addition to the nonzero elements that are represented by gray squares a color is assigned to each column by using the acyclic coloring from [20]. The coloring algorithm assigns three different colors with $c_0 = \{0, 3\}$, $c_1 = \{1, 4\}$ and $c_2 = \{2, 5\}$, such that $n_{c,H} = 3$.

In Fig. 4 the induced sparsity pattern for the third derivative is given. Furthermore, the striped squares indicate those elements that need to be computed in the dense case. The third-order adjoint model from (11) needs to be computed for each column that has at least one striped element that are 21 evaluations. Due to symmetry it is also possible to compute rows of the model. Using the computation of columns for the first half and the computation of rows for the second half results in 12 evaluations in the dense case.

In Fig. 5 the sparsity pattern of the third derivative of the example function is displayed. It is obvious that the induced sparsity pattern yields a large overestimation in this example. Furthermore, it can be seen that the colors of the Hessian can be applied for each slice of the tensor $\mathcal{T}_i$. For $n_{c,H} = 3$ and $n = 6$ this would result in 18 evaluations of (11).

The second approach for sparse third derivatives was to apply the Hessian colors twice. Using these colors for the first tangent will compress the tensor as visualized in Fig. 6. After that, each slice of the compressed tensor is compressed again by using these colors. This approach requires 9 evaluations of the adjoint model to obtain the third derivative of the example function.

Fig. 6 also belongs to the third approach where the tensor

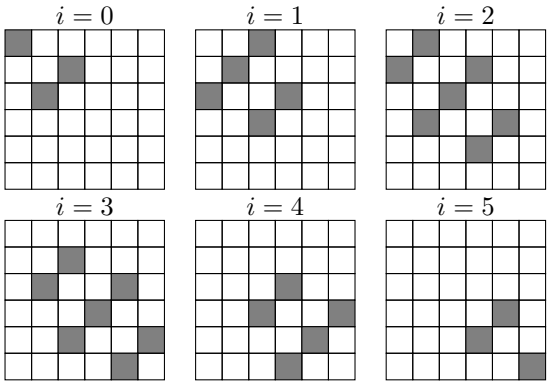

Fig. 5: Actual sparsity pattern of the third derivative

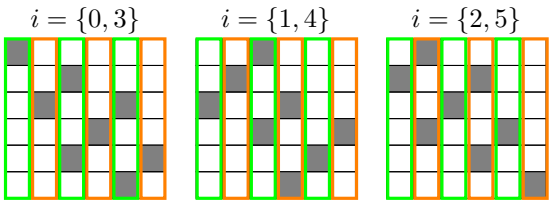

Fig. 6: Sparsity pattern of the compressed third derivative with two colors that are obtained by acyclic coloring for each subdomain

is compressed first by using the Hessian colors and then the resulting slices are colored again by using the exact sparsity pattern. The colors that are obtained by an acyclic coloring are also visualized. The number of colors for each slice is less than the number of Hessian colors, $n_{c,\mathcal{T}} = 2$. Thus, this approach would result in 6 evaluations of (11).

By increasing n in this example the number of colors that are assigned by the acyclic coloring stays constant such that the computation is possible even for large n.

IV. Summary and Outlook

In this paper we introduced our new approach for the computation of significance information that can be used for significance-driven approximate computing. The approach is based on the moments method to propagate uncertainties by using Taylor expansions. It is indispensable to use higher derivatives to improve the accuracy of the method. Due to the fact that the computation of higher derivatives can be very expensive we applied coloring techniques that are developed for the sparse Hessian computation. Unfortunately, there are no coloring techniques for third or higher derivatives. Thus, we introduced various methods to use the information of the Hessian coloring for higher derivatives. All of these methods can be used for arbitrary order of derivatives by applying them recursively. In an example we showed that the computational complexity of the third derivative computation can be significantly reduced by using the proposed methods.

This work can be extended by developing coloring algorithms that directly uses the sparsity patterns of the higher derivatives. Furthermore, the proposed approach to obtain significance information needs to be implemented and compared to the interval-adjoint significance analysis.

References

[1] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator," in *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*. ACM, 2009, pp. 195–200.

[2] S. Misailovic, M. Carbin, S. Achour, Z. Qi, and M. C. Rinard, "Chisel: reliability-and accuracy-aware optimization of approximate computational kernels," in *Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications*. ACM, 2014, pp. 309–328.

[3] J. Riehme and U. Naumann, "Significance analysis for numerical models," *WAPCO*, 2015. [Online]. Available: http://wapco.e-ce.uth.gr/2015/papers/SESSION3/WAPCO_3_1.pdf

[4] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*. SIAM, 2009.

[5] A. Griewank and A. Walther, "Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation," *ACM Transactions on Mathematical Software (TOMS)*, vol. 26, no. 1, pp. 19–45, 2000.

[6] ——, *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[7] U. Naumann, *The art of differentiating computer programs: an introduction to algorithmic differentiation*. SIAM, 2012, vol. 24.

[8] J. Deussen, J. Riehme, and U. Naumann, "Automation of significance analyses with interval splitting," in *Parallel Computing: On the Road to Exascale, Proceedings of the International Conference on Parallel Computing, ParCo 2015, 1-4 September 2015, Edinburgh, Scotland, UK*, 2015, pp. 731–740.

[9] ——, "Interval-adjoint significance analysis: A case study," *WAPCO*, 2015. [Online]. Available: http://wapco.e-ce.uth.gr/2016/papers/SESSION2/wapco2016_2_4.pdf

[10] A. Neumaier, *The wrapping effect, ellipsoid arithmetic, stability and confidence regions*. Springer, 1993.

[11] V. Vassiliadis, J. Riehme, J. Deussen, K. Parasyris, C. D. Antonopoulos, N. Bellas, S. Lalis, and U. Naumann, "Towards automatic significance analysis for approximate computing," in *Proceedings of the 2016 International Symposium on Code Generation and Optimization, CGO 2016, Barcelona, Spain, March 12-18, 2016*, 2016, pp. 182–193.

[12] V. Vassiliadis, K. Parasyris, S. Lalis, and C. D. A. abd Nikolaos Bellas, "D2.3: Advanced compiler implementation," Center for Research and Technology Hellas, Tech. Rep., April 2016. [Online]. Available: http://www.scorpio-project.eu/wp-content/uploads/2016/04/Scorpio_D2.3.pdf

[13] M. M. Putko, P. A. Newman, A. C. T. III, and L. L. Green, "Approach for uncertainty propagation and robust design in CFD using sensitivity derivatives," 2001.

[14] B. Christianson and M. Cox, "Automatic propagation of uncertainties," in *Automatic Differentiation: Applications, Theory, and Implementations*. Springer, 2006, pp. 47–58.

[15] D. Ghate and M. B. Giles, "Inexpensive Monte Carlo uncertainty analysis," *Recent Trends in Aerospace Design and Optimization. Tata McGraw-Hill, New Delhi*, pp. 203–210, 2006.

[16] M. Menshikova, "Uncertainty estimation using the moments method facilitated by automatic differentiation in Matlab," Ph.D. dissertation, 2010.

[17] M. Beckers, "Toward global robust optimization," Ph.D. dissertation, Hochschulbibliothek der Rheinisch-Westfälischen Technischen Hochschule Aachen, 2014.

[18] R. M. Gower and A. L. Gower, "Higher-order reverse automatic differentiation with emphasis on the third-order," *Mathematical Programming*, vol. 155, no. 1-2, pp. 81–103, 2016.

[19] G. Gundersen and T. Steihaug, "Sparsity in higher order methods for unconstrained optimization," *Optimization Methods and Software*, vol. 27, no. 2, pp. 275–294, 2012.

[20] A. H. Gebremedhin, F. Manne, and A. Pothen, "What color is your Jacobian? Graph coloring for computing derivatives," *SIAM review*, vol. 47, no. 4, pp. 629–705, 2005.

[21] R. C. Smith, *Uncertainty quantification: theory, implementation, and applications*. SIAM, 2013, vol. 12.