GLOBAL DISCRETE EMPIRICAL INTERPOLATION METHOD FOR NONLINEAR MODEL ORDER REDUCTION

GIOVANNI DE LUCA* AND MICHIEL E. HOCHSTENBACH*

Abstract. In the context of model order reduction (MOR) for nonlinear problems, we propose a variant of discrete empirical interpolation method (DEIM), called global DEIM (GDEIM). Since DEIM is a nearly-optimal method, i.e., the error of the reduction depends on the number and location of the selected nonlinearities to evaluate, we show that, by employing an alternative procedure to select the points, we sometimes obtain a better accuracy of the reduction with GDEIM for the same number of points as for DEIM. This is useful especially when one has restrictions on the dimension of the reduced order model (ROM) to simulate or, the other way around, when a better accuracy can be reached for a specified reduction order. To compare the accuracy level between GDEIM and DEIM before proceeding with the reduction, i.e., when the projection matrix is formed, we make use of a heuristic error estimator cheap to evaluate, which in most of our experiments is able to determine which of the two methods is best. This can speedup simulations of the obtained ROM, when inputs and/or parameters are swept for analysis. We demonstrate the usefulness of GDEIM to nonlinear, parametric, analytic functions and provide a comparison with DEIM by using our error estimator. The second contribution of this work is a mathematical formulation which extends (G)DEIM to general nonlinear function, evaluated non-componentwise. We provide accuracy and speedup results from using our extended formulation to an example for time-domain circuit simulation, described by a set of differential algebraic equations. This may be a promising direction to speedup analysis of more complex and large circuits, where it is common to have millions of unknowns in the vector-solution.

Key words. Nonlinear model reduction; global discrete empirical interpolation method; ordinary differential equations; differential algebraic equations; circuit simulation.

AMS subject classifications. 65L02, 65M02

1. Introduction.

In computational sciences, the size of systems under analysis can be very large when a high accuracy of the simulation is required, of the order of millions or even more unknowns. Such problems can be encountered in many application areas such as fluid dynamics [7], aerodynamics [3], circuit simulation [11], batch chromatography [12], etc. Many of these problems become intractable even on modern computing machines because of the huge requirements for memory and CPU time. Projection-based MOR techniques seek to reduce computational complexity of numerical problems by reducing the number of the unknowns for which the problem has to be solved. The ROM is expected to approximate the original one with a specified tolerance.

A suitable technique for nonlinear and/or parametric systems of equations is the proper orthogonal decomposition (POD) (see [8] and [9] for further details). It makes use of a Galerkin condition, forming the projection basis from collected *snapshots* (solutions) of the original model, during the simulation. It has found application in numerous areas such as fluid dynamics and compressible flow. However, its application to nonlinear problems has met difficulties because the computational cost of nonlinear evaluations is not decreased by POD, thereby defeating the objective of the reduction process.

*Version January 15, 2016. Department of Mathematics and Computer Science, TU Eindhoven, PO Box 513, 5600 MB, The Netherlands, www.win.tue.nl/∼gdeluca, www.win.tue.nl/∼hochsten.
The first author benefits from the financial support provided by the Marie Curie Action, under the European project *ASIVA14*. The second author was supported by an NWO Vidi research grant.

Many techniques have been proposed to reduce the computational complexity of evaluating the nonlinear part of the problem. Missing point estimation (MPE) was initially proposed for finite volume discretization [1]. DEIM, a discrete variant of EIM [2], is similar to MPE in the sense that it reduces the cost by choosing only a subset of spatial grid points, which reduces the cost for evaluating the complete L^2 inner products at each discrete time point. Unlike MPE, DEIM attempts to approximate each nonlinear function by using a precomputed coefficient matrix, which makes the cost of nonlinear evaluations proportional to the (hopefully small) number of chosen DEIM points, in correspondence of selected components of the vector-solution.

DEIM is worse than an orthonormal projection by no more than a factor $\mathcal{C} = \| (P^{\mathrm{T}}U)^{-1} \|_2$. Although the theoretical upper bound $\mathcal{C}$ is very large, in practical experiments it is always below (say) 100. For that reason, [4] claim that DEIM is a nearly-optimal method. Besides, the error of the reduction depends on the number and location of the selected nonlinearities to update. We show that, even by retaining different points, we sometimes obtain a better accuracy of the reduction with GDEIM. GDEIM approaches the same order of accuracy of DEIM as the number of retained points increases. However, we experience an improvement of accuracy of even 40% with respect to DEIM when the number of selected points is relatively small. By using a heuristic error estimator we can often determine whether GDEIM is more accurate than DEIM and choose the best method among the two for the reduction process before it is employed, with the possibility to save time/memory for further analysis with a smaller ROM. The estimator is cheap to evaluate and seems to be reliable in terms of L^1-, L^2- and L^∞-norm, commonly used in, e.g., statistics and engineering fields. We provide numerical experiments on analytic function in Section 5 to illustrate this.

Besides, the second contribution of this work is a deepening of the interpolation method to general nonlinear functions which cannot be evaluated in a componentwise fashion. In fact, [4] provides a DEIM formulation targeted to nonlinear functions evaluated componentwise. Here, we extend the analysis to functions evaluated non-componentwise. We show results from a numerical example in circuit simulation, which confirms the validity of the general formulation in Section 5.

The rest of the work is organized as follows: in Section 2, we provide the problem formulation; the DEIM/GDEIM theory and comparison of features, as well as the error estimator are in Section 3. We present the extended formulation for general nonlinear functions in Section 4. Numerical examples are in Section 5. Conclusions and future perspective are outlined in Section 6.

2. Problem formulation. The models' equations are sets of nonlinear and/or parametric ODEs (e.g., coming from semi-discretization of PDEs) and set of DAEs (e.g., for circuit simulation). The method is still applicable to nonlinear, parametric algebraic constraints (e.g., from discretization of steady-state problems).

The set of nonlinear ODEs is of the form

$$\frac{d}{dt}\mathbf{y}(t) = \mathbf{A}\mathbf{y}(t) + \mathbf{g}(\mathbf{y}(t)), \tag{2.1}$$

with proper initial conditions, where $\mathbf{y}(t) = [y_1(t), \ldots, y_n(t)]^{\mathrm{T}} \in \mathbb{R}^n$ is the vector-solution, $t \in [0, T]$ is the time variable, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a constant matrix and $\mathbf{g}$ is a vector-valued, nonlinear function evaluated at $\mathbf{y}(t)$ componentwise, i.e., $\mathbf{g} = [g(y_1(t)), \ldots, g(y_n(t))]^{\mathrm{T}} \in \mathbb{R}^n$, where $g : \mathfrak{I} \to \mathbb{R}$ with $\mathfrak{I} \subset \mathbb{R}$.

The set of nonlinear DAEs is of the form

$$\mathbf{D}\frac{d}{dt}\mathbf{y}(t) + \mathbf{A}\mathbf{y}(t) + \mathbf{g}(\mathbf{y}(t)) = \mathbf{b}s(t), \tag{2.2}$$

where $\mathbf{y}(t)$, $\mathbf{A}$ defined as for (2.1), $\mathbf{D} \in \mathbb{R}^{n\times n}$ is a singular, constant matrix and $\mathbf{b} \in \mathbb{R}^n$ is a column-vector (it can be a matrix as well) selecting the input $s(t) \in \mathbb{R}$. For this case, we consider that $\mathbf{g} = [g_1(y_1(t), \ldots, y_n(t)), \ldots, g_n(y_1(t), \ldots, y_n(t))]^{\mathrm{T}} \in \mathbb{R}^n$, with $g_i(\cdot, \ldots, \cdot) : \mathbb{R}^n \to \mathbb{R}$, as well as that $g_i(\cdot, \ldots, \cdot) \neq g_j(\cdot, \ldots, \cdot)$, for $i \neq j$. In this case, the componentwise evaluation does not hold, and the Jacobian of $\mathbf{g}$ is no longer strictly diagonal (as for componentwise function).

When a high accuracy of the simulation is required, the full order model dimension n might become very large, resulting in an intractable simulation time. Thus, MOR techniques generate a ROM with smaller dimension $k \ll n$, by building a projection matrix $\mathbf{V}_k \in \mathbb{R}^{n\times k}$ to approximate the full model solution $\mathbf{y}(t)$, i.e., $\mathbf{y}(t) \approx \mathbf{V}_k\widetilde{\mathbf{y}}(t)$, where $\widetilde{\mathbf{y}}(t) \in \mathbb{R}^k$ is the ROM solution. By projecting the original system onto the subspace spanned by $\mathbf{V}_k$, the ROM of (2.1) is given by

$$\frac{d}{dt}\widetilde{\mathbf{y}}(t) = \widetilde{\mathbf{A}}\widetilde{\mathbf{y}}(t) + \mathbf{V}_k^{\mathrm{T}}\mathbf{g}(\mathbf{V}_k\widetilde{\mathbf{y}}(t)). \tag{2.3}$$

For the DAEs formulation (2.2), we have

$$\widetilde{\mathbf{D}}\frac{d}{dt}\widetilde{\mathbf{y}}(t) + \widetilde{\mathbf{A}}\widetilde{\mathbf{y}}(t) + \mathbf{V}_k^{\mathrm{T}}\mathbf{g}(\mathbf{V}_k\widetilde{\mathbf{y}}(t)) = \widetilde{\mathbf{b}}s(t), \tag{2.4}$$

with $\widetilde{\mathbf{A}} = \mathbf{V}_k^{\mathrm{T}}\mathbf{A}\mathbf{V}_k \in \mathbb{R}^{k\times k}$, $\widetilde{\mathbf{D}} = \mathbf{V}_k^{\mathrm{T}}\mathbf{D}\mathbf{V}_k \in \mathbb{R}^{k\times k}$, and $\widetilde{\mathbf{b}} = \mathbf{V}_k^{\mathrm{T}}\mathbf{b} \in \mathbb{R}^k$. We employ the POD method to compute $\mathbf{V}_k$, since the ROM inherits the dominant features of the original model. However, by exploring the dimensions in the operations involved in (2.3) and (2.4), one can see that the original dimension n is still involved (to form the argument of the nonlinear function and to project it). Another drawback of POD might be that the snapshots do not contain meaningful information of the original model, provided only after the training period. In this regard, we will present challenges targeted to circuits simulation.
Our method GDEIM (as DEIM) selects a smaller number of nonlinear components to update, which translates into a non-complete inner product, speeding up the simulation time while preserving accuracy of the solutions.

3. DEIM and GDEIM. DEIM approximates a nonlinear function $\mathbf{g}(\tau)$ with $\widetilde{\mathbf{g}}(\tau)$ (τ any parameter), such that

$$\mathbf{g}(\tau) \approx \widetilde{\mathbf{g}}(\tau) = \mathbf{U}_p\mathbf{c}(\tau),$$

where $\mathbf{U}_p \in \mathbb{R}^{n\times p}$ is the basis of the subspace which the interpolated nonlinear function is projected onto, with $p \ll n$, and $\mathbf{c}(\tau) \in \mathbb{R}^p$ is the interpolating coefficients vector generated by DEIM. The basis $\mathbf{U}_p = [\mathbf{u}_1, \ldots, \mathbf{u}_p]$ can be obtained from the SVD on the matrix whose columns come from the evaluation of the nonlinear system's part only. This basis also determines the set of interpolation indices $\{\rho_1, \ldots, \rho_p\}$ which serve to interpolate the singular vectors at specific points, chosen so that the reduction error's growth is iteratively limited [4]. The indices are in the form $\mathbf{P} = [\mathbf{e}_{\rho_1}, \ldots, \mathbf{e}_{\rho_p}] \in \mathbb{R}^{n\times p}$ where $\mathbf{e}_j$ is the jth standard basis vector in correspondence of the maximum

value extracted from a residual-vector. Assume that the product $\mathbf{P}^{\mathrm{T}}\mathbf{U}$ is nonsingular (this is ensured by the selection of non-repetitive indices and by assuming the linear independence of the singular vectors in $\mathbf{U}$ [4]), the coefficient vector $\mathbf{c}(\tau)$ is determined uniquely from $\mathbf{P}^{\mathrm{T}}\mathbf{g}(\tau) \approx (\mathbf{P}^{\mathrm{T}}\mathbf{U}_p)\mathbf{c}(\tau)$ thus the oblique projection is used here

$$\widetilde{\mathbf{g}}(\tau) = \mathbf{U}_p\mathbf{c}(\tau) = \mathbf{U}_p(\mathbf{P}^{\mathrm{T}}\mathbf{U}_p)^{-1}\mathbf{P}^{\mathrm{T}}\mathbf{g}(\tau). \tag{3.1}$$

GDEIM, as DEIM, selects the points such as to minimize the reduction approximation error. Instead of searching for the entry with the largest magnitude in a vector as DEIM does, GDEIM employs a global search for the maximum on a "bunch" of singular vectors. This is shown in Alg.2.

Algorithm 1: DEIM

procedure DEIM (Input: U_p)
  $v = U_p(:,1)$
  $P = [\,]$;$\mathbf{p} = [\,]$
  for $l = 1 : p$ **do**
    $\rho_l = \max|v|$
    $v = U_p(:,l)$
    $P = [P\ e_{\rho_l}]$, $\mathbf{p} = \begin{bmatrix}\mathbf{p}\\ \rho_l\end{bmatrix}$
    if $l > 1$ **then**
      $\widetilde{v} = U_p(:,1:l-1)(U_p(\mathbf{p},1:l-1))^{-1}v(\mathbf{p})$
      $v = v - \widetilde{v}$
    end if
  end for
end procedure (Output: $\mathbf{p}$)

Algorithm 2: GDEIM

procedure GDEIM (Input: U_p)
  $V = [\,]$
  $P = [\,]$;$\mathbf{p} = [\,]$
  for $l = 1 : p$ **do**
    $[\rho_l, i] = \max|U_p|$
    $V(:,l) = U_p(:,i)$
    $P = [P\ e_{\rho_l}]$, $\mathbf{p} = \begin{bmatrix}\mathbf{p}\\ \rho_l\end{bmatrix}$
    if $l < p$ **then**
      remove ith column from U_p
      $U_p = U_p - V(V(\mathbf{p},:))^{-1}U_p(\mathbf{p},:)$
    end if
  end for
end procedure (Output: $\mathbf{p}$)

DEIM has been proven to be well-defined because of the non-singularity of the interpolated basis. For GDEIM is same, due to linearly independence of the processed singular vectors with respect to remaining ones.
The DEIM reduction error is provided by the following Lemma.

LEMMA 3.1. *Let $\boldsymbol{g} \in \mathbb{R}^n$ be an arbitrary vector. Let $\{\boldsymbol{u}_l\}_{l=1}^p \subset \mathbb{R}^n$ be a given orthonormal set of vectors. Let the DEIM approximation of order $p \leq n$ for n in the space spanned by $\{\boldsymbol{u}_l\}_{l=1}^p \subset \mathbb{R}^n$ be*

$$\widetilde{\boldsymbol{g}} = \boldsymbol{U}_p(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U}_p)^{-1}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{g},$$

where $\boldsymbol{U}_p = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p] \in \mathbb{R}^{n\times p}$ and $\boldsymbol{P} = [\boldsymbol{e}_{\rho_1}, \ldots, \boldsymbol{e}_{\rho_p}] \in \mathbb{R}^{n\times p}$, with $\{\rho_1, \ldots, \rho_p\}$ being the output of the method. An L^2 error bound for $\widetilde{\boldsymbol{g}}$ is then given by

$$\|\boldsymbol{g} - \widetilde{\boldsymbol{g}}\|_2 \leq \|(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U}_p)^{-1}\|_2 \cdot \|(\boldsymbol{I}_p - \boldsymbol{U}_p\boldsymbol{U}_p^{\mathrm{T}})\boldsymbol{g}\|_2, \tag{3.2}$$

with $\boldsymbol{I}_p$ identity matrix of dimension p [5].
Inequality (3.2) in Lemma 3.1 is applicable for GDEIM as well. Moreover, since GDEIM does a global search for the maximum value which minimizes the approximation error, the constant $\|(\mathbf{P}^{\mathrm{T}}\mathbf{U}_p)^{-1}\|_2$ from GDIEM might be lower than that from DEIM. We experience this in many cases, mostly when the number of retained points is small. However, the constant is a pessimistic bound for the error, and the true approximation error approaches smaller values. Motivated by this, we can use the constant to check which of the two method is the best in terms of accuracy. We create a metric, cheap to evaluate, to establish this. Because it is a pessimistic bound,

the metric is a heuristic one, but after many experiments on analytic functions we find out that it is reliable in most of the cases.

Consider the following

$$\begin{cases} \| \mathbf{g} - \widetilde{\mathbf{g}}_G \|_2 \leq \| (\mathbf{P}_G^{\mathrm{T}} \mathbf{U}_G)^{-1} \|_2 \cdot \| (\mathbf{I} - \mathbf{U}_G \mathbf{U}_G^{\mathrm{T}}) \mathbf{g} \|_2, \\ \| \mathbf{g} - \widetilde{\mathbf{g}}_D \|_2 \leq \| (\mathbf{P}_D^{\mathrm{T}} \mathbf{U}_D)^{-1} \|_2 \cdot \| (\mathbf{I} - \mathbf{U}_D \mathbf{U}_D^{\mathrm{T}}) \mathbf{g} \|_2, \\ \mathbf{U}_G = \mathbf{U}_D \cdot \mathbf{R}, \end{cases}$$

with $\widetilde{\mathbf{g}}_G, \widetilde{\mathbf{g}}_D$ the approximate solutions of $\mathbf{g}$ by GDEIM and DEIM, respectively, and $\mathbf{R}$ is a rotational matrix, such that $\mathbf{R}^{-1} = \mathbf{R}^{\mathrm{T}}$. Thus,

$$\begin{aligned} \| (\mathbf{P}_G^{\mathrm{T}} \mathbf{U}_G)^{-1} \|_2 \cdot \| (\mathbf{I} - \mathbf{U}_G \mathbf{U}_G^{\mathrm{T}}) \mathbf{g} \|_2 &= \| (\mathbf{P}_G^{\mathrm{T}} \mathbf{U}_G)^{-1} \|_2 \cdot \| (\mathbf{I} - \mathbf{U}_D \mathbf{R} (\mathbf{U}_D \mathbf{R})^{\mathrm{T}}) \mathbf{g} \|_2 \\ &= \| (\mathbf{P}_G^{\mathrm{T}} \mathbf{U}_G)^{-1} \|_2 \cdot \| (\mathbf{I} - \mathbf{U}_D \mathbf{U}_D^{\mathrm{T}}) \mathbf{g} \|_2, \end{aligned}$$

which implies

$$\frac{\| \mathbf{g} - \widetilde{\mathbf{g}}_G \|_2}{\| \mathbf{g} - \widetilde{\mathbf{g}}_D \|_2} \leq \frac{\| (\mathbf{P}_G^{\mathrm{T}} \mathbf{U}_G)^{-1} \|_2}{\| (\mathbf{P}_D^{\mathrm{T}} \mathbf{U}_D)^{-1} \|_2}, \tag{3.3}$$

where $\| \mathbf{g} - \widetilde{\mathbf{g}}_G \|_2$, $\| \mathbf{g} - \widetilde{\mathbf{g}}_D \|_2$ are the L^2-norm errors between the original and the reduced model obtained with method GDEIM and DEIM, respectively.
By evaluating the right-hand side of (3.3) we can hopefully predict which method is the best, both when a small number of retained points is required with an acceptable ROM accuracy level, and when the specification is the error tolerance and one of the two methods reaches it with a smaller number of points.
Specifically, when

$$\frac{\| (\mathbf{P}_G^{\mathrm{T}} \mathbf{U}_G)^{-1} \|_2}{\| (\mathbf{P}_D^{\mathrm{T}} \mathbf{U}_D)^{-1} \|_2} - 1 < 0, \tag{3.4}$$

then we say that GDEIM is better than DEIM. The estimator's behavior is an upper bound function for the ratio of the true approximation errors

$$\frac{\| \mathbf{g} - \widetilde{\mathbf{g}}_G \|_2}{\| \mathbf{g} - \widetilde{\mathbf{g}}_D \|_2} - 1. \tag{3.5}$$

From simulating several functions, we find out that for a relatively small number of iterations, the estimator (3.4) predicts correctly (3.5) for 70% of the experiments, independently of the singular values decay of the model, with the L^1-, L^2- and L^∞-norm.

4. (G)DEIM for non-componentwise nonlinear functions. Assume $\mathbf{g} = [g(\mathbf{y}(t)), \ldots, g(\mathbf{y}(t))]^{\mathrm{T}} = [g(y_1(t), y_2(t)), \ldots, g(y_1(t), y_2(t))]^{\mathrm{T}}$ in (2.2). Assume also that we are solving for the ROM solution $\widetilde{\mathbf{y}}(t)$ at time t, i.e., we have already performed GDEIM (or DEIM) and obtain $\mathbf{p} = [\rho_1, \ldots, \rho_p]$, the vector of indices corresponding to the rows of $\mathbf{g}$ to evaluate. Assume without loss of generality that $p = 1$, i.e., we need to evaluate the nonlinear function at the first index of $\mathbf{g}$ and to retrieve $m = 2$ original model's components only, say, $y_1(t)$ and $y_2(t)$. This can be done by defining a *square incidence matrix* $\mathbf{M} \in \mathbb{R}^{n \times n}$ which selects the m components needed to evaluate the p

nonlinearities selected by the MOR method. In the case above, the incidence matrix will be of the form

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \cdots & & \ddots & \vdots \\ 0 & \cdots & & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}$$

with $\operatorname{rank}(M) = m$, such that the oblique projection for the nonlinear function in (2.4) writes as

$$\mathbf{V}_k^{\mathrm{T}} \mathbf{U}_p (\mathbf{P}^{\mathrm{T}} \mathbf{U}_p)^{-1} (\mathbf{P}^{\mathrm{T}} \widetilde{\mathbf{g}}((\mathbf{M}\mathbf{V}_k)\widetilde{\mathbf{y}}(t))).$$

Note that we need neither to perform the multiplication by $\mathbf{P}^{\mathrm{T}}$ nor by $\mathbf{M}$: first we selects the rows of $\mathbf{V}_k$ with $\mathbf{M}$ corresponding to the needed components of the original model's solution $\mathbf{y}(t)$ to retrieve, then $\mathbf{g}$ is evaluated at the selected indices through $\mathbf{P}$ by (G)DEIM. For sparse models in circuit simulation we can say that usually $m \approx p$, with $m, p \ll n$, and the original dimension is not involved anymore in the computation. As done for DEIM, we define the vector of indices of the original model's components to retrieve as $\mathbf{m} = [\phi_1, \ldots, \phi_m] \in \mathbb{R}^m$. In case of componentwise evaluation it results $\mathbf{p} = \mathbf{m}$, i.e., the ρ_ith component of the vector-valued function $\mathbf{g}$ to be evaluated requires only the ϕ_ith component of the vector-solution $\mathbf{y}$ of the original model. Thus our formulation extends the DEIM's one, and it can be applied to (2.3) as well.

5. Numerical experiments. In this section we show results from the comparison of GDEIM and DEIM in terms of accuracy. We only show results from simulations on two nonlinear, parametric analytic functions, by using the estimator in (3.4) for L^2- and L^∞-norms and both when the accuracy is evaluated at one parameter value and for a bunch of parameter values.

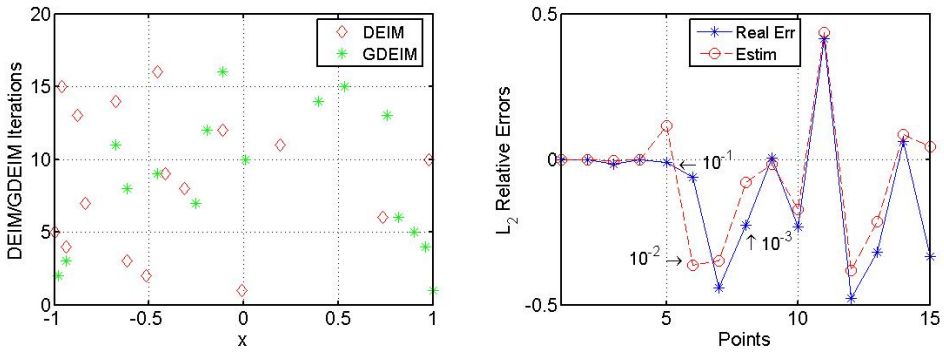
In each experiments, the original function is discretized in the one-dimensional spatial x and parameter τ domains with $n = 100$ and $q = 100$ points, respectively. For the first example, we have that

$$f_1(x, \tau) = e^{-\tau^2} + \cos(x\tau),$$

with $x \in [-1, 1]$, $\tau \in [10, 20]$. The distribution of the points (Fig. 5.1(a)) is quite different between the two methods. In Fig. 5.1(b) one can note that the relative real error of the approximation (3.5) has very similar behavior to the estimator (3.4). In particular, the plot tells that for six retained points (GDEIM/DEIM dimension), when the order of the error is 10^{-2}, GDEIM is more accurate than DEIM of about 45%, and of 25% for eight retained points. For the second example, given $g(x, \tau) = (1.3 + \tau)e^{x\tau - 1}$, the function is

$$f_2(x, \tau) = \begin{cases} g(x, \tau) & \text{if } x \in [-1, 0), \\ -g(x, \tau) & \text{if } x \in [0, 1], \end{cases}$$

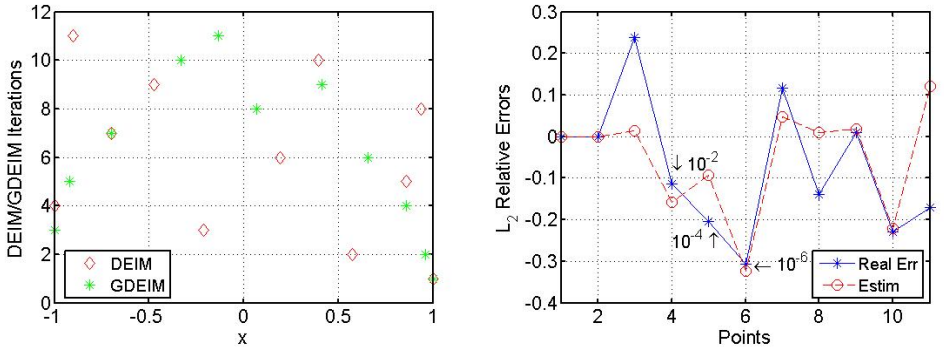
with $x \in [-1, 1]$, $\tau \in [1, \pi]$. For this example too, with different selected points (see Fig. 5.2(a)) there are still some difference between GDEIM and DEIM in accuracy (Fig. 5.2(b)). Even this time the estimator predicts correctly the real error, except at



(a) Distribution of the spatial points with the total number of iterations.

(b) Estimator (3.4) (red) and true error (3.5) (blue).

Fig. 5.1: Graphs for $f_1(x, \tau)$.



(a) Distribution of the spatial points with the total number of iterations.

(b) Estimator (3.4) (red) and true error (3.5) (blue).

Fig. 5.2: Graphs for $f_2(x, \tau)$.

the 11th point. In Table 1 we show results from the first example. We list the value of the L^2- and L^∞-norm of the real error for both GDEIM and DEIM, evaluated for one parameter value (τ_1) and for nine values ($\tau_{1:9}$), as the number of points p increases.

Now, we show simulation results from simulating the diode chain [10], whose model is represented by (2.2). From this experiment, GDEIM and DEIM retained the same points. We confirm the validity of the extended (G)DEIM to nonlinear function evaluated non-componentwise. In the following, subscript denotes the index of the component in the vector-solution and superscript denotes the discretized time in correspondence of which the solution is computed.
The nonlinear, vector-valued function is $\mathbf{g}(\mathbf{y}(t)) = [g_1(\mathbf{y}(t)), \ldots, g_{n-1}(\mathbf{y}(t)), 0]^{\mathrm{T}} \in \mathbb{R}^n \to \mathbb{R}^n$, with

$$g_i(\mathbf{y}(t)) = c_1(e^{(y_i(t)-y_{i+1}(t))/c_2} - 1),$$

where c_1, c_2 are two constants, for $i = 1, \ldots, n-1$.

p	L^2 (τ_1)		L^2 ($\tau_{1:9}$)		L^∞ (τ_1)		L^∞ ($\tau_{1:9}$)	
	DEIM	GDEIM	DEIM	GDEIM	DEIM	GDEIM	DEIM	GDEIM
6	1.01	0.94	1.01	0.95	2.16	1.91	2.20	1.94
7	0.16	0.09	0.15	0.09	0.34	0.17	0.33	0.16
8	0.04	0.03	0.04	0.03	0.08	0.07	0.08	0.07

Table 5.1: Results from $f_1(x, \tau)$: real error for p retained points, for the two norms, for a specific (τ_1) and for nine ($\tau_{1:9}$) parameters.

p	L^2 (τ_1)		L^2 ($\tau_{1:9}$)		L^∞ (τ_1)		L^∞ ($\tau_{1:9}$)	
	DEIM	GDEIM	DEIM	GDEIM	DEIM	GDEIM	DEIM	GDEIM
5	$4.60e^{-3}$	$3.60e^{-3}$	$4.60e^{-3}$	$3.70e^{-3}$	$6.90e^{-3}$	$5.90e^{-3}$	$7.10e^{-3}$	$6.00e^{-3}$
6	$2.22e^{-4}$	$1.54e^{-4}$	$2.24e^{-4}$	$1.55e^{-4}$	$4.36e^{-4}$	$2.14e^{-4}$	$4.37e^{-4}$	$2.15e^{-4}$
7	$5.54e^{-6}$	$6.19e^{-6}$	$5.45e^{-6}$	$6.09e^{-6}$	$1.16e^{-5}$	$9.92e^{-6}$	$1.14e^{-6}$	$9.68e^{-6}$

Table 5.2: Results from $f_2(x, \tau)$: real error for p retained points, for the two norms, for a specific (τ_1) and for nine ($\tau_{1:9}$) parameters.

In the area of circuit simulation, a common analysis consists of a time-domain simulation of the model. Usually, numerical integration methods are used to compute the solution of (2.2) for several discrete time points t^i, through the Newton method. We choose $[t^0, t^{\text{stop}}] = [0, 6]$ as interval of simulation, the input $u_{in}(t) = \sin(10t)$ and the original model's dimension $n = 10^5$. Then, we compare the accuracy and simulation time between a normal simulation and the one with the MOR technique. For the latter, we adopt the following strategy: we compute the full model's solutions until t^{TRAIN}, i.e., during this training period we collect n_s solutions in the snapshot matrix $\mathbf{Y} = [\mathbf{y}(t^0), \ldots, \mathbf{y}(t^{\text{TRAIN}})] \in \mathbb{R}^{n \times n_s}$; we then execute POD-GDEIM, to obtain the projection matrices $\mathbf{V}_k$ (for the POD) and $\mathbf{U}_p$ (for the GDEIM), and for the rest of the simulation we solve for the ROM. It is worth noticing that the way we apply the MOR procedure is different from optimization analysis: while for the latter one builds a ROM in an off-line phase (one-time cost), that can be reused for further simulations by changing inputs and/or parameters, here instead we construct the ROM during the simulation itself that we want to speed up (on-line phase). This gives rise to a limiting factor for the performance of the simulation, and to challenges for MOR in circuit simulation.

A good accuracy of the ROM could be obtained by using the first $k = 20$ singular vectors of the basis (obtained through SVD or through an iterative method [6]); GDEIM generates the vector $\mathbf{p} = [1, 2, \ldots, 20]^{\mathrm{T}} \in \mathbb{R}^p$ with $p = 20$ (nonlinear rows of $\mathbf{g}$ to evaluate). By inspecting the nonlinearities involved in these equations, one can extract the components of the original model's vector-solution which are involved. The vector with indices corresponding to the needed components to be retrieved is $\mathbf{m} = [1, 2, \ldots, 21]^{\mathrm{T}} \in \mathbb{R}^m$ with $m = 21$.
Fig. 5.3 shows the plots of the dynamic's components $x_2(t)$ (left) and of the L^∞ relative error

$$\text{err}(t^{\text{i}}) = \frac{\| \mathbf{y}(t^{\text{i}}) - \mathbf{V}_k \widetilde{\mathbf{y}}(t^{\text{i}}) \|_\infty}{\| \mathbf{y}(t^{\text{i}}) \|_\infty},$$

between the full model's solution and the approximation, for each discrete time point

just after the POD training period. The adopted strategy of POD-GDEIM for circuit

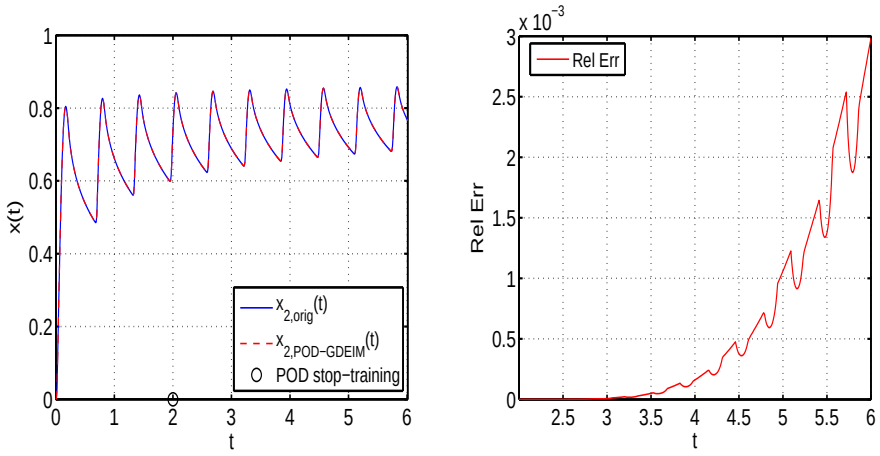


Fig. 5.3: Dynamics $x_2(t)$ (left) and error of the reduction (right).

simulation requires an on-line phase procedure to compute the ROM, after collecting the full model's solutions. As a consequence, the speedup factor (comparison between the original model's simulation time and the ROM's one) is obviously limited by the time spent for the training period. We obtain a speedup factor of 2.5 with respect to the normal simulation, which is almost inversely proportional to the ratio of the training period by the total interval of simulation, but we have a speedup of approximatively 600 after the training, i.e., the ROM's solutions are computed much faster.

6. Conclusions and future perspectives. In this work we have shown that GDEIM may be a valuable alternative to DEIM, especially when one has strict constraints on either the number of points where to evaluate the nonlinearities or when a modest level of accuracy is acceptable. By using a heuristic, cheap estimator, one can often choose with reliability between GDEIM and DEIM to speed up simulations of further analysis. This is beneficial when the dimension of the ROM is required to be as small as possible, still achieving the desired accuracy of simulations.

Besides, we have provided a mathematical formulation of (G)DEIM to general, nonlinear functions evaluated non-componentwise. We applied our extended formulation to the simulation of a circuit model (DAEs), obtaining a good total speedup factor and a speedup of two order of magnitude after the training period. Thus, the MOR technique could be applied to general nonlinear, parameterized functions.

To conclude, we would like to highlight some of the challenges we identify for circuit simulation:

1) The error dynamic between the full model and ROM in Fig. 5.3(right) increases with time. It suggests that an automated procedure is needed to evaluate, during the simulation, the accuracy of the ROM extracted through the training period. It might be necessary to update the projection matrix by collecting new snapshots. An a posteriori, cheap error estimator is needed in this direction.
2) Related to the previous problem, it is clear that repeating the training period during the simulation and the computation of the projection matrix are limiting factors for the total speedup. We can think of hierarchical partitioning

of the matrices and exploiting parallelism for the computation of new snapshots/projection matrix.

REFERENCES

[1] P. Astrid, *Reduction of Process Simulation Models: a Proper Orthogonal Decomposition Approach*, PhD thesis, Technische Universiteit Eindhoven, 2004.

[2] M. Barrault, M. Y., N. C. Nguyen, and A. T. Patera, *An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations*, Comptes Rendus Mathematique, 339 (2004), pp. 667–672.

[3] T. Bui-Thanh, M. Damodaran, and K. Willcox, *Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition*, AIAA J., 42 (2004), pp. 1505–1516.

[4] S. Chaturantabut, *Nonlinear Model Reduction via Discrete Empirical Interpolation*, PhD thesis, Rice University, Houston, Texas, 2011.

[5] S. Chaturantabut and D. C. Sorensen, *A state space error estimate for POD-DEIM nonlinear model reduction*, SIAM J. Numer. Anal., 50 (2012), pp. 46–63.

[6] M. E. Hochstenbach, *A Jacobi–Davidson type SVD method*, SIAM J. Sci. Comput., 23 (2001), pp. 606–628.

[7] K. Kunisch and S. Volkwein, *Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics*, SIAM J. Numer. Anal., 40 (2002), pp. 492–515.

[8] M. Loéve, *Probability Theory*, Van Nostrand, New York, 1955.

[9] W. H. A. Schilders, H. A. van der Vorst, and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*, vol. 13, Springer, 2008.

[10] A. Verhoeven, E. J. W. ter Maten, M. Striebel, and R. M. M. Mattheij, *Model order reduction for nonlinear IC models*, Sys. Model. Opt., (2007), pp. 476–491.

[11] T. Voss, A. Verhoeven, T. Bechtold, and E. J. W. ter Maten, *Model Order Reduction for Nonlinear Differential Algebraic Equations in Circuit Simulation*, in Progress in Ind. Math. at ECMI 2006, L. L. Bonilla, M. Moscoso, G. Platero, and J. M. Vega, eds., vol. 12, Springer Berlin Heidelberg, 2008, pp. 518–523.

[12] Y. Zhang, L. Feng, S. Li, and P. Benner, *Accelerating PDE constrained optimization by the reduced basis method: application to batch chromatography*, Int. J. Numer. Meth. Eng., 102 (2015), pp. 1111–1135.