Fast, stable interpolation of well data using the norm function

Jeremy Levesley*

March 25, 2009

Abstract

We present an iterative algorithm for computing an approximation to data in wells using the norm function in three dimensions augmented with a tensor product of two dimensional lagrange functions with one dimensional linear interpolants. This augmentation can be thought of as a trend. The algorithm avoids the stability problems associated with data which has very different separation in different directions. In this case, data is close in the vertical direction inside wells, but the wells are relatively far apart. We will give a estimate of the convergence rate of the algorithm in terms of the vertical point spacing and the horizontal well separation.

1 Introduction

In this paper we will describe a practical algorithm for interpolating data that is in m columns, each of depth d, with n pieces of data in each column. For simplicity we will assume that the data is of the form

$$\mathbf{y}_{i,j} = (\mathbf{x}_i, z_j) \in \mathbb{R}^3,$$

where $\mathbf{x}_i \in \mathbb{R}^2$, $i = 1, \cdots, m$, and $z_j = jh$, $j = 0, \cdots, n$, with $h = d/n$. Thus we are assuming that the data is uniformly spaced within the wells. We wish to interpolate data $f_{i,j}$, $i = 1, \cdots, m$, $j = 0, \cdots, n$.

In radial basis function interpolation using the norm function one would typically form an approximation

$$s(\mathbf{y}) = \sum_{i=1}^{m} \sum_{j=0}^{n} \alpha_{i,j} \phi(\mathbf{y} - \mathbf{y}_{i,j}) + b, \tag{1}$$

*Department of Mathematics, University of Leicester, Leicester LE1 7RH, England, jl1@le.ac.uk.
Research was supported by EPSRC Grant EP/F010028/1

where b is a constant. The unknown coefficients $\alpha_{i,j}$ and b ($(n+1)m+1$ unknowns) are determined by the $((n+1)m+1)$ conditions

$$s(\mathbf{y}_{i,j}) = f_{i,j}, \quad i = 1, \cdots, m, j = 0, \cdots, n,$$

and

$$\sum_{i=1}^{m} \sum_{j=0}^{n} \alpha_{i,j} = 0.$$

In the following table we will consider the problem were we have just 2 columns of depth 1, one at $(0, 0)$ and the other at $(0, 10)$. We will give the condition number of the interpolation matrix for this problem which is the ratio of the largest and smallest eigenvalue. If the condition number is large one cannot even solve the interpolation sometimes, and when one can, the computed solution is not anywhere near the exact solution. We can see below that the condition number grows with order n^2.

n	condition number
100	2.1 (5)
200	8.3 (5)
400	3.3 (6)
800	1.3 (7)

Table 1: Condition number of standard interpolation matrix

In order to get around this conditioning problem we use the straightforward observation that the norm function

$$\phi(\mathbf{y}) = \|\mathbf{y}\|, \quad \mathbf{y} \in \mathbb{R}^3,$$

is the mod function when restricted to one dimension. This makes it trivial to construct an interpolant to data in each individual column. As we will see, this interpolant behaves like a one dimensional linear polynomial at the other wells, in other words, in the far field.

2 The interpolant

In order to construct the linear interpolant in a column we introduce the functions

$$\begin{aligned}
\psi_h(\mathbf{y}) &= \frac{\phi(\mathbf{y} + h\mathbf{e}) - 2 * \phi(\mathbf{y}) + \phi(\mathbf{y} - h\mathbf{e})}{2h} \\
\chi_t(\mathbf{y}) &= \frac{\phi(\mathbf{y} - h\mathbf{e}) - \phi(\mathbf{y}) + h}{2h} \\
\chi_b(\mathbf{y}) &= \frac{\phi(\mathbf{y} + h\mathbf{e}) - \phi(\mathbf{y}) + h}{2h}
\end{aligned}$$

where $\mathbf{e} = (0, 0, 1)$ is a unit vector in the vertical direction.

It is straightforward to check the results of the following lemma.

Lemma 1. *For $h > 0$,*

(i) $\psi_h(\lambda\mathbf{e}) = 0$, $|\lambda| \geq h$,

(ii) $\psi_h(0) = 1$,

(iii) $\chi_t(\lambda\mathbf{e}) = 0$, $\lambda \leq -h$,

(iv) $\chi_t(0) = 1$,

(v) $\chi_b(\lambda\mathbf{e}) = 0$, $\lambda \geq h$,

(vi) $\chi_b(0) = 1$.

Thus these functions form a set of Lagrange functions for interpolation in each column.

In order to discuss our approximation algorithm we need to understand how these functions behave for large argument.

Proposition 2. *Let $\mathbf{y} = \mathbf{x} + \beta\mathbf{e}$ for some $\beta \in \mathbb{R}$, with $|\beta| \leq d \ll \|x\|$. Then*

(i)

$$\psi_h(\mathbf{y}) = \frac{h}{2\|\mathbf{x}\|} + \mathcal{O}\left(\frac{h}{\|\mathbf{x}\|^3}\right).$$

(ii)

$$\chi_t(\mathbf{y}) = \frac{1}{2} - \frac{2\beta - h}{2\|\mathbf{x}\|} + \mathcal{O}\left(\frac{1}{\|\mathbf{x}\|^3}\right).$$

(iii)

$$\chi_b(\mathbf{y}) = \frac{1}{2} + \frac{2\beta + h}{2\|\mathbf{x}\|} + \mathcal{O}\left(\frac{1}{\|\mathbf{x}\|^3}\right).$$

Proof. Since $\mathbf{y} = \mathbf{x} + \beta\mathbf{e}$,

$$\begin{aligned}
\phi(\mathbf{y}) &= \|\mathbf{x} + \beta\mathbf{e}\| \\
&= (\|\mathbf{x}\|^2 + \beta^2)^{1/2} \\
&= \|\mathbf{x}\| \left(1 + \frac{\beta^2}{\|\mathbf{x}\|^2}\right)^{1/2} \\
&= \|x\| \left(\sum_{k=0}^{\infty} \binom{1/2}{k} \left(\frac{\beta^2}{\|\mathbf{x}\|^2}\right)^k\right).
\end{aligned}$$

Hence, since $|(x+h)^k - 2x^k + (x-h)^k| \le Ch^2x^{k-2}$, for any $k \ge 2$,

$$\begin{aligned}
\psi_h(\mathbf{y}) &= \frac{\phi(\mathbf{y}+h\mathbf{e}) - 2*\phi(\mathbf{y}) + \phi(\mathbf{y}-h\mathbf{e})}{2h} \\
&= \frac{\|\mathbf{x}+(\beta+h)\mathbf{e}\| - 2\|\mathbf{x}+\beta\mathbf{e}\| + \|\mathbf{x}+(\beta-h)\mathbf{e}\|}{2h} \\
&= \|\mathbf{x}\| \left(\sum_{k=0}^{\infty} \left(\frac{(\beta+h)^{2k} - 2\beta^{2k} + (\beta-h)^{2k}}{2h\|\mathbf{x}\|^{2k}} \right) \right) \\
&= \frac{h}{2\|\mathbf{x}\|} - \frac{6\beta^2 h + h^3}{8\|\mathbf{x}\|^3} + \mathcal{O}\left(\frac{h}{\|\mathbf{x}\|^5} \right) \\
&= \frac{h}{2\|\mathbf{x}\|} + \mathcal{O}\left(\frac{h}{\|\mathbf{x}\|^3} \right)
\end{aligned}$$

We also have, since $|(x-h)^k - x^k| \le Chx^{k-1}$, for any $k \ge 1$

$$\begin{aligned}
\chi_t(\mathbf{y}) &= \frac{\phi(\mathbf{y}-h\mathbf{e}) - \phi(\mathbf{y}) + h}{2h} \\
&= \frac{\|\mathbf{x}+(\beta-h)\mathbf{e}\| - \|\mathbf{x}+\beta\mathbf{e}\| + h}{2h} \\
&= \frac{1}{2} + \|x\| \left(\sum_{k=0}^{\infty} \left(\frac{(\beta-h)^{2k} - \beta^{2k}}{2h\|\mathbf{x}\|^{2k}} \right) \right) \\
&= \frac{1}{2} - \frac{2\beta - h}{2\|\mathbf{x}\|} + \mathcal{O}\left(\frac{1}{\|\mathbf{x}\|^3} \right).
\end{aligned}$$

The result for χ_b is more or less identical. ■

To compute an interpolant we first form the functions

$$s_i^1(\mathbf{y}) = \sum_{j=1}^{n-1} f_{i,j}\psi_h(\mathbf{y}-\mathbf{y}_{i,j}) + f_{i,0}\chi_b(\mathbf{y}-\mathbf{y}_{i,1}) + f_{i,n}\chi_t(\mathbf{y}-\mathbf{y}_{i,n}), \quad i = 1, \cdots, m.$$

The next lemma shows that if $\mathbf{y}$ is a long way from the ith column then $s_i^1(\mathbf{y})$ behaves linearly with variation in the $\mathbf{e}$ direction.

Proposition 3. *Let* $\mathbf{y} = \mathbf{x} + z\mathbf{e}$. *For each* $i = 1, 2, \cdots, m$,

$$s_i^1(\mathbf{y}) = \left(1 + \frac{h}{2\|\mathbf{x}-\mathbf{x}_i\|} \right) \frac{f_{i,0}+f_{i,n}}{2} + \left(\frac{2df_{i,n} + h\sum_{j=1}^{n-1} f_{i,j}}{2\|\mathbf{x}-\mathbf{x}_i\|} \right) + \frac{(f_{i,0}-f_{i,n})z}{\|\mathbf{x}-\mathbf{x}_i\|} + \mathcal{O}\left(\frac{1}{\|\mathbf{x}\|^3} \right).$$

Proof. For $i = 1, \cdots, m$,

$$\sum_{j=1}^{n-1} f_{i,j}\psi_h(\mathbf{y}-\mathbf{y}_{i,j}) = \sum_{j=1}^{n-1} f_{i,j}\psi_h(\mathbf{x}-\mathbf{x}_i+(z-jh)\mathbf{e})$$

$$
\begin{aligned}
&= \frac{h}{2\|\mathbf{x}-\mathbf{x}_i\|}\sum_{j=1}^{n-1} f_{i,j} + \left\{ h\sum_{j=1}^{n-1} f_{i,j} \right\} \times \mathcal{O}\left(\frac{1}{\|\mathbf{x}-\mathbf{x}_i\|^3}\right) \\
&= \frac{h}{2\|\mathbf{x}-\mathbf{x}_i\|}\sum_{j=1}^{n-1} f_{i,j} + \mathcal{O}\left(\frac{1}{\|\mathbf{x}-\mathbf{x}_i\|^3}\right).
\end{aligned}
$$

Looking now at the other part of $s_i^1(\mathbf{y})$, we have

$$
\begin{aligned}
& f_{i,0}\chi_b(\mathbf{y}-\mathbf{y}_{i,1}) + f_{i,n}\chi_t(\mathbf{y}-\mathbf{y}_{i,n}) \\
&= f_{i,0}\chi_b(\mathbf{x}-\mathbf{x}_i+z\mathbf{e}) + f_{i,1}\chi_b(\mathbf{x}-\mathbf{x}_i+(z-d)\mathbf{e}) \\
&= f_{i,0}\left(\frac{1}{2}+\frac{2z+h}{2\|\mathbf{x}-\mathbf{x}_i\|}\right) + f_{i,n}\left(\frac{1}{2}-\frac{2(z-d)-h}{2\|\mathbf{x}-\mathbf{x}_i\|}\right) + \mathcal{O}\left(\frac{1}{\|\mathbf{x}-\mathbf{x}_i\|^3}\right) \\
&= \left(1+\frac{h}{2\|\mathbf{x}-\mathbf{x}_i\|}\right)\frac{f_{i,0}+f_{i,n}}{2} + \frac{d\, f_{i,n}}{\|\mathbf{x}-\mathbf{x}_i\|} + \frac{(f_{i,0}-f_{i,n})z}{\|\mathbf{x}-\mathbf{x}_i\|} + \mathcal{O}\left(\frac{1}{\|\mathbf{x}-\mathbf{x}_i\|^3}\right).
\end{aligned}
$$

Putting the last two equations together we have the required result. ■

Let

$$
\ell_{i,k}^1(z) = \left(1+\frac{h}{2\|\mathbf{x}_k-\mathbf{x}_i\|}\right)\frac{f_{i,0}+f_{i,n}}{2} + \left(\frac{2df_{i,n}+h\sum_{j=1}^{n-1} f_{i,j}}{2\|\mathbf{x}_k-\mathbf{x}_i\|}\right) + \frac{(f_{i,0}-f_{i,n})z}{\|\mathbf{x}_k-\mathbf{x}_i\|},
$$

and

$$
p_i^1(z) = \sum_{i\neq k} \ell_{i,k}(z).
$$

Thus p_i is the vertical variation at the ith well due to the sum of the vertical variations in the far fields of the interpolants along each well.

In order to construct our interpolant we need to build a two dimensional Lagrange basis for the points $\mathbf{x}_i$, $i = 1, \cdots, m$. We can do this in any way we like, but for consistency here we will use a radial basis function of the form (1)

$$
\rho_i(\mathbf{x}) = \sum_{k=1}^{n} \alpha_{i,k}\|\mathbf{x}-\mathbf{x}_k\| + b_i,
$$

where we compute the coefficients $\alpha_{i,k}$ and b_i by interpolation

$$
\rho_i(\mathbf{x}_j) = \delta_{i,j}, \quad i,j = 1, \cdots, n,
$$

and

$$
\sum_{k=0}^{n} \alpha_{i,k} = 0.
$$

To compute our intermediate approximation we form

$$
\sigma^1(\mathbf{y}) = \sum_{i=1}^{n} s_i^1(\mathbf{y}).
$$

If we compute the residual

$$r_{i,j} := f_{i,j} - s^1(\mathbf{y}_{i,j}), \quad i = 1, \cdots, m,\ j = 0, \cdots, n,$$

then the previous proposition tells us that for each $i = 1, \cdots, m$, $r_{i,j}$, $j = 0, \cdots, n$ have an approximately linear relationship. Thus let us form the tensor product

$$q^1(\mathbf{y}) = \sum_{i=1}^{n} \rho_i(\mathbf{x}) p_i^1(z).$$

Our first approximate interpolant is then

$$s^1(\mathbf{y}) = \sigma^1(\mathbf{y}) - q^1(\mathbf{y}).$$

We produce better approximation to the interpolant by repeating the above procedure using successive residuals as the target function.

3 Convergence and complexity

In this section we consider the error and see that it depends on the distance between the wells. We also show that the complexity of the algorithm is linear with the number of points.

Theorem 4. *For $i = 1, \cdots, m$ and $j = 0, \cdots, n$,*

$$|s^1(\mathbf{y}_{i,j}) - f_{i,j}| \leq \frac{C}{\delta^3},$$

where $\delta = \min_{i \neq k} \|\mathbf{x}_i - \mathbf{x}_k\|$.

Proof. For $i = 1, \cdots, m$ and $j = 0, \cdots, n$, using Lemma 1

$$\begin{aligned} |s^1(\mathbf{y}_{i,j}) - f_{i,j}| &= \sigma^1(\mathbf{y}_{i,j}) - q^1(y_{i,j}) - f_{i,j} \\ &= f_{i,j} + \sum_{k \neq i} s_k^1(\mathbf{y}_{i,j}) - q(\mathbf{y}_{i,j}) - f_{i,j} \\ &= \sum_{k \neq i} \left(\ell_{i,k}^1(z_j) + \mathcal{O}\left(\frac{1}{\|\mathbf{x}_i - \mathbf{x}_k\|^3} \right) \right) - p_i(z_j), \end{aligned}$$

since $q^1(\mathbf{y}_{i,j}) = \sum_{k=1}^n \rho_k(\mathbf{x}_i) p_k^1(z_j) = \sum_{k=1}^n \delta_{i,k} p_k^1(z_j) = p_i^1(z_j)$. However, by definition,

$$\sum_{k \neq i} \ell_{i,k}^1(z_j) = p_i^1(z_j),$$

and the result is established. ■

Thus we see that, as opposed to the usual case where the larger the well-separation the more difficult interpolation is (see Table 1), this algorithm converges faster the better separated the wells are.

Direct inversion of the interpolation equations would lead to an algorithm of order $(mn)^3$. It is straightforward to see that this algorithm is also of order $(nm)^2$ operations. With some more sophisticated progamming and explicit use of the far-field representations of the approximations we could develop a fast algorithm, i.e. one of the order of mn or $mn\log(mn)$. Such algorithms are available for quasiuniformly distributed data such as described by Wendland in [1, Chapter 15].

4 Numerical examples

In the first table below we list a set of numerical experiments. We generate m randomly spaced columns of data with base in the square $[10, 10]$. Each column is of height 1. The target data is randomly generated in the interval $[0, 1]$. Thus we can see that we get convergence to an interpolant in a small number of iterations. The results confirm the analysis of the previous section. The number of iterations depends on the number of columns (and thus the minimal distance between the columns if we are in a fixed area), but not the amount of data in each column.

m	n	error	iterations
2	100	5 (-12)	3
2	200	1 (-12)	3
2	400	4 (-10)	3
2	800	2 (-11)	3
4	100	8 (-11)	3
4	200	6 (-9)	3
4	400	9 (-10)	3
4	800	3 (-11)	3
8	100	4 (-11)	4
8	200	4 (-10)	4
8	400	1 (-9)	3
8	800	5 (-9)	3
16	100	2 (-11)	5
16	200	2 (-9)	7
16	400	1 (-9)	4
16	800	1 (-9)	5

Table 2: Number of iterations for m wells in [10,10] and n data in each well

In the next table we see that it is not the number of columns which effects the number of iterations but the spacing. Here we have a fixed number of 10 columns with 100 data in each column. The side length of the square is s in which these 10 columns are randomly positioned. We perform the algorithm for 20 sets of random data and compute the mean average number of iterations to achieve an error of less than 10^{-8}.

We start with s=6, as for smaller numbers we see instances of lack of convergence of the algorithm as two wells might be too near to eachother.

s	iterations
6	7.95
8	5.85
10	4.2
12	4.1
14	3.45
16	3.35
18	3.4
20	3.2

Table 3: Average number of iterations for 10 wells in $[s, s]$ with 100 data in each well

As can be seen the number of iterations decreases as the point become better spaced.

References

[1] H. Wendland, *Scattered Data Approximation*, Cambridge Monographs on Applied and Computational Mathematics (No. 17), 2004.