Fast Poisson solvers on nonequispaced grids: Multigrid and Fourier Methods compared

Gisela Pöplau[a*] and Daniel Potts[b]

[a]Institute of General Electrical Engineering, Rostock University, D–18051 Rostock, Germany
[b]Institute of Mathematics, University of Lübeck, D–23560 Lübeck, Germany

ABSTRACT

The solution of partial differential equations on adaptively generated grids play an important role in scientific computation. In this paper we compare two Poisson solvers for data on nonequispaced mesh points. A new meshless Fourier method based on NFFT is constructed in $\mathbb{R}^3$. This algorithm is compared to the well-established multigrid method working on nonequidistant meshes. Our investigations are motivated especially by simulations of the behaviour of charged particles in accelerators.

Keywords: Poisson solver, nonequispaced grid, multigrid method, Fourier method, meshless method, FFT, NFFT

1. INTRODUCTION

The development of efficient solvers for the numerical solution of Poisson's equations, for instance for the simulation of electrostatic fields, is still an important field of research, because problems get more and more demanding with respect to computing time. Further, the appropriate modelling of practical problems often requires adaptive discretisations or meshless methods.

While the efficient solution of Poisson's equation is an important topic in various fields our paper is motivated by recent developments in the simulations of the behaviour of charged particles in accelerators. Here, the fast calculation of space-charge effects as full 3D simulation becomes more and more important. The full 3D treatment is particularly challenging because the bunches typically have varying shape during their path through the accelerator ranging from very short to very long. Thus, an appropriate approximation of the space-charge density ρ which is determined by the distribution of the charged particles in a bunch requires a nonequidistant mesh for such shapes. The Poisson problem for this application reads as

$$\begin{aligned} -\Delta u &= \rho/\varepsilon_0 \quad \text{in} \quad \Omega \subset \mathbb{R}^3 \\ u &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{1.1}$$

where u denotes the electrostatic potential and ε_0 the dielectric constant of the vacuum.

In this paper we describe two different Poisson solvers working on nonequidistant grids. The first method newly introduced in section 2.1 is based on the recently developed fast Fourier transform for nonequispaced knots (NFFT), see[1] and the references therein. The construction of this new solver was inspired by an approach to fast direct solvers for equispaced problems using d-variate periodic interpolation in shift-invariant spaces which was given by Pöplau.[2] This idea provides error estimates and the construction of numerical algorithms based on FFT in a very convenient way. In order to overcome the equispaced grid in[2] we propose a new NFFT based algorithm. A software package for the NFFT can be found in.[3]

The second method given in section 2.2 uses the discretisation of the Laplacian by second order finite differences. A multigrid algorithm with a coarsening technique adapted to nonequidistant grids is applied for the solution of the resulting linear system of equations. The adaptive coarsening first introduced in[4] and further developed in[5] ensures that the same convergence rate is achieved for nonequidistant grids as well as for equidistant grids.[17]

*Supported by DESY, Hamburg
http://www-ae.e-technik.uni-rostock.de/home/poeplau/poeplau.html
E-mail: gisela.poeplau@etechnik.uni-rostock.de
http://www.math.uni-luebeck.de/potts
E-mail: potts@math.uni-luebeck.de

The numerical investigations in section 3 compare the two Poisson solvers. For equidistant meshes the behaviour of both, multigrid for finite difference discretisations and FFT based Poisson solvers is well known: The iterative multigrid method requires $\mathcal{O}(N)$ operations for N grid points, while FFT method as direct Poisson solver requires $\mathcal{O}(N \log N)$ operations on equispaced grids. Hence, the numerical effort is comparable as long as the number of grid points is not too large, that is if the $\log N$-term is in the range of mC assuming $\mathcal{O}(N) = mCN$. Here, m refers to as the number of multigrid iterations and C the number of operations necessary within every iteration step which depends on the choice of the multigrid components.[6]

Although the multigrid algorithm turns out to be one order of magnitude faster in CPU time than the NFFT Poisson solver, the use of a mesh for the discretisation include some open problems. These are, for instance, the choice of the number of mesh lines and the optimal distribution of mesh lines.[19] The advantage of the NFFT method is that it can be applied to scattered data. The here developed meshless method never discretise the Laplacian. The operator is instead applied directly to the exponentials.

2. SOLUTION ON NONEQUISPACED GRIDS

2.1. Fourier Method

First we introduce some notations. For fixed $N \in \mathbb{N}$ let I_N^d denote the set

$$I_N^d := \{\boldsymbol{k} = (k_1, \ldots, k_d)^{\mathrm{T}} \in \mathbb{Z}^d : -\frac{N}{2} \leq k_j < \frac{N}{2}; j = 1, \ldots, d\} .$$

The three-variate torus $\mathcal{T}^3$ is represented by the cube

$$\mathcal{T}^3 := \{\boldsymbol{x} = (x_1, x_2, x_3)^{\mathrm{T}} \in \mathbb{R}^3 : |x_j| \leq 1/2; j = 1, 2, 3\} ,$$

where the opposite sides of $\mathcal{T}^3$ are identified. Furthermore, let $W_2^s(\mathcal{T}^3)$ be the periodic Sobolev space of order $s \in \mathbb{R}$. This space is equipped with the norm

$$\|f\|_{s,2} := \left(\sum_{\boldsymbol{k} \in \mathbb{Z}^3} \left(1 + \|2\pi\boldsymbol{k}\|_2^2\right)^s |c_{\boldsymbol{k}}(f)|^2 \right)^{1/2} ,$$

where the Fourier coefficients are given by

$$c_{\boldsymbol{k}}(f) := \int_{\mathcal{T}^3} f(\boldsymbol{t})\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{t}}\, \mathrm{d}\boldsymbol{t} \quad (\boldsymbol{k} \in \mathbb{Z}^3) .$$

For $s = 0$ we have $W_2^0 = L_2(\mathcal{T}^3)$. In the following, we consider the problem to find $u \in W_2^s(\mathcal{T}^3)$ which satisfies the differential equation

$$-\Delta u = f \tag{2.1}$$

for given $f \in W_2^{s-2}(\mathcal{T}^3)$. In particular we have periodic boundary conditions.

For a 1–periodic three-variate integrable function $f : \mathbb{R}^3 \to \mathbb{R}$, we write

$$\mathcal{I}(f, K_N, \boldsymbol{v}) = \int_{\mathcal{T}^3} f(\boldsymbol{t}) K_N(\boldsymbol{v} - \boldsymbol{t})\, \mathrm{d}\boldsymbol{t} = \int_{\mathcal{T}^3} f(\boldsymbol{v} - \boldsymbol{t}) K_N(\boldsymbol{t})\, \mathrm{d}\boldsymbol{t} .$$

We are interested in kernels of the form

$$K_N(\boldsymbol{t}) := \sum_{\boldsymbol{k} \in I_N^3} \omega_{\boldsymbol{k}}^s\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{t}}$$

with $\omega_{\boldsymbol{k}}^s = (1 + \|2\pi\boldsymbol{k}\|_2^2)^{-s}$. In the case $s = 2m$ with $m \in \mathbb{N}$ the kernel K_N is related to the periodic version of the cardinal polyharmonic splines introduced by Madych and Nelson[7] as fundamental solutions of the polyharmonic equation (see[8] for details).

In contrast to finite difference method (FDM) we never discretise the differential operator, instead the operator is applied to the basis functions directly, such that we obtain a meshless method. Let u be a function with uniformly convergent Fourier series

$$u(\boldsymbol{v}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^3} \hat{u}_{\boldsymbol{k}}\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{v}}\,.$$

Then it follows that

$$\mathcal{I}(u, K_N, \boldsymbol{v}) = \sum_{\boldsymbol{k}\in I_N^3} \hat{u}_{\boldsymbol{k}}\omega_{\boldsymbol{k}}^s\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{v}}\,. \tag{2.2}$$

Our aim is to approximate the solution of (1.1) by functions of the form (2.2) by iterative solutions of large least squares problems. More precisely we have to determine the coefficients $\hat{u}_{\boldsymbol{k}}$ $(\boldsymbol{k} \in I_N^3)$. In general the functions f and g in (1.1) are not given explicitly. The problem is to recover $\mathcal{I}(u, K_N)$ from M samples $\boldsymbol{v}_j \in \Omega$ and R samples $\boldsymbol{w}_j \in \delta\Omega$. In the following we assume that $\Omega \subset [-1/2, 1/2]^3$, than (1.1) can be written as follows:
Given a set of M arbitrary distinct nodes $\boldsymbol{v}_j$ $(j \in I_M^1)$ with $\boldsymbol{v}_j \in \mathcal{T}^3$, and values $f_j^1 = f(\boldsymbol{v}_j)$ at the nodes $\boldsymbol{v}_j$. We are looking for a numerical method to compute an approximation of (1.1) such that

$$f_j^1 = \Delta\, \mathcal{I}(u, K_N, \boldsymbol{v}_j) \tag{2.3}$$

and furthermore given a set of R arbitrary distinct nodes $\boldsymbol{w}_j$ $(j \in I_R^1)$ with $\boldsymbol{w}_j \in \delta[-1/2, 1/2]^3$, and values $g_j^1 = g(\boldsymbol{w}_j)$ at the nodes $\boldsymbol{w}_j$. By (1.1) we obtain the equations

$$g_j^1 = \mathcal{I}(u, K_N, \boldsymbol{w}_j)\ . \tag{2.4}$$

In matrix vector notation the equations (2.3) read as

$$\boldsymbol{A}\boldsymbol{W}\hat{\boldsymbol{u}}_N = \boldsymbol{f}_M^1,$$

where the matrix $\boldsymbol{A}$ is given by

$$\boldsymbol{A} := \left(\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{v}_j}\right)_{j\in I_M^1, \boldsymbol{k}\in I_N^3}\ .$$

The vector

$$\hat{\boldsymbol{u}}_N := (\hat{u}_{\boldsymbol{k}})_{\boldsymbol{k}\in I_N^3}\ ,$$

denotes the unknown Fourier coefficients, where

$$\boldsymbol{W} :=\ \mathrm{diag}(\tilde{\omega}_{\boldsymbol{k}}^s)_{\boldsymbol{k}\in I_N^3} \quad \text{with} \quad \tilde{\omega}_{\boldsymbol{k}}^s := \frac{-\|2\pi\boldsymbol{k}\|_2^2}{(1 + \|2\pi\boldsymbol{k}\|_2^2)^s}\,.$$

is a diagonal matrix and

$$\boldsymbol{f}_M^1 := (f_j)_{j\in I_M^1}$$

is a vector with the given values in the domain $\Omega \subset \mathcal{T}^3$. From equation (2.4) we obtain

$$\boldsymbol{A}_{\mathrm{B}}\boldsymbol{W}_{\mathrm{B}}\hat{\boldsymbol{u}}_N = \boldsymbol{g}_R^1\,, \tag{2.5}$$

where the matrices $\boldsymbol{A}_{\mathrm{B}}$ and $\boldsymbol{W}_{\mathrm{B}}$ are given by

$$\boldsymbol{A}_{\mathrm{B}} := \left(\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{w}_j}\right)_{j\in I_R^1, \boldsymbol{k}\in I_N^3}$$

and

$$\boldsymbol{W}_{\mathrm{B}} :=\ \mathrm{diag}(\omega_{\boldsymbol{k}}^s)_{\boldsymbol{k}\in I_N^3}\ ,$$

respectively. The vector

$$\boldsymbol{g}_R^1 := (g_j)_{j\in I_R^1}$$

denotes the given values at the boundary. In order to find an approximate solution for (1.1) we have to solve the equation

$$\begin{bmatrix} \boldsymbol{A}\boldsymbol{W} \\ \boldsymbol{A}_{\mathrm{B}}\boldsymbol{W}_{\mathrm{B}} \end{bmatrix} \hat{\boldsymbol{u}}_N = \begin{bmatrix} \boldsymbol{f}_M^1 \\ \boldsymbol{g}_R^1 \end{bmatrix}\ . \tag{2.6}$$

Using an approach with radial basis functions in (2.2) instead of the exponentials, this method is known as Kansa's[9, 10] method.

Only in the case $M + R = N^3$ we obtain a square matrix, otherwise we suggest to solve the following standard least–squares problem: Minimise

$$\left\| \begin{bmatrix} \boldsymbol{A}\boldsymbol{W} \\ \boldsymbol{A}_{\mathrm{B}}\boldsymbol{W}_{\mathrm{B}} \end{bmatrix} \hat{\boldsymbol{u}}_N - \begin{bmatrix} \boldsymbol{f}_M^1 \\ \boldsymbol{g}_R^1 \end{bmatrix} \right\|_2 . \tag{2.7}$$

It is well–known that the normal equation

$$\left[\boldsymbol{W}\boldsymbol{A}^{\mathrm{H}} \,,\, \boldsymbol{W}_{\mathrm{B}}\boldsymbol{A}_{\mathrm{B}}^{\mathrm{H}}\right] \begin{bmatrix} \boldsymbol{A}\boldsymbol{W} \\ \boldsymbol{A}_{\mathrm{B}}\boldsymbol{W}_{\mathrm{B}} \end{bmatrix} \hat{\boldsymbol{u}}_N \;=\; \left[\boldsymbol{W}\boldsymbol{A}^{\mathrm{H}} \,,\, \boldsymbol{W}_{\mathrm{B}}\boldsymbol{A}_{\mathrm{B}}^{\mathrm{H}}\right] \begin{bmatrix} \boldsymbol{f}_M^1 \\ \boldsymbol{g}_R^1 \end{bmatrix} \tag{2.8}$$

is used to solve the least–squares problem (2.7) for over–determined systems, i.e. $M + R > N^3$.

We suggest to solve the equations (2.6) by a CG-type method applied to the normal equation, which can be considered as least–squares projection method. There are many ways, all mathematically equivalent, to implement the CG method for least–squares problems. In exact arithmetic they will all generate the same sequence of approximations, but in finite precision the achieved accuracy may differ substantially. It is important to notice that an implementation of the CG method for symmetric positive definite systems should *not* be applied directly to ill-conditioned normal equations (2.8) (see the discussion in[11]). The system (2.8) and methods derived from it are often labeled with NR (N for "Normal" and R for "Residual"). Thus CGNR denotes the Conjugate Gradient method applied to (2.8) (see,[12] CGNE in[13]). Algorithm 2.3 of[13] reads with respect to our notation as follows.

Algorithm (CGNR)

Input: $N \in \mathbb{N}$,
$M \in \mathbb{N}$, $\boldsymbol{f}_M^1$, $\boldsymbol{W}$,
$R \in \mathbb{N}$, $\boldsymbol{g}_R^1$, $\boldsymbol{W}_{\mathrm{B}}$,
$\hat{\boldsymbol{u}}_N^{(0)}$ % start vector
$\boldsymbol{r}^{(0)} = \boldsymbol{f}_M^1 - \boldsymbol{A}\boldsymbol{W}\hat{\boldsymbol{u}}_N^{(0)}$
$\boldsymbol{r}_{\mathrm{B}}^{(0)} = \boldsymbol{g}_R^1 - \boldsymbol{A}_{\mathrm{B}}\boldsymbol{W}_{\mathrm{B}}\hat{\boldsymbol{u}}_N^{(0)}$
$\boldsymbol{q} = \boldsymbol{W}\boldsymbol{A}^{\mathrm{H}}\boldsymbol{r}^{(0)} + \boldsymbol{W}_{\mathrm{B}}\boldsymbol{A}_{\mathrm{B}}^{\mathrm{H}}\boldsymbol{r}_{\mathrm{B}}^{(0)}$
$k = 0$
while (not stop) do
$\alpha = \|\boldsymbol{W}\boldsymbol{A}^{\mathrm{H}}\boldsymbol{r}^{(k)}\|_2^2 / \| \begin{bmatrix} \boldsymbol{A}\boldsymbol{W} \\ \boldsymbol{A}_{\mathrm{B}}\boldsymbol{W}_{\mathrm{B}} \end{bmatrix} \boldsymbol{q}\|_2^2$
$\hat{\boldsymbol{u}}_N^{(k+1)} = \hat{\boldsymbol{u}}_N^{(k)} + \alpha\boldsymbol{q}$
$\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \alpha\boldsymbol{A}\boldsymbol{W}\boldsymbol{q}$
$\boldsymbol{r}_{\mathrm{B}}^{(k+1)} = \boldsymbol{r}_{\mathrm{B}}^{(k)} - \alpha\boldsymbol{A}_{\mathrm{B}}\boldsymbol{W}_{\mathrm{B}}\boldsymbol{q}$
$\beta = \|\boldsymbol{W}\boldsymbol{A}^{\mathrm{H}}\boldsymbol{r}^{(k+1)} + \boldsymbol{W}_{\mathrm{B}}\boldsymbol{A}_{\mathrm{B}}^{\mathrm{H}}\boldsymbol{r}^{(k+1)}\|_2^2 / \|\boldsymbol{W}\boldsymbol{A}^{\mathrm{H}}\boldsymbol{r}_{\mathrm{B}}^{(k)} + \boldsymbol{W}_{\mathrm{B}}\boldsymbol{A}_{\mathrm{B}}^{\mathrm{H}}\boldsymbol{r}_{\mathrm{B}}^{(k)}\|_2^2$
$\boldsymbol{q} = \boldsymbol{W}\boldsymbol{A}^{\mathrm{H}}\boldsymbol{r}^{(k+1)} + \boldsymbol{W}_{\mathrm{B}}\boldsymbol{A}_{\mathrm{B}}^{\mathrm{H}}\boldsymbol{r}_{\mathrm{B}}^{(k+1)} + \beta\boldsymbol{q}$
$k = k + 1$
end while
Output: Compute the CGNR solution $\hat{\boldsymbol{u}}_N = \hat{\boldsymbol{u}}_N^{(k)}$.

From the values $\hat{\boldsymbol{u}}_N^{(k)}$ we compute the Fourier coefficients of $\mathcal{I}(u, K_N, v)$ by $\boldsymbol{W}\hat{\boldsymbol{u}}_N^{(k)}$. Finally we are able to compute values on a regular grid by a d–variate FFT.

Straightforward matrix vector multiplications with $\boldsymbol{A}$, $\boldsymbol{A}^{\mathrm{H}}$ and with $\boldsymbol{A}_{\mathrm{B}}$, $\boldsymbol{A}_{\mathrm{B}}^{\mathrm{H}}$ require $\mathcal{O}(N^3 M)$ and $\mathcal{O}(N^3 R)$ arithmetic operations, respectively, too many for our application. To speed up the matrix vector multiplication with a general matrix $\boldsymbol{A}$ and $\boldsymbol{A}_{\mathrm{B}}$ we use an approximative algorithm, which is known as NFFT. Fast and robust computation of the discrete Fourier transforms for nonequispaced data

$$f(\boldsymbol{v}_j) = \sum_{\boldsymbol{k} \in I_N^3} \hat{f}_{\boldsymbol{k}}\, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{v}_j} \qquad (j \in I_M^1) \tag{2.9}$$

and

$$h(\boldsymbol{k}) := \sum_{j\in I_M^1} \hat{f}_j \,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{v}_j} \qquad (\boldsymbol{k}\in I_N^3)\,, \tag{2.10}$$

are possible with the NFFT. Details concerning NFFT algorithms can be found for example in[1] and a software package can be found in.[3] In summary the NFFT, i.e. the fast computation of (2.9) or (2.10) requires only $\mathcal{O}(N^3\log N + (2m+1)^3 M)$ arithmetic operations where m is a constant and depend only on the accuracy of the computation.

2.2. Multigrid Method for a Finite Difference Discretisation

The finite difference method (FDM) discretises the Laplacian by second-order central differences. To give a brief explanation for nonequidistant meshes we consider equation (2.1) on a d-dimensional box $\Omega = [a_{x_1}, b_{x_1}] \times [a_{x_2}, b_{x_2}] \times \cdots \times [a_{x_d}, b_{x_d}]$. Every coordinate direction is discretised by N_{x_j} subintervals $h_{x_j,0}$, $h_{x_j,1}$, …, $h_{x_j,N_{x_j}-1}$ with $b_{x_j} - a_{x_j} = \sum_{i=0}^{N_{x_j}-1} h_{x_j,i}$ for $j = 1, \ldots, d$. Further we introduce

$$\tilde{h}_{x_j,i} = \begin{cases} \dfrac{h_{x_j,i-1}+h_{x_j,i}}{2} & i = 1,\ldots,N_{x_j}-1 \\ 0 & i = N_{x_j} \end{cases}$$

for $j = 1, \ldots, d$ which is known as mesh spacing on the dual grid. In the case $d = 3$ we simplify $(x_1, x_2, x_3) = (x, y, z)$. Thus, this discretisation leads to the following system of equations

$$\begin{aligned} & \tilde{h}_{y,j}\tilde{h}_{z,k}\left(-\tfrac{1}{h_{x,i-1}}u_{i-1,j,k} + \left(\tfrac{1}{h_{x,i-1}}+\tfrac{1}{h_{x,i}}\right)u_{i,j,k} - \tfrac{1}{h_{x,i}}u_{i+1,j,k}\right) \\ + \;& \tilde{h}_{x,i}\tilde{h}_{z,k}\left(-\tfrac{1}{h_{y,j-1}}u_{i,j-1,k} + \left(\tfrac{1}{h_{y,j-1}}+\tfrac{1}{h_{y,j}}\right)u_{i,j,k} - \tfrac{1}{h_{y,i}}u_{i,j+1,k}\right) \\ + \;& \tilde{h}_{x,i}\tilde{h}_{y,j}\left(-\tfrac{1}{h_{z,k-1}}u_{i,j,k-1} + \left(\tfrac{1}{h_{z,k-1}}+\tfrac{1}{h_{z,k}}\right)u_{i,j,k} - \tfrac{1}{h_{z,k}}u_{i,j,k+1}\right) \\ = \;& \tilde{h}_{x,i}\tilde{h}_{y,j}\tilde{h}_{z,k} f_{i,j,k} \end{aligned}$$

for $i = 1, \ldots, N_x - 1$, $j = 1, \ldots, N_y - 1$, $k = 1, \ldots, N_z - 1$. The same system of equations is obtained in the field of computational electromagnetics with the application of the Finite Integration Technique (FIT) which has been introduced by Weiland.[14]

In matrix vector notation the equations read as

$$(\tilde{\boldsymbol{H}}_z \otimes \tilde{\boldsymbol{H}}_y \otimes \boldsymbol{A}_x + \tilde{\boldsymbol{H}}_z \otimes \boldsymbol{A}_y \otimes \tilde{\boldsymbol{H}}_x + \boldsymbol{A}_z \otimes \tilde{\boldsymbol{H}}_y \otimes \tilde{\boldsymbol{H}}_x)\boldsymbol{u} = \tilde{\boldsymbol{H}}_z \otimes \tilde{\boldsymbol{H}}_y \otimes \tilde{\boldsymbol{H}}_x \boldsymbol{f}$$

with the settings

$$\tilde{\boldsymbol{H}}_x := \operatorname{diag}(\tilde{h}_{x,1}, \tilde{h}_{x,2} \ldots, \tilde{h}_{x,N_x-1}),$$

$$\boldsymbol{A}_x := \begin{pmatrix} \left(\frac{1}{h_{x,0}}+\frac{1}{h_{x,1}}\right) & -\frac{1}{h_{x,1}} & & \\ -\frac{1}{h_{x,1}} & \left(\frac{1}{h_{x,1}}+\frac{1}{h_{x,2}}\right) & -\frac{1}{h_{x,2}} & \\ & & \ddots & \\ & & -\frac{1}{h_{x,N_x-2}} & \left(\frac{1}{h_{x,N_x-2}}+\frac{1}{h_{x,N_x-1}}\right) \end{pmatrix} \in \mathbb{R}^{N_x-1\times N_x-1}.$$

The diagonal matrices $\tilde{\boldsymbol{H}}_y$ and $\tilde{\boldsymbol{H}}_z$ are defined analogously to $\tilde{\boldsymbol{H}}_x$ and the finite difference matrices $\boldsymbol{A}_y$ and $\boldsymbol{A}_z$ analogously to $\boldsymbol{A}_x$. Note the different dimensions of the matrices corresponding to the number of mesh lines in every coordinate direction. The vectors $\boldsymbol{f} = (f_{i,j,k})_{i=1,j=1,k=1}^{N_x-1,N_y-1,N_z-1}$ and $\boldsymbol{u} = (u_{i,j,k})_{i=1,j=1,k=1}^{N_x-1,N_y-1,N_z-1}$ contain the values of the right hand side and the potential at the mesh points, respectively.

State-of-the-art is the application of a multigrid method as Poisson solver. In model cases the numerical effort scales with the number of mesh points. Here, we give only the general idea of a geometrical multigrid algorithm. Details can be found in.[6,15] The multigrid algorithm operates on a certain number of grids starting with the mesh given by the discretisation of Poisson's equation. This mesh is referred to as the fine grid or the fine level. Then a sequence of coarser grids is generated by removing mesh lines. On an equidistant mesh every second mesh line is removed. Now iteratively, a raw approximation of the solution of the systems of equations is obtained by the application of a few steps of a relaxation scheme (e. g. Gauss–Seidel iteration) which is called pre-smoothing. This approximation is then improved by a correction vector obtained on the coarser grids (the so-called coarse grid correction) where restriction and interpolation work as grid transfer operators. After applying interpolation another few steps of relaxation are necessary (post-smoothing). For the space charge calculations a multigrid V-cycle is realised. This scheme goes strictly down from the fine to the coarsest grid and then up again to the fine level.

As shown in[16, 17] the coarsening strategy is crucial for the convergence of the multigrid algorithm on nonequidistant grids. The generation of coarse grids with every second grid line removed as suggested in[15] is not reasonable with the discretisations for bunches. It would lead to coarser grids with increasing aspect ratio of the mesh spacing. The convergence of a multigrid scheme on such grids would considerably slow down. Here, the removal of mesh lines follows the rule: Two neighbouring steps h_1 and h_2 remain also in the next coarser grid as long as either $h_1 \geq sh_{\text{min}}$ or $h_2 \geq sh_{\text{min}}$, where h_{min} denotes the overall minimal step size of the corresponding fine level. With the objective to obtain a decreasing aspect ratio of the mesh spacing a choice of $s = 1.6$ or $s = 1.7$ provides the best results. The application of this scheme to space-charge calculations of charged particles sometimes requires a more rigorous coarsening with $1.5 \leq s \leq 2.0$.[17]

The numerical tests have been performed with the following multigrid components: two pre- and two post-smoothing steps with Gauss–Seidel relaxation, full weighting restriction and trilinear interpolation as grid transfer operators.

3. NUMERICAL EXAMPLES

Our algorithms were implemented in C using double precision arithmetic. All numerical experiments were run under Linux 2.4.20 on an AMD Atlon XP 2700+ with 1 GB of RAM. The implementation of the CGNR uses the NFFT library Version 1.0, where we apply the NFFT/NFFT$^{\mathrm{T}}$ package[3] with Kaiser-Bessel functions, truncation parameter $m = 6$ and oversampling factor $\sigma = 2$.

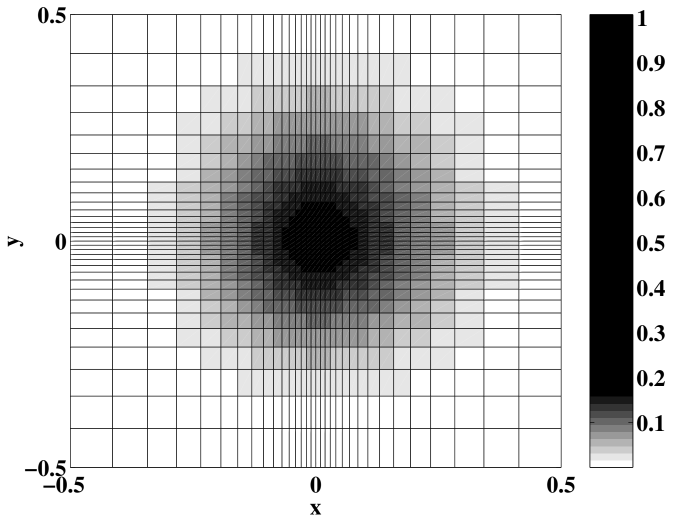

Figure 1. Potential (3.2) given on the nonequidistant grid (3.1): (x, z)-plane with $y = 0$.

Example: Let the mesh points on a nonequidistant grid given by $\boldsymbol{v}_j = (v^1_{k_1}, v^2_{k_2}, v^3_{k_3})^{\mathrm{T}}$ with

$$v^l_{k_l} = \frac{\sinh\left(\frac{6k_l}{M_l}\right)}{2\sinh(3)} \qquad (k_l = -M_l/2, \ldots, M_l/2 - 1;\ l = 1, 2, 3). \tag{3.1}$$

and $j = 0, \ldots, M - 1$, where $M = M_1 M_2 M_3$. Such meshes are typical for the calculation of space-charge effects of charged particle bunches: we have a fine discretisation according to the distribution of the particles in the centre of a box around the bunch. Further the discretisation becomes coarser outside the bunch.[5, 18]

Typically particles are assumed to have a Gaussian distribution in the begin of an accelerating process. Thus we choose the potential for the numerical test as

$$u(v_1, v_2, v_3) = \mathrm{e}^{-(16v_1^2+18v_2^2+20v_3^2)}$$

such that

$$-\Delta u(v_1, v_2, v_3) = -(1024v_1^2 + 1296v_2^2 + 1600v_3^2 - 108)\,\mathrm{e}^{-(16v_1^2+18v_2^2+20v_3^2)} \text{ in } \Omega = [-1/2, 1/2]^3. \qquad (3.2)$$

Figure 1 shows this potential in the (x, z)-plane of the upper constructed nonequidistant grid with $y = 0$.

We measured the error

$$E := \max_{j=1,\ldots,M} \frac{|f(\boldsymbol{v}_j) - \tilde{f}(\boldsymbol{v}_j)|}{|f(\boldsymbol{v}_j)|}, \qquad (3.3)$$

where $\tilde{f}$ is the approximate solution of the multigrid scheme and Fourier method, respectively. Table 1 compares the CPU

	multigrid method		Fourier method	
M	time in sec.	E	time in sec.	E
16^3	0.04	3.33e-02	0.68	2.95e-01
32^3	0.17	8.60e-03	5.76	5.19e-02
64^3	1.43	1.05e-02	41.44	3.34e-02
128^3	12.1	1.07e-02	217.8	4.85e-02

Table 1. Approximation error and computational time.

time of the multigrid method and the Fourier method. We applied the algorithms such that the error $E < 0.05$. Column 2 and 4 contain the computational time in seconds for the multigrid method and the Fourier method, respectively. For these tests we chose $M_1 = M_2 = M_3$. The Fourier method was performed with $N = M_1/2$. Note that each step of the CGNR method requires the computation of 4 NFFTs because we require in addition that the solution is zero at the boundary (see (2.4)). In our test case we set $R = 3 \cdot N \cdot N$ and stop the iteration after four steps. The multigrid method achieves the error E after two iteration steps. Note that the Fourier method is much slower, however we are not longer restricted to special grids.

REFERENCES

1. D. Potts, G. Steidl, and M. Tasche, "Fast Fourier transforms for nonequispaced data: A tutorial," in *Modern Sampling Theory: Mathematics and Applications*, J. J. Benedetto and P. J. S. G. Ferreira, eds., pp. 247 – 270, Birkhäuser, (Boston), 2001.
2. G. Pöplau, "Fast direct solvers for PDE's in shift-inariant periodic spaces," in *Approximation Theory VIII*, C. Chui and L. Schumaker, eds., pp. 325 – 333, World Scientific Publishing, Inc., (Singapore), 1995.
3. S. Kunis and D. Potts, "NFFT, Softwarepackage, C subroutine library." http://www.math.uni-luebeck.de/potts/nfft, 2002.
4. G. Pöplau and U. van Rienen, "Multigrid Algorithms for the Tracking of Electron Beams", in *Multigrid Methods* VI, (E. Dick, K. Riemslagh, J. Vierendeels, eds.), LNSCE, **14**, Springer, 2000, 214 – 220.
5. G. Pöplau, U. van Rienen, M.J. de Loos, and S.B. van der Geer, "Fast Calculation of Space Charge in Beam Line Tracking by Multigrid Techniques". in: *Mathematics in Industry (Proc. of the 4th Conf. on Scientific Computing in Electrical Engineering (SCEE–2002)),* Springer, Berlin, to appear.
6. W. Hackbusch, *Multi-Grid Methods and Applications.* Springer, Berlin, 1985.
7. W. R. Madych and S. A. Nelson, "Polyharmonic cardinal splines," *J. Approx. Theory* **60**, pp. 141 – 161, 1990.
8. G. Pöplau, "Fast solvers for elliptic PDE's with radial basis functions," *University Rostock, Preprint 96/10* , 1996.
9. E. J. Kansa, "Multiquadrics- a scattered data approximation scheme with applications to computational fluid dynamics: I. surface approximations and partial derivative estimates," *Comput. Math. Appl.* **19**, pp. 127 – 145, 1990.

10. E. J. Kansa, “Multiquadrics - a scattered data approximation scheme with applications to computational fluid dynamics: II. solutions to parabolic, hyperbolic, and elliptic partial differential equations,” *Comput. Math. Appl.* **19**, pp. 147 – 161, 1990.
11. C. C. Paige and M. A. Saunders, “LSQR: An algorithm for sparse linear equations and sparse least squares,” *ACM Transactions on Mathematical Software* **8**, pp. 43 – 71, 1982.
12. Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publ., Boston, 1996.
13. M. Hanke, *Conjugate gradient type method for ill–posed problems*, Wiley, New York, 1995.
14. T. Weiland, “Eine Methode zur Lösung der Maxwellschen Gleichungen für sechskomponentige Felder auf disketer Basis”, *AEÜ* **31**, pp. 116 – 120, 1977.
15. W.L. Briggs, Van Emden Henson, and S. McCormick, *A Multigrid Tutorial. 2nd edition.* SIAM, Philadelphia, 2000.
16. G. Pöplau and U. van Rienen, “Multigrid Solvers for Poisson’s Equation in Computational Electromagnetics”, in: Scientific Computing in Electrical Engineering (U. van Rienen, M. Günther, D. Hecht, eds.), LNCSE **18**, Springer–Verlag, Berlin, 169 – 176, 2001.
17. G. Pöplau, U. van Rienen, M.J. de Loos, and S.B. van der Geer, “Multigrid Algorithms for the Fast Calculation of Space-Charge Effects in Accelerator Design”. submitted, 2003.
18. G. Pöplau, U. van Rienen, M.J. de Loos, and S.B. van der Geer, *“A Fast 3D Multigrid Based Space-Charge Routine in the GPT Code”*, Proceedings of EPAC 2002 (Paris), pp. 1658 – 1668, 2002.
19. G. Pöplau, U. van Rienen, M.J. de Loos, and S.B. van der Geer, “A multigrid based 3D space-charge routine in the tracking code GPT”. TESLA-Report 2003-03, DESY, Hamburg (`tesla.desy.de/new_pages/TESLA/TTFnot03.html`), 2003.