Gstalter *et al.*
Adv. Model. and Simul. in Eng. Sci. (2020) 7:17
https://doi.org/10.1186/s40323-020-00151-8

Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

Toward new methods for optimization study in automotive industry including recent reduction techniques

Etienne Gstalter[1*], Sonia Assou[1], Yves Tourbier[1] and Florian De Vuyst[2]

*Correspondence:
etienne.gstalter@renault.com
[1] Renault, API: FRTCRRU C427, 1 Avenue du Golf, 78280 Guyancourt, France
Full list of author information is available at the end of the article

Abstract

In the last years, the automotive engineering industry has been deeply influenced by the use of «machine learning» techniques for new design and innovation purposes. However, some specific engineering aspects like numerical optimization study still require the development of suitable high-performance machine learning approaches involving parametrized Finite Elements (FE) structural dynamics simulation data. Weight reduction on a car body is a crucial matter that improves the environmental impact and the cost of the product. The actual optimization process at Renault SA uses numerical Design of Experiments (DOE) to find the right thicknesses and materials for each part of the vehicle that guarantees a reduced weight while keeping a good behavior of the car body, identified by criteria or sensors on the body (maximum displacements, upper bounds of instantaneous acceleration …). The usual DOE methodology generally uses between 3 and 10 times the numbers of parameters of the study (which means, for a 30-parameters study, at least 90 simulations, with typically 10 h per run on a 140-core computer). During the last 2 years, Renault's teams strived to develop a disruptive methodology to conduct optimization study. By 'disruptive', we mean to find a methodology that cuts the cost of computational effort by several orders of magnitude. It is acknowledged that standard DoEs need a number of simulations which is at least proportional to the dimension of the parameter space, leading generally to hundreds of fine simulations for real applications. Comparatively, a disruptive method should require about 10 fine evaluations only. This can be achieved by means of a combination of massive data knowledge extraction of FE crash simulation results and the help of parallel high-performance computing (HPC). For instance, in the recent study presented by Assou et al. (A car crash reduced order model with random forest. In: 4th International workshop on reduced basis, POD and PGD Model Reduction Techniques—MORTech 2017. 2017), it took 10 runs to find a solution of a 34-parameter problem that fulfils the specifications. In order to improve this method, we must extract more knowledge from the simulation results (correlations, spatio-temporal features, explanatory variables) and process them in order to find efficient ways to describe the car crash dynamics and link criteria/quantities of interest with some explanatory variables. One of the improvements made in the last months is the use of the so-called Empirical Interpolation Method (EIM, [Barrault et al.]) to identify the few time instants and spatial nodes of the FE-mesh (referred to as magic points) that "explain" the behavior of the body during the crash, within a dimensionality reduction approach. The EIM method replaces a former *K*-Means algorithm (Davies et al. in IEEE Trans Pattern Anal Mach Intell, 1(2):224–227, 1979) which was processed online, for each ROM. Instead, the computation of EIM method is done offline, once for all, for each simulation. This new method allows us to compute a ROM quite faster, and to reduce the number of features that we use for the regression step (~ 100). The nonlinear regression step is achieved by a standard Random Forest (RF, [Breiman. Mach Learn 45:5–32, 2001]) algorithm. Another improvement of the method is the characterization of numerical features describing the shape of the body, at a nodal scale. The characteristics of orientation of the elements surrounding a mesh node must be taken into account to describe the behavior of the node during the crash. The actual method integrates some numerical features, computed from the orientation of the elements around each node, to explain the node behavior. The paper is organized as follows: The introduction states the scientific and industrial context of the research. Then, the ReCUR Method is detailed, and the recent improvements are highlighted. Results are presented and discussed before having some concluding remarks on this piece of work.

Keywords: Car crash simulation, Optimization, Shape (thickness), Nonintrusive data-driven model, Reduced-order modeling (ROM), CUR factorization, Feature extraction, Empirical interpolation method (EIM), Random forest, Regression

Introduction

Scientific context

The increasing amount of data produced by the Finite Elements (FE) solvers and the continuous seek for accuracy in the numerical models have led to the need of "understanding the data". The ability to understand a complex system through a big amount of data, to determine tendencies or correlations by some artificial intelligence algorithms is strategic—let say vital—in industry nowadays.

In the past decade, various numerical methods to reduce the dimensionality of the digital twin of a system have been proposed.

There are two main classes of reduction methods. The class of intrusive methods tries to directly reduce the system of equations to solve, by means of projection techniques over a reduced basis of suitable functions (projection-based Galerkin methods, weighted residual methods) or interpolation techniques (collocation methods with suitable reduced-order bases). The process of derivation of intrusive ROMs may be a heavy task involving strong code development efforts into an open source code.

The other class is the so-called nonintrusive approach. It is based on the analysis of a data archive (database) of simulation results (coming from high-fidelity structural dynamics FE codes). The idea is to extract knowledge from the data, in the form of a metamodel that tries to reproduce simulation results with comparable accuracy but in a faster way. Non-intrusive methods are intended to be used for data analysis with pattern search and feature extraction, search of explanatory variables, model selection, nonlinear regression techniques. It can be seen as a 'grey-box' metamodeling technique that makes help of internal states for deriving a mapping between entries (configuration, shape) and outputs (objectives, criteria).

In the context of car-crash numerical analysis involving fast dynamics FE computations, there are many internal variables including displacements, velocities, stresses, and thermodynamic variables. Generally, displacements or incremental displacement variables and only used for the sake of simplicity.

This paper is aimed at introducing an original nonintrusive method where the learning phase only requires few FE crash simulations (say less than 10) while predicting the car crash behavior in a wide range of parameters (say more than 20). We believe that this can be made possible because each car crash dynamic simulation explores a broad area of the state + parameter space during the transient phase. The question is to know how to extract this knowledge and apply it to different configurations.

Industrial context

The Alliance Renault–Nissan–Mitsubishi has always set as a priority the better understanding of car structural dynamics behaviour. In the automotive industry, each physical crash test costs between \$20 and \$90 K, whereas a car crash simulation costs less than \$10. The numerical revolution of the twenty-first century has introduced many strategic numerical tools for car crash simulation, with models that are closer to the reality and under reasonable computational costs.

The increasing capacity of computing resources has led to parallel High-Performance Computing (HPC), to run crash simulations in 7 h in 2019, instead of 12 h in the past 5 years.

However, the complexity of the models is also increasing (from 3 to 15 million Finite Element degrees-of-freedom) due to the sake of accuracy, and the optimization search space gets wider and wider. From 10 parameters in the past decade, some optimization problem now has hundreds of parameters (thicknesses, materials, presence/absences of parts …). Let us remark that the number of edges of a hypercube of dimension 300 is $\sim 10^{90}$, i.e. roughly the number of particles in our universe.

The actual method used by Renault's engineers is based on a DoE (Design of computer Experiment), with multiple sets of parameters. The simulations are run using a FE high-fidelity solver on HPC facilities. Then from the results a surrogate model is built (response surface methodology, RSM). The optimum of the cost function is then searched, and a new FE simulation is run to check the accuracy of the surrogate model. If the current design vector matches the specifications, the process stops. If not, a new response surface is computed thanks to the new simulation point (active learning) (Fig. 1).

In order to build the surrogate model, one needs many simulations (3 to 10 times the number of parameters), and its cost is important. Moreover, it uses only a small percentage of the data from the simulation.

To better our optimization process, we needed to create a method that get satisfying results with only 2 or 3 simulations, by using a large amount of data from each simulation.

Methods

The ReCUR method

The ReCUR method stands for Regression-CUR [1, 2], a regression technique based on a CUR approximation, used to extract specific rows and columns of the matrix and to compute a regression with those extractions as features (Fig. 2).

The actual regression problem is solved by Random Forest-type algorithms.

The main goal of such a method is to predict various nodal fields during a car crash.

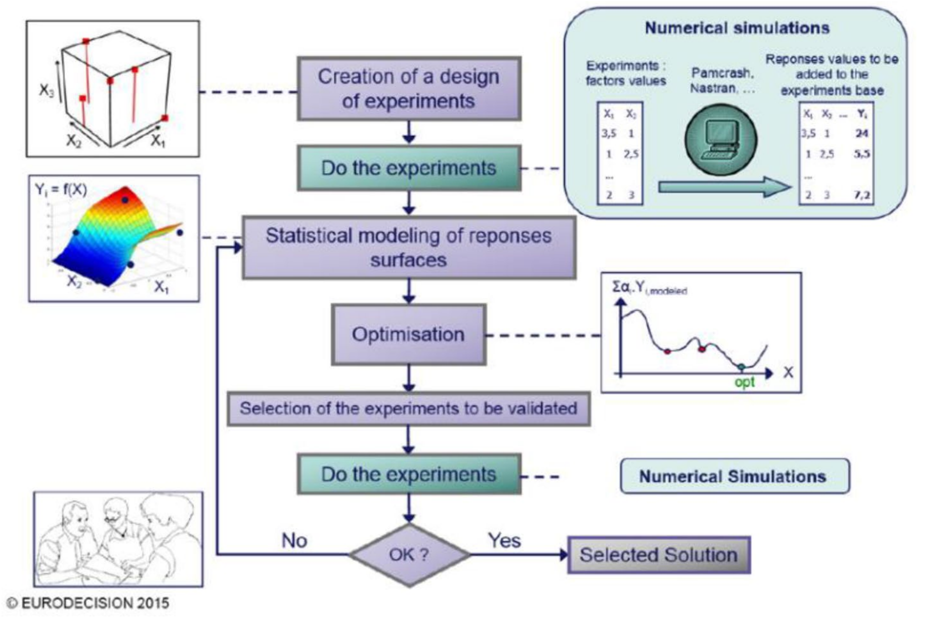
Creation of a design of experiments
Do the experiments
Statistical modeling of reponses surfaces
Optimisation
Selection of the experiments to be validated
Do the experiments
OK ?
No
Yes
Selected Solution
Numerical simulations
Experiments : factors values
Pamcrash, Nastran, ...
Reponses values to be added to the experiments base
Numerical Simulations
$Y_1 = f(X)$
$\Sigma\alpha_i.Y_{i,modeled}$
opt
© EURODECISION 2015

Fig. 1 Optimization process at Renault

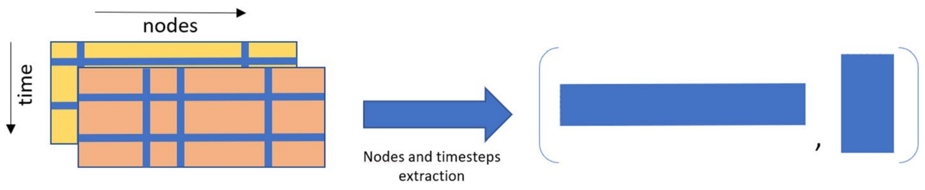

Fig. 2 Nodes and timesteps extraction from field data matrices

The first step of the method is to define which field we want to predict on the new simulation. According to that field, we are going to select some other fields that can explain the prediction field. For instance, if we want to focus on the displacement prediction on the X-axis, we can consider that the displacement on X, Y and Z-axis are explicative fields. Any other nodal explicative field can be added, as long as it is filtered to avoid noise.

For each of those fields, we are going to extract some rows (corresponding to time steps) and columns (corresponding to node records of the FE mesh) that may be useful to explain the evolution of the field. These time-step and node selections will be considered thereafter as features in a Random Forest (RF) [3] regression that computes the prediction of a target field with information from explicative ones.

The original CUR method [4] is the approximation of a rectangular matrix A, by the product of three matrices C, U, and R such as:

$$A \approx CUR \tag{1}$$

where C is an extraction of *columns* from A, R an extraction of *rows* from A and U a (reduced) coefficient matrix.

In our method, we only keep the fact that the function basis is separable. All the difficulty of the method now remains in the way we choose C and R.

A clustering method [5] (K-Means) was first implemented [6], to find some rows and columns that correctly approximate the initial matrix.

However, we had to run the clustering for each ROM, in the online phase, and the computation was very dependent of the matrix (field data) and the numbers of clusters considered.

The computational time of the nodes and timesteps of interest was a large part of the global computational time (between 60 and 70%). The average duration of a ROM computing was 60 min on 28 cores. The ROM is saved into a binary file, that represents about 10 GB.

In the present paper, the Empirical Interpolation Method (EIM, [7, 8]) replaces the clustering procedure by an offline phase. It will be detailed in the next section.

Let us consider simulation data with nn nodes on the FE mesh and nt timesteps.

Once we have a list of n_R rows (each one containing information on one timestep) and n_C columns (each one containing information on one node), we can construct a regression basis. Considering that we need as many individuals as possible, with a reasonable number of features, we decide to create two matrices, as following:

$$R^T \otimes \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{nt} \; and \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{nn} \otimes C. \tag{2}$$

With $\otimes$ representing the Kronecker product, defined as follows:

Considering A a matrix of size $(m \times n)$ and B a matrix of size $(p \times q)$:

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}.$$

The two matrices of (2) both contains $(nn \times nt)$ rows.

Thus, let's build the regression basis as the concatenation of the two previous matrices:

$$X_0 = \left[R^T \otimes \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{nt} ; \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{nn} \otimes C \right] \tag{3}$$

X_0 contains $(nn \times nt)$ rows and $(n_R + n_C)$ columns.

The framework of study involves the different sheet thicknesses as parameters: we want to find the optimal solution, i.e. the best set of thicknesses for the crash scenario.

In order to interpolate in the parameter space, we build a set of μ^i matrices, for each parameter. Each μ^i matrix contains nn columns and nt rows that describes the evolution of one parameter.

In the following paper, the thickness is the only parameter considered.

Regarding the thickness, we can get a thickness nodal value (the mean of the elements' thicknesses around the node considered) and build the μ matrix of thicknesses as follows:

$$\mu = \begin{pmatrix} \mu_{1,1} & \cdots & \mu_{1,nn} \\ \vdots & \ddots & \vdots \\ \mu_{nt,1} & \cdots & \mu_{nt,nn} \end{pmatrix} \tag{4}$$

with $\mu_{i,j}$ the thickness at the ith timestep at the jth node.

We can now build μ_R and μ_C, as extractions of μ for both sets of clusters:

$$\mu_R = \begin{pmatrix} \mu_{1,1} & \cdots & \mu_{1,nn} \\ \vdots & \ddots & \vdots \\ \mu_{n_R,1} & \cdots & \mu_{n_R,nn} \end{pmatrix} \quad \text{and} \quad \mu_C = \begin{pmatrix} \mu_{1,1} & \cdots & \mu_{1,n_C} \\ \vdots & \ddots & \vdots \\ \mu_{nt,1} & \cdots & \mu_{nt,n_C} \end{pmatrix} \tag{5}$$

with nn the number of nodes, nt the number of timestep, n_C the number of "node" clusters C and n_R the number of "time" clusters R. A crash simulation result file usually contains between 80 and 150 timesteps. Since the number of EIM modes cannot exceed the rank of the data matrix, $1 < n_C, n_R < 150$.

The regression matrix is obtained by the product of X_0 with the design parameters considered (in our case thicknesses): each cell of X_0 is multiplied by the thickness of the corresponding column (thickness of the cluster at specific time and node) and corresponding row (the thickness of the node at the specific timestep).

In the current studies, we consider the initial thickness, in order to be able to interpolate and to predict without needing the entire simulation. Thus, in Eq. (4): $\mu_{1,k} = \mu_{2,k} = \ldots = \mu_{n_R,k} \forall k \in [1 : nn]$.

Let us consider, with E(x) the integer part of x:

$$\forall i \in 1, nt.nn, n_i = E\left(\frac{i}{nn}\right) + 1. \tag{6}$$

Let us define the $((nn \times nt), (n_C + n_R))$ matrix $\widetilde{\mu}$ as:

$$\forall (i,j) \in (1, nt.nnx1, n_C + n_R), \widetilde{\mu}_{i,j} = \mu_{1,n_i}. \tag{7}$$

Considering (.) the Hadamard product, with $R^{\mu} = R \cdot \mu_R$ and $C^{\mu} = C \cdot \mu_C$:

$$X = \left[\left(R^{\mu T} \otimes \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{nt} \right); \left(\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{nn} \otimes C^{\mu} \right) \right] . \widetilde{\mu}. \tag{8}$$

To summarize, each cell of X_0 has been multiplied by thicknesses from the cluster and the node of the corresponding row. Then, we can add as many columns in X as we want, each added column is a custom regression feature for the RF algorithm.

Prediction matrices are created in order to be able to predict new configurations with the reduced-order model. They are computed in the exact same way, with the new set of thicknesses.

For instance, an average study uses 6 Million FE models, with 150 timestep computations, we can extract up to 150 EIM modes (times and nodes). Thus, the Random Forest problem has $+300$ features.

One of the improvements of the ReCUR method is the use of a nodal indicator of the shape: by extracting the coordinates of each node of the mesh from the simulation data, we can compute the coordinates of the normal to each finite element.

Thus, we can compute a tuple of mean coordinates from the surrounding elements' normal.

This artefact, that could be seen as a node "orientation" helps us to dissociate the behaviour of two nodes with the same thickness, the same material, but different orientations of elements in space.

The target column of the regression matrix is the target field, transformed from a matrix to a big column by staking all the matrix column vectors.

The ReCUR method with empirical interpolation

In the present paper, EIM [7, 8] replaces the clustering procedure by an offline phase.

The original EIM [7] extracts empirical interpolation points of a function within a greedy procedure. From a set of functions, EIM builds both interpolation points (magic points) and interpolation functions that minimize the worst case of interpolation error. Each iterate of EIM adds a new empirical interpolation point and a new interpolation function that corrects the worst interpolation error at the previous EIM iterate. At the end of the EIM procedure, we get an interpolation that controls the interpolation error within all functions of the set in a uniform way.

We decided to adapt a Discrete-EIM (DEIM, [9]) algorithm to match our need of a list of nodes and timesteps in the field matrix, that could be used as features in the RF model [10].

The algorithm is presented below:

Let us consider ϕ_i as the *ith* component of the interpolator $\Im$, identified by the equality on the interpolation point, and $\vec{x}$ as a column vector of A, $(\vec{x})_i$ the *ith* component of the vector.

Initialisation:

- Find the maximum absolute value of the elements of A.
- Consider i_1 and k_1 the indexes of the line and column that contain this maximum.
- Compute $\overrightarrow{q_1} = \frac{\overrightarrow{x^{k_1}}}{\left(\overrightarrow{x^{k_1}}\right)_{i_1}}$ with x^{k_1} the k_1 column of A.
- Build the interpolator $\Im^1\vec{x} = \phi_1(x).\overrightarrow{q_1}$ with $(\Im^1\overrightarrow{x})_{i_1} = (\overrightarrow{x})_{i_1}$.

For the $\mathrm{j}+1$ iteration:

- Compute $A - \Im^j A$ and identify the maximum value.
- Consider i_{j+1} and k_{j+1} the index of the line and column that contain this maximum.
- Compute $\overrightarrow{q_{j+1}} = \frac{\overrightarrow{x^{k_{j+1}}}}{\left(\overrightarrow{x^{k_{j+1}}}\right)_{i_{j+1}}}$ with $x^{k_{j+1}}$ the k_{j+1} column of $A - \Im^j A$.
- Identify the interpolator basis: $\Im^{j+1}\vec{x} = \phi_1(x).\overrightarrow{q_1} + \ldots + \phi_j(x).\overrightarrow{q_j}$ with $(\Im^{j+1}\overrightarrow{x})_{i_{j+1}} = (\overrightarrow{x})_{i_{j+1}}, \ldots, (\Im^{j+1}\overrightarrow{x})_{i_1} = (\overrightarrow{x})_{i_1}$..

Once we have the (i) that refers to several rows of the initial matrix A, we can also compute EIM on the transpose matrix A^T to get some nodes' index.

A "break" criterion has also been implemented, in order to specify a maximum residual instead of a number of modes. In this case, the algorithm stops as soon as the residual is under the specified value.

The main benefit of such an algorithm is its offline properties. The EIM computation is done once and for all, for the maximum number of EIM modes ($rank(A)$).

For each simulation's result, an EIM is done, and the results are available for any ROM we want to produce from the simulations.

Another important benefit of the EIM-based algorithm is its efficiency.

The figure below shows the residual of the projection of A for each iteration:

Considering that there might be some correlation between two pieces of simulation data, we also developed a strategy to compute the EIM modes in an imbricated way.

After the computation of the EIM modes from the 1st data, we apply those modes on the 2nd data, and the computation of the modes for the 2nd data is done on the image of it by the projection from the 1st EIM computation.

The 3rd data is processed with the EIM modes from calculus 1 and 2, and so on.

Results and discussion

The following case is a true frontal crash study, on a basket of 34 parts of a car body structure, each one designated by its thickness and material. The parameter's field is the thickness of the parts.

Many criteria can be used in the specification of the study: from intrusions (relative displacement of 2 nodes) to maximum plastic strain in shells elements, each criterion defines a particular requirement for the safety of the occupants.

The aim of the study is to reduce the weight of the body structure (the reference design), with a multi-objective criterion: to validate the requirements in terms of nodal intrusion on several zones of the FE mesh, and to keep the highest score of Carry-Over, which indicates the proportion of parts whom thickness has not varied more than $\pm$ 10%. This last criterion comes from the factory process: if the thickness of a part has not varied more than $\pm$ 10%, we can consider that the processes during car production will not be impacted by the variation, which means that there is no increase of cost.

A sample set of 1000 admissible configurations is then generated. The nondominated Pareto graph (Fig. 3) shows the different sets of parameters and the gain of mass for each configuration.

Each point is a configuration, described by its Carry-Over Ratio (y axis) with respect to the gain of mass (x axis). Thus, the reference configuration (100) stands at 1 of CO-ratio and 0 for the gain of mass (Fig. 4).

We decided to choose five sets of parameters and to run each simulation, to get the intrusions for each set (Fig. 5).

The intrusion results are gathered in the Table 1:

It seems that the simulation nb 408 is the best of the 5 runs. According to the Fig. 3, nb 834 allows a bigger weight loss, but the intrusions are not validating the requirements. We now decide to predict with a ROM based on nb 408 and nb 834 simulations, the sets

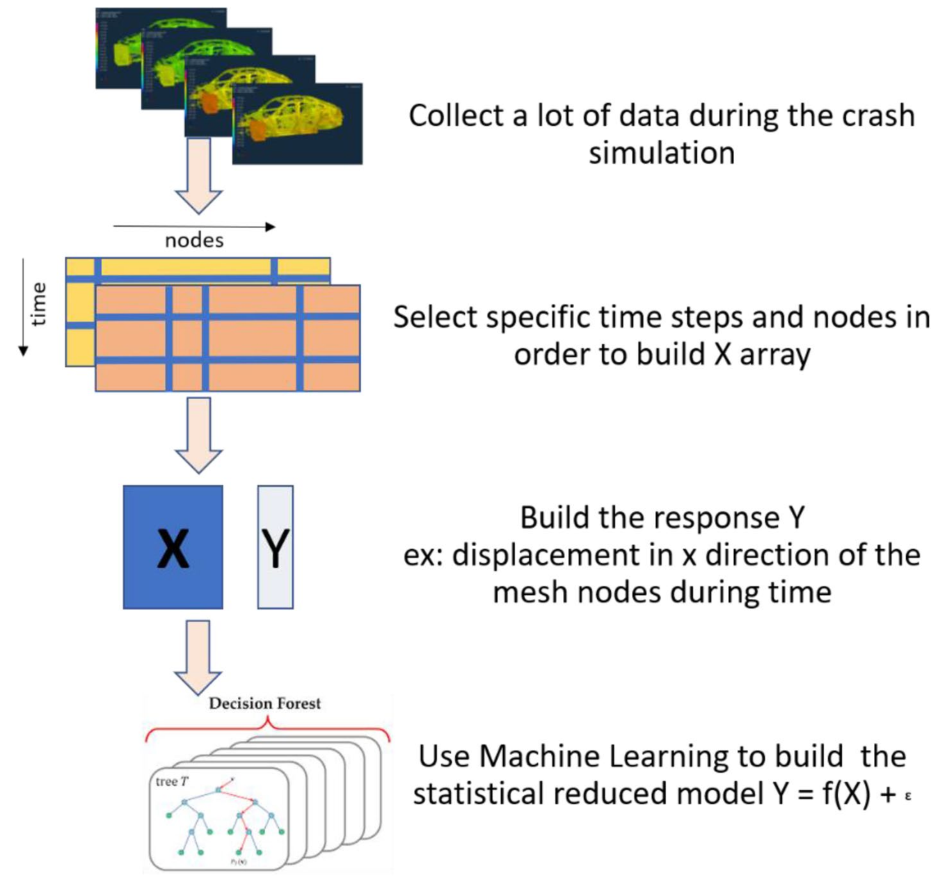
Collect a lot of data during the crash simulation
nodes
time
Select specific time steps and nodes in order to build X array
X
Y
Build the response Y
ex: displacement in x direction of the mesh nodes during time
Decision Forest
tree T
Use Machine Learning to build the statistical reduced model Y = f(X) + ε

Fig. 3 ReCur process with EIM-based vector extraction

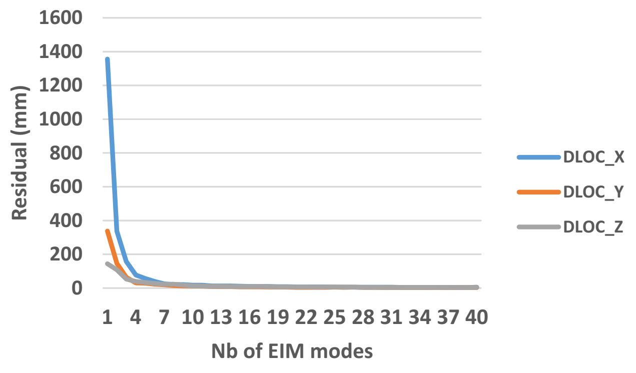

Fig. 4 Temporal evolution of the residual on three directions (408). DLOC refers to the local displacement field, on X, Y, and Z axis

of parameters that allows a bigger mass loss than nb 408, without decreasing the carry-over ratio.

In every ROM, one node at one timestep represents one statistical unit.

As oppose to most methods, which uses a small proportion of the data to validate, the ROM created with ReCUR is trained with 30% of the data from the nb 408 and nb

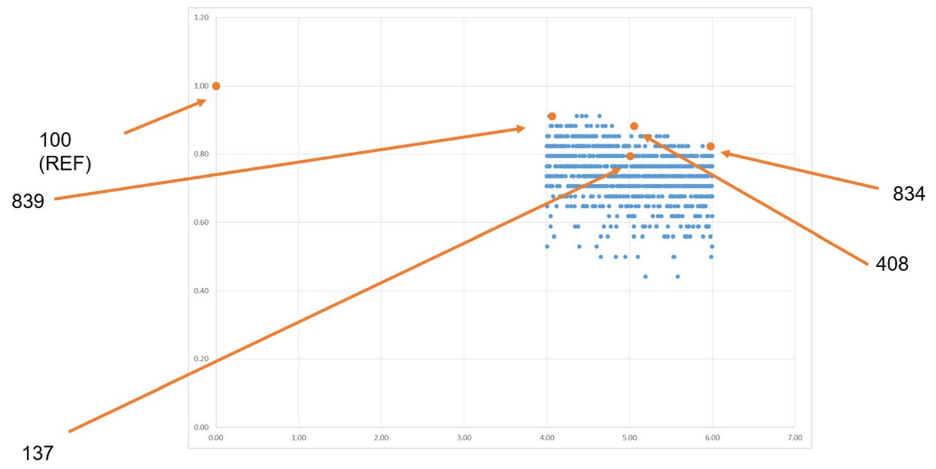

Fig. 5 The Pareto graph of the study, with the selected points in orange

834 simulation (randomly drawn) This proportion is due to the fact that we have a big amount of data (1 to 10 million units). The remaining 70% is used for validation.

Each tree of the forest is randomized by the bootstrapping units, and for each node of the tree, there is also a bootstrap on features. Among the +300 features, only 20 features, and 1% of the units are drawn to build a regression tree. It is only by repeating this process more than 200 times that we reduce the variance of the ROM, in order to get information from each available feature (Fig. 6).

The following figure represents the ROM on the validation set:

The error of prediction (5%) on the validation set is similar to the one on the training set. However, 5% may represent more than 30 mm of displacement on the car body. Considering that some features may be missing, we are conscious that the ROM may not be accurate but will sort the 1000 configurations by their responses. Considering that the numerical dispersion of the solver is often more than 5%, it would not be thoughtful to have a more accurate ROM.

Once the ROM is computed, we predict the intrusion of each point in the set of configurations. The three sets with best predictions (according to intrusion analysis) are identified and plotted on the Pareto graph.

Each point still represents one configuration, described by its Carry-Over Ratio (y axis) with respect to the gain of mass (x axis). Thus, the reference configuration (100) stands at 1 of CO-ratio and 0 for the gain of mass (Fig. 7).

The ROM predicts the following intrusions:

The results of the prediction with the (408–834) ROM are much higher than the intrusion in those simulation. It is explained by the actual Regression method, Random Forest, which sort each value (node, timestep) according to its features values. Some individuals with similar feature values have been classified with the nodes we are following, decreasing the accuracy of the model by the difference between their response and our nodes'.

According to the latest result, the better set is the nb 659, even though it does not seem to validate the requirements.

A simulation of the configuration nb 659 is run on solver, to check the "real" intrusions:

Table 1 Intrusion analysis results on the 5 sets of parameters, each column representing a specification criterion

ODB65_Tip_toe_intrusion_1	ODB65_Tip_toe_intrusion_2	ODB65_Tip_toe_intrusion_3	ODB65_Tip_toe_intrusion_4	ODB65_Heel_Intrusion_1	ODB65_Heel_Intrusion_2	ODB65_Heel_Intrusion_3	ODB65_Heel_Intrusion_4	ODB65_Brake_Pedal_Intrusion_1	ODB65_Brake_Pedal_Intrusion_2
95	*95*	*95*	*95*	*50*	*50*	*50*	*50*	*130*	*130*
79	92	***108***	82	15	19	28	16	90	80
72	82	***96***	76	13	16	24	10	78	68
80	92	***106***	87	17	21	30	18	90	79
81	89	93	67	16	19	28	15	88	77
78	88	***98***	75	14	18	26	14	87	77

OODB65_Brake_Pedal_Intrusion_3	ODB65_Brake_Pedal_Intrusion_4	ODB65_Clutch_Pedal_Intrusion_1	ODB65_Clutch_Pedal_Intrusion_2	ODB65_Doorway_reduction_Lower	ODB65_Doorway_reduction_Upper	ODB65_BAV_X_Reduction_AV_B_Pillar_G	ODB65_BAV_X_Reduction_AR_B_Pillar_G	ODB65_Steering_Column_A_Point_Disp_X_Max	Number of configuration
130	*130*	*130*	*130*	*20*	*20*	*10*	*2*	*20*	Spec
91	108	99	93	6	***26***	1	2	0	137
79	97	89	83	6	***26***	1	2	0	839
89	107	102	95	6	***30***	2	2	0	834
81	100	101	96	7	***28***	2	2	0	408
84	102	98	93	6	***26***	1	2	0	100

Bolditalic signifies values higher than specifications

Italic values are specification values

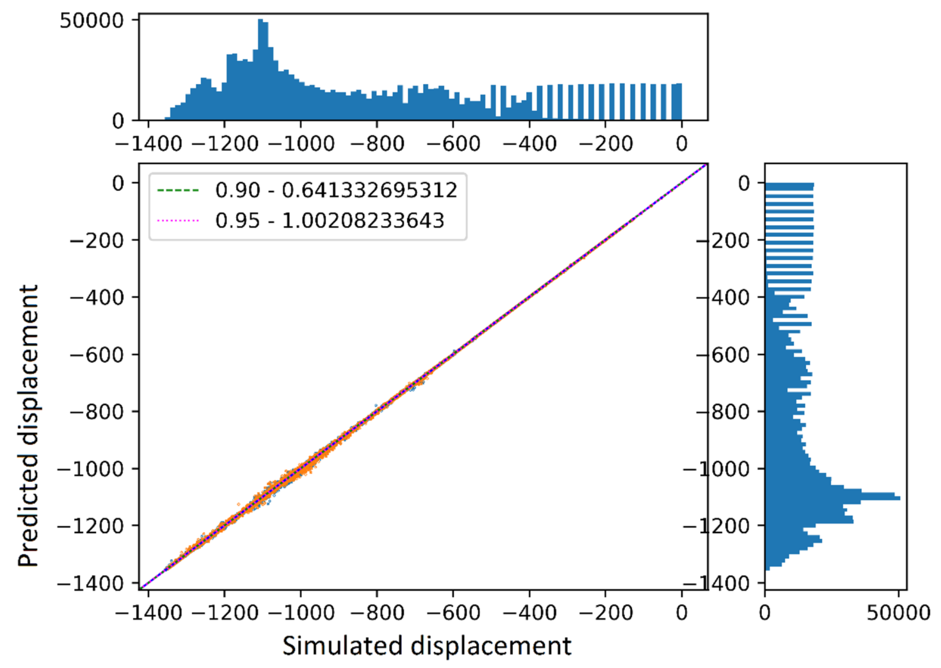

Fig. 6 ROM prediction on validation set

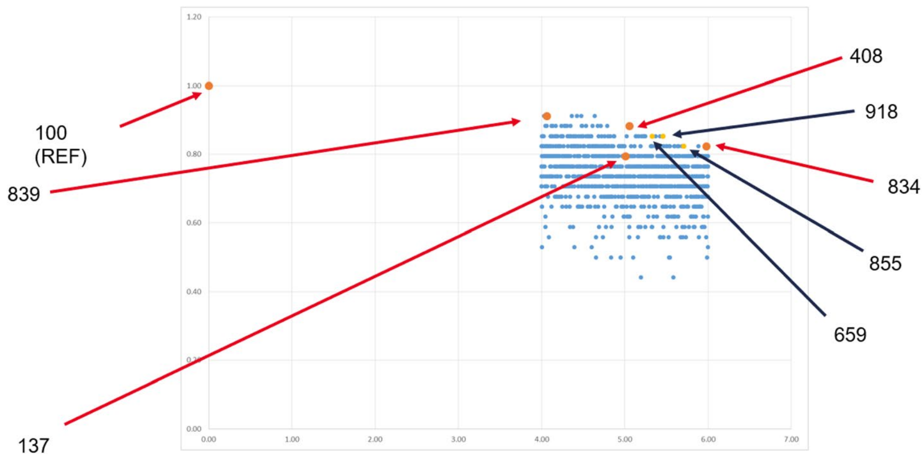

Fig. 7 The Pareto graph of the study, with the 3 new points in black

It seems that the set nb 659 validates the requirements, with a reasonable carry-over ration (85%) and a good loss of weight: −5.3 kg (the whole part basket represents 77 kgs).

The set of parameters nb 659 is a specification-ok point found with only 5 runs.

We can observe that the real intrusion of nb 659 quite different from the ROM prediction of Table 2. Such differences are due to the number of computations used in the ROM (2) and the severe difference between the two behaviours in crash of those points. The accuracy of the ROM may be increased by adding new features to the X matrix, among which materials, spotwelds, or part geometry.

Table 2 Predictions of intrusions for sets 918, 855,659, each column representing a specification criterion

ODB65_Tip_toe_ intrusion_1	ODB65_Tip_toe_ intrusion_2	ODB65_Tip_toe_ intrusion_3	ODB65_Tip_toe_ intrusion_4	ODB65_Heel_ Intrusion_1	ODB65_Heel_ Intrusion_2	ODB65_Heel_ Intrusion_3	ODB65_Heel_ Intrusion_4	ODB65_Brake_ Pedal_Intrusion_1	ODB65_ Brake_Pedal_ Intrusion_2
95	*95*	*95*	*95*	*50*	*50*	*50*	*50*	*130*	*130*
121	***147***	***145***	85	16	25	34	20	129	114
102	***122***	***127***	79	14	21	30	16	113	100
88	***104***	***121***	88	15	22	32	19	103	92
OODB65_Brake_ Pedal_Intrusion_3	**ODB65_Brake_ Pedal_Intrusion_4**	**ODB65_Clutch_ Pedal_Intrusion_1**	**ODB65_Clutch_ Pedal_Intrusion_2**	**ODB65_Doorway_ reduction_Lower**	**ODB65_Doorway_ reduction_Upper**	**ODB65_BAV_X_ Reduction_AV_B_ Pillar_G**	**ODB65_BAV_X_ Reduction_AR_B_ Pillar_G**	**ODB65_ Steering_ Column_A_ Point_Disp_X_ Max**	**Number of configuration**
130	*130*	*130*	*130*	*20*	*20*	*10*	*2*	*20*	Spec
123	***148***	***147***	***136***	6	16	2	2	0	918
108	130	125	117	6	19	2	2	0	855
101	121	116	108	6	14	1	2	0	659

Bolditalic signifies values higher than specifications

Italic values are specification values

Table 3 Real intrusion for 659 set., each column representing a specification criterion

ODB65_ Tip_toe_ intrusion_1	ODB65_ Tip_toe_ intrusion_2	ODB65_ Tip_toe_ intrusion_3	ODB65_ Tip_toe_ intrusion_4	ODB65_ Heel_ Intrusion_1	ODB65_ Heel_ Intrusion_2	ODB65_ Heel_ Intrusion_3	ODB65_ Heel_ Intrusion_4	ODB65_ Brake_ Pedal_ Intrusion_1	ODB65_ Brake_ Pedal_ Intrusion_2	ODB65_ Brake_ Pedal_ Intrusion_3	ODB65_ Brake_ Pedal_ Intrusion_4	ODB65_ Clutch_ Pedal_ Intrusion_1	ODB65_ Clutch_ Pedal_ Intrusion_2	Number of configuration
95	*95*	*95*	*95*	*50*	*50*	*50*	*50*	*130*	*130*	*130*	*130*	*130*	*130*	Spec
66	74	77	44	17	15	13	1	46	30	43	72	71	56	659

Italic values are specification values

Concluding remarks

The new method of extraction of the index of nodes and snapshots using EIM allows us to compute a ROM in very little time (20 min on 24 cores, instead of 60 min with the previous clustering method), and even though the accuracy deserves to be better in the actual models, the ROM helps the engineering teams in the optimization loop, to select several runs to study, step by step (Table 3).

The previous DOE study would have cost 100+ simulations, the ROM study has cost only 6 simulations to find a configuration that matches the specifications.

The current method is a new approach for decision-making processes into the Alliance. It allows engineers to explore the design space in a faster way by means of a ROM-based surrogate model that strongly reduces the number of calls of fine car crash simulations. During the process, a crash simulation is used to validate the new candidate design point, but also to build a new ROM surrogate model for the next step.

Perspectives

The issue of parallelization of the EIM algorithm will be addressed in the next developments. It will allow to improve the reduction of the computational time of the offline phase.

The current version of ReCUR focuses on a single crash scenario (frontal crash or rear crash for instance). In an optimization study, several scenarios must be considered, to make sure that the structure behavior is satisfying regarding every possible case: frontal, lateral, rear crash, vibrations. The future version of ReCUR must consider the multiplicity of scenarios.

In the last study, the thickness was the only parameter of interpolation. Considering multiples types of parameters—such as materials, local shape and neighbourhood—is also a lead to improve the ReCUR method.

Acknowledgements
Not applicable.

Authors' contributions
EG computed the EIM-related part of the method, conducted the study and analysed the results. SA, EG and YT computed the ReCUR method. FdV conducted the initial work on the adaptation of EIM methods on crash studies. All authors read and approved the final manuscript.

Funding
CIFRE Scholarship, funded by Renault SAS.

Availability of data and materials
The datasets generated during the current study are not publicly available due to industrial property but are available from the corresponding author on reasonable request.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] Renault, API: FRTCRRUC427, 1 Avenue du Golf, 78280 Guyancourt, France. [2] Laboratoire LMAC, Université de Technologie de Compiègne, Alliance Sorbonne Université, API: FRTCRRUC427, 1 Avenue du Golf, 78280 Guyancourt, France.

Received: 8 November 2019 Accepted: 20 March 2020
Published online: 07 April 2020

References
1. Charrier M, Jezequel L, Dessombz O, Tourbier Y. Strategic decision support through combinatorial optimization with costly evaluation function. In: NAFEMS conference. 2017.

2. Le Guennec Y, Brunet J-P, Zohra Daim F, Chau M, Tourbier Y. A parametric and non-intrusive reduced order model of car crash simulation. Comput Methods Appl Mech Eng. 2018;338:186–207. https://doi.org/10.1016/j.cma.2018.03.005.
3. Breiman L. Random forests. Mach Learn. 2001;45:5–32. https://doi.org/10.1023/A:1010933404324.
4. Goreinov SA, Zamarashkin NL, Tyrtyshnikov EE. Pseudo-skeleton approximations by matrices of maximal volume. Math Zametki. 1997;62(4):619–23. https://doi.org/10.4213/mzm1644.
5. Davies DL, Bouldin DW. A cluster separation measure. IEEE Trans Pattern Anal Mach Intell. 1979;1(2):224–7.
6. Assou S, Charrier M, Gstalter E Brihi C, Jézéquel L, Dessombz O, Tourbier Y. A car crash reduced order model with random forest. In: 4th International workshop on reduced basis, POD and PGD Model Reduction Techniques—MORTech 2017. 2017.
7. Barrault M, Maday Y, Nguyen NC, Patera A. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. C R Acad Sci Paris Ser I. 2004. https://doi.org/10.1016/j.crma.2004.08.006.
8. Eftang JL, Grepl MA, Patera AT. A posteriori error bounds for the empirical interpolation method. C R Math. 2010;348:575–9.
9 Saifon C, Danny S. Nonlinear model reduction via discrete empirical interpolation. Soc Ind Appl Math. 2010;32(5):2737–64.
10. Fritzen F, Haasdonk B, Ryckelynck D, Schöps S. An algorithmic comparison of the hyper-reduction and the discrete empirical interpolation method for a nonlinear thermal problem. Math Comput Appl. 2018. https://doi.org/10.3390/mca23010008.

Publisher's Note