https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions
Open Access
EGU

Fast Barnes Interpolation

Bruno K. Zürcher
Federal Office of Meteorology and Climatology MeteoSwiss, Zurich, Switzerland
Correspondence: Bruno K. Zürcher (bruno.zuercher@meteoswiss.ch)

Abstract. Barnes interpolation is a method that is widely used in geospatial sciences like meteorology to remodel data values recorded at irregularly distributed points into a representative analytical field. When implemented naively, the computational complexity of Barnes interpolation depends directly on both the number of sample points and the number of grid points. In the era of highly resolved grids and overwhelming numbers of sample points, which originate e.g. from the Internet of Things or from crowd-sourced data, this computation can be quite demanding even on high-performance machines.

This paper presents new approaches how very good approximations of Barnes interpolation can be implemented using fast algorithms. Two use cases are in particular considered, namely (1) where the used grid is embedded in the Euclidean plane and (2) where the grid is located on the unit sphere.

1 Introduction

In the early days of numerical weather prediction, various objective analysis methods were developed to automatically produce the initial conditions from the irregularly spaced observational data (Daley, 1991), one of which was Barnes interpolation (Barnes, 1964). However, objective analysis soon lost its significance in this field against statistical approaches. Today, Barnes method is still used to create grid-based isoline visualizations of geospatial data, like for instance in the meteorological workstation project NinJo (Koppert et al., 2004), which is commonly developed by the Deutscher Wetterdienst, the Bundeswehr Geophysical Service, MeteoSwiss, the Danish Meteorological Institute and the Meteorological Service of Canada.

Barnes interpolation $f(\boldsymbol{x}) : D \longrightarrow \mathbb{R}$ for an arbitrary point $\boldsymbol{x} \in D$ and a given set of sample points $\boldsymbol{x_k} \in D$ with observation values $f_k \in \mathbb{R}$ for $k = 1, ..., N$ is defined as

$$f(\boldsymbol{x}) = \frac{\sum_{k=1}^{N} f_k \cdot w_k(\boldsymbol{x})}{\sum_{k=1}^{N} w_k(\boldsymbol{x})} \tag{1}$$

with Gaussian weights

$$w_k(\boldsymbol{x}) = \mathrm{e}^{-\frac{d(\boldsymbol{x},\boldsymbol{x_k})^2}{2\sigma^2}}, \tag{2}$$

a distance function $d : D \times D \longrightarrow \mathbb{R}$ and a Gaussian width parameter σ.

If Barnes interpolation is computed in a straightforward way for a regularly spaced $W \times H$ grid, the computational complexity is given by $\mathcal{O}(N \cdot W \cdot H)$ as easily can be seen from the threefold nested loops of the algorithm given below. Consequently, for big values of N or dense grids, a naive implementation of Barnes interpolation turns out to be unreasonably slow.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions
Open Access
EGU

Algorithm Naive Barnes Interpolation

Input: sample point coordinates $\boldsymbol{x_k} = (x_k, y_k)$ and sample values f_k for $k = 1, ..., N$ and fall-off parameter σ.
Output: $W \times H$ field F holding Barnes interpolation values.

for $i = 1$ **to** W **do**
  for $j = 1$ **to** H **do**
    Set numerator $p = 0$ and denominator $q = 0$.
    for $k = 1$ **to** N **do**
      Compute distance between current grid point $\boldsymbol{g}[i,j]$ and current sample point $\boldsymbol{x_k}$ as $d = d(\boldsymbol{g}[i,j], \boldsymbol{x_k})$.
      Compute Gaussian weight $w = \mathrm{e}^{-\frac{d^2}{2\sigma^2}}$.
      Update $p \mathrel{+}= w \cdot f_k$ and $q \mathrel{+}= w$.
    end for
    Set $F[i,j] = p \,/\, q$.
  end for
end for
return F.

The fact that the Gaussian weight function quickly approaches 0 for increasing distances leads to a first improvement attempt, which consists in neglecting all terms in the sums of (1), for which the weights w_k drop below a certain limit w_0, e.g. $w_0 = 0.001$. This is equivalent to take only observation points $\boldsymbol{x_k}$ into account that lie within the distance $r_0 = \sigma\sqrt{-2 \ln w_0}$ from the interpolation point $\boldsymbol{x}$. Thus, the described procedure requires the ability to quickly extract all observation points $\boldsymbol{x_k}$ that lie within a distance r_0 from point $\boldsymbol{x}$. Data structures that support such searches are e.g. so called k-d-trees (Bentley, 1975) or quadtrees (Finkel and Bentley, 1974).

This improved approach actually reduces the required computation time by a constant factor, but the computational complexity remains in the same order. To see this, note that a specific sample point $\boldsymbol{x_k}$ contributes to the interpolation value of exactly those grid points that are contained in the circular disk $B_{r_0}(\boldsymbol{x_k}) = \{\boldsymbol{q} \in D \mid d(\boldsymbol{x_k}, \boldsymbol{q}) < r_0\}$ of radius r_0 around it. Note also that in a regularly spaced grid the number of affected grid points is roughly the same for each sample point. If now the number of grid points W and H is in each dimension increased by a factor κ – i.e. the grid becomes denser – the number of grid points contained in $B_{r_0}(\boldsymbol{x_k})$ grows accordingly, namely by a factor κ^2, which shows that the algorithmic costs rise in direct dependency to W and H. Since this is obviously as well true for the number of sample points N, the improved algorithm also has complexity $\mathcal{O}(N \cdot W \cdot H)$.

In this paper, we discuss a new and fast technique to compute very good approximations of Barnes interpolation. The theoretical background is presented in Sect. 2 and 3. After, we investigate two use cases for the domain D,

(i) $D = \mathbb{R}^2$ the Euclidean plane with the usual distance $d(\boldsymbol{p}, \boldsymbol{q}) = \|\boldsymbol{p} - \boldsymbol{q}\|_2$ in full detail in Sect. 4 and 5.4,

(ii) $D = \mathcal{S}^2 = \{\boldsymbol{p} \in \mathbb{R}^3 \mid \|\boldsymbol{p}\|_2 = 1\}$ the unit sphere with $d(\boldsymbol{p}, \boldsymbol{q})$ the spherical distance between $\boldsymbol{p}$ and $\boldsymbol{q}$ as a broad outline in Sect. 5.5.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions

2 Conclusions from Central Limit Theorem

For a set $\{X_k\}_{k=1}^n$ of independent and identically distributed random variables with mean μ and variance σ^2, the central limit theorem (Klenke, 2020) states that the probability distribution of their sum converges to a normal distribution, if n approaches infinity, formally

$$P\left[\frac{X_1+\cdots+X_n}{\sqrt{n}} \leq a\right] \quad \xrightarrow[n\to\infty]{} \quad \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{a} \mathrm{e}^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2} dt. \tag{3}$$

Without loss of generality we assume in the further discussion $\mu = 0$. Let $p(x)$ denote the PDF (probability density function) of the scaled random variables $\{\frac{1}{\sqrt{n}}X_k\}_{k=1}^n$ that consequently have the variance $\frac{\sigma^2}{n}$. Since the PDF of a sum of random variables corresponds to the convolution of their individual PDFs, we find on the other hand

$$P\left[\frac{X_1+\cdots+X_n}{\sqrt{n}} \leq a\right] = \int_{-\infty}^{a} p^{*n}(x)\,dx,$$

where $p^{*n}(x)$ denotes the n-fold convolution of $p(x)$ with itself. With that result we can write relationship (3) equivalently in an unintegrated form as

$$p^{*n}(x) \quad \xrightarrow[n\to\infty]{} \quad \frac{1}{\sqrt{2\pi}\sigma}\,\mathrm{e}^{-\frac{x^2}{2\sigma^2}}, \tag{4}$$

which leads directly to

Approximation 1. *For sufficiently large n, the n-fold self-convolution of a probability density function $p(x)$ with mean $\mu = 0$ and variance $\frac{\sigma^2}{n}$ approximates a Gaussian with mean 0 and variance σ^2, i.e.*

$$p^{*n}(x) \approx \frac{1}{\sqrt{2\pi}\sigma}\,\mathrm{e}^{-\frac{x^2}{2\sigma^2}}.$$

Note that this approximation is valid for arbitrary PDFs $p(x)$ with mean $\mu = 0$ and variance $\frac{\sigma^2}{n}$.

A particular simple PDF is given by the uniform distribution. We therefore define a family of uniform PDFs $\{u_n(x)\}_{n=1}^{\infty}$, of which each member $u_n(x)$ has mean 0 and variance $\frac{\sigma^2}{n}$. These uniform PDFs can be expressed by means of elementary rectangular functions

$$r_n(x) = \begin{cases} 1 & \text{for } |x| \leq \sqrt{\frac{3}{n}}\sigma \\ 0 & \text{otherwise} \end{cases} \quad \text{where } n = 1, 2, \cdots, \tag{5}$$

such that

$$u_n(x) = \frac{1}{2\sqrt{\frac{3}{n}}\sigma}\, r_n(x) = \begin{cases} \frac{1}{2\sqrt{\frac{3}{n}}\sigma} & \text{for } |x| \leq \sqrt{\frac{3}{n}}\sigma \\ 0 & \text{otherwise} \end{cases}$$

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

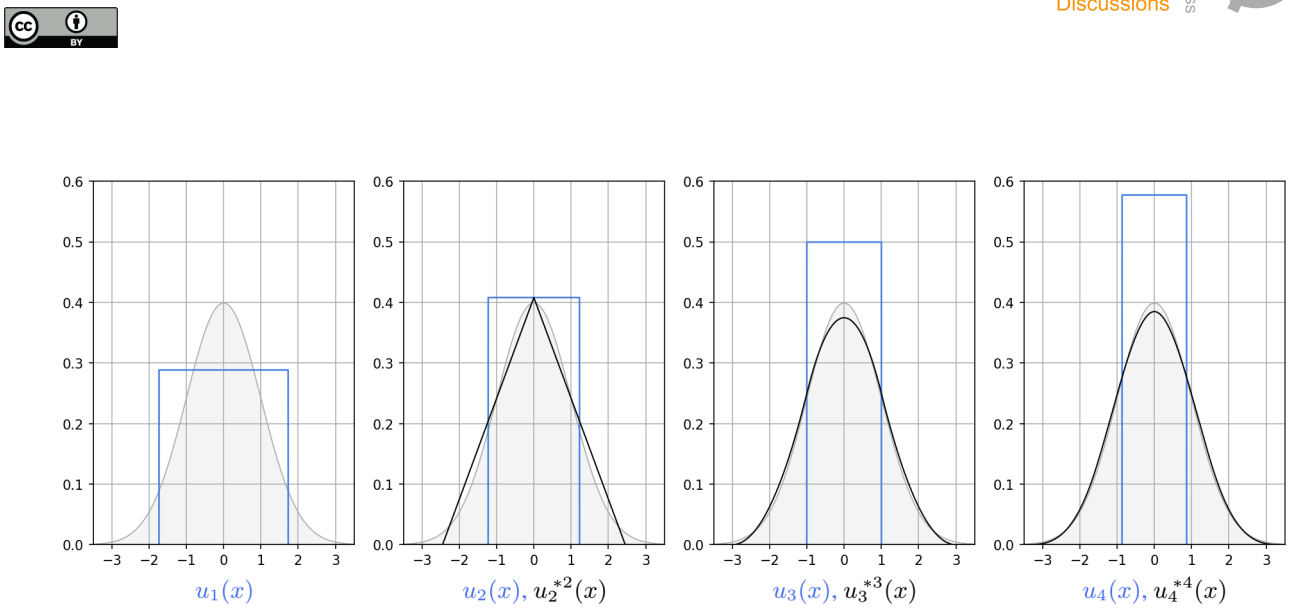

Figure 1. From left to right: In blue the plot of the PDFs of the uniform distributions $u_1(x)$, $u_2(x)$, $u_3(x)$ and $u_4(x)$ for $\sigma = 1$. In black their self-convolutions $u_2^{*2}(x)$, $u_3^{*3}(x)$ and $u_4^{*4}(x)$. The area covered by the PDF of the normal distribution is indicated in grey.

where $n = 1, 2, \cdots$. According to convergence relation (4) and approximation 1, the series of the n-fold self-convolutions $\{u_n^{*n}(x)\}_{n=1}^{\infty}$ converges to a Gaussian with mean 0 and variance σ^2. The converging behavior can actually be examined visually in Fig. 1, which plots the n-fold self-convolution of the first few family members.

The central limit theorem can also be stated more generally for i.i.d. m-dimensional random vectors $\{\boldsymbol{X_k}\}_{k=1}^{n}$, refer for instance to (Muirhead, 1982; Klenke, 2020). Supposing the $\boldsymbol{X_k}$ to have a mean vector $\boldsymbol{\mu} = \boldsymbol{0}$ and a covariance matrix $\boldsymbol{\Sigma}$, we can follow the same line of argument as in the one-dimensional case. Let $p(\boldsymbol{x})$ be the joint PDF of the scaled random variables $\{\frac{1}{\sqrt{n}}\boldsymbol{X_k}\}_{k=1}^{n}$, which therefore have a zero mean vector and the covariance matrix $\frac{1}{n}\boldsymbol{\Sigma}$. Then the limit law writes in m dimensions as

$$p^{*n}(\boldsymbol{x}) \quad \xrightarrow[n\to\infty]{} \quad \frac{1}{(2\pi)^{\frac{m}{2}}\sqrt{\det\boldsymbol{\Sigma}}}\, \mathrm{e}^{-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}}.$$

For the remainder of the discussion, we fix the number of dimensions to $m = 2$ and for the sake of better readability, we write the vector argument $\boldsymbol{x}$ in its component form (x, y), if it is appropriate. In case the random vectors $\boldsymbol{X_k}$ are isotropic, i.e. do not have any preference in a specific spatial direction, the covariance matrix is a multiple of the identity matrix $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$ and the limit law simplifies in two dimensions to

$$p^{*n}(\boldsymbol{x}) \quad \xrightarrow[n\to\infty]{} \quad \frac{1}{2\pi\sigma^2}\, \mathrm{e}^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}\|^2}. \tag{6}$$

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions

In the following step, we are aiming to substitute $p(\boldsymbol{x})$ with the members of the family $\{u_n^{(2)}(\boldsymbol{x})\}_{n=1}^{\infty}$ of two-dimensional uniform distributions over a square-shaped domain, which are defined as

$$u_n^{(2)}(x,y) = u_n(x)\cdot u_n(y) = \frac{n}{12\sigma^2}\, r_n(x)\cdot r_n(y)$$
$$= \begin{cases} \frac{n}{12\sigma^2} & \text{for } |x|,|y| \le \sqrt{\frac{3}{n}}\,\sigma \\ 0 & \text{otherwise} \end{cases} \quad \text{for } n = 1,2,\cdots .$$

With this definition the members $u_n^{(2)}$ have a mean vector $\mathbf{0}$ and an isotropic covariance matrix $\frac{\sigma^2}{n}\boldsymbol{I}$ and hence satisfy the prerequisite of limit law (6). Note also that $u_n^{(2)}(\boldsymbol{x})$ is separable because $u_n^{(2)}(x,y) = u_n(x)\cdot u_n(y)$. As a consequence of the latter, the n-fold self-convolution of $u_n^{(2)}(\boldsymbol{x})$ is itself separable, i.e.

$$\left(u_n^{(2)}\right)^{*n}(x,y) = \left(u_n(x)\cdot u_n(y)\right)^{*n}$$
$$= {u_n}^{\overset{x}{*}n}(x)\cdot {u_n}^{\overset{y}{*}n}(y)$$
$$= \left(\frac{n}{12\sigma^2}\right)^n {r_n}^{\overset{x}{*}n}(x)\cdot {r_n}^{\overset{y}{*}n}(y), \tag{7}$$

where the operators $\overset{x}{*}$ denote one-dimensional convolution in x-direction and $\overset{y}{*}$ one-dimensional convolution in y-direction, respectively. Substituting $p(\boldsymbol{x})$ in (6) with the r.h.s. of (7) we obtain

$$\left(\frac{n}{12\sigma^2}\right)^n {r_n}^{\overset{x}{*}n}(x)\cdot {r_n}^{\overset{y}{*}n}(y) \quad \xrightarrow[n\to\infty]{} \quad \frac{1}{2\pi\sigma^2}\,\mathrm{e}^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}\|^2},$$

or expressed as

Approximation 2. *For sufficiently large n, the n-fold self-convolution of the two-dimensional uniform probability density function $\frac{n}{12\sigma^2}\, r_n(x)\cdot r_n(y)$ approximates a bivariate Gaussian with mean vector $\mathbf{0}$ and covariance matrix $\sigma^2\boldsymbol{I}$, i.e.*

$$\left(\frac{n}{12\sigma^2}\right)^n {r_n}^{\overset{x}{*}n}(x)\cdot {r_n}^{\overset{y}{*}n}(y) \approx \frac{1}{2\pi\sigma^2}\,\mathrm{e}^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}\|^2},$$

where $r_n(x)$ is an elementary rectangular function defined as

$$r_n(x) = \begin{cases} 1 & \textit{for } |x| \le \sqrt{\frac{3}{n}}\,\sigma \\ 0 & \textit{otherwise} \end{cases} \quad \textit{for } n = 1,2,\cdots .$$

3 Barnes Interpolation as Series of Convolutions

Let $\varphi_{\boldsymbol{\mu},\sigma}(\boldsymbol{x})$ denote the PDF of a two-dimensional normal distribution with mean vector $\boldsymbol{\mu}$ and isotropic variance σ^2, i.e.

$$\varphi_{\boldsymbol{\mu},\sigma}(\boldsymbol{x}) = \frac{1}{2\pi\sigma^2}\,\mathrm{e}^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}-\boldsymbol{\mu}\|^2}.$$

Note that $\varphi_{\mathbf{0},\sigma}(\boldsymbol{x})$ corresponds to the r.h.s. of approximation 2. Further let $\delta_{\boldsymbol{a}}(\boldsymbol{x})$ denote the Dirac or unit impulse function at location $\boldsymbol{a}$ with the property $\delta_{\boldsymbol{a}} * f(\boldsymbol{x}) = f(\boldsymbol{x}-\boldsymbol{a})$. Then we can write

$$\varphi_{\boldsymbol{\mu},\sigma}(\boldsymbol{x}) = \frac{1}{2\pi\sigma^2}\,\mathrm{e}^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}-\boldsymbol{\mu}\|^2} = \delta_{\boldsymbol{\mu}} * \left(\frac{1}{2\pi\sigma^2}\,\mathrm{e}^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}\|^2}\right)$$
$$= \delta_{\boldsymbol{\mu}} * \varphi_{\mathbf{0},\sigma}(\boldsymbol{x}).$$

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions

Thus, a Gaussian weighted sum as found in the numerator of Barnes interpolation (1) for the Euclidean plane $\mathbb{R}^2$ can be written as convolutional operation

$$\sum_{k=1}^{N} f_k \cdot \mathrm{e}^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}-\boldsymbol{x_k}\|^2} = 2\pi\sigma^2 \sum_{k=1}^{N} f_k \cdot \varphi_{\boldsymbol{x_k},\sigma}(\boldsymbol{x})$$
$$= 2\pi\sigma^2 \sum_{k=1}^{N} f_k \cdot (\delta_{\boldsymbol{x_k}} * \varphi_{\boldsymbol{0},\sigma})(\boldsymbol{x})$$

and due to the distributivity and the associativity with scalar multiplication of the convolution operator follows

$$= 2\pi\sigma^2 \sum_{k=1}^{N} (f_k \cdot \delta_{\boldsymbol{x_k}}) * \varphi_{\boldsymbol{0},\sigma}(\boldsymbol{x})$$
$$= 2\pi\sigma^2 \left(\sum_{k=1}^{N} f_k \cdot \delta_{\boldsymbol{x_k}} \right) * \varphi_{\boldsymbol{0},\sigma}(\boldsymbol{x}).$$

Substituting $\varphi_{\boldsymbol{0},\sigma}(\boldsymbol{x})$ with approximation 2, we obtain for sufficiently large n

$$\sum_{k=1}^{N} f_k \cdot \mathrm{e}^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}-\boldsymbol{x_k}\|^2}$$
$$\approx 2\pi\sigma^2 \left(\frac{n}{12\sigma^2}\right)^n \left(\sum_{k=1}^{N} f_k \cdot \delta_{\boldsymbol{x_k}} \right) * \left({r_n}^{\overset{\mathrm{x}}{*}n}(x) \cdot {r_n}^{\overset{\mathrm{y}}{*}n}(y) \right).$$

For the denominator of Barnes interpolation (1) we can use the same expression, but set the coefficients f_k to 1. Since the common factors in the numerator and the denominator cancel each other, we can state

Approximation 3. *For sufficiently large n, Barnes interpolation for the Euclidean plane $\mathbb{R}^2$ can be approximated by*

$$f(x,y) \approx \frac{\left(\sum_{k=1}^{N} f_k \cdot \delta_{\boldsymbol{x_k}}\right) * \left({r_n}^{\overset{\mathrm{x}}{*}n}(x) \cdot {r_n}^{\overset{\mathrm{y}}{*}n}(y) \right)}{\left(\sum_{k=1}^{N} \delta_{\boldsymbol{x_k}}\right) * \left({r_n}^{\overset{\mathrm{x}}{*}n}(x) \cdot {r_n}^{\overset{\mathrm{y}}{*}n}(y) \right)}, \tag{8}$$

provided that the quotient is defined.

In other words, Barnes interpolation can very easily be approximated by the quotient of two convolutional expressions, both consisting of an irregularly spaced Dirac-comb, followed by a sequence of convolutions with a one-dimensional rectangular function of width $2\sigma\sqrt{3/n}$, executed n-times in x-direction and n-times in y-direction. As the convolution operation is commutative, the convolutions can basically be carried out in any order. The sequence shown in approximation 3, evaluated from left to right, is however especially favorable regarding the computational effort.

4 Discretization

Approximation 3 leads in a straightforward way to a very efficient algorithm for an approximate computation of Barnes interpolation on a regular grid Γ that is embedded in the Euclidean plane $\mathbb{R}^2$. Let

$$\Gamma = \{\, (i \cdot \Delta s, j \cdot \Delta s) \in \mathbb{R}^2 \mid 0 \le i < W, 0 \le j < H \,\},$$

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions
Open Access
EGU

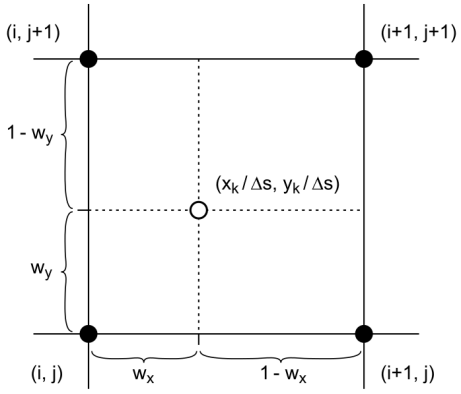

Figure 2. The nearer a sample point $\boldsymbol{x_k}$ is located to a grid point, the larger the weight assigned to it. Grid point (i, j) e.g. receives a weight of $(1 - w_x) \cdot (1 - w_y)$.

be a grid of dimension $W \times H$ with a grid point spacing Δs. Without loss of generality, we assume that all sample points $\boldsymbol{x_k}$ are sufficiently good contained in the interior of Γ. We now compute in an iterative procedure for each point (i, j) of grid Γ the convolutional expression that corresponds to the numerator (or denominator) of approximation 3. The intermediate fields that result from this iteration are denoted by $F^{(m)}$, where m indicates the iteration stage.

The first step consists in discretizing the expression $\sum f_k \cdot \delta_{\boldsymbol{x_k}}$, i.e. in injecting the values f_k at their respective sample points $\boldsymbol{x_k}$ into the underlying grid. For this purpose, the field $F^{(0)}$ is initialized with 0. From the definition of Dirac's impulse function in two dimensions

$$\delta_{\mathbf{0}}(x, y) = \lim_{\alpha \to 0} \frac{1}{\alpha^2} r_\alpha(x, y)$$

$$\text{where } r_\alpha(x, y) = \begin{cases} 1 & \text{for } |x|, |y| \leq \frac{1}{2\alpha} \\ 0 & \text{otherwise} \end{cases},$$

we deduce for its discrete version that the grid cell containing the considered sample point receives the weight $1/\Delta s^2$, while all other cells are left unchanged with weight 0. Since a sample point $\boldsymbol{x_k}$ in general does not coincide with a grid point (refer also to Fig. 2) and in order to achieve a good localization, the Dirac impulse $\delta_{\boldsymbol{x_k}}$ is distributed in a bilinear way to its four neighbouring grid points according to step 5 of algorithm 1.

Note that if a grid point (i, j) is affected by several sample points, the determined weight fractions are accumulated accordingly in the respective field element $F^{(0)}[i, j]$. For the final calculation of quotient (8), the factor $1 / \Delta s^2$ cancels out and can therefore be omitted in algorithm 1, but for reasons of mathematical correctness it is shown here. Since we have N input points and we perform a fixed number of operations for each of them, the complexity of algorithm 1 is given by $\mathcal{O}(N)$.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions
Open Access
EGU

Algorithm 1 Inject observation values f_k into grid Γ

Input: sample point coordinates $\boldsymbol{x_k} = (x_k, y_k)$ and sample values f_k for $k = 1, ..., N$.

Output: $W \times H$ field $F^{(0)}$ with injected sample values.

1: Initialize field $F^{(0)}$ with 0.

2: **for** $k = 1$ **to** N **do**

3: Determine indices $i = \lfloor x_k \,/\, \Delta s \rfloor$ and $j = \lfloor y_k \,/\, \Delta s \rfloor$ of lower left neighbouring grid point.

4: Compute weights $w_x = x_k \,/\, \Delta s - i$ and $w_y = y_k \,/\, \Delta s - j$, which are both contained in $[0, 1)$.

5: Distribute sample value f_k in bilinear way among four neighbouring grid points, i.e.

$F^{(0)}[i, j] \mathrel{+}= (1 - w_x) \cdot (1 - w_y) \cdot f_k \,/\, \Delta s^2$

$F^{(0)}[i, j+1] \mathrel{+}= (1 - w_x) \cdot w_y \cdot f_k \,/\, \Delta s^2$

$F^{(0)}[i+1, j] \mathrel{+}= w_x \cdot (1 - w_y) \cdot f_k \,/\, \Delta s^2$

$F^{(0)}[i+1, j+1] \mathrel{+}= w_x \cdot w_y \cdot f_k \,/\, \Delta s^2$

6: **end for**

7: **return** $F^{(0)}$.

The other algorithmic fragment we require, is the computation of a one-dimensional convolution with the rectangular function $r_n(x)$ as defined in (5). Employing the definition of convolution we obtain

$$h(x) = g * r_n(x) = \int_{-\infty}^{\infty} g(x-t) \cdot r_n(t) dt$$
$$= \int_{-\tau}^{\tau} g(x-t) dt = \int_{x-\tau}^{x+\tau} g(t) dt \tag{9}$$

where $\tau = \sigma\sqrt{3/n}$. With other words, the convolution $g * r_n$ at point x is simply the integral of $g(x)$ in the window $[x-\tau, x+\tau]$. Transferred to a one-dimensional grid with spacing Δs, the rectangular function $r_n(x)$ reads as rectangular pulse

$$r_T[k] = \begin{cases} 1 & \text{for } |k| \le T \\ 0 & \text{otherwise} \end{cases} \qquad k \in \mathbb{Z}\,,$$

with a width parameter T that is gained by rounding $\tau / \Delta s$ to the nearest integer number

$$T = \left\lfloor \frac{\tau}{\Delta s} + \frac{1}{2} \right\rfloor = \left\lfloor \sqrt{\frac{3}{n}} \frac{\sigma}{\Delta s} + \frac{1}{2} \right\rfloor . \tag{10}$$

Then equation (9) translates in the discrete case to

$$h[k] = g * r_T[k] \;=\; \sum_{i=-\infty}^{\infty} g[k-i] \cdot r_T[i] \cdot \Delta s$$
$$= \sum_{i=-T}^{T} g[k-i] \cdot \Delta s \;=\; \sum_{i=k-T}^{k+T} g[i] \cdot \Delta s.$$

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions

The element $h[k]$ corresponds to $h(k \cdot \Delta s)$ and $g[i]$ to $g(i \cdot \Delta s)$. Equivalently to the continuous case, the value $h[k]$ results up to a factor Δs from putting a window of length $2T+1$ centrally over the sequence element $g[k]$ and summing up all elements covered by it.

Assuming that we already computed $h[k-1]$, it is immediately clear that the following value $h[k]$ results from moving the window by one position to the right and thus can be obtained very easily from $h[k-1]$ by adding the newly enclosed sequence element $g[k+T]$, but subtracting element $g[k-T-1]$ that falls outside the window.

Algorithm 2 Convolution with rectangular pulse of length $2T+1$

Input: sequence $g[k]$ with $k = 1, ..., L$ and length $2T+1$ of rectangular pulse.

Output: the convolution $g * r_T$.

1: Compute $w = \sum_{i=-T}^{T} g[1-i]$, where all elements $g[k]$ which are not defined are set to 0.
2: Set $h[1] = w \cdot \Delta s$.
3: **for** $k = 2$ **to** L **do**
4: Update $w \mathrel{+}= g[k+T] - g[k-T-1]$, where all elements $g[k]$ which are not defined are set to 0.
5: Set $h[k] = w \cdot \Delta s$.
6: **end for**
7: **return** sequence $h[k]$ with $k = 1, ..., L$, which is the convolution $g * r_T$.

As in the case of algorithm 1 before, the factor Δs cancels in the final calculation of (8) and can therefore also be omitted here. Thus, algorithm 2 has $2T$ additions in step 1 and another $2(L-1)$ additions in the loop of step 3. Assuming that T is much smaller than L, an algorithmic complexity of $\mathcal{O}(L)$ results.

Now we are able to formulate algorithm 3 that computes convolutional expressions as found in the numerator and denominator of approximation 3.

Note that due to the commutativity of $\overset{x}{*}$ and $\overset{y}{*}$ the outer loop over index k can be moved inward within the loops over the rows and the columns, respectively. With this alternate loop layout, the field is first traversed row-wise in a single pass, where each row is in one sweep n-times convolved with r_T. After, the field is traversed column-wise and each column is n-times convolved. In such a way, more economic strategies with respect to memory access can be achieved and moreover, this loop order is very well suited for parallel execution, such that algorithm 3 can be computed very efficiently. Since in practice n is chosen constant (proven values for n lie between 3 and 6), the algorithmic complexity is $\mathcal{O}(W \cdot H)$.

Algorithms 1 and 3 now allow us to state the final algorithm 4, which implements the approximative computation of Barnes interpolation (8).

If the denominator $Q^{(n)}[i,j]$ in step 4 is 0, which is the case if the grid point (i,j) has at least in one dimension a greater distance than $2nT$ from the nearest sample point, the corresponding field value $F[i,j]$ is undefined and set to NaN.

Since algorithm 1 and 3 are invoked twice and step 8 employs another $W \cdot H$ divisions, the overall algorithmic complexity of the presented approach is limited to $\mathcal{O}(N + W \cdot H)$, which is a drastic improvement compared to the costs of $\mathcal{O}(N \cdot W \cdot H)$ of the naive implementation.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions
Open Access
EGU

Algorithm 3 n-fold convolution with a rectangular pulse in two dimensions

Input: $W \times H$ input field $F^{(0)}$, the length of the rectangular pulse $2T+1$ and the number of convolutions n to be carried out.

Output: n-fold convolved $W \times H$ field $F^{(n)}$, which is equal to $F^{(0)} * ({r_T}^{\overset{x}{*}n} \cdot {r_T}^{\overset{y}{*}n})$.

1: **for** $k = 1$ **to** n **do**
2: Rename the $(k-1)$-fold convolved field $F^{(k-1)}$ to F.
3: **for** $i = 1$ **to** W **do**
4: Convolve i-th field row $F[i,.]$ according to algorithm 2 with rectangular pulse r_T and store result back in respective field row.
5: **end for**
6: **for** $j = 1$ **to** H **do**
7: Convolve j-th field column $F[.,j]$ according to algorithm 2 with rectangular pulse r_T and store result back in respective field column.
8: **end for**
9: Rename F, which is now the k-fold convolved field, to $F^{(k)}$.
10: **end for**
11: **return** $F^{(n)}$.

Algorithm 4 Approximation of Barnes interpolation

Input: sample point coordinates $\boldsymbol{x_k} = (x_k, y_k)$ and sample values f_k for $k = 1, ..., N$, the number of iterations n and fall-off parameter σ.

Output: $W \times H$ field F that approximates Barnes interpolation.

1: Set $T = \left\lfloor \sqrt{\frac{3}{n}} \frac{\sigma}{\Delta s} + \frac{1}{2} \right\rfloor$.
2: To obtain n-fold convolved numerator field $P^{(n)}$ do:
3: Determine initial numerator field $P^{(0)}$ by invoking algorithm 1 and injecting sample values $\{f_k\}_{k=1}^{N}$.
4: Compute $P^{(n)}$ by invoking algorithm 3 with field $P^{(0)}$ and the rectangular pulse width $2T+1$.
5: To obtain n-fold convolved denominator field $Q^{(n)}$ do:
6: Determine initial denominator field $Q^{(0)}$ by invoking algorithm 1 and injecting constant sample values $\{1\}_{k=1}^{N}$.
7: Compute $Q^{(n)}$ by invoking algorithm 3 with field $Q^{(0)}$ and the rectangular pulse width $2T+1$.
8: Compute field F by dividing $P^{(n)}$ and $Q^{(n)}$ element-wise, i.e. by setting $F[i,j] = P^{(n)}[i,j] \,/\, Q^{(n)}[i,j]$.
9: **return** F.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022


Geoscientific Model Development Discussions

5 Results and Further Considerations

5.1 Test Setup

The described algorithm – in the further discourse denoted with "convolution" – was tested in an experimental setup that contained in total 3490 QFF values (air pressure reduced to mean sea level) obtained from measurements at meteorological stations distributed over Europe and dating from the 27 July 2020 at 12:00 UTC. More specifically, we considered the geographic area $D = [-26°E, 49°E] \times [34.5°N, 72°N] \subset \mathbb{R}^2$ equipped with the Euclidean distance function defined on $D \times D$, i.e. we measured distances in a first examination directly in the plate carrée projection neglecting the curved shape of the earth. The values of the QFF data range from 992.1 hPa to 1023.2 hPa.

The convolution interpolation algorithm is subsequently compared with the results of two alternate algorithms. The first of them – referred as "naive" – is given by the naive implementation as stated in the introduction. The second one – denoted with "radius" – consists of the improved naive algorithm that considers in its innermost loop only those observation points whose Gaussian weights w exceed 0.001 and thus are located within a radius of 3.717σ around the interpolation point. For this purpose, the implementation performs a so-called radius search on a k-d-tree, which contains all observation points. Such a radius search can be achieved with a worst case complexity of $\mathcal{O}(\sqrt{N})$.

All algorithms were implemented in Python using the Numba just-in-time compiler (Lam et al., 2015) in order to reduce the unsteadiness of purely interpreted Python code. The tests were conducted on a computer with a customary 2.6 GHz Intel i7-6600U processor with two cores, which is in fact only of minor importance since the tested code was written in single-threaded manner. All time measurements were performed ten times and the best value among them was set as the final execution time of the respective algorithm.

5.2 Visual Results

In general, Barnes' method yields a remarkable good interpolation and results in an aesthetic illustration for regions where the distance between the sample points has the same order of magnitude as the used Gaussian width parameter σ. However, if the distance between adjacent sample points is large compared to σ, this method exhibits some shortcomings because then the interpolation converges towards a step function with steep transitions. This effect can be clearly identified, for example, in the generation of plateaus of almost constant value over the Atlantic ocean in Fig. 3a. In the limit case, if $\sigma \to 0$, the interpolation produces a Voronoi tessellation with cells of constant value around a sample point that are bordered by discontinuities towards the neighbouring cells.

The comparison of the isoline visualizations in Figs. 3a and 3b shows in the well-defined areas an excellent agreement between the two approaches. The result for the radius algorithm is similarly consistent with the other two and is therefore not depicted.

Note that the shaded, i.e. the undefined areas of Fig. 3b correspond to those areas, where Barnes interpolation produces the plateau effect. In this sense one can state that the convolution algorithm filters out the problematic areas in an inherent way.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions
Open Access
EGU

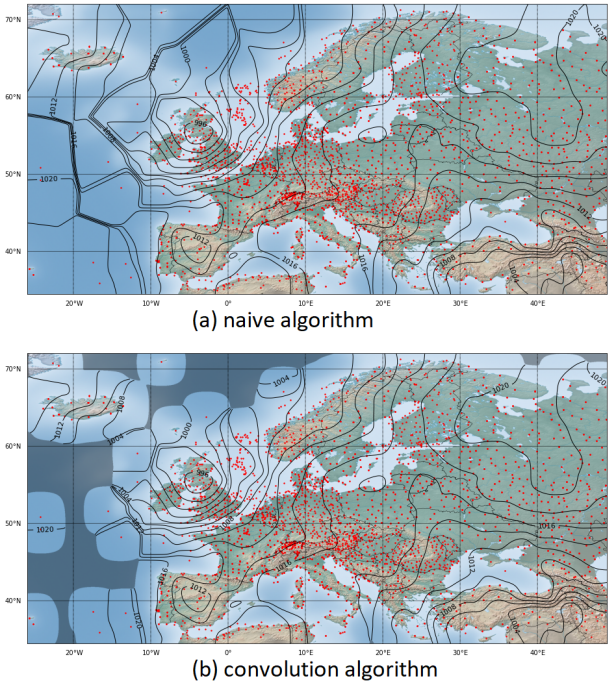

Figure 3. Image (a) shows exact Barnes interpolation with naive algorithm for 3490 sample points depicted in red, a 2400×1200 grid with a resolution 32 grid points$/^\circ$ and $\sigma = 1.0^\circ$.
Image (b) shows approximate Barnes interpolation with convolution algorithm for the same settings as for the naive algorithm above. The applied 4-fold convolution uses a rectangle mask of size 57. Areas where denominator of (8) drops below a value of 0.001 or is even 0 are rendered with a darker shade.

5.3 Time Measurements

For a grid of constant size, the measured execution times in Table 1 show for the naive and the radius algorithm a linear dependence to the number N of considered sample points, while they are almost constant for the presented convolution algorithm. The costs of the injection step are obviously more or less inexistent compared to the costs of the subsequent steps of the algorithm. This fact is in entire agreement to the deduced complexity $\mathcal{O}(N + W \cdot H)$, since in our test setup the grid size $W \times H$ clearly dominates over N.

Note also that the speed-up factor between the naive and the convolution algorithm ranges for the considered number of sample points roughly between 25 and 1000.

If the grid size is varied, all considered algorithms reveal a linear-like dependence to the number of points in the grid as can be seen in Table 2 and Fig. 5. For smaller grid sizes the convolution algorithm provides a speed-up factor of around 2000 compared to the naive implementation, but for bigger grids the factor drops below 1000.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions
Open Access
EGU

Number of Sample Points	Algorithm		
	Naive	Radius	Convol.
54	6.198	0.961	0.247
218	21.558	1.776	0.248
872	78.407	4.097	0.245
3490	280.764	11.840	0.247

Table 1. Execution times (in s) of the investigated algorithms for varying numbers of sample points. The grid size of 2400×1200 points with a resolution of 32 points$/^\circ$ and the Gaussian width $\sigma = 1.0^\circ$ are kept constant. The convolution algorithm applied a 4-fold convolution.

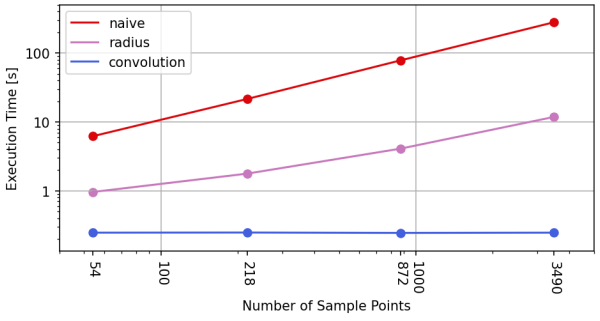

Figure 4. Plot of execution times from Table 1 against number of sample points. Both axis use a logarithmic scale.

This effect can be explained by the fact that the crucial parts of the convolution algorithm access memory for smaller grids with a high spatial and temporal locality and thus making optimal use of the highly efficient CPU cache memory (Patterson and Hennessy, 2014). For bigger grids the number of cache misses increases, which result in a slightly degraded performance.

Grid Size	Resol.	Algorithm		
		Naive	Radius	Convol.
300×150	4 pt$/^\circ$	4.415	0.203	0.002
600×300	8 pt$/^\circ$	17.626	0.782	0.011
1200×600	16 pt$/^\circ$	70.871	3.031	0.047
2400×1200	32 pt$/^\circ$	283.735	11.881	0.247
4800×2400	64 pt$/^\circ$	1134.265	47.044	1.261

Table 2. Execution times (in s) of the investigated algorithms for varying grid sizes and resolutions, respectively. The number of sample points $N = 3490$ and the Gaussian width $\sigma = 1.0^\circ$ are kept constant. The convolution algorithm applied a 4-fold convolution.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions

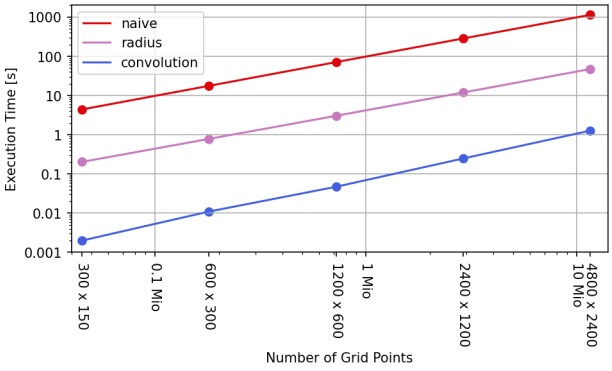

Figure 5. Plot of execution times from Table 2 against number of grid points. Both axis use a logarithmic scale.

As to be expected, the Gaussian width parameter σ has no decisive impact on the execution times measured for the naive and the convolution algorithm (refer to Table 3 and Fig. 6). The radius algorithm, on the other hand, shows a quadratic dependence, since the relevant area around a grid point – and thus also the average number of sample points to be considered – grows quadratically with the radius of influence, which in turn depends linearly on σ.

The unstable time measurements for the naive algorithm are caused by a peculiar execution time behavior of the exponential function. Although for distances $d > 38.6\sigma$, the value of (2) is less than the smallest representable float and therefore results in 0, its computational cost increases in Python to a multiple of the time needed for shorter distances. With growing σ, this computation overhead is less and less noticeable in the test series.

Gaussian Width σ	**Algorithm**		
	Naive	Radius	Convol.
0.25°	652.025	2.376	0.247
0.5°	560.094	4.579	0.246
1.0°	280.477	11.864	0.245
2.0°	126.241	37.244	0.244
4.0°	125.848	122.956	0.245

Table 3. Execution times (in s) of the investigated algorithms for varying Gaussian width parameters σ. The number of sample points $N = 3490$ and the grid size of 2400×1200 points with a resolution of 32 points$/^\circ$ are kept constant. The convolution algorithm applied a 4-fold convolution.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions

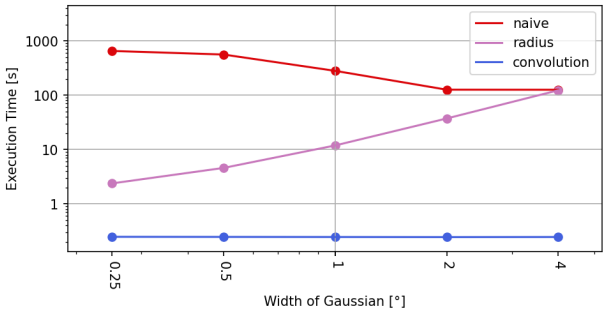

Figure 6. Plot of execution times from Table 3 against Gaussian width parameter. Both axis use a logarithmic scale.

5.4 Algorithm Fine Tuning

In the discretization step, we round the window width parameter T to the nearest integer, i.e. we set

$$T = \left\lfloor \sqrt{\frac{3}{n}} \frac{\sigma}{\Delta s} + \frac{1}{2} \right\rfloor \tag{10 recap}$$

and then repeatedly convolve the input fields with a rectangular pulse signal $r_T[k]$ that has a length $2T+1$. In the extreme, if n exceeds $12\sigma^2/\Delta s^2$ and hence $T = 0$, the underlying pulse signal degrades to $r_0[k]$, which is identical to the discrete unit pulse, the neutral element with respect to convolution. Under these circumstances, the algorithm stops to render a meaningful interpolation. Since normally $\sigma > \Delta s$, the respective bound for n can go into the hundreds, which is why the described extreme case is typically not encountered for real applications.

Nevertheless, a general consequence of the rounding operation is that the effectively obtained Gaussian width σ_{eff} for our algorithm only approximates the desired width σ. Let for the following considerations $u_T[k]$ be the uniform probability distribution that corresponds to $r_T[k]$, i.e.

$$u_T[k] = \frac{1}{2T+1} r_T[k].$$

By algorithmic design, ${\sigma_{\text{eff}}}^2$ is the variance of $u_T[k]$, which is n-times convolved with itself and which is employed on a grid with point spacing Δs. Therefore is

$$\begin{aligned} {\sigma_{\text{eff}}}^2 &= n \operatorname{Var}(u_T) = n \sum_{k=-T}^{T} \frac{1}{2T+1} (k\Delta s)^2 \\ &= \frac{2n\Delta s^2}{2T+1} \sum_{k=1}^{T} k^2 = \frac{n}{3} T(T+1) \Delta s^2. \end{aligned} \tag{11}$$

For $T \geq 1$, the integer number T can be fixed by (10) to the unit interval

$$\sqrt{\frac{3}{n}} \frac{\sigma}{\Delta s} - \frac{1}{2} < T \leq \sqrt{\frac{3}{n}} \frac{\sigma}{\Delta s} + \frac{1}{2},$$

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022


Geoscientific Model Development Discussions

which in turn allows, when substituted into (11), to derive sharp bounds for σ_{eff}

$$\sigma^2 - \frac{n}{12}\,\Delta s^2 \;<\; {\sigma_{\text{eff}}}^2 \;\le\; \sigma^2 + 2\sqrt{\frac{n}{3}}\,\sigma\,\Delta s + \frac{n}{4}\,\Delta s^2.$$

The last relation reveals that the range of possible values for σ_{eff} grows with increasing n until σ_{eff} collapses to 0 when $n > 12\sigma^2/\Delta s^2$, as we have seen further above.

Summarized can be stated that two diametrically opposed effects determine the result quality of the presented algorithm. From the perspective of the central limit theorem, the approximation performance is better for large n, while on the other hand σ_{eff} tends to be closer to the target σ for small n.

If full accuracy is required for the Gaussian width and the grid spacing Δs is not strictly given in advance, one can tune the spacing such that the resulting width σ_{eff} corresponds exactly to the desired σ. For $T \ge 1$, equation (11) allows then to determine the necessary grid step as

$$\Delta s = \sqrt{\frac{3}{nT(T+1)}}\;\sigma.$$

If however the grid is fixed a priori and cannot be modified, the only option to attain the requested σ exactly is to refrain from using a stringent rectangular pulse and to employ a slightly more complicated signal. For this purpose, we set this time

$$T = \left\lfloor \frac{1}{2}\left(\sqrt{1 + \frac{12}{n}\frac{\sigma^2}{\Delta s^2}} - 1\right)\right\rfloor,$$

which is gained from taking the positive solution of the quadratic equation (11) for T. Now we have $n\,\text{Var}(u_T) \le \sigma^2 < n\,\text{Var}(u_{T+1})$ and we define the linearly blended signal

$$\begin{aligned} r_{T,\alpha}[k] &= (1-\alpha)\,r_T[k] + \alpha\,r_{T+1}[k] \\ &= \begin{cases} 1 & \text{for } |k| \le T \\ \alpha & \text{for } |k| = T+1 \\ 0 & \text{otherwise} \end{cases} \qquad k \in \mathbb{Z}\,, \end{aligned} \tag{12}$$

for $0 \le \alpha < 1$. This modified signal $r_{T,\alpha}[k]$ is basically the pure rectangular signal $r_T[k]$ of unit elements with a trailing element α appended at both ends. Due to the continuity of the variance, there must be a specific $\tilde{\alpha}$ for which $n\,\text{Var}(u_{T,\tilde{\alpha}}) = \sigma^2$, whereas $u_{T,\alpha}[k] = \frac{1}{2(T+\alpha)+1}\,r_{T,\alpha}[k]$ designates the probability distribution of $r_{T,\alpha}[k]$. With

$$\begin{aligned} \text{Var}(u_{T,\alpha}) &= \frac{1}{2(T+\alpha)+1}\sum_{k=-T-1}^{T+1} r_{T,\alpha}[k]\,(k\,\Delta s)^2 \\ &= \frac{\Delta s^2}{2(T+\alpha)+1}\left(2\alpha(T+1)^2 + 2\sum_{k=1}^{T} k^2\right) \\ &= \frac{\Delta s^2}{2(T+\alpha)+1}\left(2\alpha(T+1)^2 + \frac{1}{3}T(T+1)(2T+1)\right) \end{aligned}$$

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions

	Convolution with $r_T[k]$				Convolution with $r_{T,\tilde{\alpha}}[k]$			
n	T	σ_{eff}	RMSE	t_{exec}	T	$\tilde{\alpha}$	RMSE	t_{exec}
1	55	1.0013	0.3557	0.161	54	0.9260	0.3551	0.175
2	39	1.0078	0.1334	0.188	38	0.6868	0.1327	0.216
3	32	1.0155	0.0628	0.216	31	0.4922	0.0606	0.257
4	28	1.0282	0.0492	0.243	27	0.2083	0.0367	0.298
5	25	1.0286	0.0431	0.270	24	0.2799	0.0266	0.337
6	23	1.0383	0.0496	0.297	22	0.1256	0.0213	0.378
7	21	1.0260	0.0356	0.324	20	0.4372	0.0178	0.419
8	20	1.0458	0.0549	0.351	19	0.0956	0.0154	0.459
9	18	1.0010	0.0137	0.378	17	0.9804	0.0136	0.500
10	18	1.0551	0.0639	0.405	17	0.0316	0.0121	0.539
20	12	1.0078	0.0111	0.676	11	0.8922	0.0059	0.943
50	8	1.0825	0.0916	1.488	7	0.3125	0.0024	2.159

Table 4. Key numbers of the original and the optimized convolution algorithm in dependency of the number of performed convolutions n, where $N = 3490$, grid size 2400×1200, resolution 32 points$/^\circ$ and $\sigma = 1.0^\circ$. The root mean square error RMSE is computed for the sub-area displayed in Fig. 9 and with respect to the exact results of the naive algorithm. The execution times t_{exec} are measured in seconds.

we conclude that the wanted $\tilde{\alpha}$ is given by

$$\tilde{\alpha} = \frac{(2T+1)\left(\sigma^2 - \frac{1}{3}T(T+1)\,n\,\Delta s^2\right)}{2\left((T+1)^2\,n\,\Delta s^2 - \sigma^2\right)}. \tag{13}$$

Remember that the central limit theorem is valid for any PDF, which means that the mathematical framework presented in the previous chapters is also valid for the modified signal $r_{T,\tilde{\alpha}}[k]$. Therefore we receive an optimized approximative Barnes interpolation by marginally adapting algorithm 2 to compute the convolution with $r_{T,\tilde{\alpha}}[k]$ instead of $r_T[k]$. To do so, steps 2. and 5. of it have to be rewritten to $h[k] = (w + \tilde{\alpha} \cdot (g[k+T+1] + g[k-T-1])) \cdot \Delta s$. Although the optimized approach requires $2L$ additions and L multiplications more than the original one, the adapted algorithm 2 remains in the complexity class $\mathcal{O}(L)$. Measurements (refer to Table 4 and Fig. 7) show that the optimized interpolation in fact needs only about 10% to 30% more time for the depicted range of convolution rounds than the original one.

The unstable behavior of the original convolution algorithm with respect to the number of performed convolutions can be best observed in the RMSE plot of Fig. 8, where the baseline is given by the exact interpolation of the naive algorithm. This finds also its visual correspondence in the upper row of the maps shown in Fig. 9a, more precisely e.g. in the fluctuating diameter of the small high pressure area west of the Balearic Islands.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions

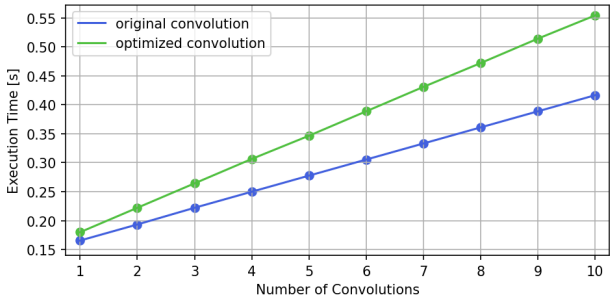

Figure 7. Plot of execution times from Table 4 against number of performed convolutions for both, the original and the optimized convolution algorithm.

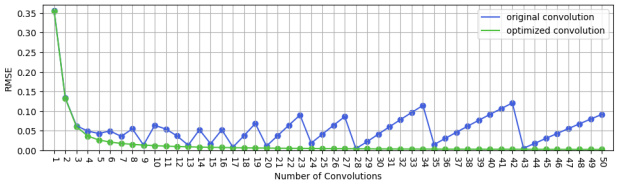

Figure 8. Root mean square error with respect to the exact Barnes interpolation in dependency of the number of performed convolutions for both, the original and the optimized convolution algorithm.

In contrast to that, the optimal convolution algorithm shows a stable convergence towards the exact interpolation obtained by the naive algorithm, which manifests in strictly monotonic decreasing RMSE values. Moreover, these results suggest that 3 or 4 performed convolution rounds achieve already a very good approximation of Barnes interpolation.

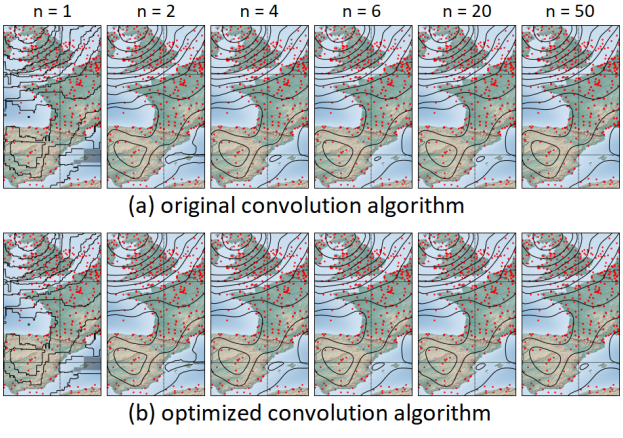

Figure 9. Results for different number of performed convolutions n. Upper row (a) with results for original convolution algorithm and lower row (b) with results for optimized convolution algorithm.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions

5.5 Application on Sphere Geometry

So far we applied the convolution algorithm on sample points contained in the plane $\mathbb{R}^2$ and using the Euclidean distance measure. For geospatial applications this simplification is acceptable as long as the considered area is sufficiently small enough. If we deal with data, which is distributed over a larger region – as it is actually the case in our test setup – it becomes necessary to take the curvature of the earth into account.

The adaptation of the naive Barnes interpolation algorithm to the spherical geometry on $\mathcal{S}^2$ consists merely in the exchange of the Euclidean distance with the spherical distance. Since the calculation of the spherical distance between two points involves several trigonometric function calls, the price of such a switchover is accordingly high and consequently exact Barnes interpolation for a spherical geometry is in our tests roughly a factor of 2.5 times slower as that with the Euclidean approach. In other words, an already effortful algorithm becomes even more effortful.

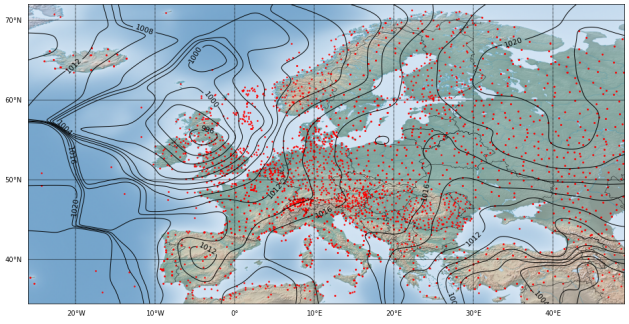

Figure 10. Exact Barnes interpolation with naive algorithm for the same setup as for Fig. 3 but using spherical distances from $\mathcal{S}^2$ instead of Euclidean distances from $\mathbb{R}^2$. The generated isolines are notably different in the northern part, while they are quite similar in the south.

However, the distance calculation does not occur explicitly in the convolution algorithm, since the latter by virtue of the central limit theorem is inherently tied to the Euclidean distance measure. Therefore, the convolution algorithm has to be transferred to the spherical geometry with a different method.

The chosen approach is to first map the sample points to a map projection, which distorts the region of interest as minimal as possible and then to apply the convolution algorithm directly in that map space. The resulting field is finally mapped back to original map projection and provides there an approximation of Barnes interpolation with respect to the spherical geometry of $\mathcal{S}^2$.

Projection types that are considered suitable for this purpose are conformal map projections. Conformal maps preserve angles and shapes locally, while distances and areas underlie a certain distortion. Often used conformal maps are (Snyder, 1987)

- Mercator projection for regions of interest that stretch in east-west direction around the equator,
- transverse Mercator projection for regions with north-south orientation,

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions

- Lambert conformal conic projection for regions in the mid-latitudes that stretch in east-west direction or
- polar stereographic projection for polar regions.

In order to replicate Fig. 10 with our fast optimized convolution algorithm, we therefore use for our test setup with sample points in the mid-latitudes a Lambert conformal conic map projection. We choose the two standard parallels that define the exact projection at latitudes of $42.5°N$ and $65.5°N$, such that our region of interest is evenly captured by them. By nature of Lambert conformal conic maps, the chosen map scale is exactly adopted along these two latitudes, while it is too small between them and too large beyond them. Similarly, for a grid with a formal resolution of 32 grid points$/°$ that is embedded into this map, the specified resolution applies only exactly along the standard parallels, while the effective resolution between them is smaller and beyond them larger.

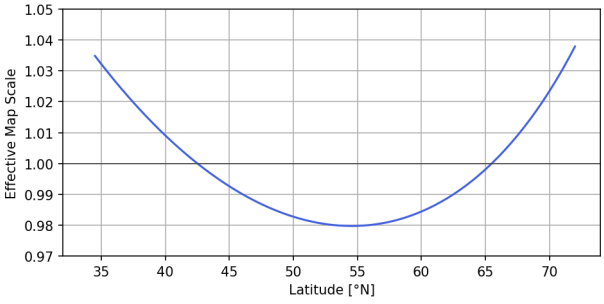

Figure 11. The effective scale of a Lambert conformal conic map in dependency of the latitude, if the scale at $42.5°N$ and $65.5°N$ is set to 1.0. The minimum scale of 0.98 is reached at a latitude of $54.5°N$.

We now employ the optimized convolution algorithm with a nominal Gaussian width parameter $\sigma = 1.0°$ on the Lambert conformal conic map grid postulated above, in which we injected the given sample points beforehand. The resulting field shown in Fig. 12 thereby experiences a twofold approximation, the first one caused by the distortion of the map and the second one due to the approximation property of the convolution algorithm.

In a last step, the result field is mapped back to the target map, which uses in our case a plate carrée projection. In order to do this, the location of a target field element is looked up in the Lambert map source field and subsequently its element value is determined by bilinear interpolation of the four neighbouring source field element values. This last operation performs an averaging and thus adds a further small error to the final result shown in Fig. 13. Similar to the case for the Euclidean distance, the comparison with the exact Barnes interpolation on $\mathcal{S}^2$ in Fig. 10 yields a very good correspondence. This perception is also supported by measurement of the RMSE, which adds up for the same sub-area as investigated in Table 4 to 0.0467, which is negligible larger than the corresponding 0.0367 measured for the Euclidean case.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions
Open Access
EGU

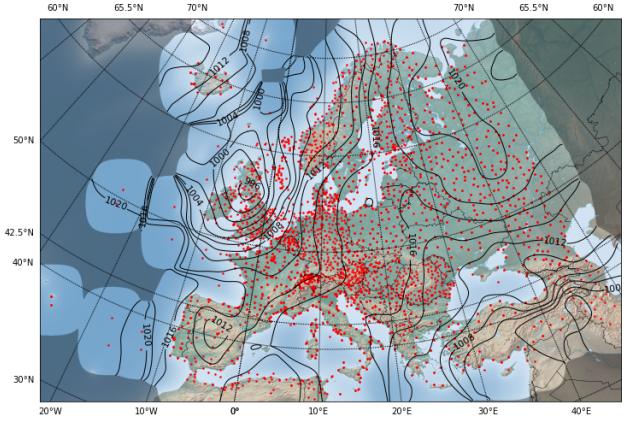

Figure 12. Optimized Barnes interpolation algorithm applied on Lambert conformal map on a 2048×1408 grid with a resolution 32 grid points$/^\circ$ along the standard parallels at $42.5^\circ N$ and $65.5^\circ N$, $\sigma = 1.0^\circ$ and a 4-fold convolution.

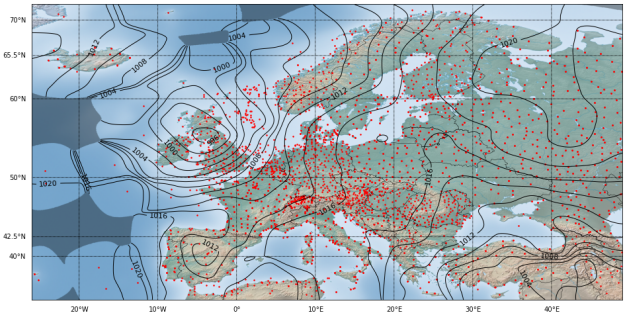

Figure 13. Resulting field from Fig. 12 projected back to a plate carrée map.

5.6 Round-off Error Issues

Computing the convolution of a rectangular pulse in floating point arithmetic using the moving window technique as described in algorithm 2 of chapter 4 is extremely efficient, but is also prone to imprecisions since round-off errors are steadily accumulated during the progress. Different approaches are known to reduce this error. The Kahan summation algorithm (Kahan, 1965), for instance, implements an intelligent error compensation scheme, at the expense of requiring essentially more basic operations than used for ordinary addition.

Another, in the context of Barnes interpolation very effective error reduction technique, is to center the numbers to be added around 0.0, where the mesh density of representable floating point numbers is highest. For this purpose, an offset

$$\bar{f} = \frac{1}{2}\left(\min_{1 \le k \le N} f_k + \max_{1 \le k \le N} f_k\right)$$

is initially subtracted from the sample values f_k, such that their range is exactly located around 0.0. The presented convolution algorithm is then applied to the shifted sample values. In a final step, the elements of the resulting field $F[i, j]$ are shifted back

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions
Open Access
EGU

Target Map Grid Size	Resol.	$\mathcal{S}^2$ Algorithm		Split Times		Lambert Map Grid Size
		Naive	Convol.	Actual Convol.	Back Proj.	
300×150	4 pt/°	10.792	0.005	0.004	0.001	256×176
600×300	8 pt/°	42.987	0.020	0.016	0.004	512×352
1200×600	16 pt/°	174.081	0.089	0.070	0.019	1024×704
2400×1200	32 pt/°	700.089	0.408	0.326	0.082	2048×1408
4800×2400	64 pt/°	2802.574	1.828	1.493	0.335	4096×2816

Table 5. Execution times (in s) for the naive and the optimized convolution algorithms using spherical distances on $\mathcal{S}^2$ for varying grid sizes. The number of sample points $N = 3490$ and the Gaussian width $\sigma = 1.0°$ are kept constant. The convolution algorithm applied a 4-fold convolution and was executed on a Lambert map grid of the indicated size. The two split time columns show the separated execution times for the actual convolution and the subsequent back projection to the plate carrée map. For the investigated scenarios, the optimized convolution algorithm is more than 1000 times faster than the naive one.

to the original range by adding $\bar{f}$ to each of them. This modification needs basically $N + W \cdot H$ extra additions, such that the computational complexity of the convolution algorithm is not harmed and stays unchanged.

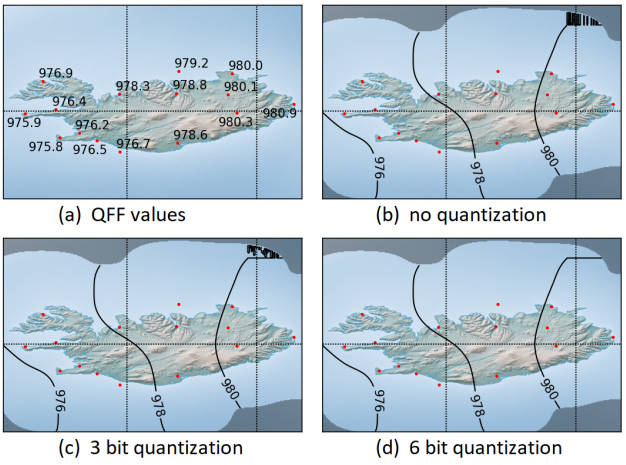

Figure 14. Constellation of QFF values over Iceland with generated isolines for $\sigma = 0.75$. Note the 980.0 hPa observation value in the northeast of Iceland, which triggers the creation of the faulty isoline for the same value. By increasing the quantization degree of the resulting field values, the visual appearance is gradually improved.

Minimal numerical round-off errors can generate surprisingly prominent artefacts when areas of constant or near-constant data are rendered with an isoline visualization. In such cases, the value obtained by the convolution algorithm fluctuates

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

seemingly randomly around the true constant value, and if it happens that one of the rendered isolines represents exactly this value, the visualized result may appear like a fractal curve as shown in Fig. 14. A counter measure against this effect is to apply a quantization scheme to the resulting values, which basically suppresses a suitable number of least significant bits and rounds the value to the nearest eligible floating point number. After this operation the obtained values in areas of constant data are actually constant and thus result in a more pleasant isoline visualization. In areas with varying data values, the quantization of the result data has no harmful impact.

6 Conclusions

We presented a new technique for computing very good approximations of Barnes interpolation, which promise speed-up factors for realistic scenarios that can reach into the hundreds or even into the thousands when applied on a spherical geometry. The underlying convolutional algorithm exhibits a computational complexity of $\mathcal{O}(N+W\cdot H)$, in which the number of sample points N is decoupled from the grid size $W \times H$.

The usage of the algorithm is not restricted to $\mathbb{R}^2$ or $\mathcal{S}^2$ and it can be easily extended to higher dimensional spaces $\mathbb{R}^n$. The algorithm allows to incorporate quality measures that assign each sample point $\boldsymbol{x_k}$ a weight of certainty c_k, which specifies how much this point shall contribute to the overall result. To achieve this, the terms in the two sums in (1) are simply multiplied by the additional factor c_k, and likewise the same factors then also appear in approximation 3.

Barnes interpolation is often used in the context of successive correction methods (Cressman, 1959; Bratseth, 1986) with or without a first guess from a background field. In this technique, the interpolation is not performed just once, but applied several times with decreasing Gaussian width parameters to the residual errors in order to minimize them successively. Needless to say, that instead of exact Barnes interpolation, the convolutional algorithm can equally be used for the method of successive correction.

Since the presented solution for spherical geometries is only suitable for the treatment of local maps, we plan in a next step to generalize the approach to global maps. This could for instance be done by smoothly merging local Barnes interpolation approximation patches into a global approximation.

Furthermore, we also want to provide a statement about the quality of the calculated approximation depending on the number of performed convolutions by deriving a theoretical upper bound for the maximum possible error.

Code and data availability. The formal algorithms introduced in this paper are provided as Python implementation on GitHub https://github.com/MeteoSwiss/fast-barnes-py under the terms of the BSD 3-clause license and are archived on Zenodo https://doi.org/10.5281/zenodo.6792664. There are also the sample dataset and the scripts included, which allow to reproduce the figures and tables presented earlier.

Competing interests. The author declares that he has no conflict of interest.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Acknowledgements. All map backgrounds were made with Natural Earth, which provides free vector and raster map data at naturalearth-data.com.

https://doi.org/10.5194/gmd-2022-116
Preprint. Discussion started: 19 July 2022

Geoscientific Model Development Discussions

References

Barnes, S. L.: A Technique for Maximizing Details in Numerical Weather Map Analysis, J. Appl. Meteorol., 3, 396 – 409, 1964.

Bentley, J. L.: Multidimensional Binary Search Trees Used for Associative Searching, Commun. ACM, 18, 509 – 517, 1975.

Bergthórsson, P. and Döös, B. R.: Numerical Weather Map Analysis, Tellus, 7, 329 – 340, 1955.

Bratseth, A. M.: Statistical interpolation by means of successive corrections, Tellus A: Dynamic Meteorology and Oceanography, 38, 439 – 447, 1986.

Cressman, G. P.: An Operational Objective Analysis System, Mon. Weather Rev., 87, 367 – 374, 1959.

Daley, R.: Atmospheric Data Analysis, Cambridge University Press, Cambridge, 1991.

van Dijk, G.: Distribution Theory: Convolution, Fourier Transform, and Laplace Transform, De Gruyter Graduate Lectures, De Gruyter Berlin, 2013.

Finkel, R. A. and Bentley, J. L.: Quad Trees, a Data Structure for Retrieval on Composite Keys, Acta Inform., 4, 1 – 9, 1974.

Kahan, W.: Further remarks on reducing truncation errors, Commun. ACM, 8, 40, 1965.

Klenke, A.: Probability Theory: a Comprehensive Course, Third Edition, Universitext, Springer, 2020.

Koch, S. E., desJardins, M., and Kocin, P. J.: An Interactive Barnes Objective Map Analysis Scheme for Use with Satellite and Conventional Data, J. Clim. Appl. Meteorol., 22, 1487 – 1503, 1983.

Koppert, H. J., Pedersen, T. S., Zürcher, B., and Joe, P.: How to make an international meteorological workstation project successful, Twentieth International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology, 84th AMS Annual Meeting, Seattle, WA, 11.1, 2004.

Lam, S. K., Pitrou, A., and Seibert, S.: Numba: A LLVM-based Python JIT Compiler, LLVM '15: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, 1 – 6, 2015.

Muirhead, R. J.: Aspects of Multivariate Statistical Theory, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, Inc., 1982.

Patterson, D. A. and Hennessy, J. L.: Computer Organization and Design: the Hardware/Software Interface, 5th edition, Elsevier, 2014.

Snyder, J. P.: Map Projections: A Working Manual, US Geological Survey Professional Paper 1395, US Government Printing Office, Washington, 1987.