Gonzalez-Rodriguez, P., Moscoso, M., & Kindelan, M. (2015). Laurent expansion of the inverse of perturbed, singular matrices. *Journal of Computational Physics*, 299, pp. 307–319.

DOI:10.1016/j.jcp.2015.07.006

Laurent expansion of the inverse of perturbed, singular matrices

Pedro Gonzalez-Rodriguez *, Miguel Moscoso, Manuel Kindelan

A B S T R A C T

In this paper we describe a numerical algorithm to compute the Laurent expansion of the inverse of singularly perturbed matrices. The algorithm is based on the resolvent formalism used in complex analysis to study the spectrum of matrices. The input of the algorithm are the matrix coefficients of the power series expansion of the perturbed matrix. The matrix coefficients of the Laurent expansion of the inverse are computed using recursive analytical formulae. We show that the computational complexity of the proposed algorithm grows algebraically with the size of the matrix, but exponentially with the order of the singularity. We apply this algorithm to several matrices that arise in applications. We make special emphasis to interpolation problems with radial basis functions.

Keywords: Laurent series, Inverse, Radial basis functions, Interpolation.

1. Introduction

In many situations one has to compute the solution of perturbed equations $A(\delta)\vec{x} = \vec{b}$, where the unperturbed matrix $A(0)$ is singular. The perturbed matrix $A(\delta)$ may or may not be singular for $\delta \neq 0$. Some important applications where this problem arises are finite difference methods for the solution of singular perturbation problems [9,10], differential equations with singular constant coefficients [7], least square problems [26], asymptotic linear programming [21,22], and Markov chains [19,1,8].

In this paper, we consider the problem of inverting analytically perturbed matrices

$$A(\delta) = A_0 + \sum_{k=1}^{\infty} \delta^k A_k \,, \tag{1}$$

where $A_0 = A(0)$ is singular. If the inverse $A^{-1}(\delta)$ exists in some punctured disc around $\delta = 0$, we compute its Laurent series expansion in the form

$$A^{-1}(\delta) = \sum_{k=-s}^{\infty} \delta^k H_k \,, \tag{2}$$

where $H_{-s} \neq 0$, and s is the order of the singularity. If the inverse $A^{-1}(\delta)$ does not exist, we compute the Laurent series of the Drazin inverse [5]. Once the coefficients H_k of the Laurent series have been obtained, the inverse can be computed accurately for any value of δ, no matter how small. In fact, Avrachenkov and Lasserre [4] proposed an iterative algorithm for inverting perturbed operators using analytic perturbation theory for linear operators (see the books [20,6] for details).

* Corresponding author at: Universidad Carlos III de Madrid, Avenida de la Universidad 30, 28911 Leganés, Spain. Fax: +34 916249129.
E-mail addresses: pgonzale@ing.uc3m.es (P. Gonzalez-Rodriguez), moscoso@math.uc3m.es (M. Moscoso), kinde@ing.uc3m.es (M. Kindelan).

This work has a twofold aim. On the one hand, we describe in detail a numerical algorithm to implement this procedure. The order s of the singularity is obtained as a by-product of the algorithm. We present a detailed analysis of the computational complexity and show that it grows exponentially with s. On the other hand, we discuss what we think is a novel and very interesting application. We apply this algorithm to interpolation problems with Radial Basis Functions (RBF) in the limit when they become flat as the perturbation parameter δ approaches zero. In fact, it is well known [11,23,25,27] that in this limit RBF interpolation converges to polynomial interpolation. With the proposed approach, i.e., by computing the Laurent expansion of the inverse of the RBF matrix, we can explicitly derive the exact form of the interpolating polynomial obtained in the limit of infinitely flat basis functions. We have already applied the algorithm described here to the solution of partial differential equations with the RBF method [18].

The RBF matrix is a real symmetric matrix and, therefore, we are particularly interested in this type of matrices. In fact, the computation of the widely used Moore–Penrose pseudoinverse involves the inverse of a symmetric matrix. In addition to RBF interpolation problems, we will also show examples from different applications where the matrices to be inverted are not real symmetric and/or where the inverse $A^{-1}(\delta)$ does not exist. The algorithm described here can be applied to any square matrix with the only restriction that the zero-eigenvalues are semi-simple. This means that their algebraic and geometric multiplicities agree, which is the case for symmetric matrices.

The paper is organized as follows. Section 2 contains the main mathematical concepts and definitions used throughout the paper. In Section 3 we consider the case of regularly perturbed matrices. In Section 4 we consider the case of singularly perturbed matrices. Section 5 describes the numerical algorithm in detail and analyzes the computational complexity. Section 6 presents results obtained with matrices that arise in RBF interpolation and in other applications. Section 7 contains the main conclusions of the paper.

2. Basic concepts and definitions

Here, we introduce some mathematical concepts and results that will be used throughout the paper. In particular, we briefly review the resolvent formalism which applies concepts from complex analysis to the study of the spectrum of operators. In this section, we consider the case of unperturbed operators that do not depend on any parameter. These operators can be invertible or not.

Any matrix A can be written as the spectral sum

$$A = \sum_{i=0}^{p} \lambda_i P_i + D_i \,, \tag{3}$$

where P_i and D_i are the eigenprojection and the eigennilpotent associated with the eigenvalue λ_i, respectively. They have the properties

$$P_i P_j = \begin{cases} P_j & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \,, \quad \text{and} \quad P_i D_j = \begin{cases} D_j & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \,. \tag{4}$$

For ease of presentation, and because our main application is RBF interpolation, we will consider real symmetric matrices in what follows. We will give the modifications for more general matrices when needed.

A symmetric matrix is diagonalizable and, therefore, the nilpotents D_i are zero. Hence, the spectral representation of an $N \times N$ symmetric matrix with eigenvalues $\lambda_1, \ldots, \lambda_p$ (not necessarily all simple) takes the form

$$A = \sum_{i=0}^{p} \lambda_i P_i \,. \tag{5}$$

Furthermore, if A is symmetric, the projections P_i onto the λ_i-eigenspaces E_i, $i = 0, 1, \ldots, p$, are orthogonal projections.

If all the eigenvalues are different from 0 and, therefore, A is invertible, the spectral decomposition of A^{-1} is given by

$$A^{-1} = \sum_{i=1}^{p} \frac{P_i}{\lambda_i} \,. \tag{6}$$

Throughout this paper we use the index $i = 0$ only for the eigenvalue $\lambda_0 = 0$. All other eigenvalues are numbered starting from $i = 1$.

We now define the matrix-valued function

$$R(\lambda) = (A - \lambda \mathbb{I})^{-1} \,, \tag{7}$$

called the resolvent of A. The resolvent is an analytic function of the complex variable λ in the complementary set of the spectrum $\sigma(A)$ of A. The eigenvalues λ_j of the matrix A are the poles of the resolvent. If zero is not an eigenvalue of A and, therefore A is invertible, $R(0)$ is the inverse of A. In this case, the spectral decomposition of the resolvent can be written as

$$R(\lambda) = \sum_{i=1}^{p} \frac{P_i}{\lambda_i - \lambda} \,. \tag{8}$$

It is easy to check (8) by using (4) and (5). Indeed,

$$(A-\lambda\mathbb{I})R(\lambda)=\left(\sum_{i=1}^{p}(\lambda_i-\lambda)P_i\right)\left(\sum_{j=1}^{p}\frac{P_j}{\lambda_j-\lambda}\right)=\sum_{i=1}^{p}P_i=\mathbb{I}. \tag{9}$$

Since the resolvent is an analytic function, we may use contour integration. If C_j is a positively-oriented contour enclosing the eigenvalue λ_j but not other eigenvalues of A, then

$$\frac{1}{2\pi i}\int_{C_j}R(\lambda)d\lambda=\frac{1}{2\pi i}\int_{C_j}\sum_{i=0}^{p}\frac{P_i}{\lambda_i-\lambda}d\lambda=\frac{1}{2\pi i}\int_{C_j}\frac{P_j}{\lambda_j-\lambda}d\lambda=-P_j\,. \tag{10}$$

Therefore, the contour integral of the resolvent around the eigenvalue λ_j of A gives the eigenprojection

$$P_j=-\frac{1}{2\pi i}\int_{C_j}R(\lambda)d\lambda\,, \tag{11}$$

for the eigenvalue λ_j.

Given any $N\times N$ complex matrix A of index k, $\mathbb{C}^N$ admits the direct decomposition $\mathbb{C}^N=\text{range}(A^k)\oplus\ker(A^k)$. The index of a matrix is the smallest integer k for which $\text{range}(A^{k+1})=\text{range}(A^k)$. The index is 0 if and only if A is nonsingular. If A is singular, the index of A is equal to the multiplicity of the zero eigenvalue as a root of the minimal polynomial, i.e. the size of the largest block with zero diagonal in its Jordan form.

The eigenprojection P_0 corresponding to the eigenvalue 0 is the idempotent matrix such that $\text{range}(P_0)=\ker(A^k)$ and $\ker(P_0)=\text{range}(A^k)$. In other words, P_0 is the projection onto $\ker(A^k)$ along $\text{range}(A^k)$. Thus, if A is symmetric, P_0 is an orthogonal projection (if A is symmetric, the index is 1 and $\text{range}(A)=\text{row}(A)$).

Also A and A^{-1} (if it exists) can be written in terms of the resolvent. It is straightforward to show that

$$A=-\frac{1}{2\pi i}\int_{C}\lambda R(\lambda)d\lambda\,,\quad\text{and}\quad A^{-1}=-\frac{1}{2\pi i}\int_{C}\frac{R(\lambda)}{\lambda}d\lambda\,, \tag{12}$$

where C is a positively-oriented contour enclosing all the eigenvalues of A.

If A is not invertible, $\lambda_0=0$ is an eigenvalue and, therefore, $\lambda=0$ is a pole of the resolvent. If $\lambda=0$ is a simple pole, then we can write

$$R(\lambda)=-\frac{P_0}{\lambda}+Q(\lambda)\,, \tag{13}$$

where $Q(\lambda)=\sum_{i=1}^{p}\frac{P_i}{\lambda_i-\lambda}$ is the analytic part of $R(\lambda)$ which admits the power expansion

$$Q(\lambda)=A^{\#}+\sum_{p=1}^{\infty}\lambda^p(A^{\#})^{p+1}\,, \tag{14}$$

for small λ. In (14), we have used the fact that $\left(\frac{\partial}{\partial\lambda}\right)^n Q(\lambda)=n!Q(\lambda)^{n+1}$ (see page 37 of [20]). Explicit formulas for P_0 and $A^{\#}\equiv Q(0)$ are

$$P_0=-\frac{1}{2\pi i}\int_{C_0}R(\lambda)d\lambda\quad\text{and} \tag{15}$$

$$A^{\#}=-\frac{1}{2\pi i}\int_{C_0}\frac{R(\lambda)}{\lambda}d\lambda\,, \tag{16}$$

where C_0 is a positively-oriented contour enclosing $\lambda_0=0$ but no other eigenvalues of A. Equation (16) is obtained from (13) using Cauchy's residue theorem. This equation will be essential throughout the paper. If A is symmetric or has index 1, $A^{\#}$ is the inverse of A restricted to the range of A. In this case, $A^{\#}$ is called the group inverse. For more general matrices, $A^{\#}$ is the inverse of A restricted to the range of A^k and is denoted Drazin inverse, A^D.

3. Regular perturbation case

In this section, we consider the case in which an operator depends on a parameter δ and admits a power series of the form

$$A(\delta)=A_0+\delta A_1+\delta^2A_2+\dots\,. \tag{17}$$

Furthermore, we consider the case in which the operator (17) is regularly perturbed. By a regular perturbation we mean that the dimension of the kernel of $A(\delta)$ with $\delta \neq 0$ is the same as the dimension of the kernel of $A(0)$. Note that because the operator depends on a parameter δ, so do the eigenvalues, projection operators, and the inverse $A^{\#}(\delta)$. In what follows, the goal is to compute the matrix coefficients of the power series of $A^{\#}(\delta)$.

As before, the resolvent of $A(\delta)$ is defined as

$$R(\lambda, \delta) = (A(\delta) - \lambda \mathbb{I})^{-1}. \tag{18}$$

Now, $R(\lambda, \delta)$ is an analytical function in the two variables λ and δ in all domains that do not contain an eigenvalue $\lambda(\delta)$ of the perturbed matrix $A(\delta)$. Therefore, we can write $R(\lambda, \delta)$ as a power series in δ with coefficients $R_k(\lambda)$ depending on λ. That is,

$$R(\lambda, \delta) = R(\lambda, 0) + \sum_{k=1}^{\infty} \delta^k R_k(\lambda). \tag{19}$$

Formula (19) is called the second Neumann series for the resolvent, and it is uniformly convergent for sufficiently small δ and $\lambda \in \mathbb{C} \backslash \sigma(A)$, where $\sigma(A)$ denotes the spectrum of A. The coefficients $R_k(\lambda)$ in (19) are given by (see Appendix A)

$$R_k(\lambda) = \sum_{p=1}^{k} (-1)^p \sum_{\nu_1+\nu_2+\cdots+\nu_p=k} R(\lambda) A_{\nu_1} R(\lambda) A_{\nu_2} \ldots A_{\nu_p} R(\lambda). \tag{20}$$

In (20), we have denoted $R(\lambda) = R(\lambda, 0)$.

Because the kernel of a regularly perturbed operator $A(\delta)$ does not change when δ changes from 0 to a value different than 0, we can consider a contour C_0 that only encloses the eigenvalue $\lambda_0 = 0$, but not other eigenvalues of $A(\delta)$. Therefore, (16) becomes

$$A^{\#}(\delta) = -\frac{1}{2\pi i} \int_{C_0} \frac{R(\lambda, \delta)}{\lambda} d\lambda \tag{21}$$

for all δ. Substitution of the power series (19) in (21), and integrating term by term we obtain the power series

$$A^{\#}(\delta) = A_0^{\#} + \sum_{n=1}^{\infty} \delta^n A_n^{\#}. \tag{22}$$

The coefficients $A_n^{\#}$ are given by

$$A_0^{\#} = A^{\#}(0) \quad \text{and} \quad A_n^{\#} = -\frac{1}{2\pi i} \int_{C_0} \frac{R_n(\lambda)}{\lambda} d\lambda, \tag{23}$$

with $R_n(\lambda)$ given in (20). Following [4], the matrices $A_n^{\#}$, $n = 1, 2, \ldots$, can be expressed as sums of matrix products as (see Appendix B)

$$A_n^{\#} = \sum_{p=1}^{n} (-1)^p \sum_{\substack{\nu_1+\cdots+\nu_p=n \\ \mu_1+\cdots+\mu_{p+1}=p+1}} S_{\mu_1} A_{\nu_1} S_{\mu_2} \ldots A_{\nu_p} S_{\mu_{p+1}}, \tag{24}$$

where $S_0 \equiv -P_0$, $S_k \equiv (A_0^{\#})^k$ for $k = 1, 2, \ldots$, $\nu_i > 0$ and $\mu_i \geq 0$. The main point here is that the matrix coefficients $A_n^{\#}$ can be computed in terms of P_0 and $A_0^{\#}$ only.

4. Singular perturbation case

Now, we consider a singularly perturbed matrix $A(\delta)$. Expansion (17) is a singular perturbation if there is a splitting of the zero eigenvalue of the unperturbed matrix A_0 when $\delta \neq 0$. This is the case of interest in many applications, such as RBF interpolation, because the matrices are such that A_0 is singular but $A(\delta)$ is not for $\delta \neq 0$.

In this section we show that, by using a reduction process, a singular perturbation problem can be transformed into a regular one [20]. The key is to construct the inverse of the operator restricted to the subspace determined by the *total* projection $P_{A0}(\delta)$ onto the *total* eigenspace associated to the set of eigenvalues of $A(\delta)$ that depart from zero at $\delta = 0$ ($P_{A0}(\delta)$ is a sum of eigenprojections). The group of the perturbed eigenvalues $\lambda_i(\delta)$ such that $\lambda_i(\delta) \to 0$ as $\delta \to 0$ is denoted as the 0-group.

We define the operator

$$A_p(\delta) = A(\delta) P_{A0}(\delta) = P_{A0}(\delta) A(\delta) = P_{A0}(\delta) A(\delta) P_{A0}(\delta), \tag{25}$$

which admits the Taylor series

$$A_p(\delta) = \sum_{k=1}^{\infty} \delta^k \tilde{A}_k \,. \tag{26}$$

Now, the problem of finding the inverse of $A(\delta)$ restricted to range$(\mathbb{I} - P_{A0}(\delta))$ is equivalent to the one of finding the inverse of $A_p(\delta)$. Note that $P_{A0}(\delta)$ commutes with $A(\delta)$. Hence, range$(P_{A0}(\delta))$ and range$(\mathbb{I} - P_{A0}(\delta))$ are invariant under $A(\delta)$ and, therefore, the vector space $\mathbb{C}^n$ can be decomposed into the direct sum of these subspaces, so $\mathbb{C}^n =$ range$(\mathbb{I} - P_{A0}(\delta)) \oplus$ range$(P_{A0}(\delta))$. Thus,

$$\begin{aligned} &A^{\#}(\delta) = (A(\delta)[\mathbb{I} - P_{A0}(\delta)] + A(\delta)P_{A0}(\delta))^{\#} = \\ &(A(\delta)[\mathbb{I} - P_{A0}(\delta)])^{\#} + (A(\delta)P_{A0}(\delta))^{\#} = A^{g}(\delta) + \frac{1}{\delta}(\frac{1}{\delta}A(\delta)P_{A0}(\delta))^{\#} \,, \end{aligned} \tag{27}$$

where $A^g(\delta)$ is the inverse of the matrix restricted to the complementary subspace of range(P_{A0}), and is given by (22)–(24). Hence, it remains to compute the inverse of $A(\delta)$ restricted to range$(P_{A0}(\delta))$. That is, the problem is reduced to one of finding the inverse of the operator

$$B(\delta) = \frac{1}{\delta}A(\delta)P_{A0}(\delta) = \frac{1}{\delta}P_{A0}(\delta)A(\delta) = \frac{1}{\delta}P_{A0}(\delta)A(\delta)P_{A0}(\delta) = \sum_{k=0}^{\infty} \delta^k \tilde{A}_{k+1} \,. \tag{28}$$

Let us study in more detail the behavior of the eigenvalues of $A(\delta)$ that split from zero at $\delta = 0$. In general, they constitute one or several branches of analytic functions of δ that depart from $\lambda_0 = 0$ at different speeds. Let r be the slowest rate at which the eigenvalues grow with delta. Then, this branch of eigenvalues grows as

$$\lambda(\delta) = \alpha\delta^r + o(\delta^r), \tag{29}$$

where r is a positive rational number [20]. The operator (28) is regularly perturbed if $r = 1$.

The process of reducing the problem for $A(\delta)$ to the problem for $B(\delta)$ is called the reduction process. It follows that the slowest growth rate of the eigenvalues of the new operator (28) is reduced to $r - 1$. For convenience, we rewrite the Taylor series for $B(\delta)$ as

$$B(\delta) = B_0 + \sum_{n=1}^{\infty} \delta^n B_n \,. \tag{30}$$

Thus, the unperturbed operator is now $B_0 = \tilde{A}_1 = P_{A0}(0)A_1P_{A0}(0)$ (see expansions (17) and (26)). Notice that the operator $B(\delta)$ can be calculated using the Cauchy integral formula

$$B(\delta) = \frac{1}{2\pi i\delta}\int_{C_0} \lambda R(\lambda, \delta)d\lambda \,, \tag{31}$$

where C_0 is a contour enclosing only the eigenvalues that depart from zero at $\delta = 0$. Using the second Neumann series for the resolvent (19) with coefficients given in (20), we get $B_0 = P_{A0}(0)A_1P_{A0}(0)$ and

$$B_n = -\sum_{p=1}^{n+1}(-1)^p \sum_{\substack{\nu_1+\cdots+\nu_p=n+1 \\ \mu_1+\cdots+\mu_{p+1}=p-1}} S_{\mu_1}A_{\nu_1}S_{\mu_2}\ldots A_{\nu_p}S_{\mu_{p+1}} \,, \tag{32}$$

with $S_0 \equiv -P_0$, $S_k \equiv (A_0^{\#})^k$ for $k = 1, 2, \ldots$, $\nu_i > 0$ and $\mu_i \geq 0$.

Once the Taylor series for $B(\delta)$ is known, we can apply to $B(\delta)$ the same algorithm as to $A(\delta)$ to find

$$B^{\#}(\delta) = B_0^{\#} + \sum_{n=1}^{\infty} \delta^n B_n^{\#} \tag{33}$$

using (24) with $S_0 \equiv -P_{B0}$, and $S_k \equiv (B_0^{\#})^k$ instead.

The reduction process finishes if $r = 1$. Otherwise, we can apply the reduction process to the operator $B(\delta)$. Thus, we compute the matrix coefficients C_n of the new matrix

$$C(\delta) = \frac{1}{\delta}B(\delta)P_{B0}(\delta) = C_0 + \sum_{n=1}^{\infty} \delta^n C_n \,, \tag{34}$$

where $P_{B0}(\delta)$ is the total projection corresponding to the 0-group of $B(\delta)$. To find the matrices C_n we use (32) with B_{ν_i} instead of A_{ν_i} and $S_k = (B_0^{\#})^k$. When the reduction process is applied a finite number of times $r = s$, we obtain a regularly

perturbed operator $Z(\delta)$ and the process comes to an end. When the process finishes, we get an expression for the Laurent series of A^{-1} in the form

$$A^{-1}(\delta) = \cdots + \frac{1}{\delta^2} C^{\#}(\delta) + \frac{1}{\delta} B^{\#}(\delta) + A^{\#}(\delta)\,. \tag{35}$$

The matrices H_k in the equation (2) can be expressed in terms of the $A^{\#}, B^{\#}, \ldots$. For example, for a singularity of order $s = 2$ we obtain

$$\begin{aligned} H_{-2} &= C_0^{\#} \\ H_{-1} &= B_0^{\#} + C_1^{\#} \\ H_0 \;\; &= A_0^{\#} + B_1^{\#} + C_2^{\#} \end{aligned} \tag{36}$$

Once H_k with $k = -s, -s+1, \ldots, 0$ are found, the regular terms $H_1, H_2, \ldots$ in (2) can be easily obtained using the recursive formula [4]

$$H_{m+1} = -\sum_{i=0}^{m+p} \left(\sum_{j=0}^{p} H_{-j}\, A_{i+j+1} \right) H_{m-i}\,, \qquad m = 0,\, 1,\, \ldots\,. \tag{37}$$

Notice that, given any matrix A, we can always calculate the Laurent series of the Moore–Penrose pseudoinverse $A^{\dagger}(\delta) = (A^T(\delta)A(\delta))^{\#} A^T(\delta)$ where $A^T(\delta)A(\delta)$ is symmetric. Furthermore, if the inverse $A^{-1}(\delta)$ exists, then $A^{-1}(\delta) = A^{\dagger}(\delta)$.

5. Algorithm

Here, we describe in detail the algorithm to compute the first M terms of the Laurent series expansion of the perturbed inverse (2), given the power series of a matrix (1). The matrix coefficients H_k in (2) are computed using (36) and (37).

If the order s of the singularity is known, we apply the reduction process s times (from $i = 1$ to s). Thus, we start with $i = 1$ and use matrices A_n to compute matrices $A_n^{\#}$ using (24), and matrices B_n using (32). For $i = 2$, we use matrices B_n to compute matrices $B_n^{\#}$ and matrices C_n. We repeat this process s times. Notice from equation (36) that we need to compute $A_n^{\#}$ up to $n = 0$, $B_n^{\#}$ up to $n = 1$, $C_n^{\#}$ up to $n = 2$. Thus, if we use i as an index for the different letters ($i = 1$ for A, $i = 2$ for B, and so on), then the maximum subindex that we have to use in (24) to compute $\text{letter}(i)_n^{\#}$ is $n = i - 1$. The maximum index that we have to use in equation (32) to compute $\text{letter}(i)_n$ is $n = 2s - i + 1$.

Iteration i of the reduction process is described below. It computes matrices $\text{letter}(i)_n^{\#}$ and $\text{letter}(i+1)_n$ given matrices $\text{letter}(i)_n$ following the next steps:

1. Compute the pseudo-inverse $\text{letter}(i)_0^{\#}$ of the matrix $\text{letter}(i)_0$.
2. Compute $S_0 = -P_0$, where P_0 is the projection onto the null space of $\text{letter}(i)_0$.
3. Compute $S_k = (\text{letter}(i)_0^{\#})^k$, $k = 2, 3, \ldots, 2s - i + 1$.
4. Compute $\text{letter}(i)^{\#}$:
   a. First loop: compute $\text{letter}(i)_n^{\#}$ from $n = 1$ to $i - 1$ using (24).
      i. Second loop: sum from $p = 1$ to n.
         A. Compute the ν's: the l_ν combinations of p natural numbers that sum up to n.
         B. Compute the μ's: the l_μ combinations of $p + 1$ non-negative integers that sum up to $p + 1$.
         C. Sum all the products $(-1)^p S_{\mu_1} \text{letter}(i)_{\nu_1} S_{\mu_2} \ldots \text{letter}(i)_{\nu_p} S_{\mu_{p+1}}$.
5. Compute $\text{letter}(i+1)$:
   a. Compute $\text{letter}(i+1)_0 = P_0\, \text{letter}(i)_1\, P_0$.
   b. First loop: compute $\text{letter}(i+1)_n$ from $n = 1$ to $2s + 1 - (i+1)$ using (32).
      i. Second loop: sum from $p = 1$ to $n + 1$.
         A. Compute the ν's: the l_ν combinations of p natural numbers that sum up to $n + 1$.
         B. Compute the μ's: the l_μ combinations of $p + 1$ non-negative integers that sum up to $p - 1$.
         C. Sum all the products $(-1)^p S_{\mu_1} \text{letter}(i)_{\nu_1} S_{\mu_2} \ldots \text{letter}(i)_{\nu_p} S_{\mu_{p+1}}$ and multiply by -1 from $p = 1$ to $n + 1$.

After the reduction process has been completed for $i = 1$ to s, steps 1–4 of the algorithm have to be carried out again with $i = s + 1$ in order to compute $\text{letter}(s+1)^{\#}$ which is needed to compute the Laurent series of the inverse (35). Then, the singular terms of the Laurent series of the inverse (H_i from $i = -s$ to $i = 0$) are computed using (36), and the regular terms (H_i from $i = 1$ to $i = M$) are computed using (37).

In many cases, the order of the singularity s in (2) is not known a priori and, therefore, it has to be determined during the iterative process. To this end, we start by assuming a first order singularity ($s = 1$) and compute the Laurent series of the inverse using the algorithm just described. Then, we increase s by 1 and compute the new Laurent series. For each value of s we compute $\| \sum_{i=-s}^{0} H_i\, A_{-i} - \mathbb{I} \|$. When this norm is smaller than a certain tolerance we stop the process. In this way, we determine both the Laurent series of the inverse and the order of the singularity.

Table 1
Number of matrix products and computational time.

s	N_p	N	Time (sec)	Symbolic (sec)
1	14	3	0.084	$3.1\cdot 10^{-4}$
2	598	6	0.05	0.11
3	18 694	8	0.165	1.39
4	544 254	13	5.05	14 507
5	15 287 758	15	148.5	
6	420 038 854	17	4619	
7	11 368 586 038			
8	304 362 660 958			
9	8 081 425 679 758			

Steps 1 and 2 in the algorithm require some attention, as the projections onto the null spaces may be sensitive to small numerical errors. To this end, we obtain a truncated singular value decomposition of letter$(i)_0 = U\bar{\Sigma}V^T$, where $\bar{\Sigma}$ is a diagonal matrix whose diagonal entries are the singular values larger than a certain threshold (we use threshold $= 10^{-10}$). Then, the pseudoinverse is given by letter$(i)_0^{\#} = V\bar{\Sigma}^{\dagger}U^T$. If the matrix is symmetric or has index 1, we compute the orthogonal projection P_0 onto the kernel of letter$(i)_0$ through the projection matrix $SS^{\dagger}$, where S is the matrix whose columns are the right singular vectors corresponding to the singular values smaller than the threshold. If the matrix has index $k > 1$, we compute the oblique projection P_0 onto the kernel of letter$(i)_0$ along the range(letter$(i)_0^k$) as follows. We define the matrices M and N whose columns are basis of the kernel of letter$(i)_0$ and the orthogonal complement of the image of letter$(i)_0$. Then, the projection $P_0 = M(N^TM)^{-1}N^T$.

In the above algorithm, the most computationally expensive step is 4aiC, as it involves a very large number of matrix products (see Section 5.1). In terms of memory requirements, storage of the μ's in step 5aiA–B is the most demanding part of the algorithm. The maximum size of the matrix where we store all the combinations of μ's to compute letter$(i+1)_n$ grows as $(4s-1)!/((2s-1)!(2s)!) \times (2s+1)$. For example, for $s = 7$ the size is $20\,058\,300 \times 15$, so we have to store 300 874 500 integer numbers.

5.1. Numerical complexity

The most time-consuming part of this algorithm is the computation of all the matrix products to calculate B. The number of products depends on the order of the singularity s as

$$N_p = \#\text{products to calculate } B = 2\sum_{n=1}^{2s-1}\sum_{p=1}^{n+1}\left(\frac{n!(2p-1)!}{(n-p+1)!((p-1)!)^3}\right)$$

This quantity grows approximately as $0.6759 * (28.8636)^s$. If we want to calculate the number of flops to calculate B we have to multiply N_p by $2N^3$ where $N \times N$ is the dimension of the square matrix A.

After calculating B you also have to calculate $B^{\#}$, C, etc. but this is fast compared with B. Table 1 shows the number of matrix products, N_p, to calculate B up to $s = 9$. The fourth column of the table shows the computational time for computing the Laurent series of the inverse for a perturbed matrix of size $N \times N$ on an Intel Core i7 950 @3.07 GHz processor using a *Matlab* code not optimized. Notice that the computational time grows exponentially with the order of the singularity, s, and algebraically with the size of the matrix, $O(N^3)$.

Our method can be considered as semi-analytical since the output of the method is a Laurent series for the inverse in terms of powers of the parameter δ, but the implementation just described involves a very large amount of numerical computations. Thus, the computational times shown in Table 1 should be compared with the computational times corresponding to the other alternative to compute the Laurent series of the inverse, namely, the use of a symbolic language. Column five of Table 1 shows the computational times using *Mathematica* as symbolic language (we use the command `Inverse[A]` where `A` is the power series of the matrix (17)). Notice that as N grows our algorithm becomes much faster than the symbolic language. In fact, for matrices of size greater than 13, *Mathematica* is not able to compute the inverse while, with our algorithm, we can compute the inverses of rather large matrices, as long as the order of the singularity is smaller than seven. In Ref. [3] a detailed analysis of the computational complexity involved in computing the Laurent series of the inverse using a symbolic language and a comparison with the computational complexity of the reduction process is carried out.

6. Numerical examples

6.1. RBF interpolation

As a first example, we compute the Laurent series of the inverse of the RBF interpolation matrix in 1D with three equispaced centers using multiquadrics as RBFs [18]. We use this example to illustrate in detail how the numerical algorithm

described in Section 5 can be used to derive the Laurent series of the inverse of a nearly singular matrix. For this simple example, the 1D RBF interpolation matrix with equispaced nodes is given by the symmetric matrix

$$A(\delta)=\begin{bmatrix} 1 & \sqrt{1+\delta} & \sqrt{1+4\delta} \\ \sqrt{1+\delta} & 1 & \sqrt{1+\delta} \\ \sqrt{1+4\delta} & \sqrt{1+\delta} & 1 \end{bmatrix}. \tag{38}$$

The parameter δ is the product of the inter-nodal distance and the shape parameter of the RBF multiquadrics that controls their flatness (for details, see, for example, [18]). The Taylor series of this matrix is given by

$$A(\delta)\approx \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} + \delta \begin{bmatrix} 0 & 1/2 & 2 \\ 1/2 & 0 & 1/2 \\ 2 & 1/2 & 0 \end{bmatrix} + \delta^2 \begin{bmatrix} 0 & -1/8 & -2 \\ -1/8 & 0 & -1/8 \\ -2 & -1/8 & 0 \end{bmatrix} + \delta^3 \begin{bmatrix} 0 & 1/16 & 4 \\ 1/16 & 0 & 1/16 \\ 4 & 1/16 & 0 \end{bmatrix} + \\ + \delta^4 \begin{bmatrix} 0 & -5/128 & -10 \\ -5/128 & 0 & -5/128 \\ -10 & -5/128 & 0 \end{bmatrix} + \delta^5 \begin{bmatrix} 0 & 7/256 & 28 \\ 7/256 & 0 & 7/256 \\ 28 & 7/256 & 0 \end{bmatrix} + \dots . \tag{39}$$

It is apparent that $A(\delta)$ becomes singular in the limit $\delta \to 0$. Moreover, the leading order terms of the eigenvalues are 3, -2δ, and $-2\delta^2/3$. Hence, the singularity is of order $s=2$, and we have to execute the algorithm described in Section 5 two times, first for $i=1$ and then for $i=2$. The matrix coefficients in (39) are the input data for the algorithm.

For $i=1$ (letter$(1)=A$), the algorithm yields matrices $A_j^{\#}$ and B_j. We start by computing matrices $A_0^{\#}$ and P_0 according to steps 1 and 2 of the algorithm. We obtain

$$A_0^{\#}=\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}, \quad \text{and} \quad P_0=\begin{bmatrix} 2/3 & -1/3 & -1/3 \\ -1/3 & 2/3 & -1/3 \\ -1/3 & -1/3 & 2/3 \end{bmatrix}. \tag{40}$$

In step 3 we compute the matrices $(A_0^{\#})^k$, $k=2,3,4$. Step 4 is not executed since the upper limit of the outer loop 4a is $i-1=-1$. Matrices $(A_0^{\#})^k$, $k=2,3$ are used in step 5 to compute matrix B_0 (step 5a) and matrices B_1, B_2 and B_3 (step 5b). Note that for $i=2$, $B=$ letter(i) and the upper limit of loop 5b is $2s+1-i=3$. Thus, we obtain

$$B(\delta)=\begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix} + \delta \begin{bmatrix} 8/9 & 2/9 & -10/9 \\ 2/9 & -4/9 & 2/9 \\ -10/9 & 2/9 & 8/9 \end{bmatrix} + \delta^2 \begin{bmatrix} -44/27 & -17/27 & 64/27 \\ -17/27 & 28/27 & -17/27 \\ 64/27 & -17/27 & -44/27 \end{bmatrix} + \\ + \delta^3 \begin{bmatrix} 104/27 & 7/4 & -166/27 \\ 7/4 & -68/27 & 7/4 \\ -166/27 & 7/4 & 104/27 \end{bmatrix} + \dots . \tag{41}$$

Once $B(\delta)$ is obtained, we repeat the same algorithm for $i=2$ ($B=$ letter(2)). We compute matrices $B_l^{\#}$ and C_l, $l=0,1,2,\dots$, in terms of matrices B_j in (41). Matrices $B_0^{\#}$ and P_0 are obtained according to steps 1 and 2 resulting in

$$B_0^{\#}=\begin{bmatrix} -1/4 & 0 & 1/4 \\ 0 & 0 & 0 \\ 1/4 & 0 & -1/4 \end{bmatrix}, \qquad P_0=\begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 1 & 0 \\ 1/2 & 0 & 1/2 \end{bmatrix}. \tag{42}$$

In step 3, we compute the matrices $(B_0^{\#})^k$ for $k=2$, and 3. These matrices are used in step 4 to compute

$$B_1^{\#}=\begin{bmatrix} -1/4 & 0 & 1/4 \\ 0 & 0 & 0 \\ 1/4 & 0 & -1/4 \end{bmatrix}.$$

Matrices $(B_0^{\#})^k$ with $k=2$ and 3 are also used in step 5 to compute C_0 in step 5a, and C_1 and C_2 in step 5b (the upper limit of loop 5b is $2s+1-i=2$). Thus, we obtain

$$C(\delta)=\begin{bmatrix} -1/9 & 2/9 & -1/9 \\ 2/9 & -4/9 & 2/9 \\ -1/9 & 2/9 & -1/9 \end{bmatrix} + \delta \begin{bmatrix} 10/27 & -17/27 & 10/27 \\ -17/27 & 28/27 & -17/27 \\ 10/27 & -17/27 & 10/27 \end{bmatrix} + \delta^2 \begin{bmatrix} -31/27 & 7/4 & -31/27 \\ 7/4 & -68/27 & 7/4 \\ -31/27 & 7/4 & -31/27 \end{bmatrix}.$$

Finally, we have to execute steps 1 and 4, with $i=s+1=3$, in order to compute matrices $C_0^{\#}$ (step 1) and $C_1^{\#}$, $C_2^{\#}$ (step 4). We obtain

$$C^{\#}(\delta)=\begin{bmatrix} -1/4 & 1/2 & -1/4 \\ 1/2 & -1 & 1/2 \\ -1/4 & 1/2 & -1/4 \end{bmatrix} + \delta \begin{bmatrix} -1/2 & 5/4 & -1/2 \\ 5/4 & -3 & 5/4 \\ -1/2 & 5/4 & -1/2 \end{bmatrix} + \delta^2 \begin{bmatrix} 5/36 & -25/144 & 5/36 \\ -25/144 & -1/9 & -25/144 \\ 5/36 & -25/144 & 5/36 \end{bmatrix} + \dots .$$

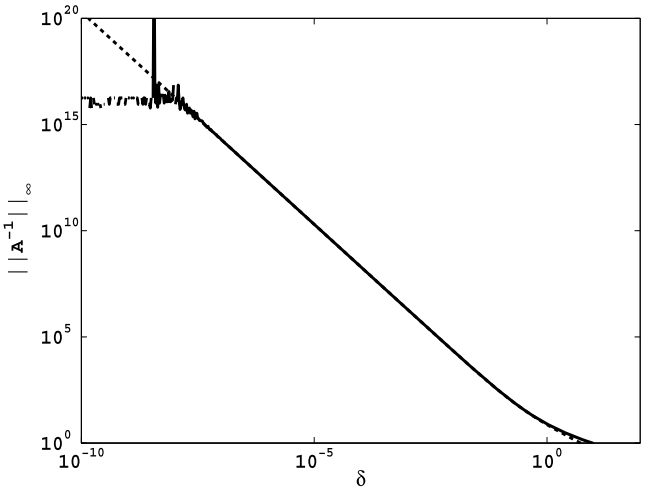

Fig. 1. Solid line: infinite norm of the inverse of matrix $A(\delta)$ (38) as a function of δ. Dashed line: infinite norm of the error of the inverse of A computed using its Laurent series (43).

Equation (36) is then used to compute the singular part of the Laurent series of the inverse whose matrix coefficients are given by

$$H_{-2} = \begin{bmatrix} -1/4 & 1/2 & -1/4 \\ 1/2 & -1 & 1/2 \\ -1/4 & 1/2 & -1/4 \end{bmatrix}, \quad H_{-1} = \begin{bmatrix} -3/4 & 5/4 & -1/4 \\ 5/4 & -3 & 5/4 \\ -1/4 & 5/4 & -3/4 \end{bmatrix}, \quad H_0 = \begin{bmatrix} 0 & -1/16 & 1/2 \\ -1/16 & 0 & -1/16 \\ 1/2 & -1/16 & 0 \end{bmatrix}.$$

Equation (37) is used to obtain the regular part of the Laurent series of the inverse. In this way, we find that

$$A^{-1}(\delta) = \frac{1}{\delta^2}\begin{bmatrix} -1/4 & 1/2 & -1/4 \\ 1/2 & -1 & 1/2 \\ -1/4 & 1/2 & -1/4 \end{bmatrix} + \frac{1}{\delta}\begin{bmatrix} -3/4 & 5/4 & -1/4 \\ 5/4 & -3 & 5/4 \\ -1/4 & 5/4 & -3/4 \end{bmatrix} + \begin{bmatrix} 0 & -1/16 & 1/2 \\ -1/16 & 0 & -1/16 \\ 1/2 & -1/16 & 0 \end{bmatrix} + \\ + \delta\begin{bmatrix} -1/4 & 21/32 & -3/4 \\ 21/32 & -1 & 21/32 \\ -3/4 & 21/32 & -1/4 \end{bmatrix} + \delta^2\begin{bmatrix} 3/4 & -485/256 & 7/4 \\ -485/256 & 3 & -485/256 \\ 7/4 & -485/256 & 3/4 \end{bmatrix} + \dots . \tag{43}$$

Fig. 1 shows with a solid line the infinite norm of the inverse of matrix $A(\delta)$ (38) computed numerically. Notice, that for δ smaller than $\approx 10^{-8}$ matrix A becomes ill-conditioned and round-off errors prevent the numerical computation of the inverse. However, the Laurent series can be used to accurately compute the inverse for any value of δ, no matter how small (dashed line).

The Laurent series of the inverse obtained in (43) can be used, for example, to find the RBF interpolation weights that result from the multiplication of $A^{-1}(\delta)$ by the values of the function at the nodes. For instance, consider the case in which the values of the function at the nodes are 0, 1, and 0. Hence, multiplying the Laurent series (43) by the vector $[0, 1, 0]^T$ results in the following Laurent expansion for the interpolation weights:

$$\boldsymbol{\alpha}(\delta) = \sum_{k=-2}^{\infty} \boldsymbol{\alpha}_k \delta^k = \frac{1}{\delta^2}\begin{bmatrix} 1/2 \\ -1 \\ 1/2 \end{bmatrix} + \frac{1}{\delta}\begin{bmatrix} 5/4 \\ -3 \\ 5/4 \end{bmatrix} + \begin{bmatrix} -1/16 \\ 0 \\ -1/16 \end{bmatrix} + \dots . \tag{44}$$

The value of the RBF interpolant at position x is obtained by multiplying the weights (44) by the corresponding multiquadrics basis functions [12], i.e.,

$$r(x;\delta) = \boldsymbol{\alpha}(\delta)^T \begin{bmatrix} \sqrt{1+\delta(x-(-1))^2} \\ \sqrt{1+\delta(x-0)^2} \\ \sqrt{1+\delta(x-1)^2} \end{bmatrix}. \tag{45}$$

Of particular interest is the limit of increasingly flat basis functions given by $\delta \to 0$. It is well known [11,23,25,27] that in this limit RBF interpolation converges to polynomial interpolation. We can use the Laurent series to find out the interpolating polynomial in this limit. Expanding (45) in powers of $\delta(x - x_i)^2$, $x_i = -1, 0, 1$, results in

$$r(x;\delta) = \boldsymbol{\alpha}(\delta)^T \begin{bmatrix} 1+\delta\,\dfrac{(x+1)^2}{2} - \delta^2\,\dfrac{(x+1)^4}{8} + \cdots \\ 1+\delta\,\dfrac{x^2}{2} - \delta^2\,\dfrac{x^4}{8} + \cdots \\ 1+\delta\,\dfrac{(x-1)^2}{2} - \delta^2\,\dfrac{(x-1)^4}{8} + \cdots \end{bmatrix}. \tag{46}$$

Collecting terms of order unity leads to

$$\begin{aligned} \lim_{\delta\to 0} r(x;\delta) = &\left\{ \boldsymbol{\alpha}_0^T \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \boldsymbol{\alpha}_{-1}^T \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix} + \boldsymbol{\alpha}_{-2}^T \begin{bmatrix} -1/8 \\ 0 \\ -1/8 \end{bmatrix} \right\} + \\ &+ x \left\{ \boldsymbol{\alpha}_{-1}^T \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + \boldsymbol{\alpha}_{-2}^T \begin{bmatrix} -1/2 \\ 0 \\ 1/2 \end{bmatrix} \right\} + x^2 \left\{ \boldsymbol{\alpha}_{-1}^T \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} + \boldsymbol{\alpha}_{-2}^T \begin{bmatrix} -3/4 \\ 0 \\ -3/4 \end{bmatrix} \right\} = \\ &= 1 - x^2 . \end{aligned} \tag{47}$$

If the values of the function at the nodes are arbitrary, $[f_{-1}, f_0, f_1]^T$, the resulting interpolation polynomial in the limit $\delta \to 0$ is

$$p(x) = f_{-1}\frac{x(x-1)}{2} + f_0\,(1+x)(1-x) + f_1\frac{x(x+1)}{2},$$

which coincides with the Lagrange interpolation polynomial.

The same procedure can be used to find the Laurent series of the inverse of the RBF interpolation matrix in 2D. For instance, using an equispaced five node stencil $[(0,0),(0,1),(1,0),(0,-1),(-1,0)]$ the RBF interpolation matrix is,

$$A(\delta) = \begin{bmatrix} 1 & \sqrt{\delta+1} & \sqrt{\delta+1} & \sqrt{\delta+1} & \sqrt{\delta+1} \\ \sqrt{\delta+1} & 1 & \sqrt{2\delta+1} & \sqrt{4\delta+1} & \sqrt{2\delta+1} \\ \sqrt{\delta+1} & \sqrt{2\delta+1} & 1 & \sqrt{2\delta+1} & \sqrt{4\delta+1} \\ \sqrt{\delta+1} & \sqrt{4\delta+1} & \sqrt{2\delta+1} & 1 & \sqrt{2\delta+1} \\ \sqrt{\delta+1} & \sqrt{2\delta+1} & \sqrt{4\delta+1} & \sqrt{2\delta+1} & 1 \end{bmatrix} \tag{48}$$

Using the same procedure described above results in the following first three terms of the Laurent series of the inverse

$$\begin{aligned} A^{-1}(\delta) \approx &\frac{1}{\delta^2} \begin{bmatrix} -\frac{4}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & -\frac{1}{3} & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} & -\frac{1}{3} \\ \frac{1}{3} & -\frac{1}{3} & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} & -\frac{1}{3} \end{bmatrix} + \frac{1}{\delta} \begin{bmatrix} -\frac{32}{9} & \frac{13}{18} & \frac{13}{18} & \frac{13}{18} & \frac{13}{18} \\ \frac{13}{18} & -\frac{41}{36} & \frac{11}{18} & -\frac{23}{36} & \frac{11}{18} \\ \frac{13}{18} & \frac{11}{18} & -\frac{41}{36} & \frac{11}{18} & -\frac{23}{36} \\ \frac{13}{18} & -\frac{23}{36} & \frac{11}{18} & -\frac{41}{36} & \frac{11}{18} \\ \frac{13}{18} & \frac{11}{18} & -\frac{23}{36} & \frac{11}{18} & -\frac{41}{36} \end{bmatrix} + \\ &+ \begin{bmatrix} \frac{2}{27} & -\frac{19}{216} & -\frac{19}{216} & -\frac{19}{216} & -\frac{19}{216} \\ -\frac{19}{216} & -\frac{25}{108} & \frac{31}{216} & \frac{29}{108} & \frac{31}{216} \\ -\frac{19}{216} & \frac{31}{216} & -\frac{25}{108} & \frac{31}{216} & \frac{29}{108} \\ -\frac{19}{216} & \frac{29}{108} & \frac{31}{216} & -\frac{25}{108} & \frac{31}{216} \\ -\frac{19}{216} & \frac{31}{216} & \frac{29}{108} & \frac{31}{216} & -\frac{25}{108} \end{bmatrix} + \cdots . \end{aligned} \tag{49}$$

In this case, the RBF interpolating polynomial in the limit $\delta \to 0$ is

$$p(x,y) = f_{0,0}\,(1-x^2-y^2) + f_{0,1}\,\frac{y(y+1)}{2} + f_{1,0}\,\frac{x(x+1)}{2} + f_{0,-1}\,\frac{y(y-1)}{2} + f_{-1,0}\,\frac{x(x-1)}{2},$$

where $f_{0,0}$, $f_{0,1}$, $f_{1,0}$, $f_{0,-1}$, and $f_{-1,0}$, are the values of the function at the corresponding nodes.

If we add one extra node and consider the six nodes stencil $[(0,0),(0,1),(1,0),(0,-1),(-1,0),(1,1)]$, the resulting RBF interpolation polynomial in the limit $\delta \to 0$ becomes

$$\begin{aligned} p(x,y) = &f_{0,0}\,(1+xy-x^2-y^2) + f_{0,1}\,\frac{y(y+1-2x)}{2} + f_{1,0}\,\frac{x(x+1-2y)}{2} + \\ &+ f_{0,-1}\,\frac{y(y-1)}{2} + f_{-1,0}\,\frac{x(x-1)}{2} + f_{1,1}\,xy . \end{aligned} \tag{50}$$

For instance, we can use the six node stencil to interpolate function

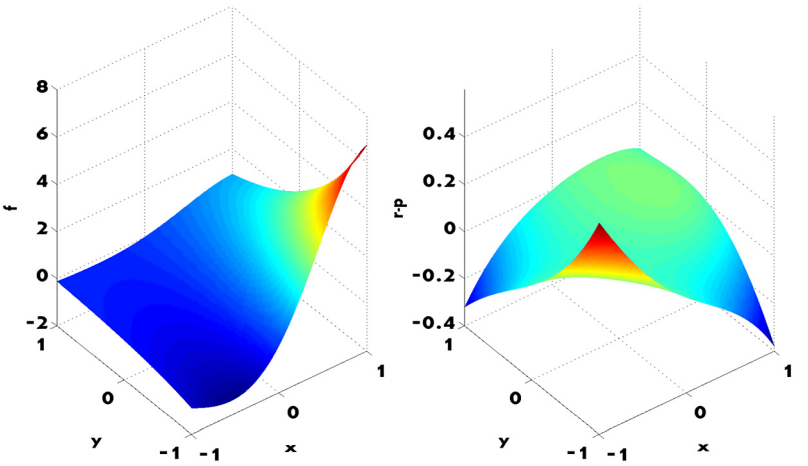

Fig. 2. Left: function (51). Right: difference between RBF interpolation for $\delta = 0.05$ using the six nodes stencil $[(0,0),(0,1),(1,0),(0,-1),(-1,0),(1,1)]$, and the RBF interpolation in the limit $\delta \to 0$ (Eq. (50)). (For interpretation of the colors in this figure, the reader is referred to the web version of this article.)

$$f(x,y) = e^{x-y}\,\sin(2x) \tag{51}$$

which is shown in the left side of Fig. 2. The weights of the RBF interpolation in the case $\delta = 0.05$ are $\boldsymbol{\alpha}(\delta)^T = \begin{bmatrix} 925.7 & -469.9 & -976.4 & 173.4 & -303.8 & 657.8 \end{bmatrix}$. We have also computed the interpolation using the interpolating polynomial in the limit $\delta \to 0$ (50). The right side of Fig. 2 shows the difference between RBF interpolation for $\delta = 0.05$ and the RBF interpolation in the limit $\delta \to 0$. Notice that this difference is $O(\delta)$.

It should be mentioned that the method works equally well for equispaced and non-equispaced nodes. In [18], we use this algorithm to derive the Laurent series of the inverse of RBF-FD matrices in 1D, 2D and 3D using both equispaced and non-equispaced nodes for singularities of order up to seven.

Other alternatives have been proposed in the past to compute the leading order approximation to RBF interpolants in the limit $\delta \to 0$ [13–16].

6.2. Other applications

The algorithm in Section 5 works also in the case of complex matrices. As a simple example we have considered the matrix

$$A(\delta) = \begin{bmatrix} 1 & -(1+\delta)^2\, e^{\pi(1+\delta)\,i} \\ -(1+\delta)^2\, e^{\pi(1+\delta)\,i} & 1 \end{bmatrix}. \tag{52}$$

This matrix arises in problems of wave scattering by $N = 2$ point-like scatterers. Matrix (52) is the Foldy–Lax matrix [17,24] when the two point-like scatterers are at a distance $(d+\delta)$, where $d = \pi/k_0$ is the resonant distance and k_0 is the wavenumber. The Taylor series of this matrix is

$$A(\delta) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + \delta \begin{bmatrix} 0 & 2+\pi\,i \\ 2+\pi\,i & 0 \end{bmatrix} + \delta^2 \begin{bmatrix} 0 & 1-\dfrac{\pi^2}{2}+2\pi\,i \\ 1-\dfrac{\pi^2}{2}+2\pi\,i & 0 \end{bmatrix} + \ldots. \tag{53}$$

Applying the algorithm in Section 5 results in a singularity of order 1 with a Laurent series for the inverse given by

$$A^{-1}(\delta) = \frac{1}{\delta} \begin{bmatrix} -0.0721 + 0.1133\,i & 0.0721 - 0.1133\,i \\ 0.0721 - 0.1133\,i & -0.0721 + 0.1133\,i \end{bmatrix} + \begin{bmatrix} 0.5153 + 0.0327\,i & -0.0153 - 0.0327\,i \\ -0.0153 - 0.0327\,i & 0.5153 + 0.0327\,i \end{bmatrix} + \ldots.$$

We now consider the case of non-symmetric matrix. The following example was analyzed in [21] and was adapted from [7]. Consider the matrix

$$A(\delta) = \frac{1}{3} \begin{bmatrix} -3 & -5 & -4 \\ 6 & 5 & -2 \\ -3 & 2 & 10 \end{bmatrix} + \frac{\delta}{3} \begin{bmatrix} 6 & 5 & 4 \\ -6 & -2 & 2 \\ 3 & -2 & -7 \end{bmatrix}. \tag{54}$$

In this example, both A_0 and A_1 are singular although $A^{-1}(\delta)$ exits for $\delta > 0$. Applying the algorithm described in Section 5 we find that the Laurent series of the inverse is

$$A^{-1}(\delta) = \frac{1}{\delta}\begin{bmatrix} 2 & 14/9 & 10/9 \\ -2 & -14/9 & -10/9 \\ 1 & 7/9 & 5/9 \end{bmatrix} + \begin{bmatrix} -1 & -41/27 & -28/27 \\ 2 & 77/27 & 46/27 \\ -1 & 34/27 & 14/27 \end{bmatrix} + \dots . \tag{55}$$

The method can also be used in cases in which the perturbed matrix $A(\delta)$ is also singular for all δ. In those cases, the algorithm described in Section 5 computes the Laurent series of the Drazin inverse A^D. Consider, for instance, the following perturbed matrix analyzed in [2]

$$A(\delta) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \delta \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} . \tag{56}$$

The *index* of this matrix is 1 ($\text{range}(A^2) = \text{range}(A)$). Applying the algorithm to matrix (56) results in

$$A^D(\delta) = A^{\#}(\delta) = \frac{1}{\delta}\begin{bmatrix} 1 & -1 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \delta \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} . \tag{57}$$

Terms of order δ^2 or higher are zero in this case. It is straightforward to verify that matrix A^D given in (57) satisfies the properties of the Drazin inverse, namely $A^2\,A^D = A$, $A^D\,A\,A^D = A^D$, and $A\,A^D = A^D\,A$.

7. Conclusions

In this paper we have discussed the implementation of an algorithm to obtain the Laurent series of the inverse of singularly perturbed matrices semi-analytically. The method is semi-analytical because the Laurent series is derived from analytical formulas, but the implementation involves a significant amount of numerical computations. Among other advantages, this algorithm can be applied to solve ill-conditioned linear systems, as the ill-conditioning is completely eliminated through the semi-analytical computation of the Laurent series of the inverse matrix.

The algorithm implemented here is efficient compared to the computation of the inverse of a matrix using a symbolic language such as Mathematica or Maple. We have shown that complexity of the proposed algorithm grows algebraically with the size N of the matrix as cN^3, where c is a constant, but exponentially with the order of the singularity s of the matrix.

Acknowledgements

This work has been supported by Spanish MICINN Grants FIS2013-41802-R and CSD2010-00011.

Appendix A. Computation of the coefficients $R_k(\lambda)$

To compute the coefficients $R_k(\lambda)$ in (19), consider a point (λ', δ) for which (18) exists. Then, we can write

$$\begin{aligned} R(\lambda,\delta) &= \left(A_0 - \lambda'\mathbb{I} - \left((\lambda-\lambda')\mathbb{I} - \tilde{A}(\delta)\right)\right)^{-1} \\ &= \left(\left[\mathbb{I} - \left((\lambda-\lambda')\mathbb{I} - \tilde{A}(\delta)\right) R(\lambda',0)\right](A_0 - \lambda'\mathbb{I})\right)^{-1} \\ &= R(\lambda',0)\left[\mathbb{I} - \left((\lambda-\lambda')\mathbb{I} - \tilde{A}(\delta)\right) R(\lambda',0)\right]^{-1} , \end{aligned} \tag{A.1}$$

where $\tilde{A} = A - A_0 = \sum_{i=1}^{n} \delta^i A_i$. If we now set $\lambda' = \lambda$ and denote $R(\lambda) \equiv R(\lambda, 0)$ we obtain

$$\begin{aligned} &R(\lambda,\delta) = R(\lambda)[\mathbb{I} + \tilde{A}(\delta)R(\lambda)]^{-1} = R(\lambda)\sum_{k=0}^{\infty}[-\tilde{A}(\delta)R(\lambda)]^k = \\ &R(\lambda) - R(\lambda)(\delta A_1 + \delta^2 A_2 + \dots)R(\lambda) + \\ &R(\lambda)(\delta A_1 + \delta^2 A_2 + \dots)R(\lambda)(\delta A_1 + \delta^2 A_2 + \dots)R(\lambda) + \dots = \\ &R(\lambda) - \delta R(\lambda)A_1 R(\lambda) - \delta^2 R(\lambda)A_2 R(\lambda) - \dots + \delta^2 R(\lambda)A_1 R(\lambda)A_1 R(\lambda) + \\ &\delta^3(R(\lambda)A_2 R(\lambda)A_1 R(\lambda) + R(\lambda)A_1 R(\lambda)A_2 R(\lambda)) + \dots , \end{aligned} \tag{A.2}$$

for sufficiently small δ. From (19) and (A.2) and using the definition of $\tilde{A}$, we readily obtain the coefficients (20).

Appendix B. Computation of the coefficients $A_n^{\#}$'s

To compute the coefficients $A_n^{\#}$'s in expansion (24), we substitute the coefficients (20) into (23). That is,

$$\begin{aligned} A_n^{\#} &= \frac{1}{2\pi i}\int_{C_0}\frac{R_n(\lambda)}{\lambda}d\lambda \\ &= \frac{1}{2\pi i}\int_{C_0}\frac{1}{\lambda}\sum_{\nu_1+\nu_2+\cdots+\nu_p=k}(-1)^p R(\lambda)A_{\nu_1}R(\lambda)A_{\nu_2}\ldots A_{\nu_p}R(\lambda)d\lambda \\ &= \sum_{\nu_1+\nu_2+\cdots+\nu_p=k}(-1)^p\frac{1}{2\pi i}\int_{C_0}\frac{1}{\lambda}R(\lambda)A_{\nu_1}R(\lambda)A_{\nu_2}\ldots A_{\nu_p}R(\lambda)d\lambda \\ &= \sum_{\nu_1+\nu_2+\cdots+\nu_p=k}(-1)^p \mathrm{Res}_{\lambda=0}\left(\frac{1}{\lambda}R(\lambda)A_{\nu_1}R(\lambda)A_{\nu_2}\ldots A_{\nu_p}R(\lambda)\right) \end{aligned} \tag{B.1}$$

Now we substitute $R(\lambda)$ by its Laurent series $\frac{1}{\lambda}P_0+\sum_{n=0}^{\infty}\lambda^n(A^{\#})^{n+1}$ and collect the terms with $\frac{1}{\lambda}$ which have the form:

$$\sum_{\sigma_1+\sigma_2+\cdots+\sigma_{p+1}=0} S_{\sigma_1+1}A_{\nu_1}S_{\sigma_2+1}\ldots A_{\nu_p}S_{\sigma_{p+1}+1} \tag{B.2}$$

then we change indexes to $\mu_i=\sigma_i+1$ for $i=1,\ldots,p+1$ and we get:

$$\sum_{\mu_1+\mu_2+\cdots+\mu_{p+1}=p+1} S_{\mu_1}A_{\nu_1}S_{\mu_2}\ldots A_{\nu_p}S_{\mu_{p+1}} \tag{B.3}$$

From here we substitute in the last expression for Q_n and take into account the tensor notation we get eq. (24).

References

[1] K.E. Avrachenkov, J.B. Lasserre, The fundamental matrix of singularly perturbed Markov chains, Adv. Appl. Probab. 31 (1999) 679–697.
[2] K.E. Avrachenkov, M. Haviv, Perturbation of null spaces with application to the eigenvalue problem and generalized inverses, Linear Algebra Appl. 369 (2003) 1–25.
[3] K.E. Avrachenkov, M. Haviv, P.G. Howlett, Inversion of analytic matrix functions that are singular at the origin, SIAM J. Matrix Anal. Appl. 22 (2001) 1175–1189.
[4] K.E. Avrachenkov, J.B. Lasserre, Analytic perturbation of generalized inverses, Linear Algebra Appl. 438 (2013) 1793–1813.
[5] A. Ben-Israel, T.N.E. Greville, Generalized Inverses: Theory and Applications, Springer, New York, NY, ISBN 0-387-00293-6, 2003.
[6] H. Baumgartel, Analytic Perturbation Theory for Matrices and Operators, Birkhäuser, Basel, 1985.
[7] S.L. Campbell, C.D. Meyer, N.J. Rose, Applications of the Drazin inverse to linear systems of differential equations with singular constant coefficients, SIAM J. Appl. Math. 31 (1976) 411–425.
[8] F. Delebecque, A reduction process for perturbed Markov chains, SIAM J. Appl. Math. 43 (1983) 325–350.
[9] F.W. Dorr, The numerical solution of singular perturbations of boundary value problems, SIAM J. Numer. Anal. 7 (1970) 281–313.
[10] F.W. Dorr, An example of ill-conditioning in the numerical solution of singular perturbation problems, Math. Comput. 25 (1971) 271–283.
[11] T.A. Driscoll, B. Fornberg, Interpolation in the limit of increasingly flat radial basis functions, Comput. Math. Appl. 43 (2002) 413–422.
[12] G.E. Fasshauer, Meshfree Approximation Methods with MATLAB, World Scientific Publishing Co., Singapore, 2007.
[13] B. Fornberg, G. Wright, Stable computation of multiquadric interpolation for all values of the shape parameters, Comput. Math. Appl. 48 (2004) 853–867.
[14] B. Fornberg, C. Piret, Stable algorithm for flat radial basis functions on a sphere, SIAM J. Sci. Comput. 30 (2008) 60–80.
[15] B. Fornberg, E. Larsson, N. Flyer, Stable computations with Gaussian radial basis functions, SIAM J. Sci. Comput. 33 (2011) 869–892.
[16] B. Fornberg, E. Lehto, C. Powell, Stable calculation of Gaussian-based RBF-FD stencils, Comput. Math. Appl. 65 (2013) 627–637.
[17] L.L. Foldy, The multiple scattering of waves, Phys. Rev. 67 (1945) 107–119.
[18] P. González, V. Bayona, M. Moscoso, M. Kindelan, Laurent series based RBF-FD method to avoid ill-conditioning, Eng. Anal. Bound. Elem. 52 (2015) 24–31.
[19] R. Hassin, M. Haviv, Mean passage times and nearly uncoupled Markov chains, SIAM J. Discrete Math. 5 (1992) 386–397.
[20] T. Kato, Perturbation Theory for Linear Operators, Springer, 1995.
[21] B.F. Lamond, A generalized inverse method for asymptotic linear programming, Math. Program. 43 (1989) 71–86.
[22] B.F. Lamond, An efficient basis update for asymptotic linear programming, Linear Algebra Appl. 184 (1993) 83–102.
[23] E. Larsson, B. Fornberg, Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions, Comput. Math. Appl. 49 (2005) 103–130.
[24] M. Lax, Multiple scattering of waves, Rev. Mod. Phys. 23 (1951) 287–310.
[25] R. Schaback, Multivariate interpolation by polynomials and radial basis functions, Constr. Approx. 21 (2005) 293–317.
[26] G.W. Stewart, On the perturbation of pseudo-inverses, projections and linear least squaresproblems, SIAM Rev. 15 (1977) 634–662.
[27] G.B. Wright, B. Fornberg, Scattered node compact finite difference-type formulas generated from radial basis functions, J. Comput. Phys. 212 (2006) 99–123.